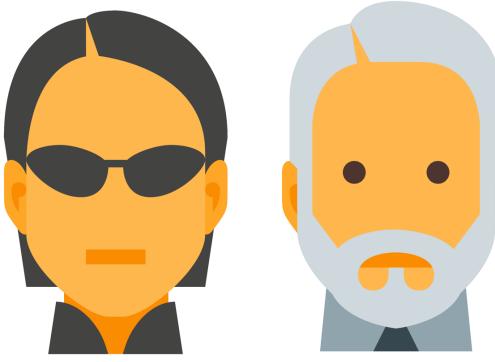


155 PATTERNS



Wilmer Krisp

278 SOFTWARE ARCHITECTURE
DESIGN MODELS

COMPLETE CATALOG OF ALL CLASSICAL
PATTERNS IN THE ARCHIMATE LANGUAGE

01

Design Patterns

Elements of Reusable Object-Oriented Software

GANG OF FOUR

02

Enterprise patterns

Catalog of Patterns of Enterprise Application Architecture

MARTIN FOWLER

03

Analysis Patterns

Reusable Object Models

MARTIN FOWLER

04

Domain Driven Design

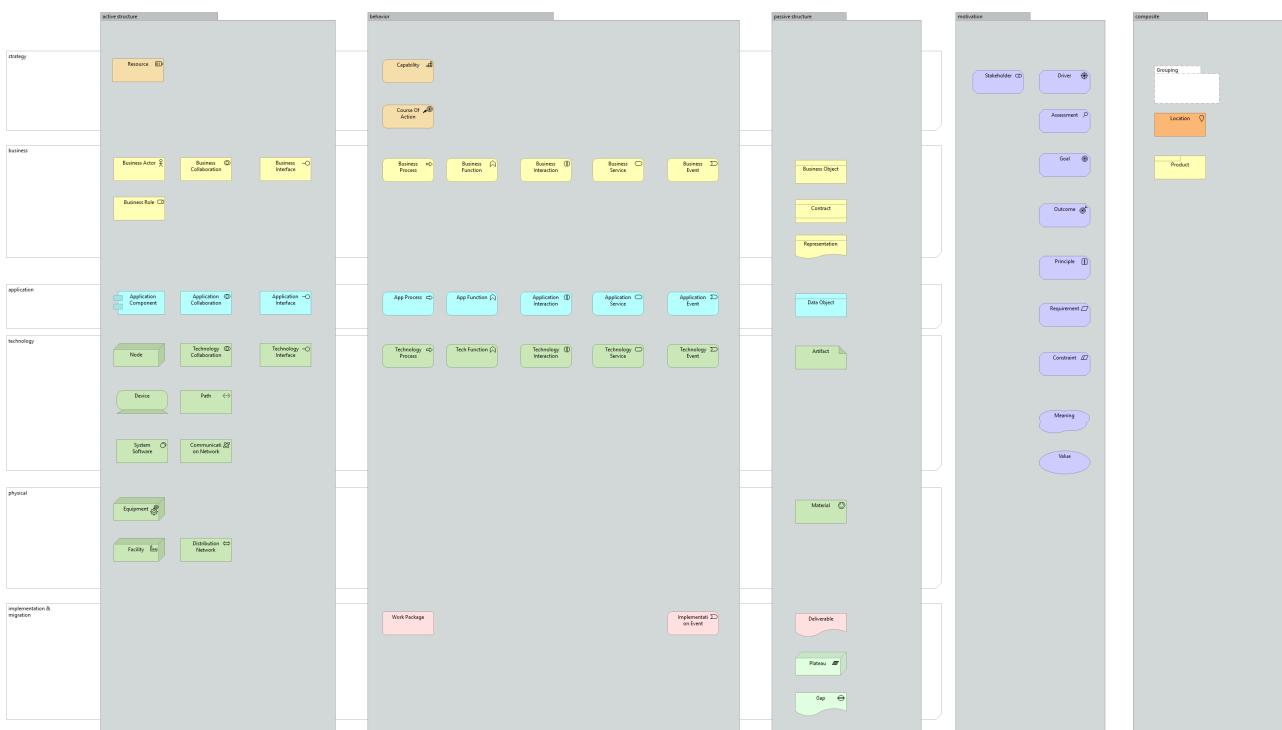
Tackling Complexity in the Heart of Software.

ERIC EVANS

USED NOTATION

ARCHIMATE METAMODEL

The Open Group



DESIGN PATTERNS 10

CREATIONAL PATTERNS	11
ABSTRACT FACTORY	12
BUILDER	15
FACTORY METHOD	17
PROTOTYPE	19
SINGLETON	21
STRUCTURAL PATTERNS	23
ADAPTER OF CLASS	24
ADAPTER OF OBJECT	26
BRIDGE	29
COMPOSITE	31
DECORATOR	33
FAÇADE	35
FLYWEIGHT	38
FLYWEIGHT + COMPOSITE	40
PROXY	41
BEHAVIORAL PATTERNS	43
CHAIN OF RESPONSIBILITY	44
COMMAND	46
INTERPRETER	48
ITERATOR	50
MEDIATOR	52
MEMENTO	54
OBSERVER	56
STATE	58
STRATEGY	60
TEMPLATE METHOD	62
VISITOR	65

ENTERPRISE PATTERNS 67

BUSINESS LOGIC	68
DOMAIN MODEL	69
SERVICE LAYER	70
TRANSACTION SCRIPT	71
TABLE MODULE	72
DATA SOURCES	73
ACTIVE RECORD	74
DATA MAPPER	75
ROW DATA GATEWAY	76
TABLE DATA GATEWAY	77
MODELING BEHAVIOR	78
IDENTITY MAP	79
LAZY LOAD	80
UNIT OF WORK.	81
MODELING STRUCTURE HIERARCHY	82
CLASS TABLE INHERITANCE	83
CONCRETE TABLE INHERITANCE	84
INHERITANCE MAPPERS	85

SINGLE TABLE INHERITANCE	86
MODELING STRUCTURE RELATIONS	87
ASSOCIATION TABLE MAPPING	88
DEPENDENT MAPPING	89
EMBEDDED VALUE	90
FOREIGN KEY MAPPING	91
IDENTITY FIELD	92
SERIALIZED LOB	93
METADATA	94
METADATA MAPPING	95
QUERY OBJECT	96
REPOSITORY	97
WEB REPRESENTATION CONTROLLER	98
MODEL VIEW CONTROLLER	99
APPLICATION CONTROLLER	100
FRONT CONTROLLER	101
PAGE CONTROLLER	102
WEB REPRESENTATION VIEW	103
TEMPLATE VIEW	104
TRANSFORM VIEW	105
TWO STEP VIEW	106
DISTRIBUTED PROCESSING	107
DATA TRANSFER OBJECT	108
REMOTE FAÇADE	109
PARALLEL PROCESSING	110
COARSE-GRAINED LOCK	111
IMPLICIT LOCK	112
OPTIMISTIC OFFLINE LOCK	113
PESSIMISTIC OFFLINE LOCK	114
SESSION STATE	115
CLIENT SESSION STATE	116
DATABASE SESSION STATE	117
SERVER SESSION STATE	118
COMMON PATTERNS	119
GATEWAY	120
LAYER SUPERTYPE	121
MAPPER	122
MONEY	123
PLUGIN	124
RECORD SET	125
REGISTRY	126
SEPARATED INTERFACE	127
SERVICE STUB	128
SPECIAL CASE	129
VALUE OBJECT	130

<u>ANALYSIS PATTERNS</u>	131
---------------------------------	------------

ACCOUNTABILITY	133
PARTY	134
ACCOUNTABILITY	135
ORGANIZATION HIERARCHIES	137
ORGANIZATION STRUCTURE	138

ACCOUNTABILITY KNOWLEDGE LEVEL	139
PARTY TYPE GENERALIZATIONS	140
HIERARCHIC ACCOUNTABILITY	141
OPERATING SCOPES	142
POST	143
OBSERVATIONS AND MEASUREMENTS	144
QUANTITY	145
CONVERSION RATIO	146
OBSERVATIONS AND MEASUREMENTS	147
COMPOUND UNITS	148
MEASUREMENT	149
OBSERVATION	150
SUBTYPING OBSERVATION CONCEPTS	151
PROTOCOL	152
DUAL TIME RECORD	153
REJECTED OBSERVATION	154
ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION	155
ASSOCIATED OBSERVATION	156
PROCESS OF OBSERVATION	157
OBSERVATIONS FOR CORPORATE FINANCE	158
ENTERPRISE SEGMENT	159
MEASUREMENT PROTOCOL	160
RANGE	161
OBSERVATIONS FOR CORPORATE FINANCE	162
PHENOMENON WITH RANGE	165
REFERRING TO OBJECTS	166
NAME	167
IDENTIFICATION SCHEME	168
OBJECT MERGE	169
OBJECT EQUIVALENCE	170
REFERRING TO OBJECTS	171
INVENTORY AND ACCOUNTING	172
ACCOUNT	173
TRANSACTIONS	174
SUMMARY ACCOUNT	175
MEMO ACCOUNT	176
POSTING RULES	177
INVENTORY AND ACCOUNTING	178
INDIVIDUAL INSTANCE METHOD	179
POSTING RULE EXECUTION	180
POSTING RULES FOR MANY ACCOUNTS	181
CHOOSING ENTRIES	182
ACCOUNTING PRACTICE	183
SOURCES OF AN ENTRY	184
BALANCE SHEET AND INCOME STATEMENT	185
CORRESPONDING ACCOUNT	186
SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)	187
SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)	188
BOOKING ENTRIES TO MULTIPLE ACCOUNTS	189
PLANNING	190
PROPOSED AND IMPLEMENTED ACTION	191
COMPLETED AND ABANDONED ACTIONS	192
SUSPENSION	193
PLAN	194

PROTOCOL	195
RESOURCE ALLOCATION	196
PLANNING	197
PLANNING (NO OUTCOME)	198
OUTCOME AND START FUNCTIONS	199
TRADING	200
CONTRACT	201
PORTFOLIO	202
QUOTE	203
SCENARIO	204
TRADING	205
DERIVATIVE CONTRACTS	206
FORWARD CONTRACTS	207
OPTIONS	208
PRODUCT	209
SUBTYPE STATE MACHINES	210
PARALLEL APPLICATION AND DOMAIN HIERARCHIES	211
DERIVATIVE CONTRACTS	212
TRADING PACKAGES	213
MULTIPLE ACCESS LEVELS TO A PACKAGE	214
MUTUAL VISIBILITY	215
TRADING PACKAGES	216
LAYERED ARCHITECTURE FOR INFORMATION SYSTEMS	217
TWO-TIER ARCHITECTURE	218
THREE-TIER ARCHITECTURE	219
PRESENTATION AND APPLICATION LOGIC	220
DATABASE INTERACTION	221
TYPE MODEL DESIGN	222
IMPLEMENTING ASSOCIATIONS	223
IMPLEMENTING GENERALIZATION	224
OBJECT CREATION	225
OBJECT DESTRUCTION	226
ENTRY POINT.	227
IMPLEMENTING CONSTRAINTS	228

DOMAIN DRIVEN DESIGN 229

MODEL AND STRUCTURAL ELEMENTS	230
MODEL-DRIVEN DESIGN	231
LAYERED ARCHITECTURE (ASYMMETRIC)	234
HEXAGONAL ARCHITECTURE (SYMMETRIC)	236
COMPOSITE UI	240
ENTITIES	241
VALUE-OBJECTS	243
DOMAIN SERVICES	245
MODULES	250
AGGREGATES	251
AGGREGATE ROOT	253
BEHAVIOR-FOCUSED AGGREGATE ROOT	254
MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION	255
PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES	256
REFERENCE OTHER AGGREGATES BY IDENTITY ONLY	257
FACTORIES	258

REPOSITORIES	260
SUPPLE DESIGN	264
UBIQUITOUS LANGUAGE	265
INTENTION-REVEALING INTERFACES	266
SIDE-EFFECT FREE FUNCTIONS	267
ASSERTIONS	268
CONCEPTUAL CONTOURS	269
STANDALONE CLASSES	270
CLOSURE OF OPERATIONS	271
MODEL INTEGRITY AND CONTEXT	272
BOUNDED CONTEXT	273
CONTINUOUS INTEGRATION	274
STRATEGIC CONTEXT MAP	275
CONTEXTUAL MAP	276
SHARED KERNEL	277
CUSTOMER-SUPPLIER TEAMS	278
CONFORMIST	279
ANTICORRUPTION LAYER	280
SEPARATE WAYS	281
OPEN HOST SERVICE	282
PUBLISHED LANGUAGE	283
DISTILLATION	284
CORE DOMAIN	285
GENERIC SUBDOMAINS	286
DOMAIN VISION STATEMENT	287
HIGHLIGHTED CORE	288
COHESIVE MECHANISMS	289
SEGREGATED CORE	290
ABSTRACT CORE	291
LARGE-SCALE STRUCTURE	292
EVOLVING ORDER	293
SYSTEM METAPHOR	294
RESPONSIBILITY LAYERS	295
KNOWLEDGE LEVEL	299
PLUGGABLE COMPONENT FRAMEWORK	300
ADDITIONAL PATTERNS	301
TYPES OF CONSISTENCY	302
EVENT SOURCING	303
EVENT PROCESSOR	304
EVENT DISPATCHER	305
INTERNAL DOMAIN EVENTS	306
EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS	307
STATIC DOMAIN EVENTS CLASS	308
ONE SUBDOMAIN PER BOUNDED CONTEXT	309
THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS	310
THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS	311
INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE	312
INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES	313
INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS	314
INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE	315
DEPENDENCY INJECTION	316
DEPENDENCY INVERSION	318
INVERSION OF CONTROL	319
SERVICE LOCATOR	320

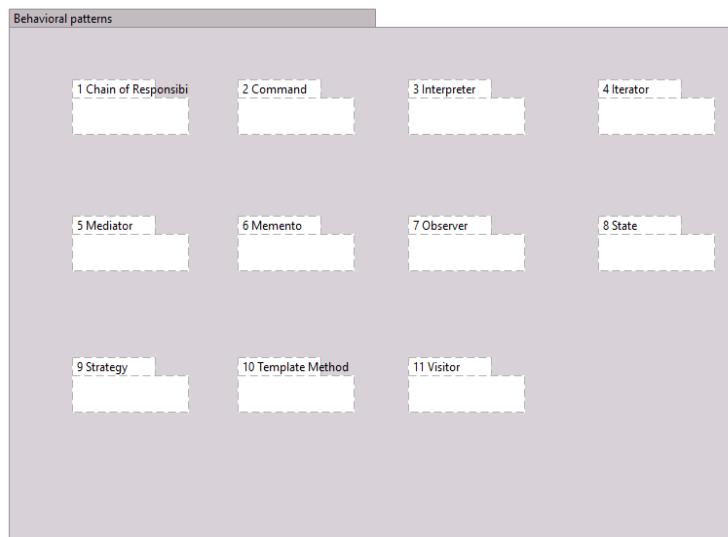
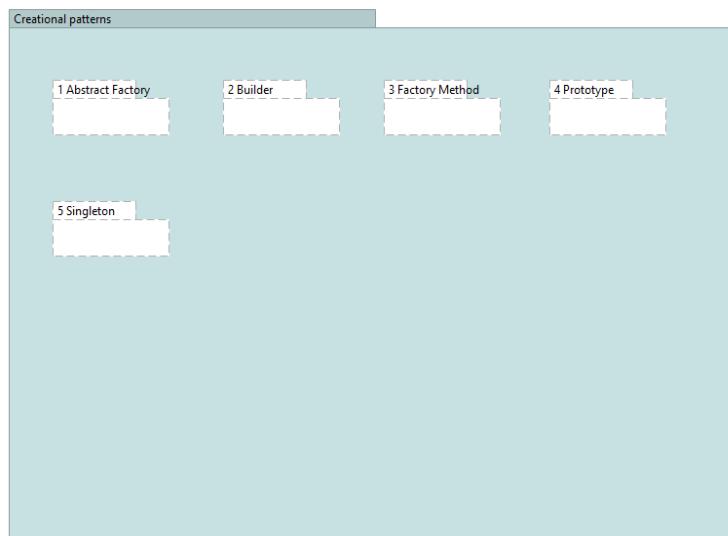
CQRS	321
CQS	322
WRAP LOW-LEVEL EXCEPTIONS	323
EXTRACT DEPENDENCY FROM INTERFACE TO COSNTRUCTOR	324
INTERFACE SEGREGATION	325
CLEAN ARCHITECTURE	326

01

Design Patterns

Elements of Reusable Object-Oriented Software

GANG OF FOUR



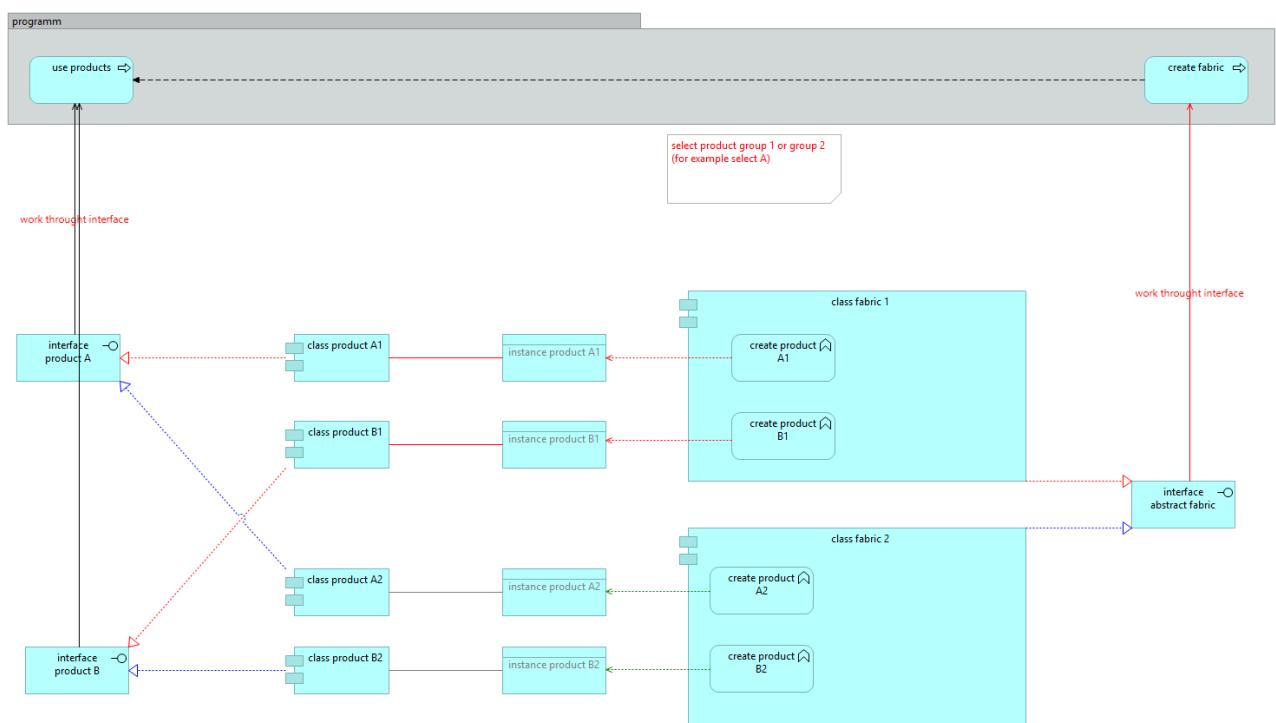
CREATIONAL PATTERNS

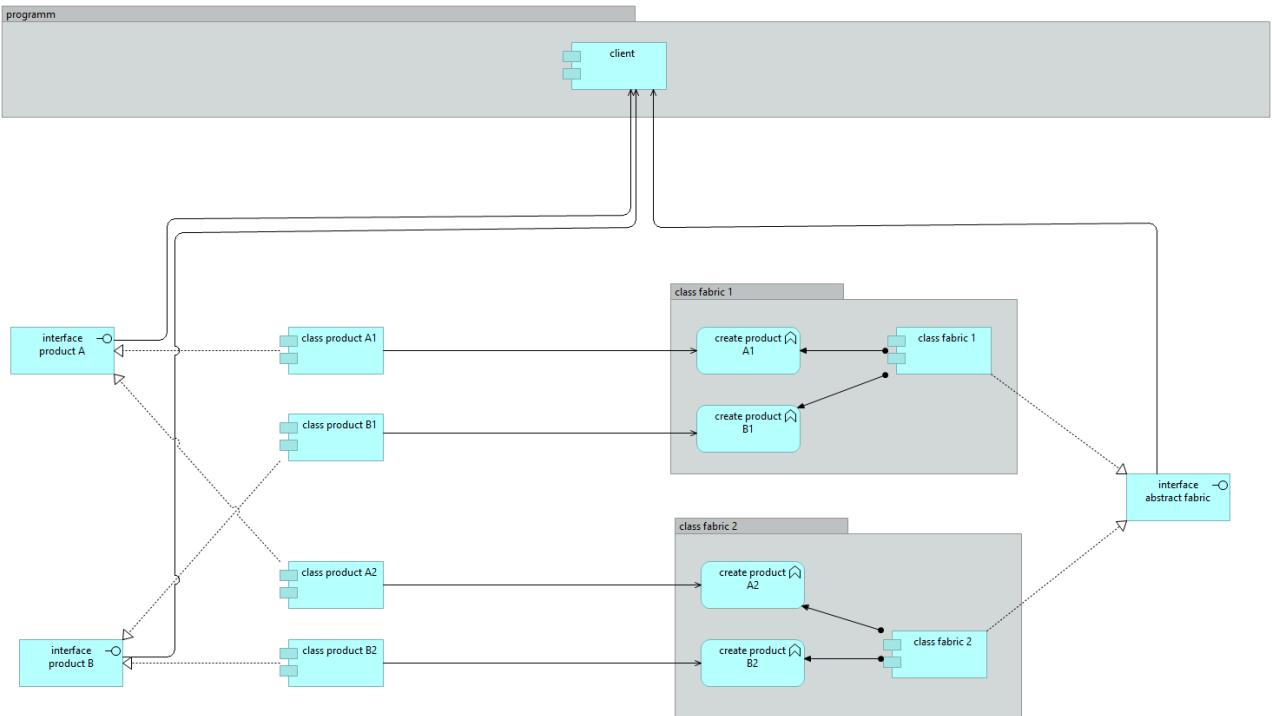
DESIGN PATTERNS

GoF

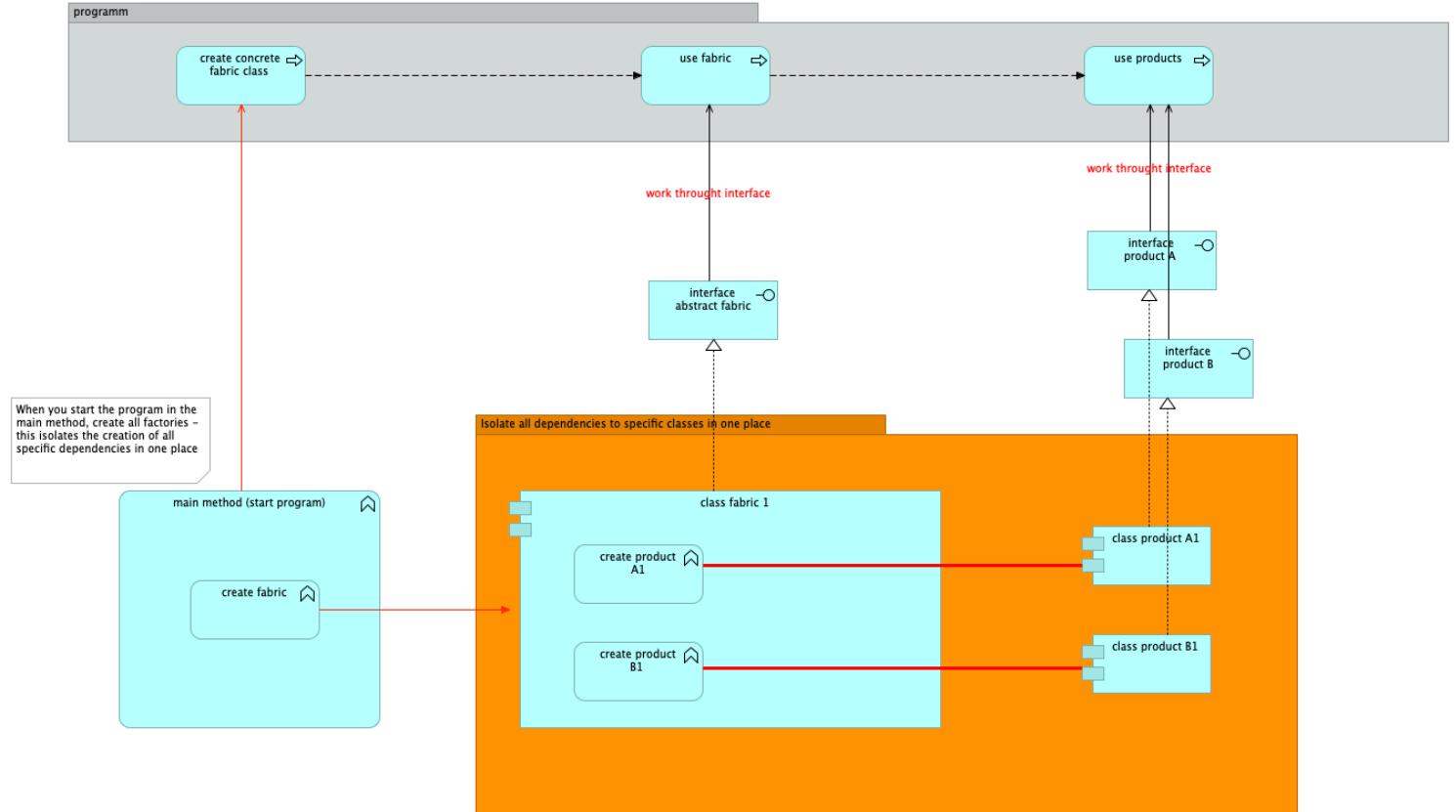


ABSTRACT FACTORY

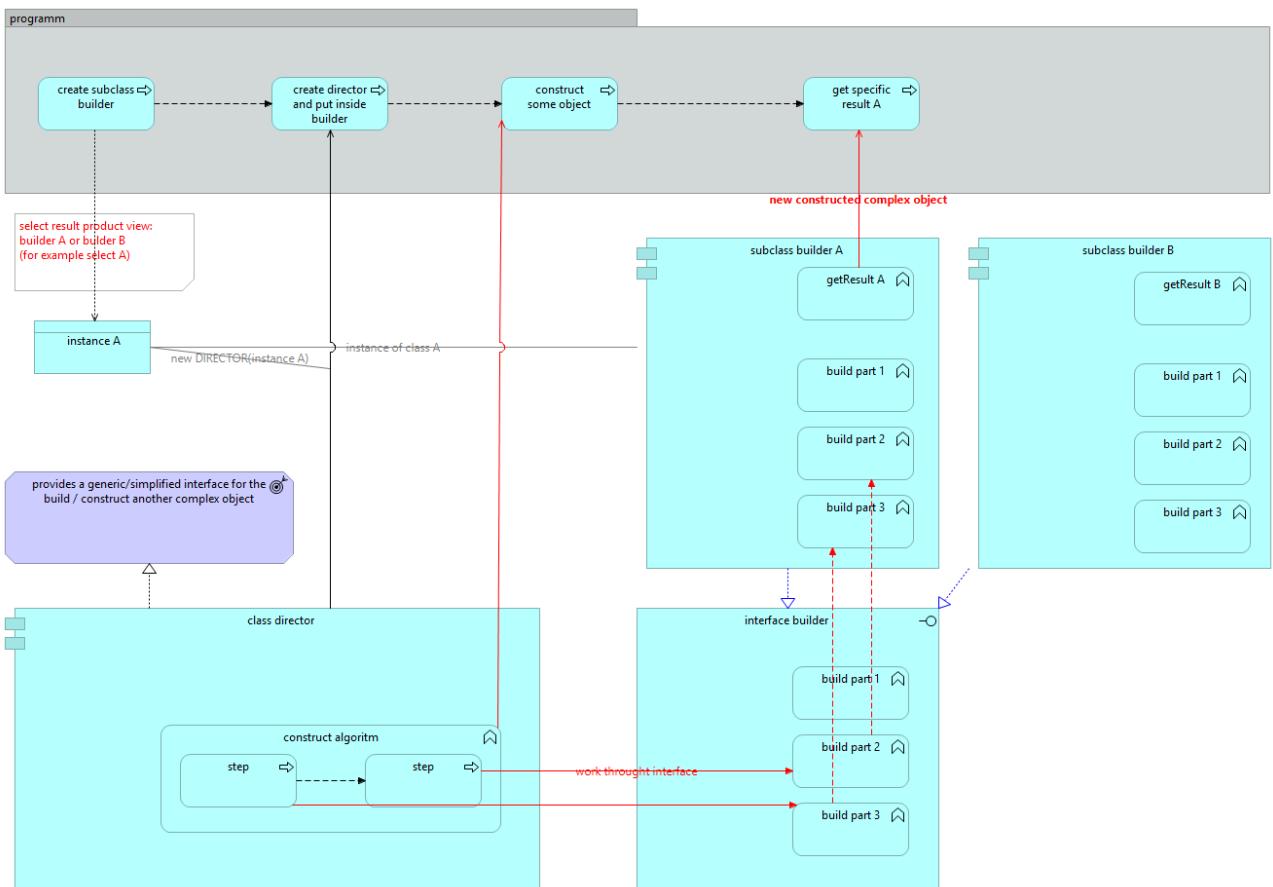


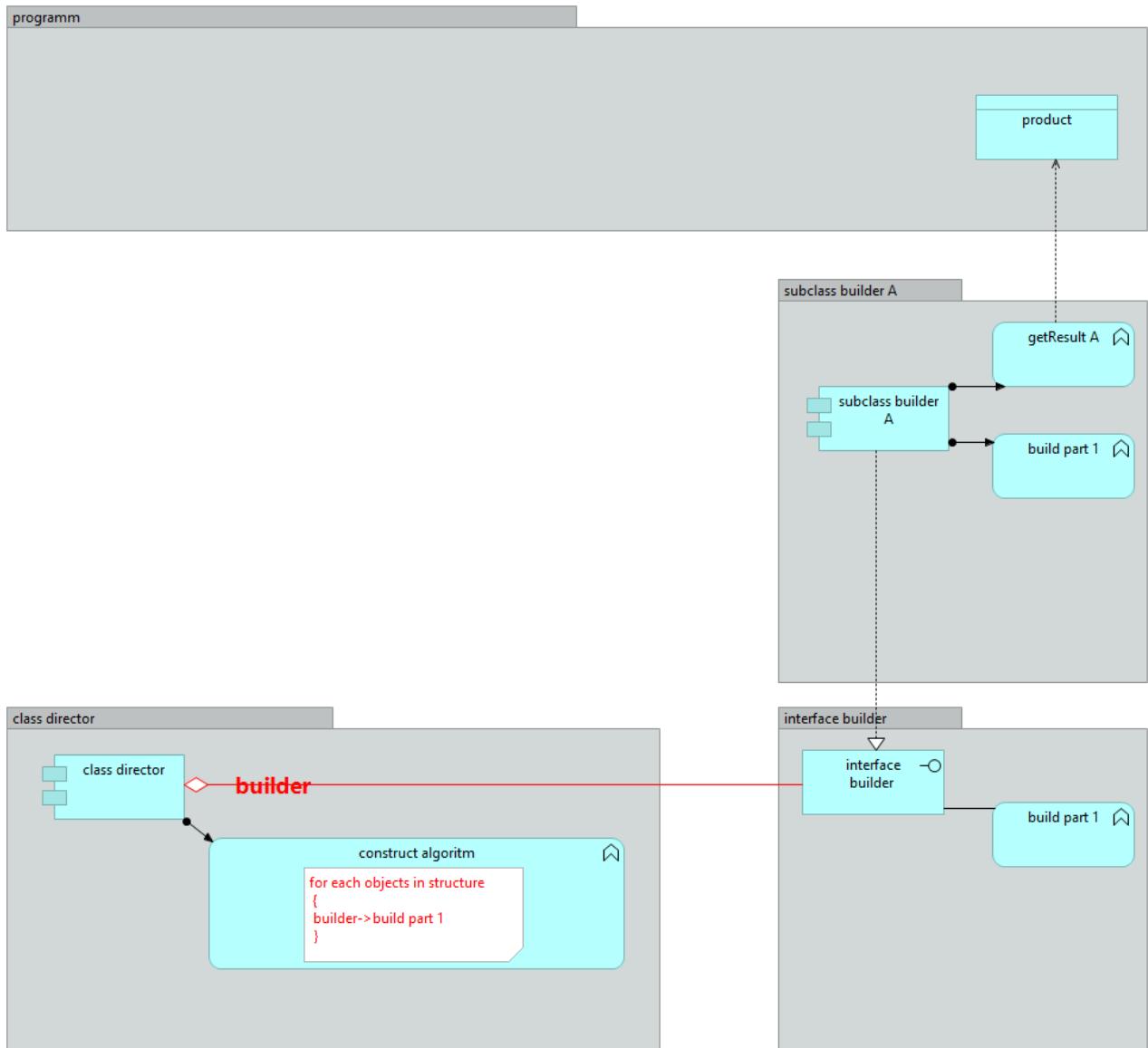


In the whole program (except for the factory inside and one factory creation) we work only with interfaces (and do not refer to specific classes)

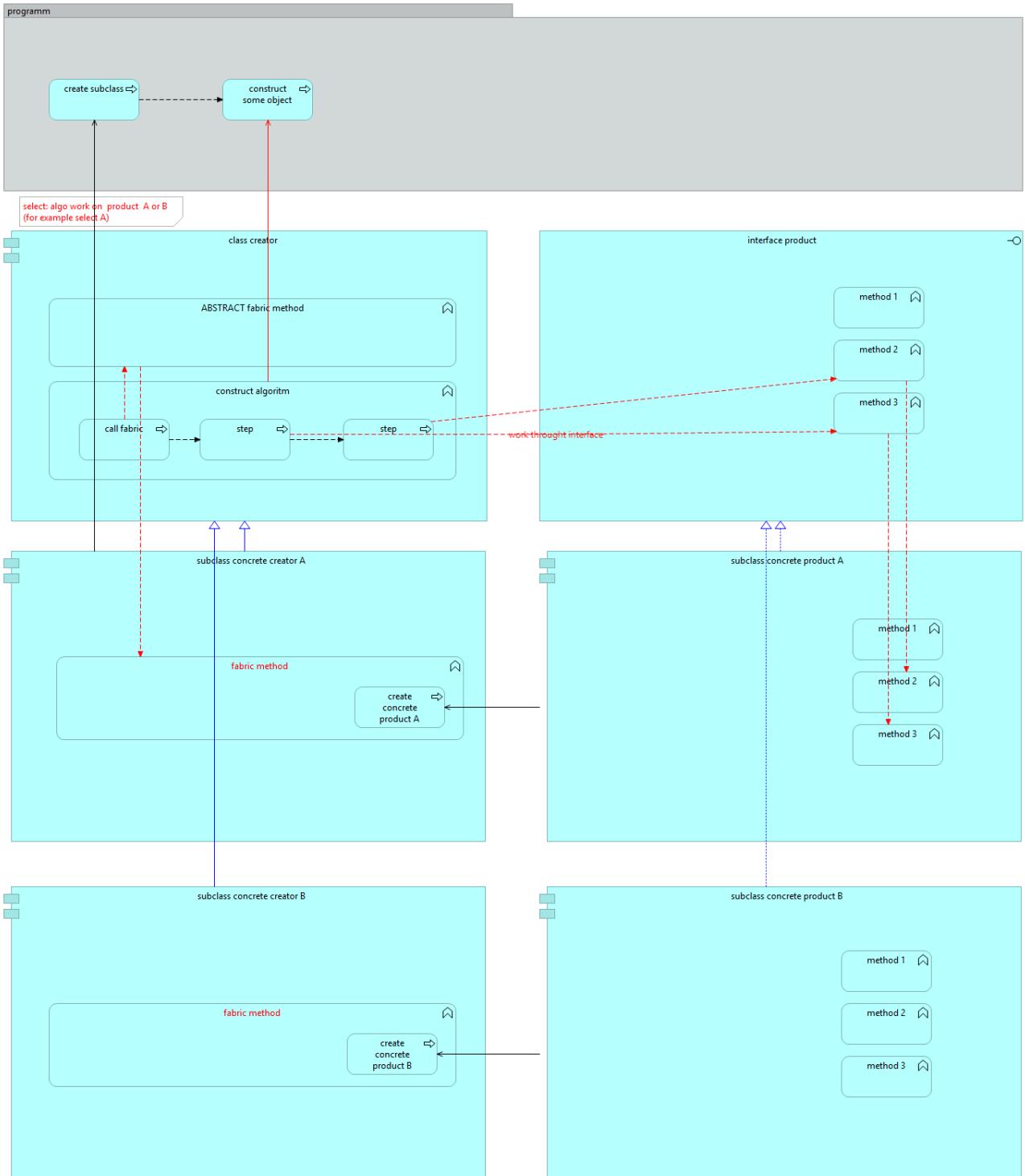


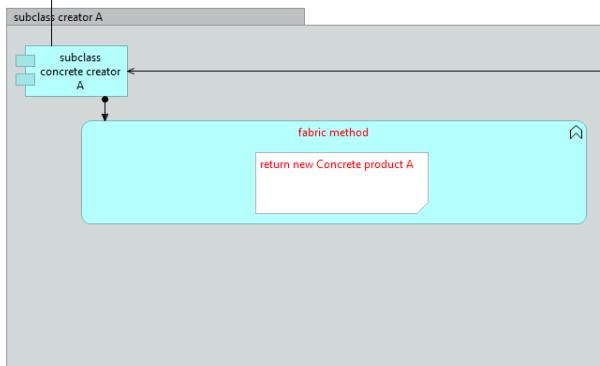
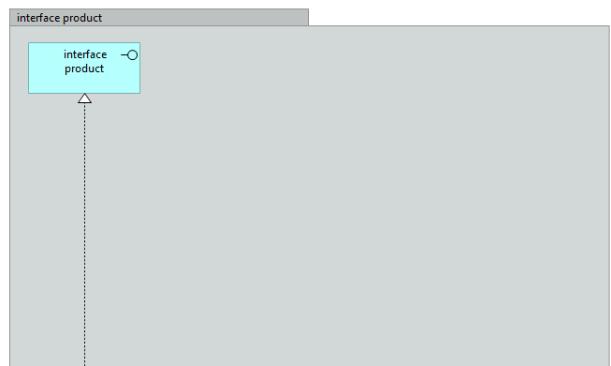
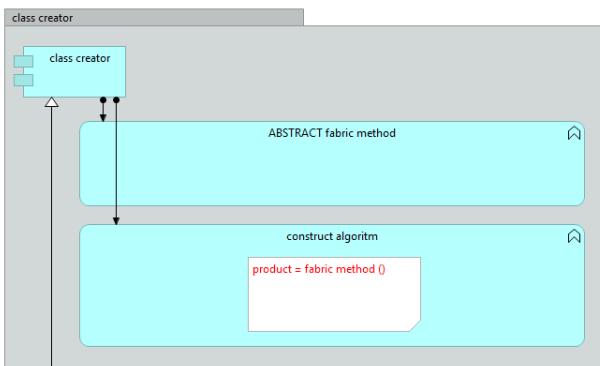
BUILDER



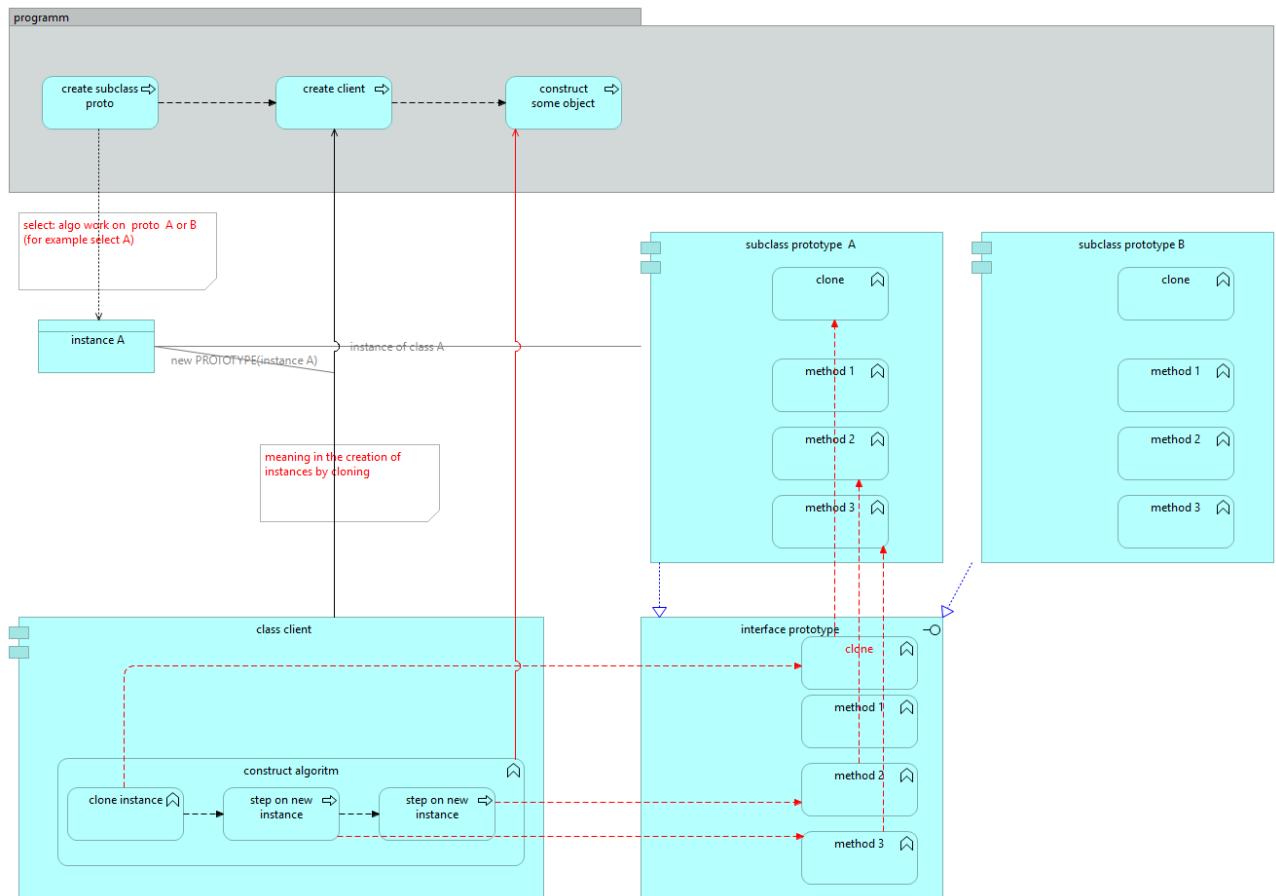


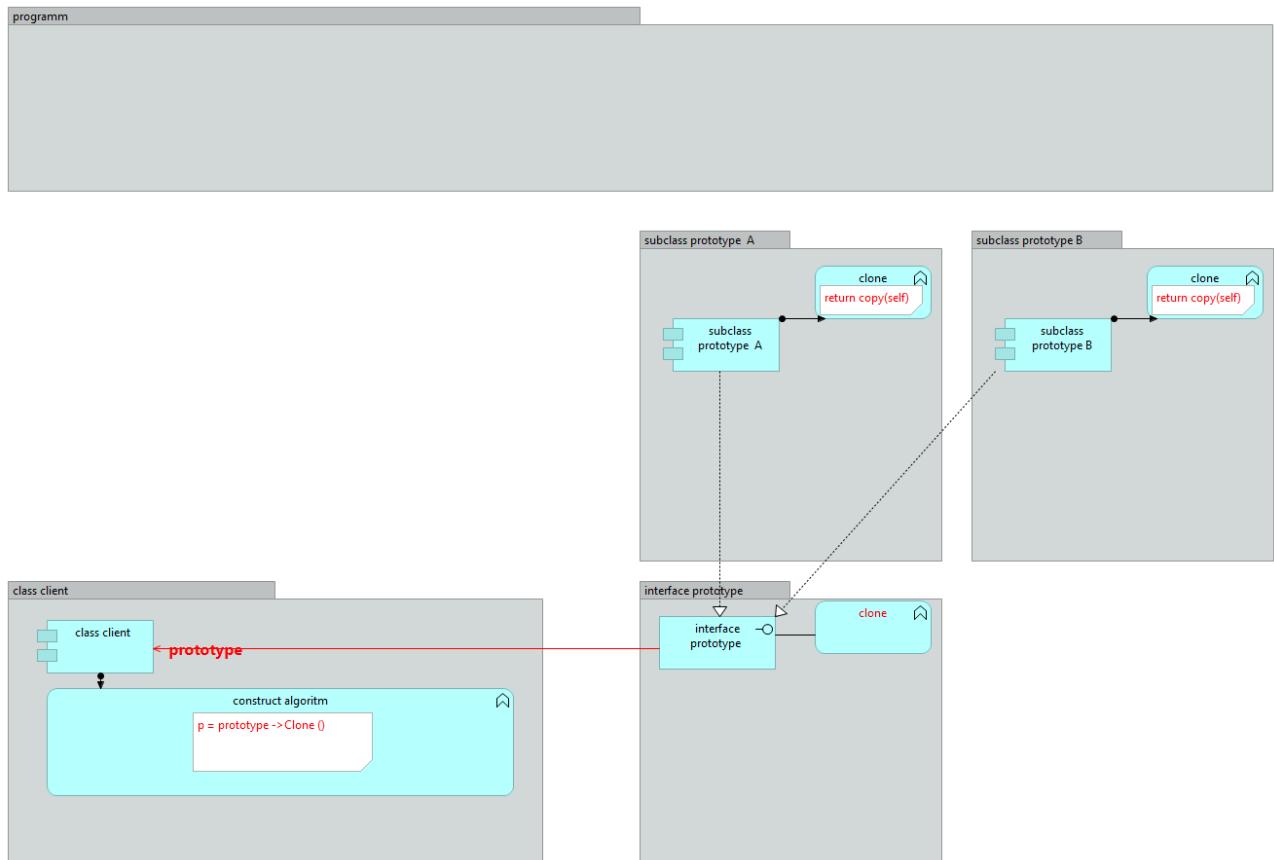
FACTORY METHOD



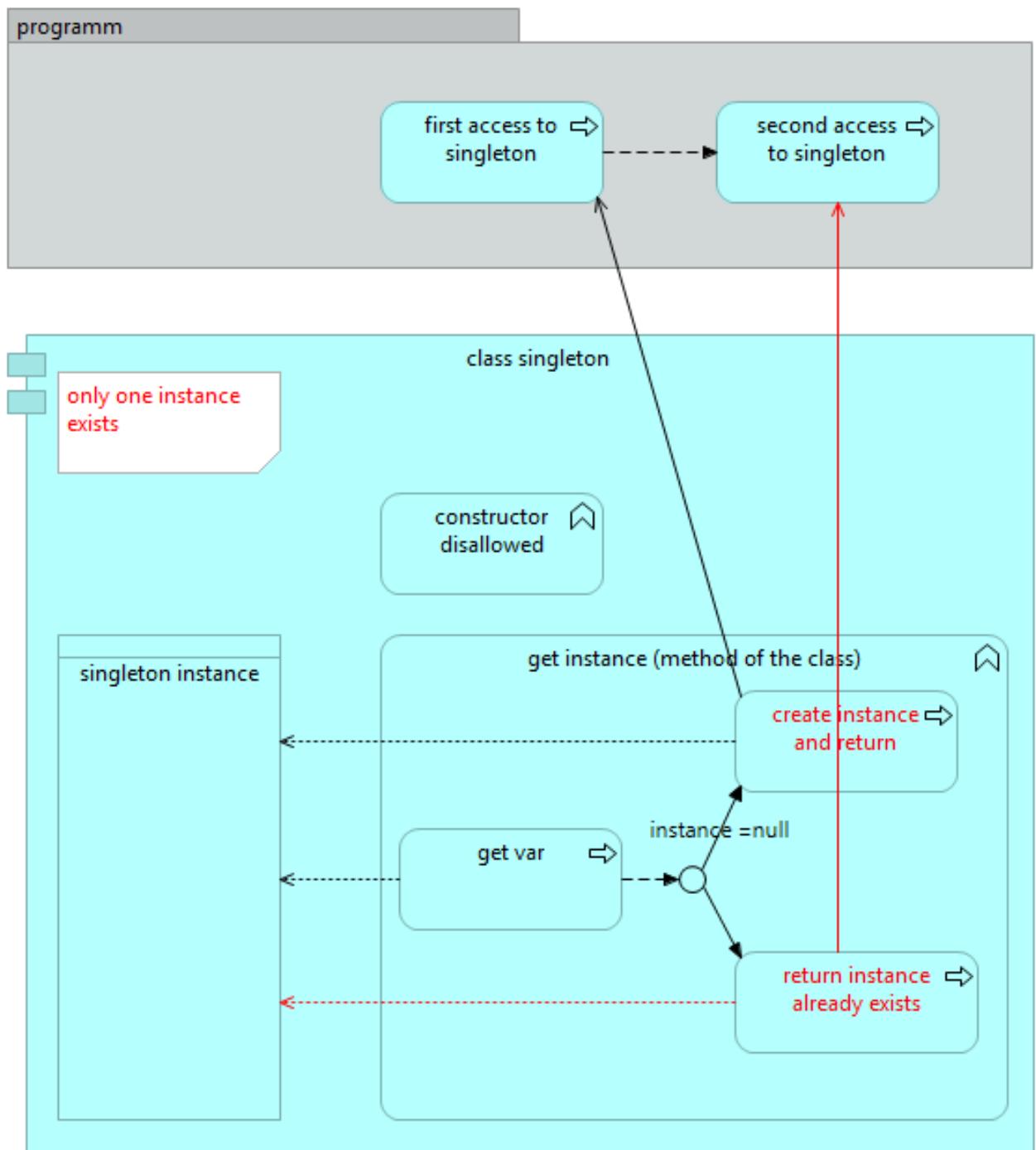


PROTOTYPE





SINGLETON



programm

class singleton

 class singleton

 singleton instance

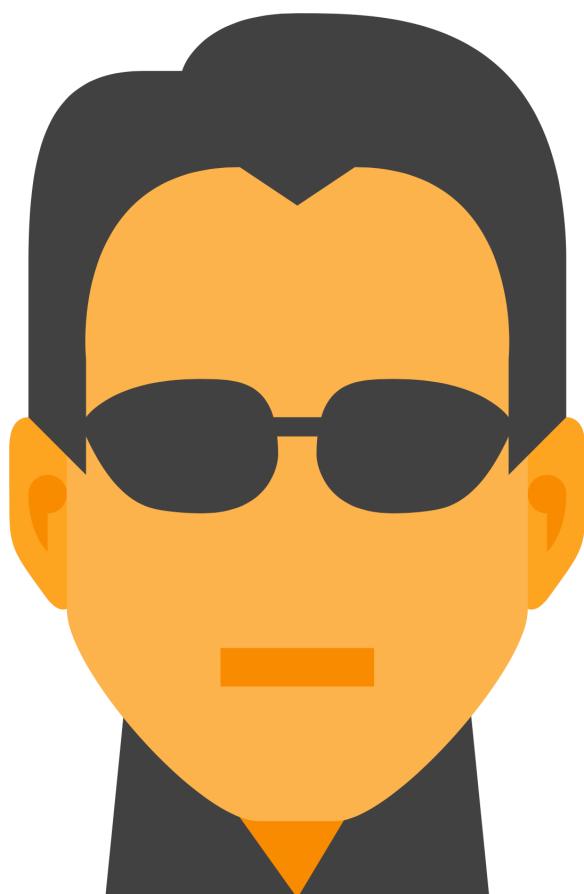
get instance (method of the class)

```
static method  
{  
    return static singleton instance  
}
```

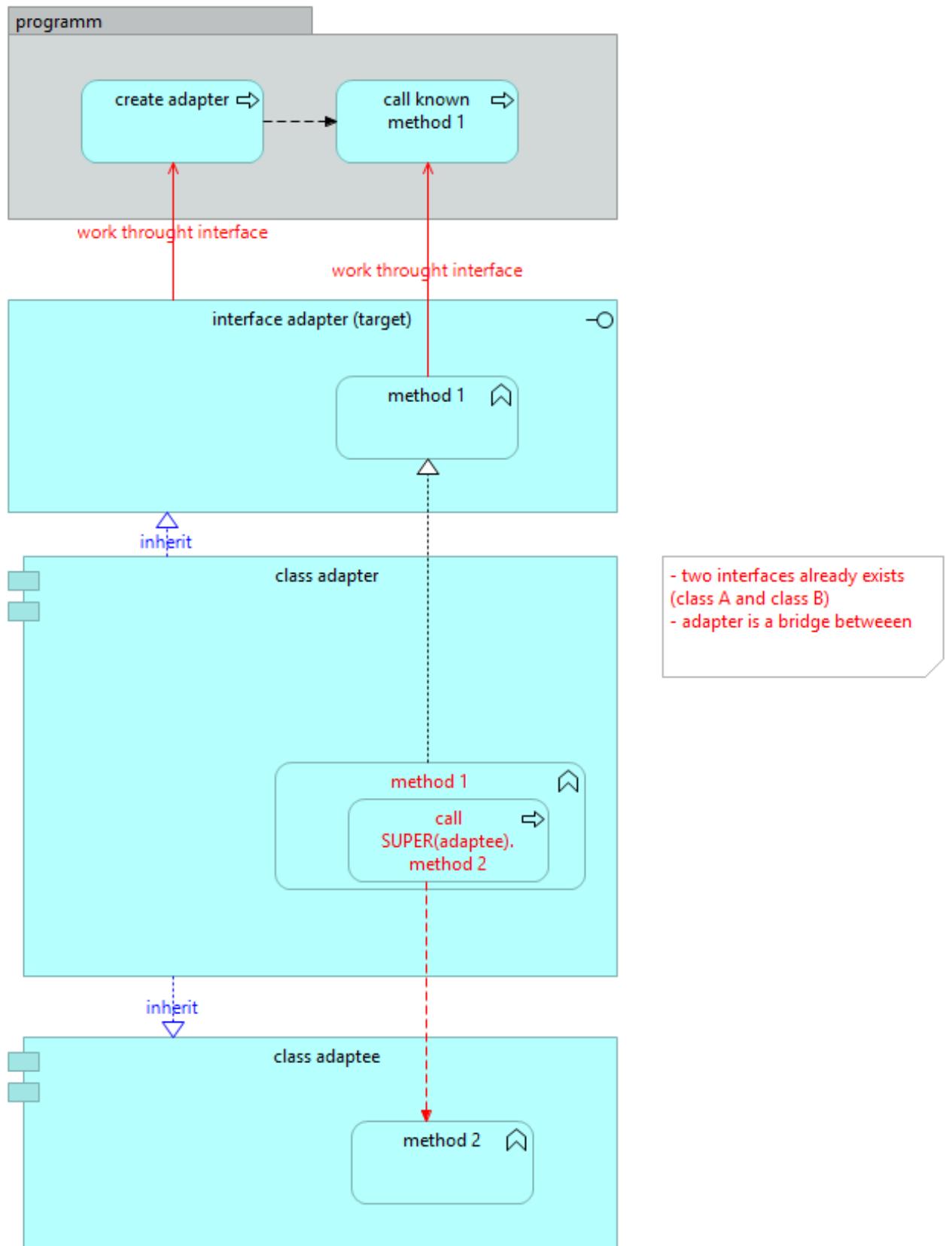
STRUCTURAL PATTERNS

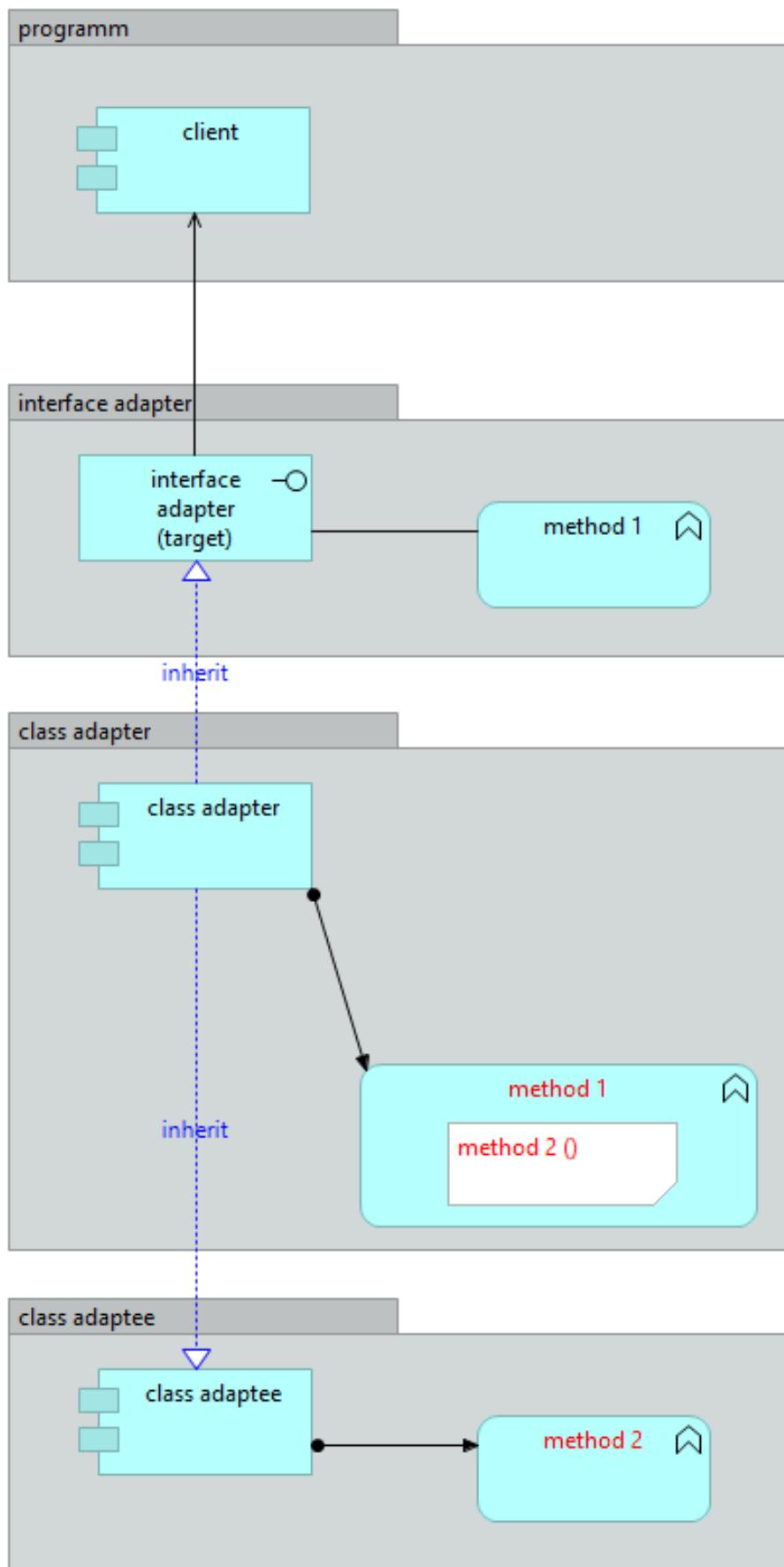
DESIGN PATTERNS

GoF

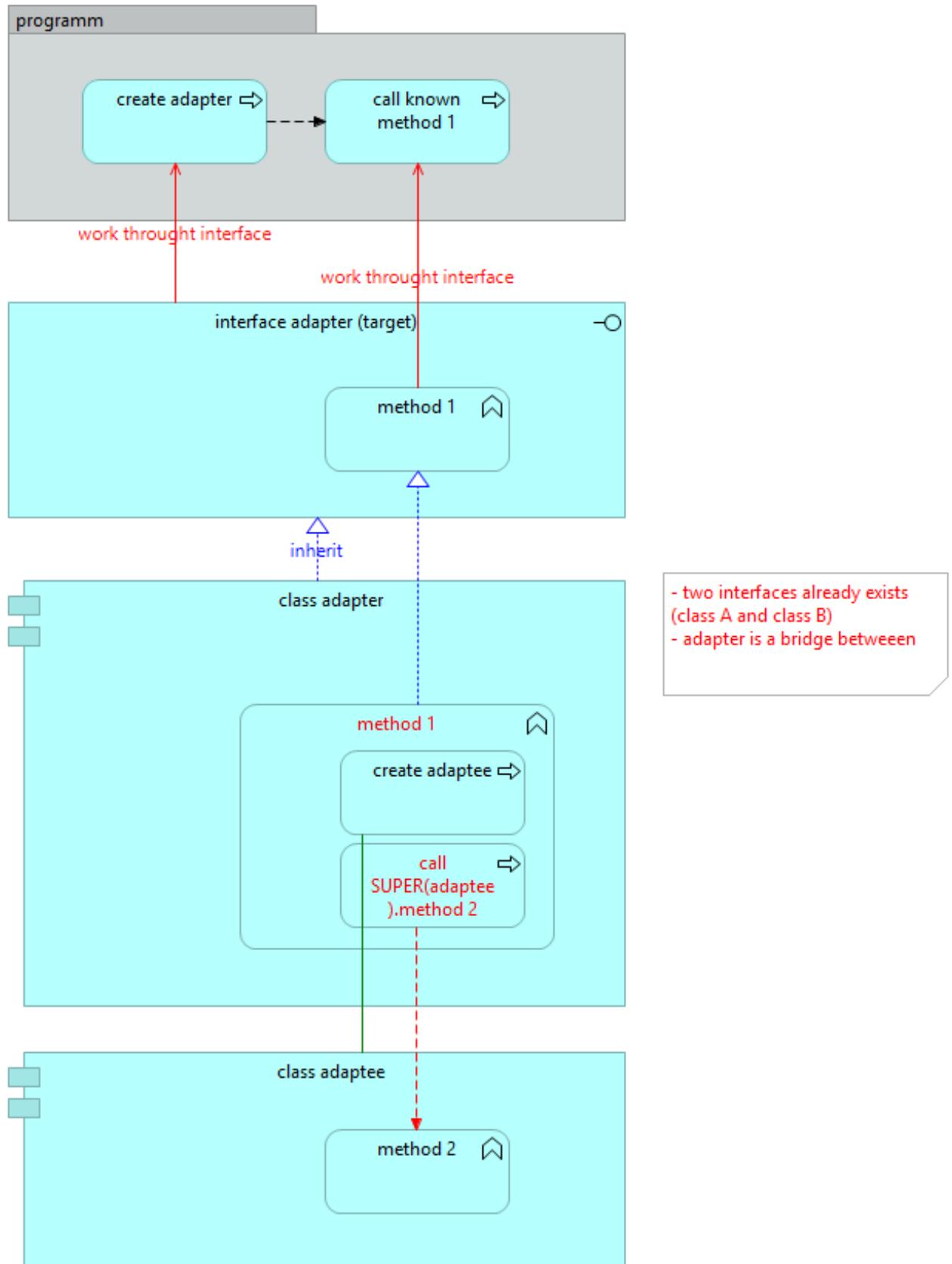


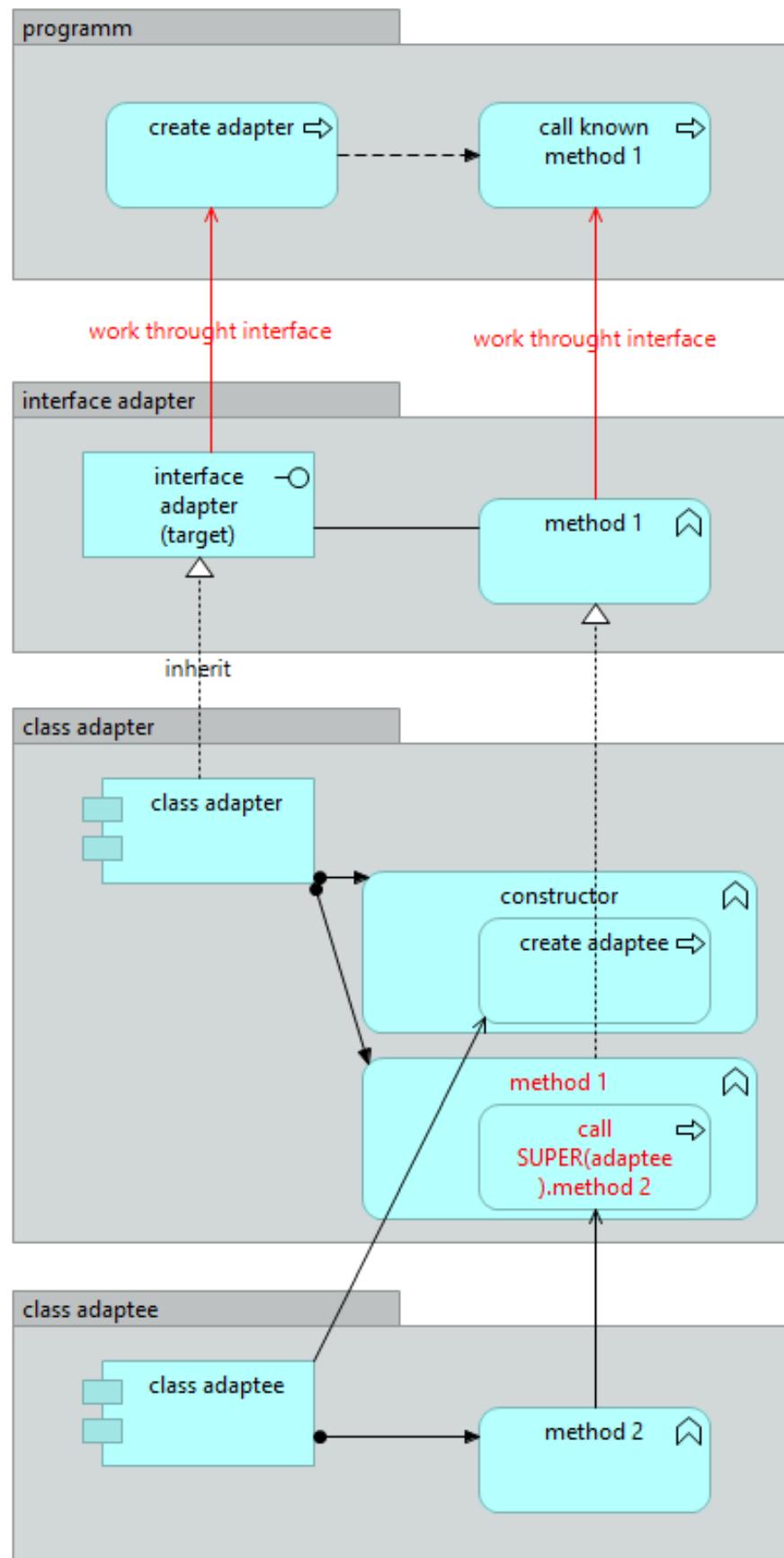
ADAPTER OF CLASS

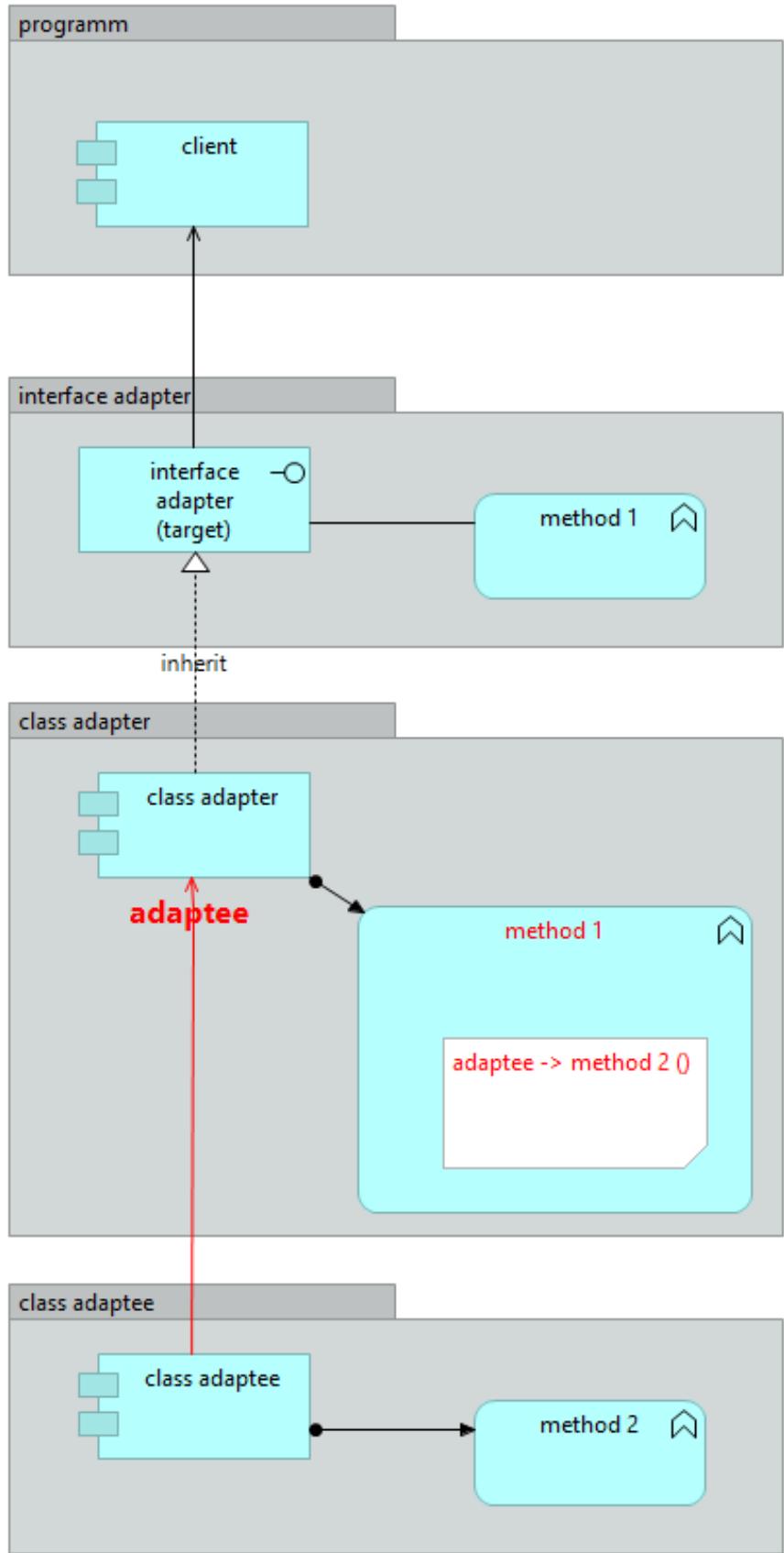




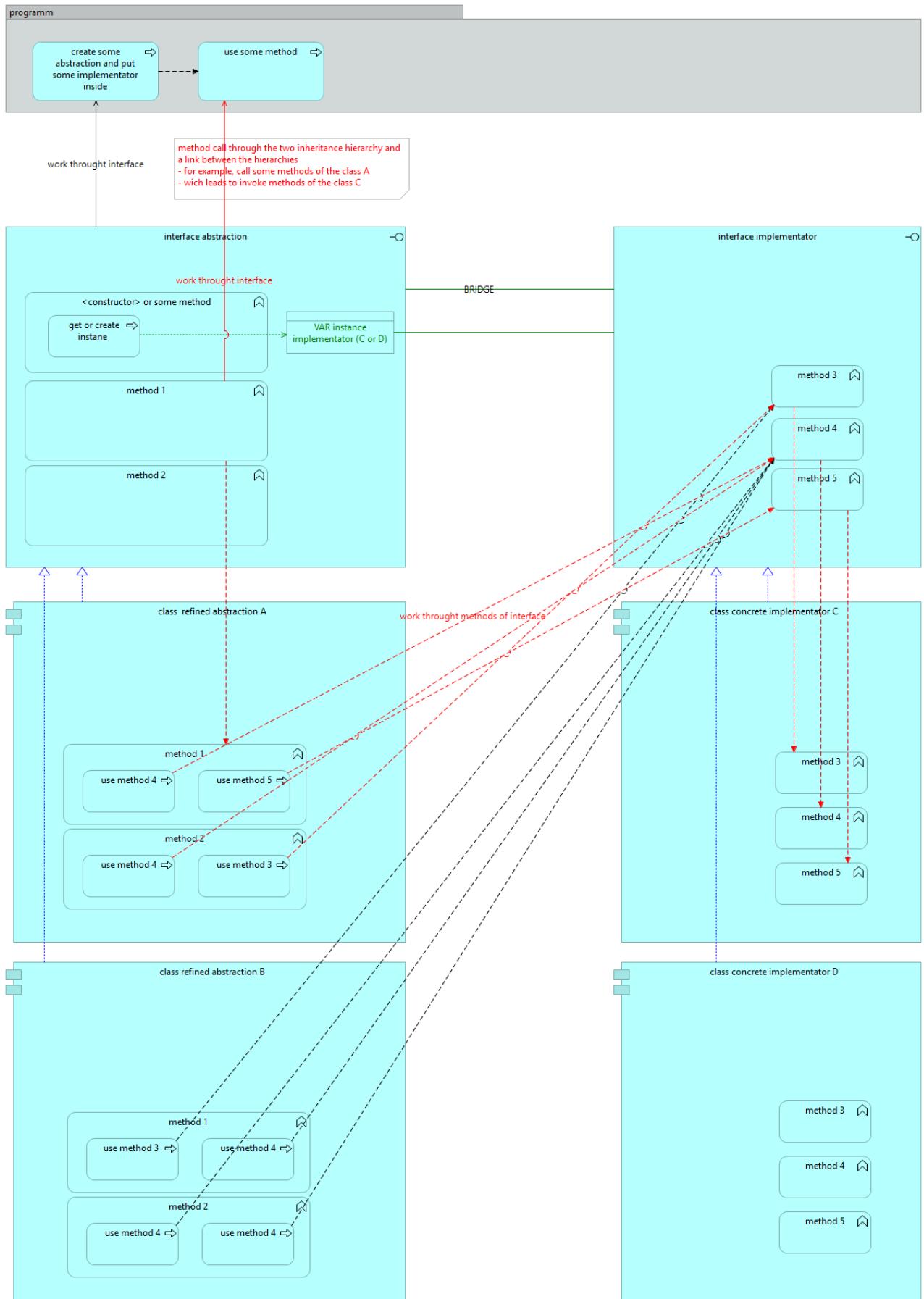
ADAPTER OF OBJECT

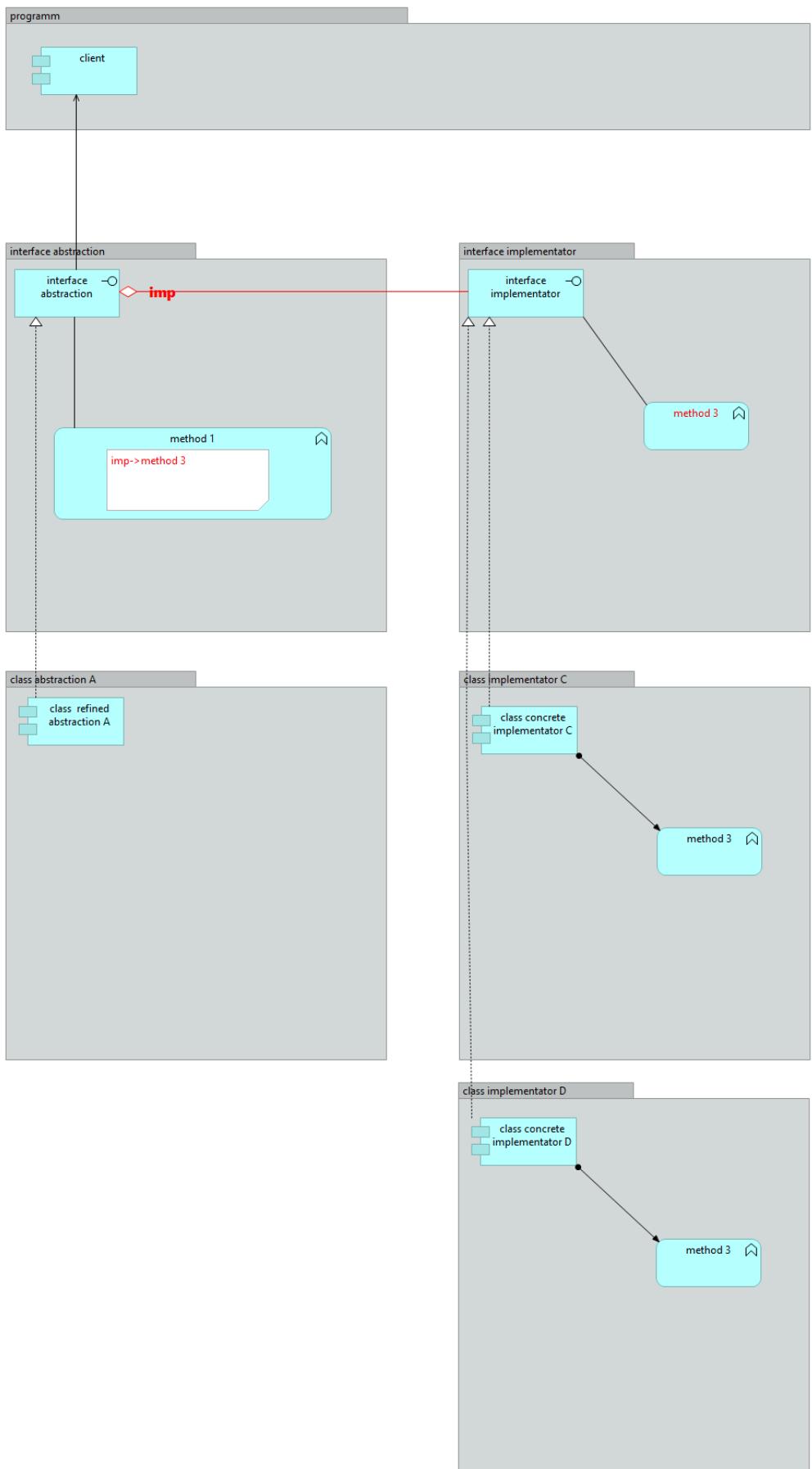




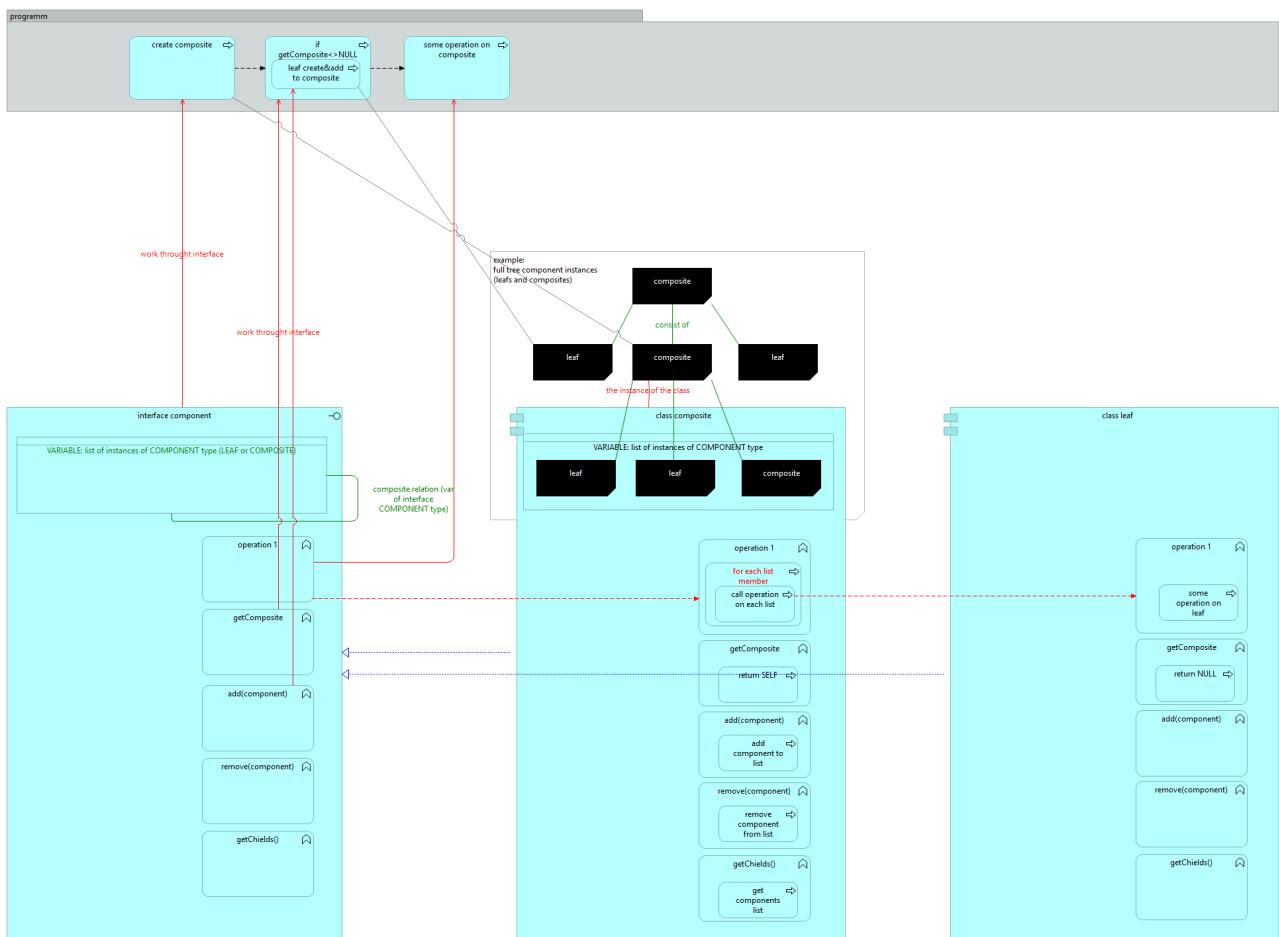


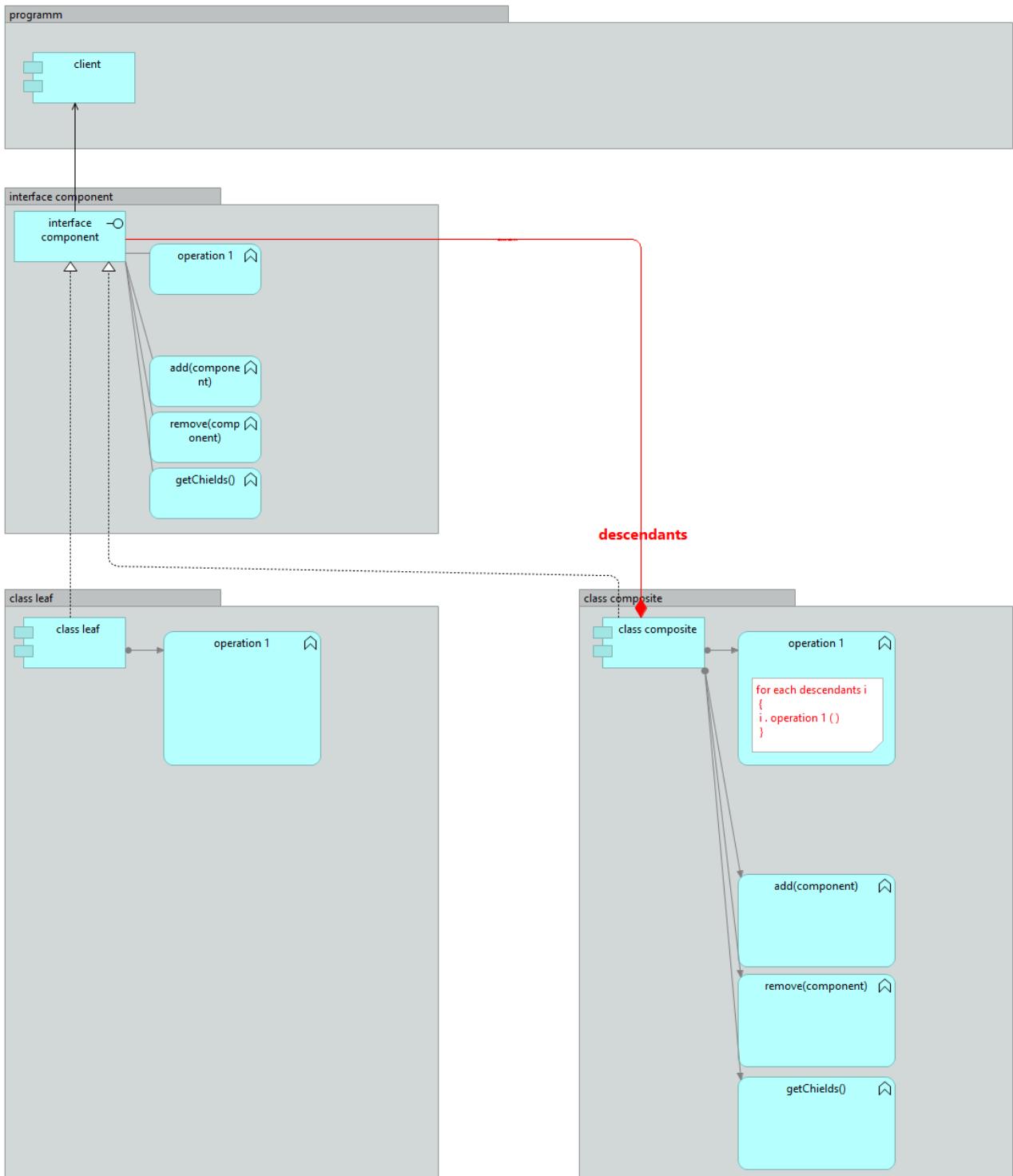
BRIDGE



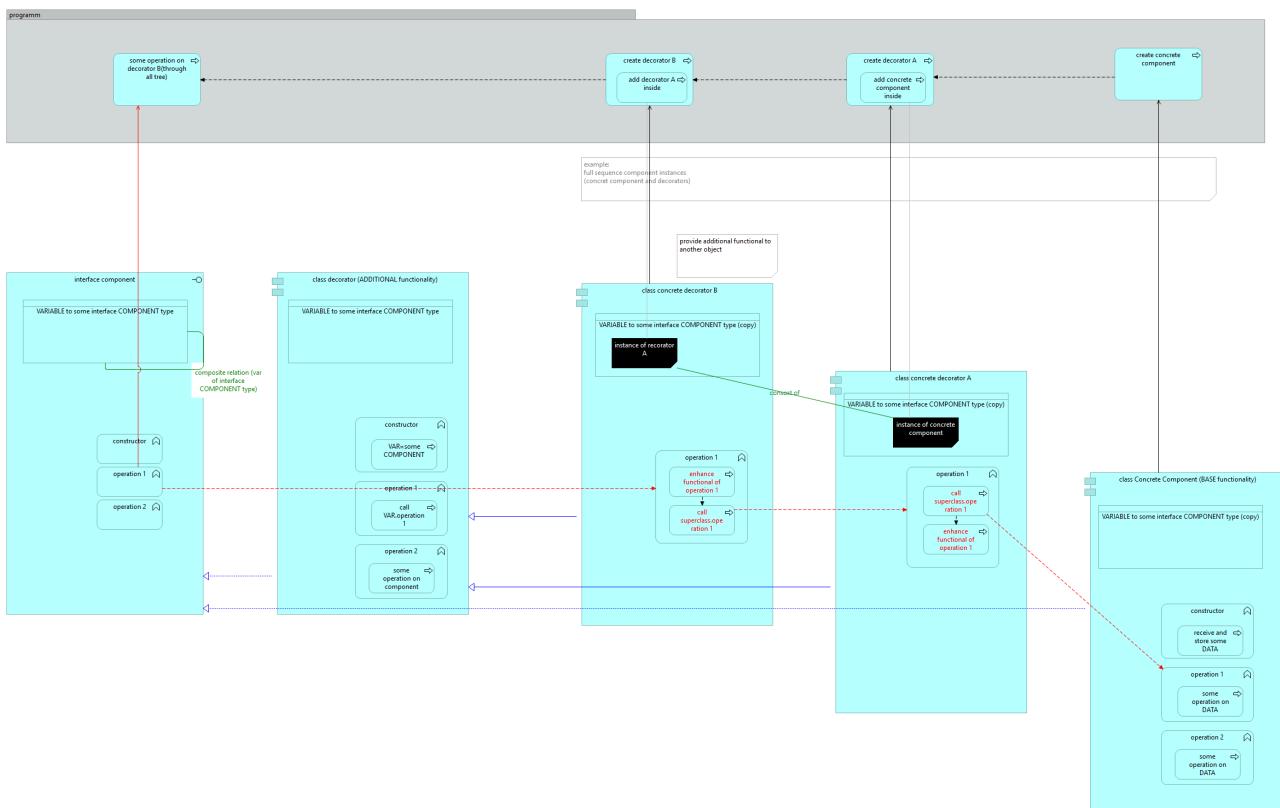


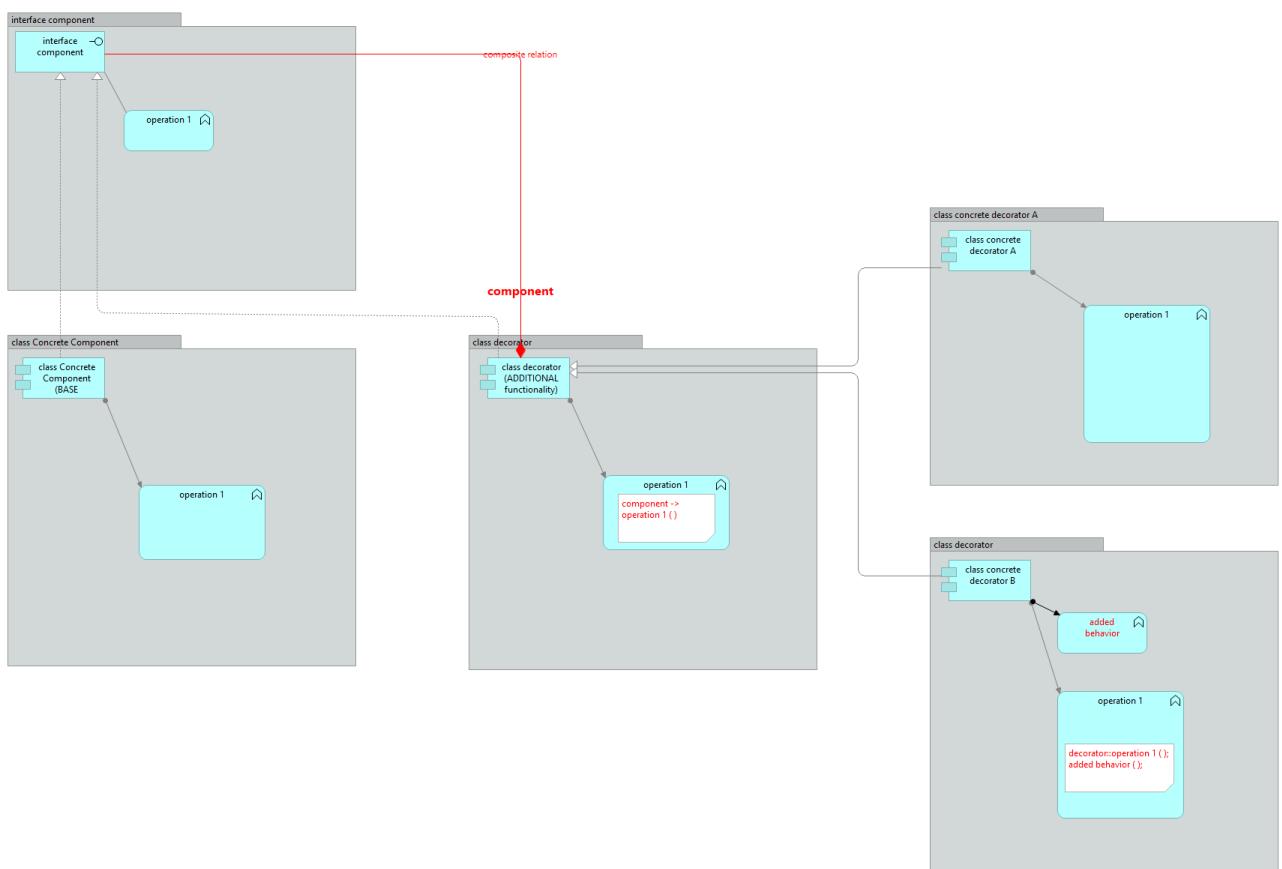
COMPOSITE



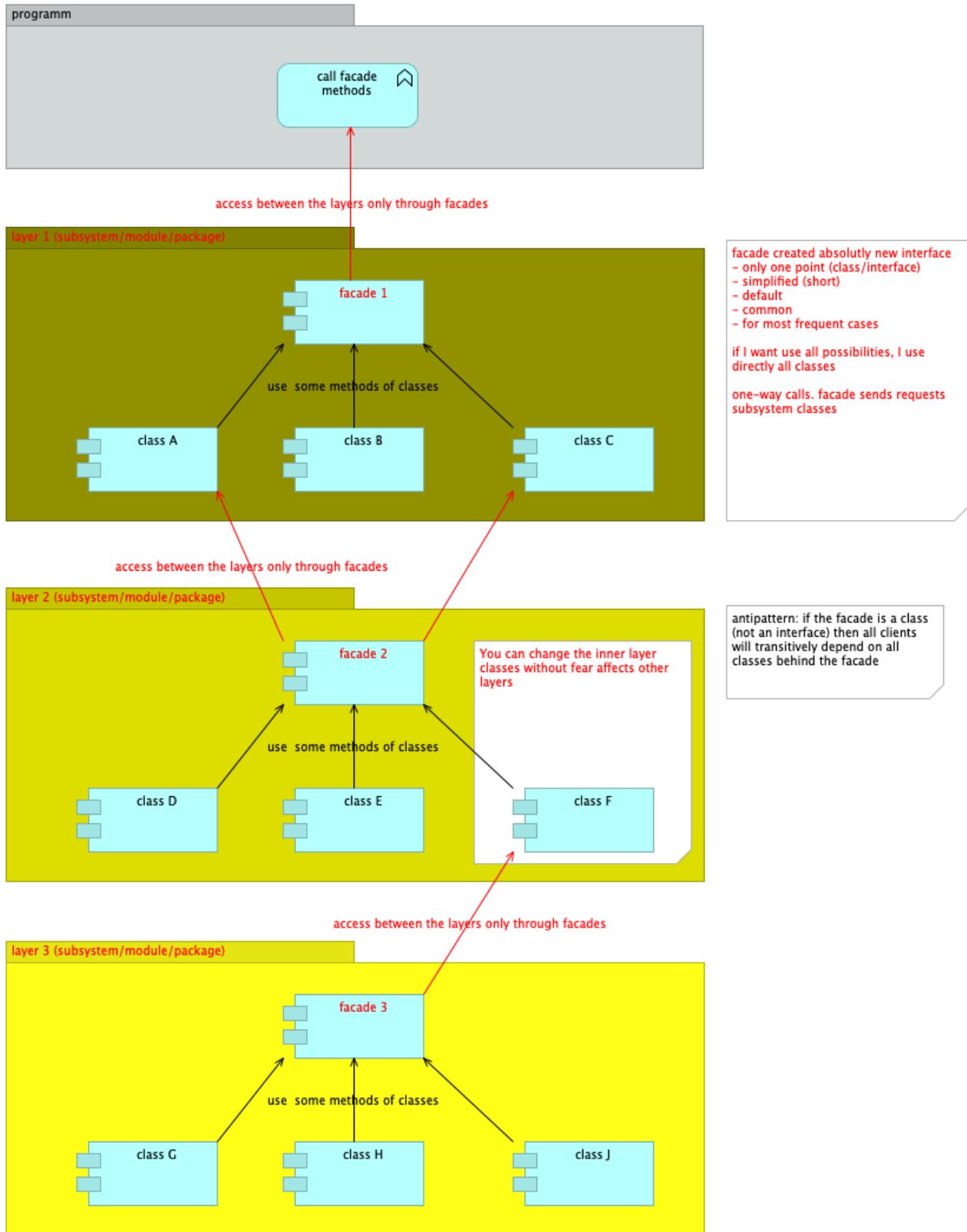


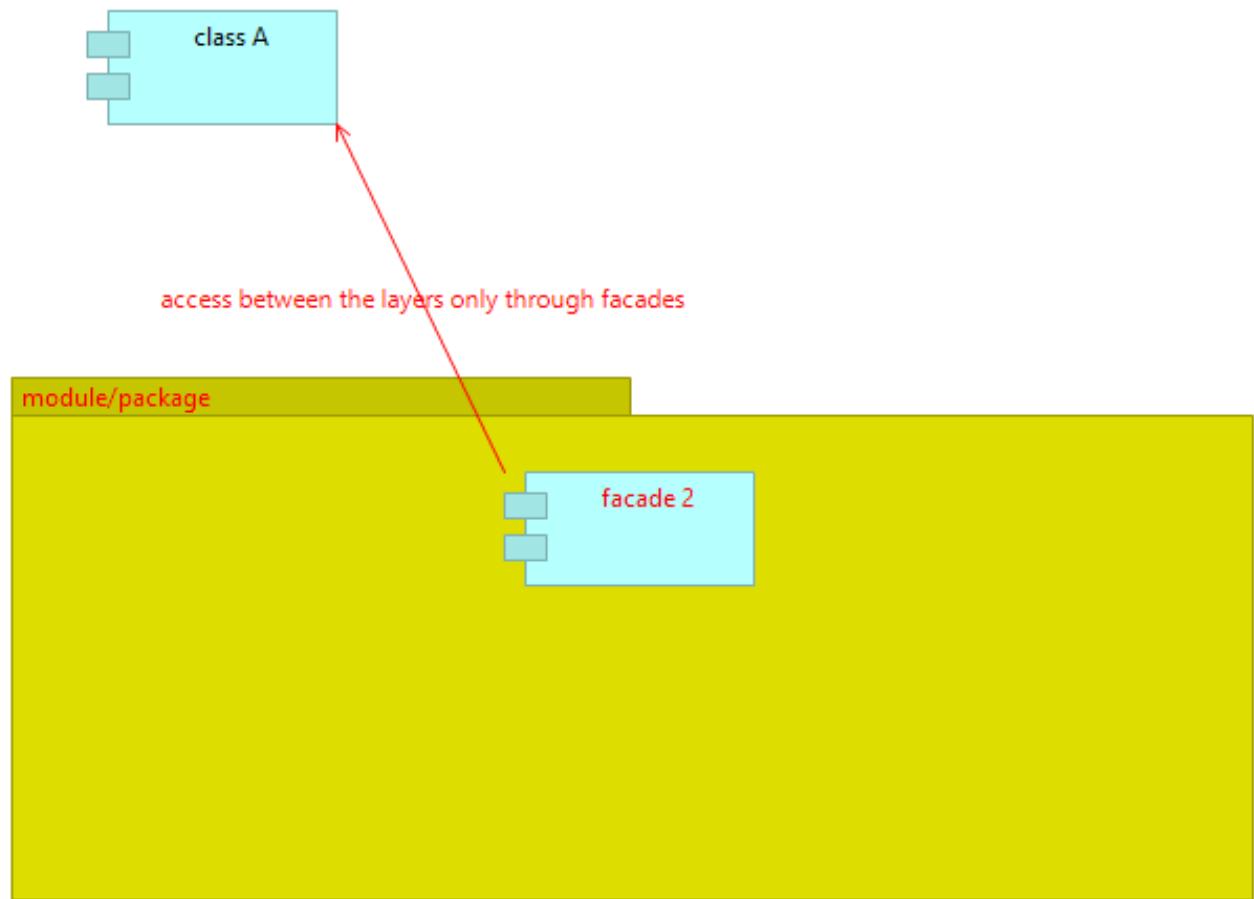
DECORATOR

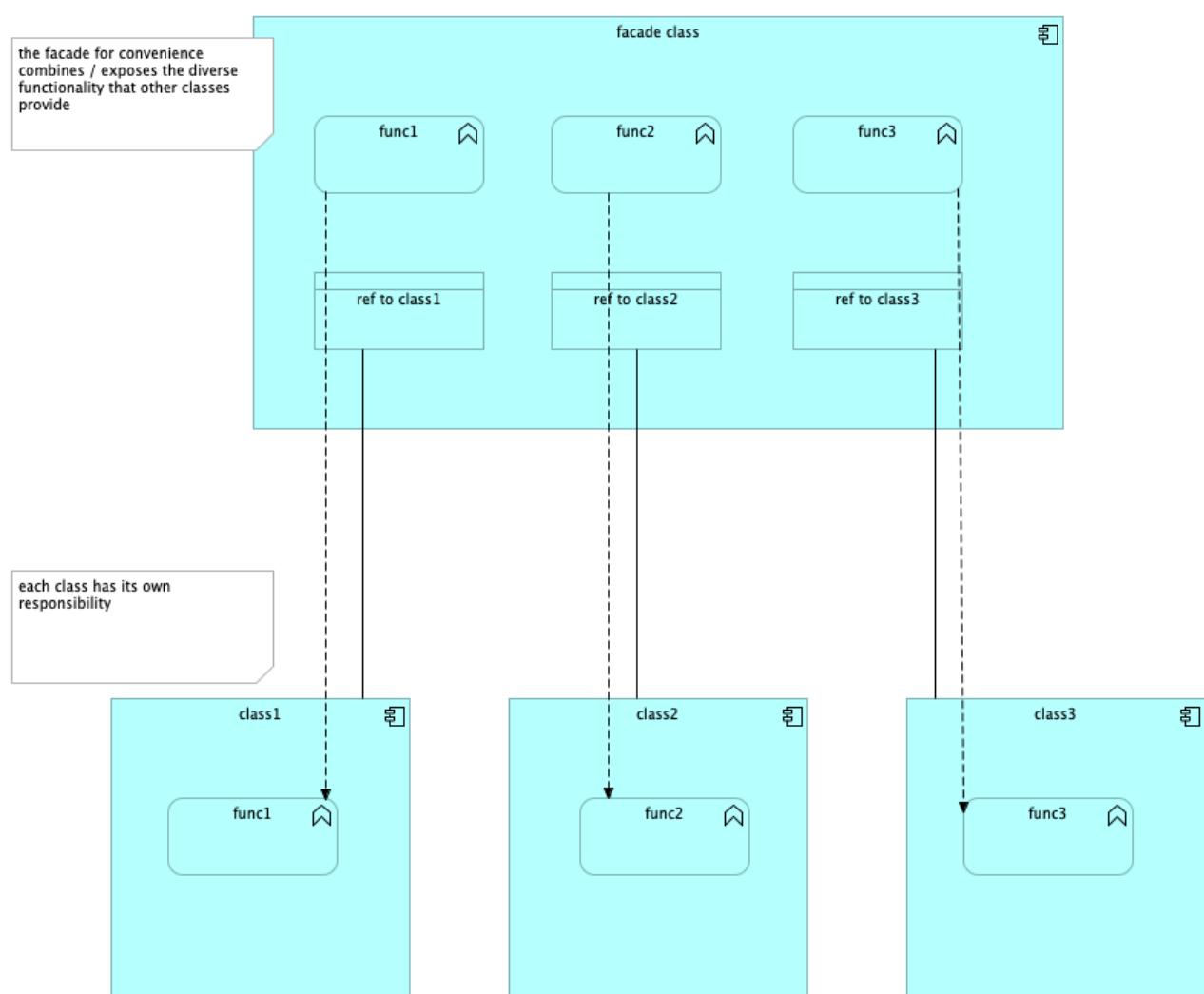




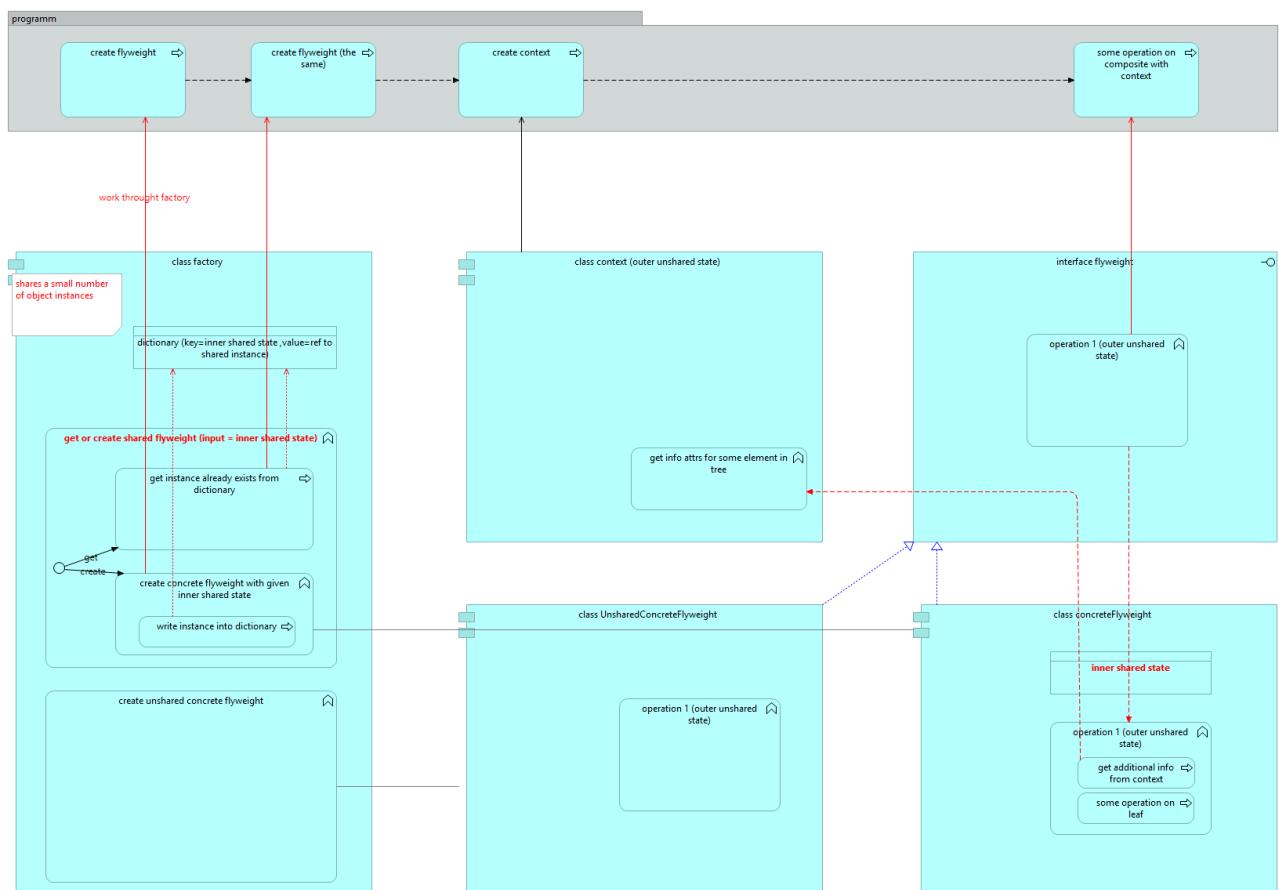
FACADE

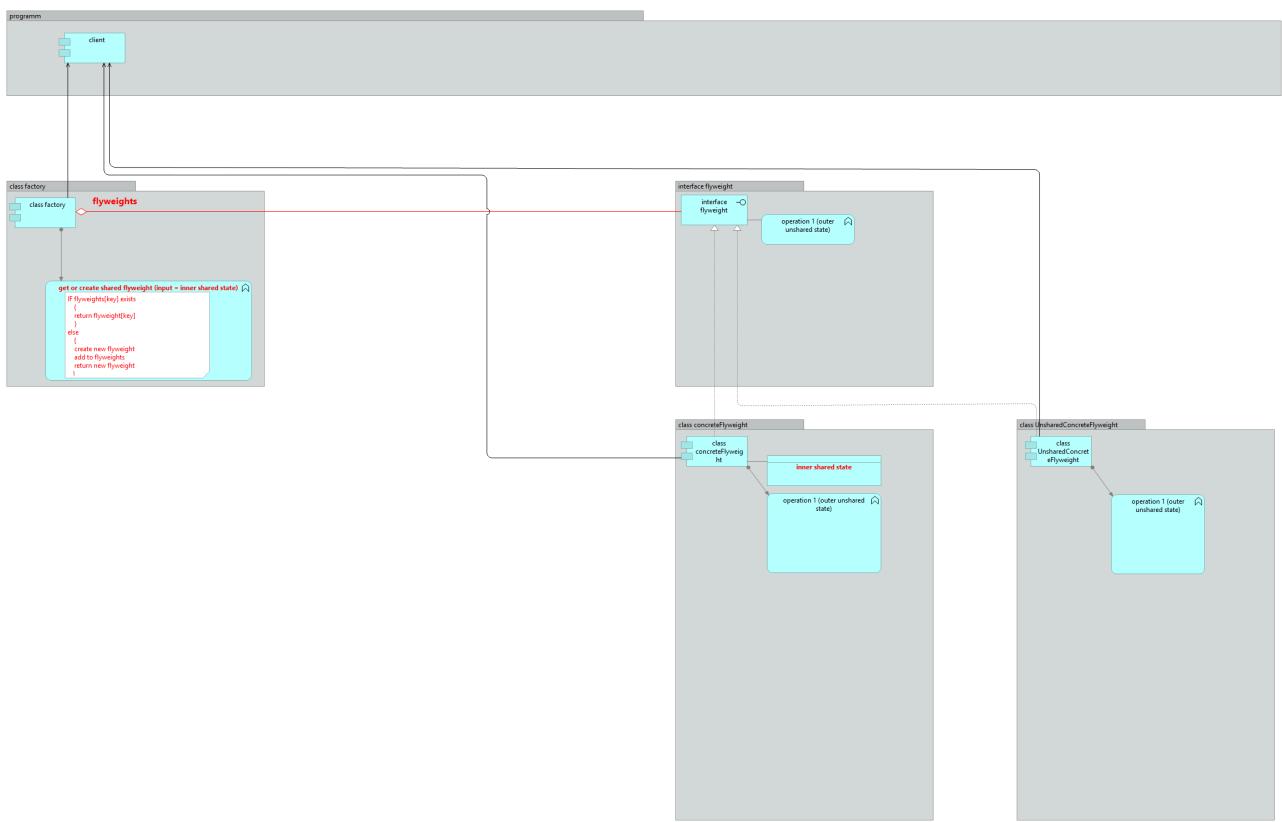




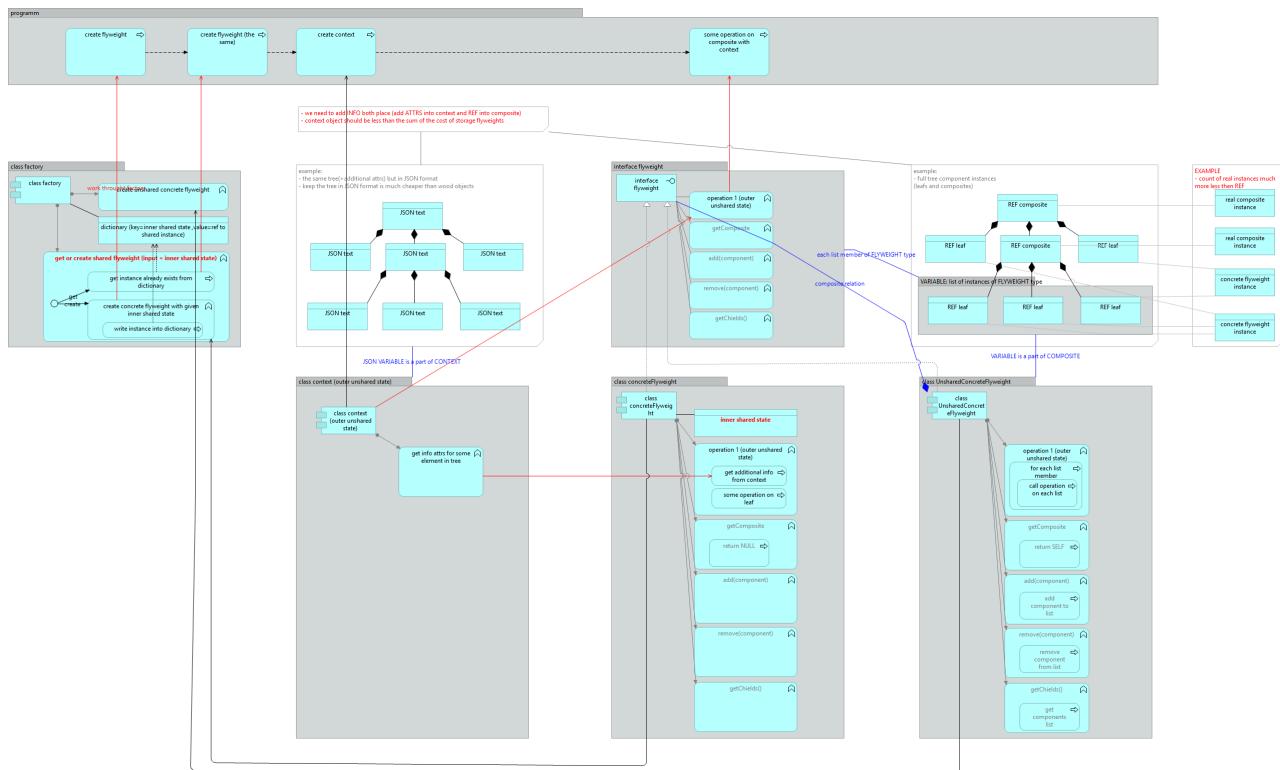


FLYWEIGHT

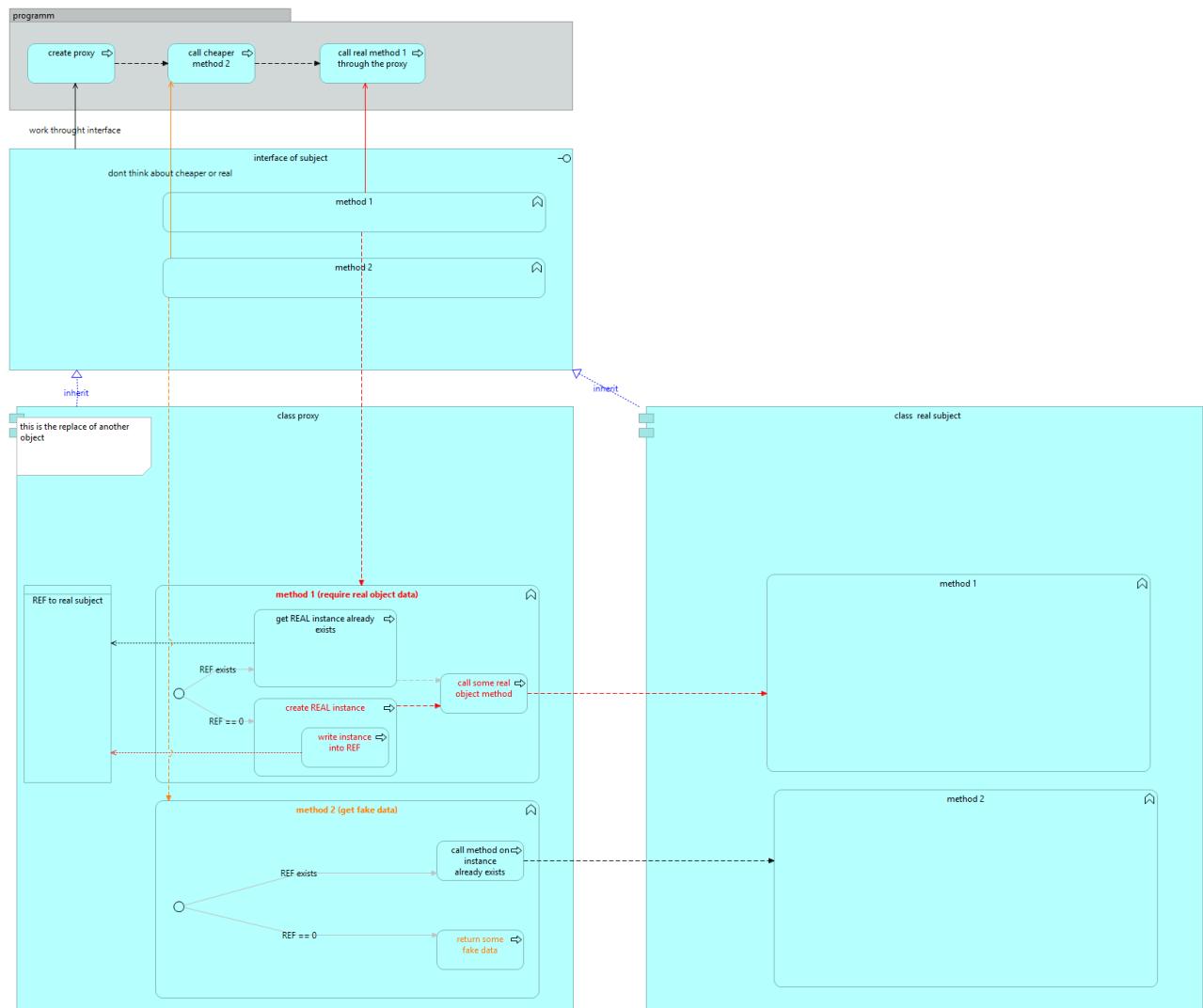


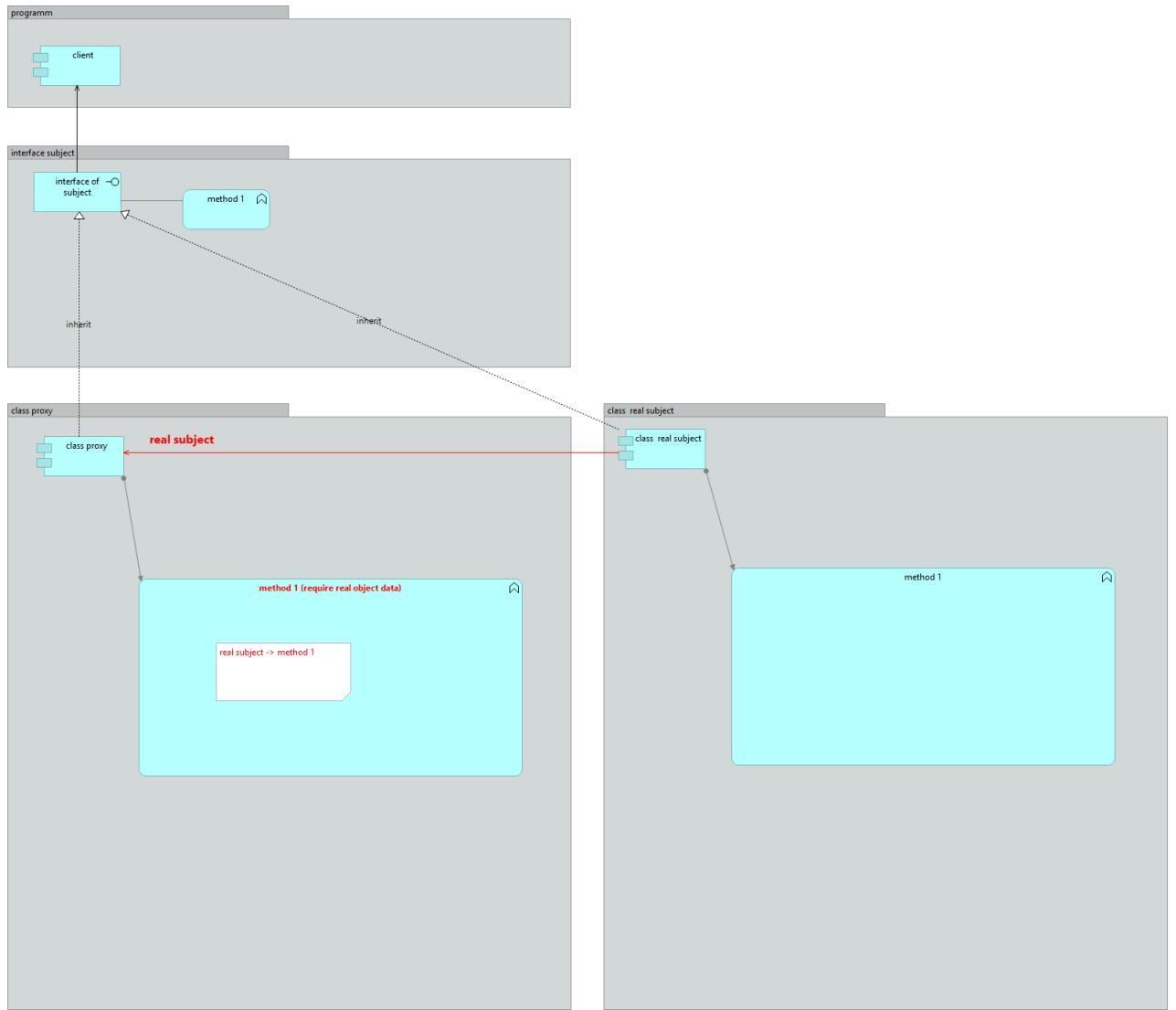


FLYWEIGHT + COMPOSITE



PROXY





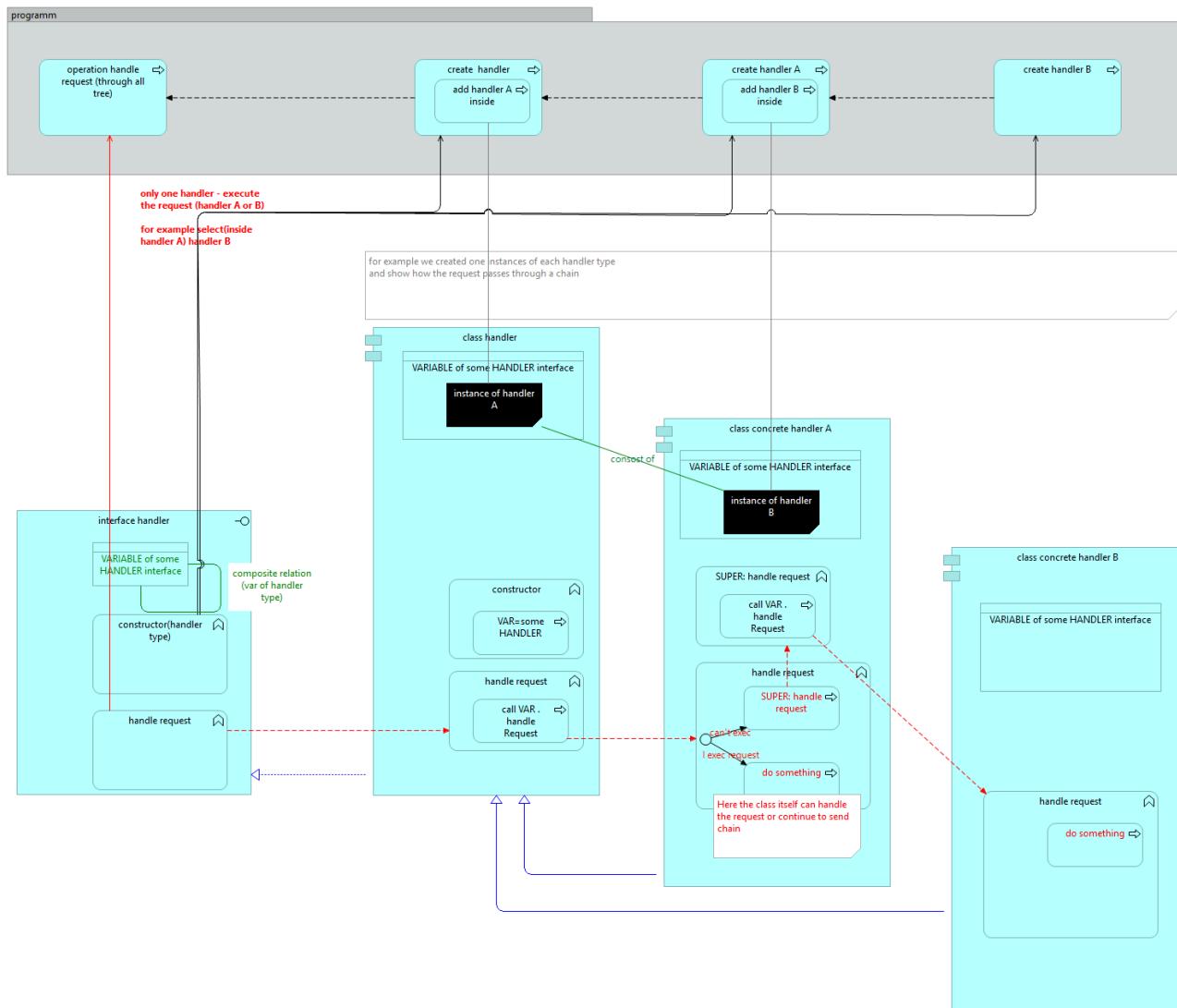
BEHAVIORAL PATTERNS

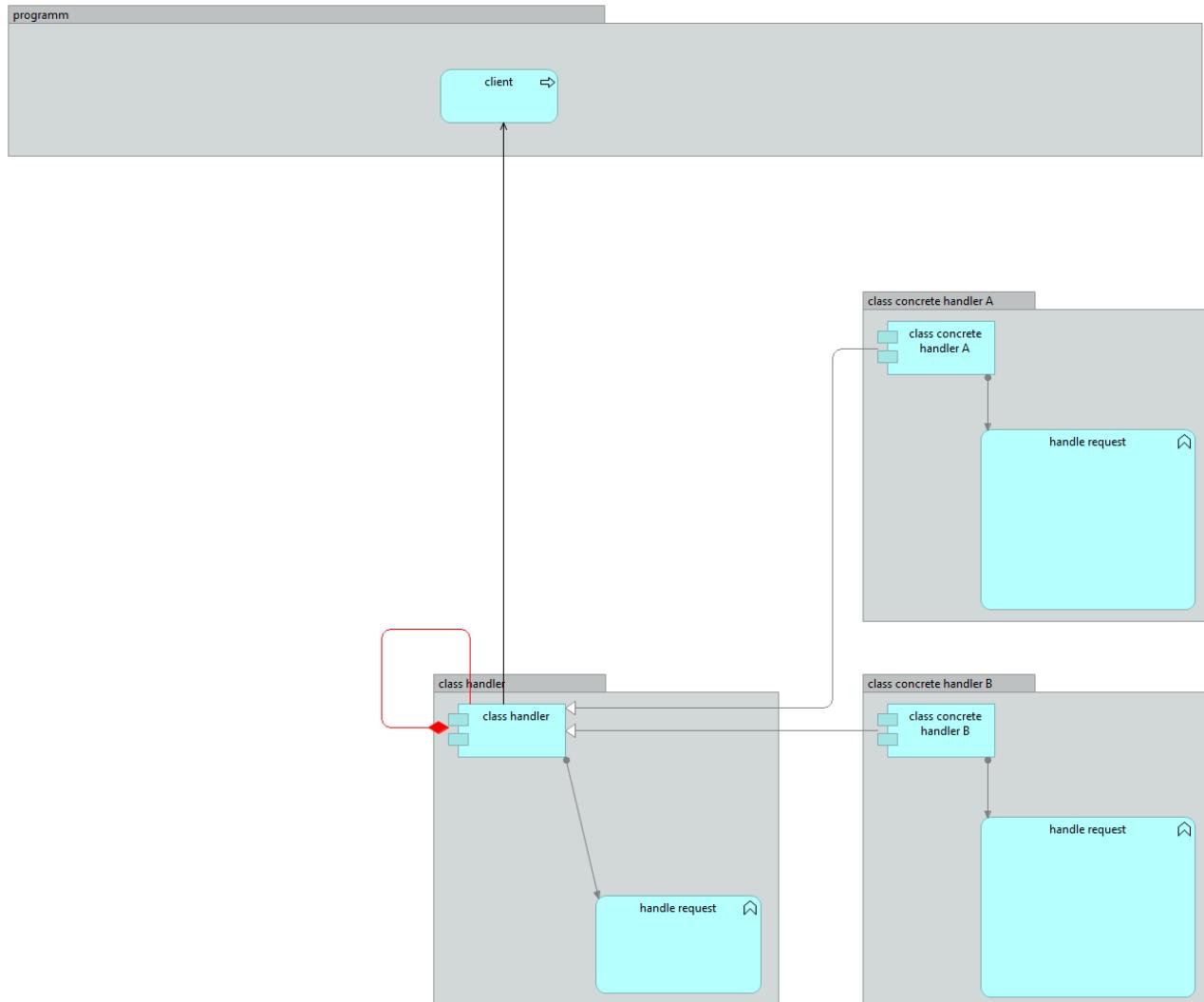
DESIGN PATTERNS

GoF

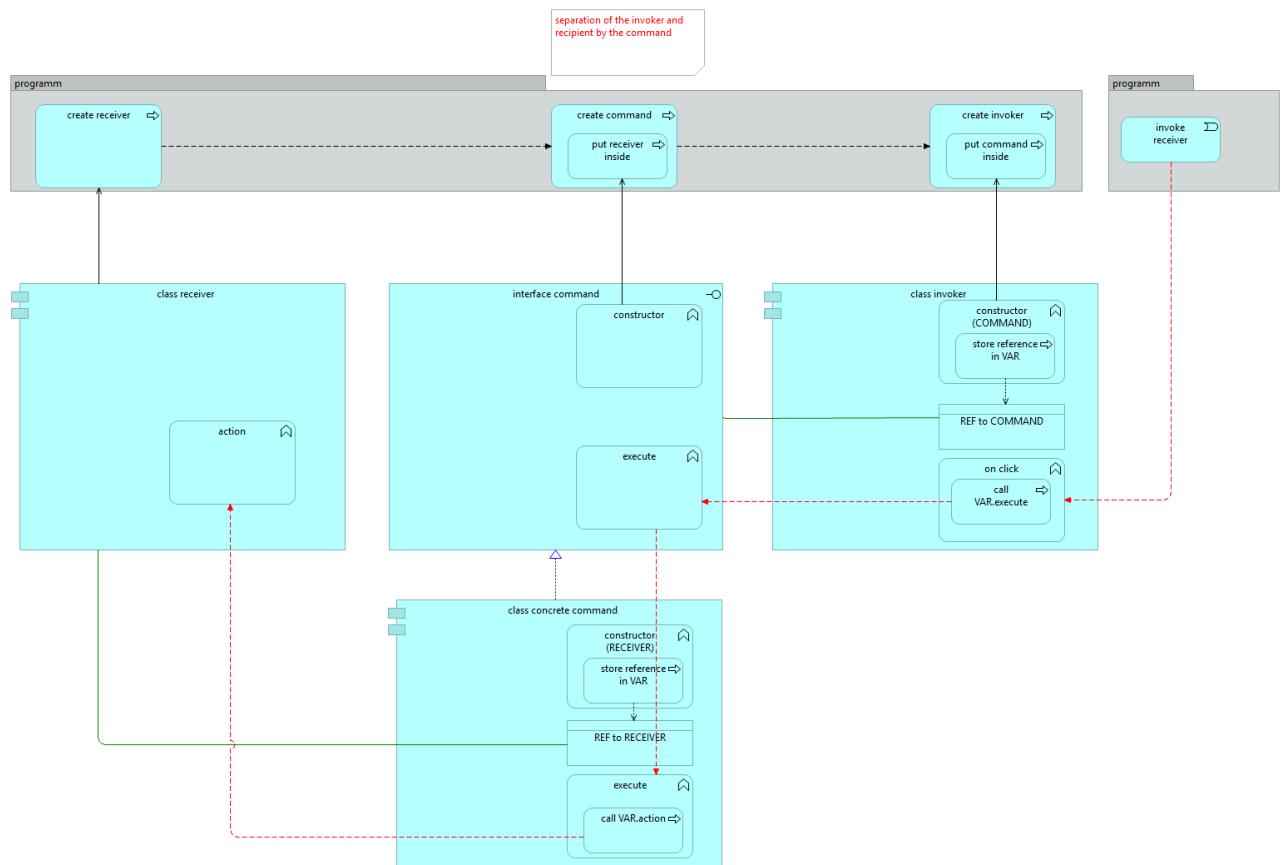


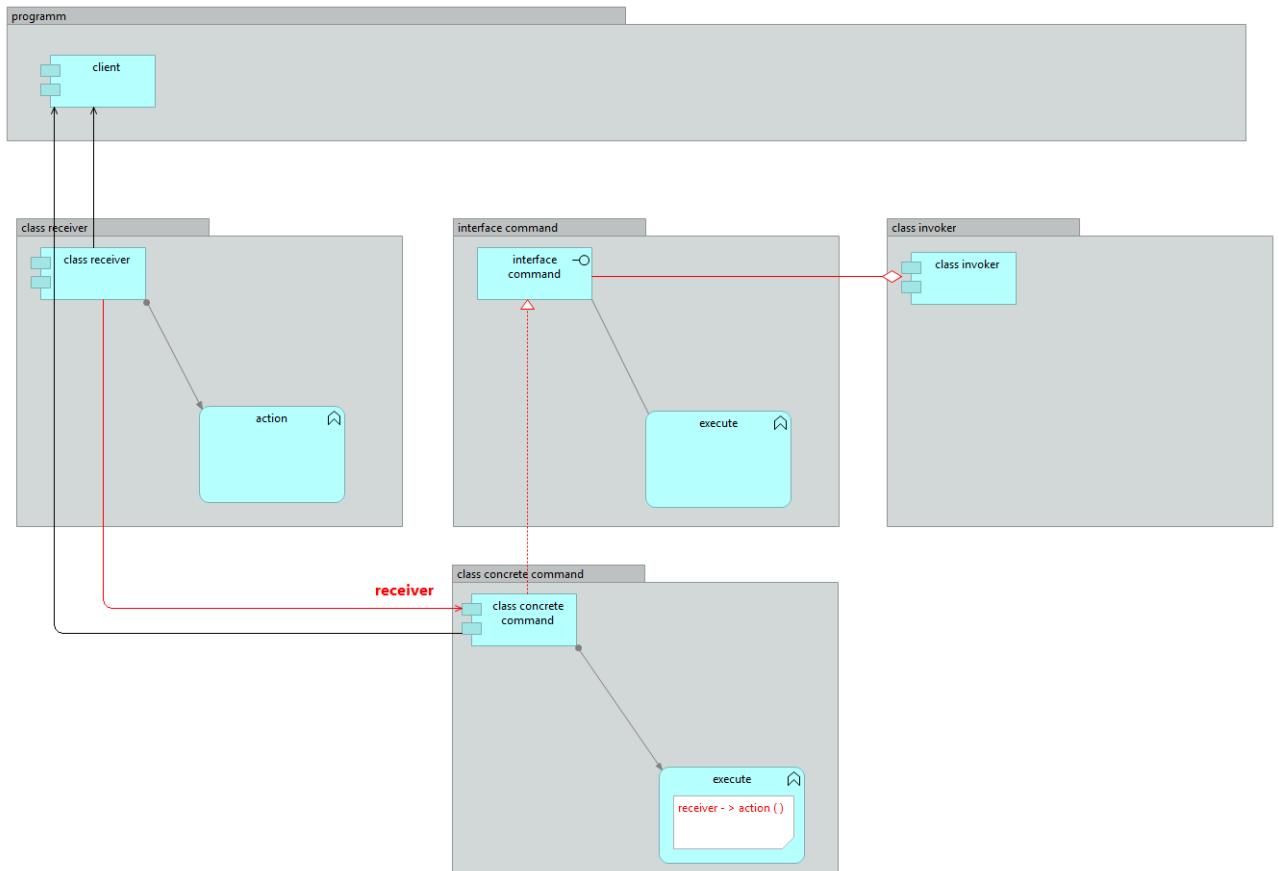
CHAIN OF RESPONSIBILITY



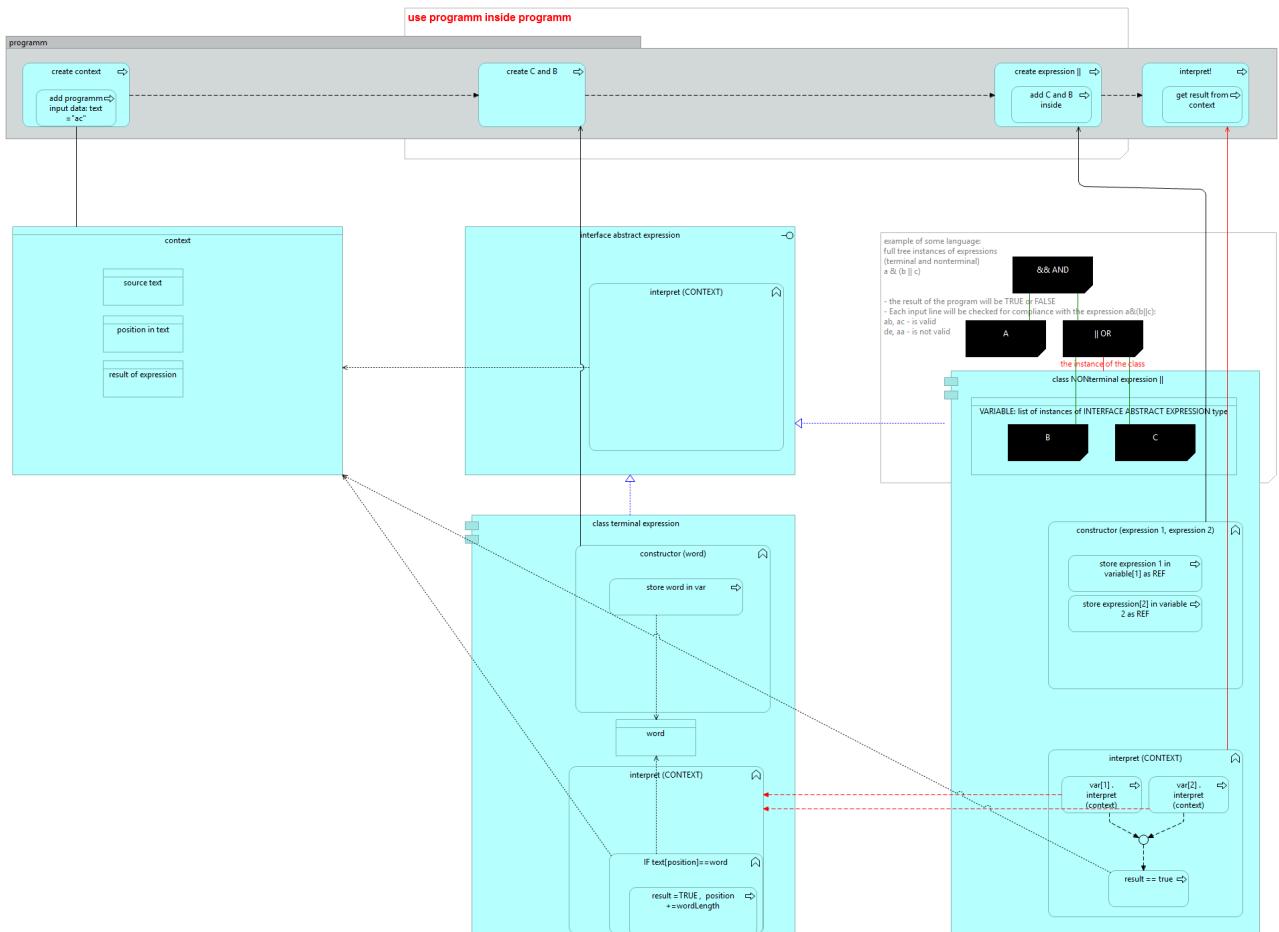


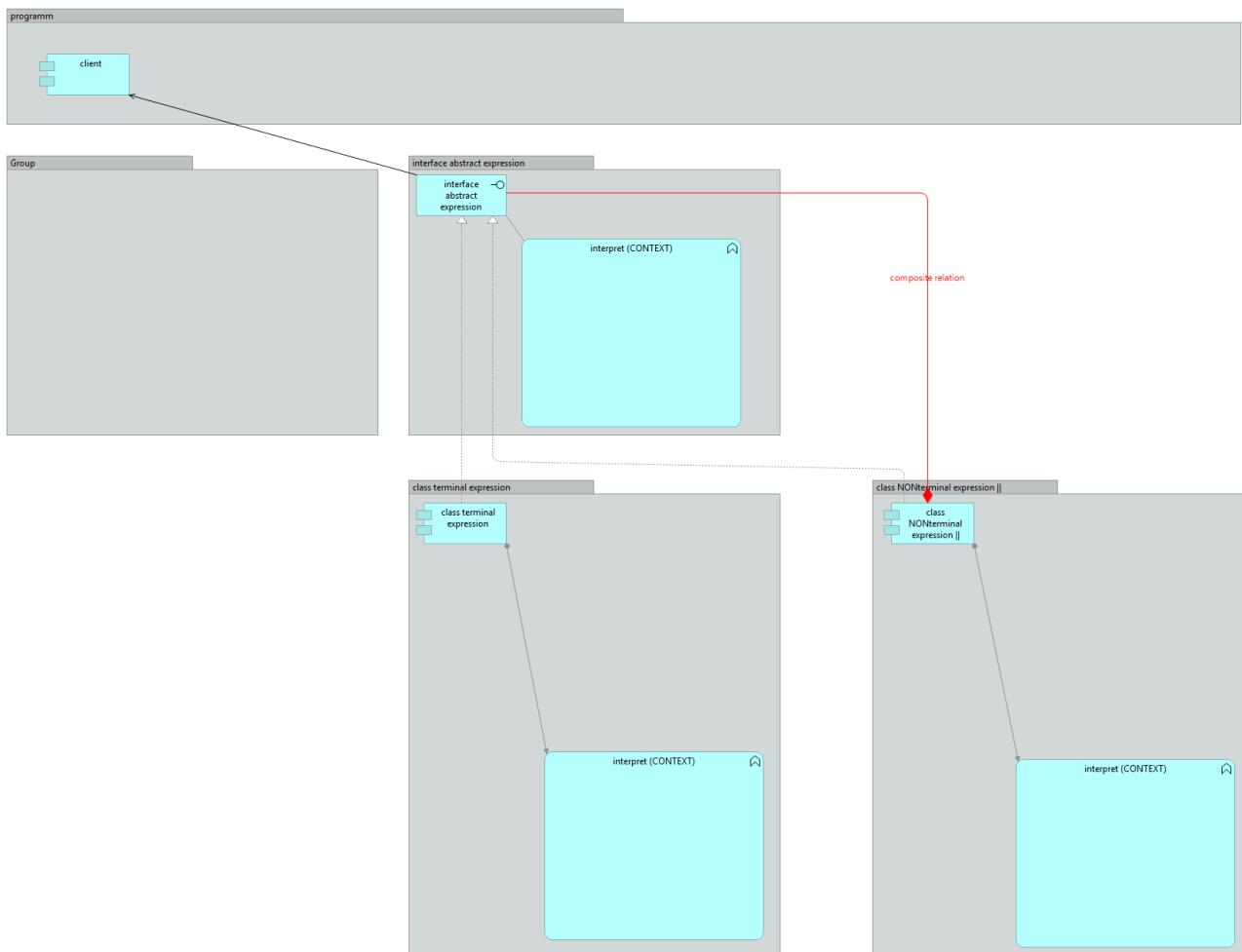
COMMAND



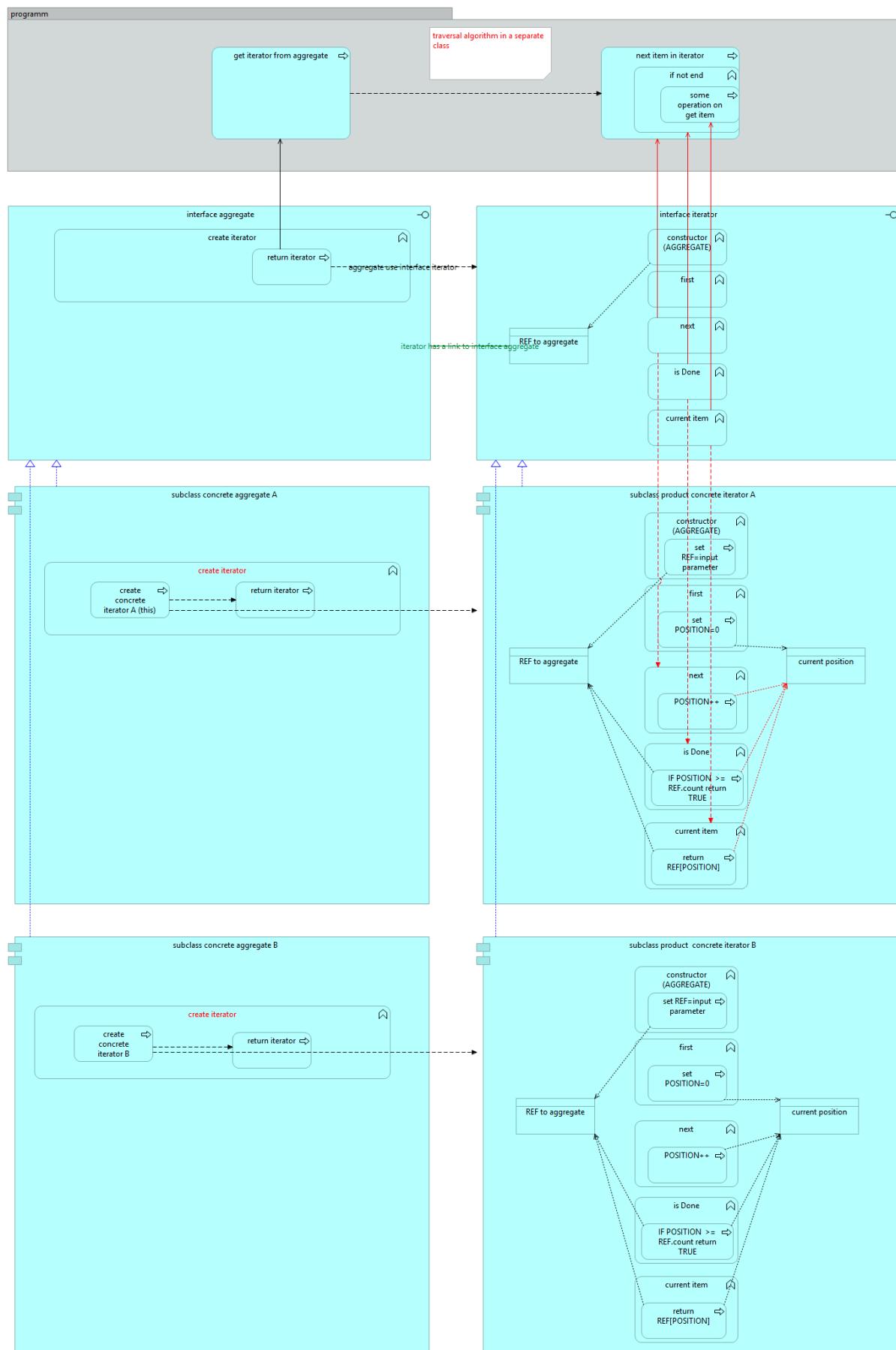


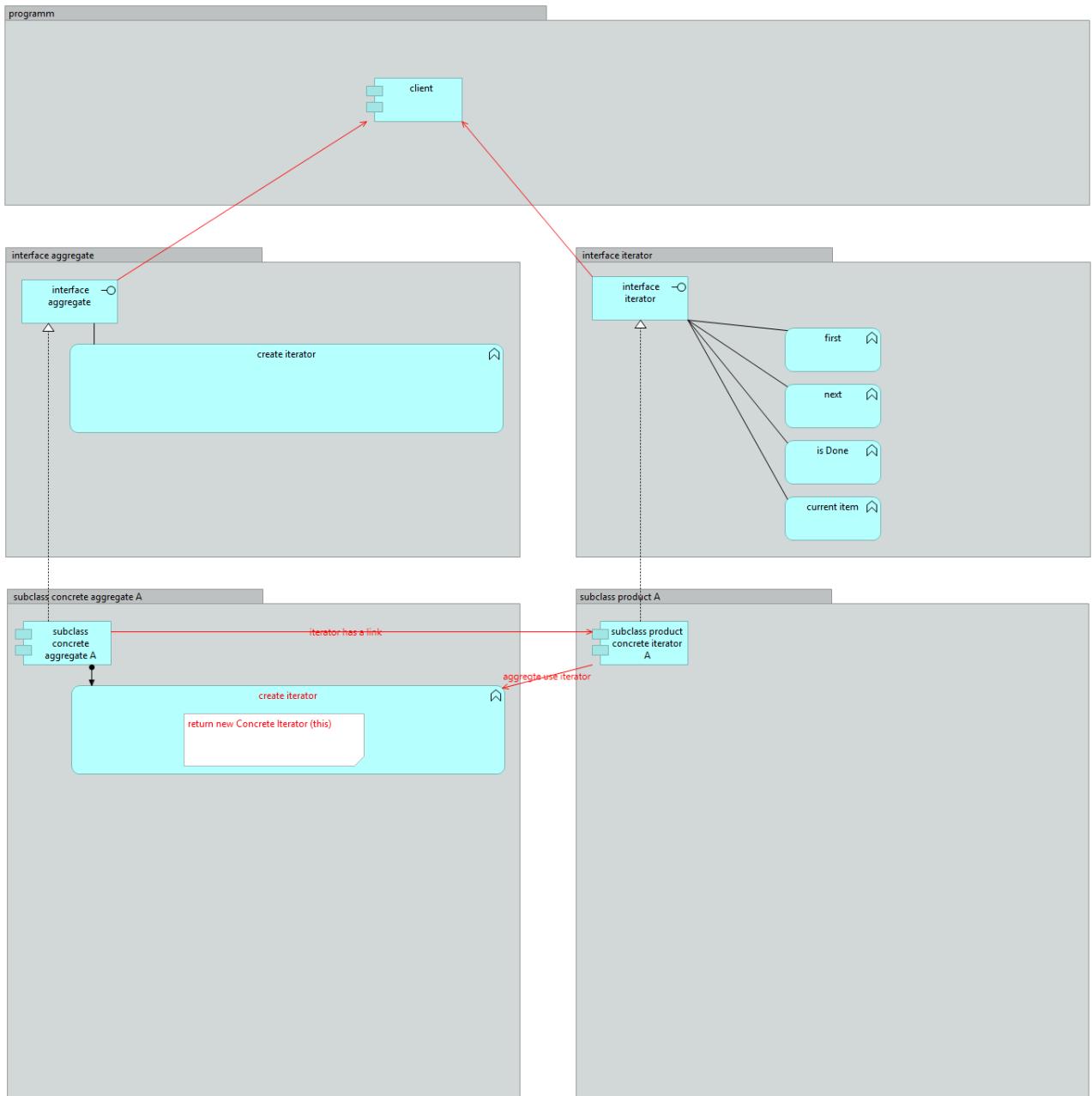
INTERPRETER



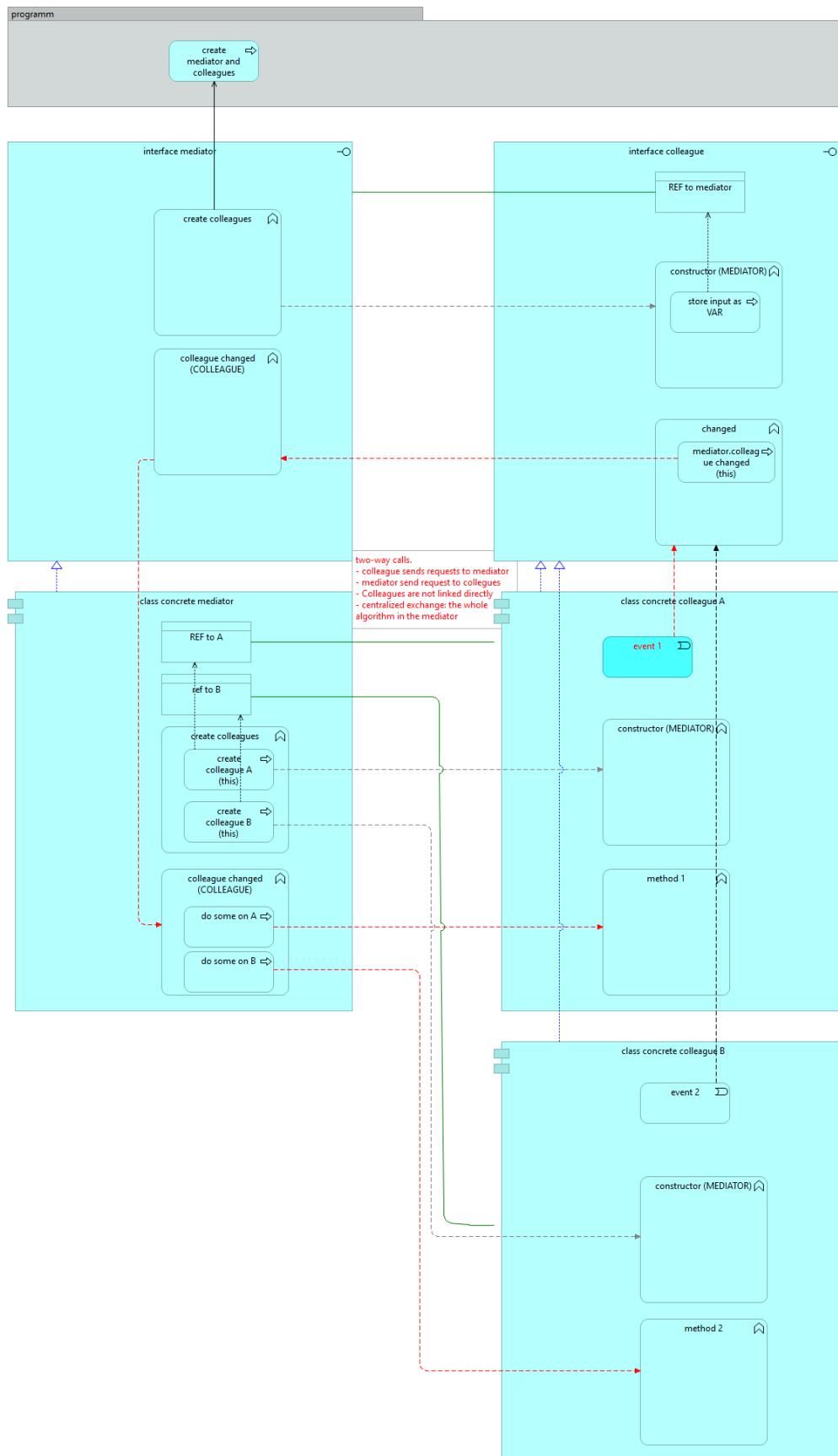


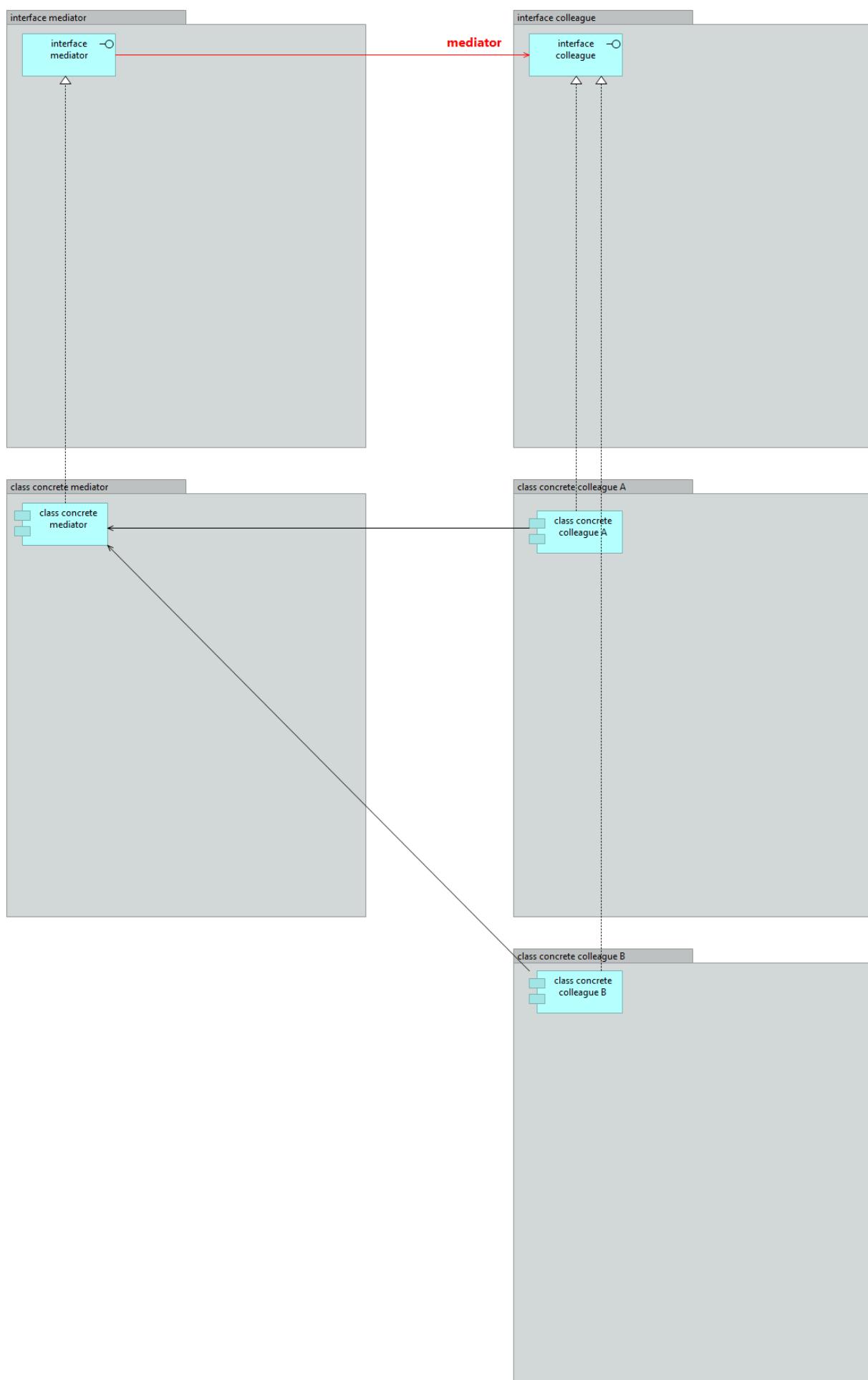
ITERATOR



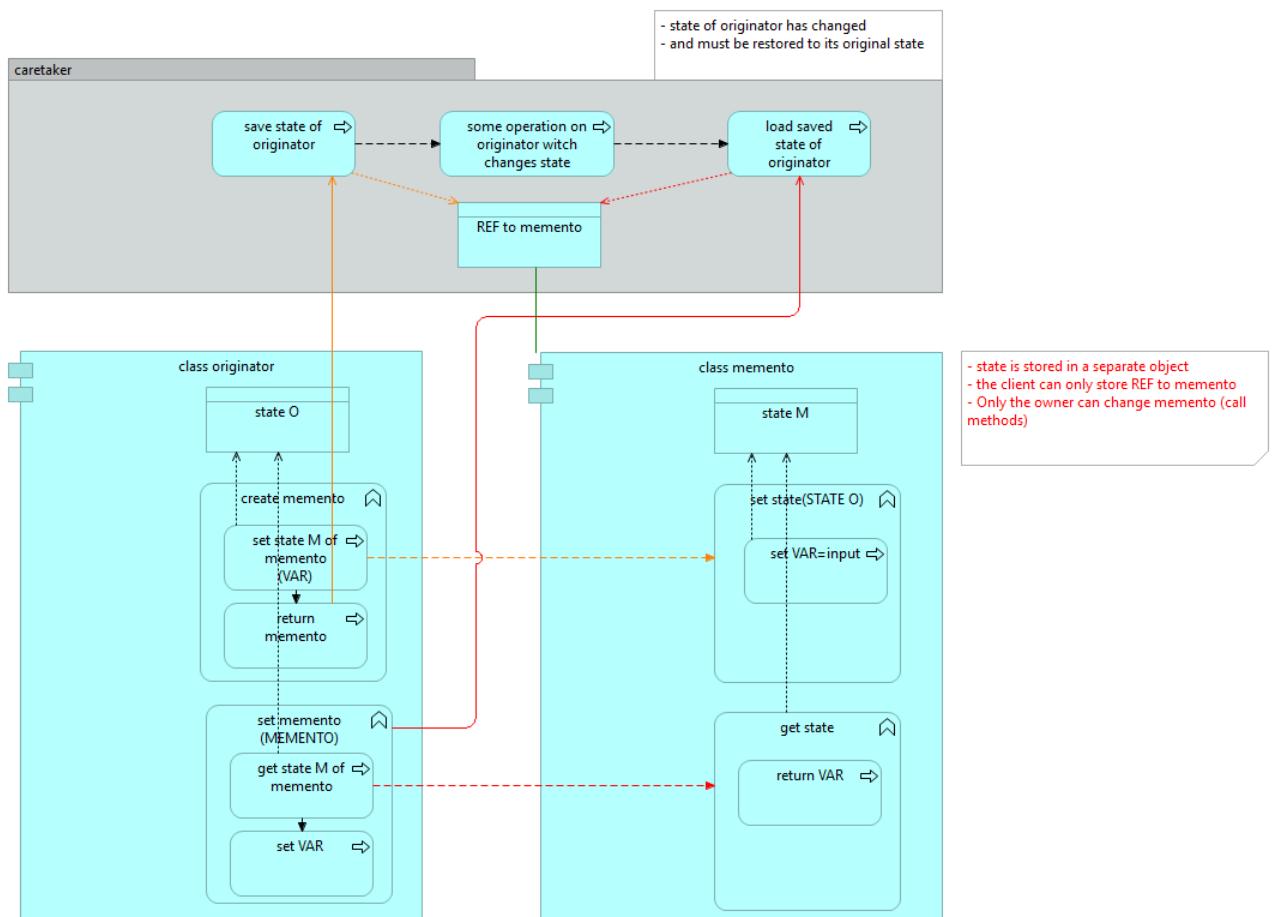


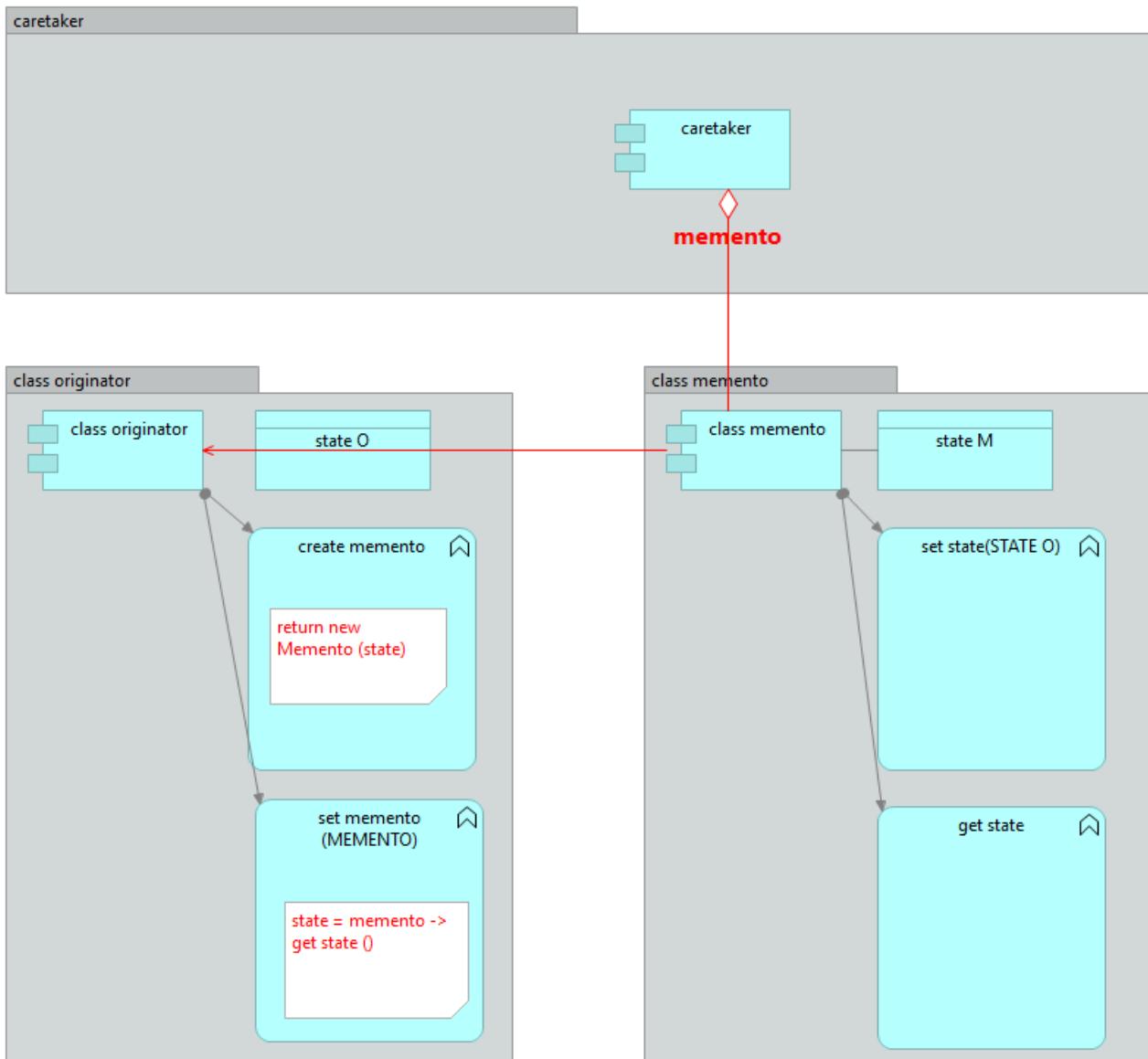
MEDIATOR



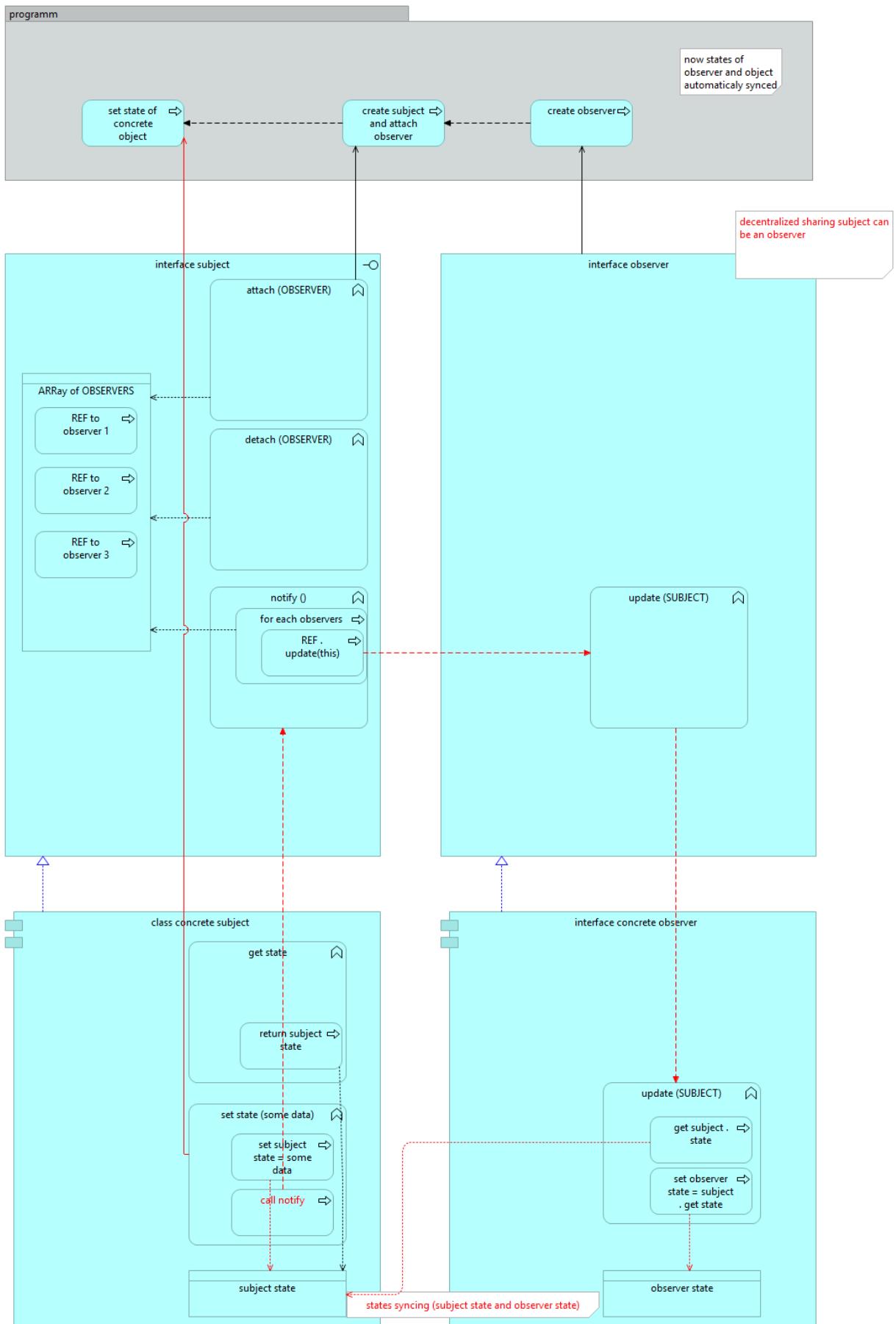


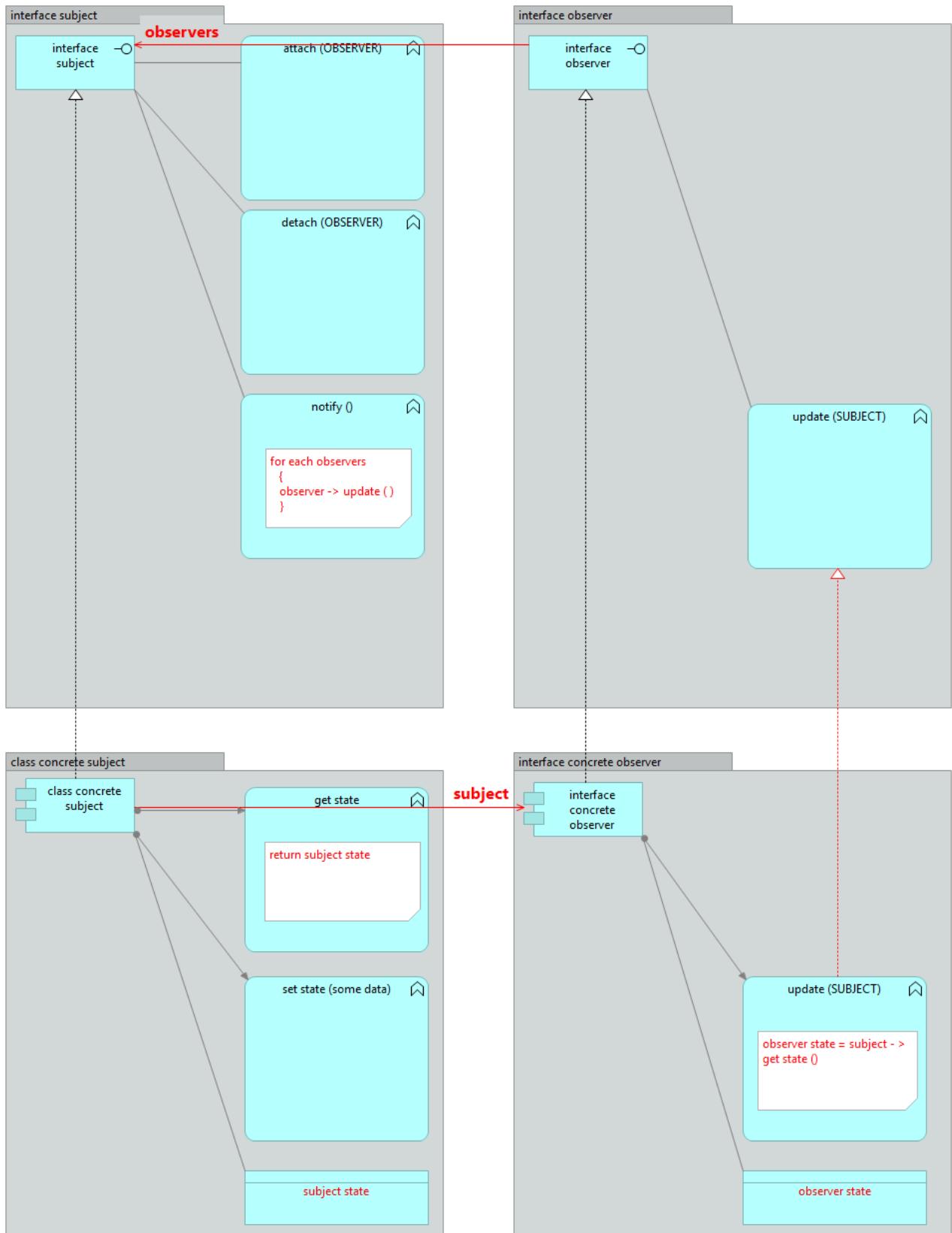
MEMENTO



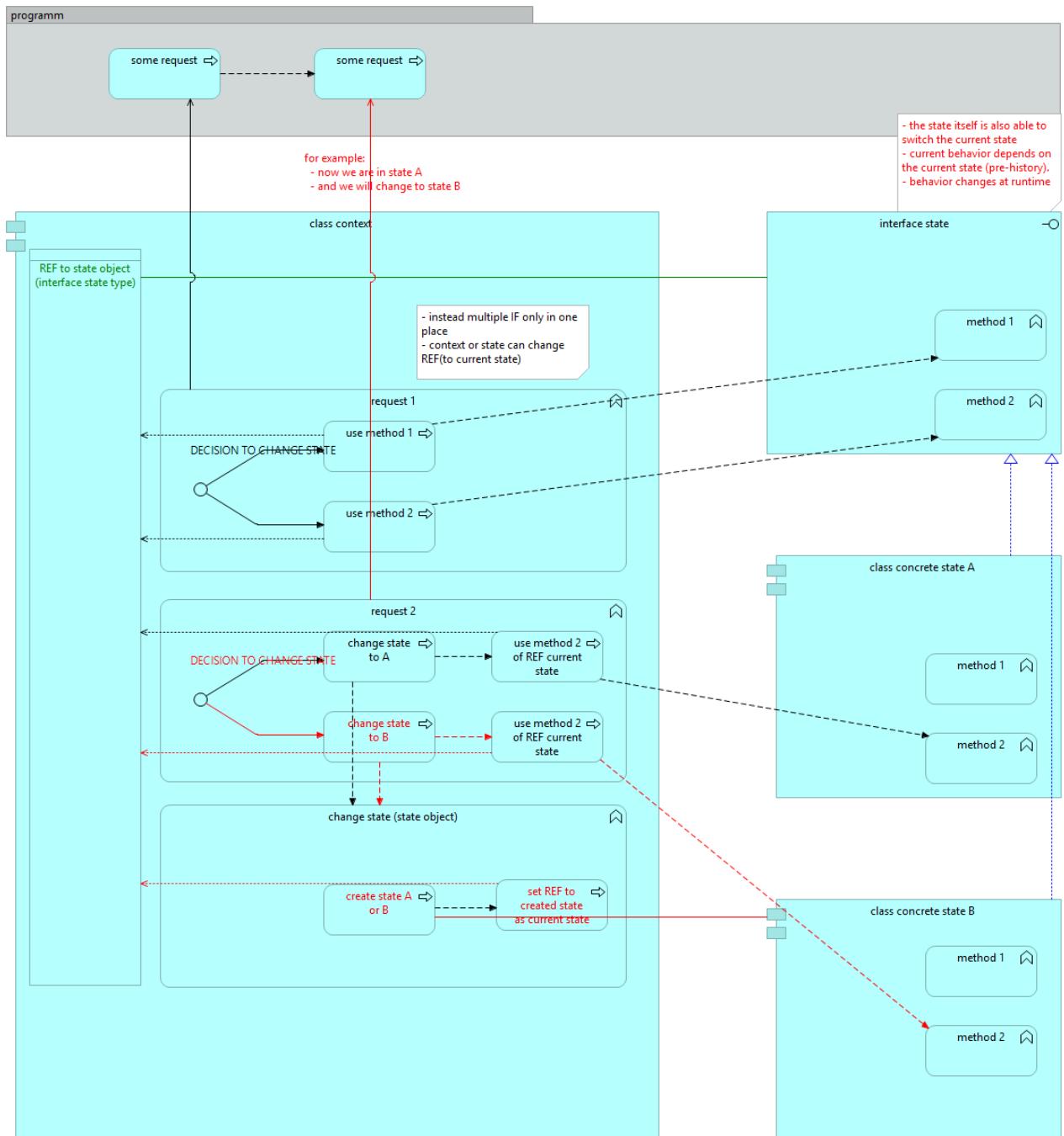


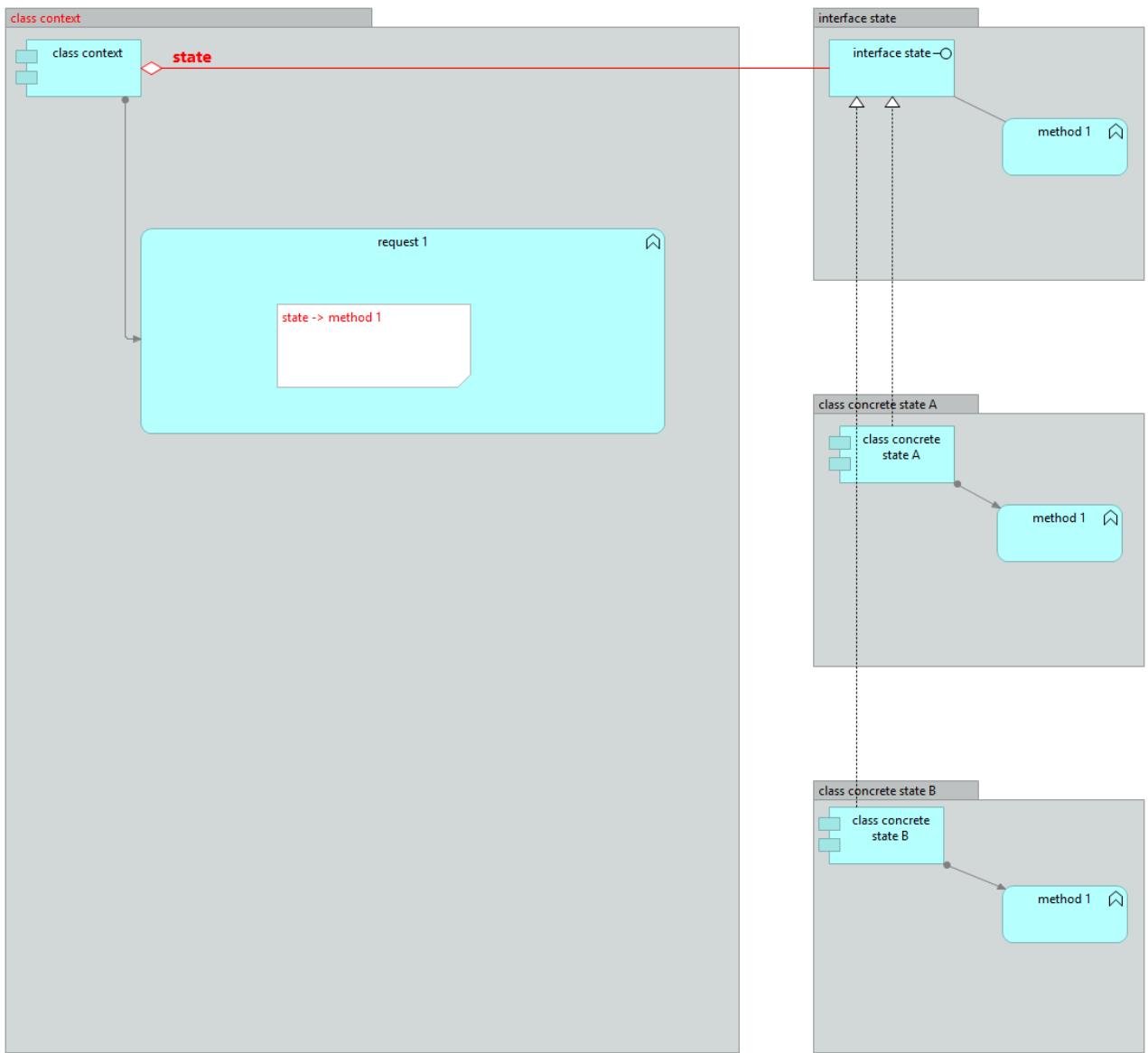
OBSERVER



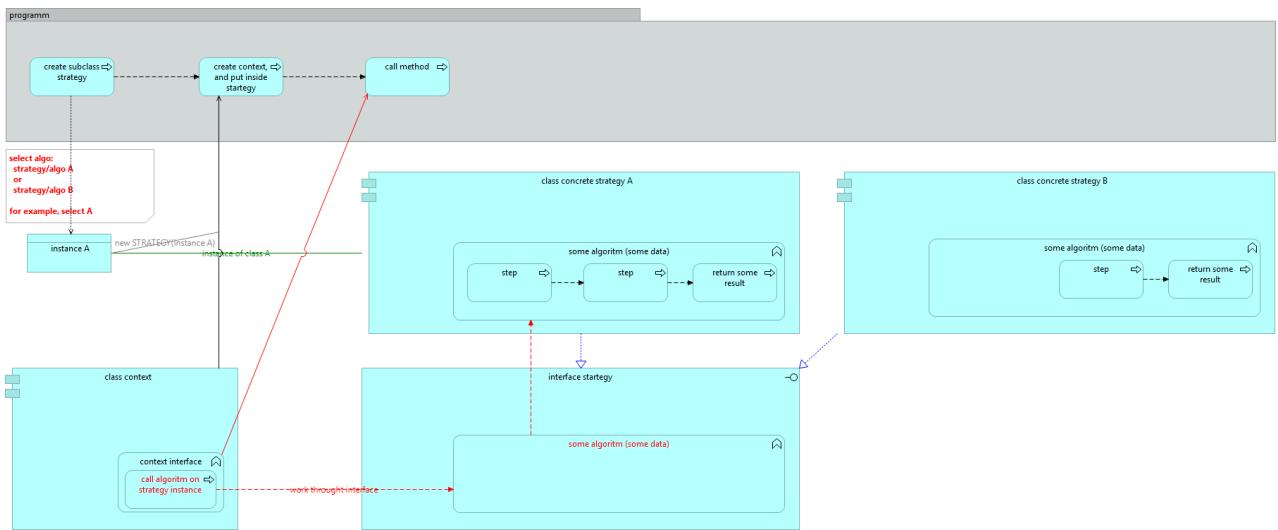


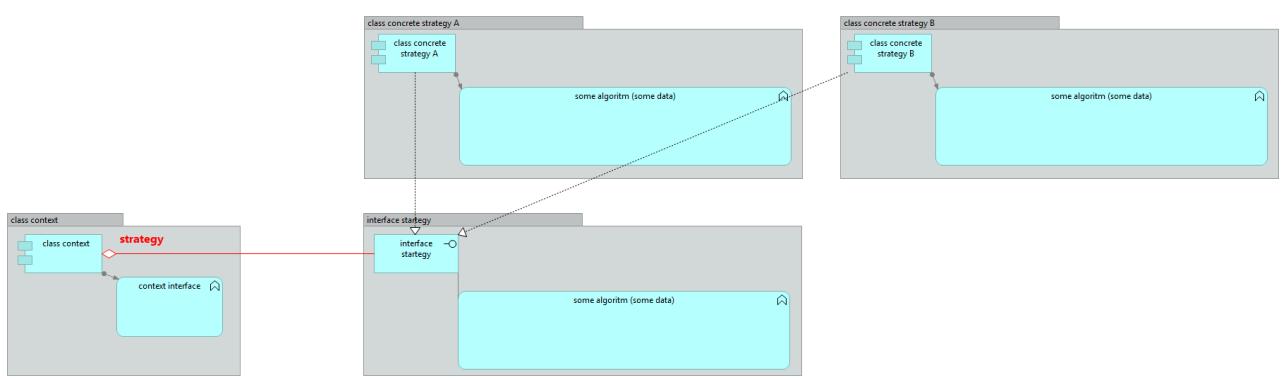
STATE



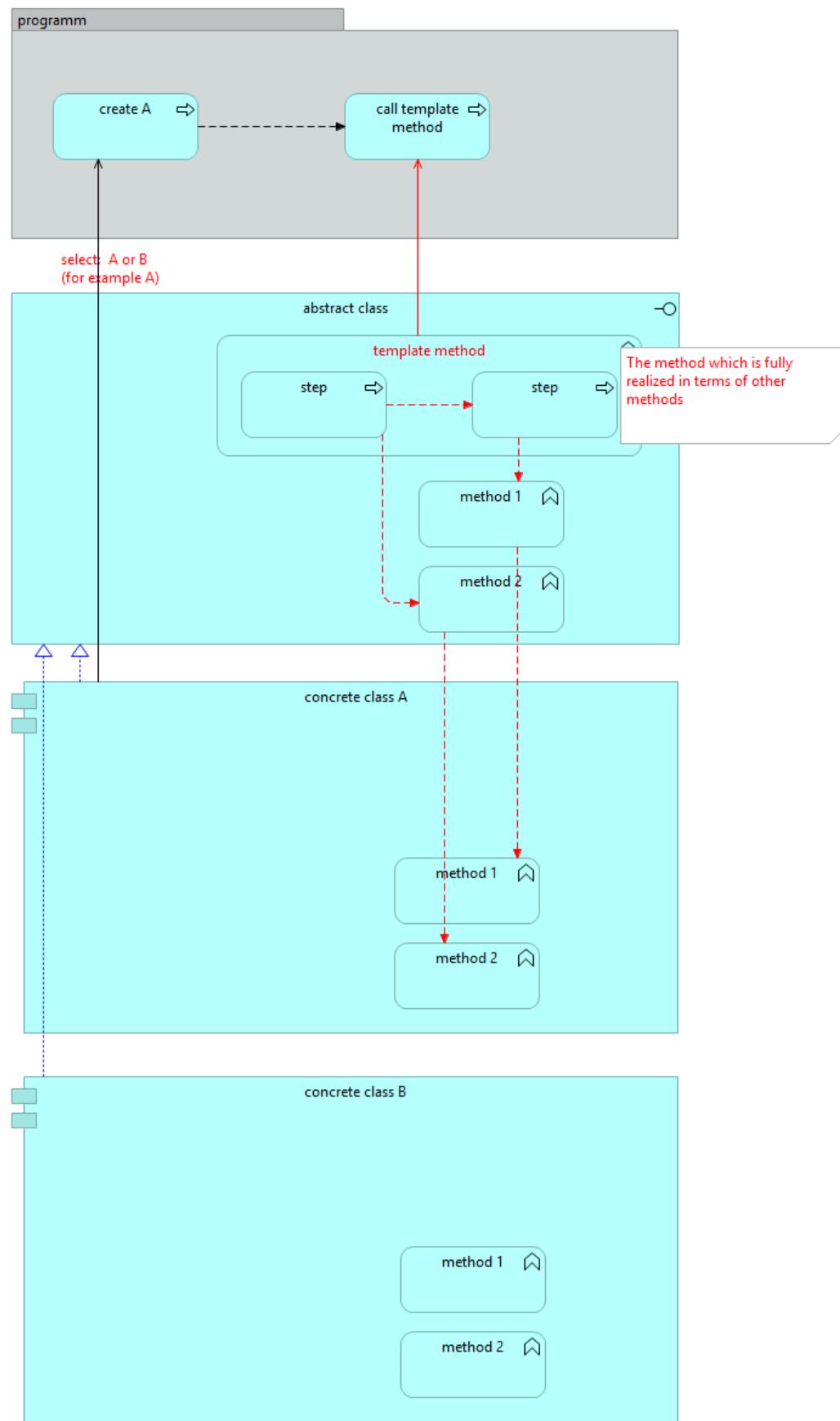


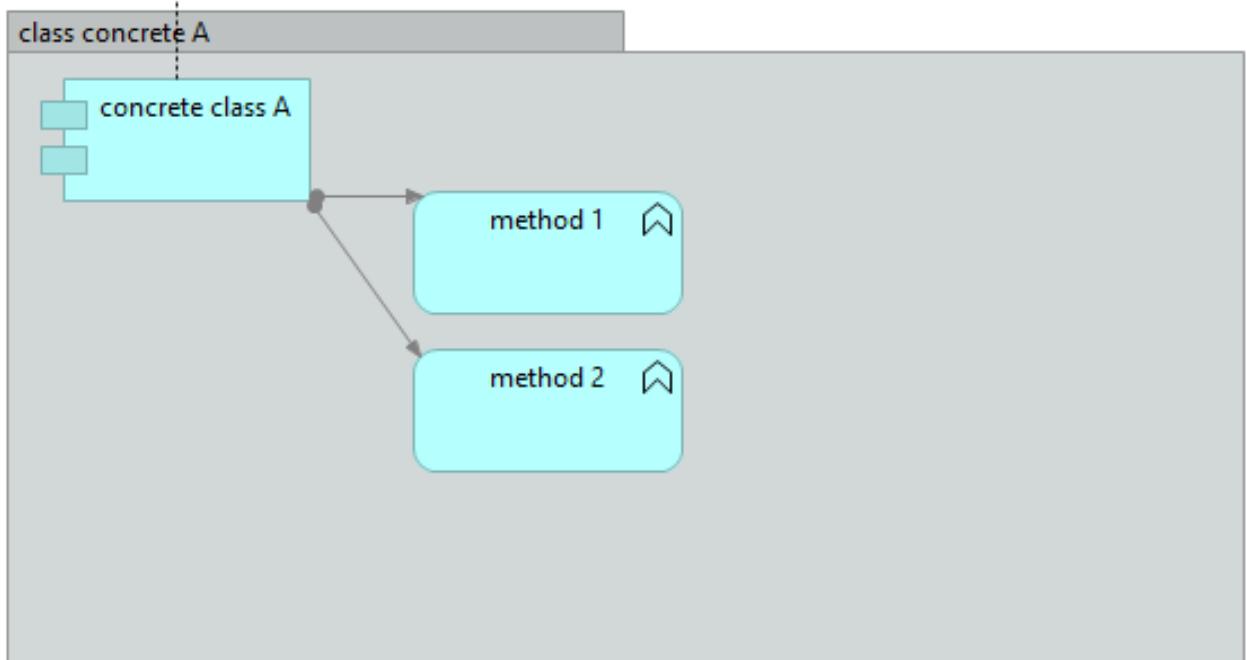
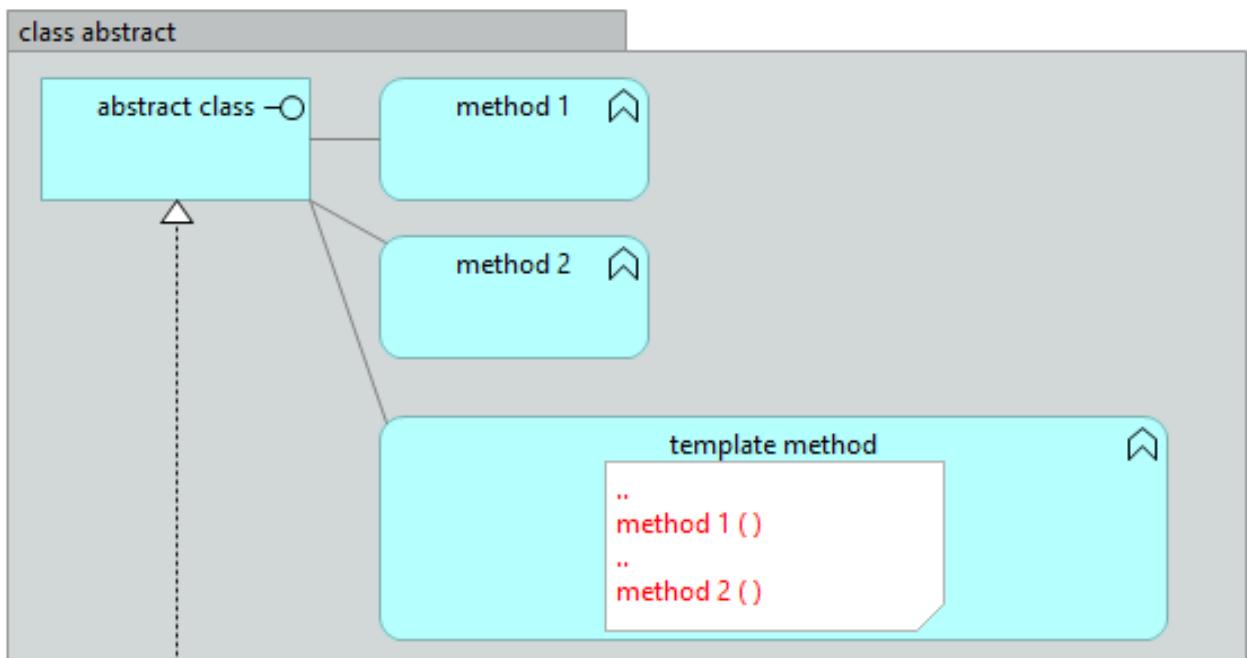
STRATEGY

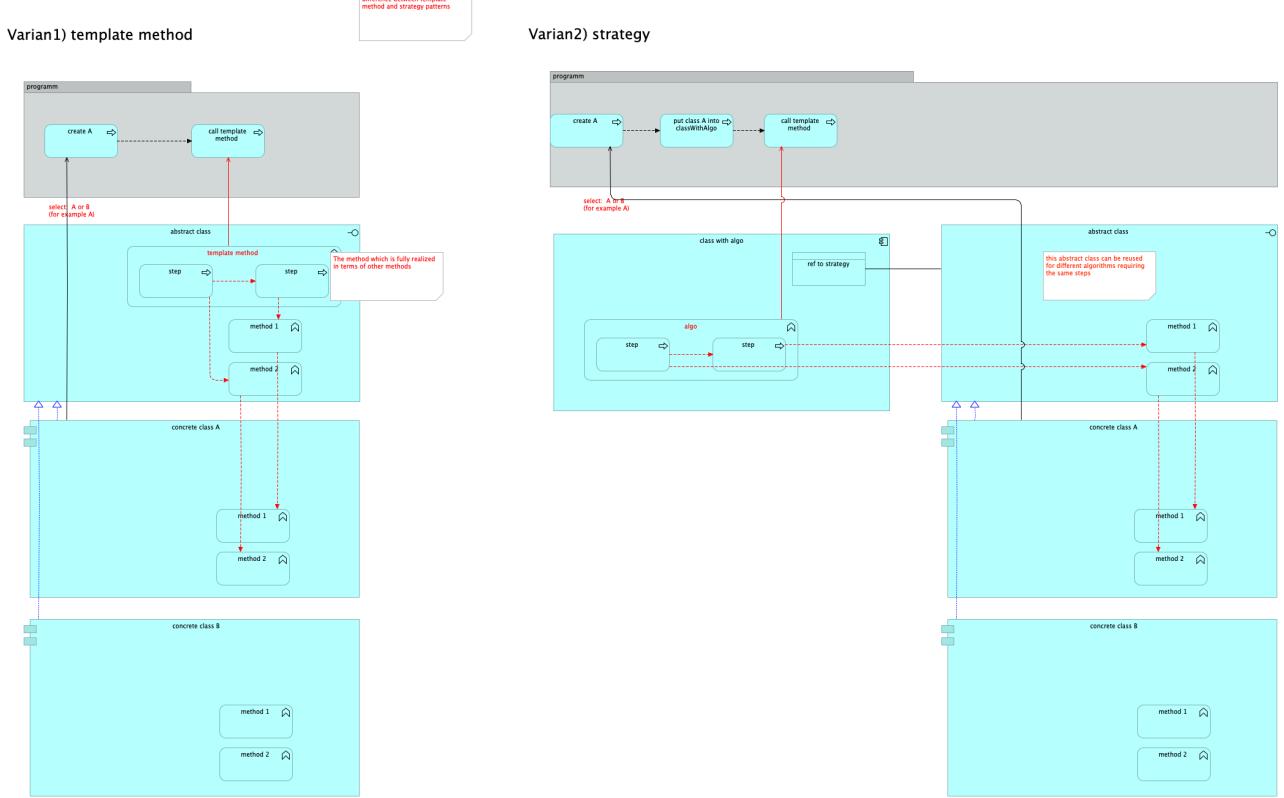




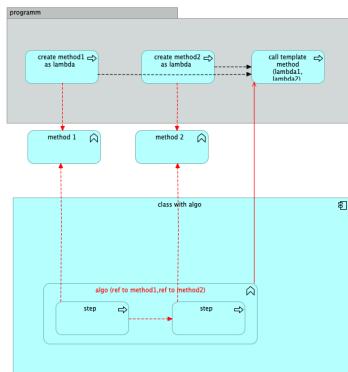
TEMPLATE METHOD



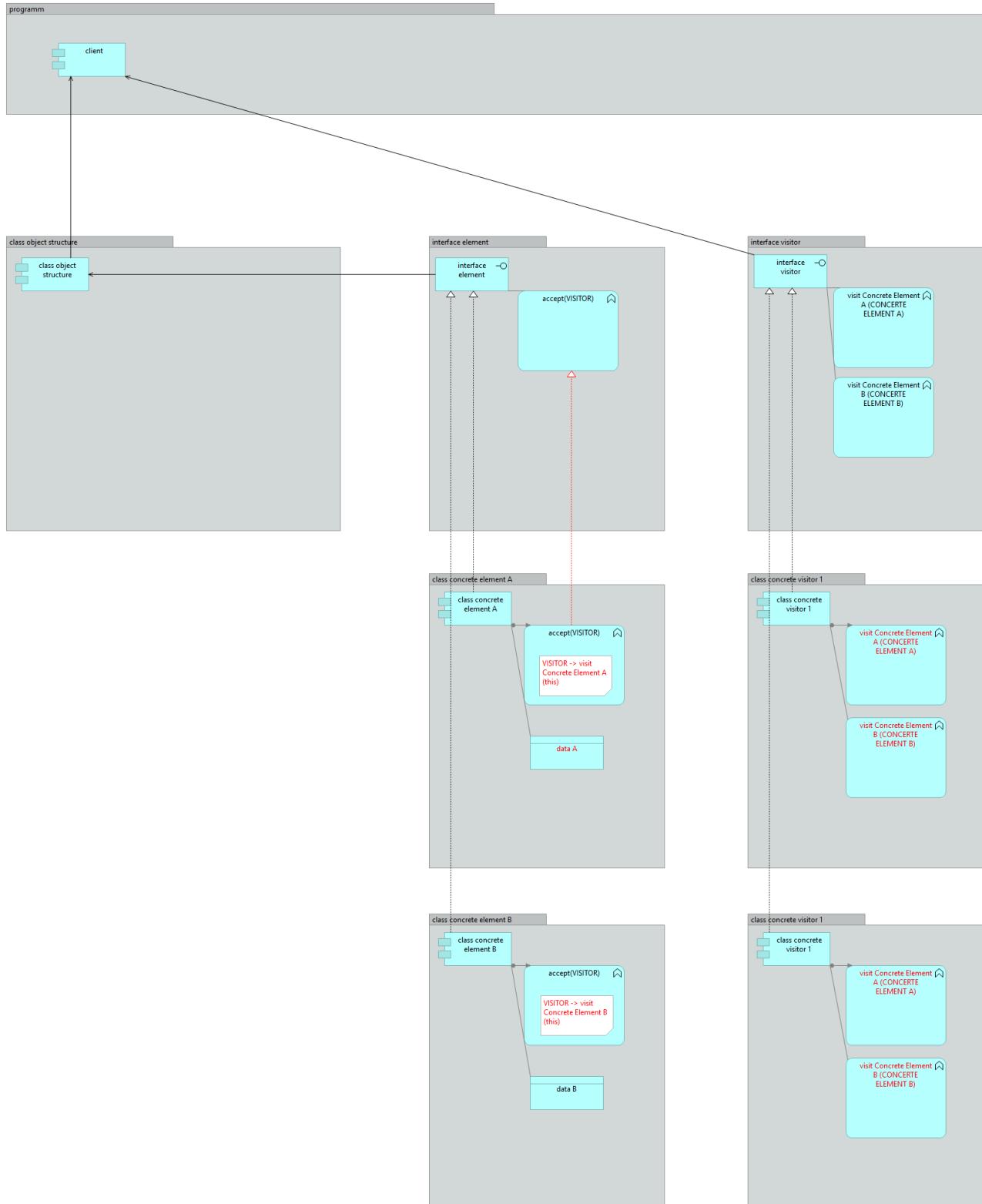


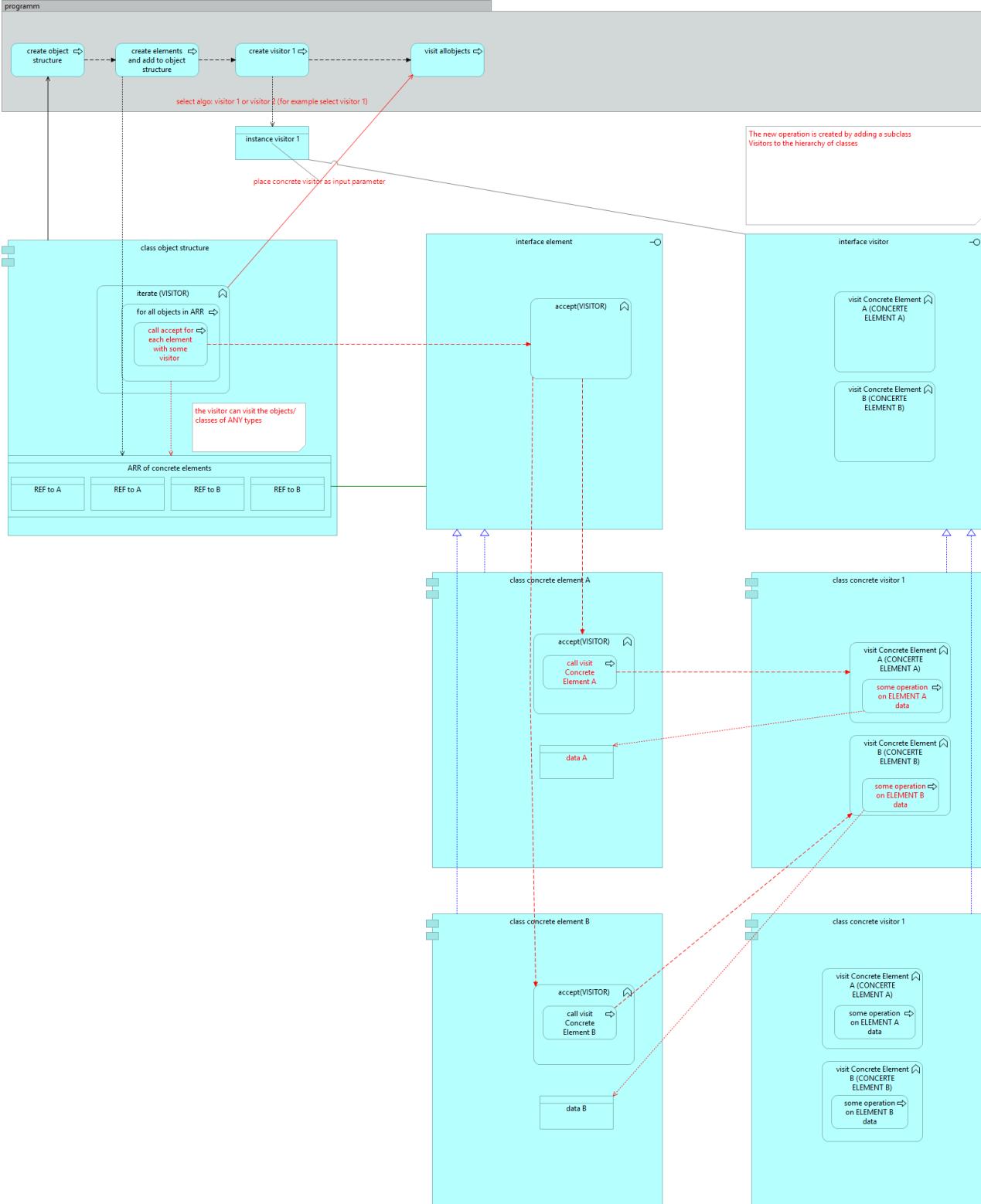


Varian3) lambdas as input parameters



VISITOR





02

Enterprise Patterns

Catalog of Patterns of Enterprise
Application Architecture

MARTIN FOWLER



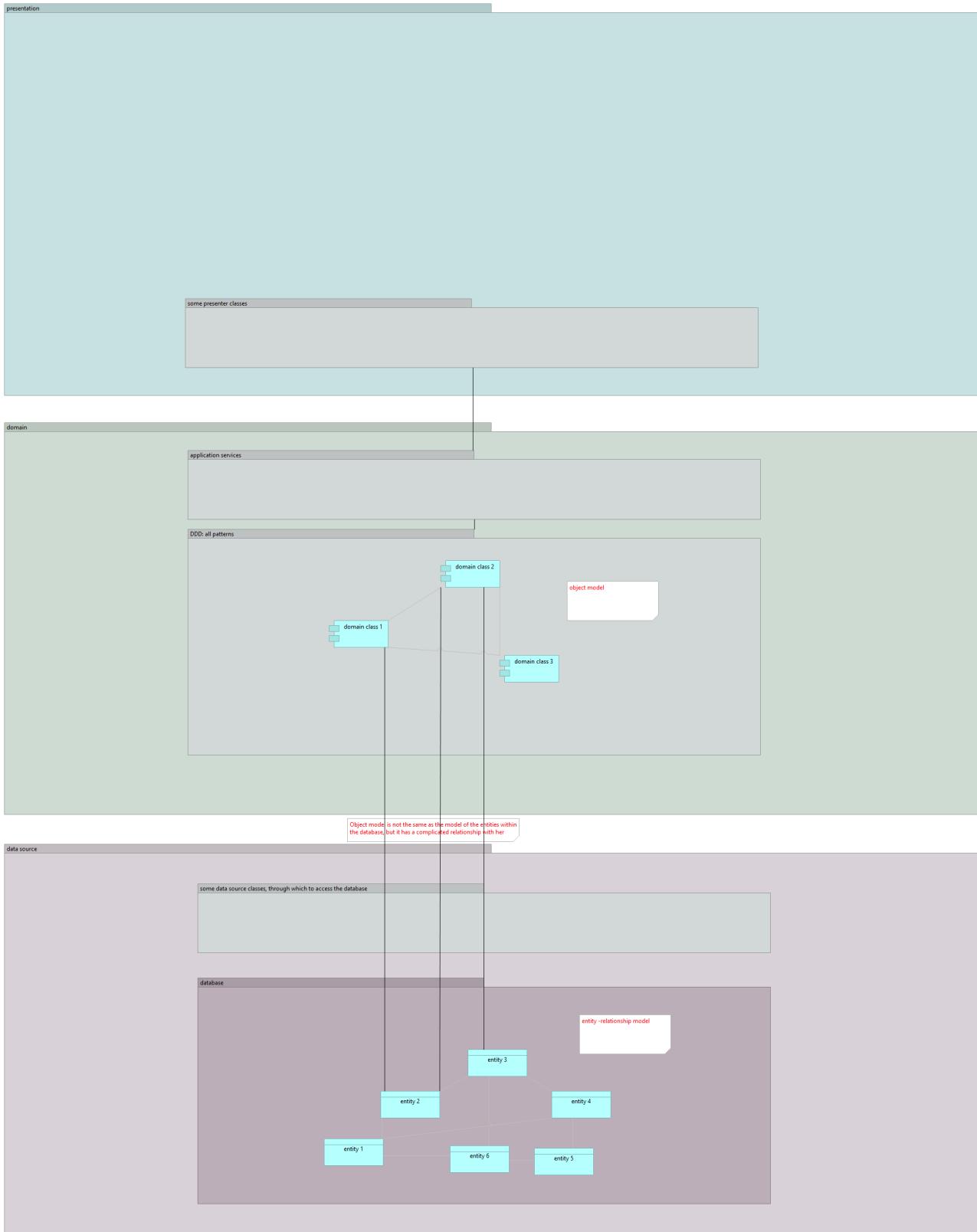
BUSINESS LOGIC

ENTERPRISE PATTERNS

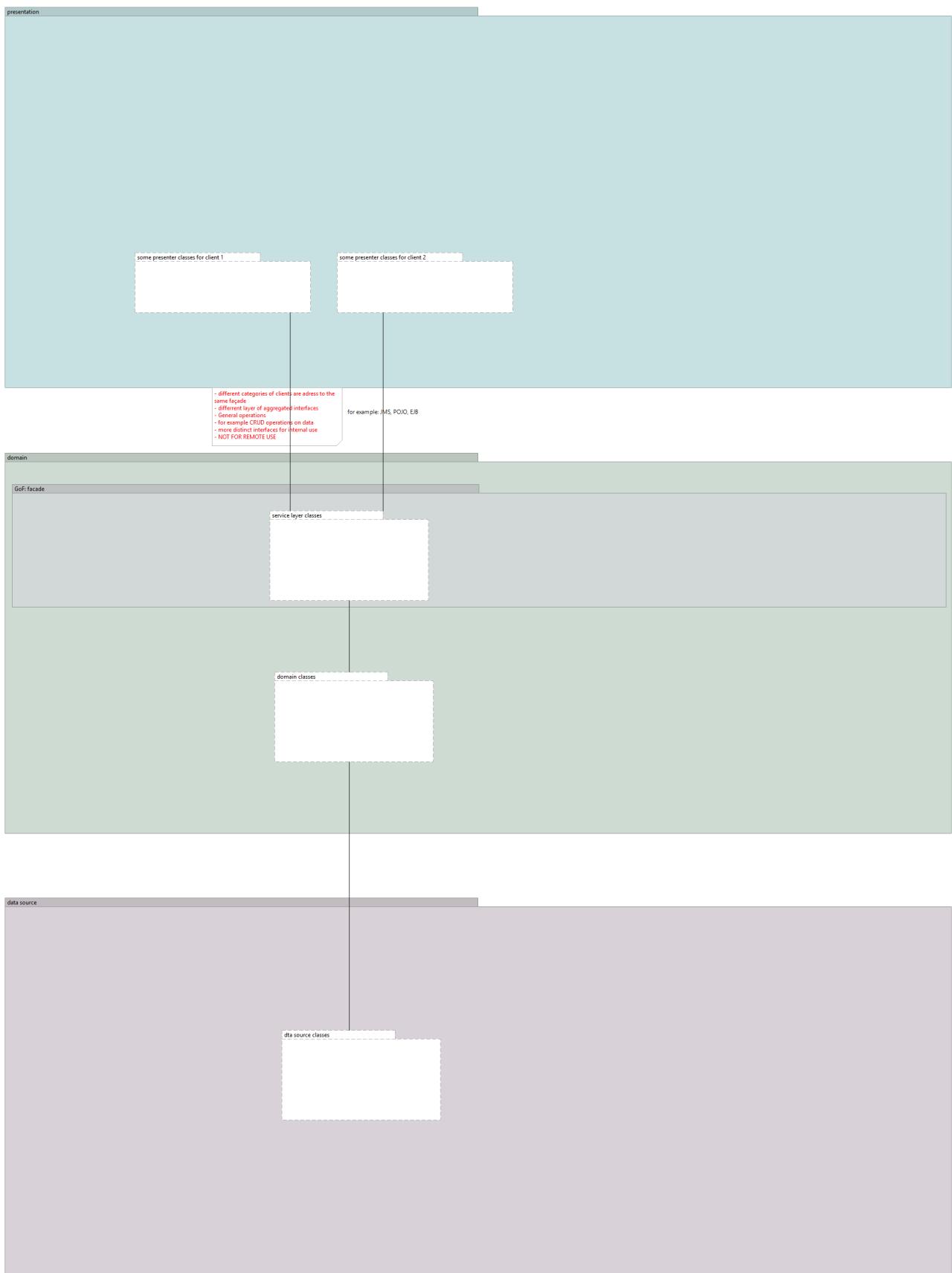
Martin Fowler



DOMAIN MODEL



SERVICE LAYER



TRANSACTION SCRIPT

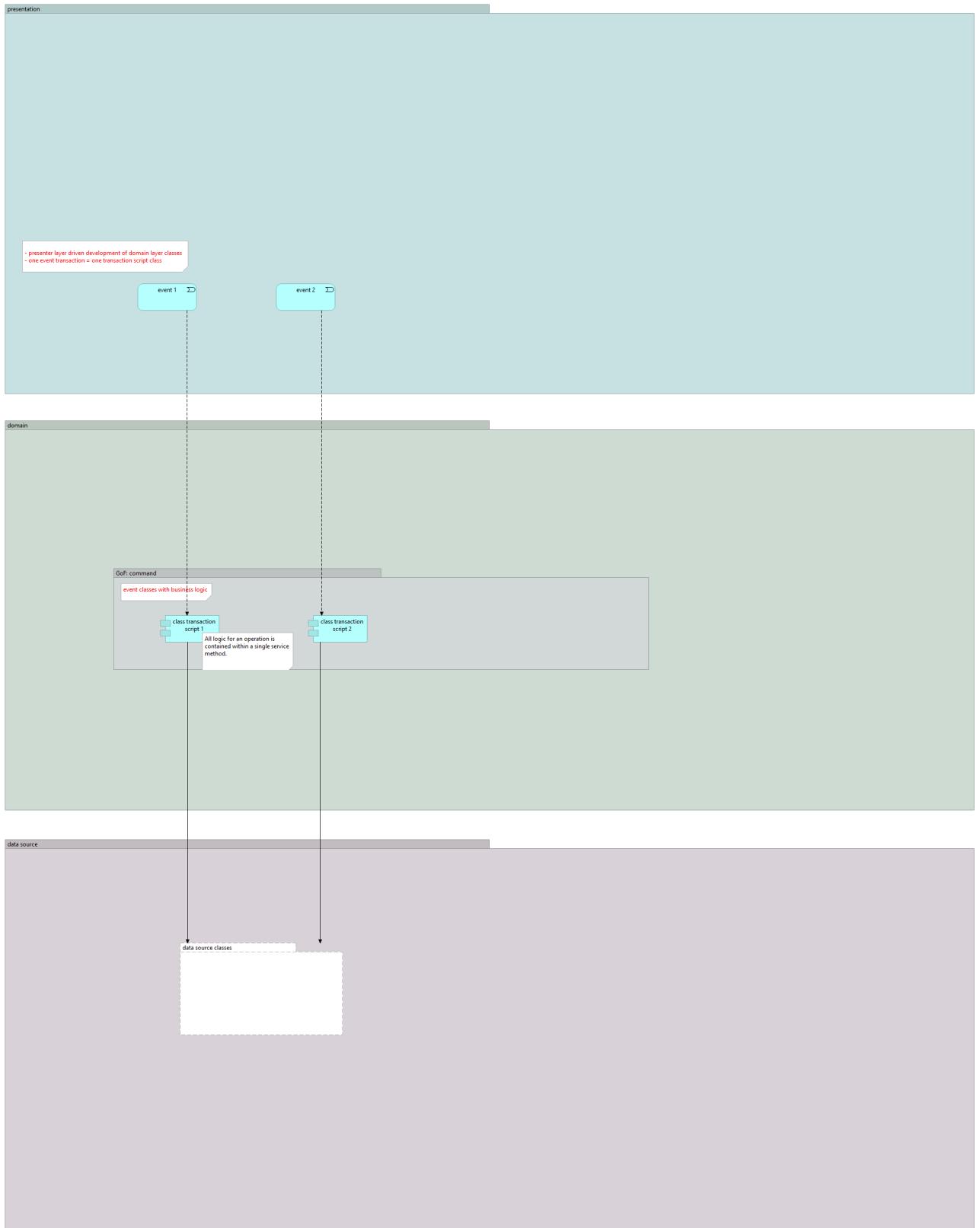
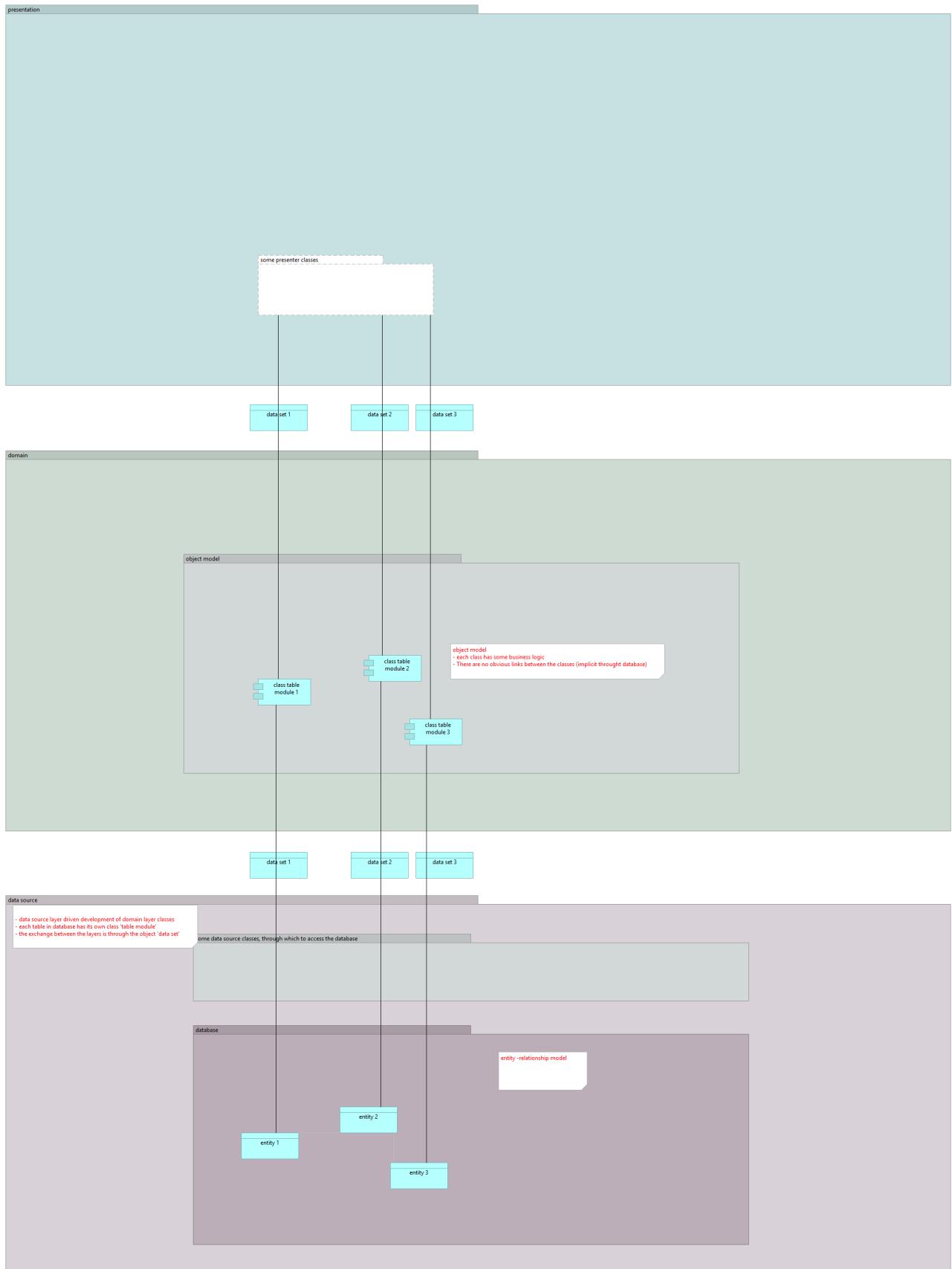


TABLE MODULE



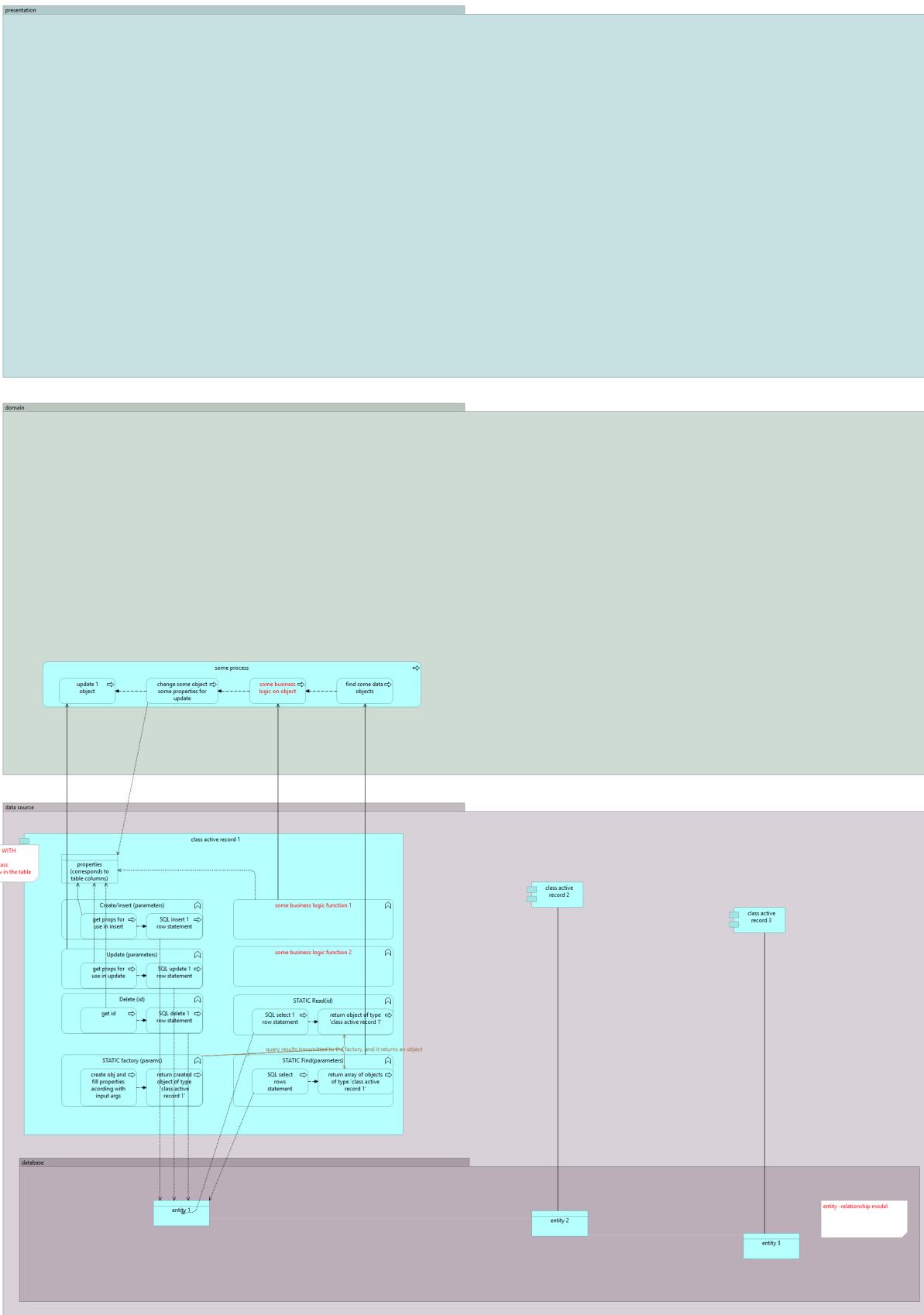
DATA SOURCES

ENTERPRISE PATTERNS

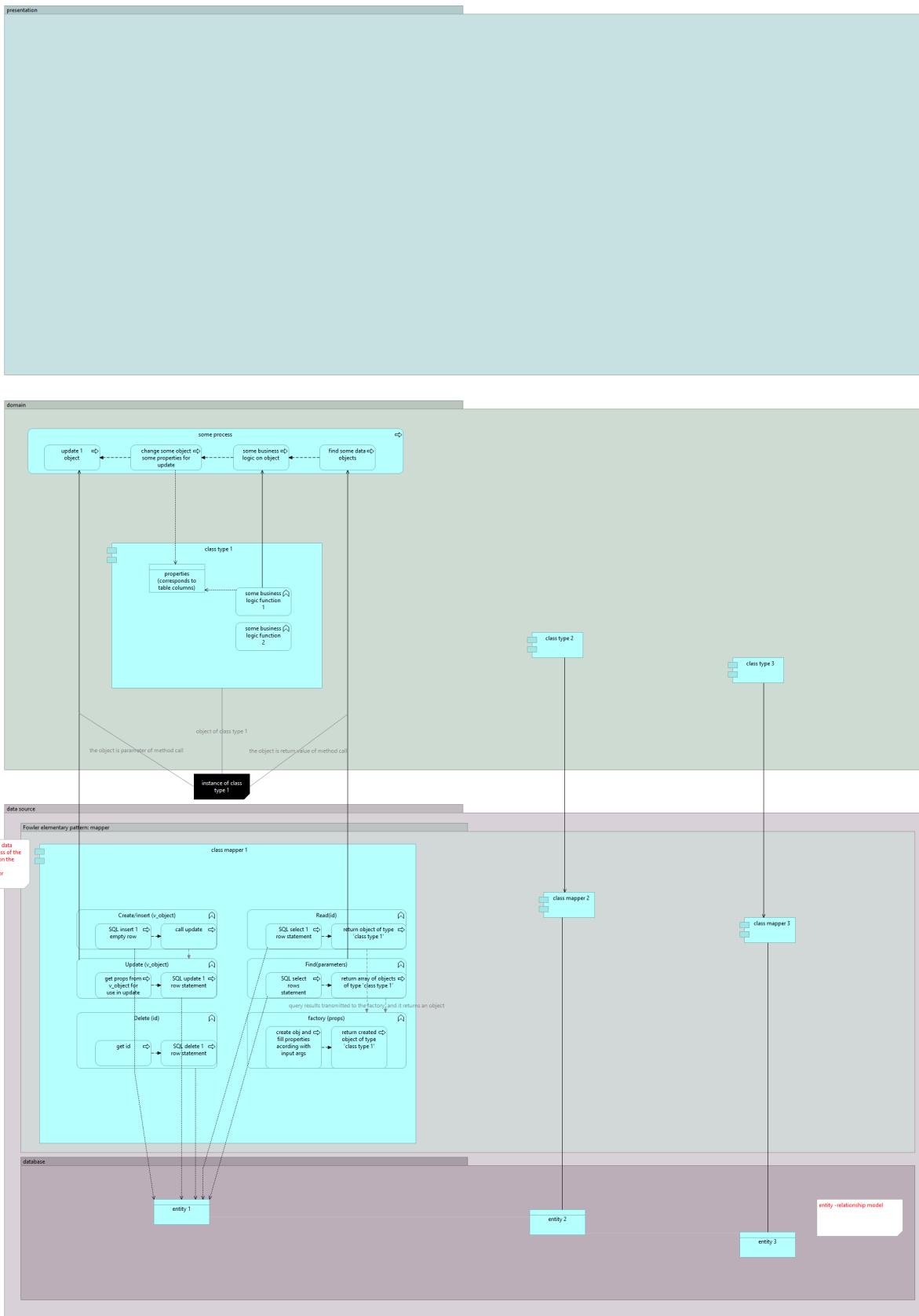
Martin Fowler



ACTIVE RECORD



DATA MAPPER



ROW DATA GATEWAY

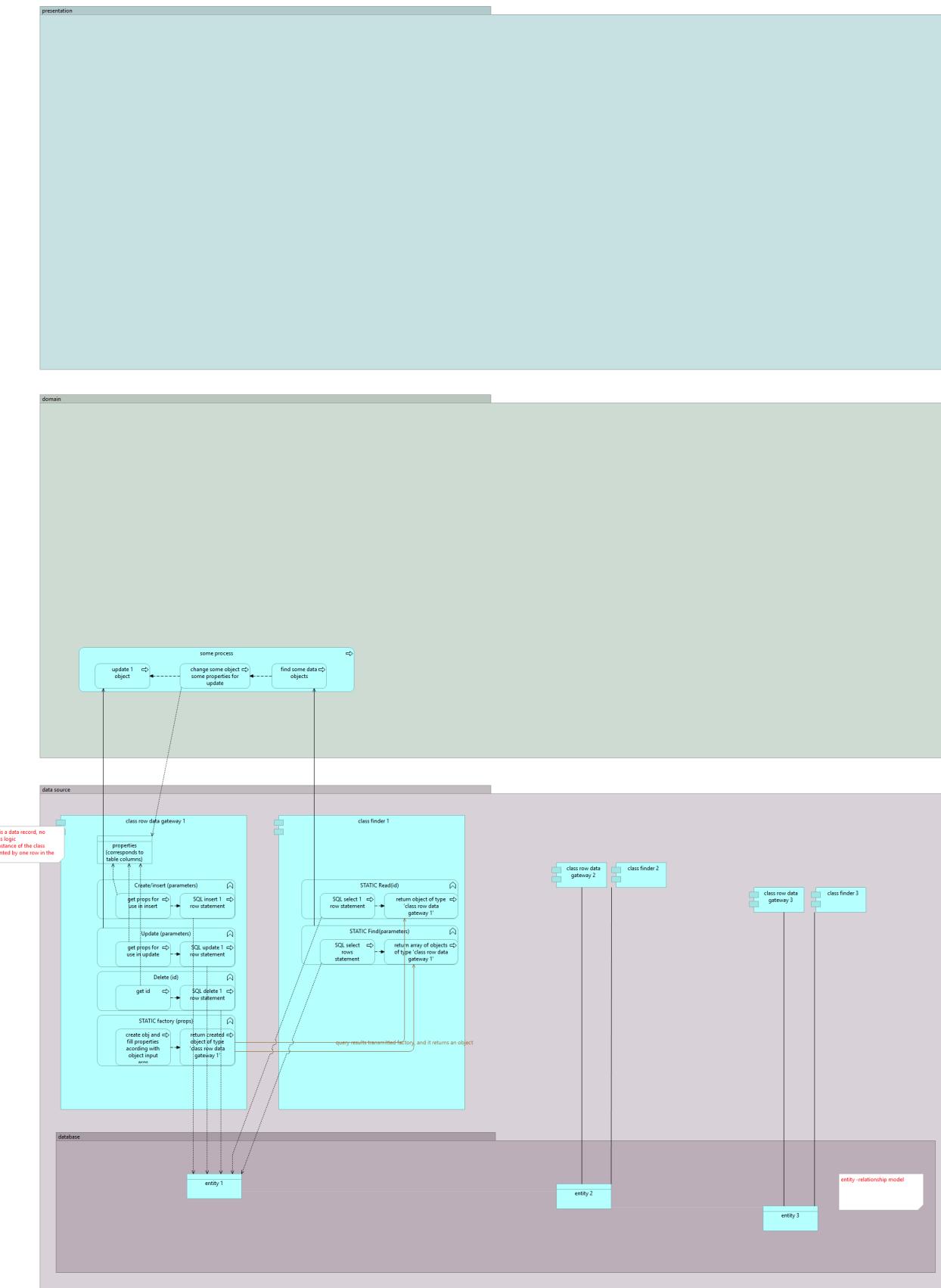
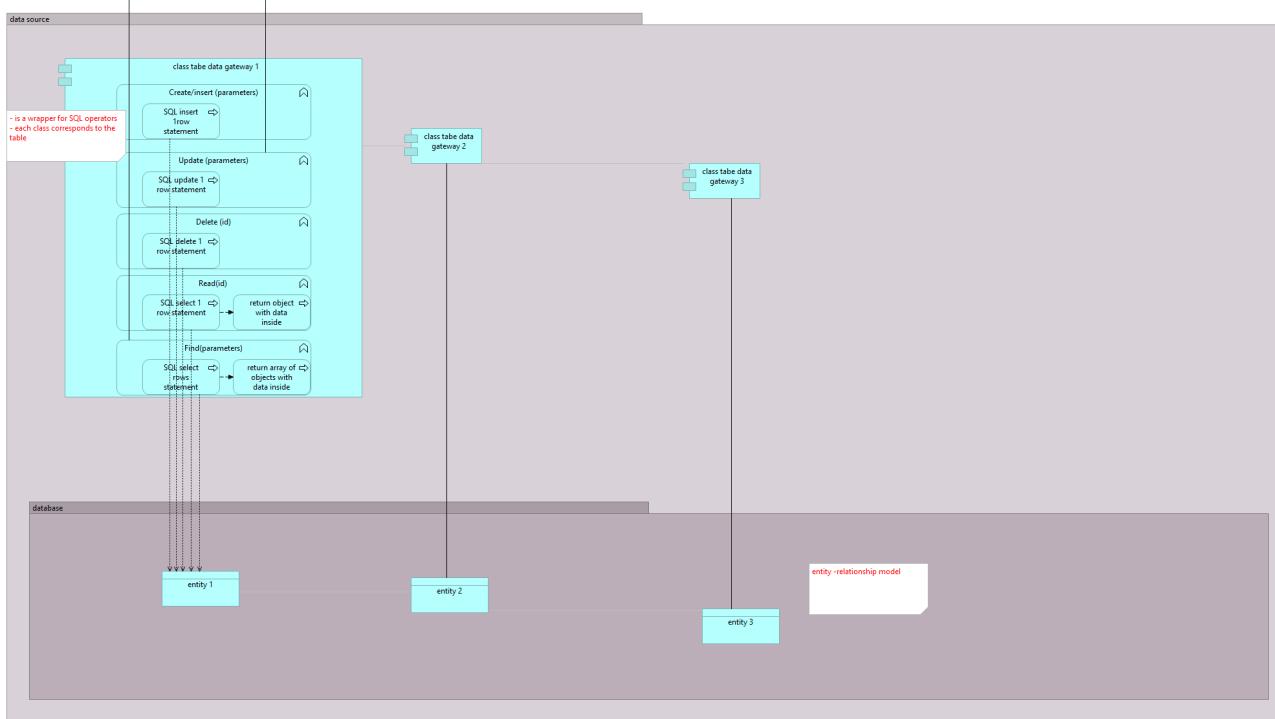
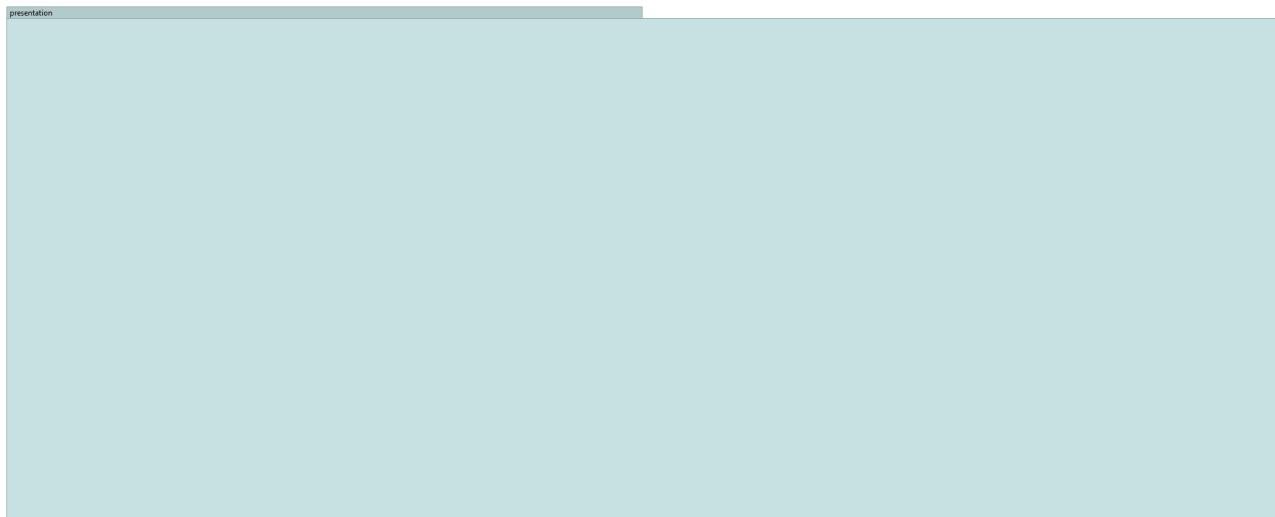


TABLE DATA GATEWAY



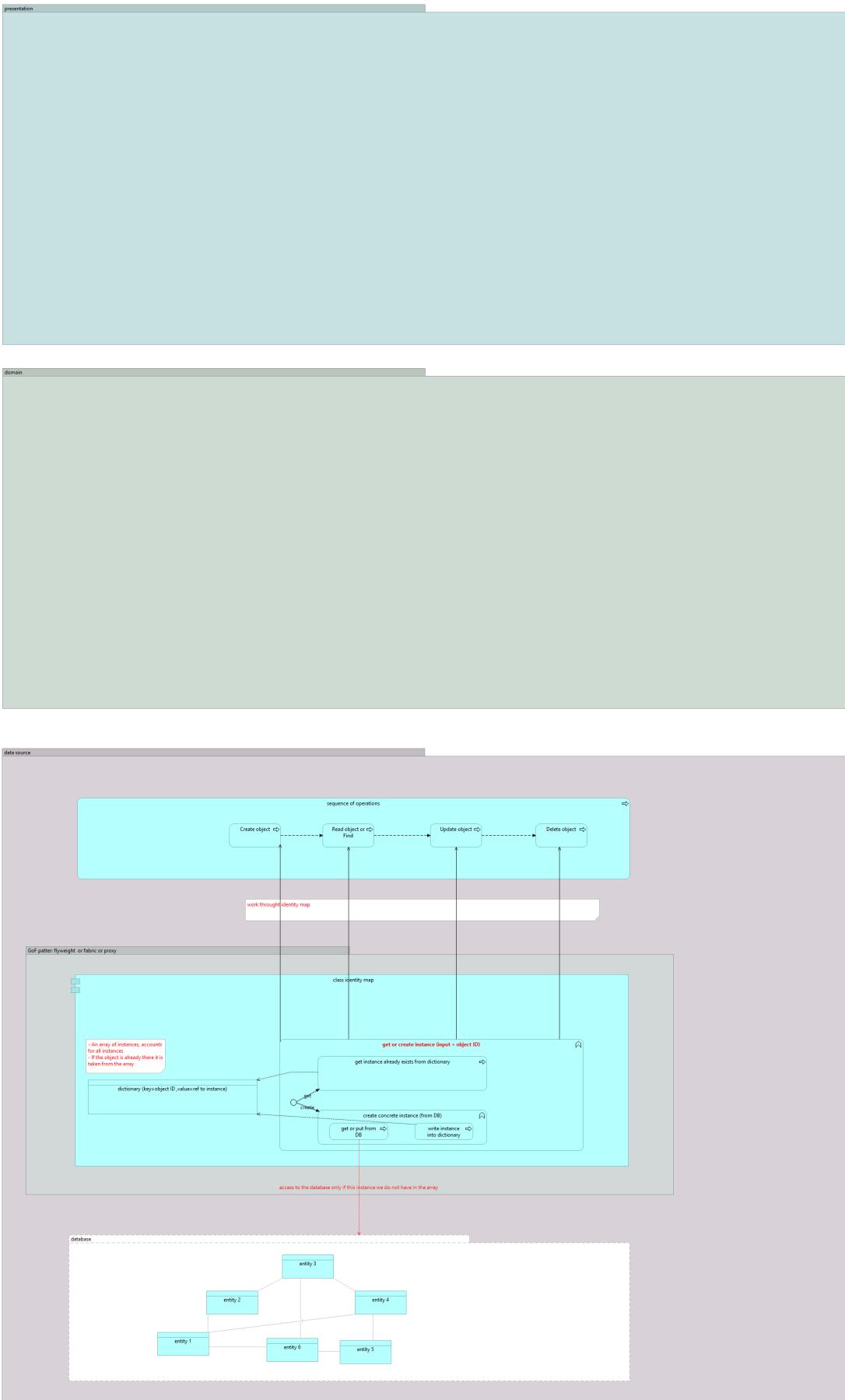
MODELING BEHAVIOR

ENTERPRISE PATTERNS

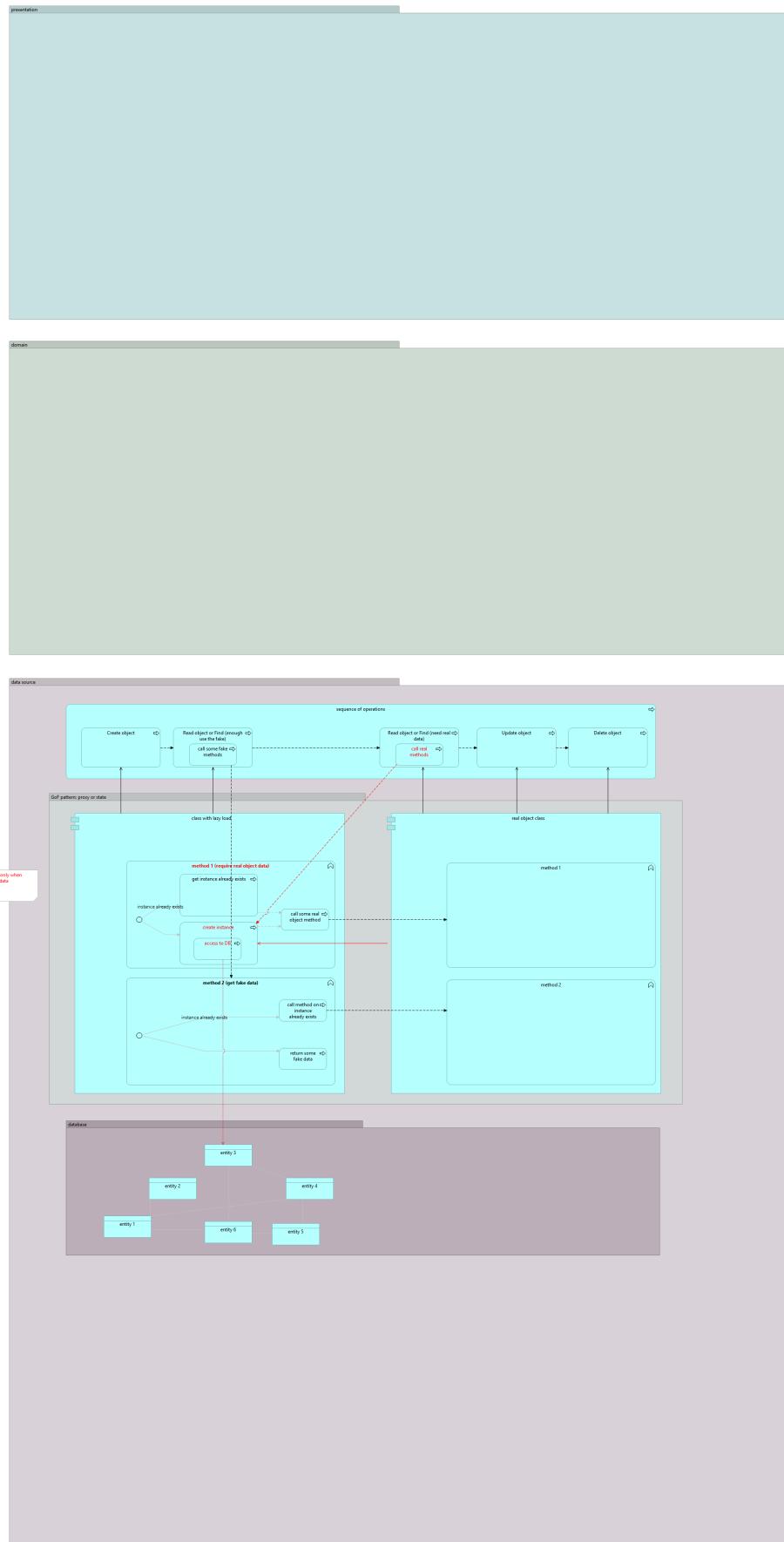
Martin Fowler



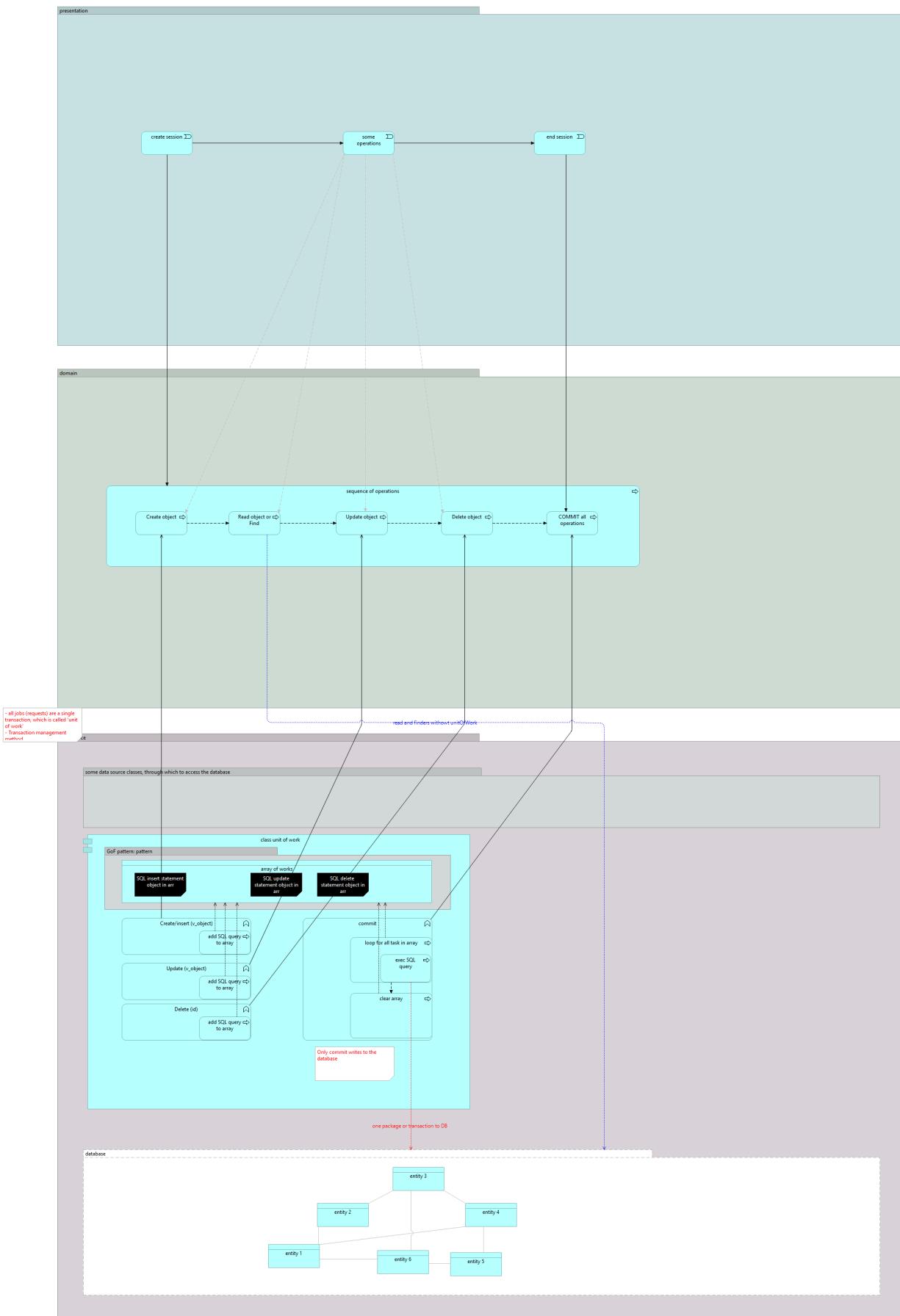
IDENTITY MAP



LAZY LOAD



UNIT OF WORK.



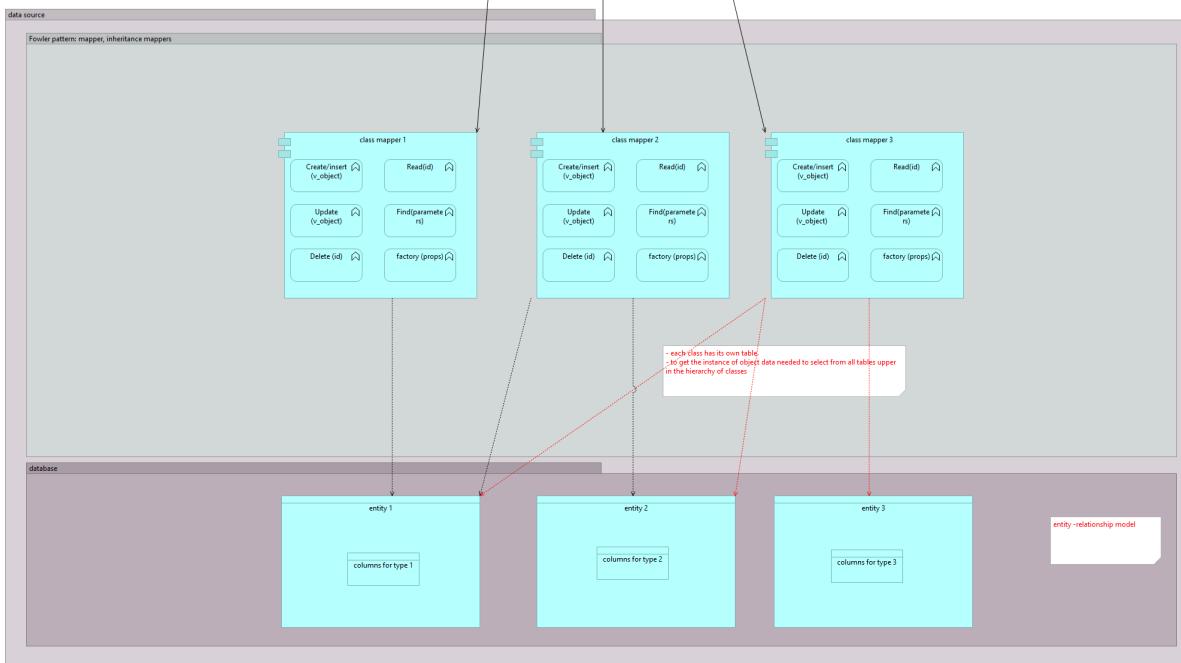
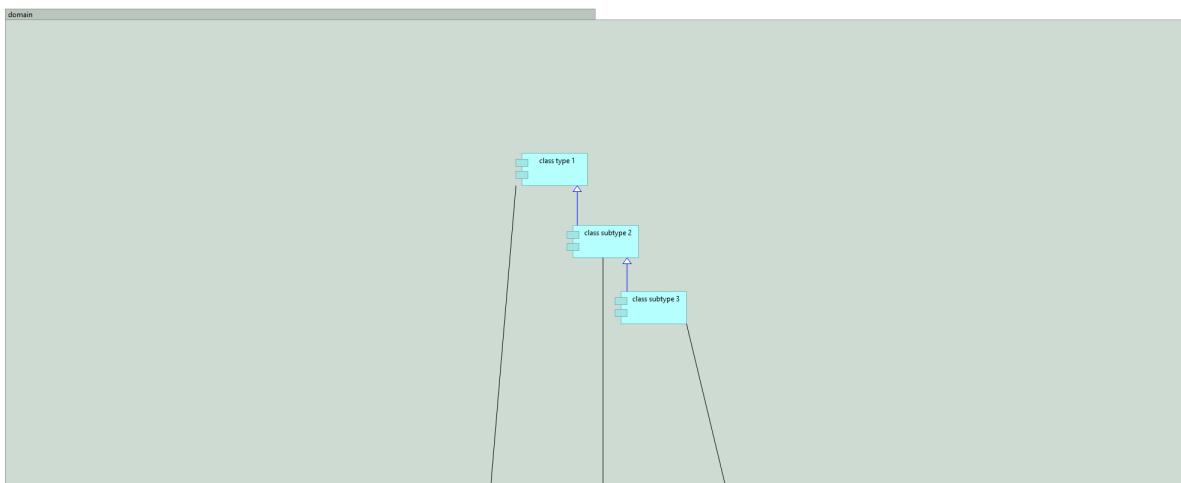
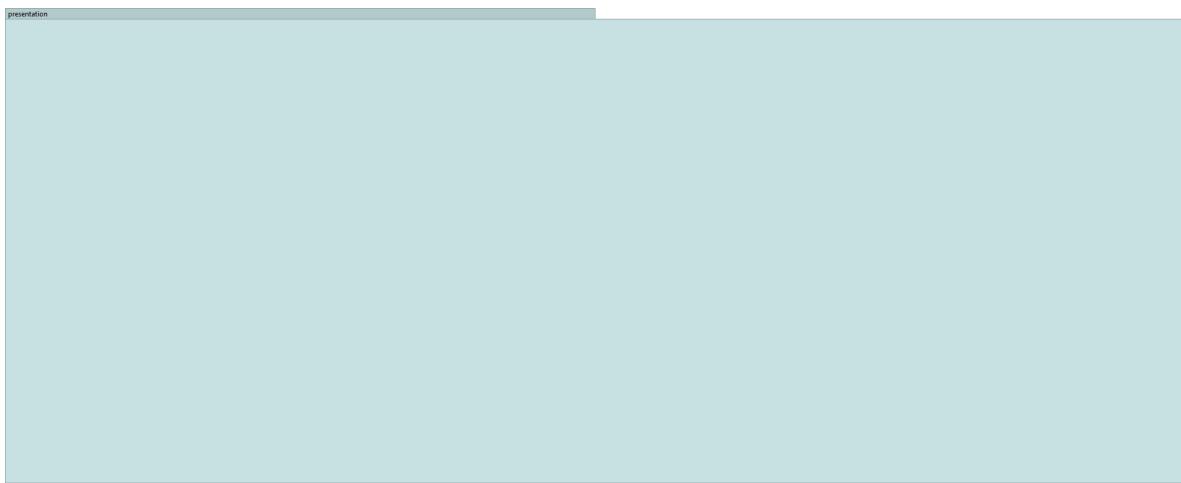
MODELING STRUCTURE HIERARCHY

ENTERPRISE PATTERNS

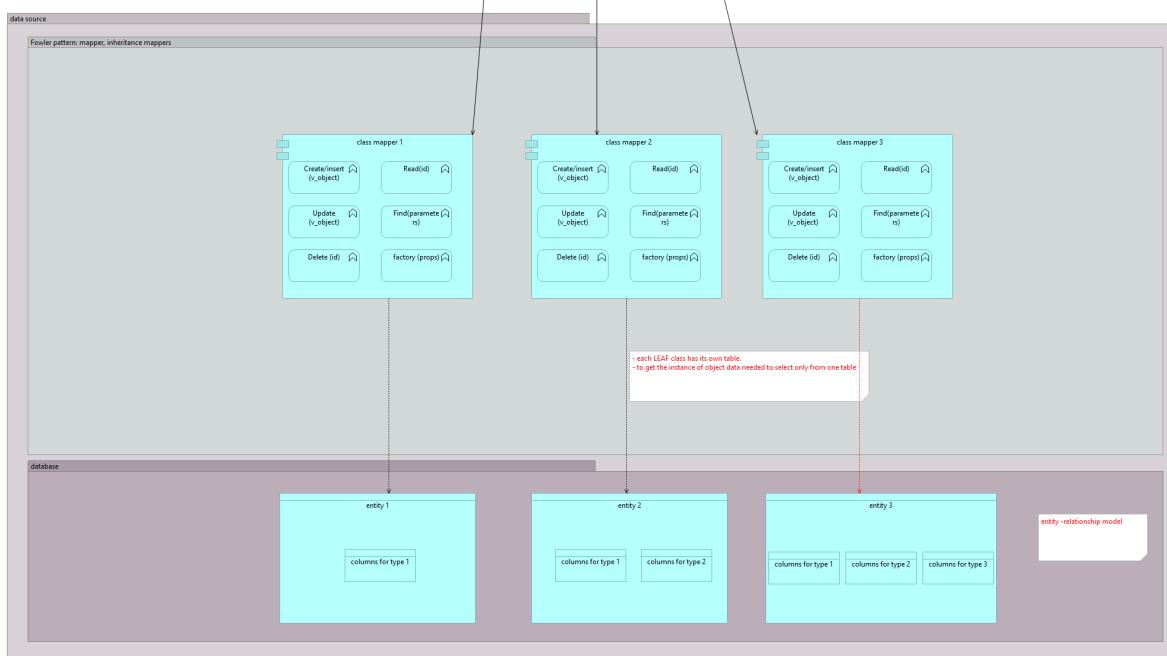
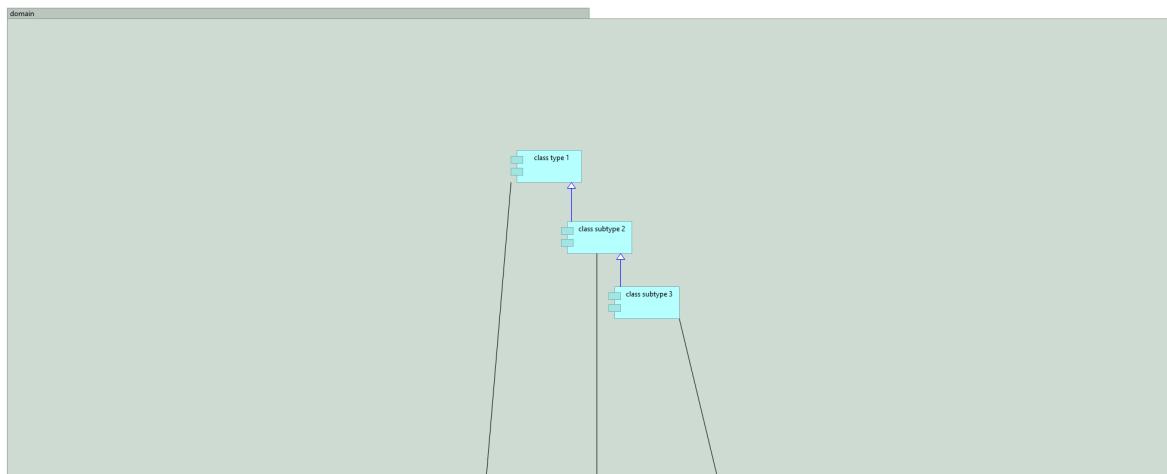
Martin Fowler



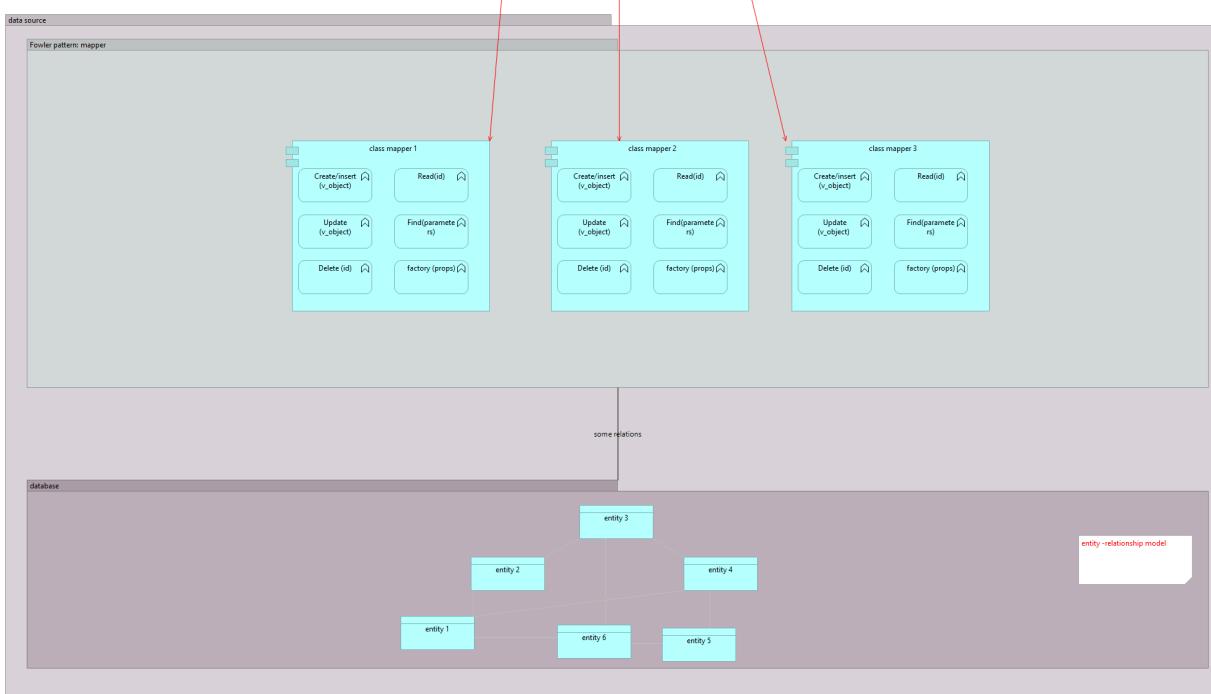
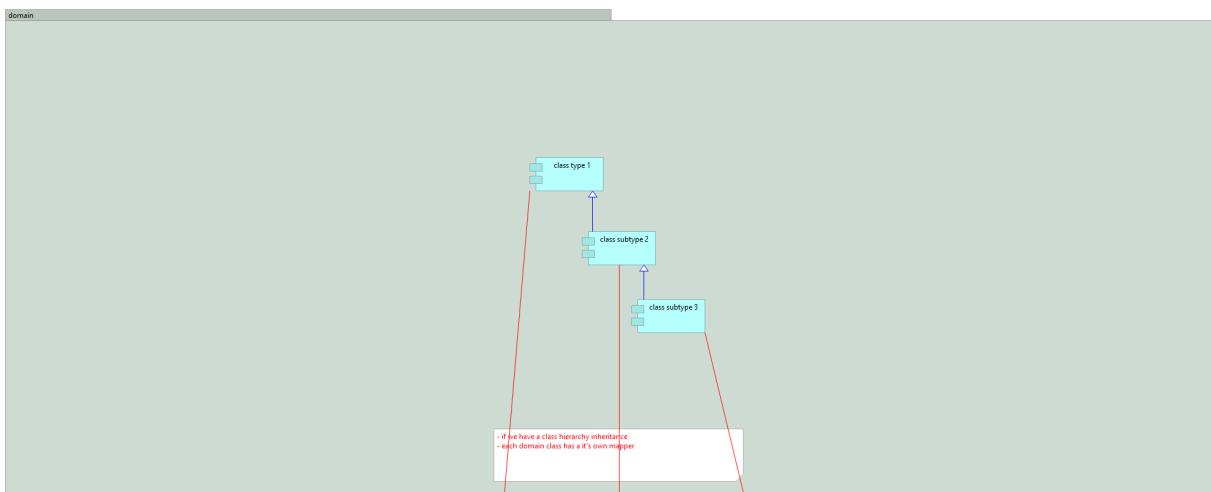
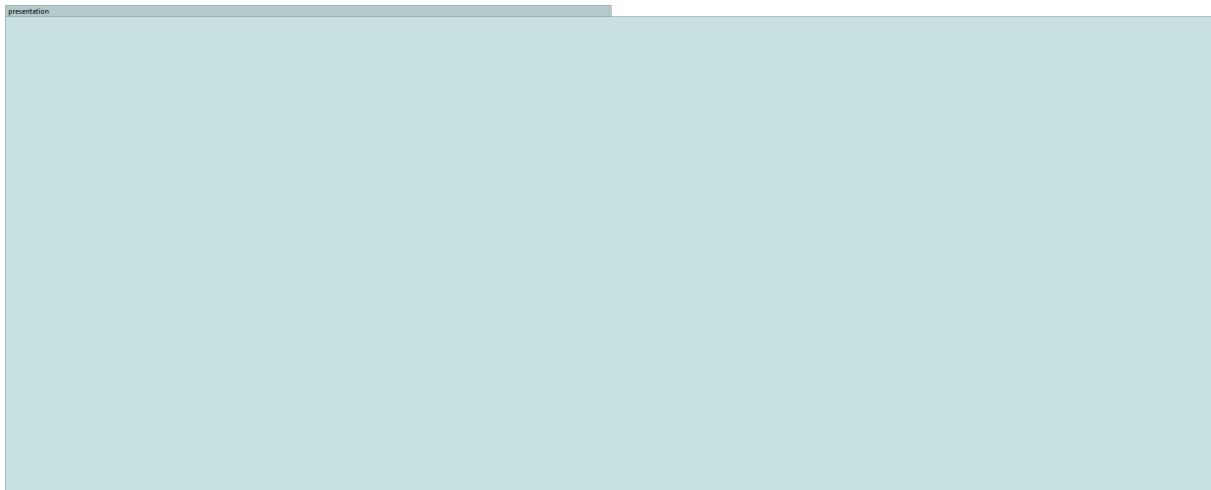
CLASS TABLE INHERITANCE



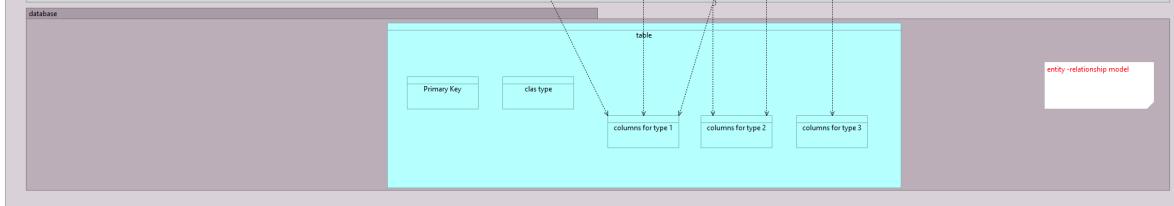
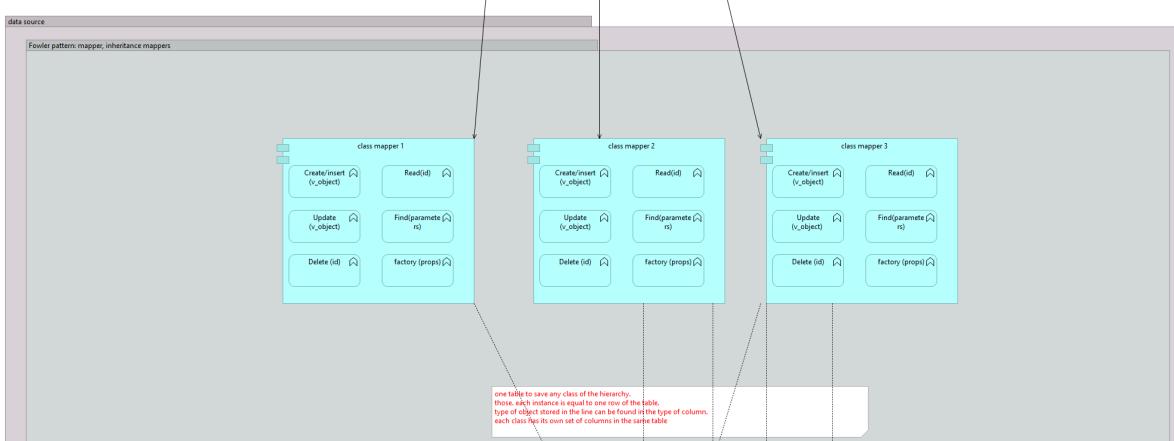
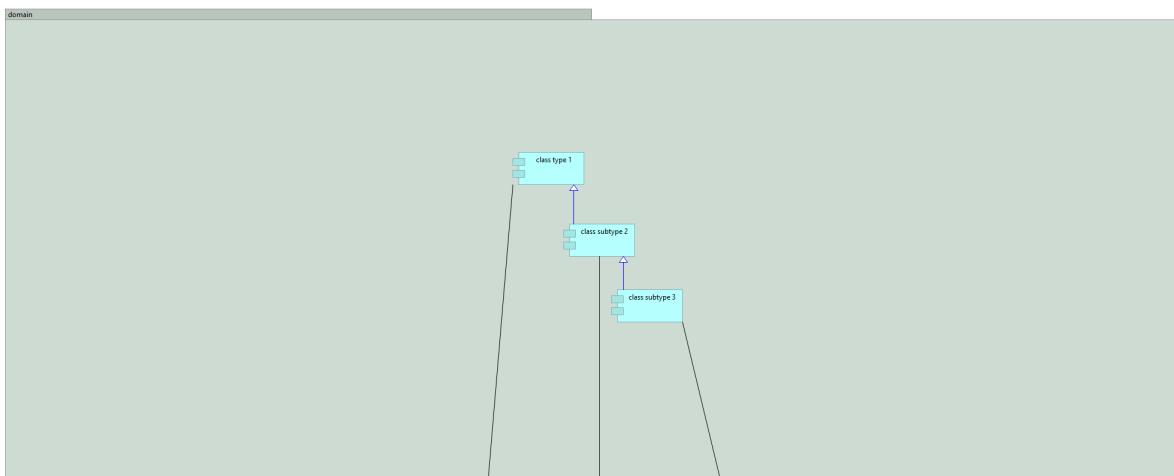
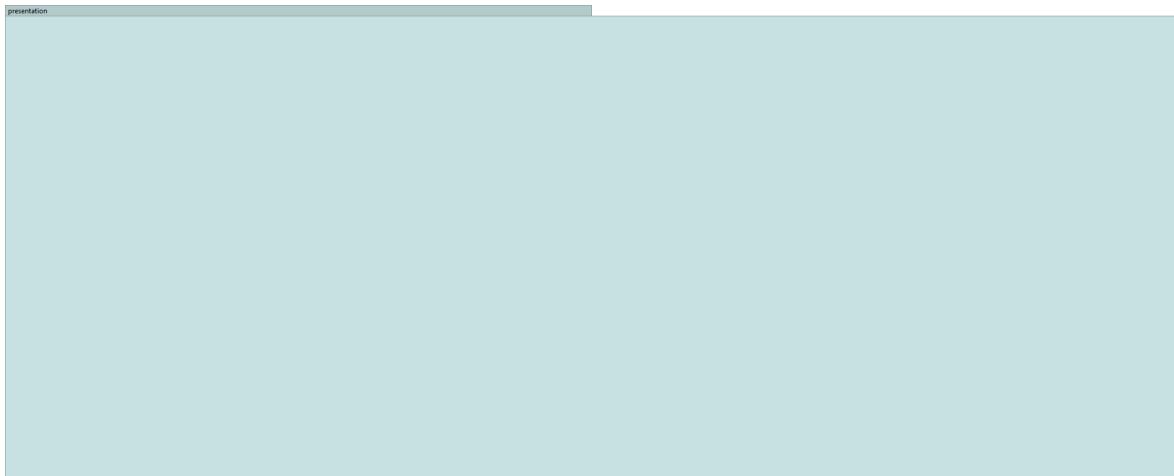
CONCRETE TABLE INHERITANCE



INHERITANCE MAPPERS



SINGLE TABLE INHERITANCE



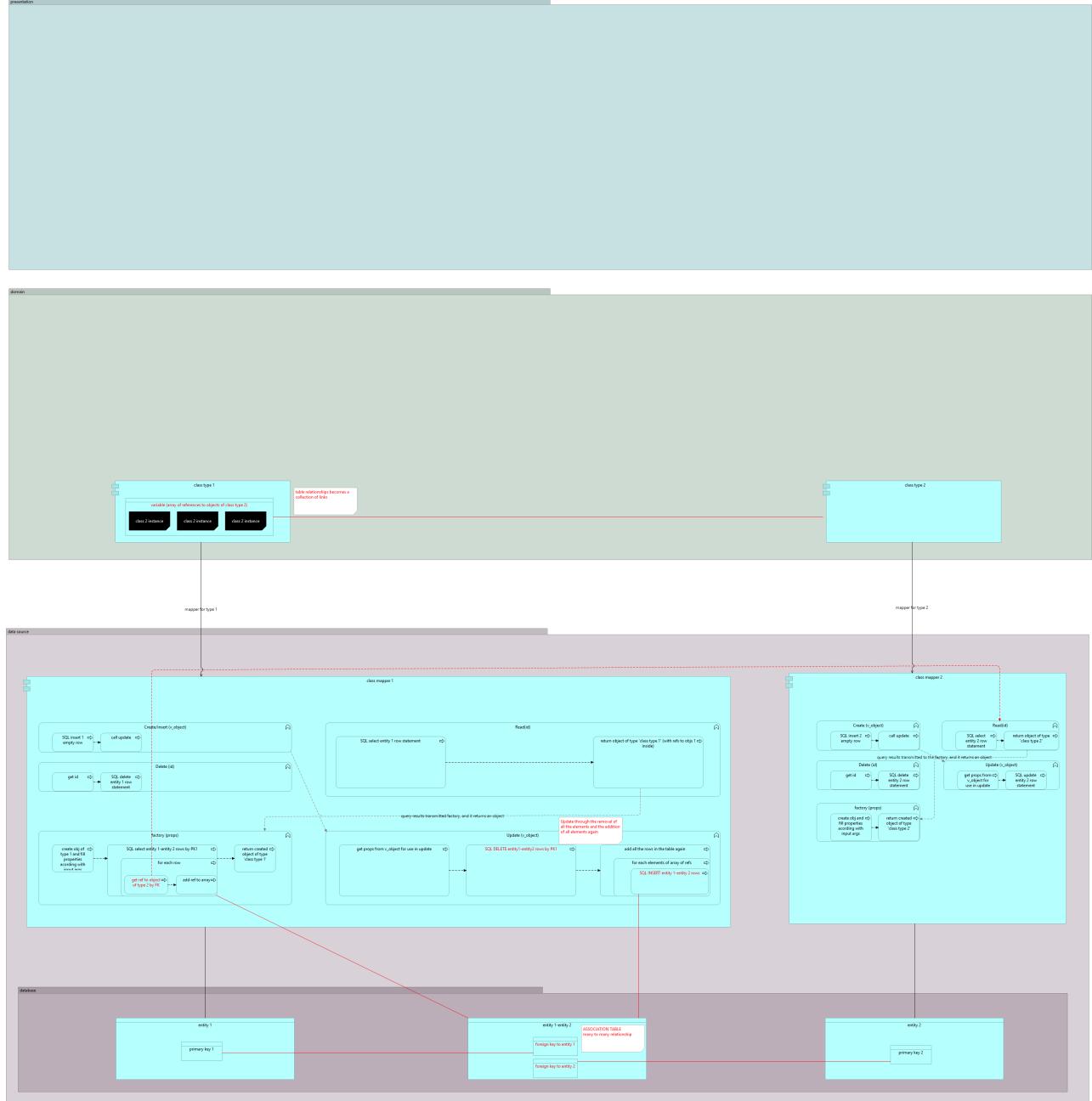
MODELING STRUCTURE RELATIONS

ENTERPRISE PATTERNS

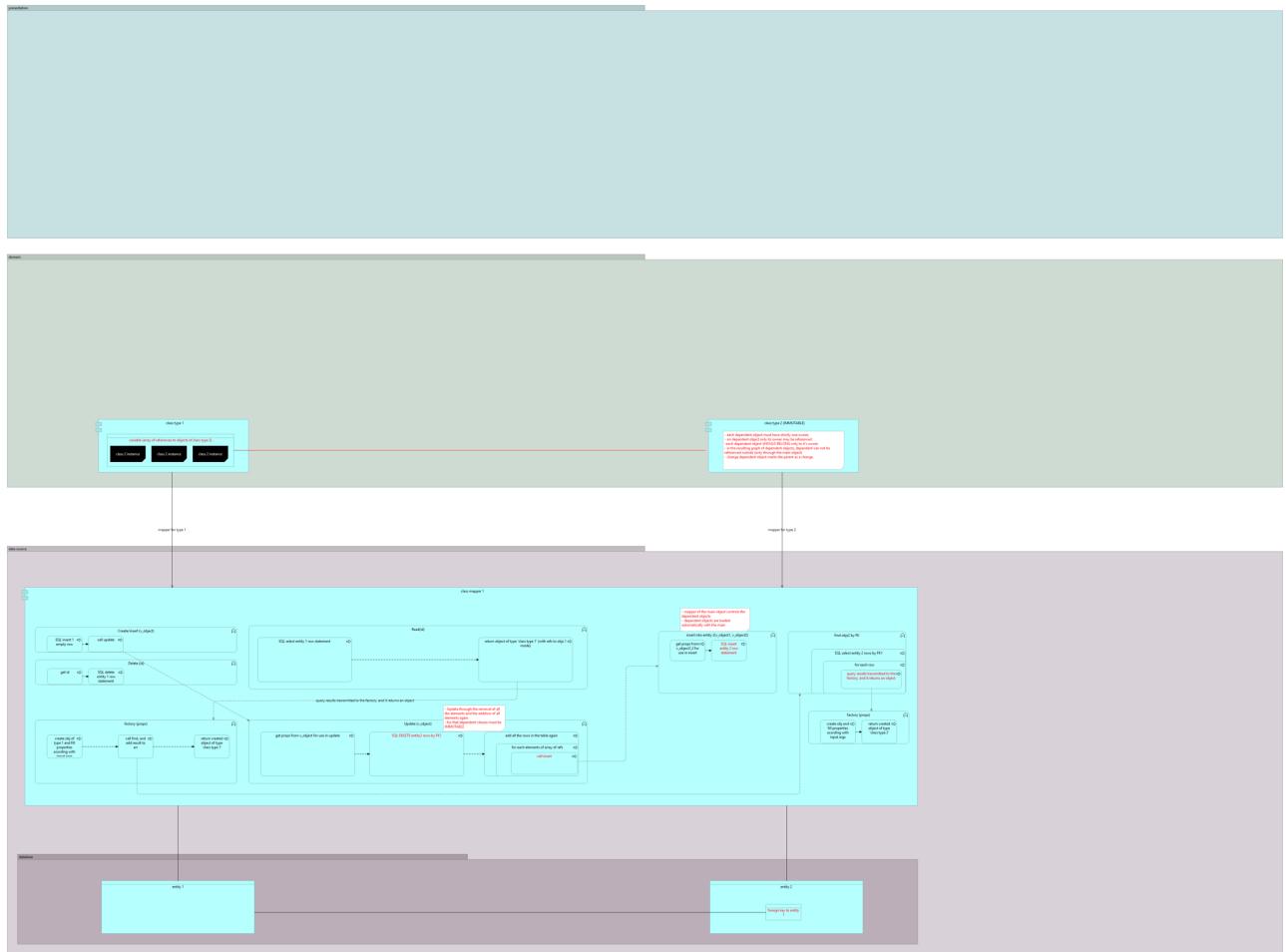
Martin Fowler



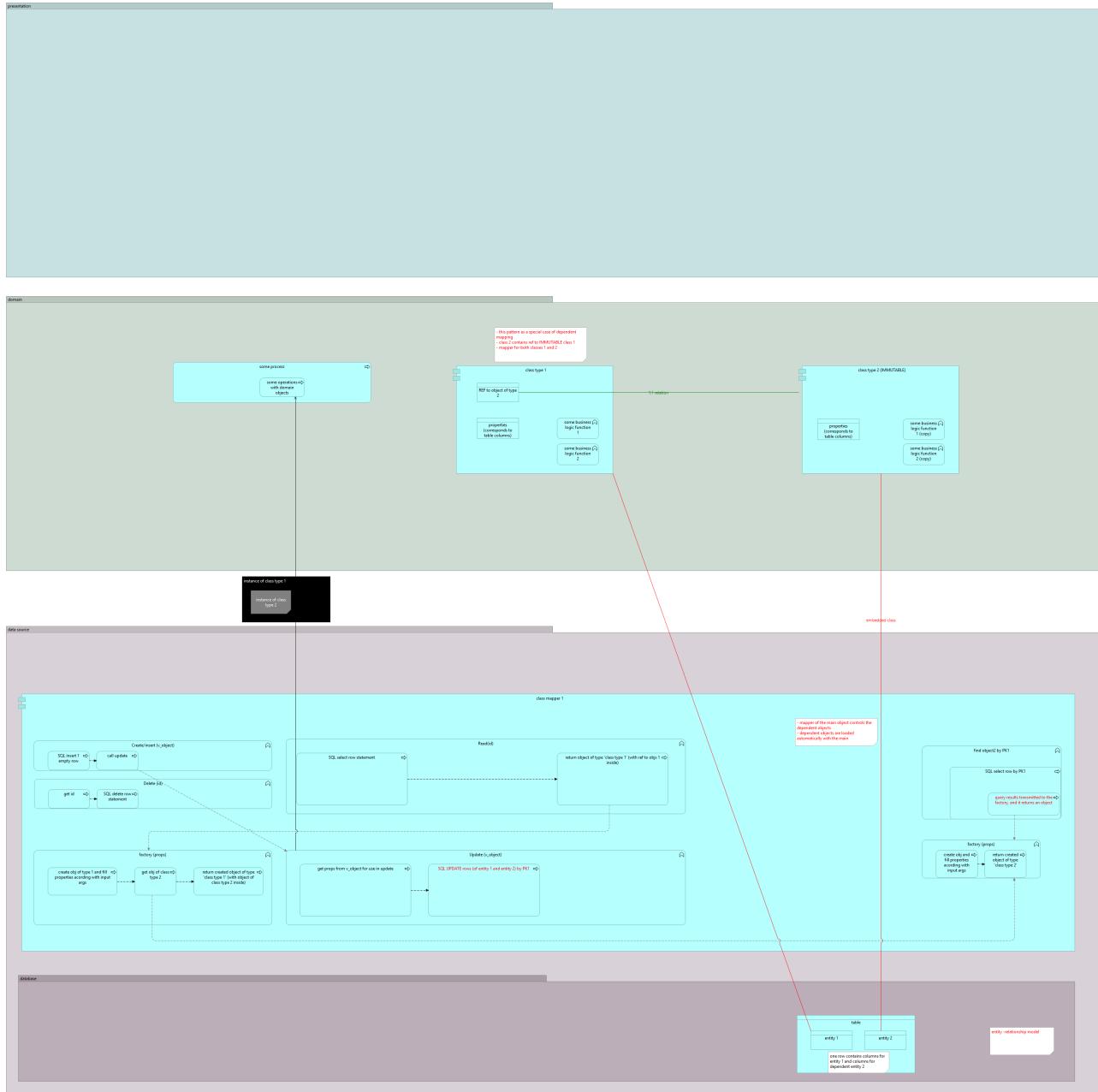
ASSOCIATION TABLE MAPPING



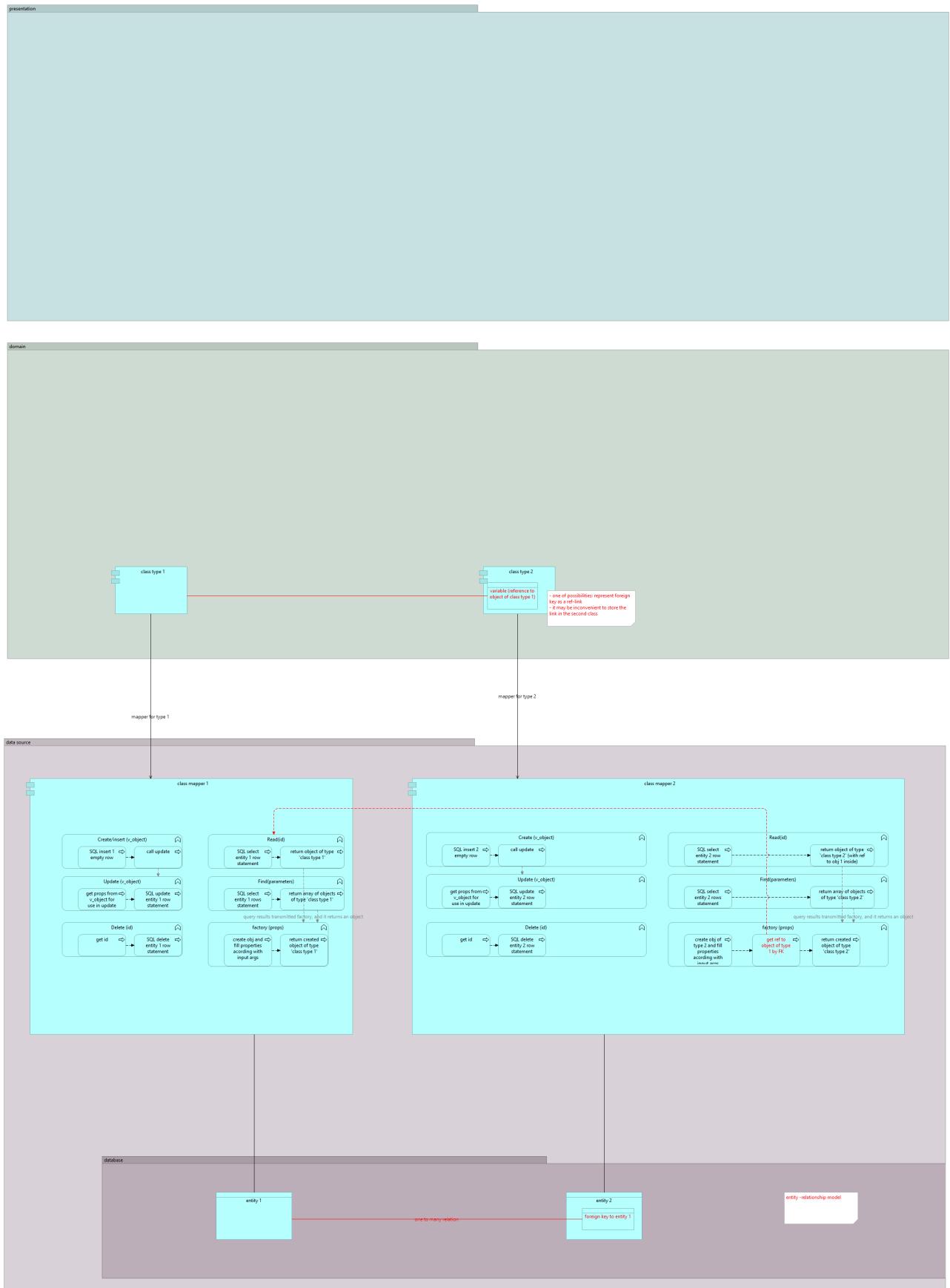
DEPENDENT MAPPING



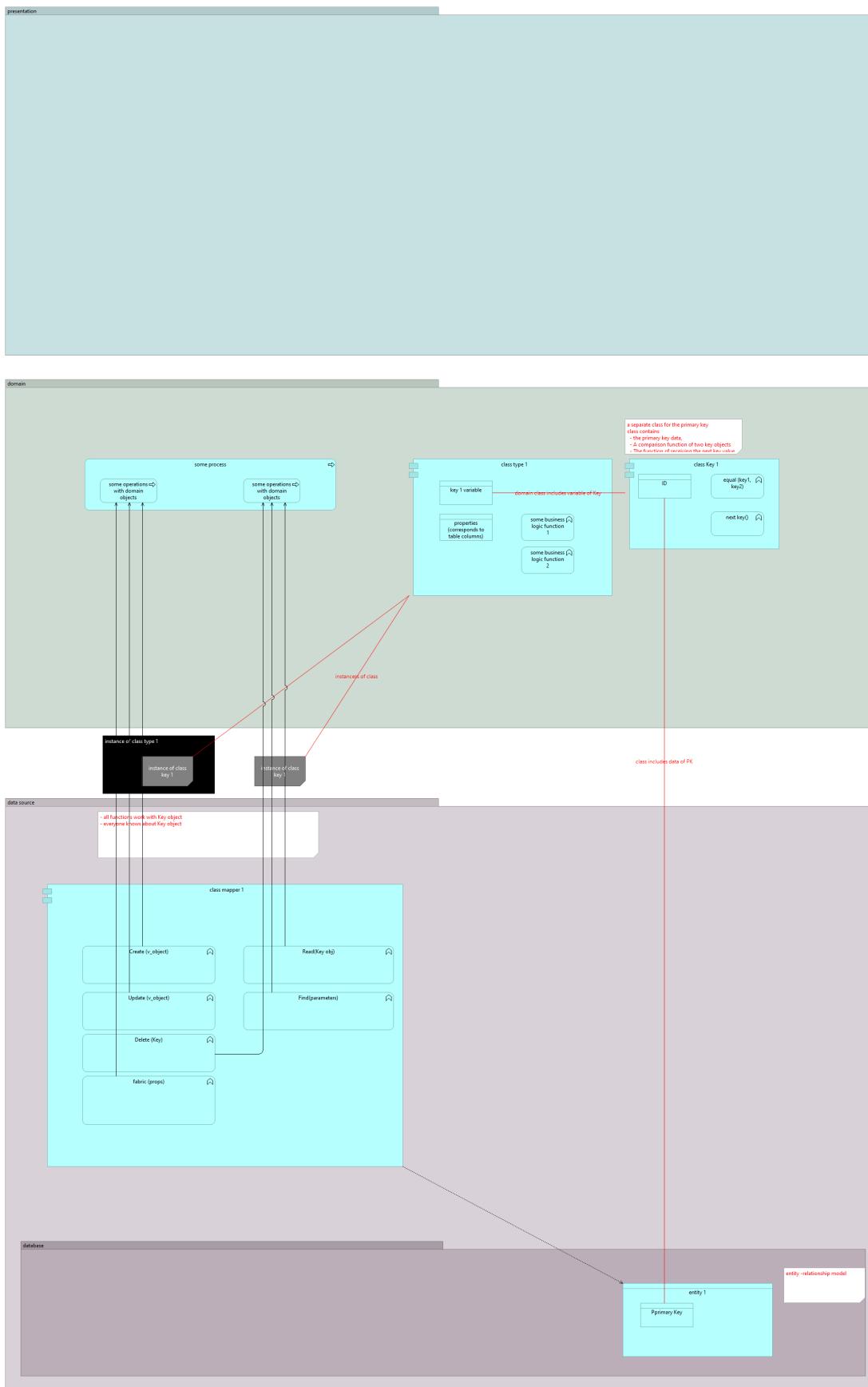
EMBEDDED VALUE



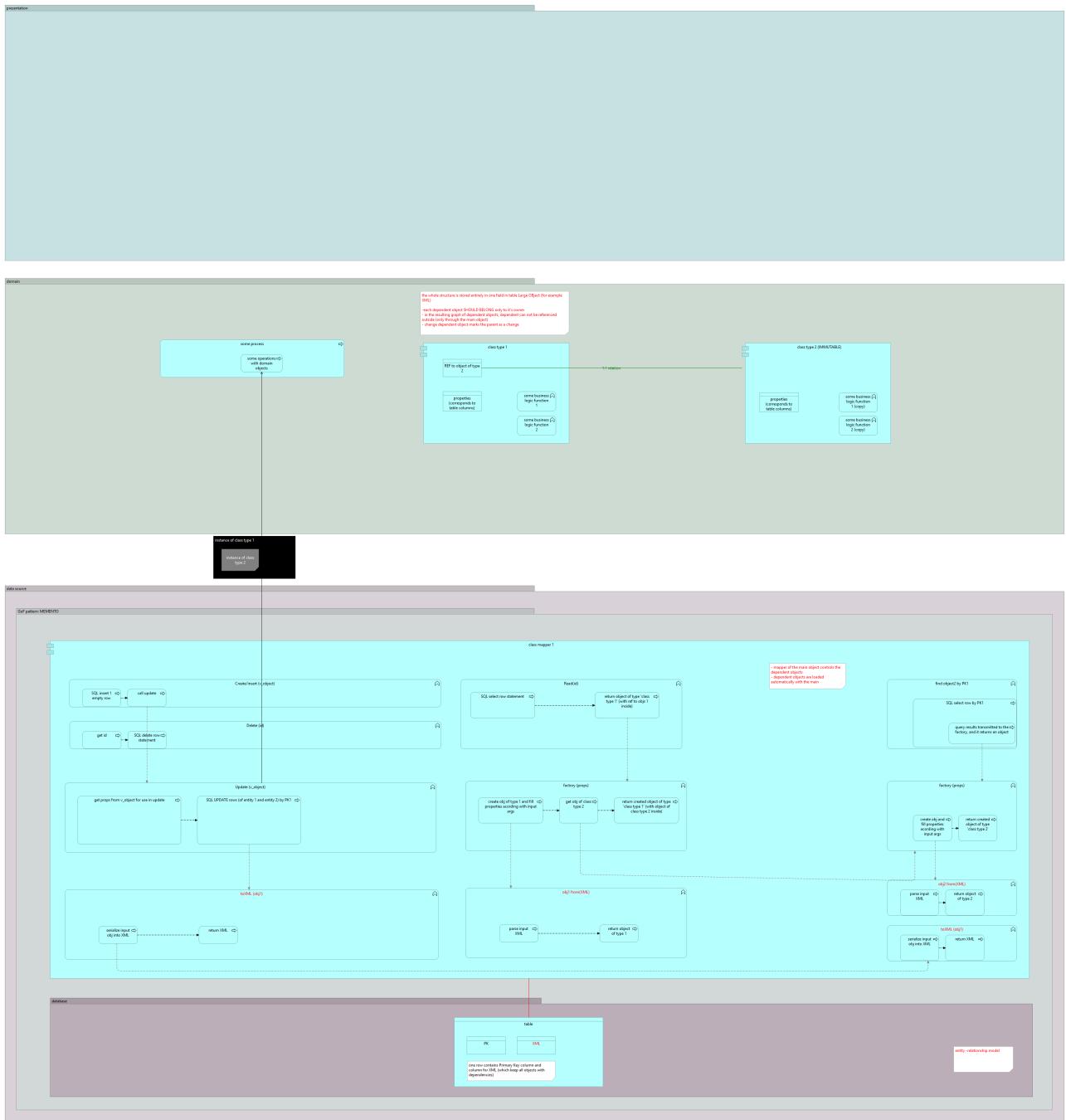
FOREIGN KEY MAPPING



IDENTITY FIELD



SERIALIZED LOB



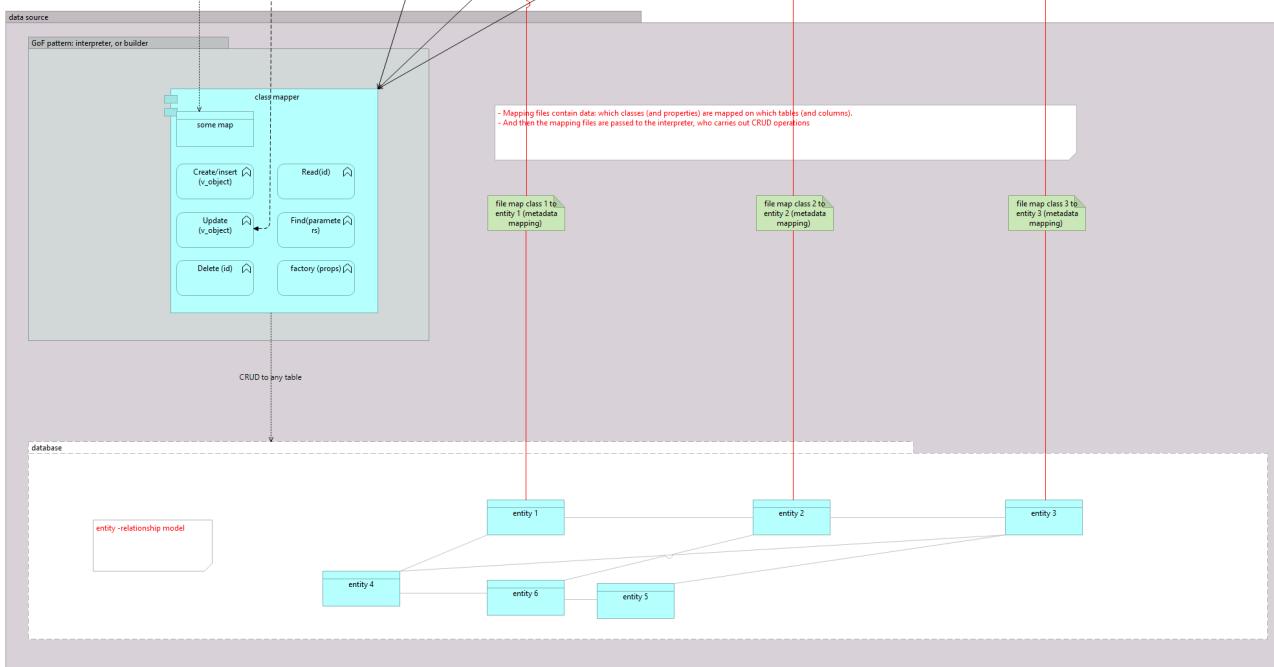
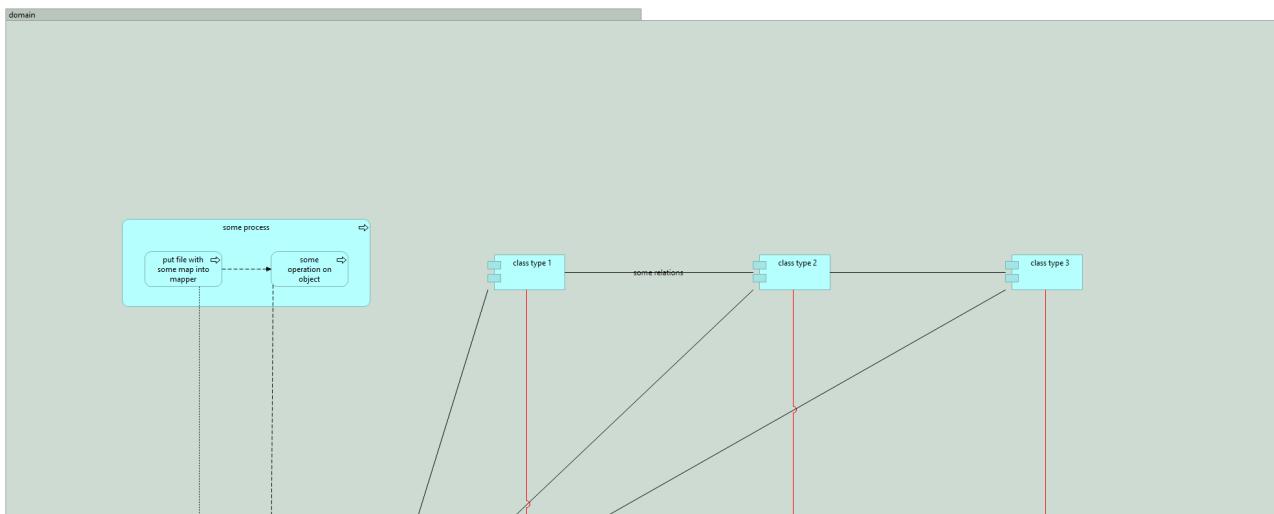
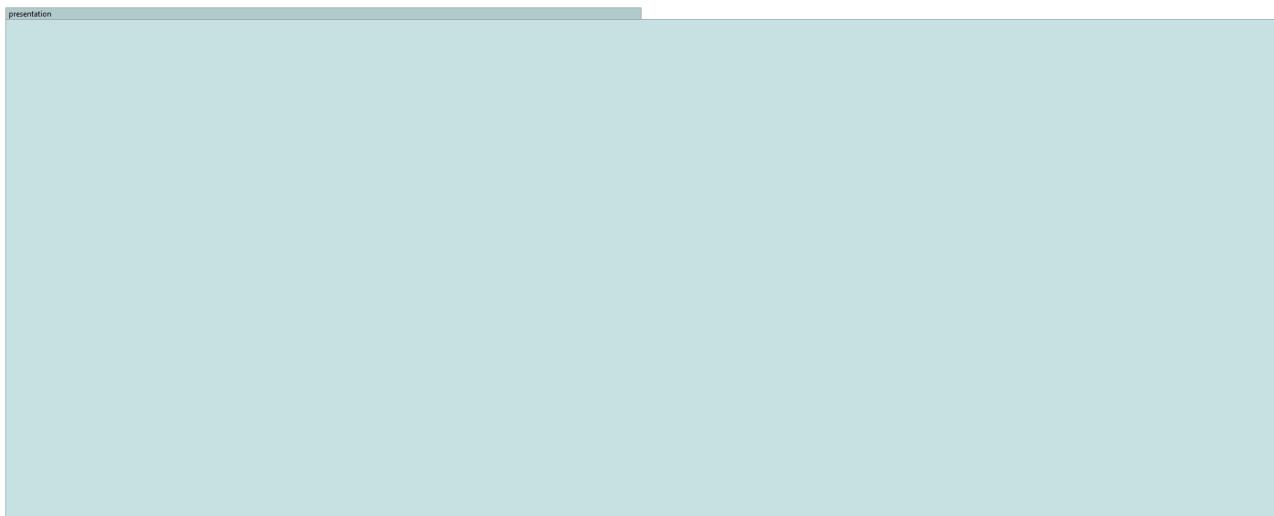
METADATA

ENTERPRISE PATTERNS

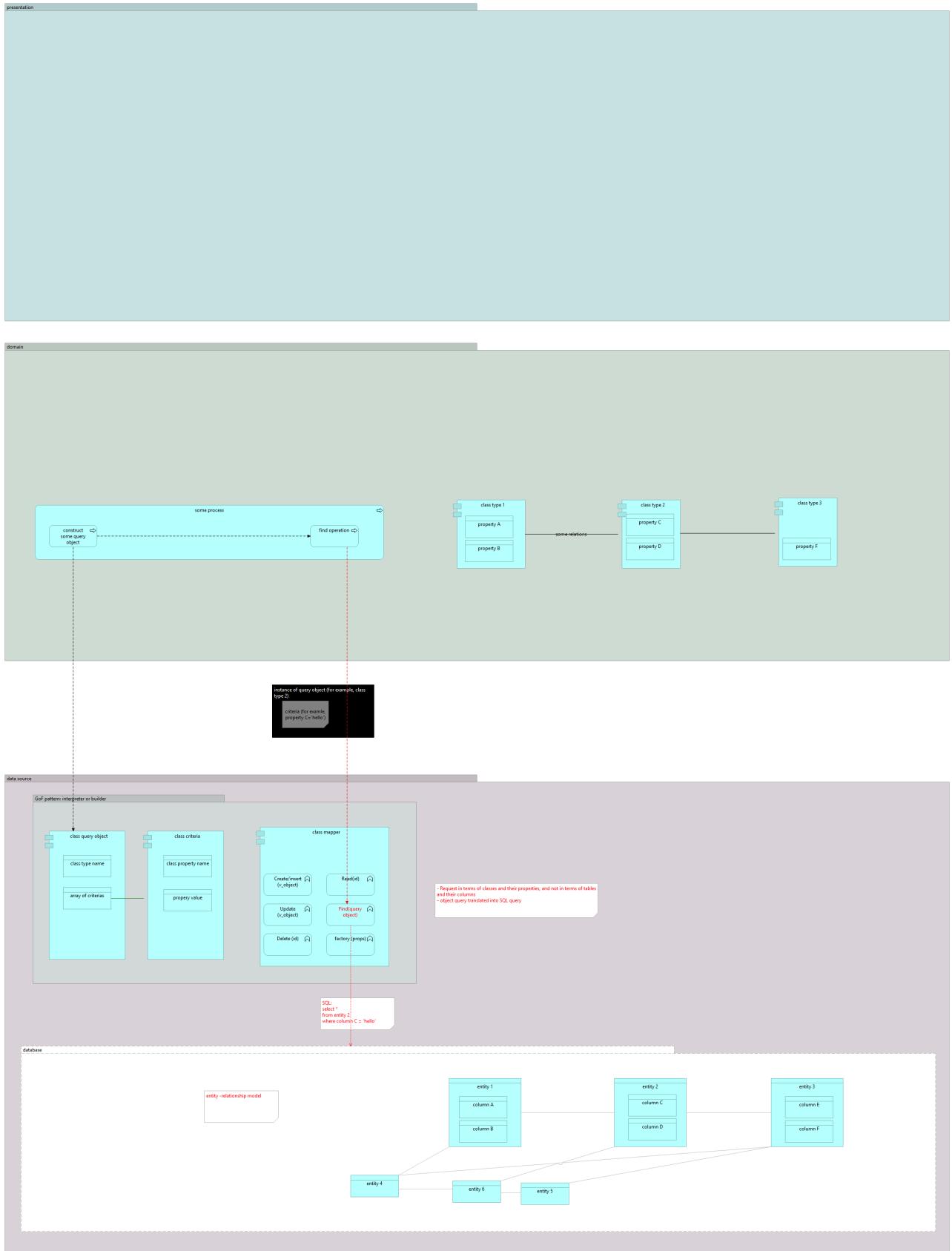
Martin Fowler



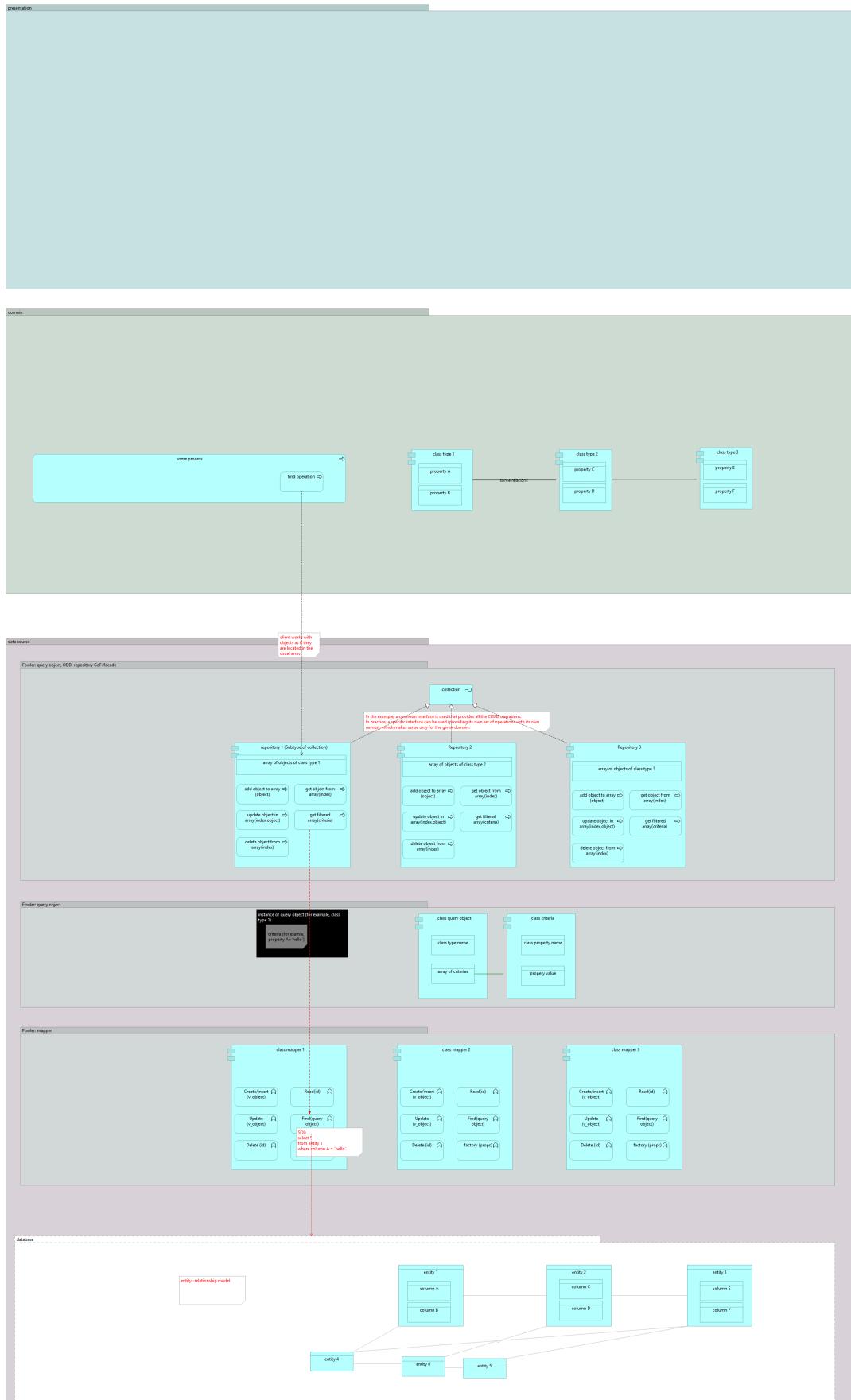
METADATA MAPPING



QUERY OBJECT



REPOSITORY



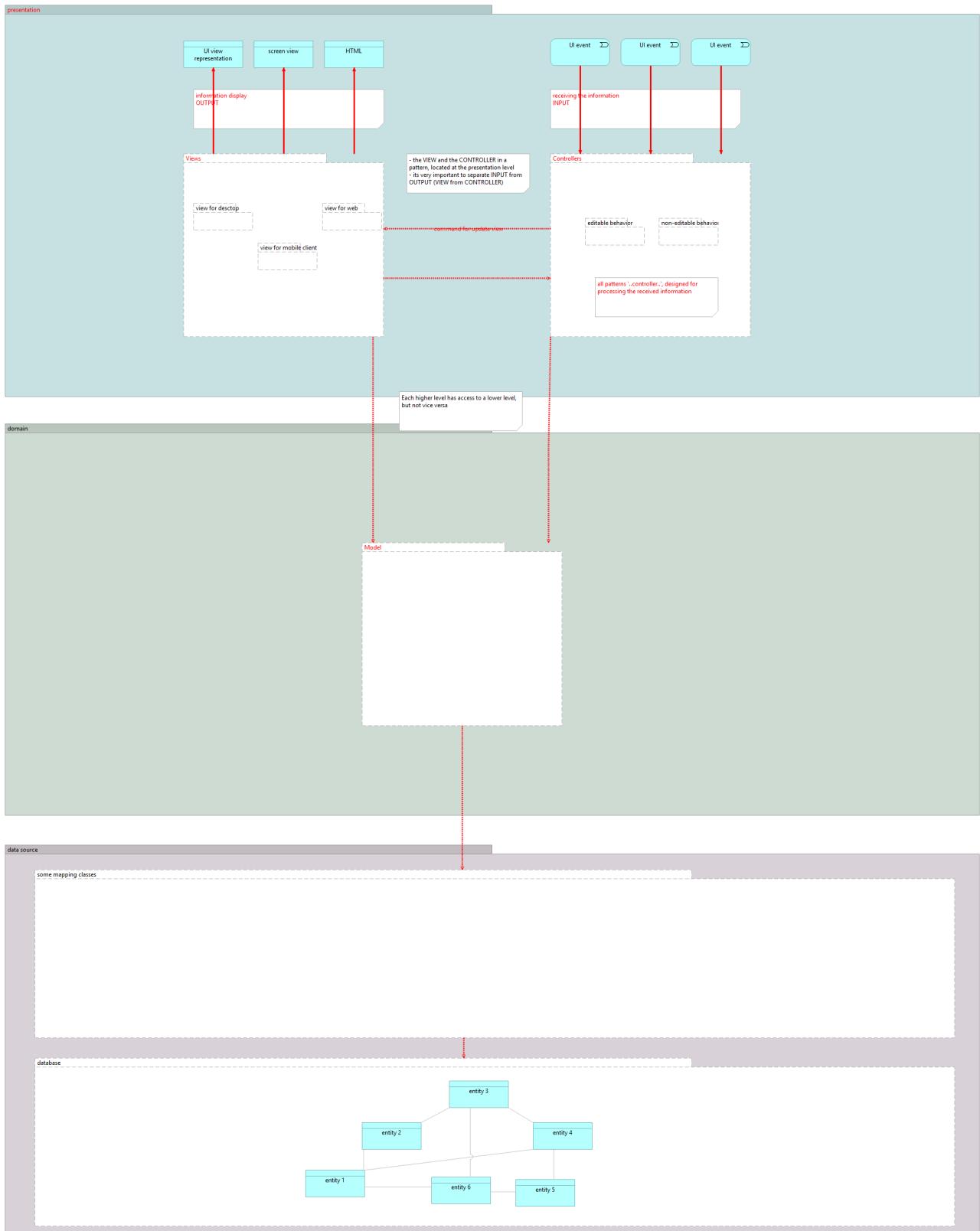
WEB REPRESENTATION CONTROLLER

ENTERPRISE PATTERNS

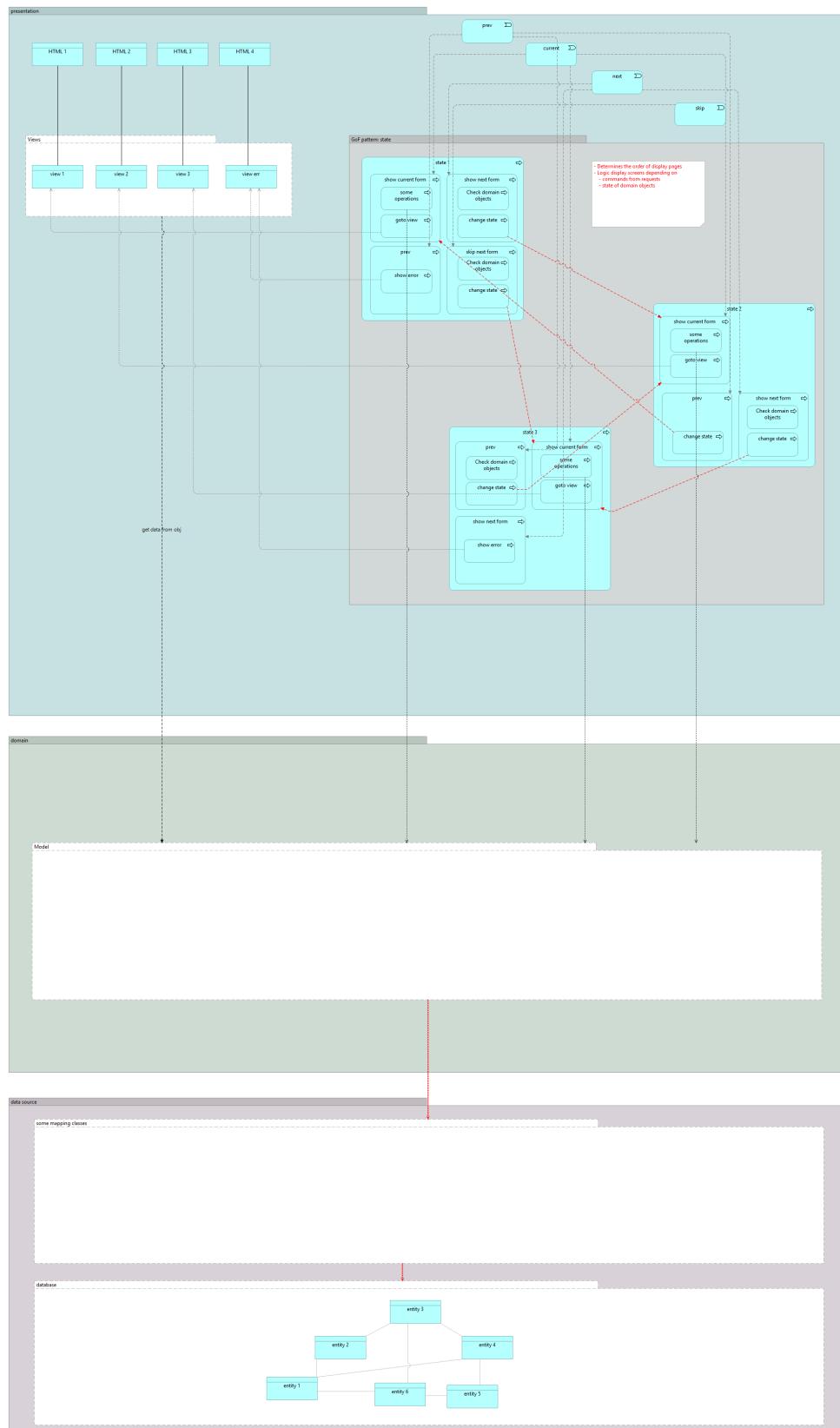
Martin Fowler



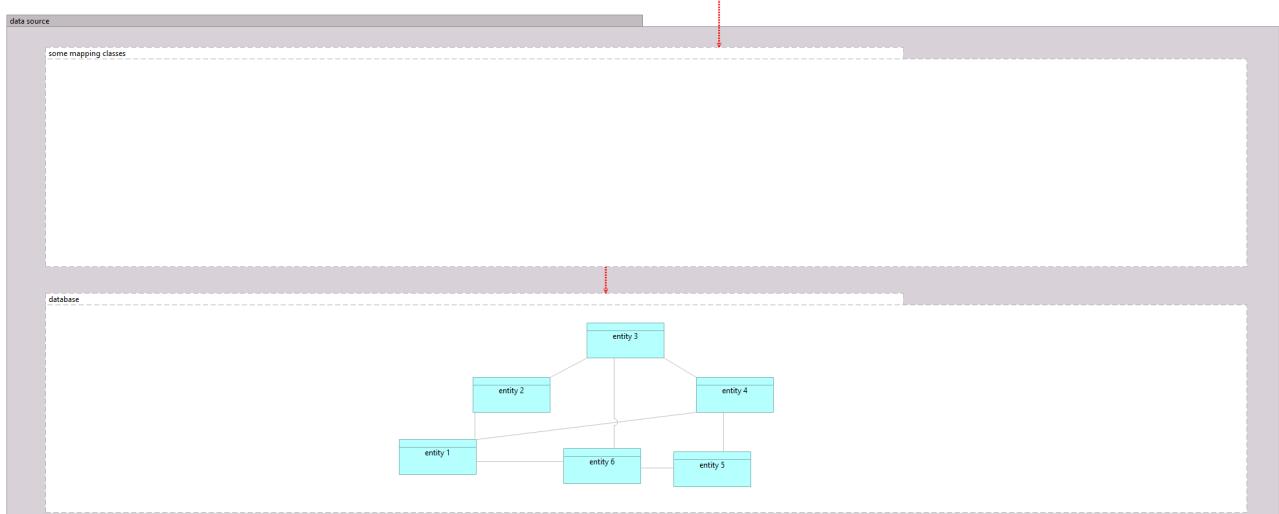
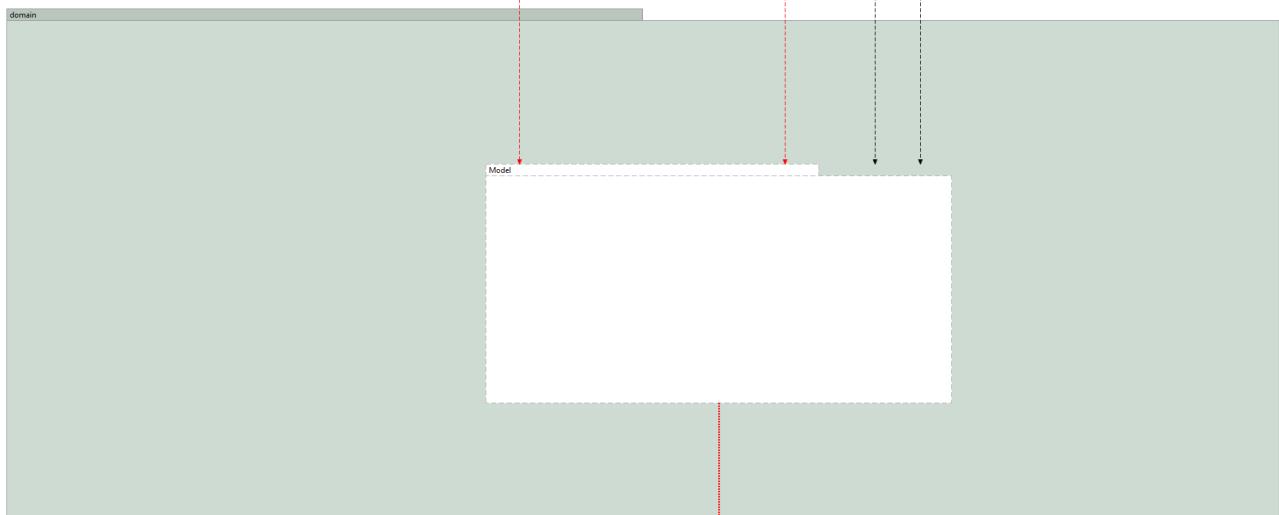
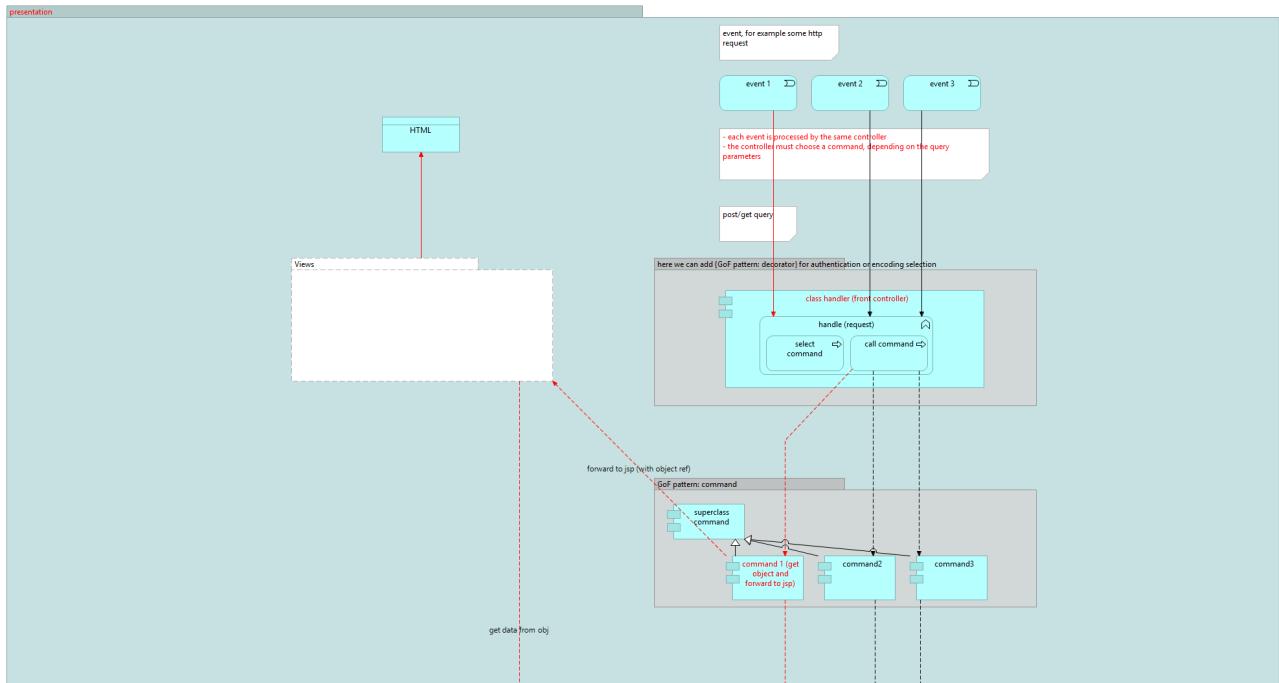
MODEL VIEW CONTROLLER



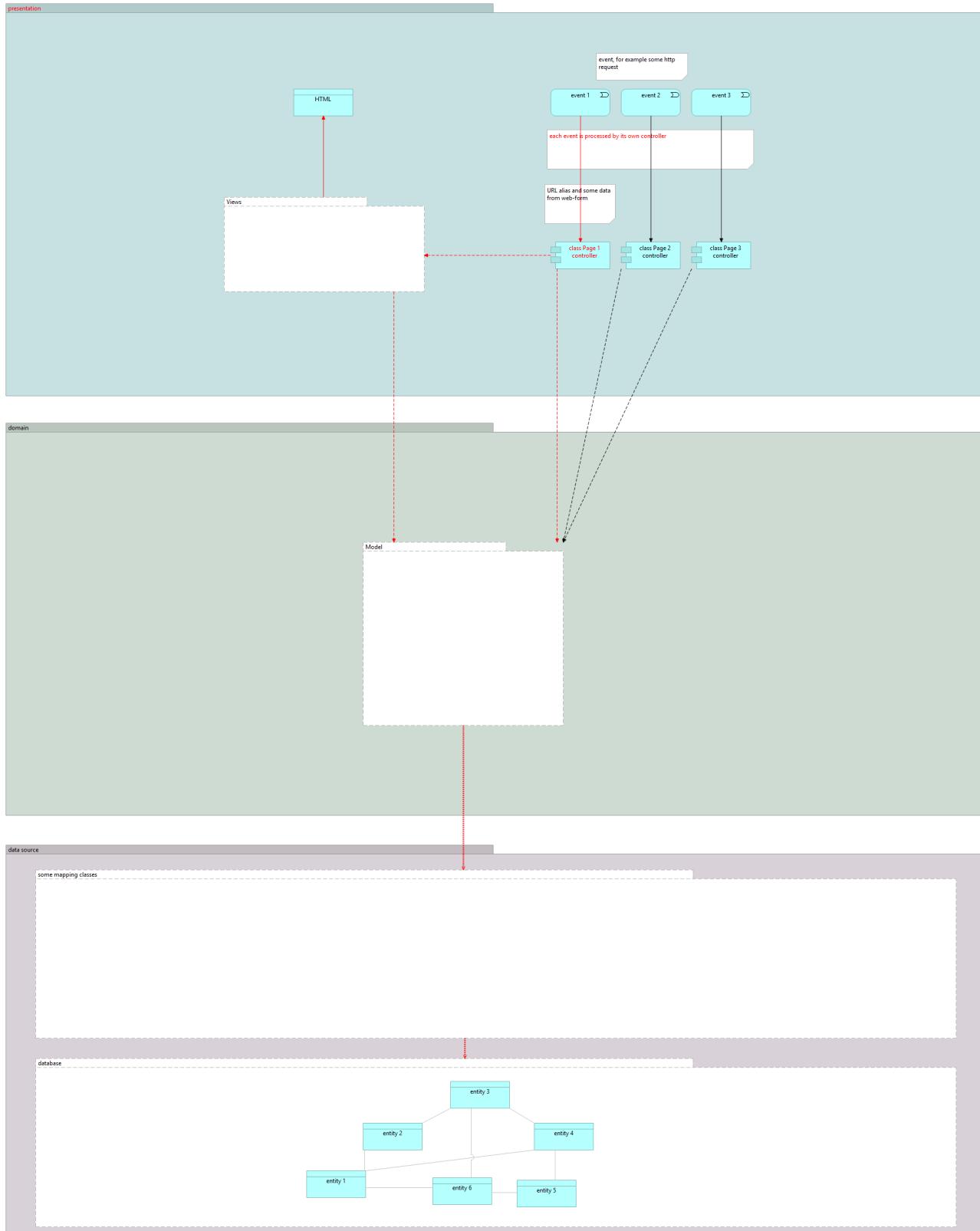
APPLICATION CONTROLLER



FRONT CONTROLLER



PAGE CONTROLLER



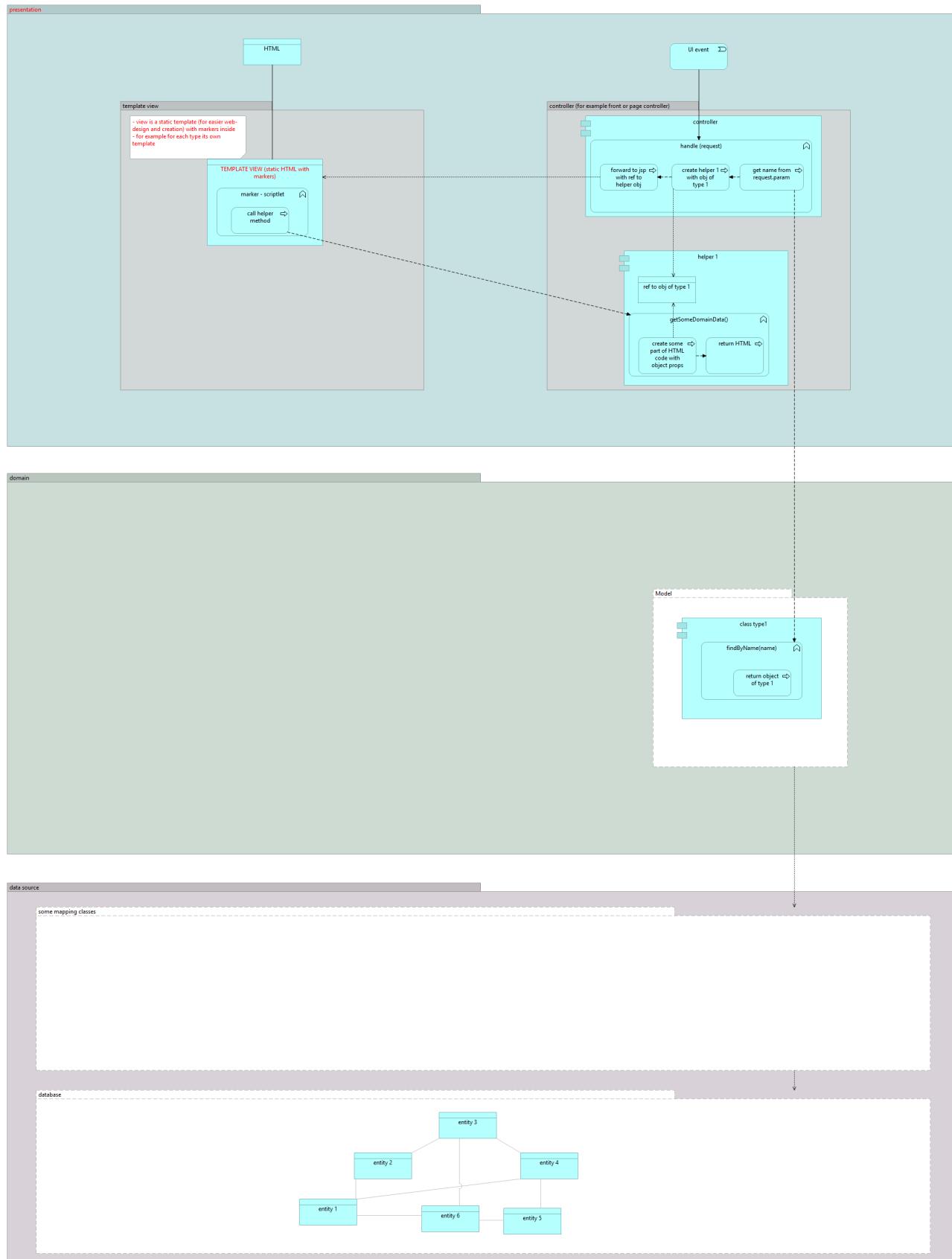
WEB REPRESENTATION VIEW

ENTERPRISE PATTERNS

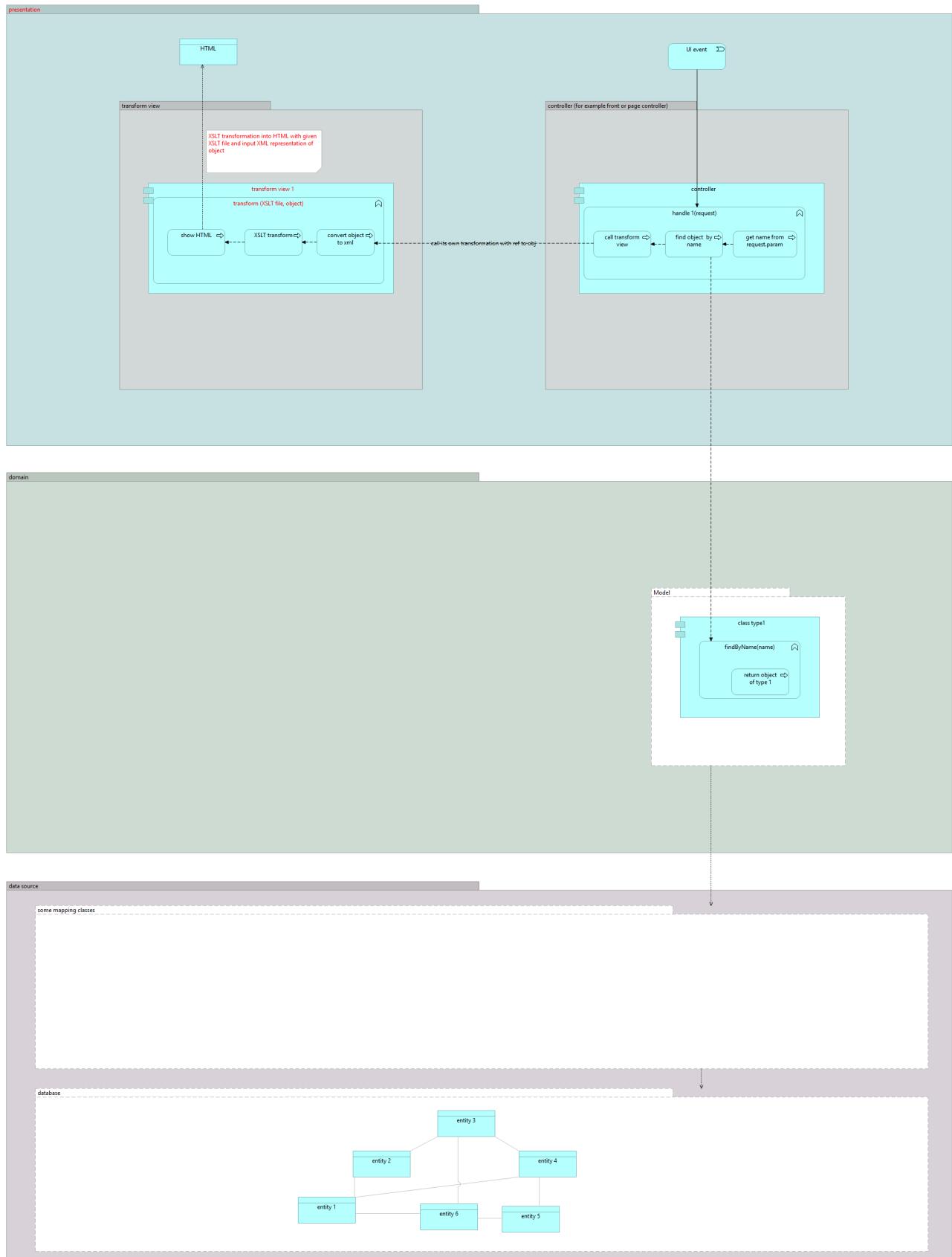
Martin Fowler



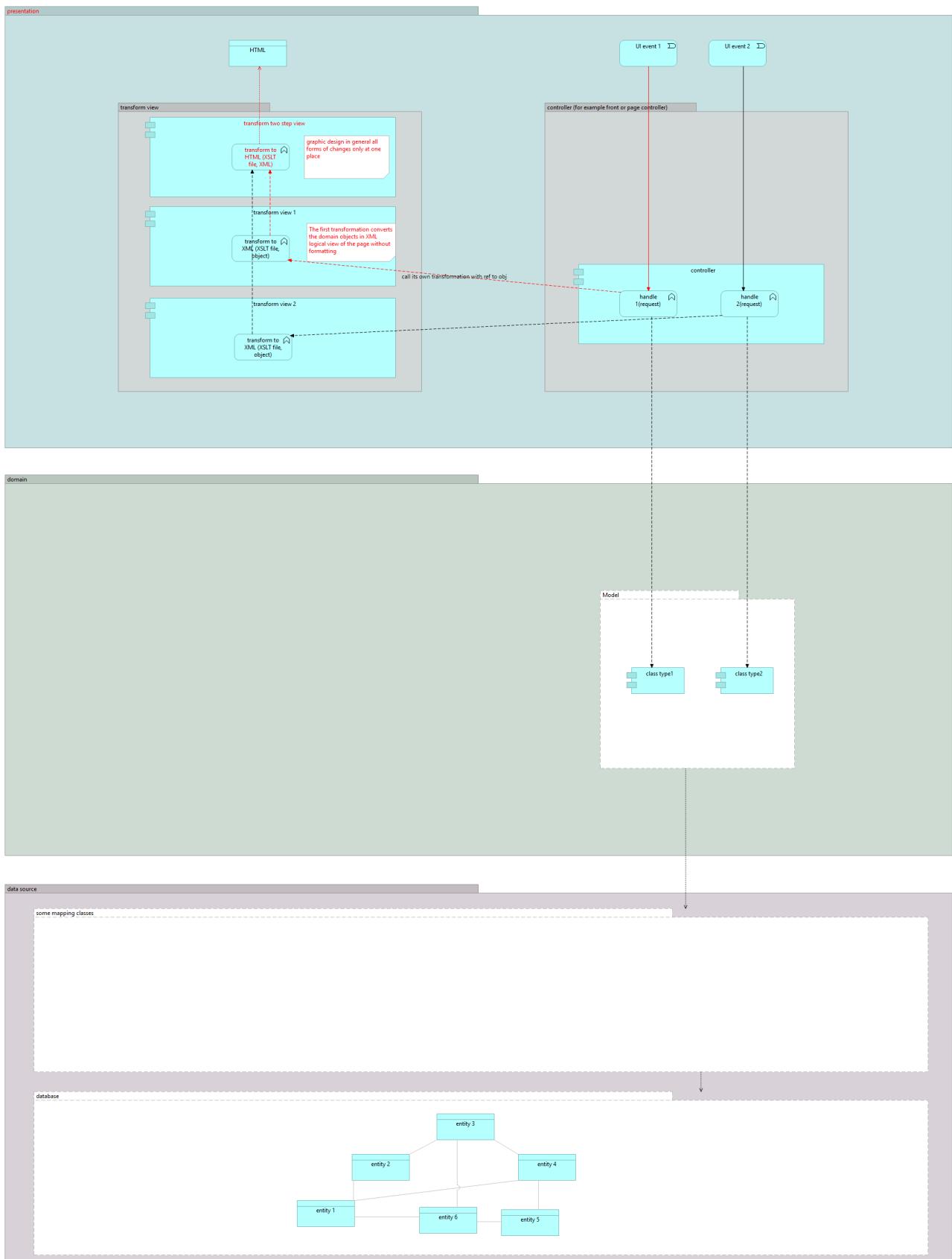
TEMPLATE VIEW



TRANSFORM VIEW



TWO STEP VIEW



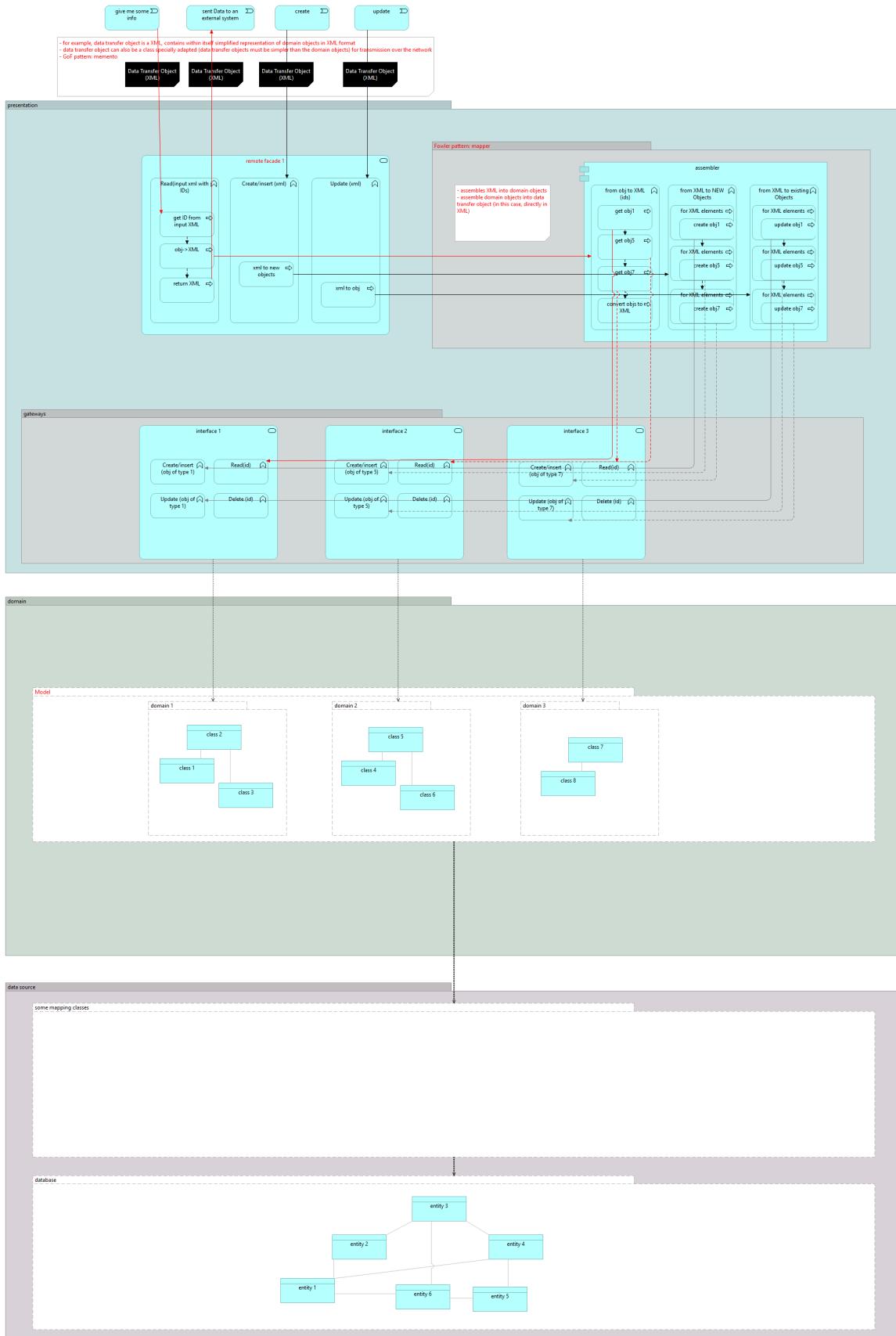
DISTRIBUTED PROCESSING

ENTERPRISE PATTERNS

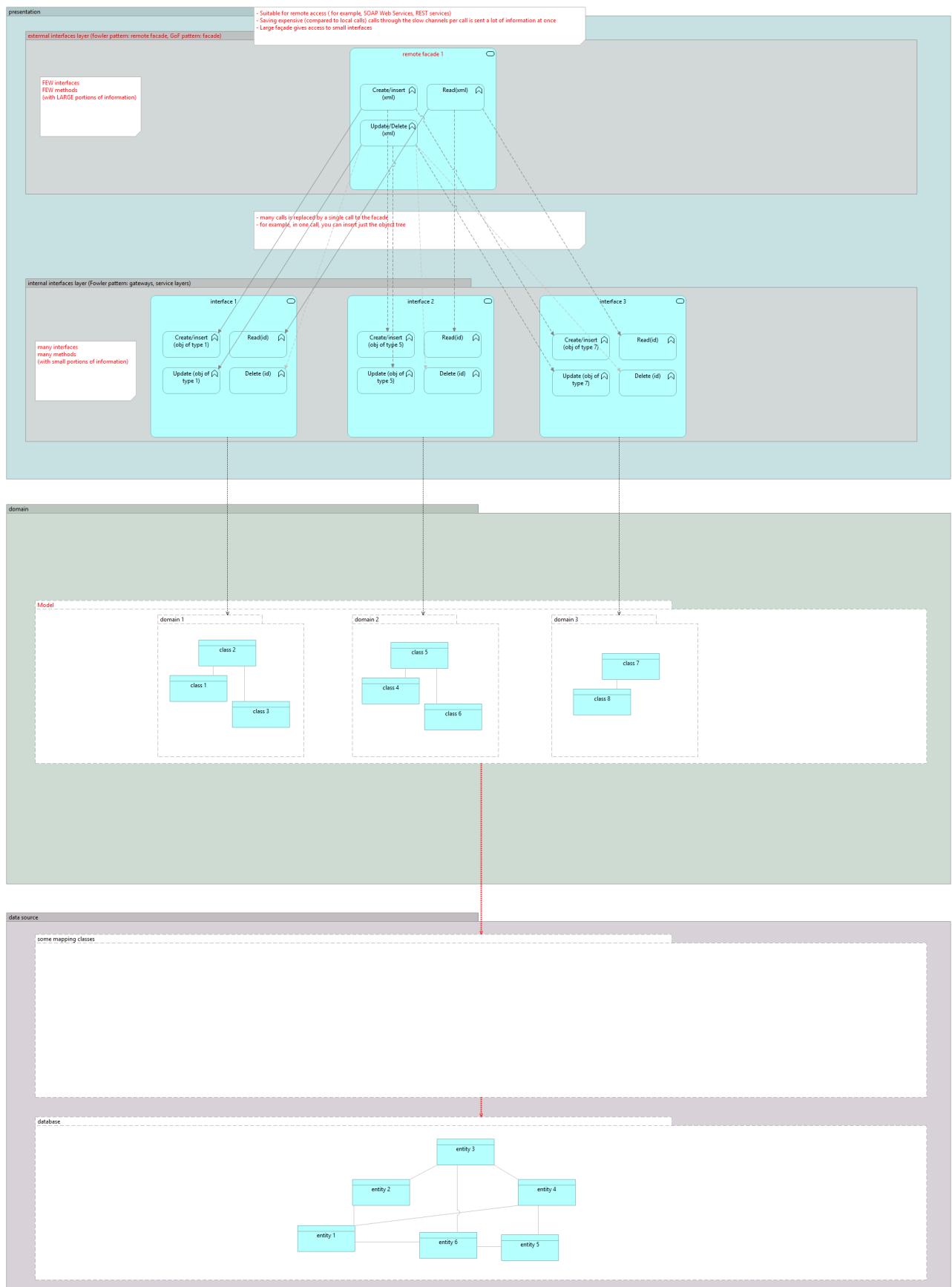
Martin Fowler



DATA TRANSFER OBJECT



REMOTE FAÇADE



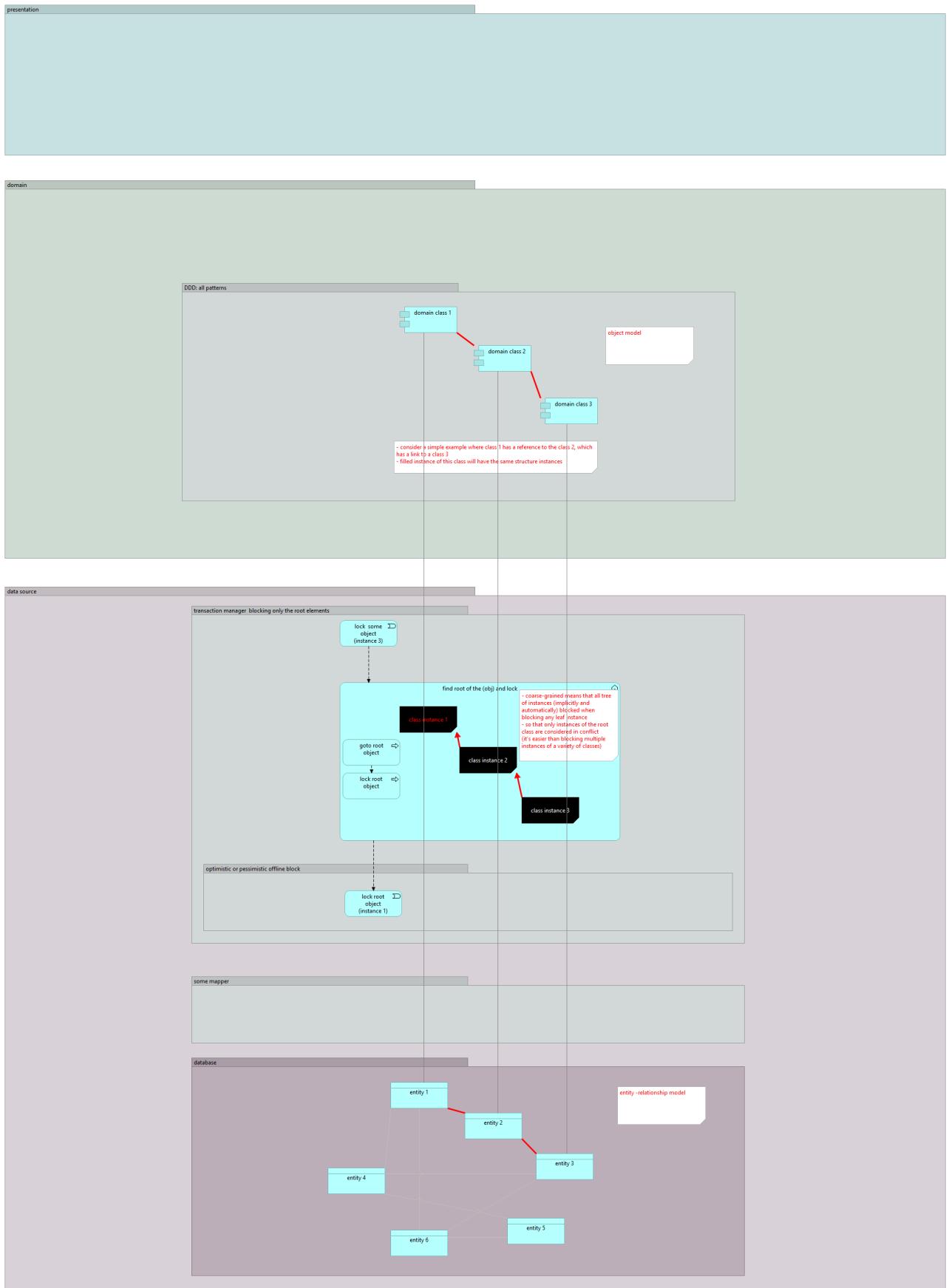
PARALLEL PROCESSING

ENTERPRISE PATTERNS

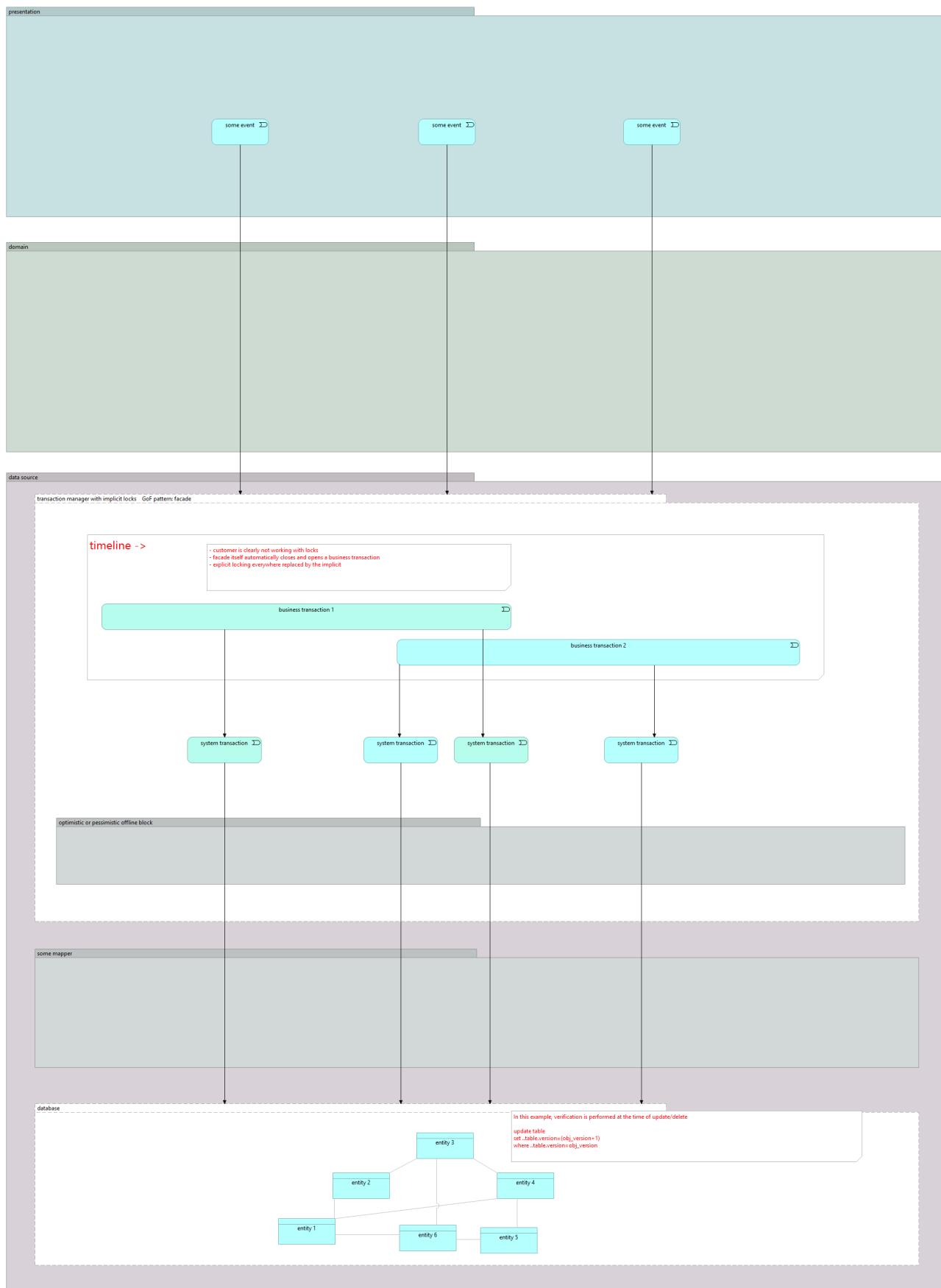
Martin Fowler



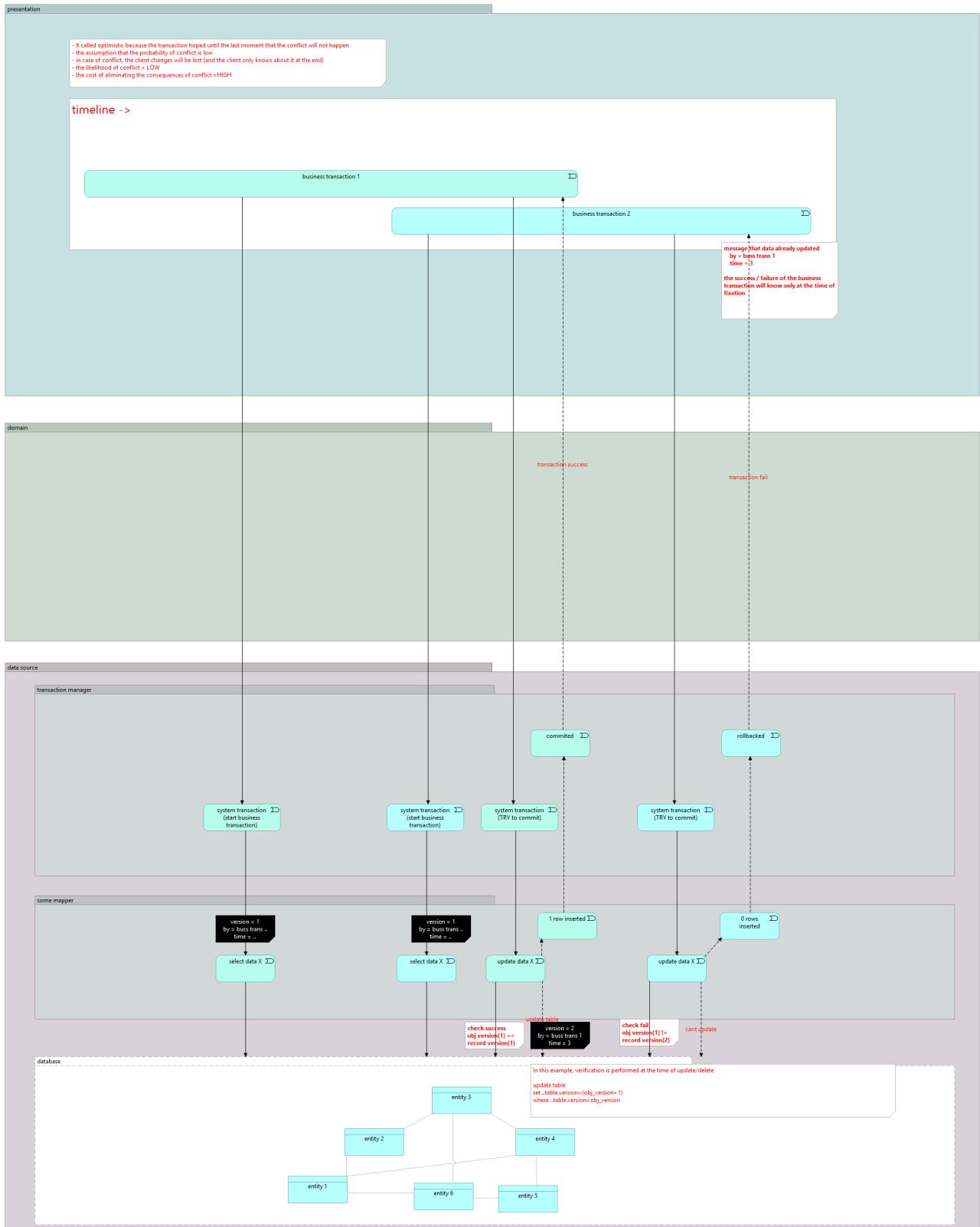
COARSE-GRAINED LOCK



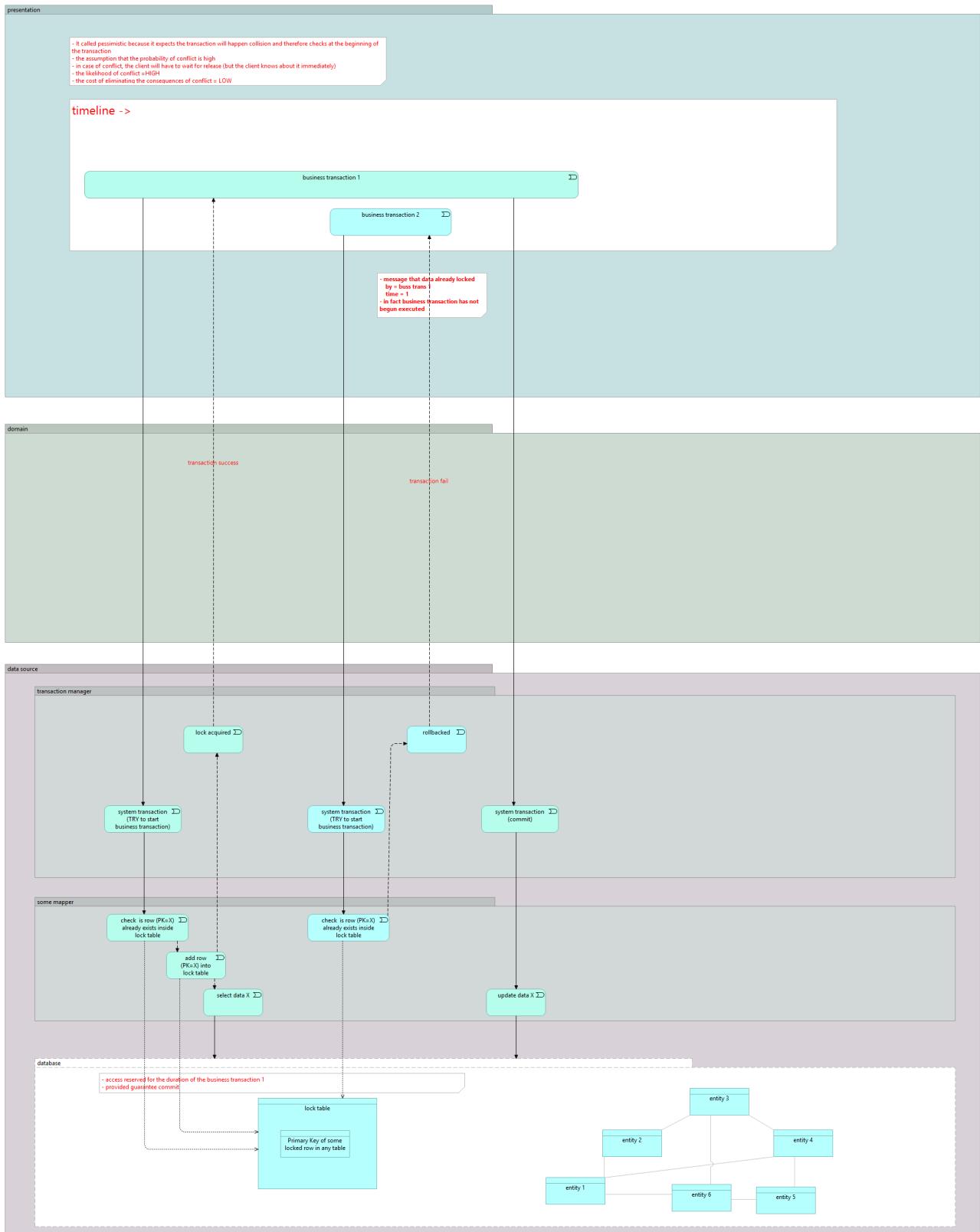
IMPLICIT LOCK



OPTIMISTIC OFFLINE LOCK



PESSIMISTIC OFFLINE LOCK



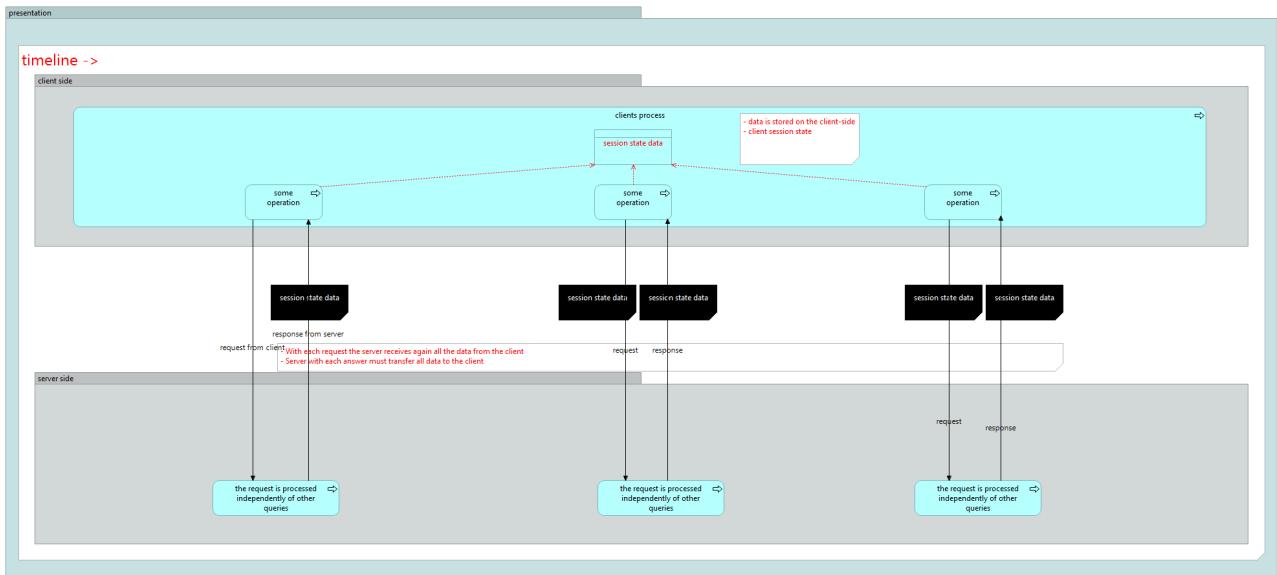
SESSION STATE

ENTERPRISE PATTERNS

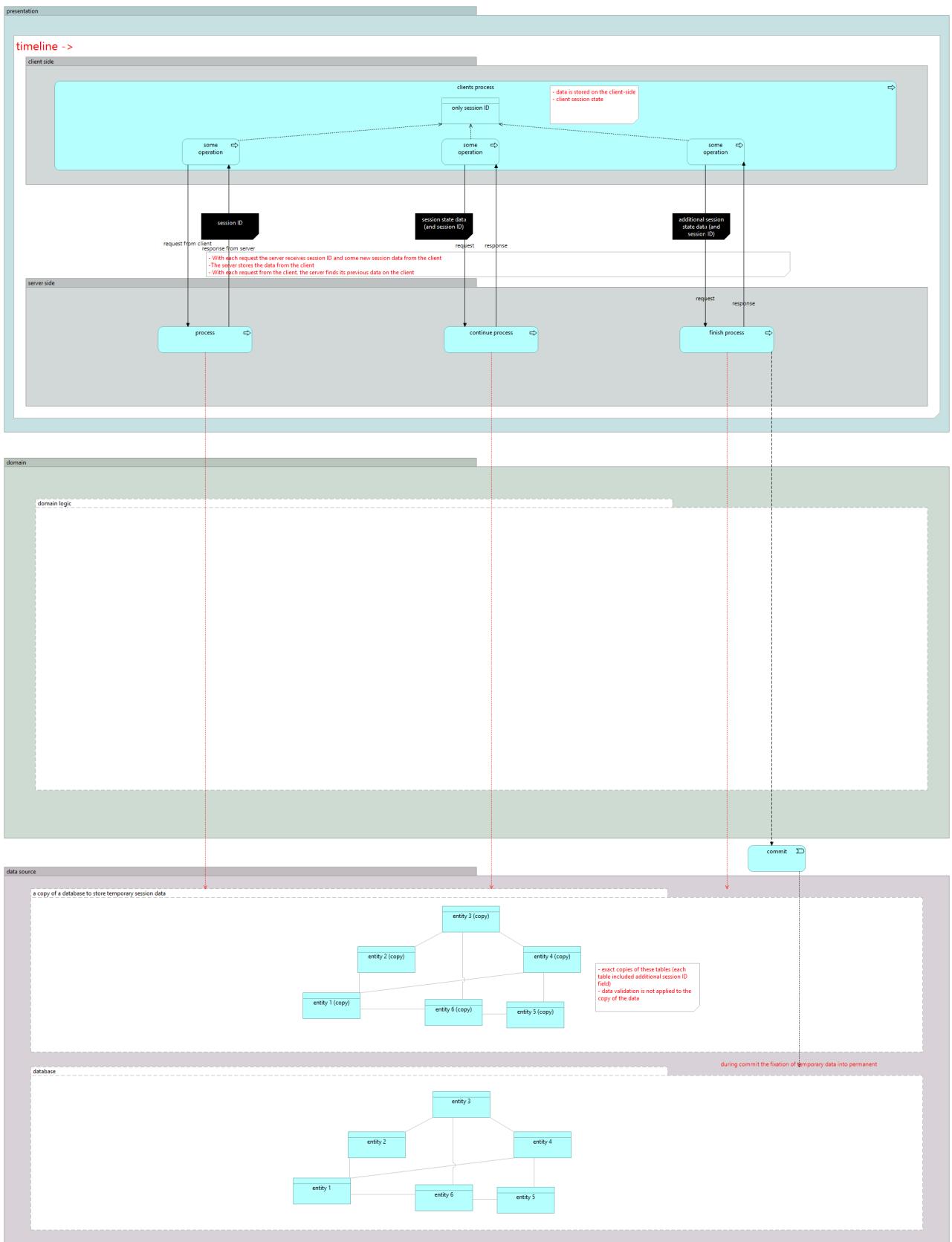
Martin Fowler



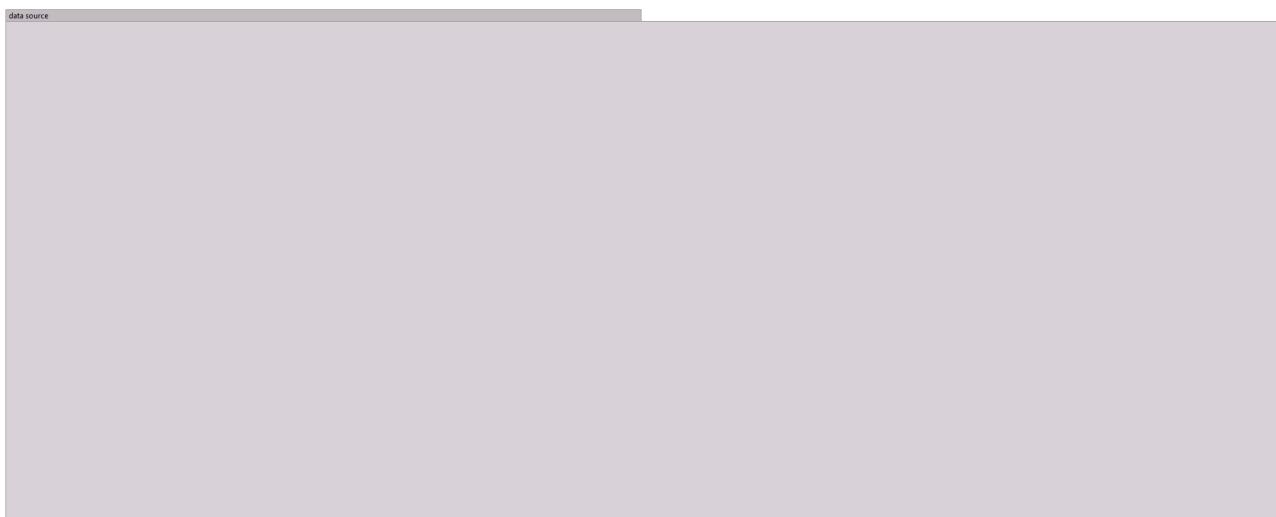
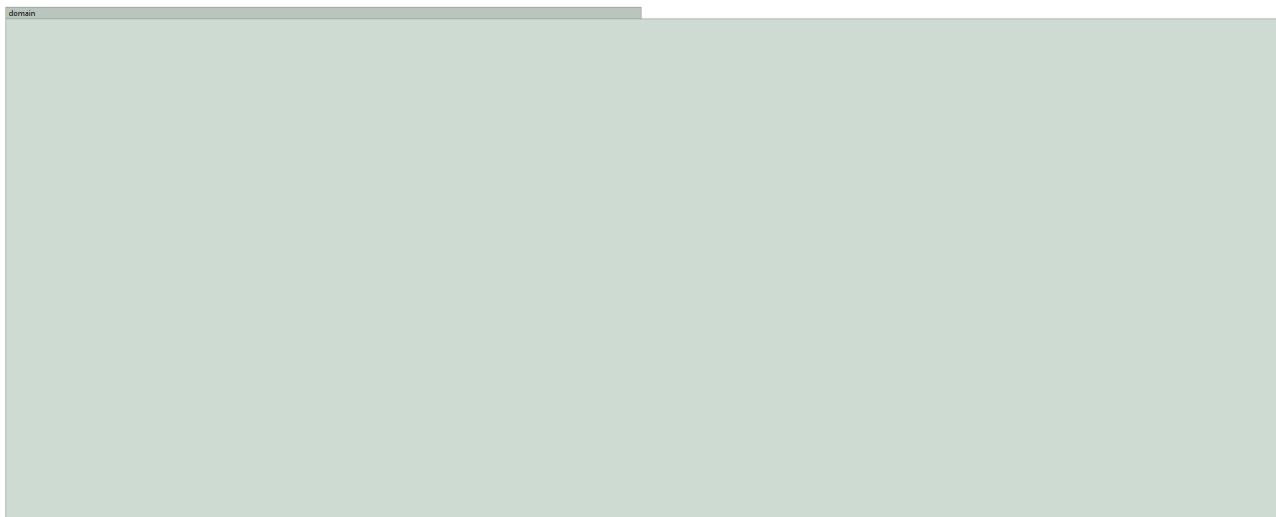
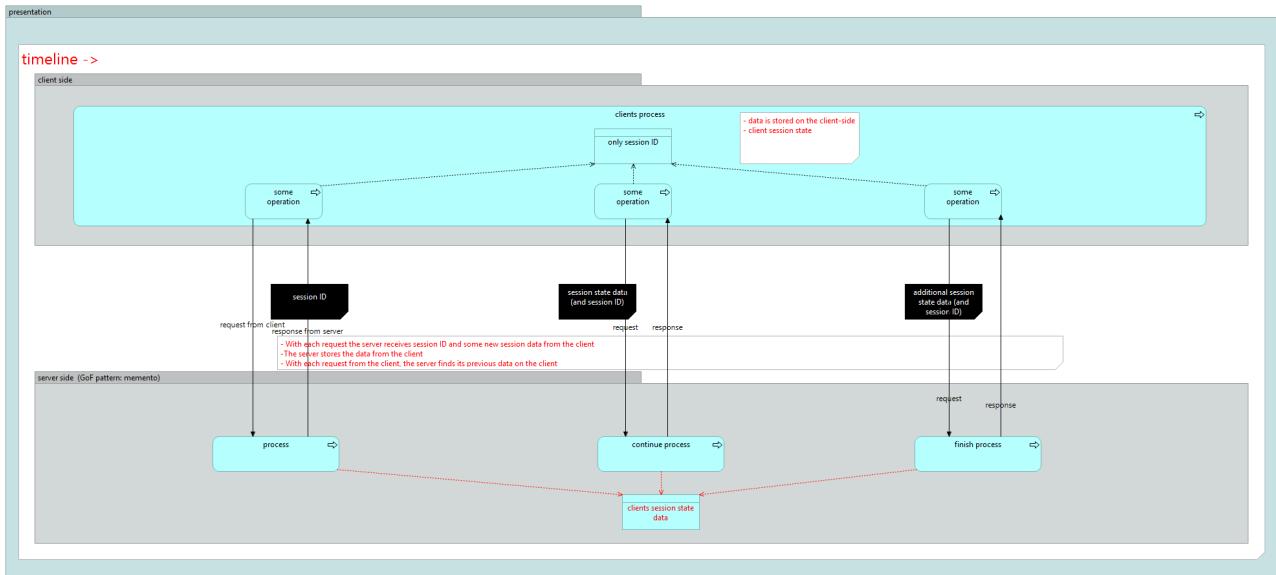
CLIENT SESSION STATE



DATABASE SESSION STATE



SERVER SESSION STATE



COMMON PATTERNS

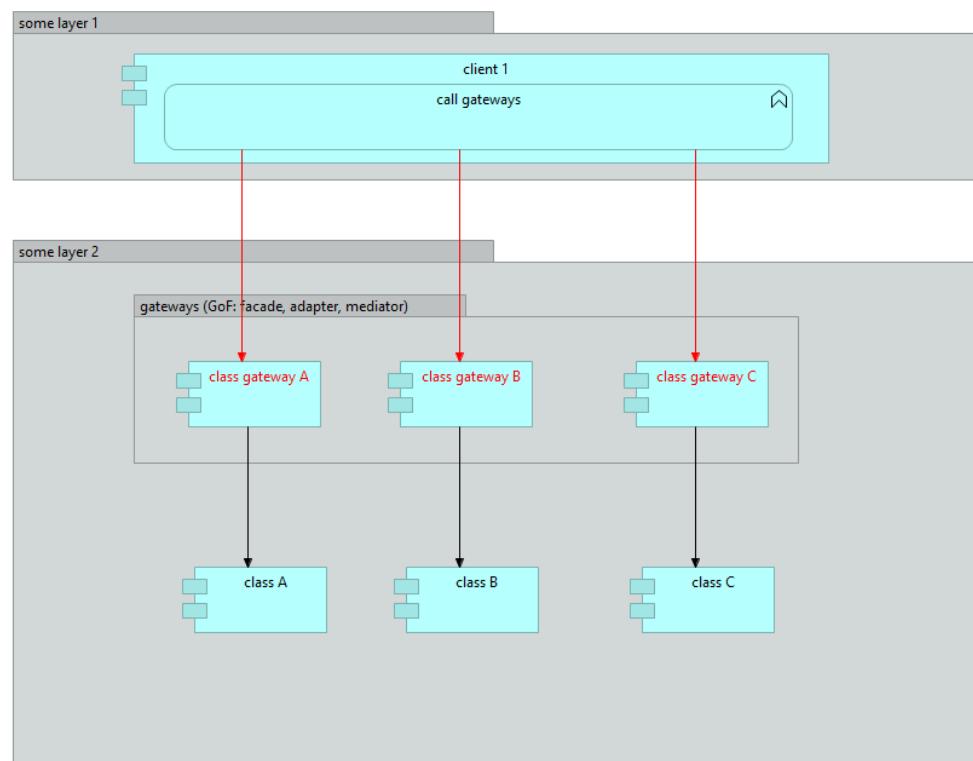
ENTERPRISE PATTERNS

Martin Fowler

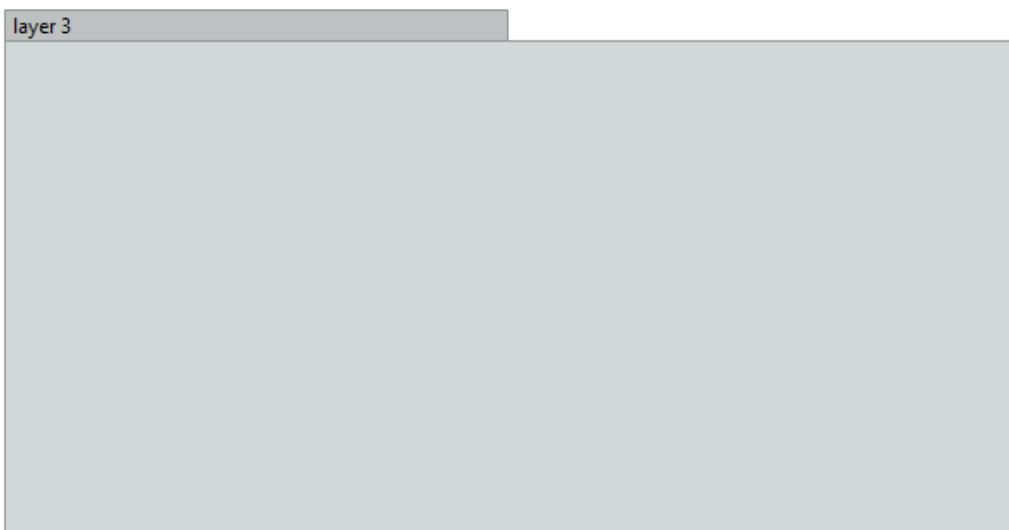
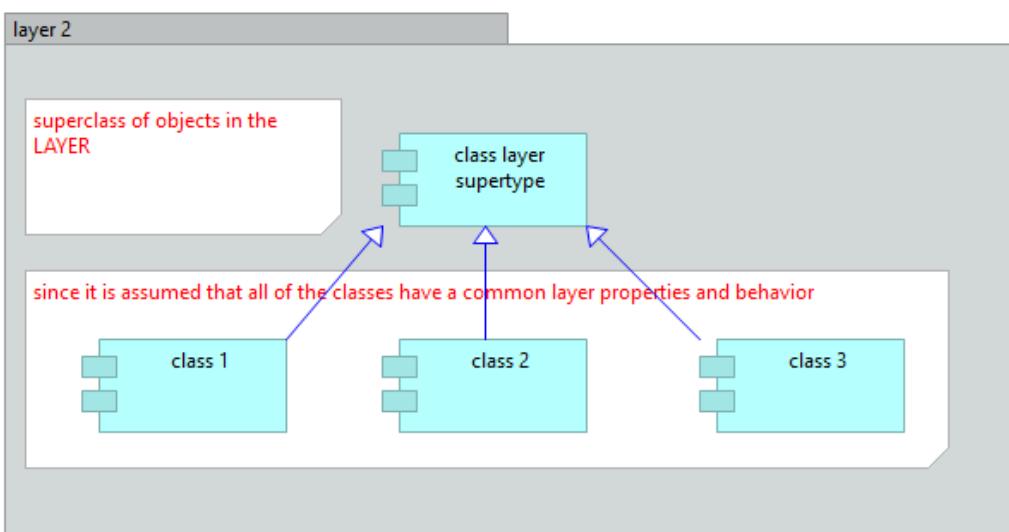
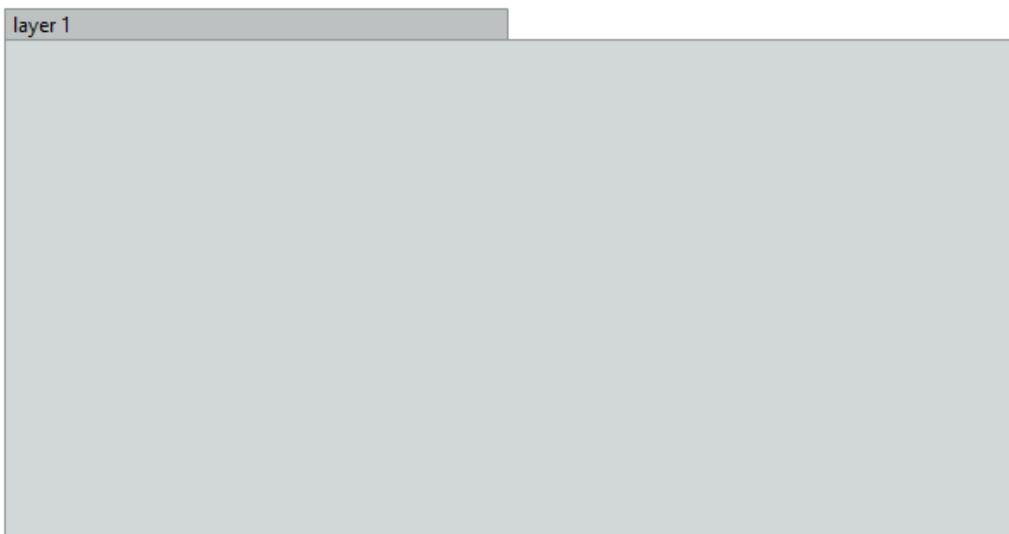


GATEWAY

- 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



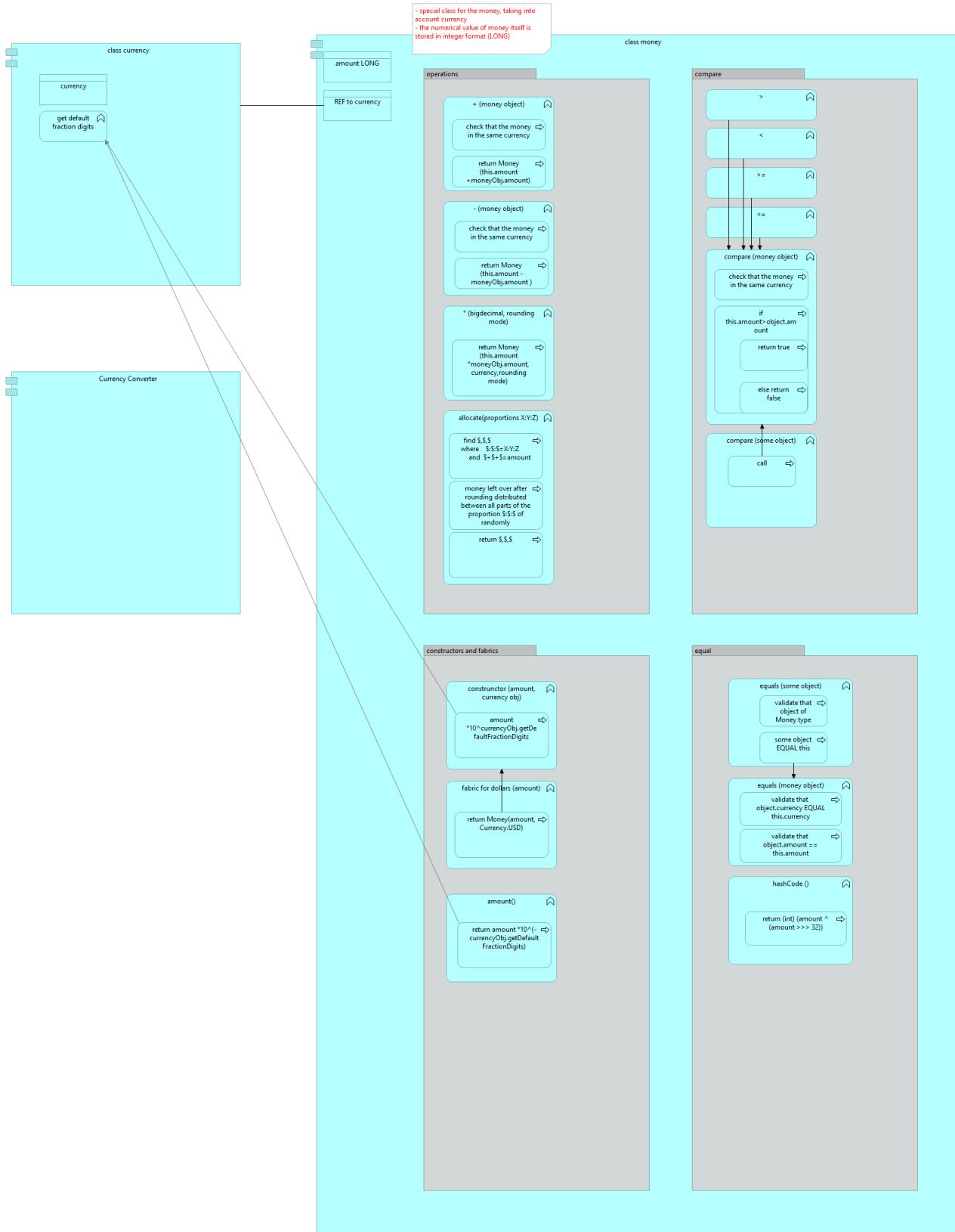
LAYER SUPERTYPE



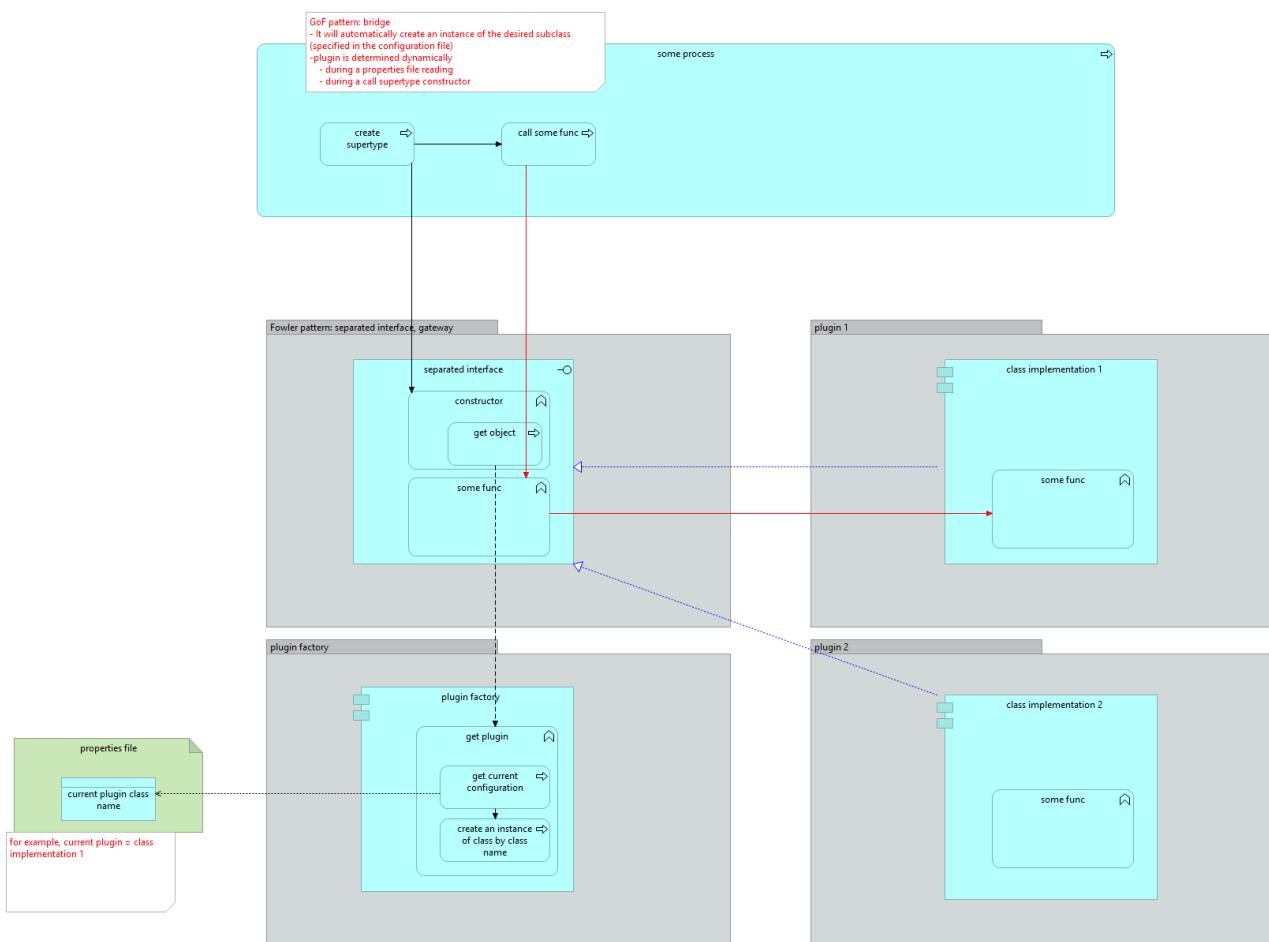
MAPPER



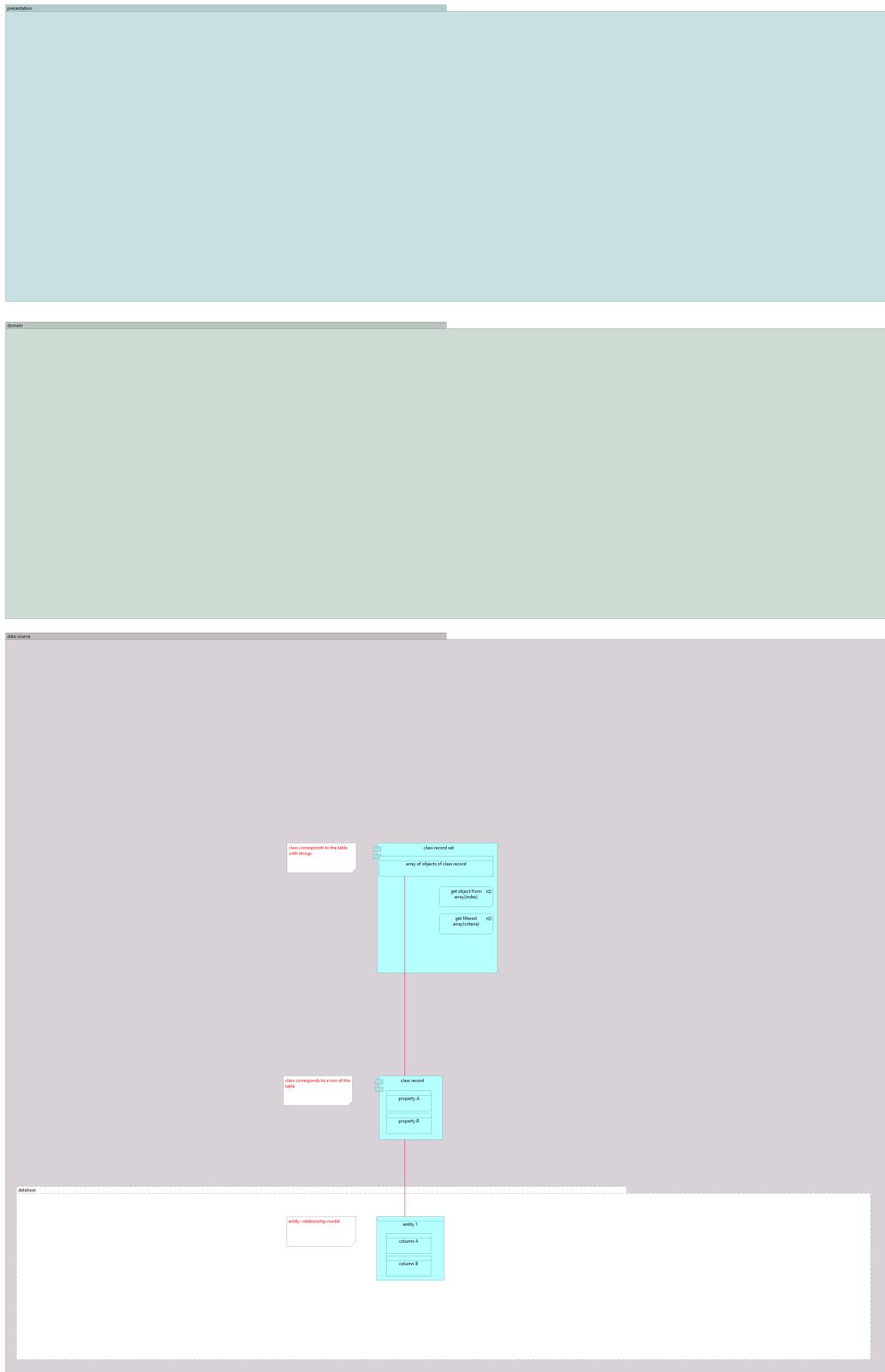
MONEY



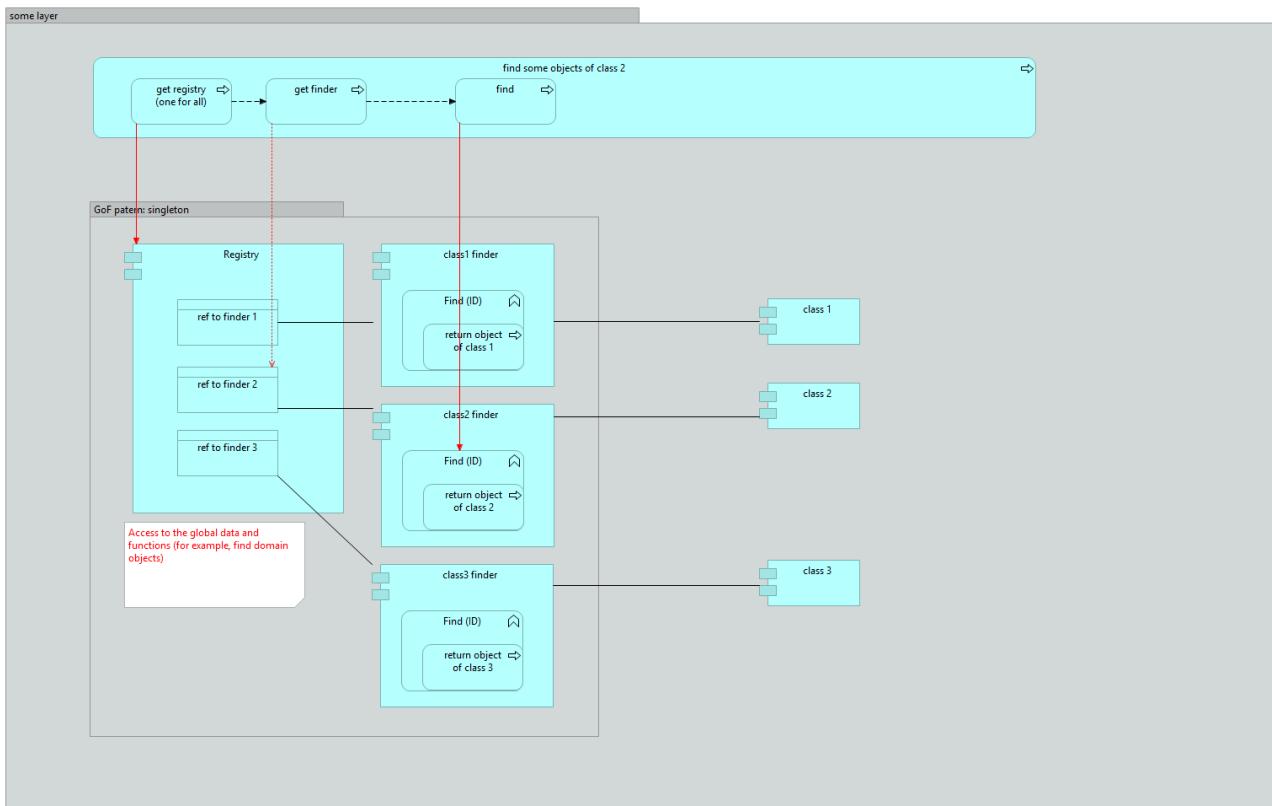
PLUGIN



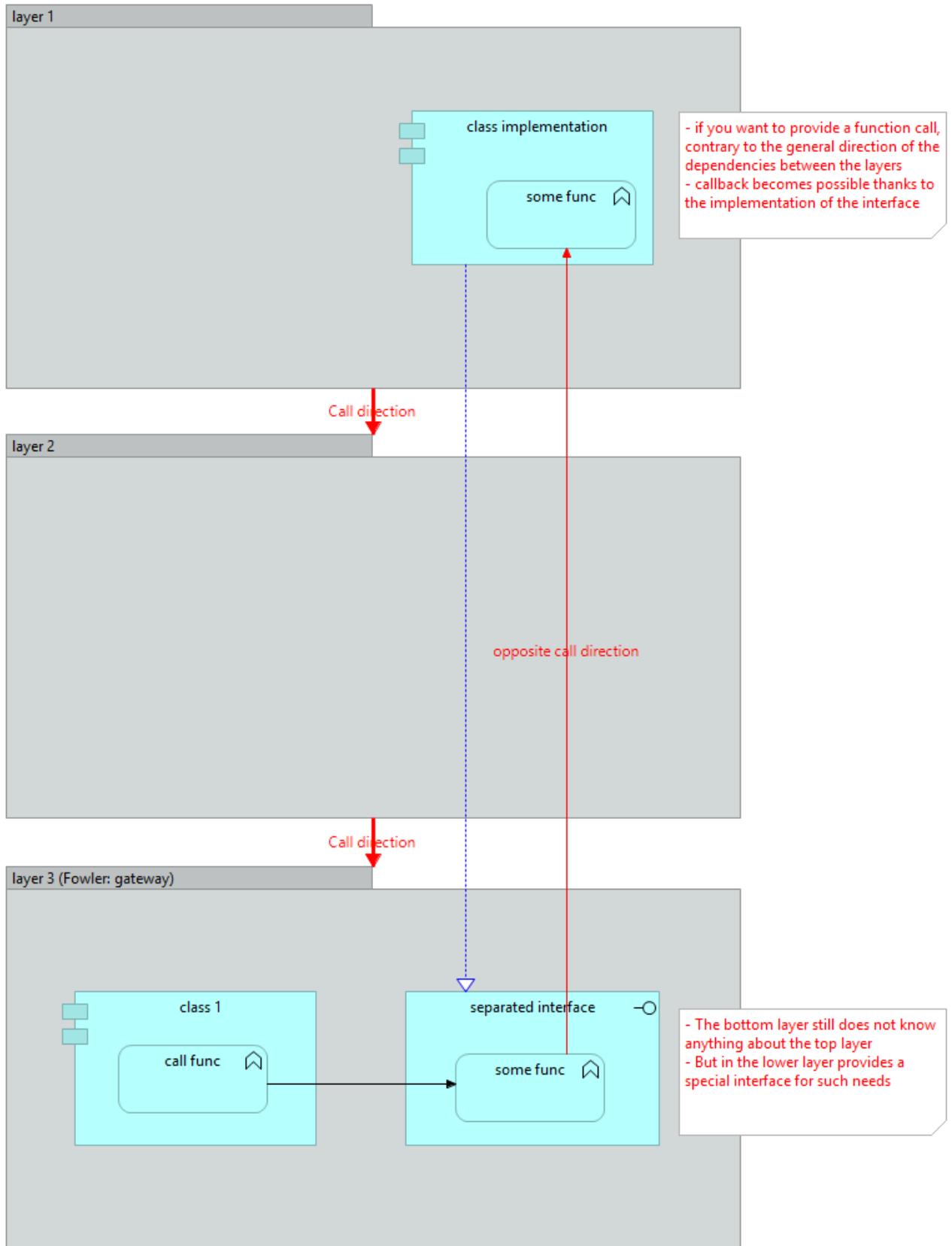
RECORD SET



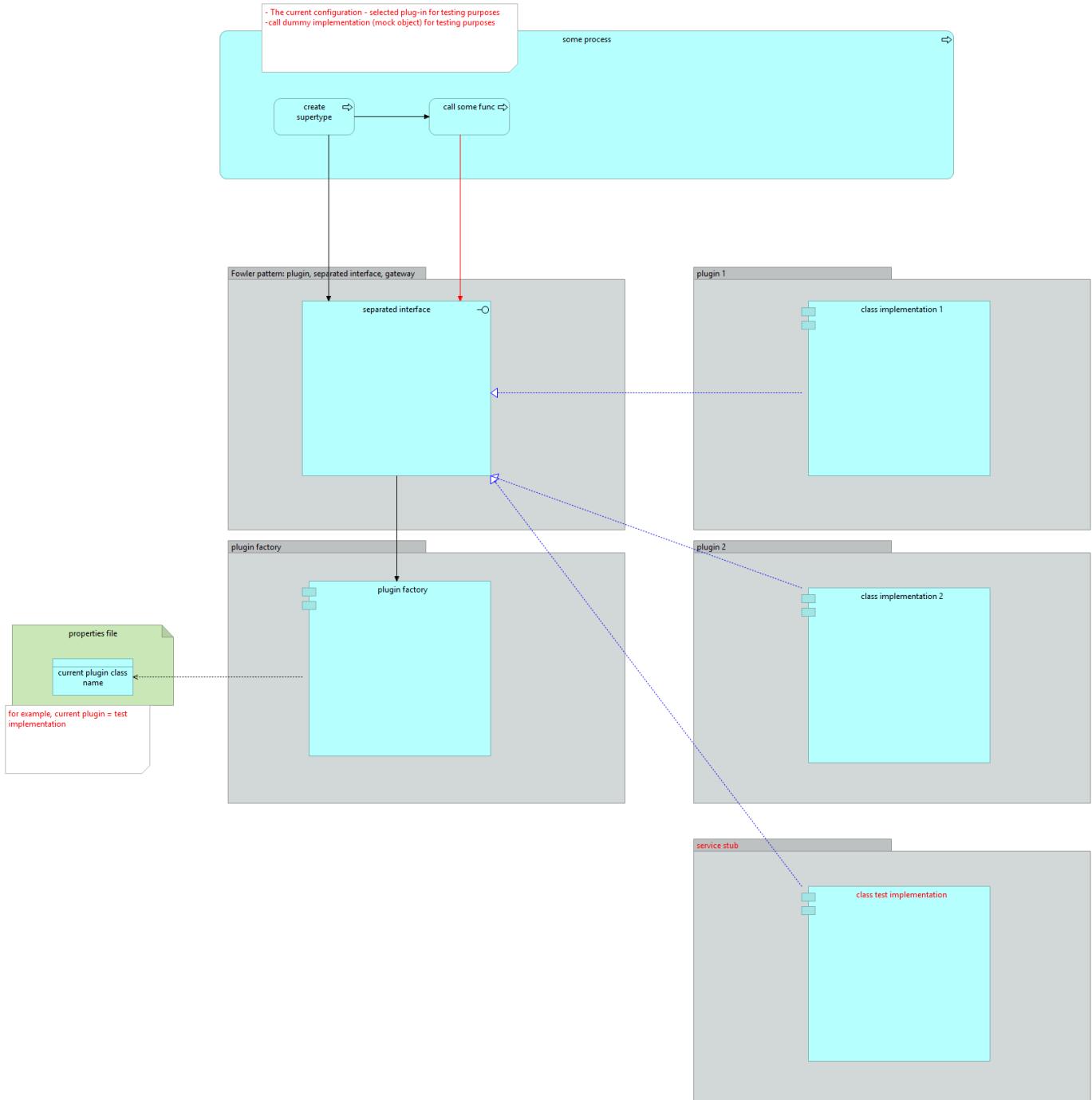
REGISTRY



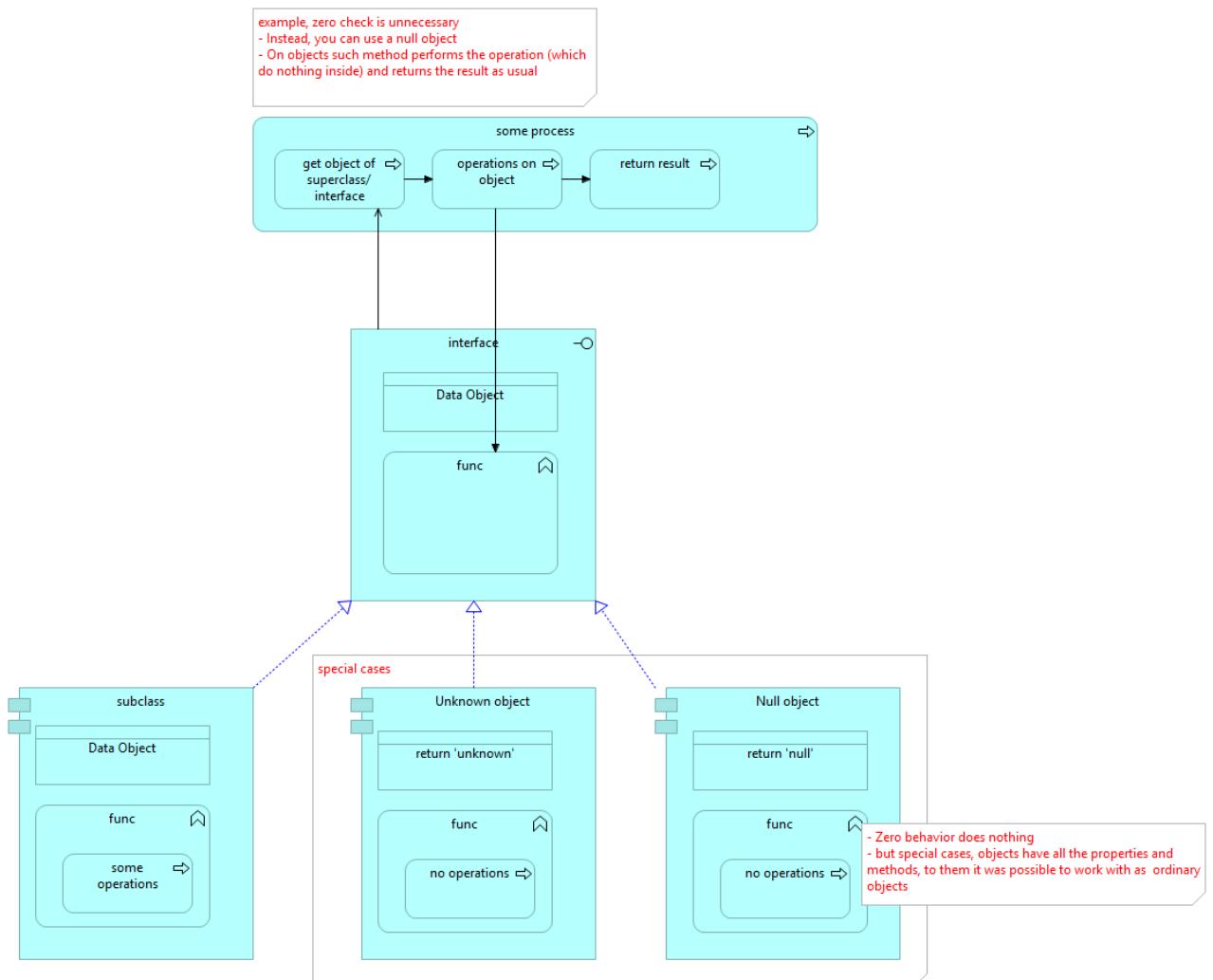
SEPARATED INTERFACE



SERVICE STUB

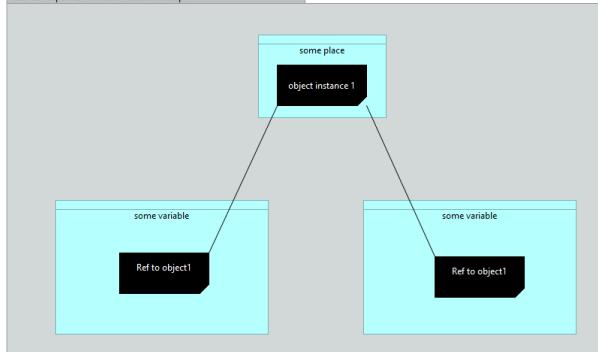


SPECIAL CASE

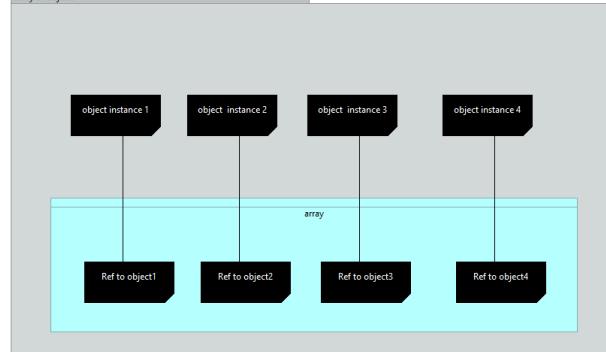


VALUE OBJECT

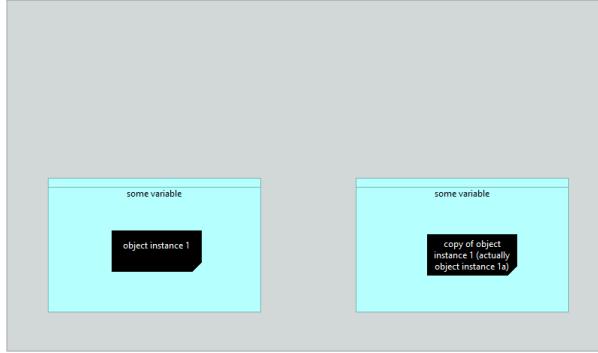
class concept and link transmission concept



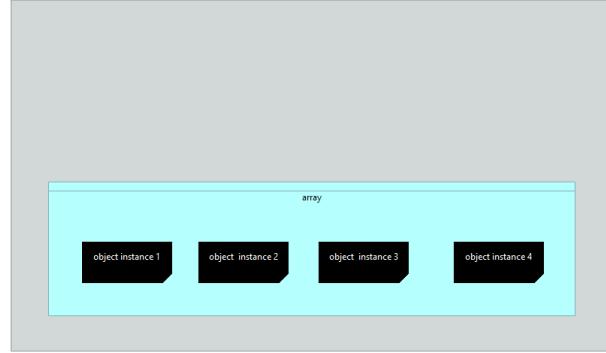
array of objects



value object concept and by value transfer concept



array of value objects

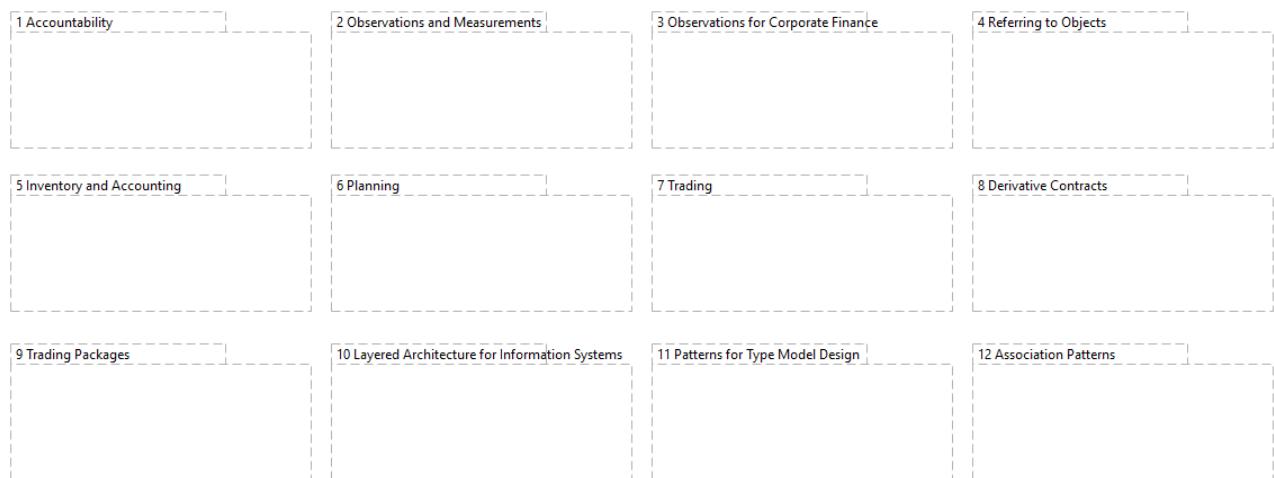


03

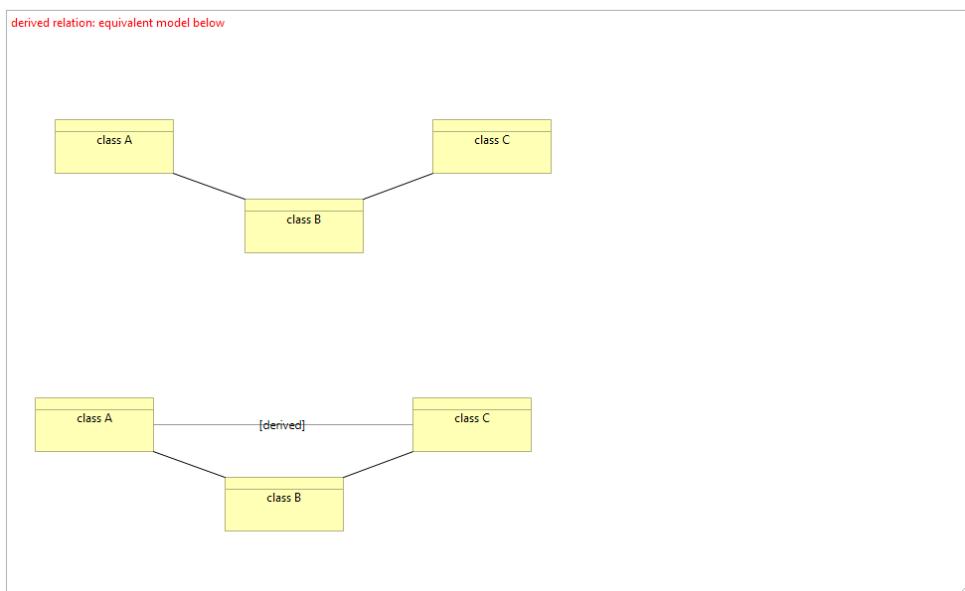
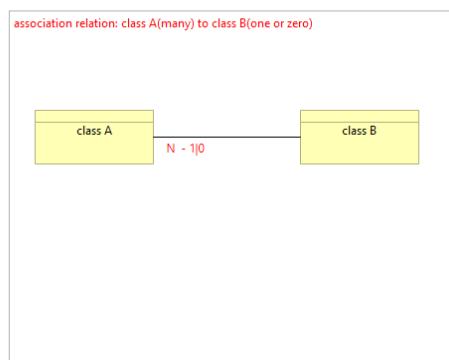
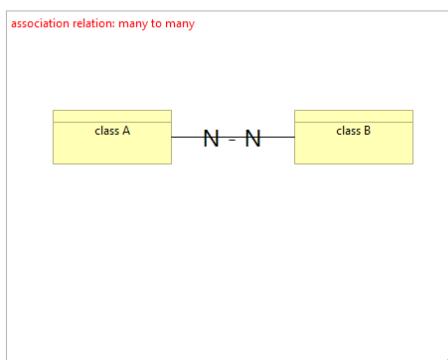
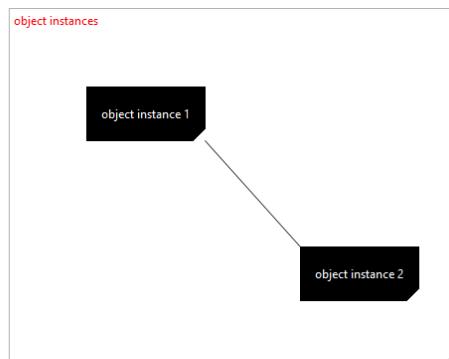
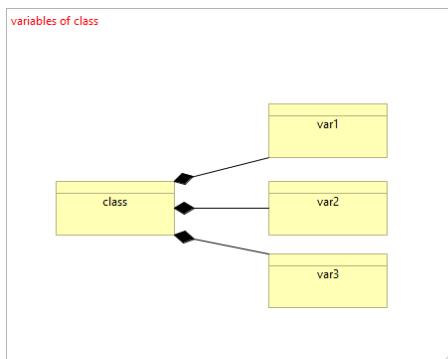
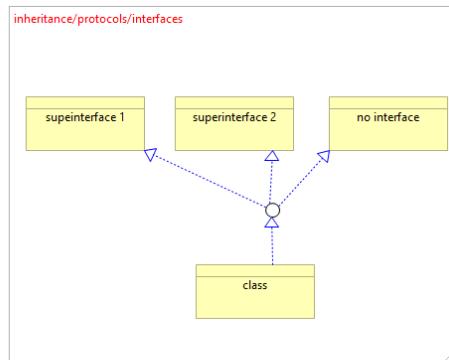
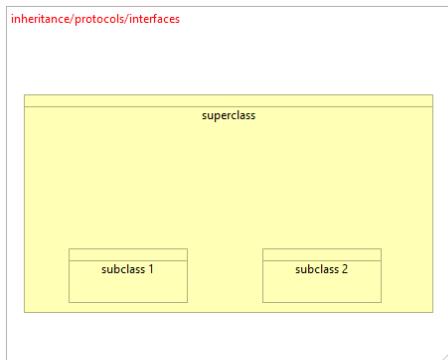
Analysis Patterns

Reusable Object Models

MARTIN FOWLER



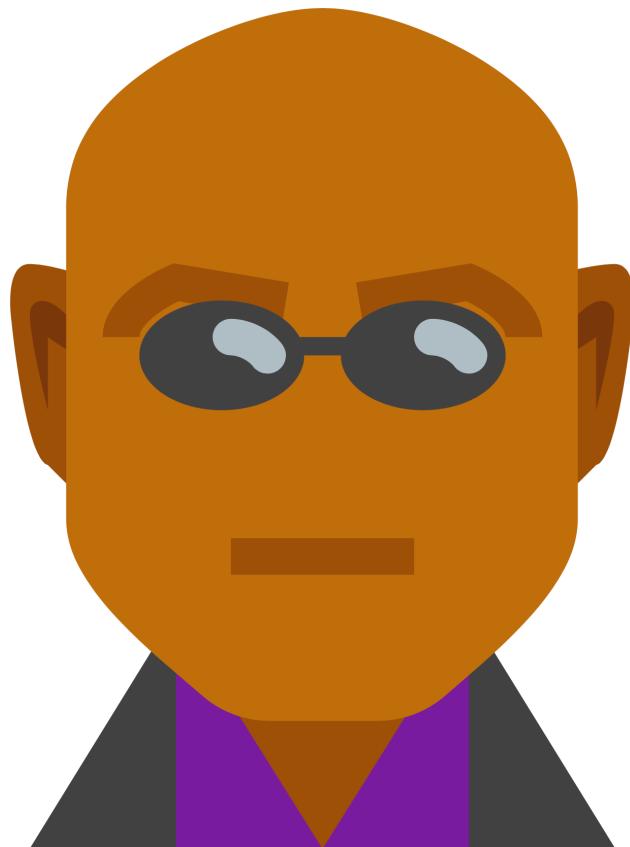
USED NOTATION



ACCOUNTABILITY

ANALYSIS PATTERNS

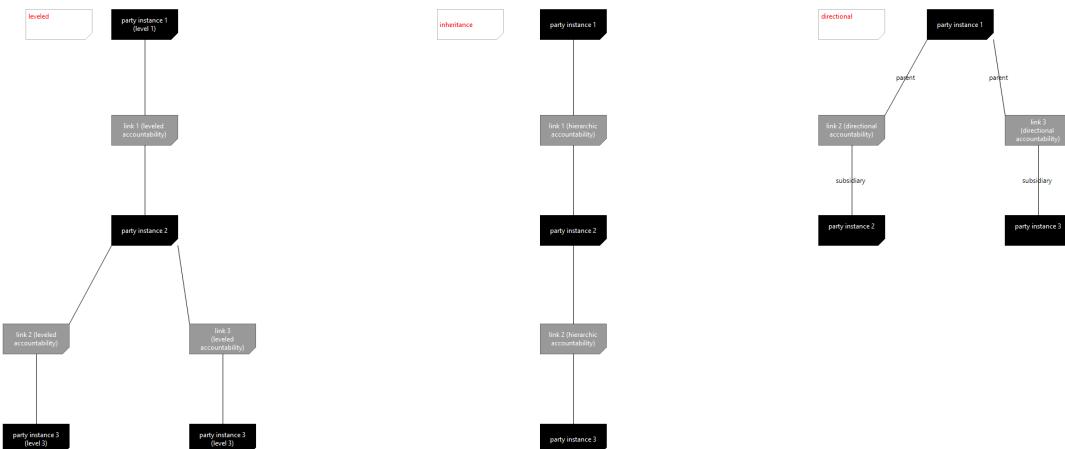
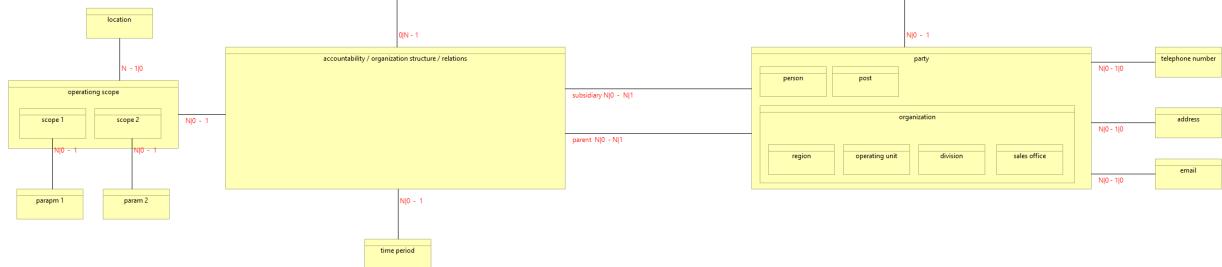
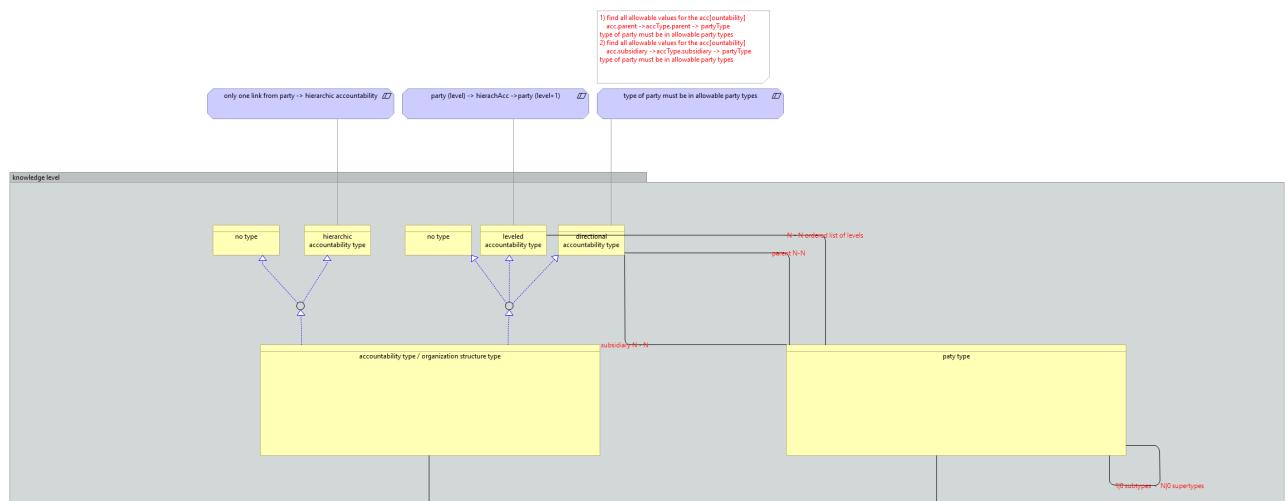
Martin Fowler

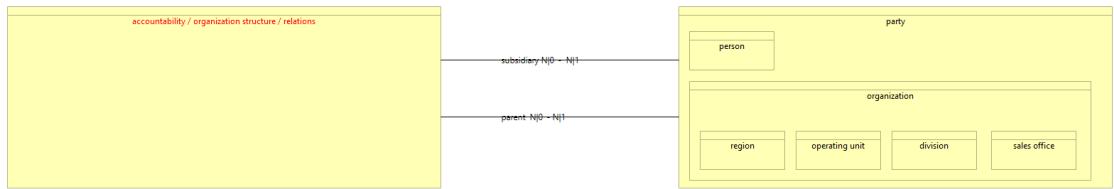


PARTY



ACCOUNTABILITY

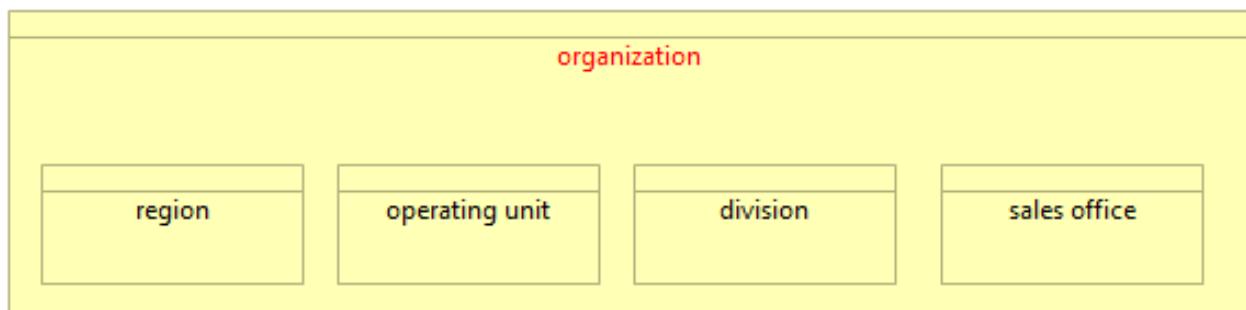




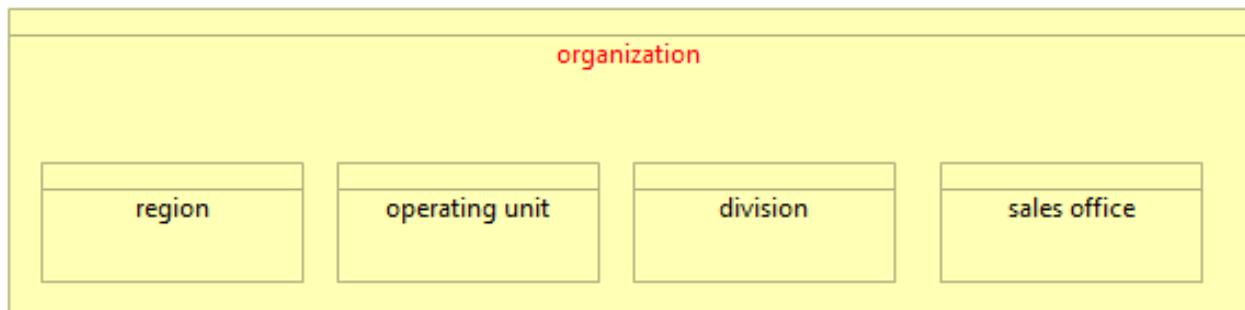
time period

```
graph LR; K[time period]
```

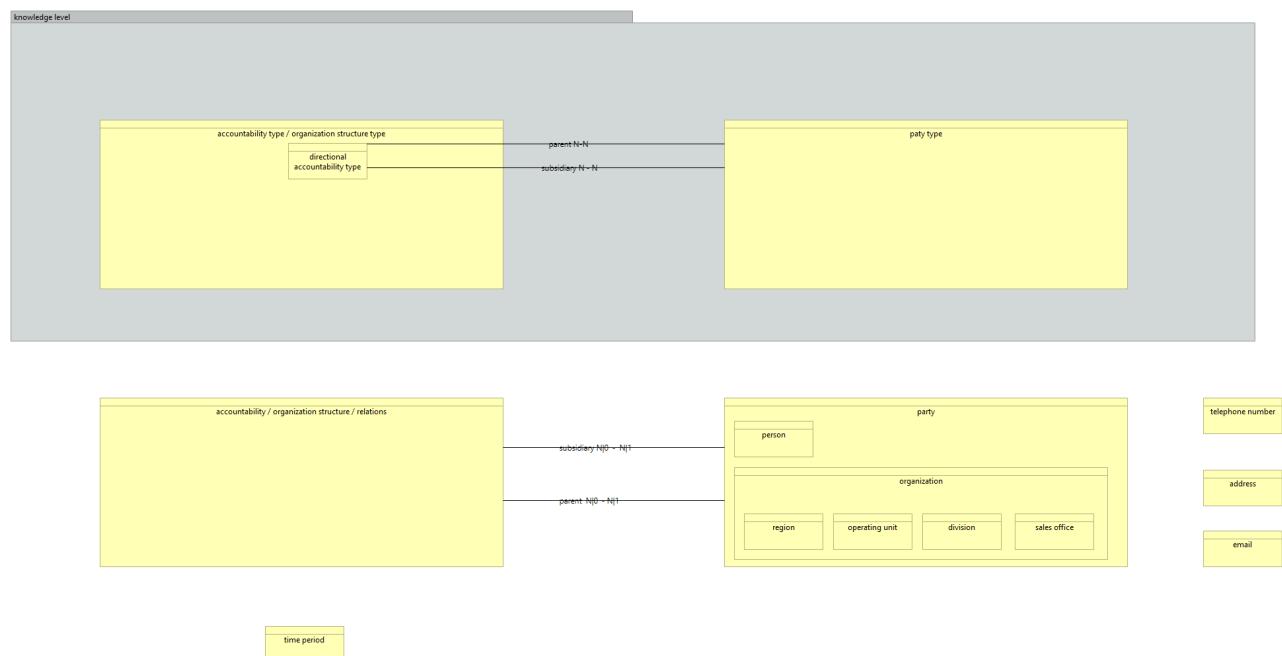
ORGANIZATION HIERARCHIES



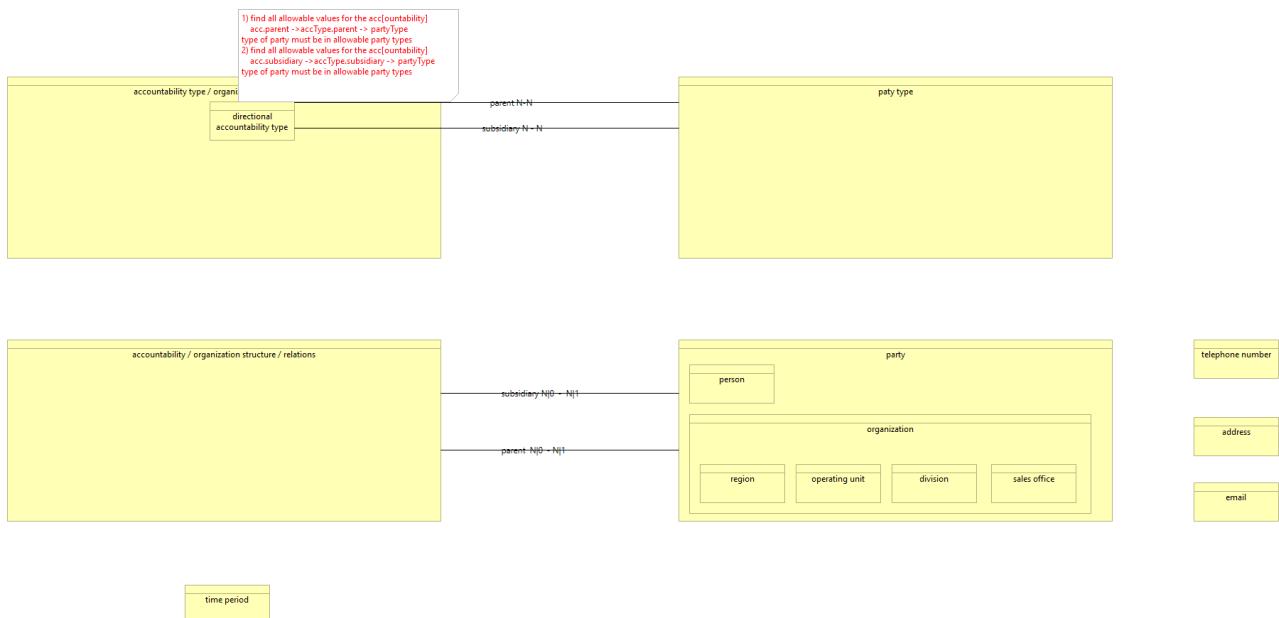
ORGANIZATION STRUCTURE



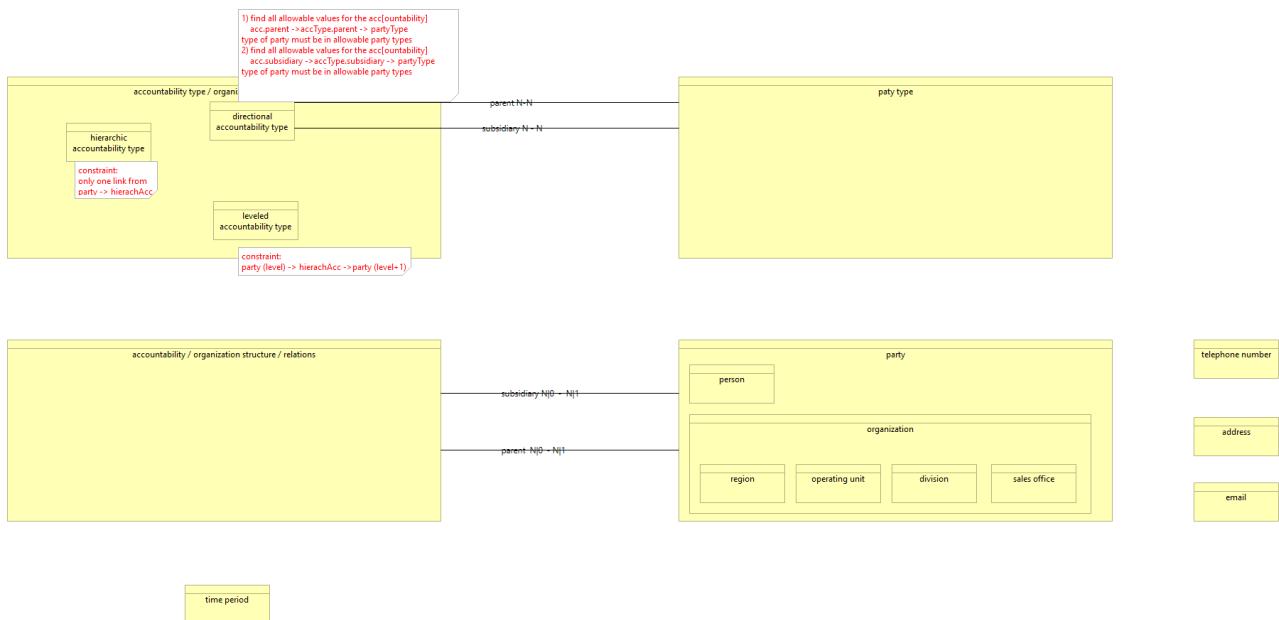
ACCOUNTABILITY KNOWLEDGE LEVEL



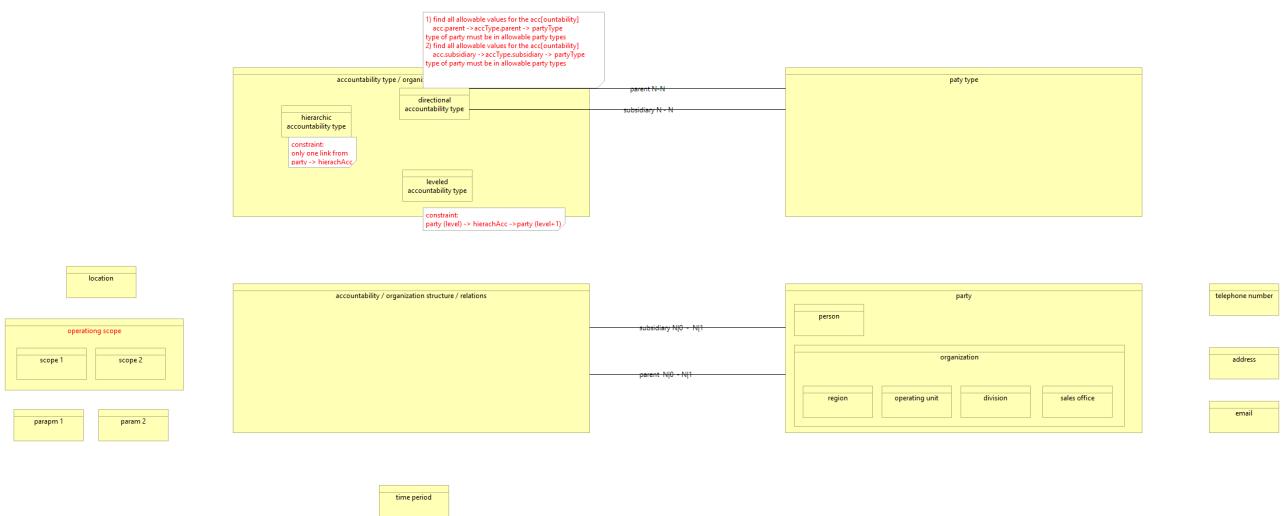
PARTY TYPE GENERALIZATIONS



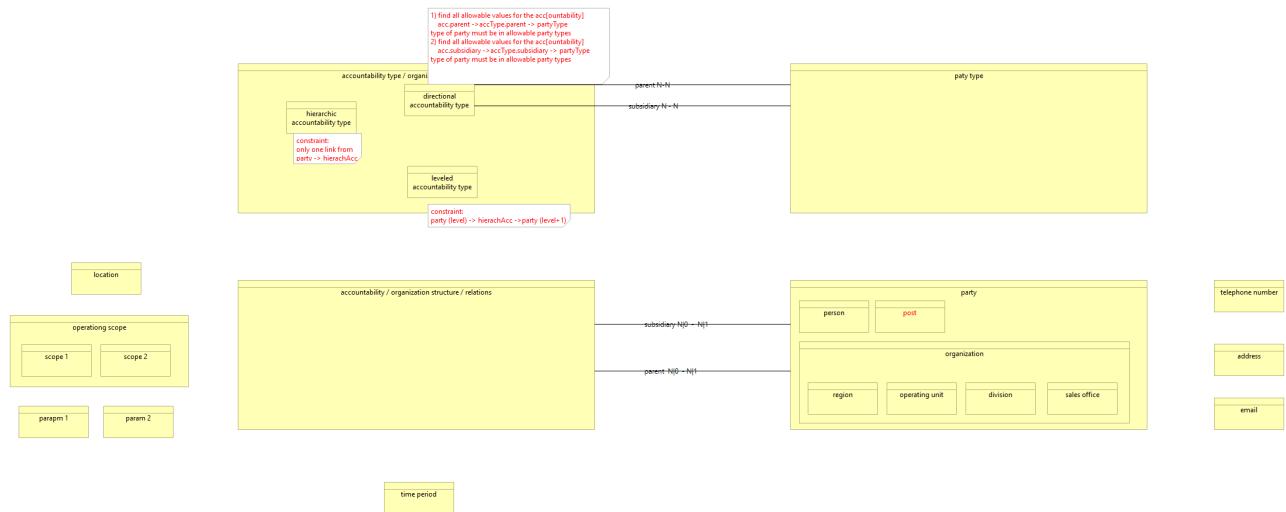
HIERARCHIC ACCOUNTABILITY



OPERATING SCOPES



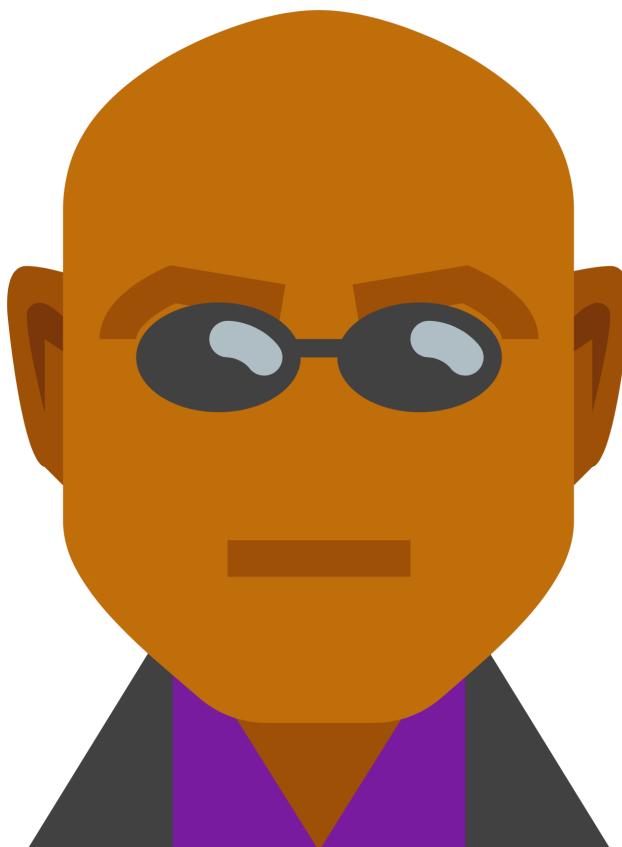
POST



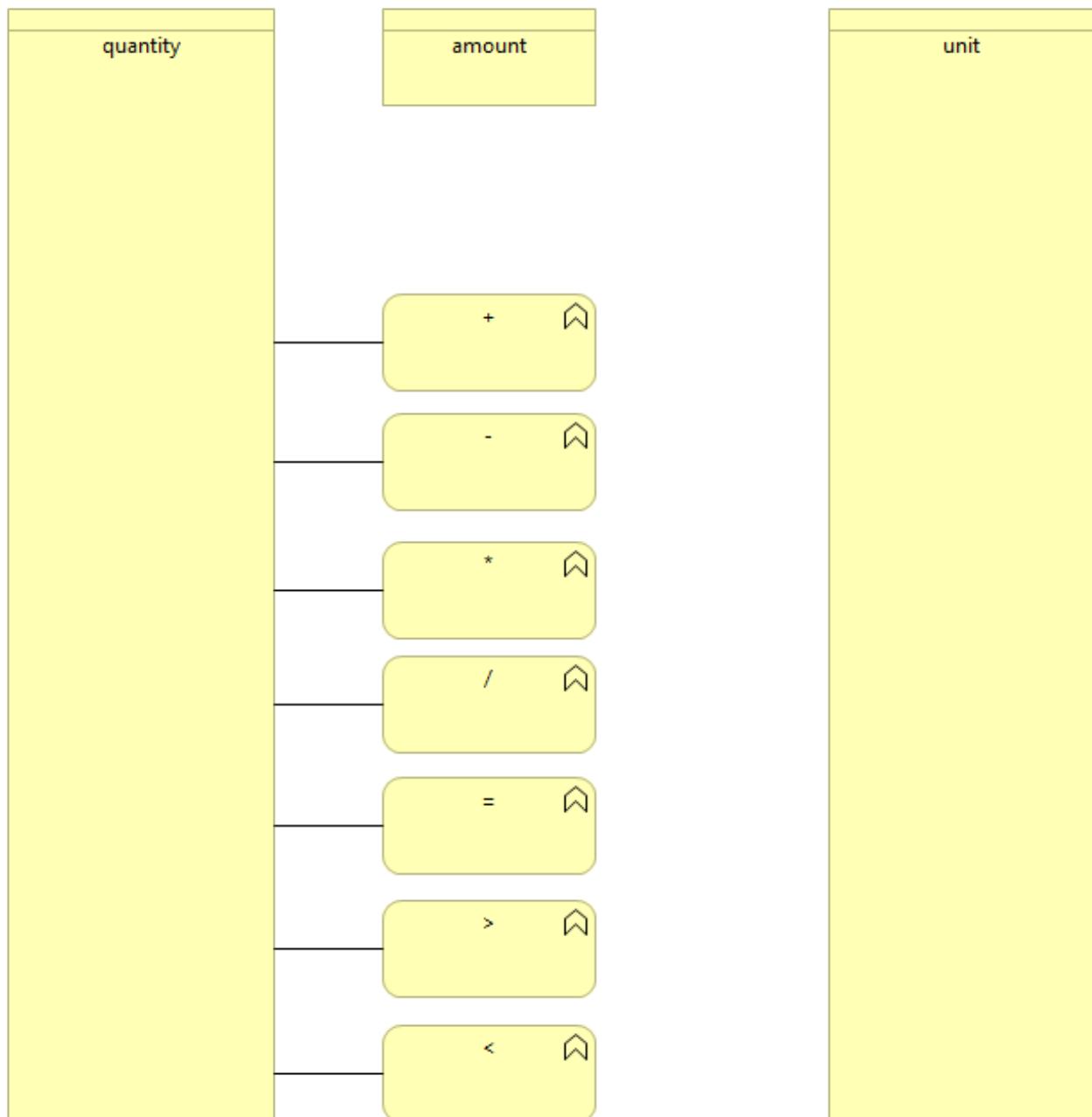
OBSERVATIONS AND MEASUREMENTS

ANALYSIS PATTERNS

Martin Fowler



QUANTITY

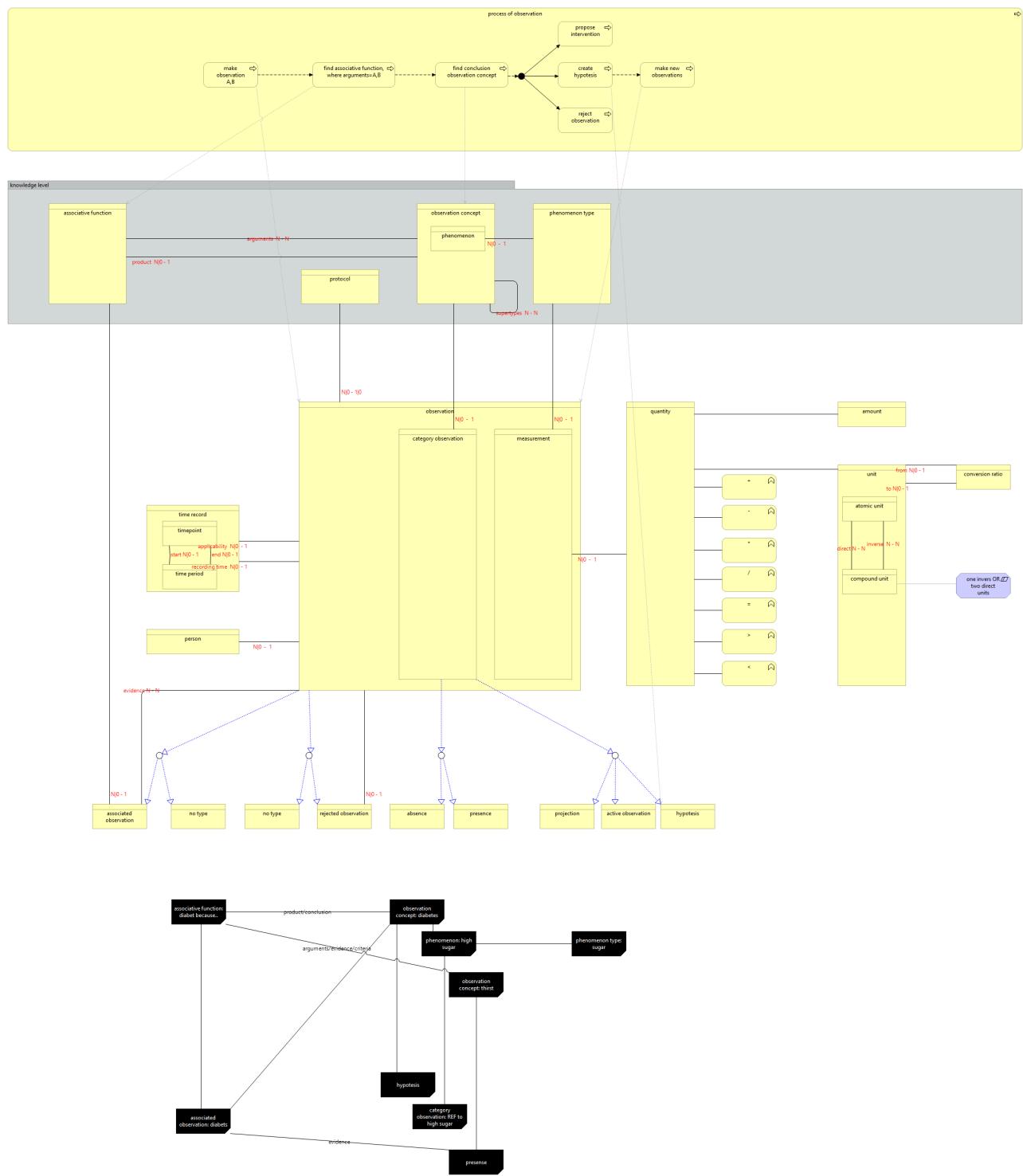


CONVERSION RATIO

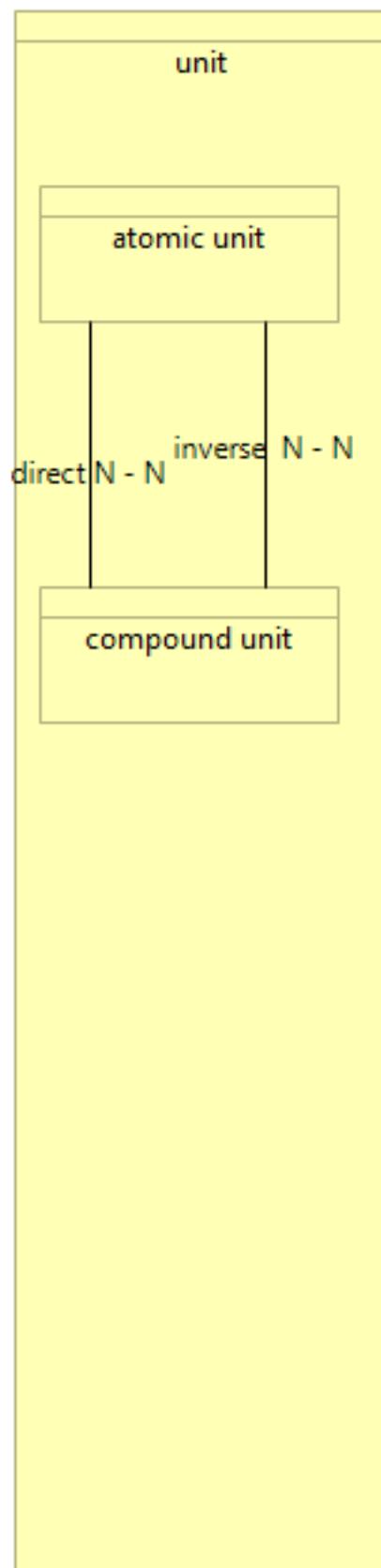
unit

conversion ratio

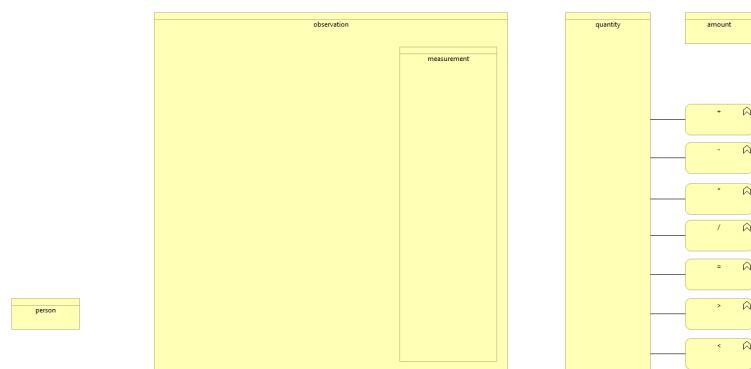
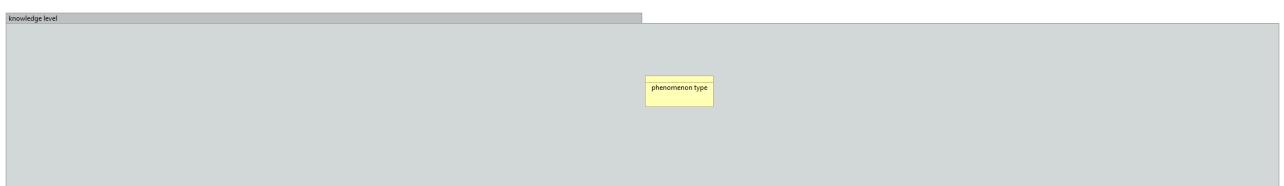
OBSERVATIONS AND MEASUREMENTS



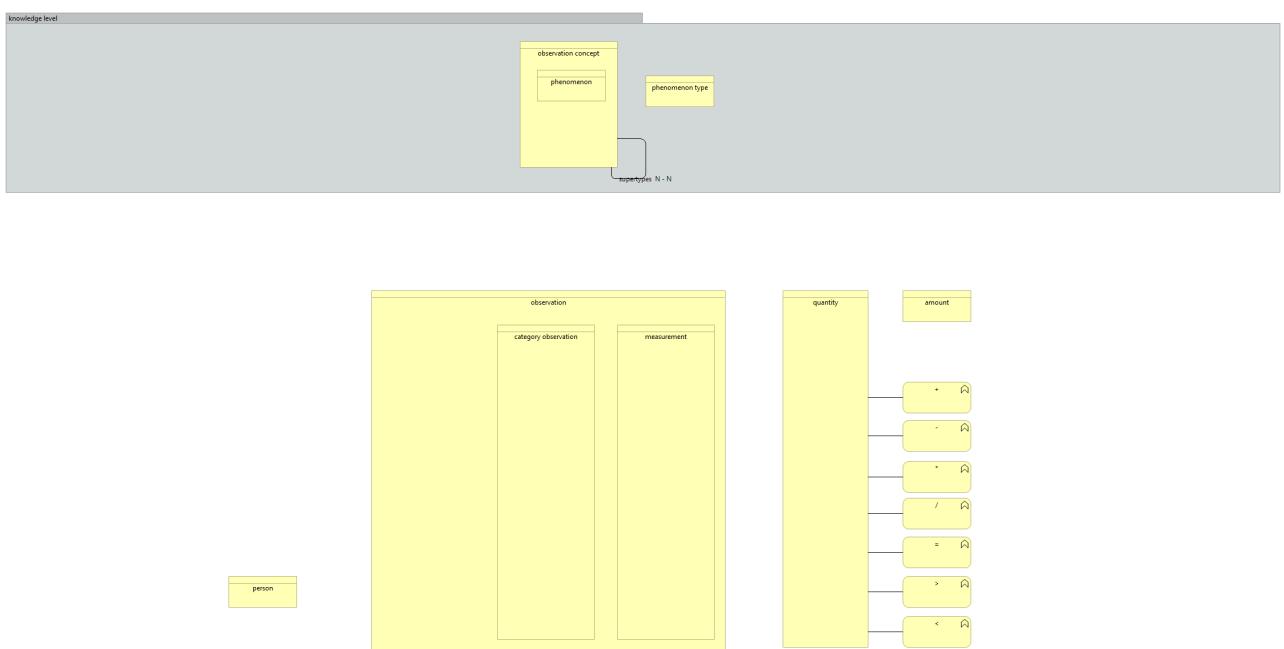
COMPOUND UNITS



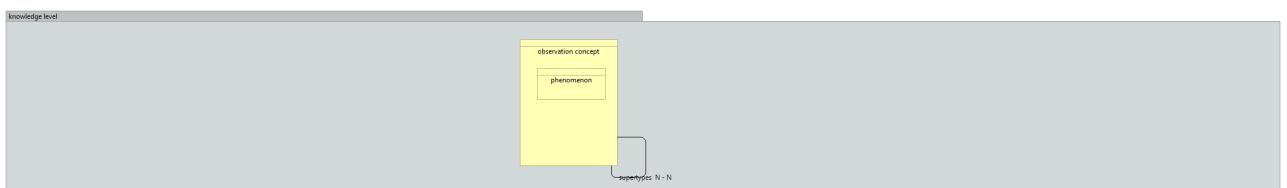
MEASUREMENT



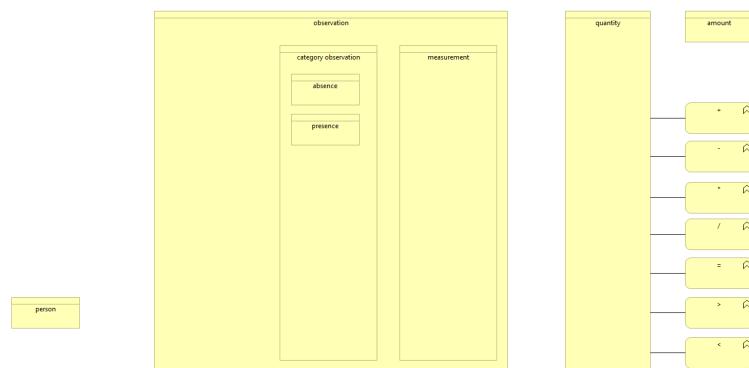
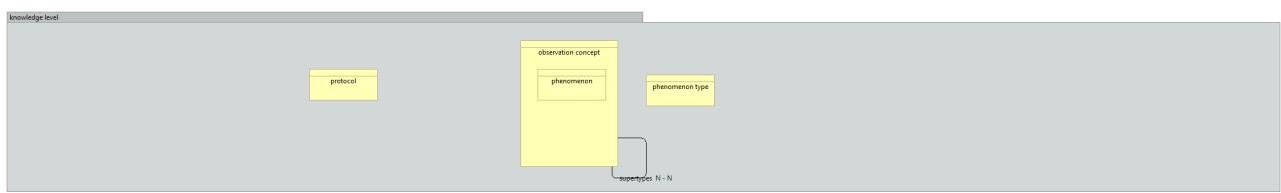
OBSERVATION



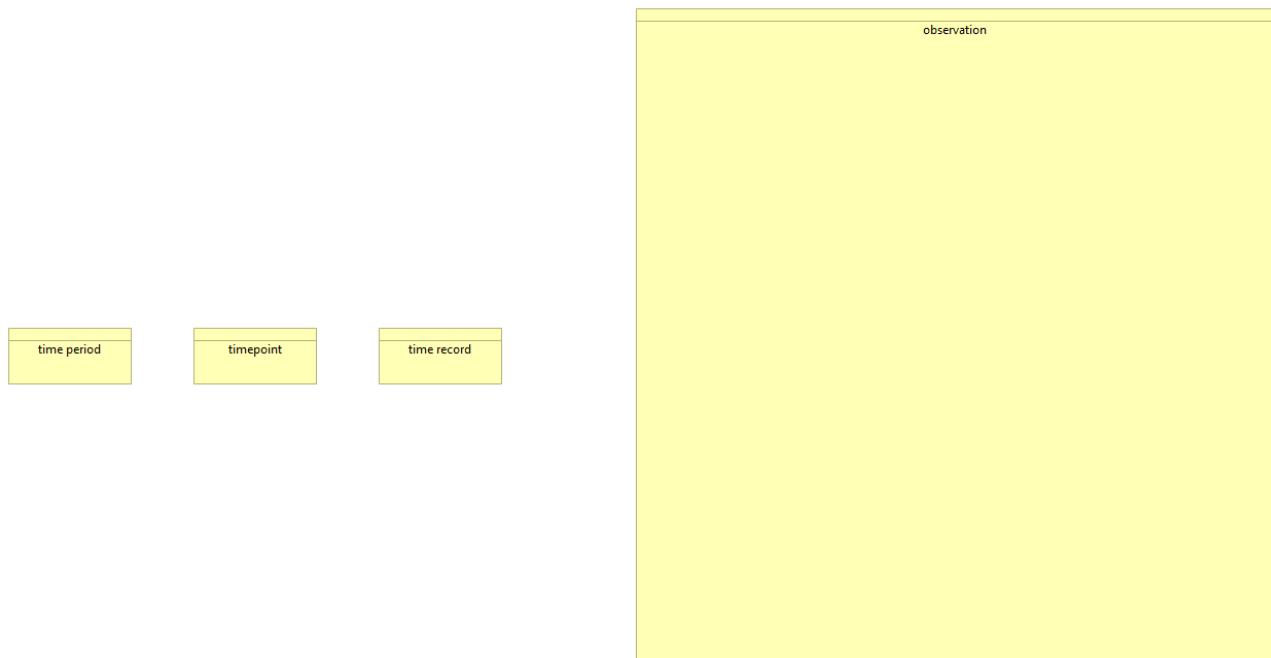
SUBTYPING OBSERVATION CONCEPTS



PROTOCOL



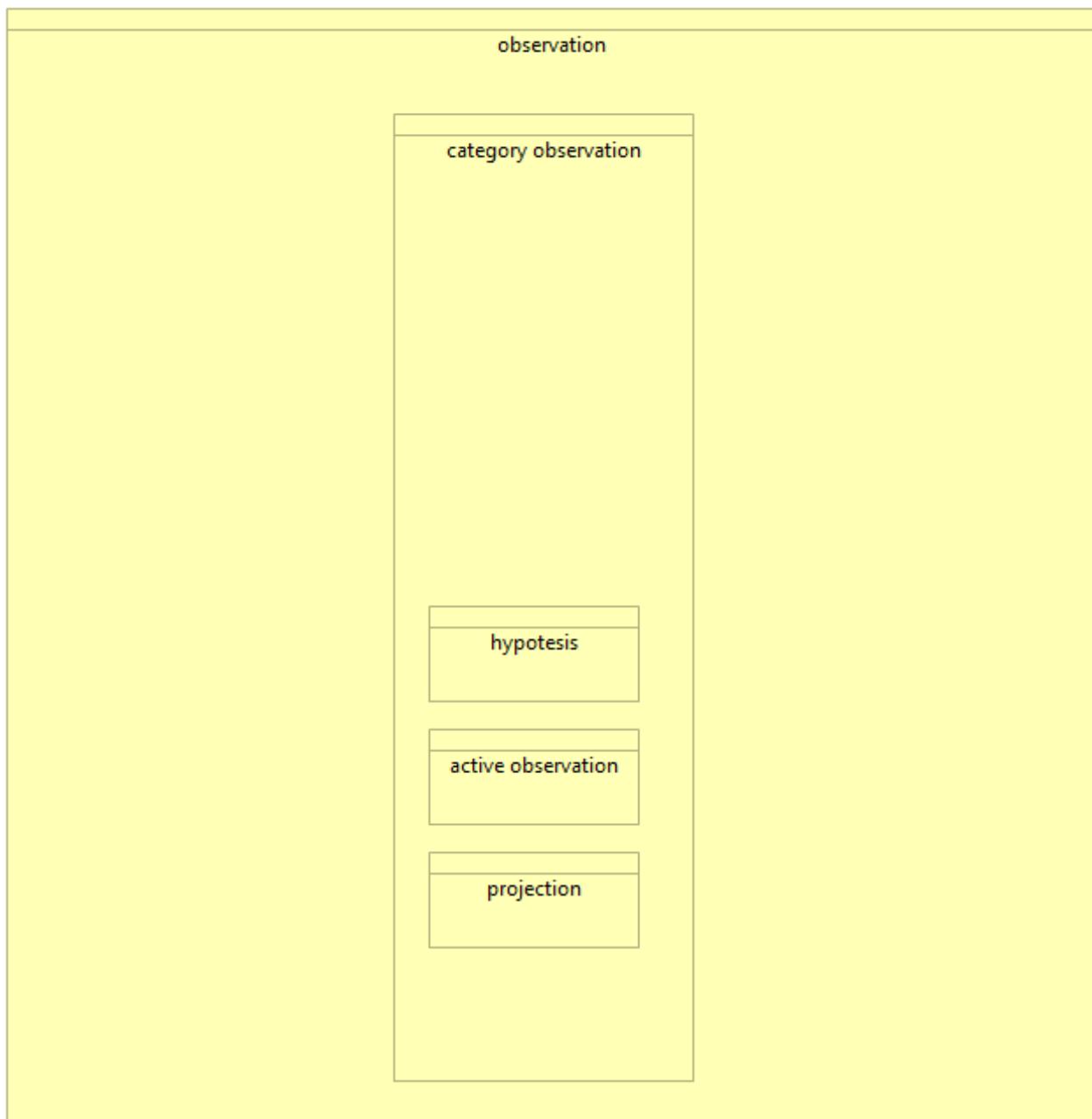
DUAL TIME RECORD



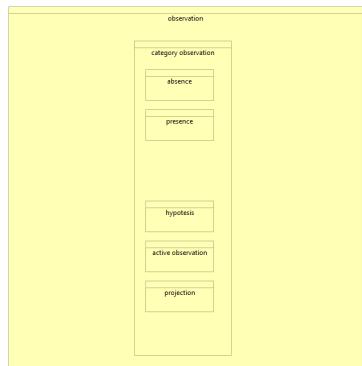
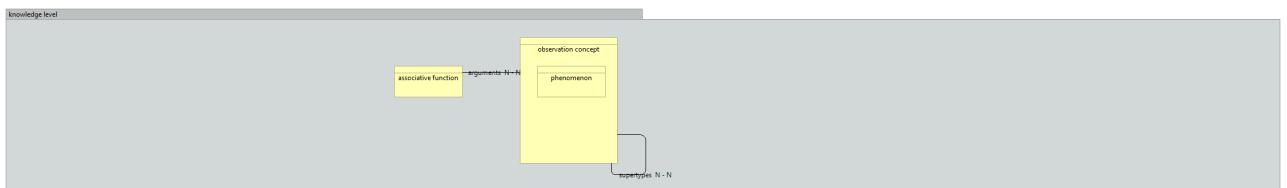
REJECTED OBSERVATION

observation

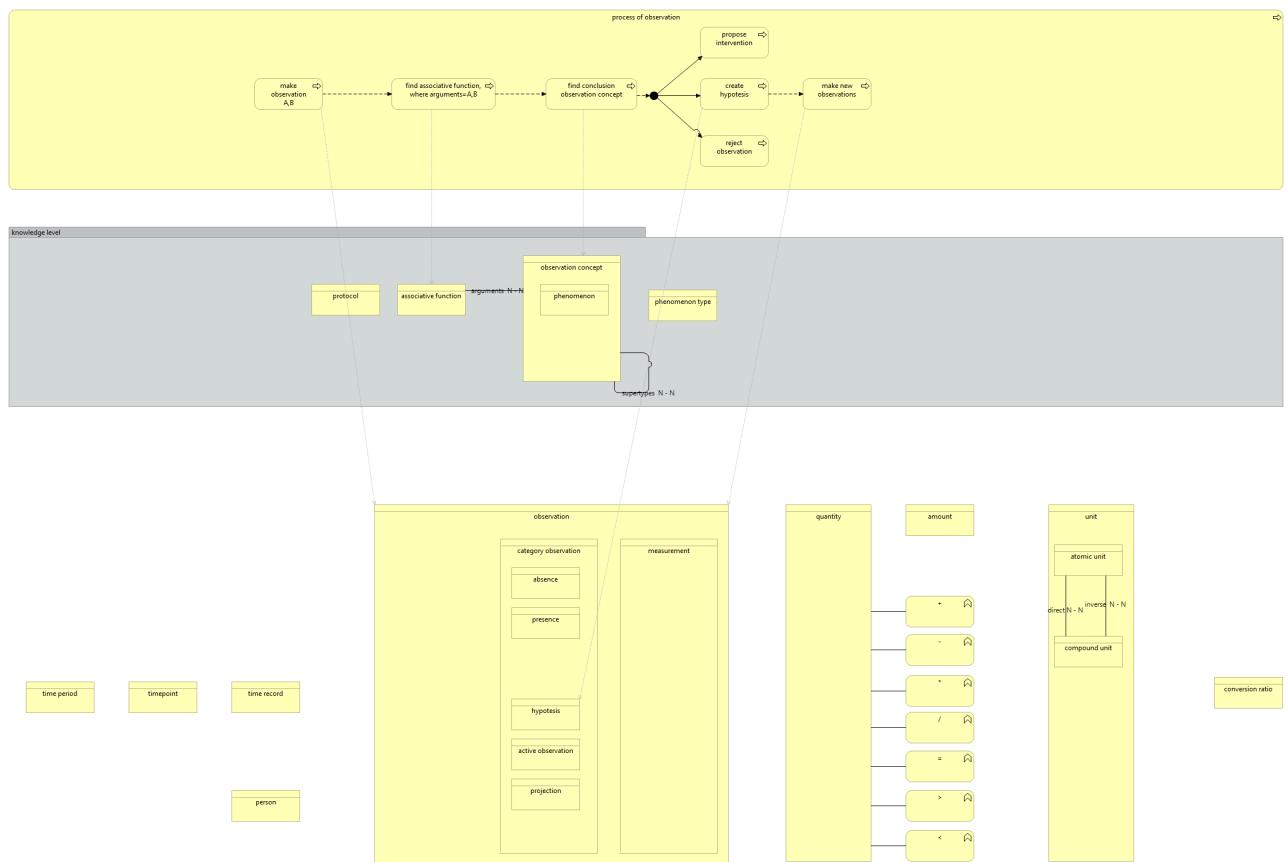
ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION



ASSOCIATED OBSERVATION



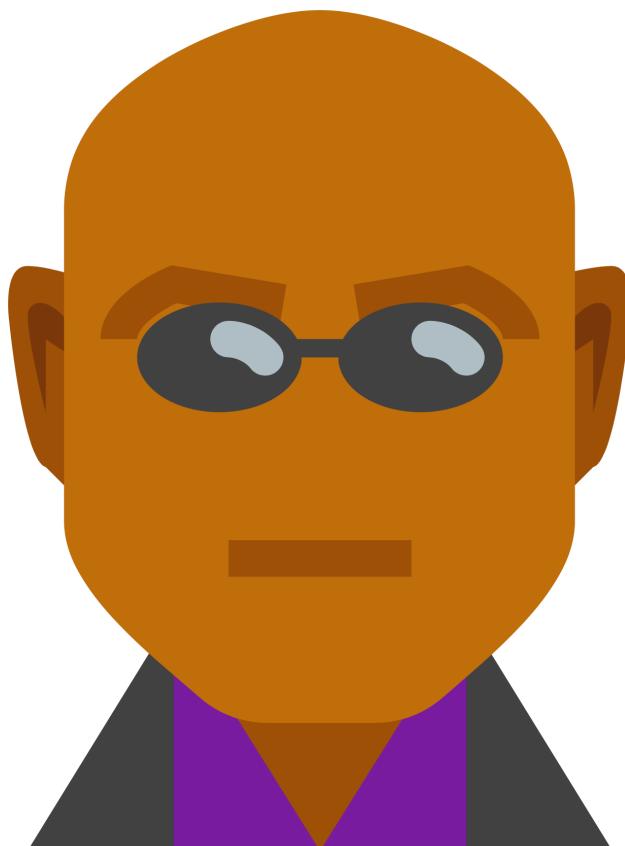
PROCESS OF OBSERVATION



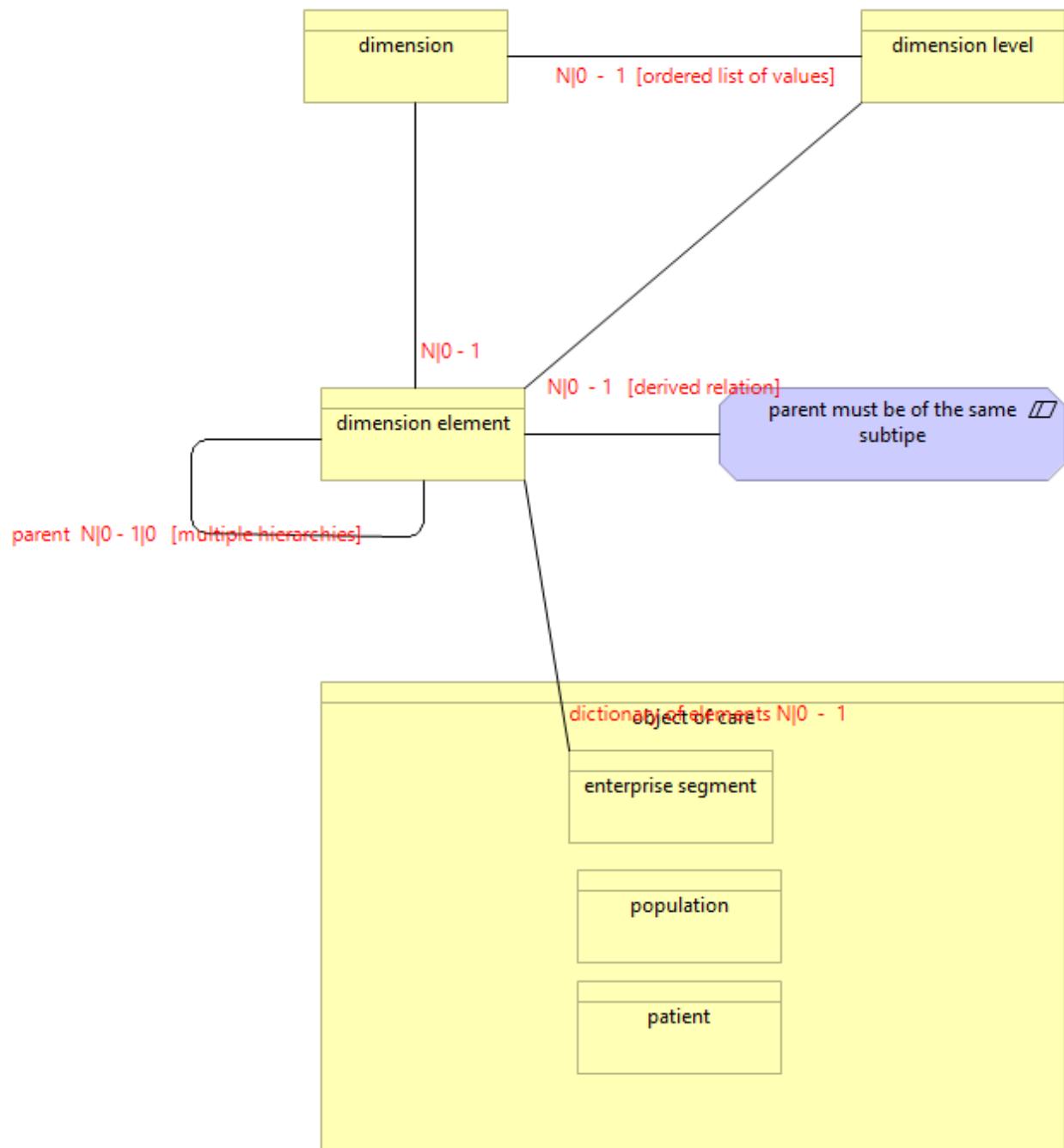
OBSERVATIONS FOR CORPORATE FINANCE

ANALYSIS PATTERNS

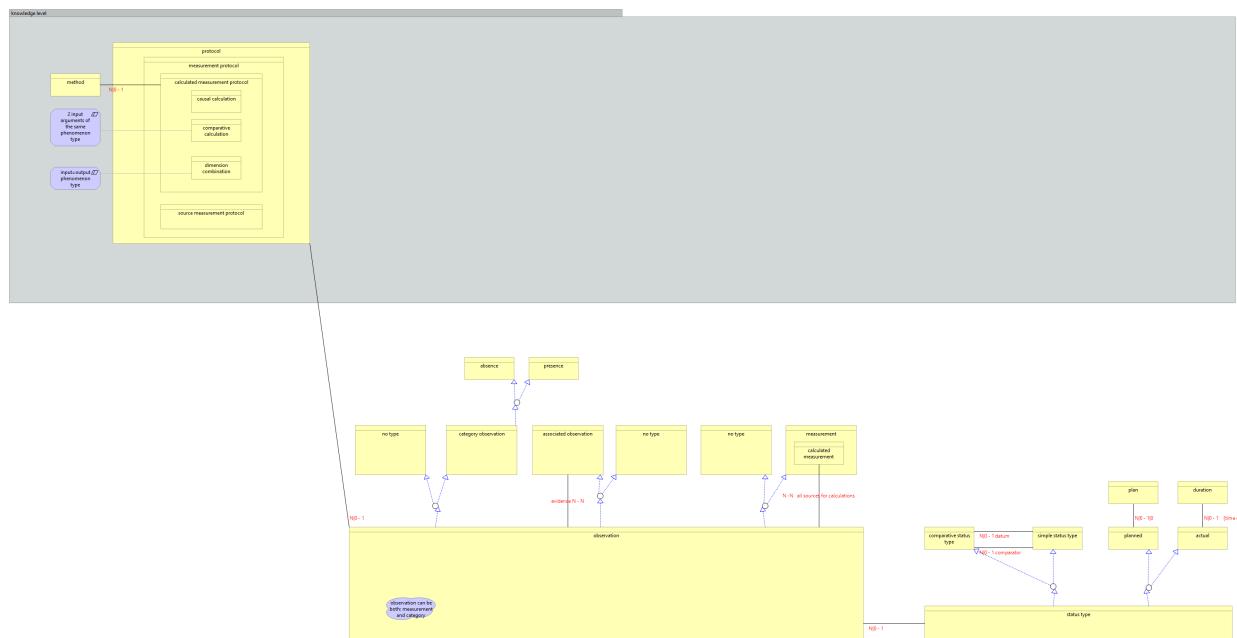
Martin Fowler



ENTERPRISE SEGMENT



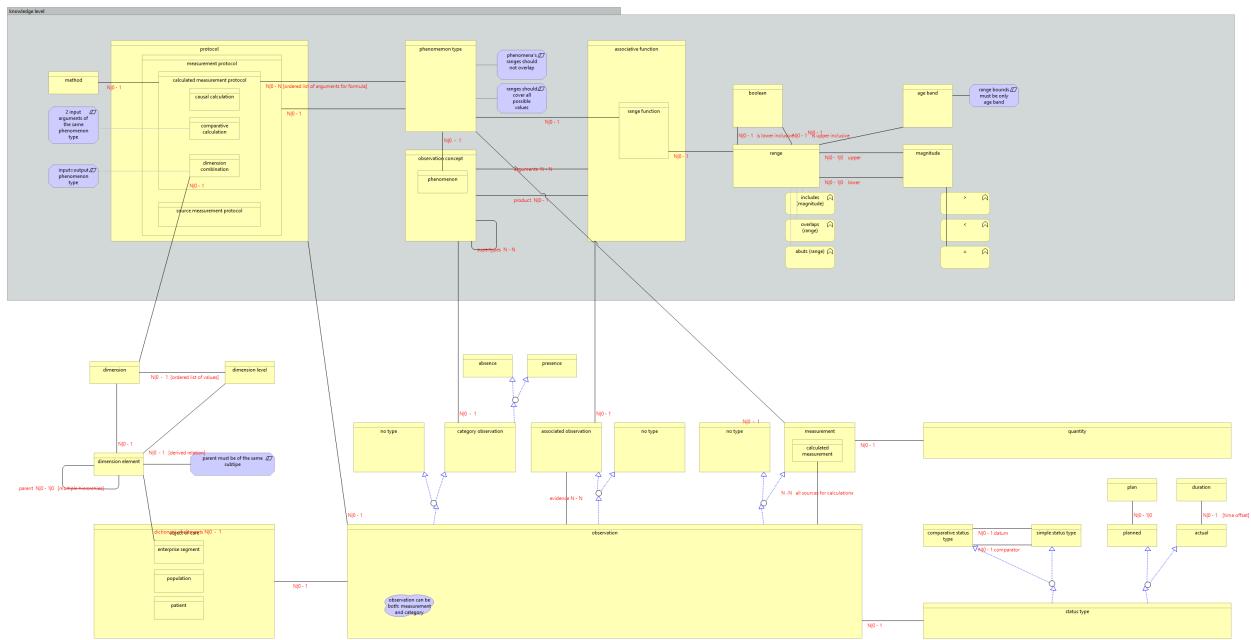
MEASUREMENT PROTOCOL

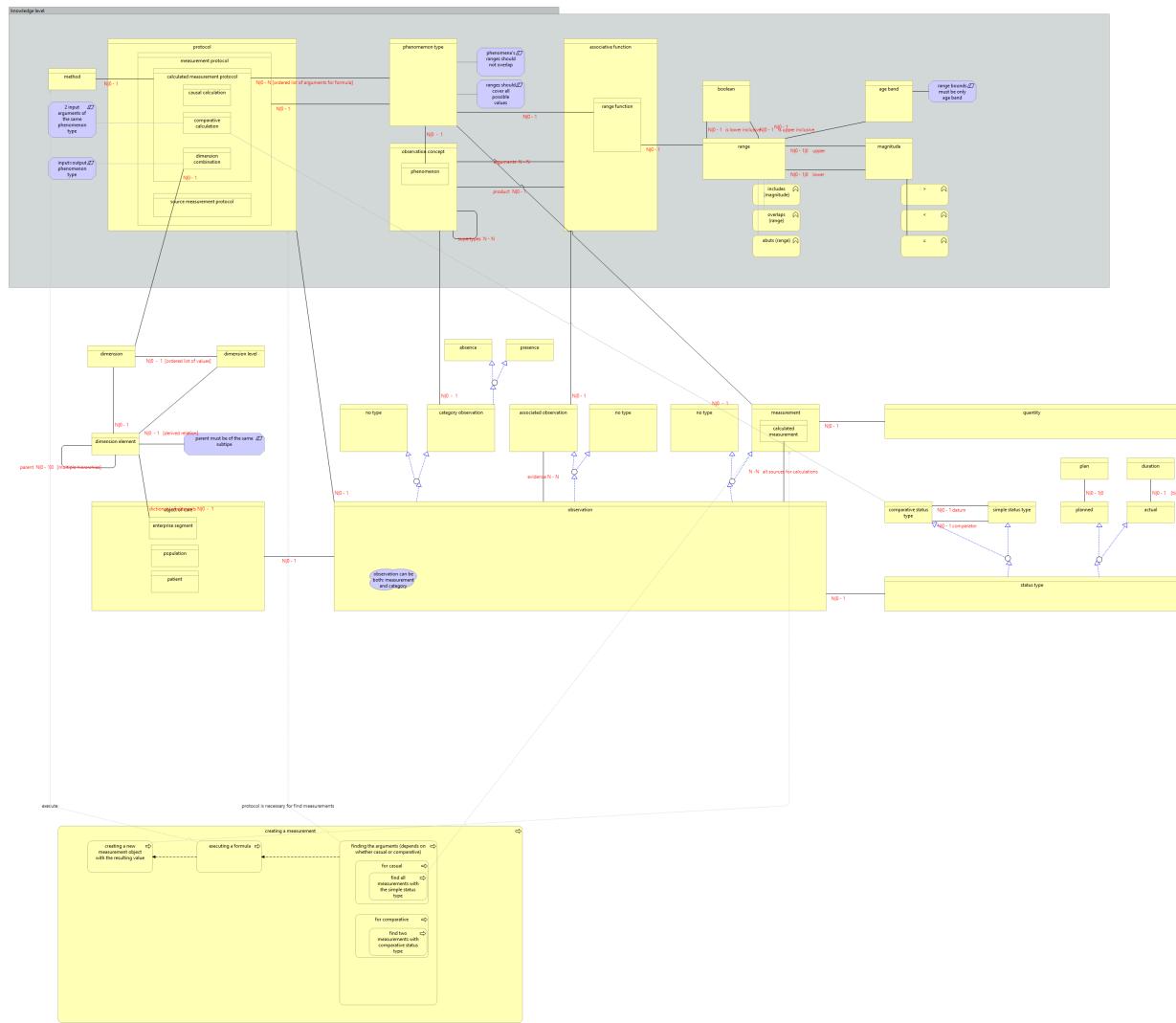


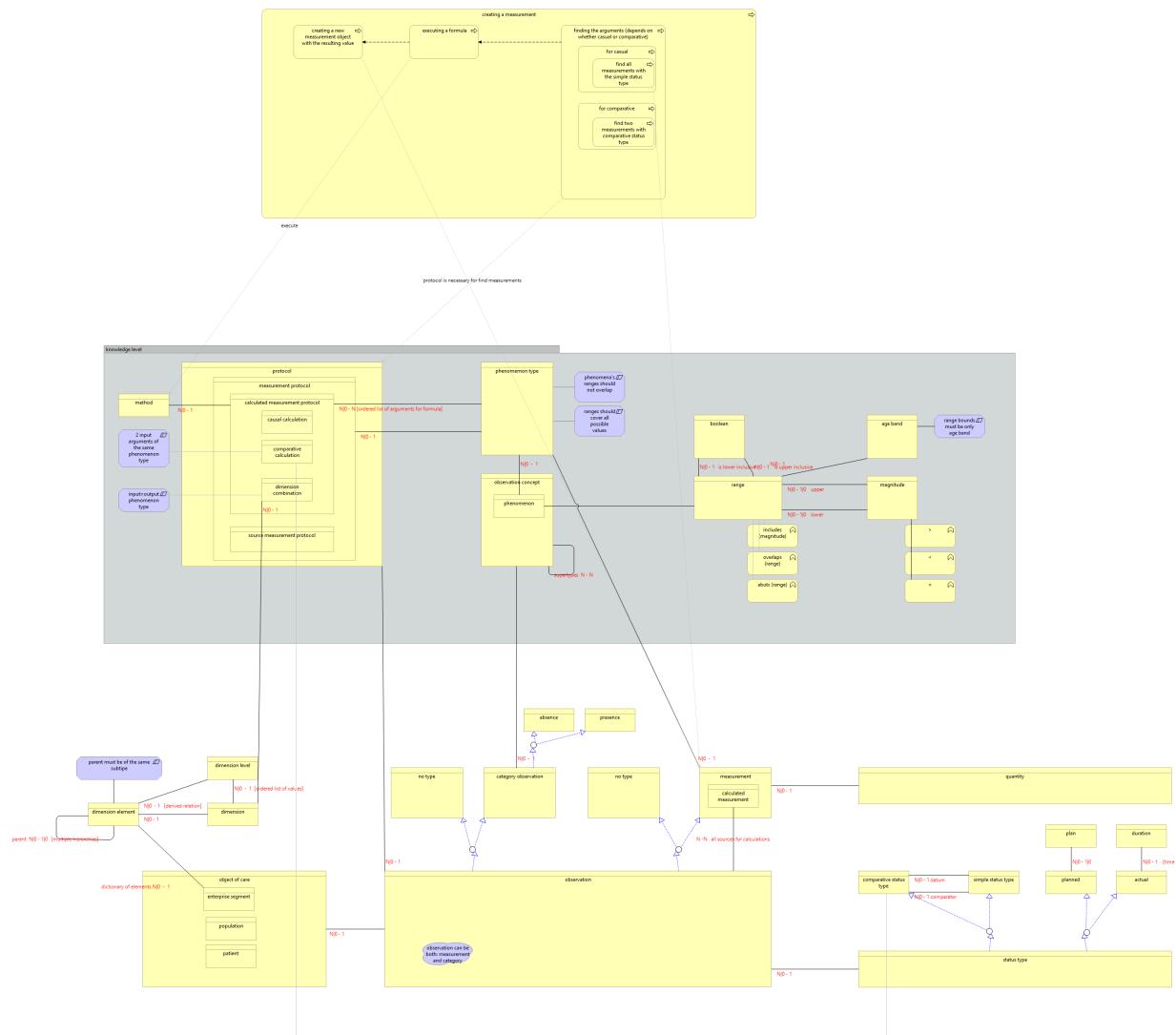
RANGE



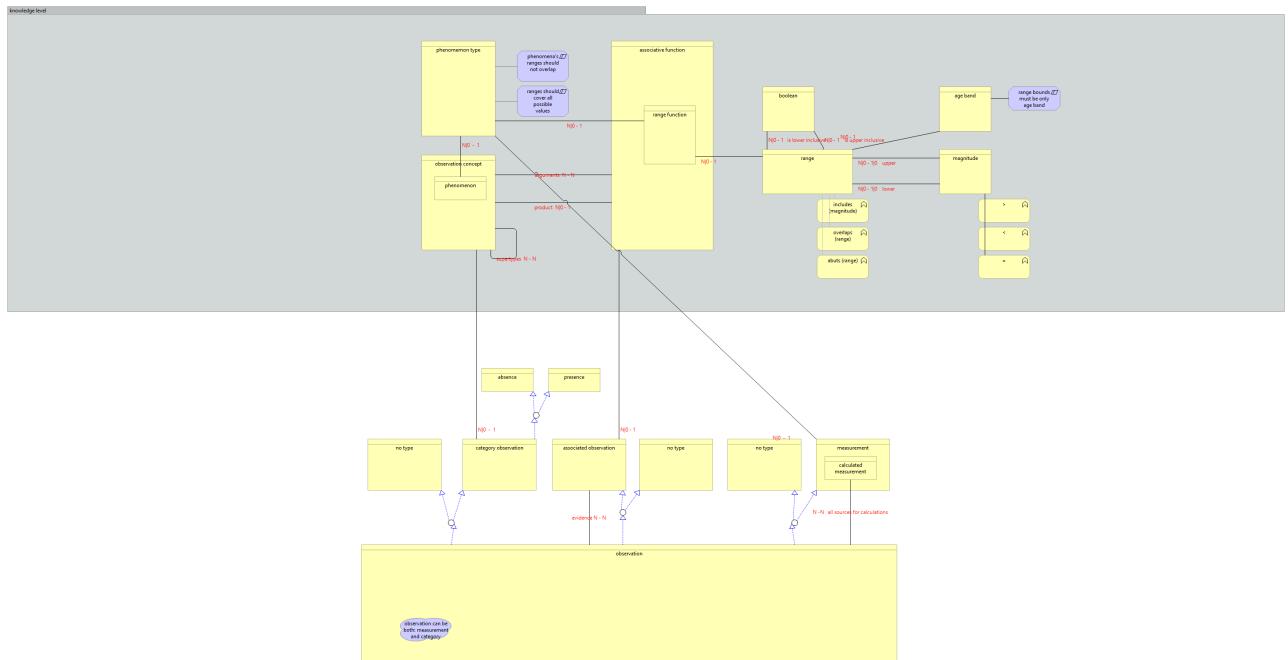
OBSERVATIONS FOR CORPORATE FINANCE







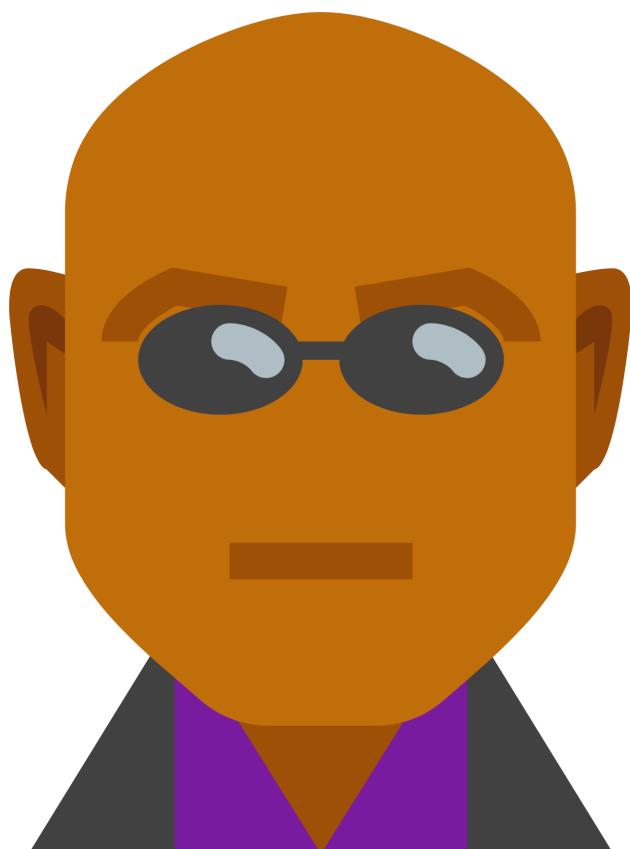
PHENOMENON WITH RANGE



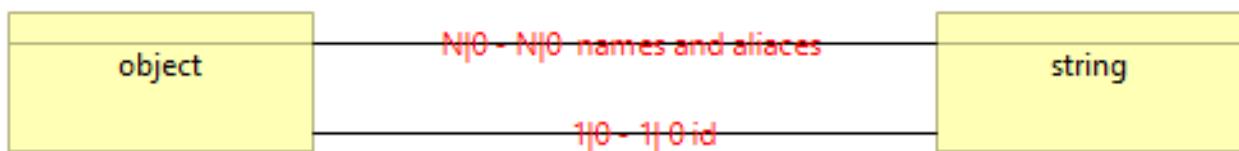
REFERRING TO OBJECTS

ANALYSIS PATTERNS

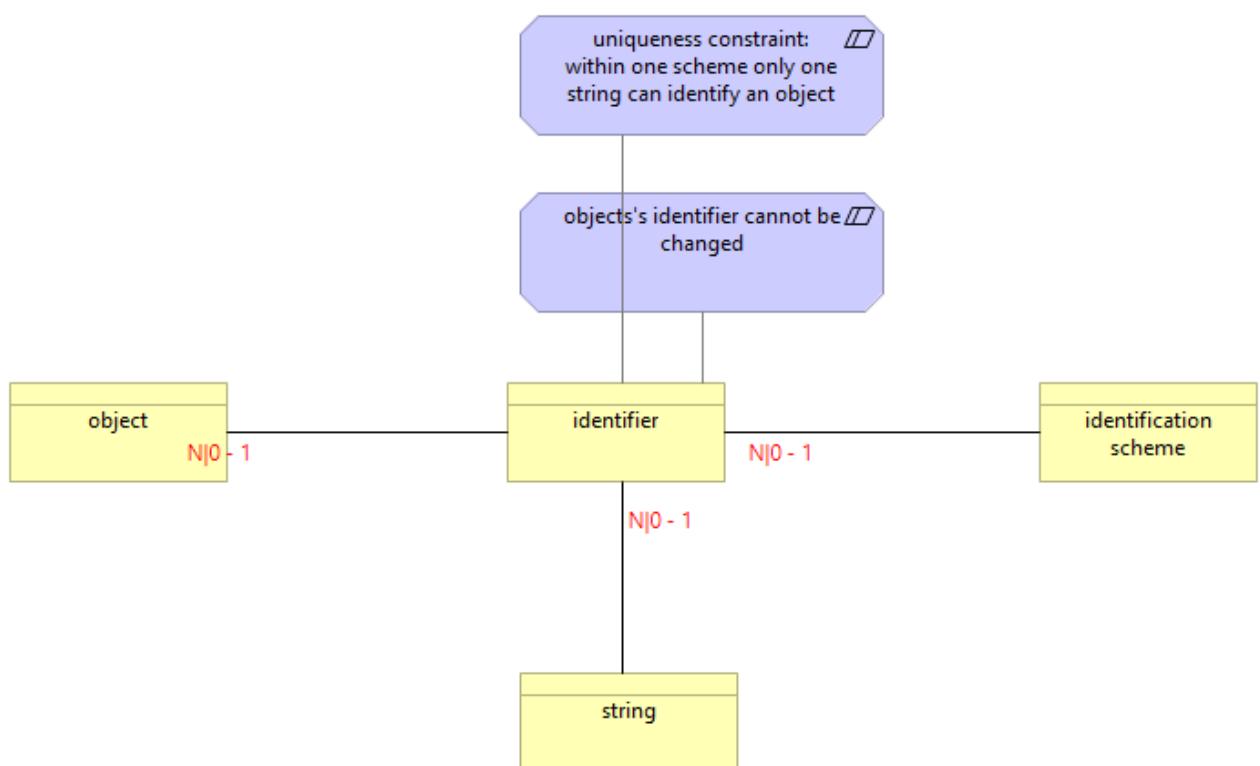
Martin Fowler



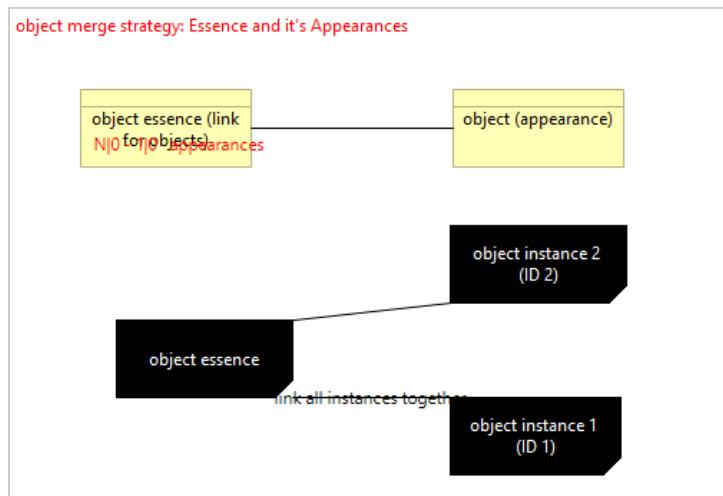
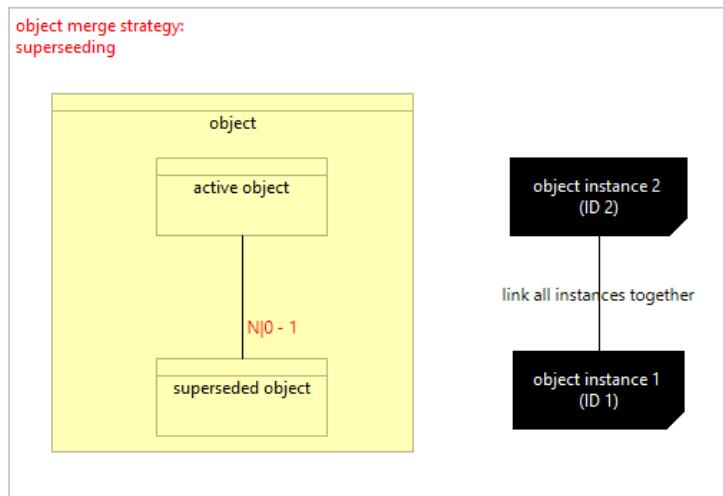
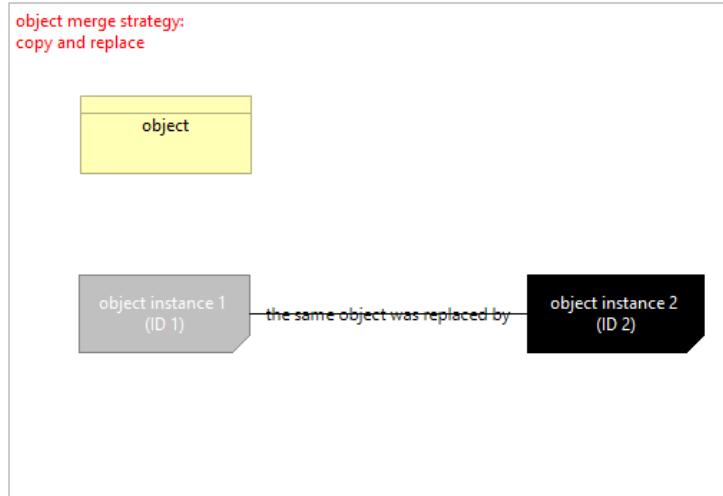
NAME



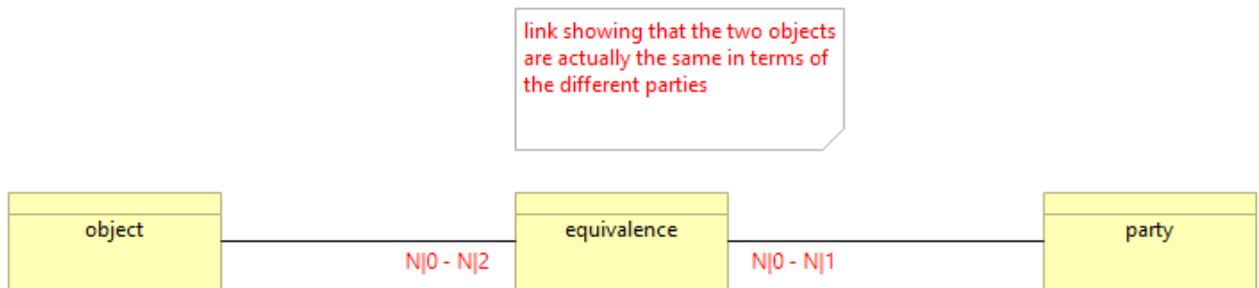
IDENTIFICATION SCHEME



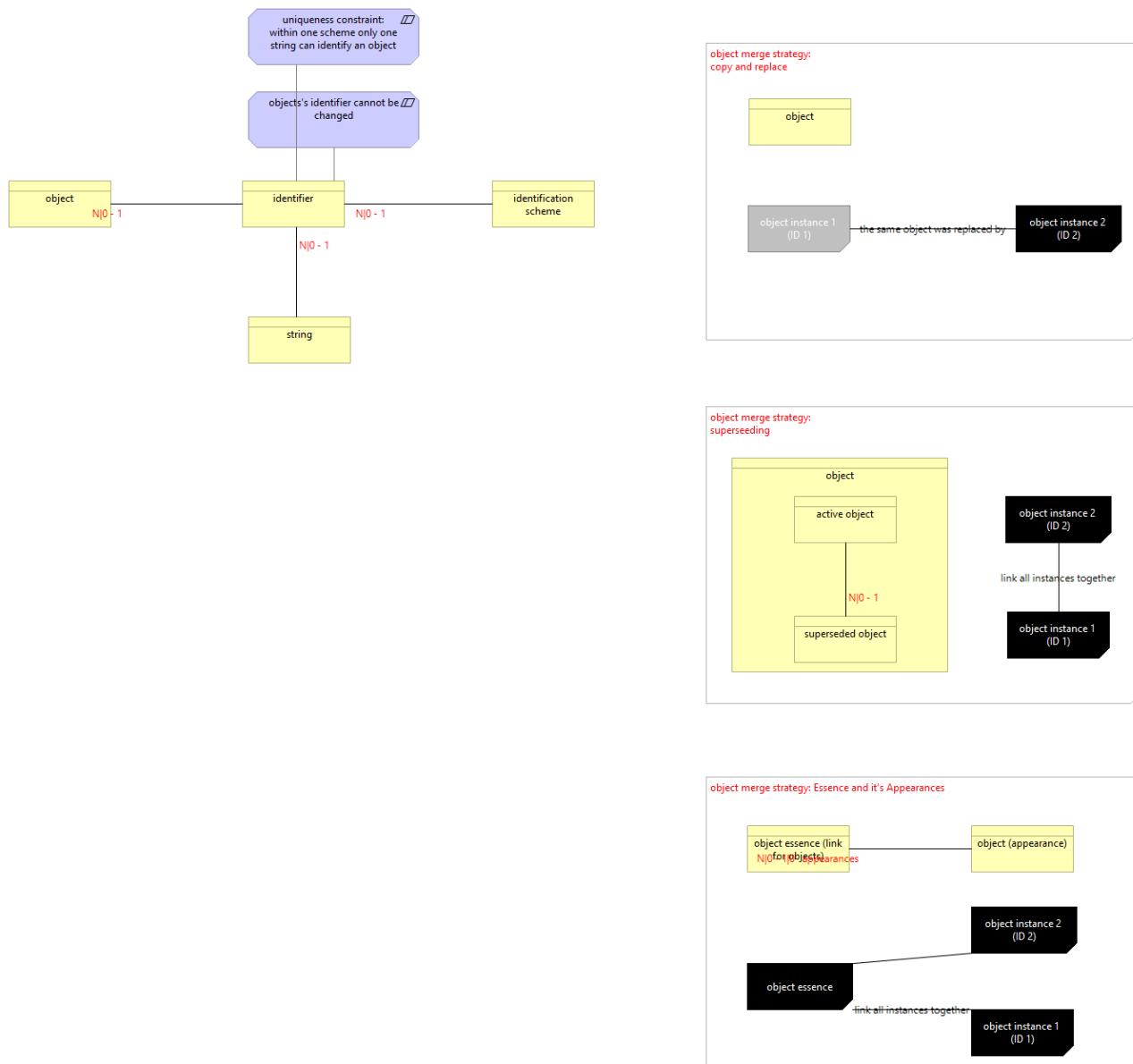
OBJECT MERGE



OBJECT EQUIVALENCE



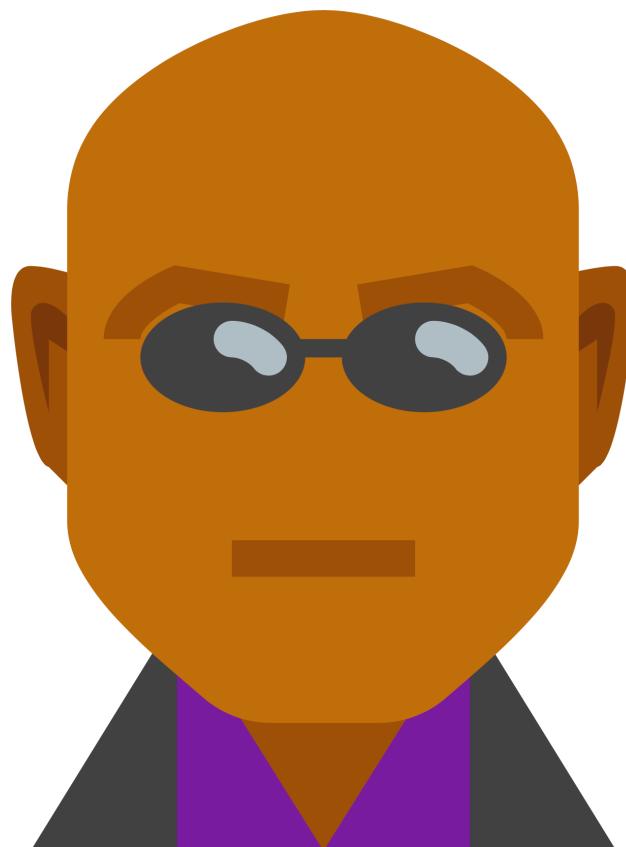
REFERRING TO OBJECTS



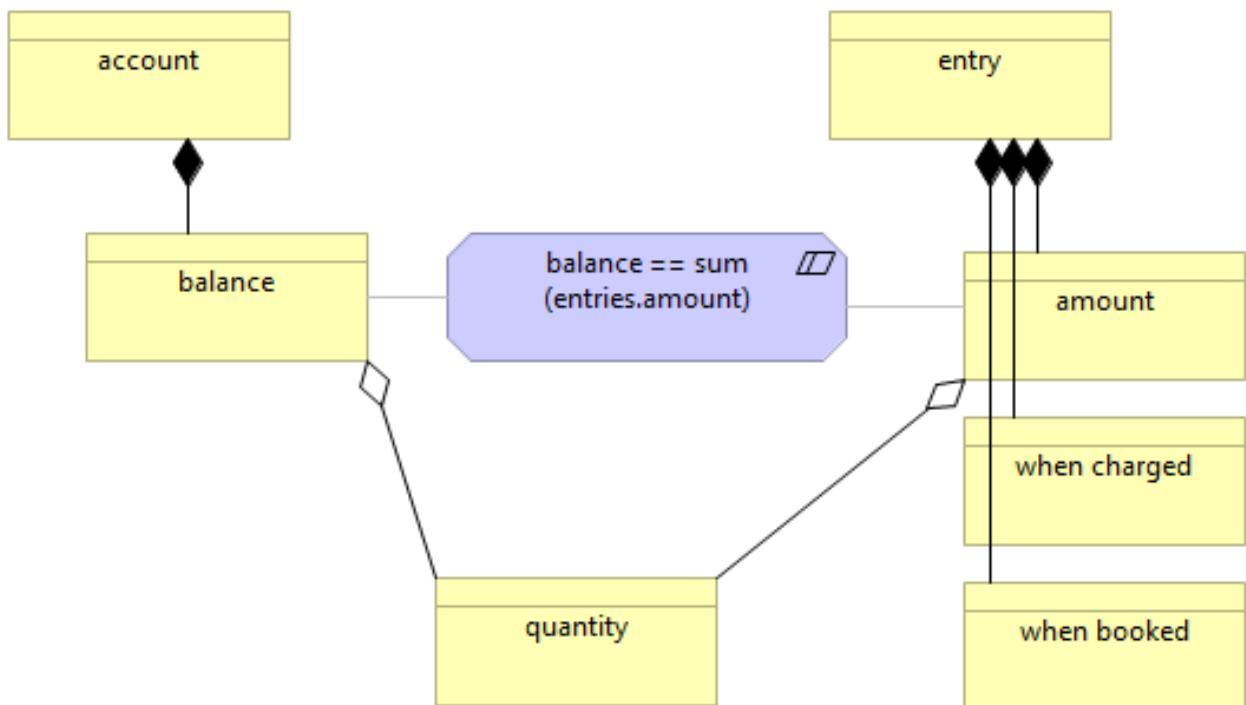
INVENTORY AND ACCOUNTING

ANALYSIS PATTERNS

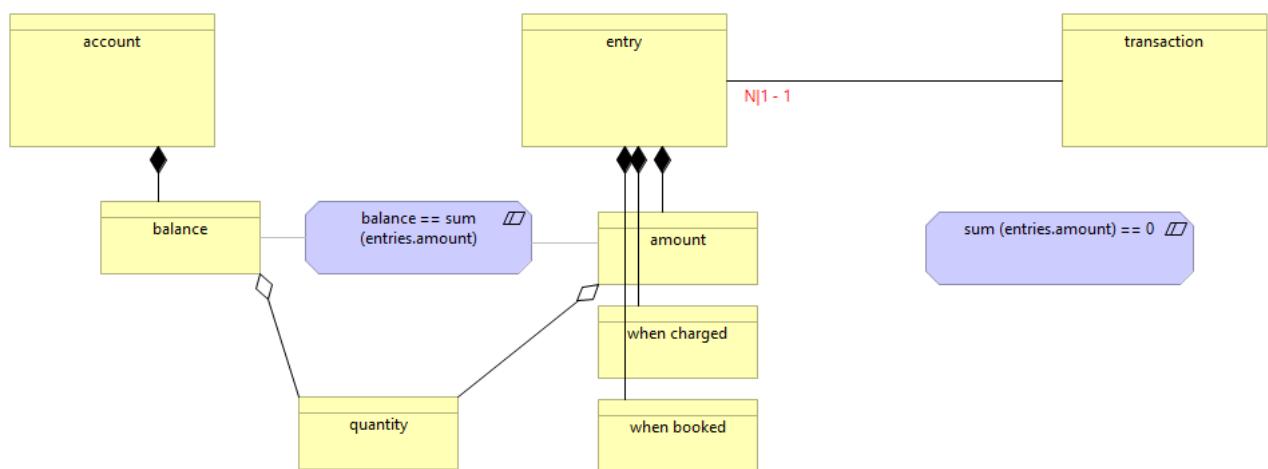
Martin Fowler



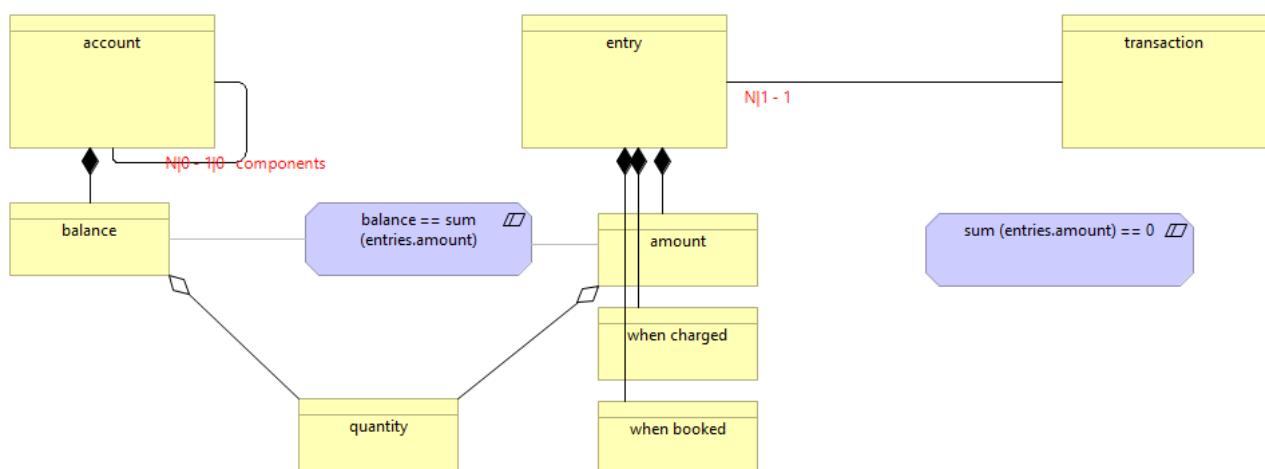
ACCOUNT



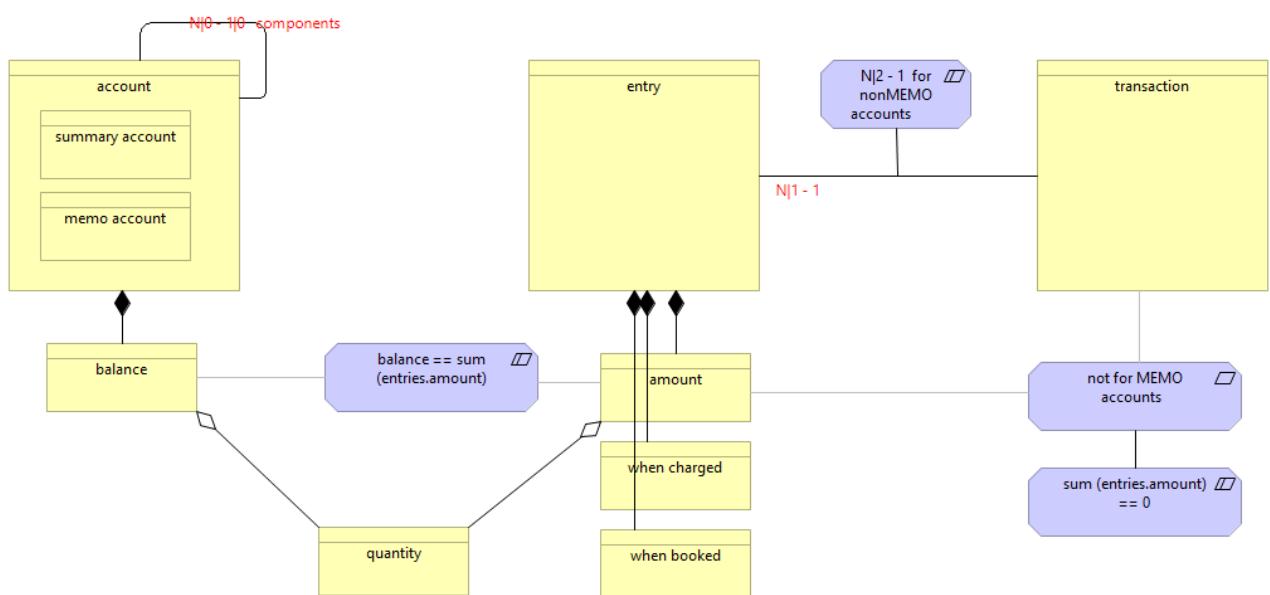
TRANSACTIONS



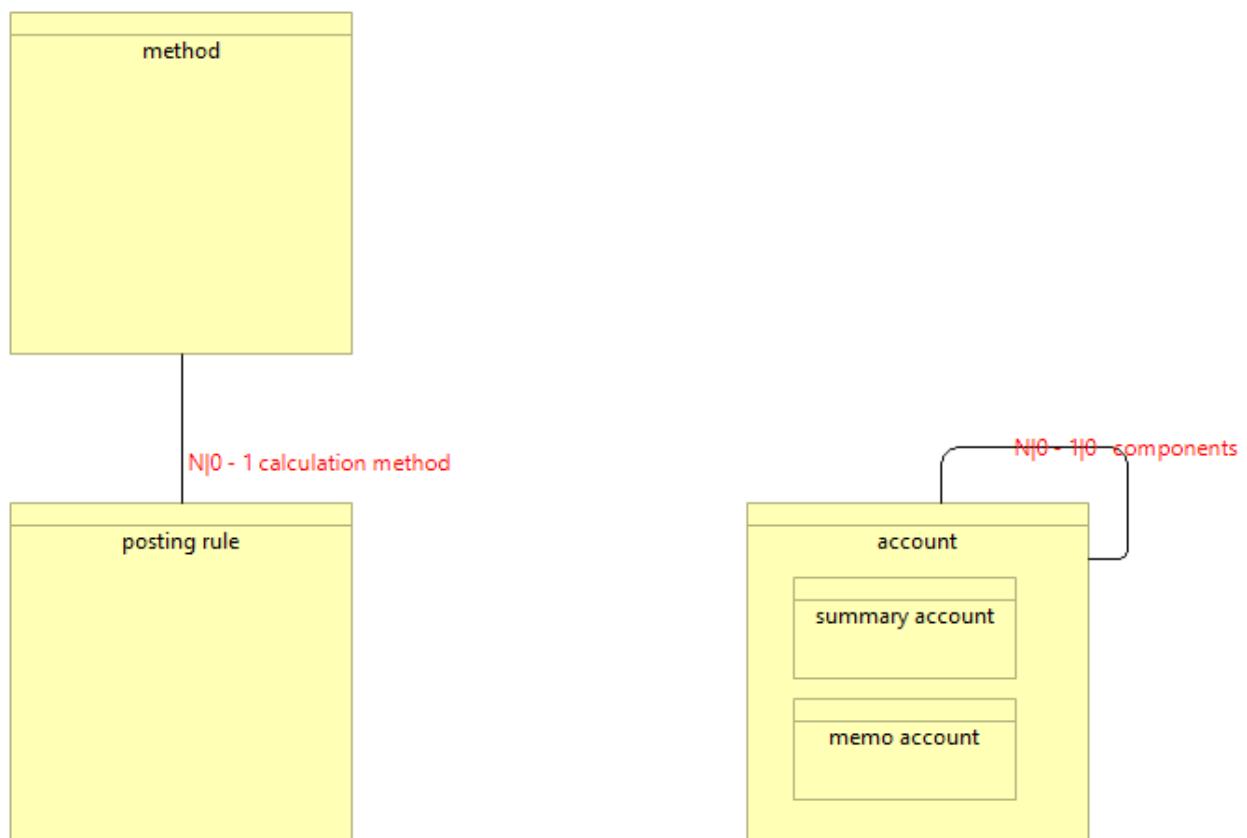
SUMMARY ACCOUNT



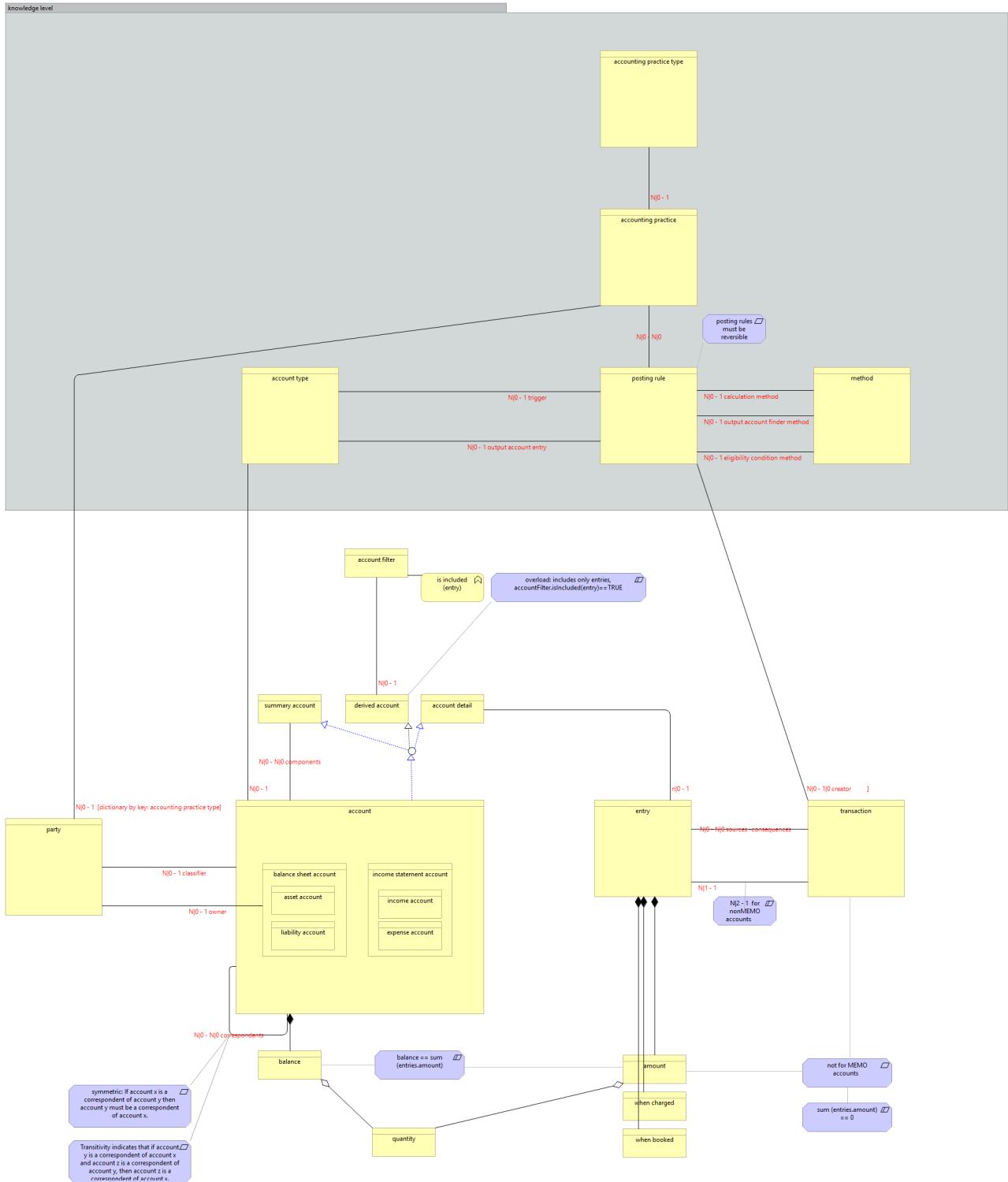
MEMO ACCOUNT



POSTING RULES



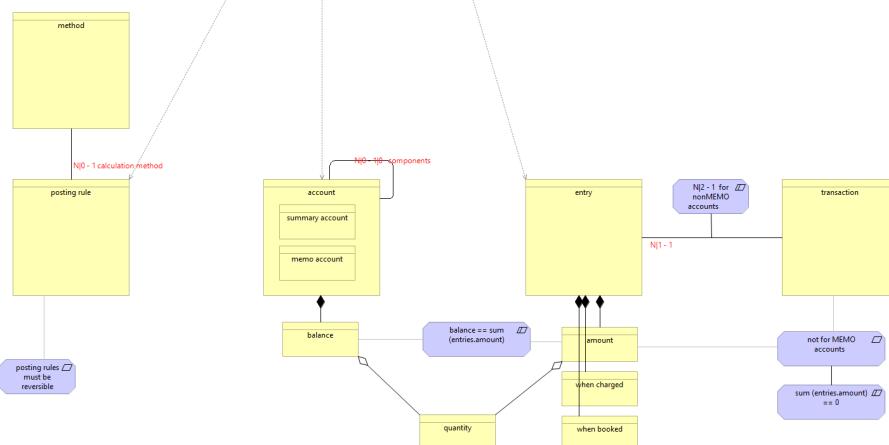
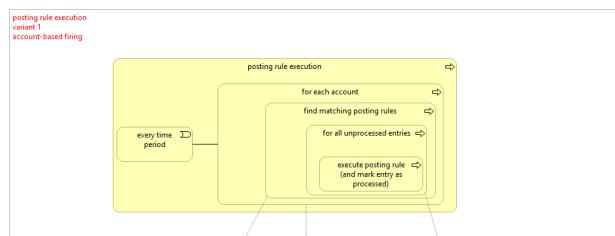
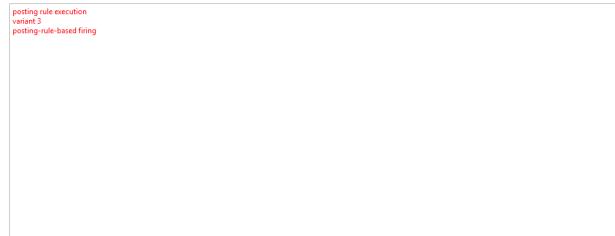
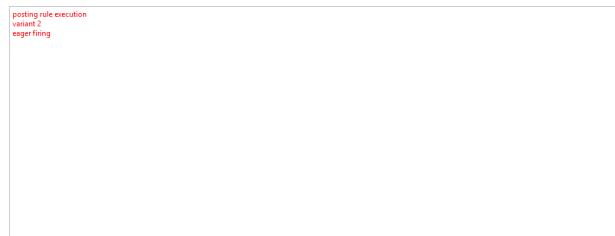
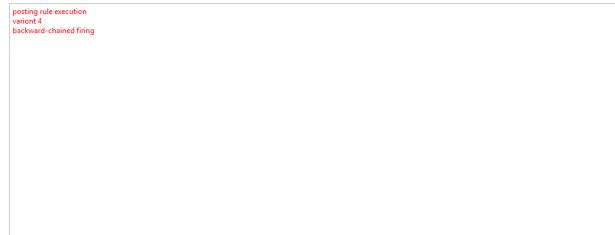
INVENTORY AND ACCOUNTING



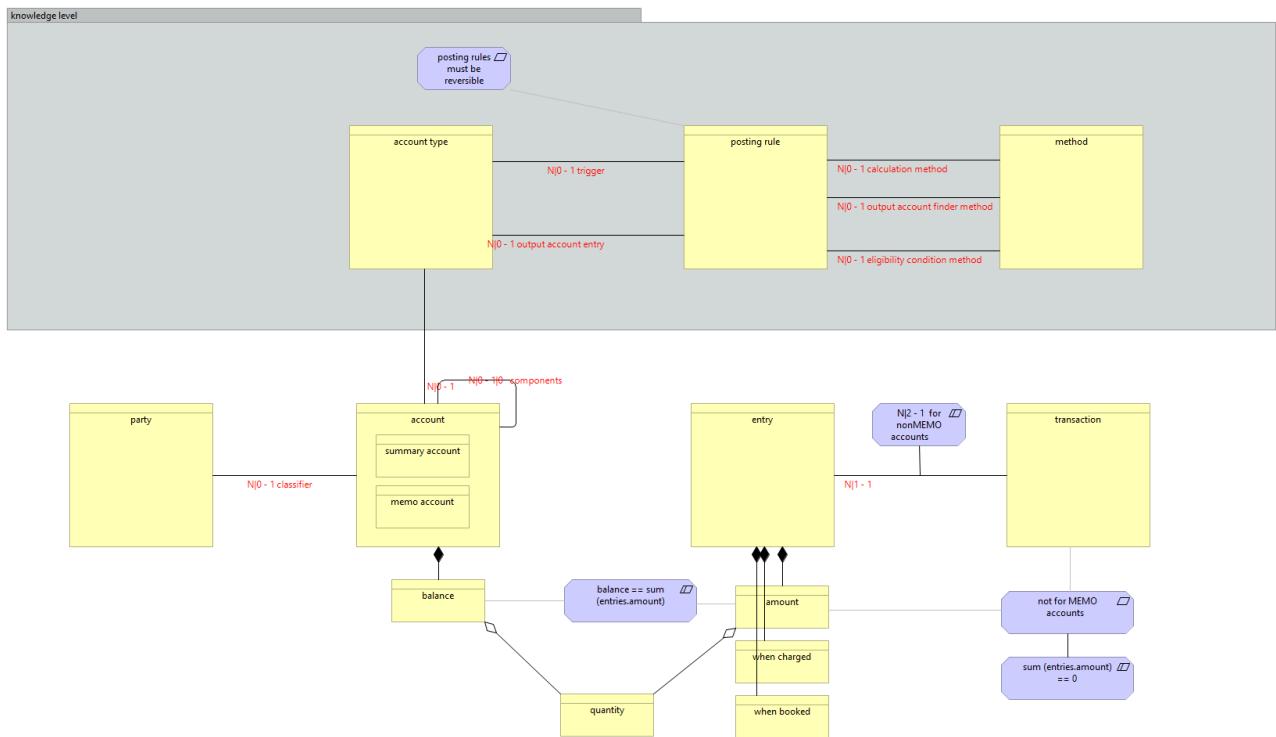
INDIVIDUAL INSTANCE METHOD



POSTING RULE EXECUTION



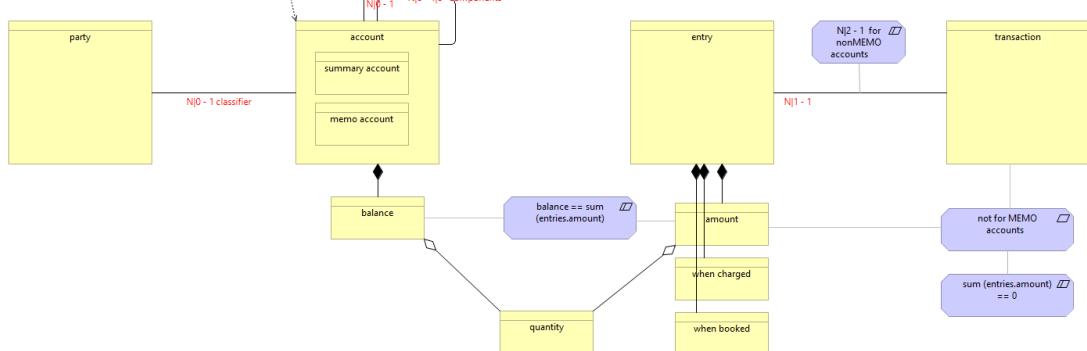
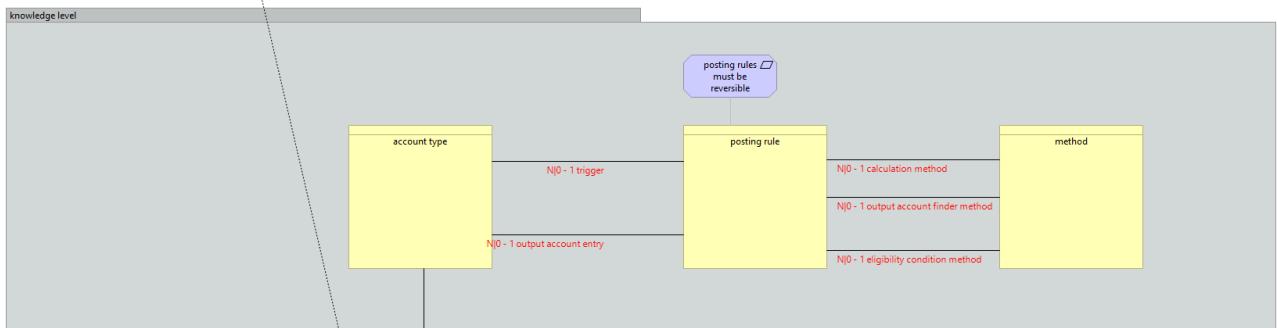
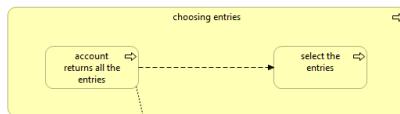
POSTING RULES FOR MANY ACCOUNTS



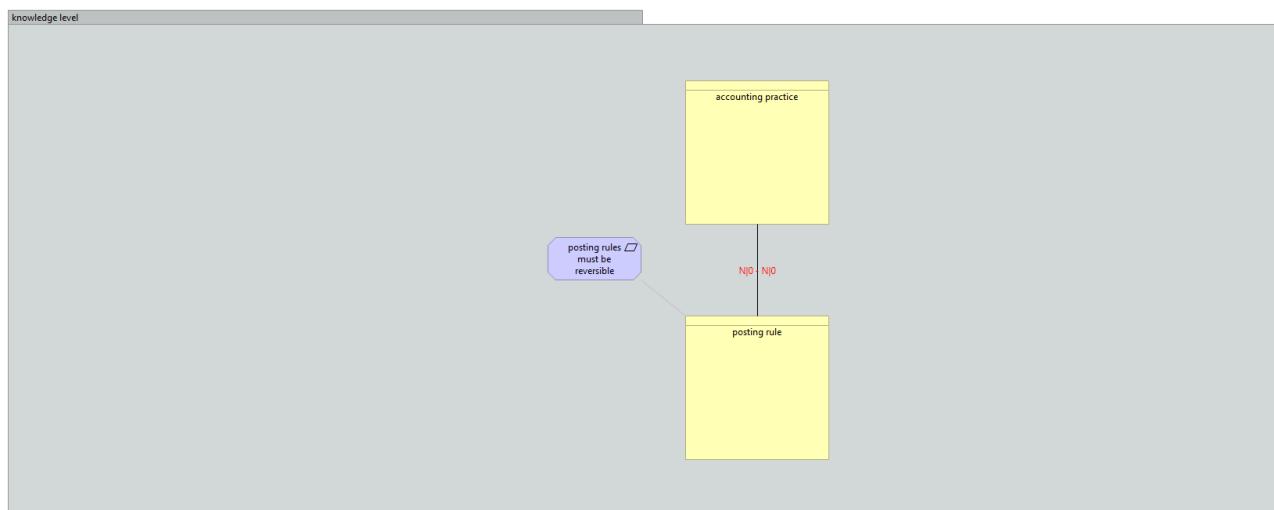
CHOOSING ENTRIES

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

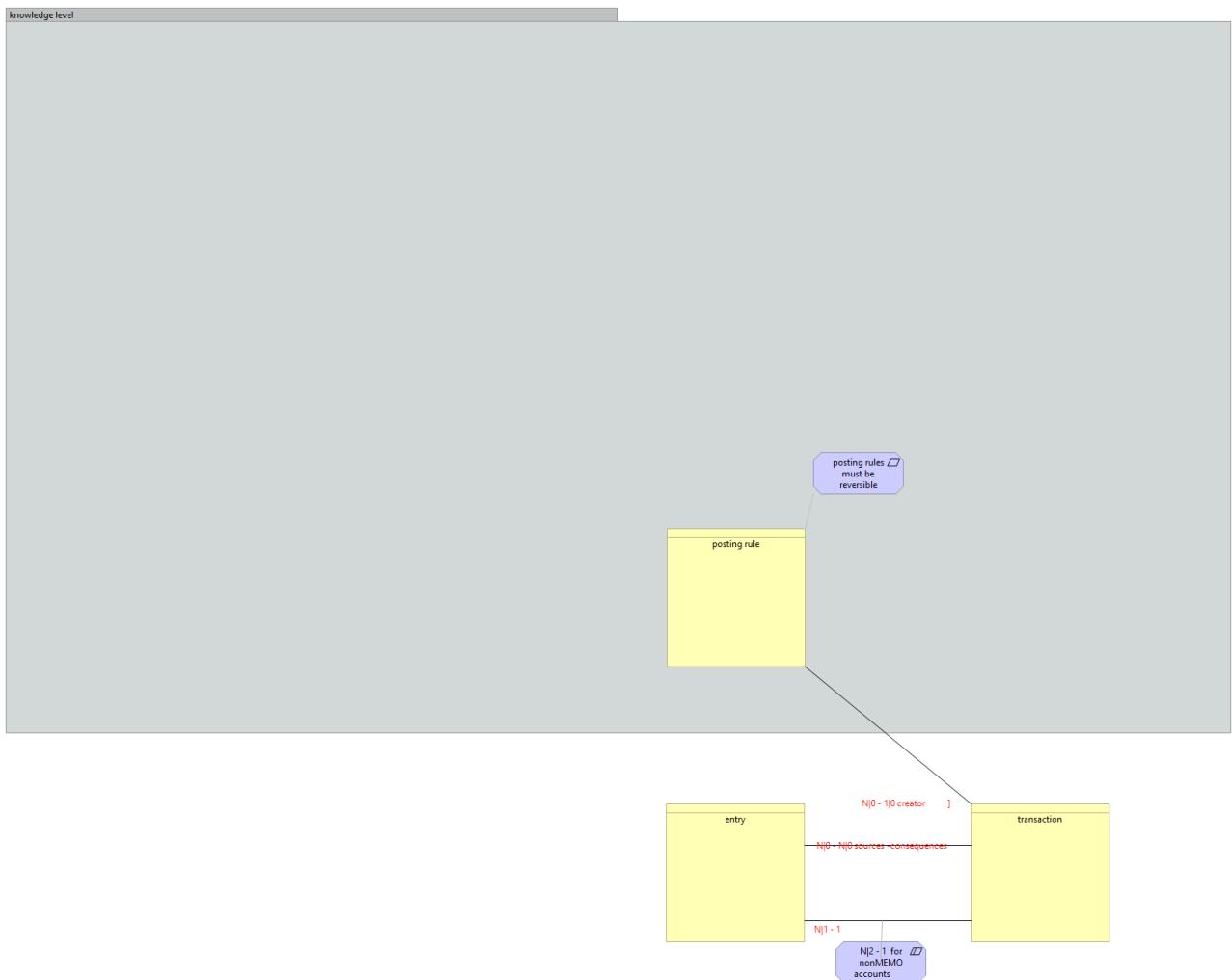
selection of entries for the application of the posing rules:
variant 1
The account returns all the entries, and the client selects the entries it needs.



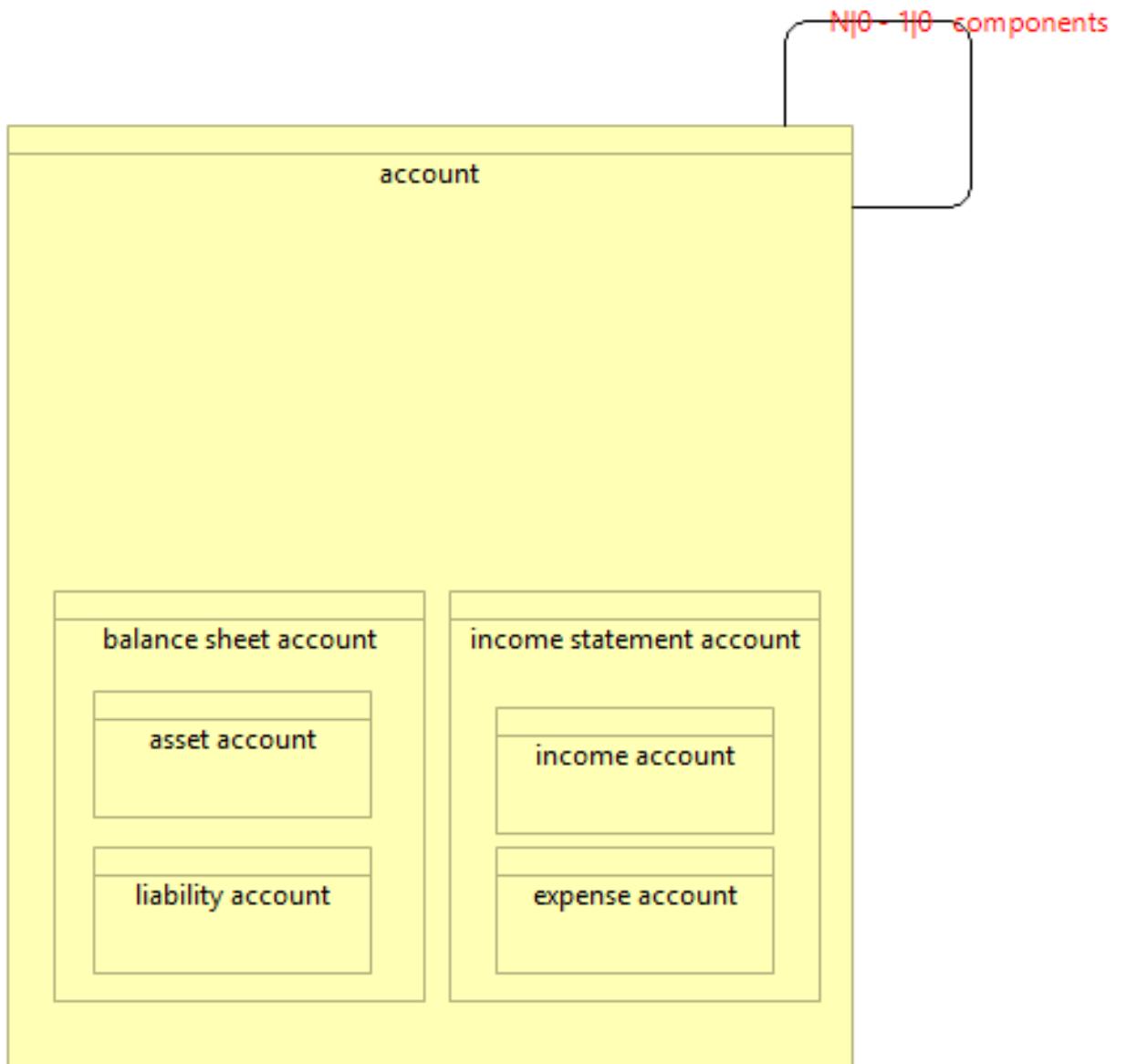
ACCOUNTING PRACTICE



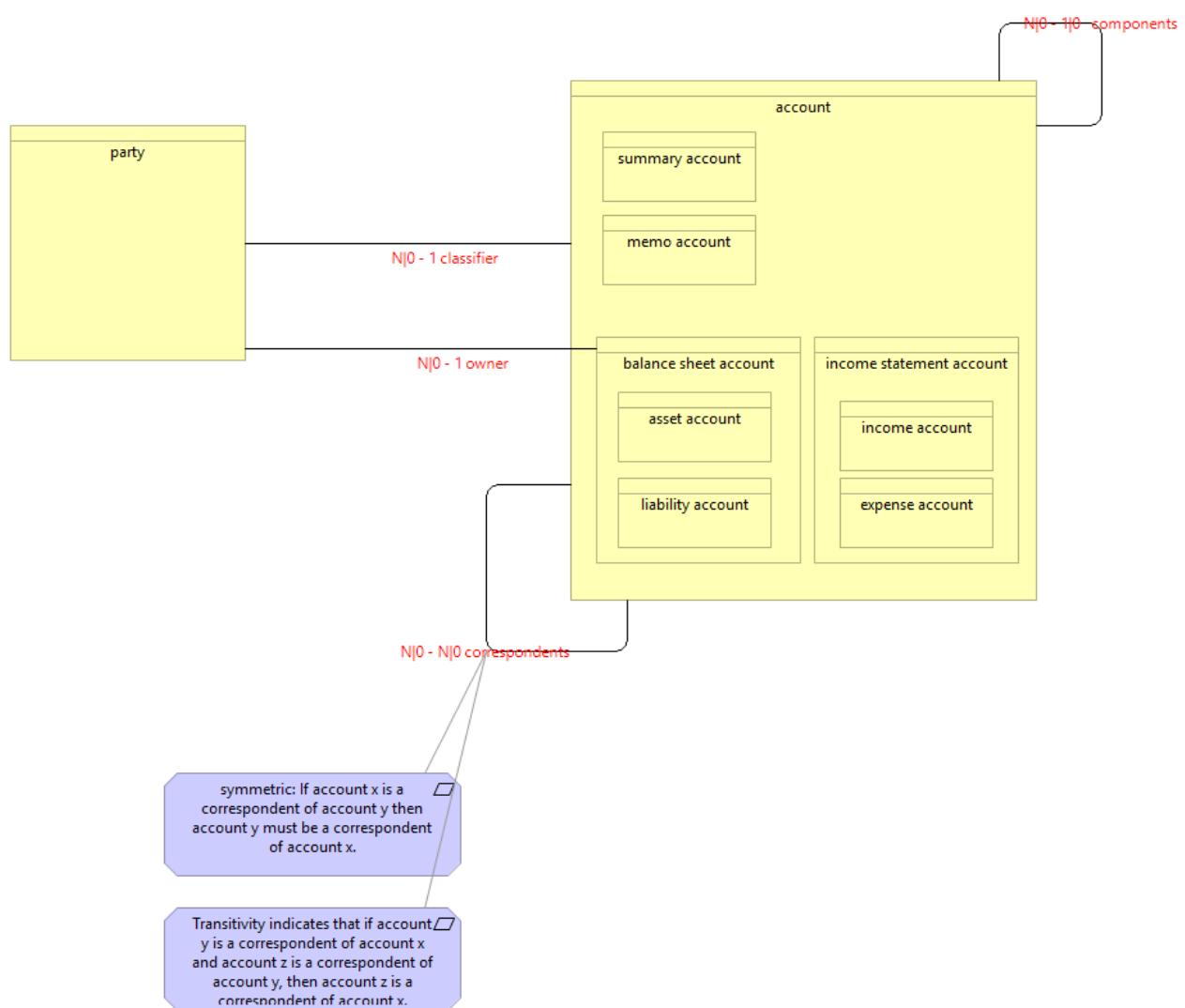
SOURCES OF AN ENTRY



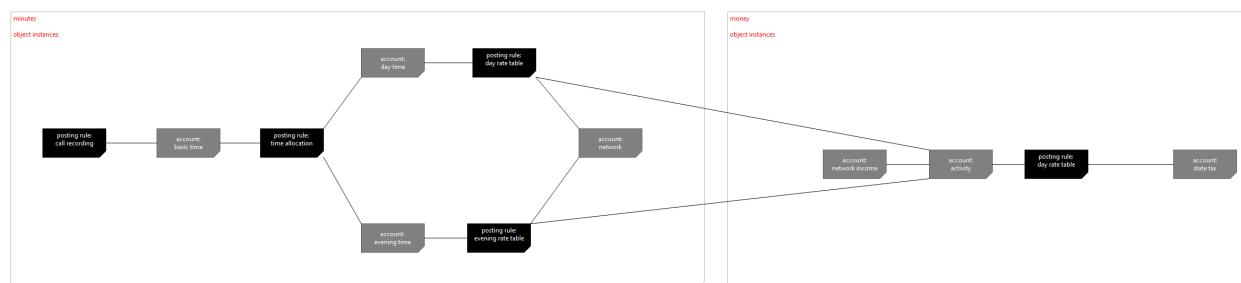
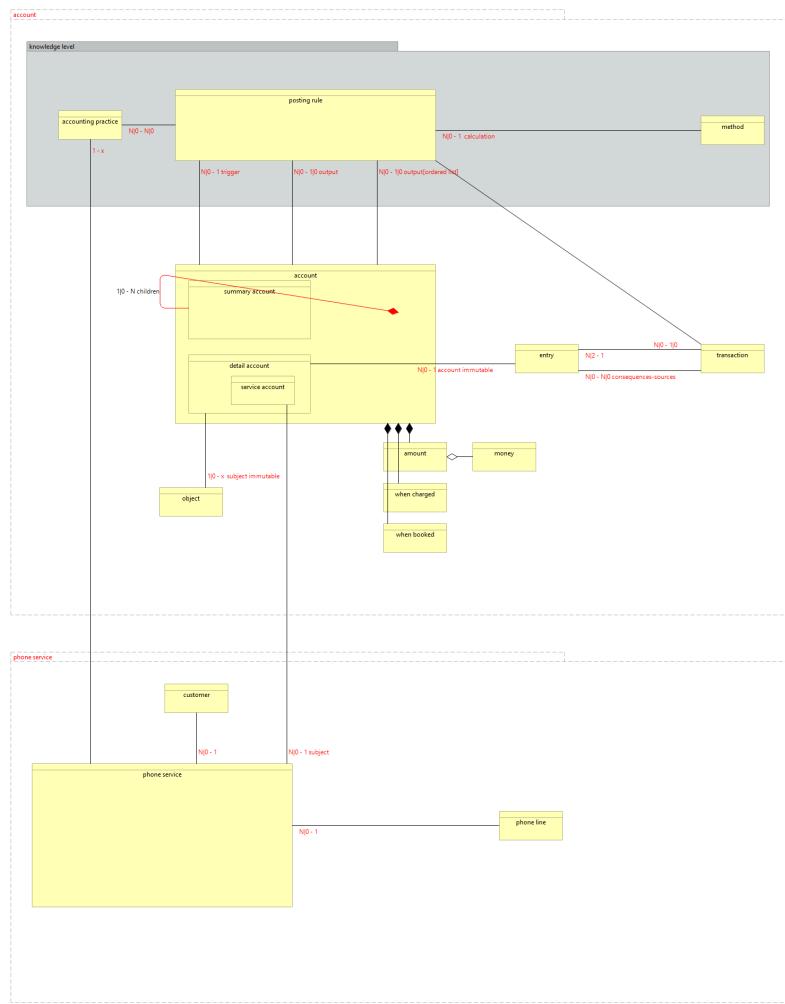
BALANCE SHEET AND INCOME STATEMENT



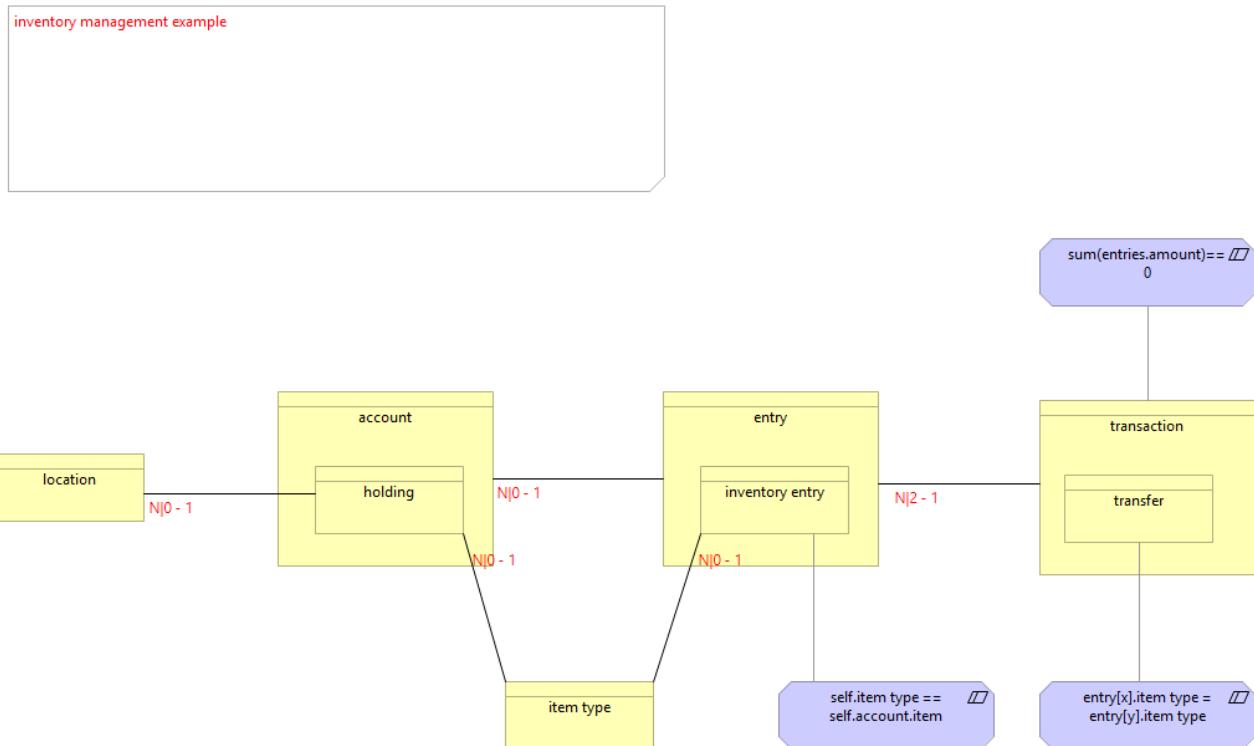
CORRESPONDING ACCOUNT



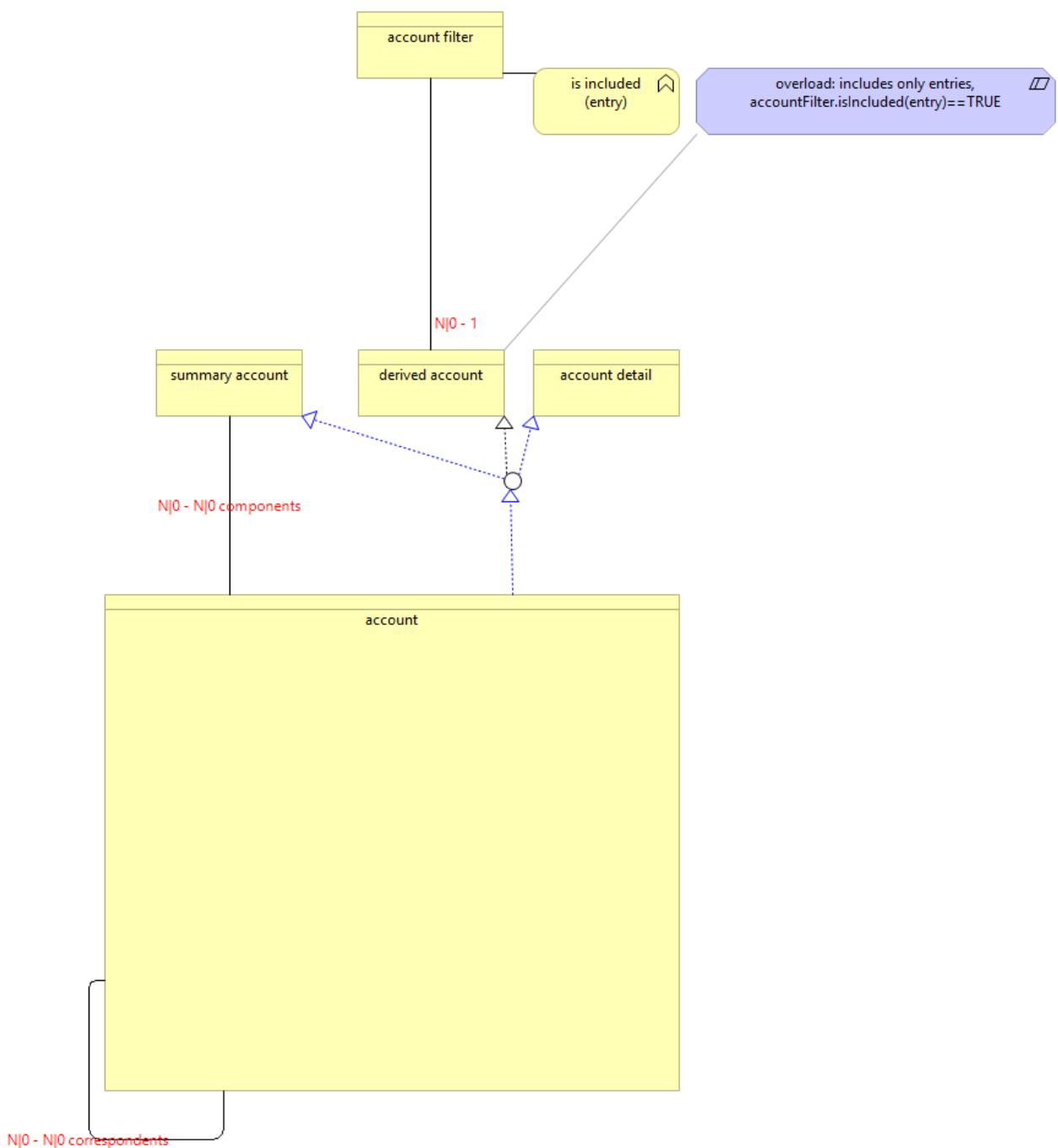
SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)



SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)



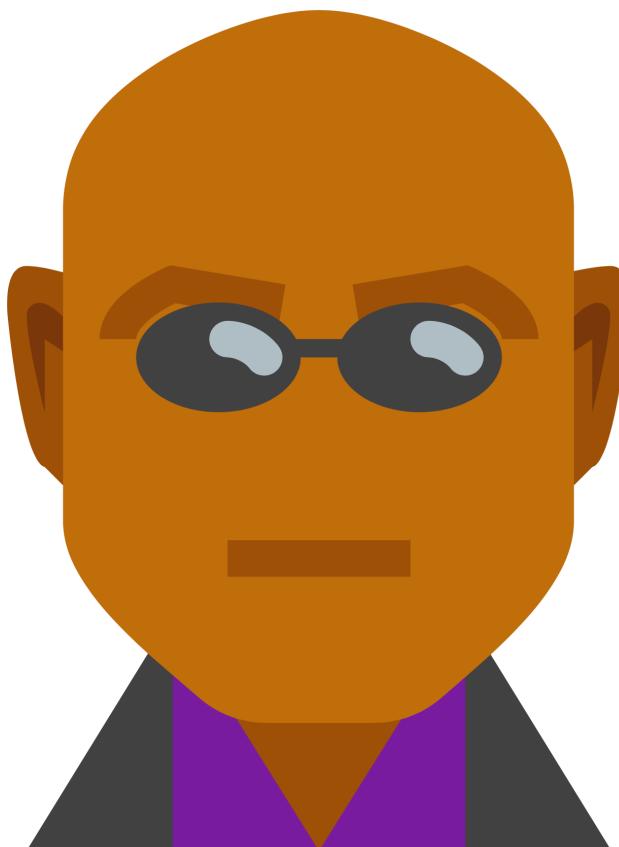
BOOKING ENTRIES TO MULTIPLE ACCOUNTS



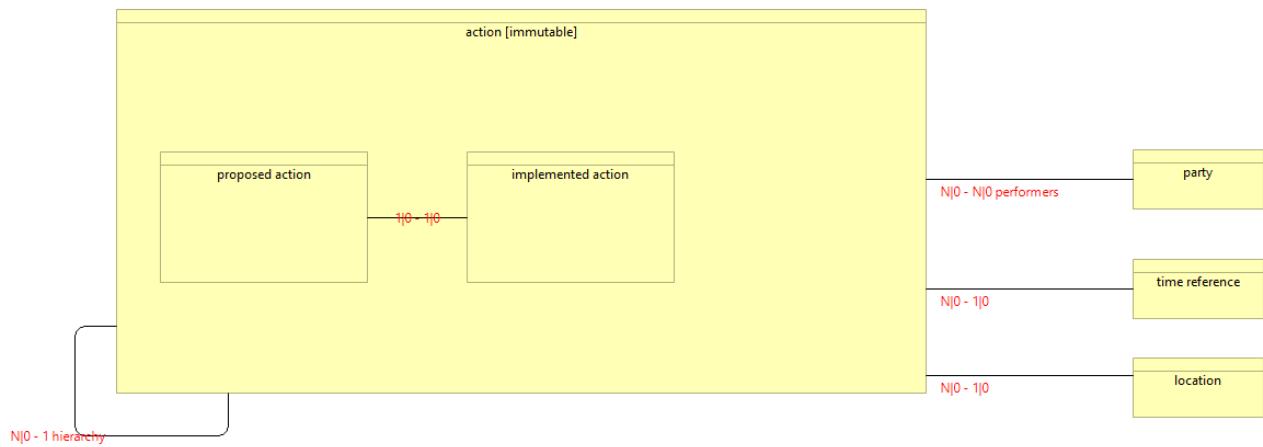
PLANNING

ANALYSIS PATTERNS

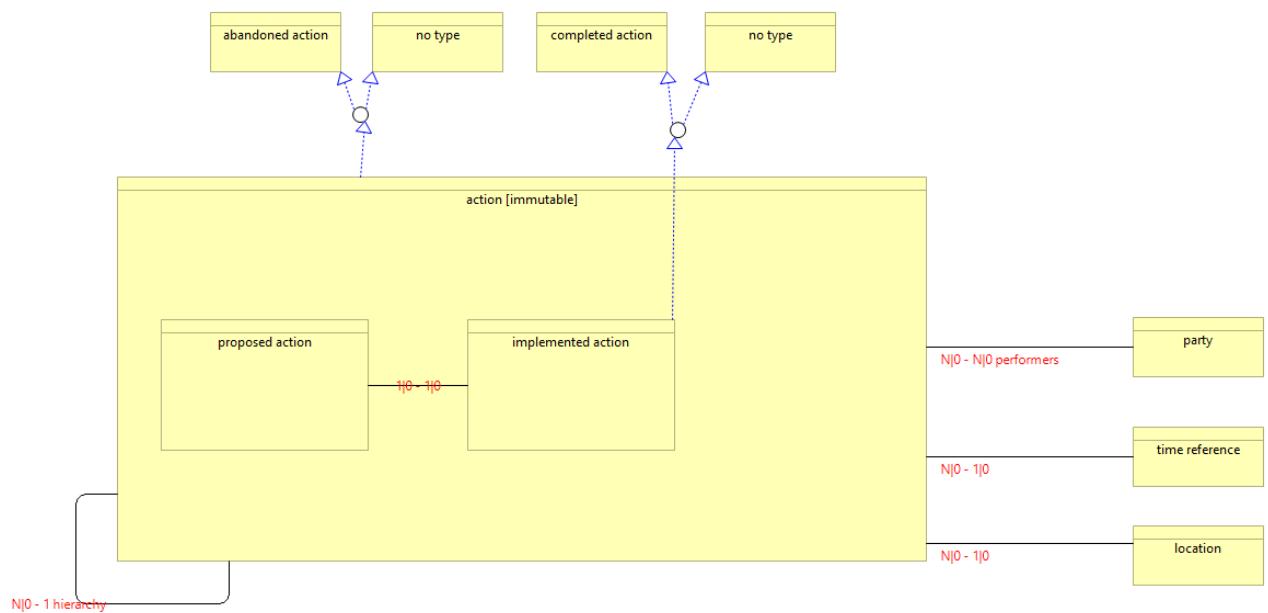
Martin Fowler



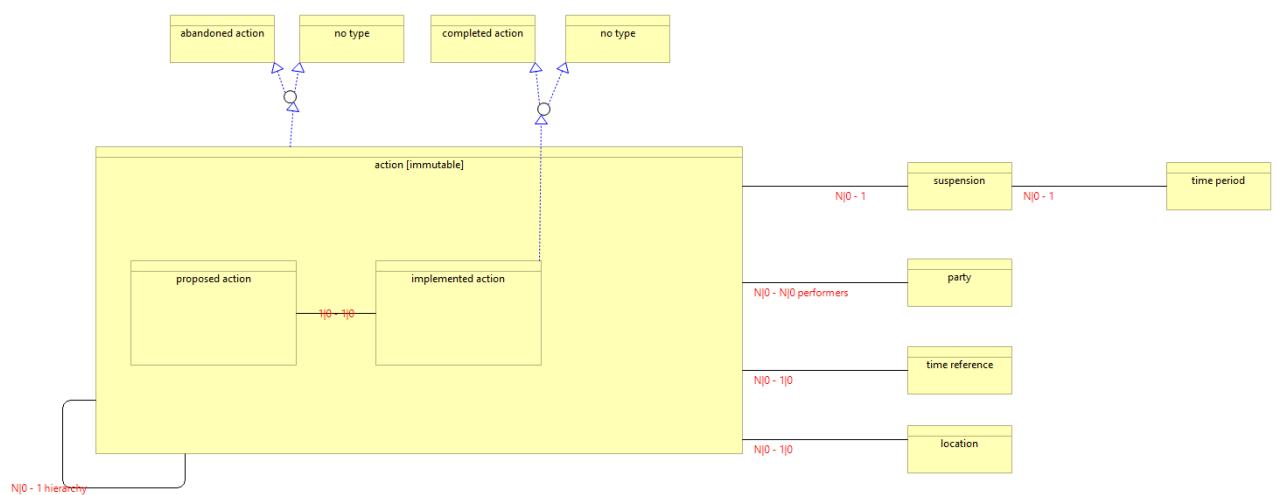
PROPOSED AND IMPLEMENTED ACTION



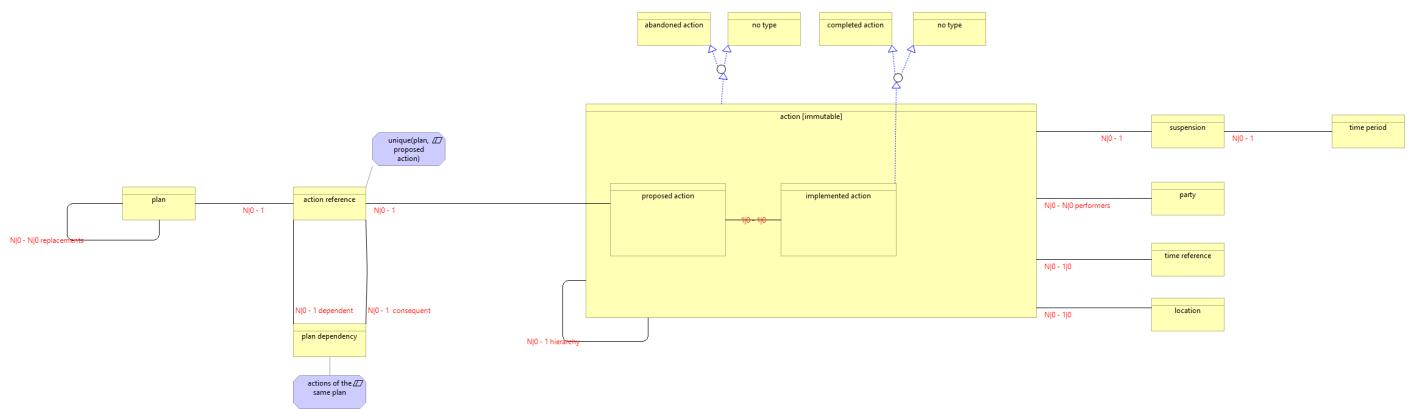
COMPLETED AND ABANDONED ACTIONS



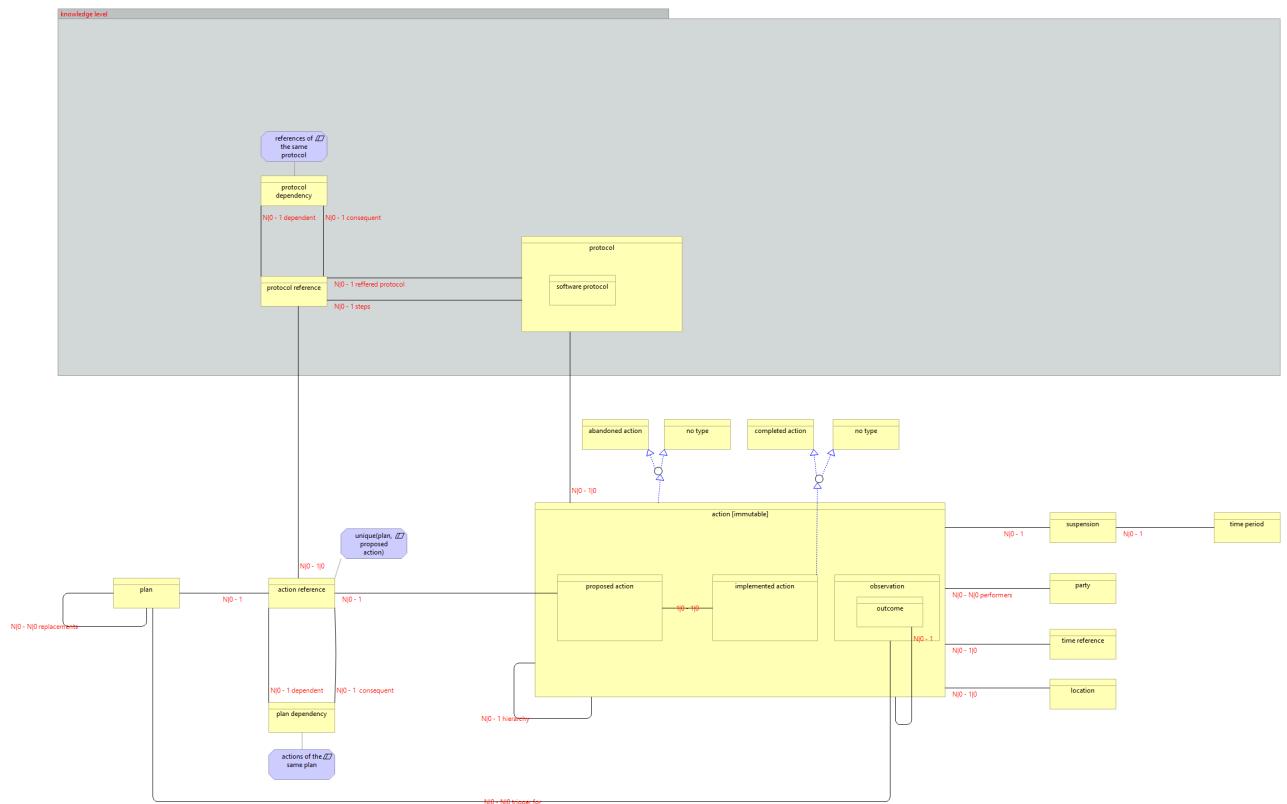
SUSPENSION



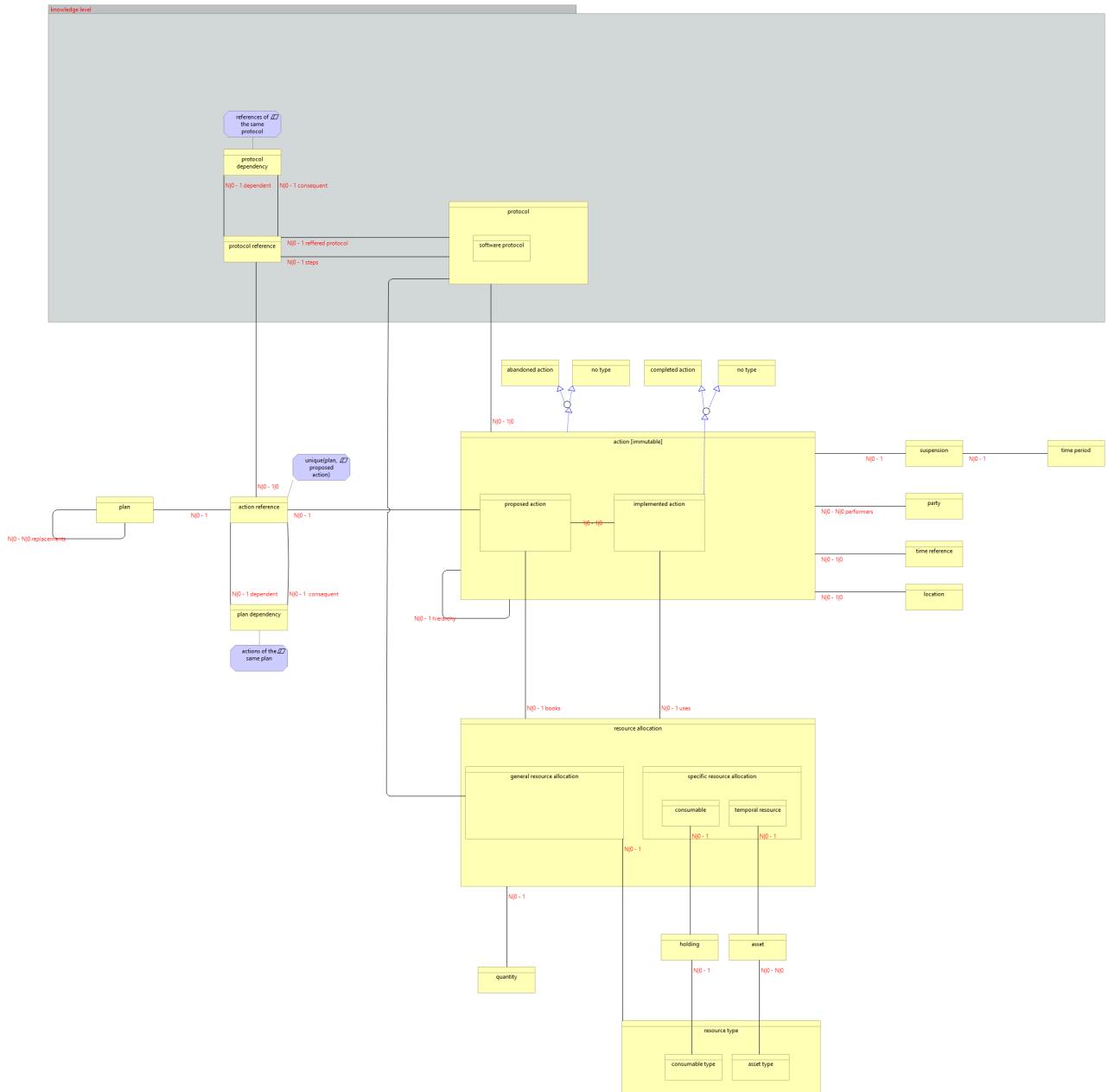
PLAN



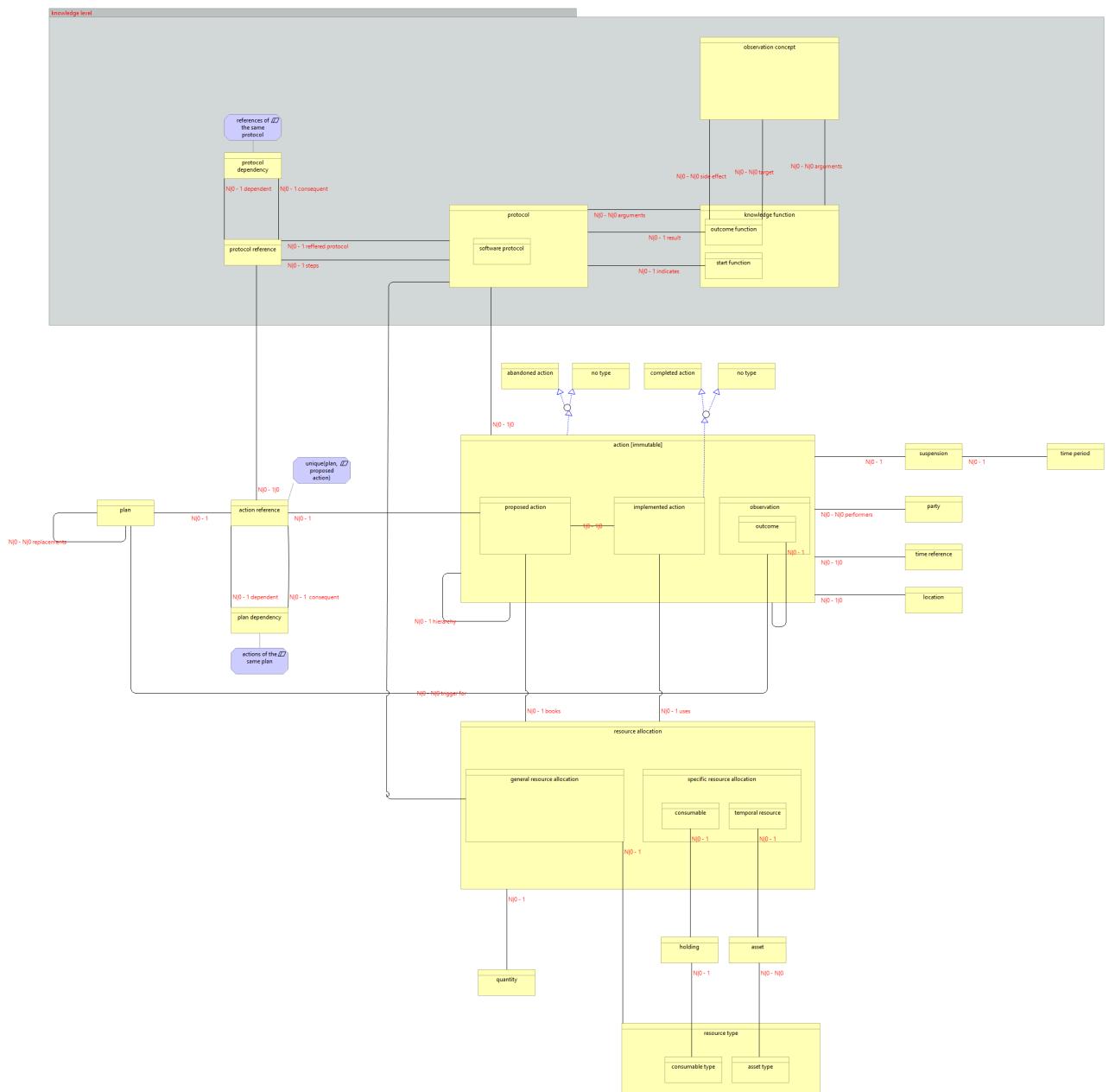
PROTOCOL



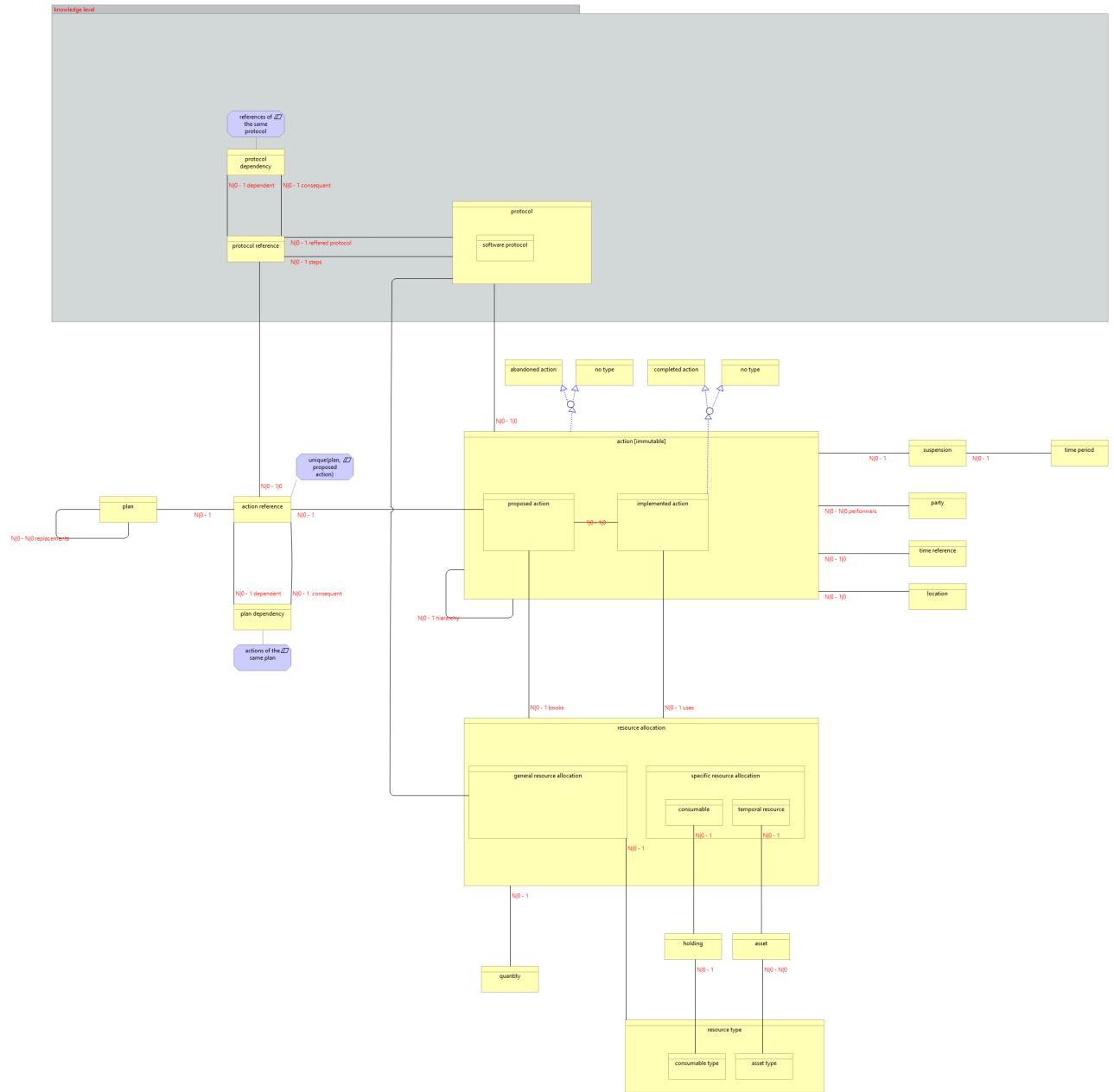
RESOURCE ALLOCATION



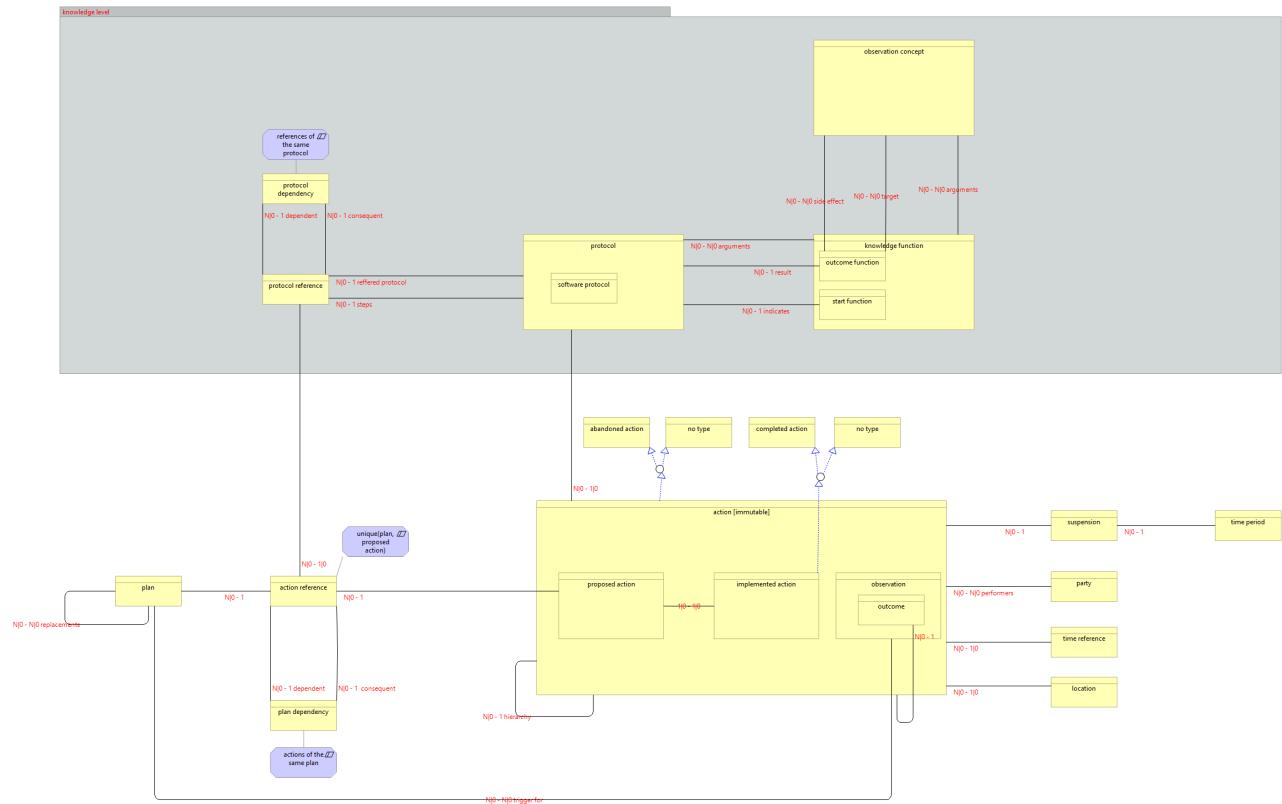
PLANNING



PLANNING (NO OUTCOME)



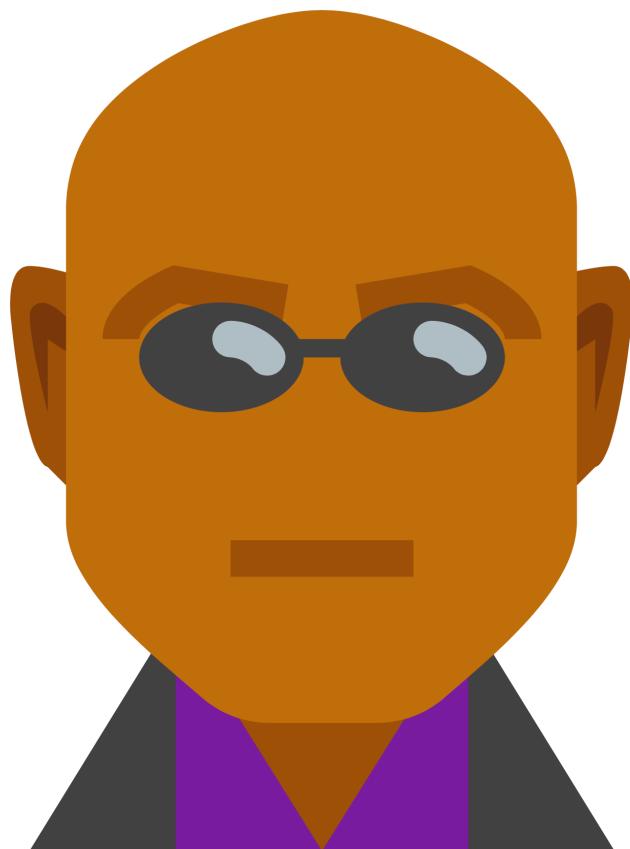
OUTCOME AND START FUNCTIONS



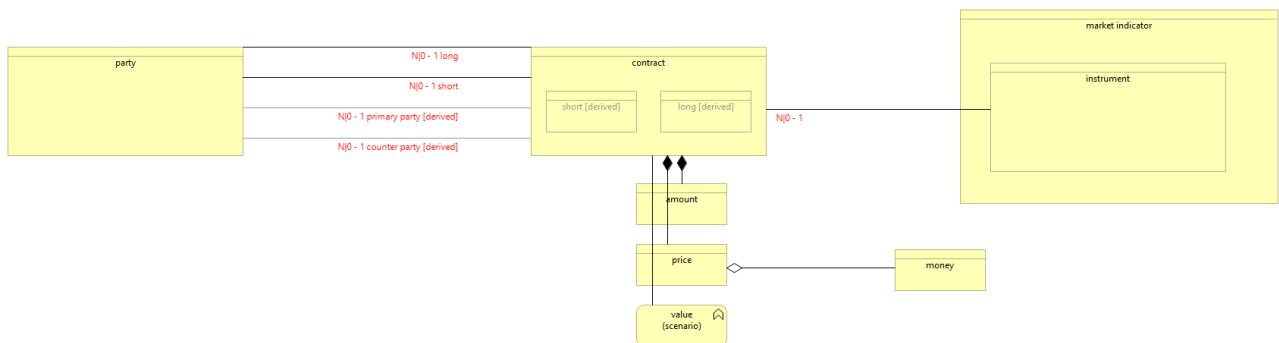
TRADING

ANALYSIS PATTERNS

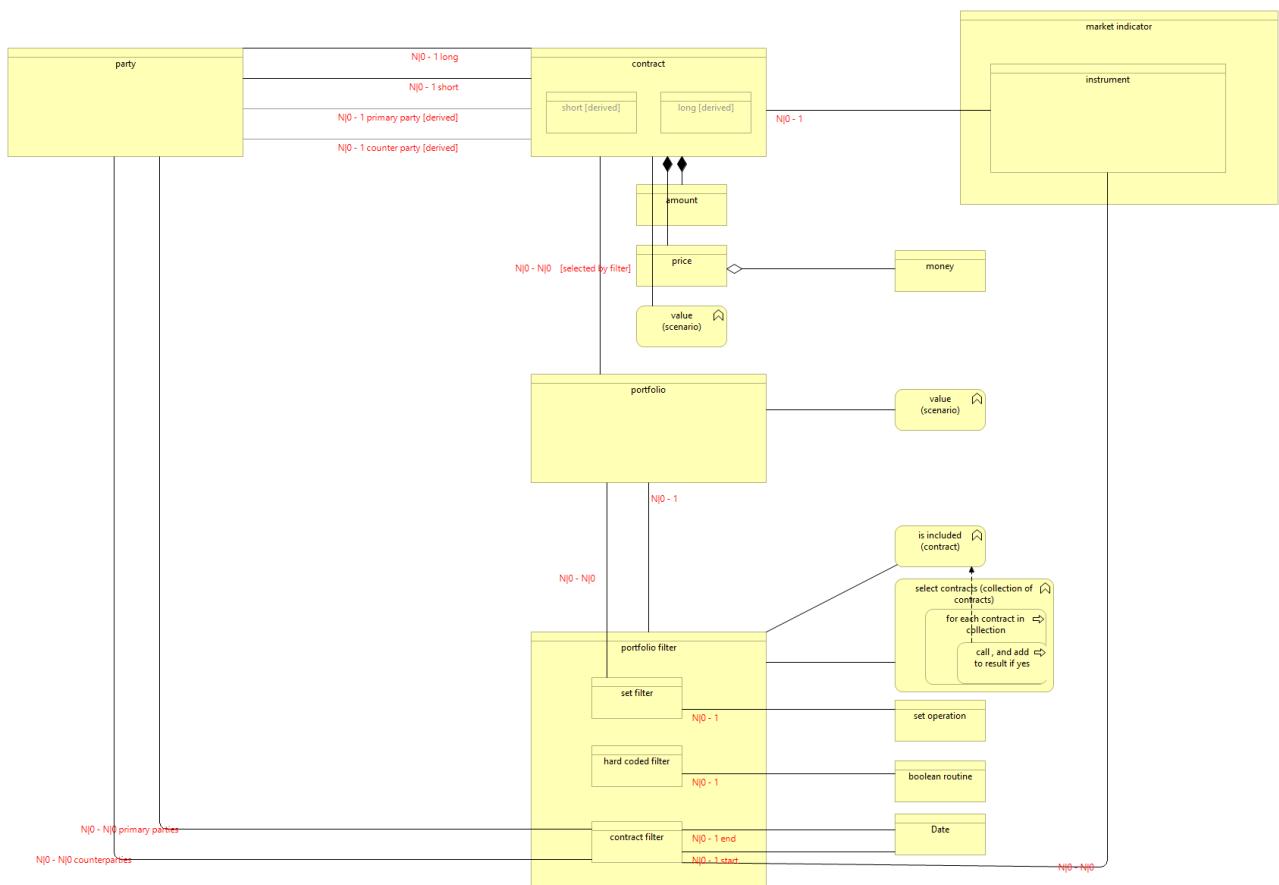
Martin Fowler



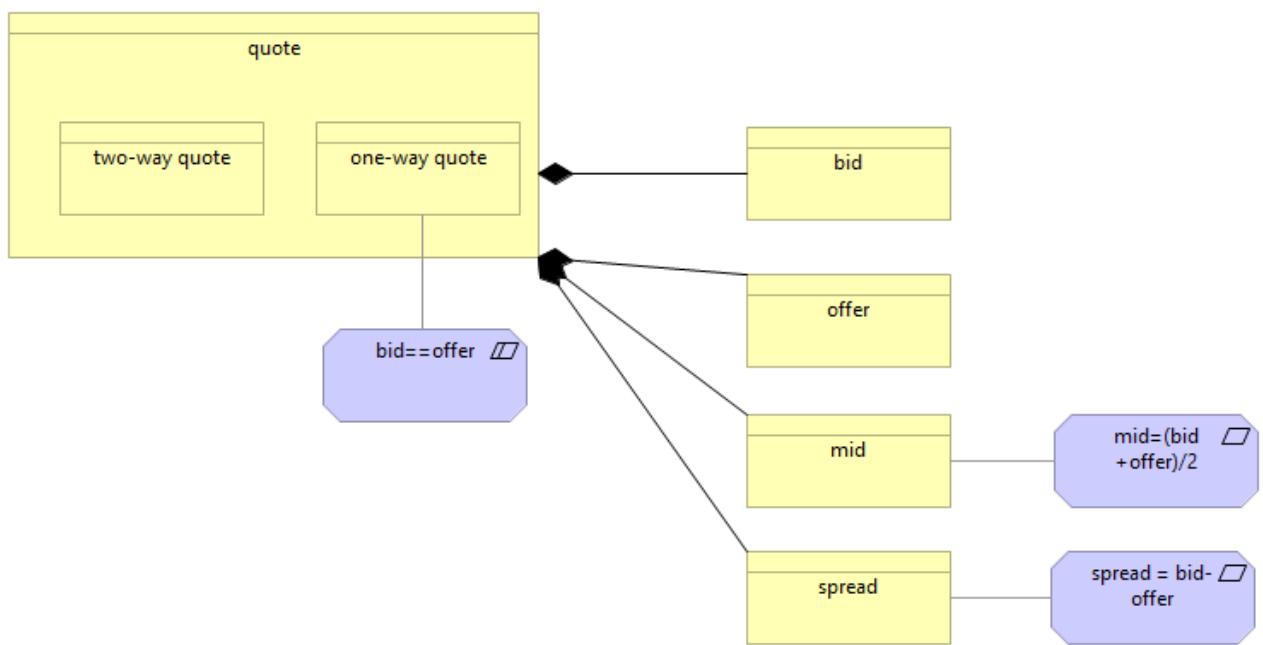
CONTRACT



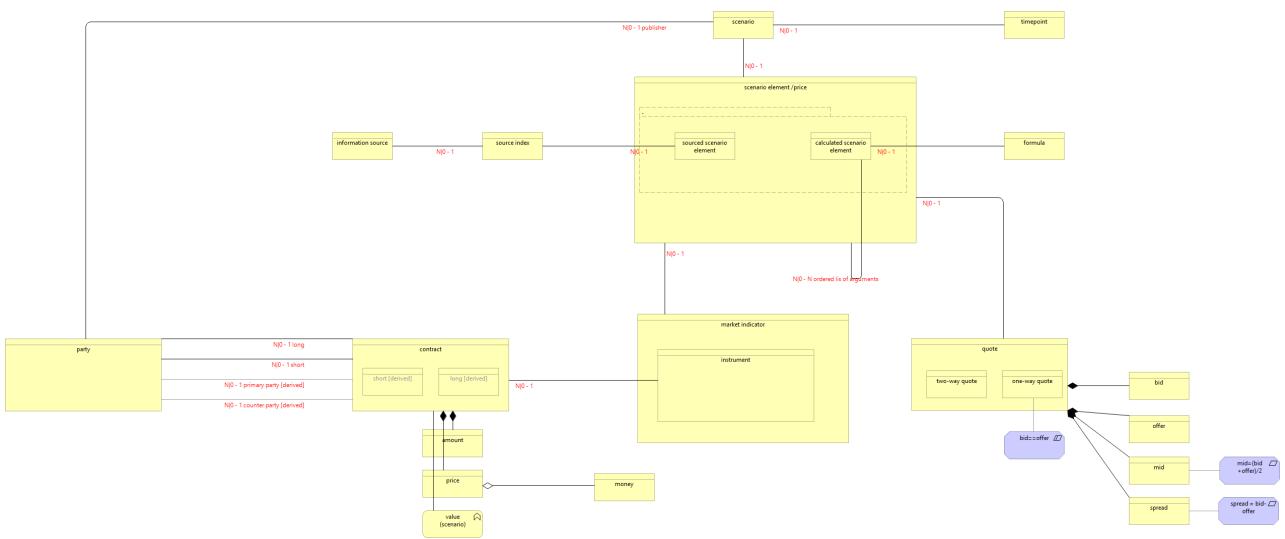
PORTFOLIO



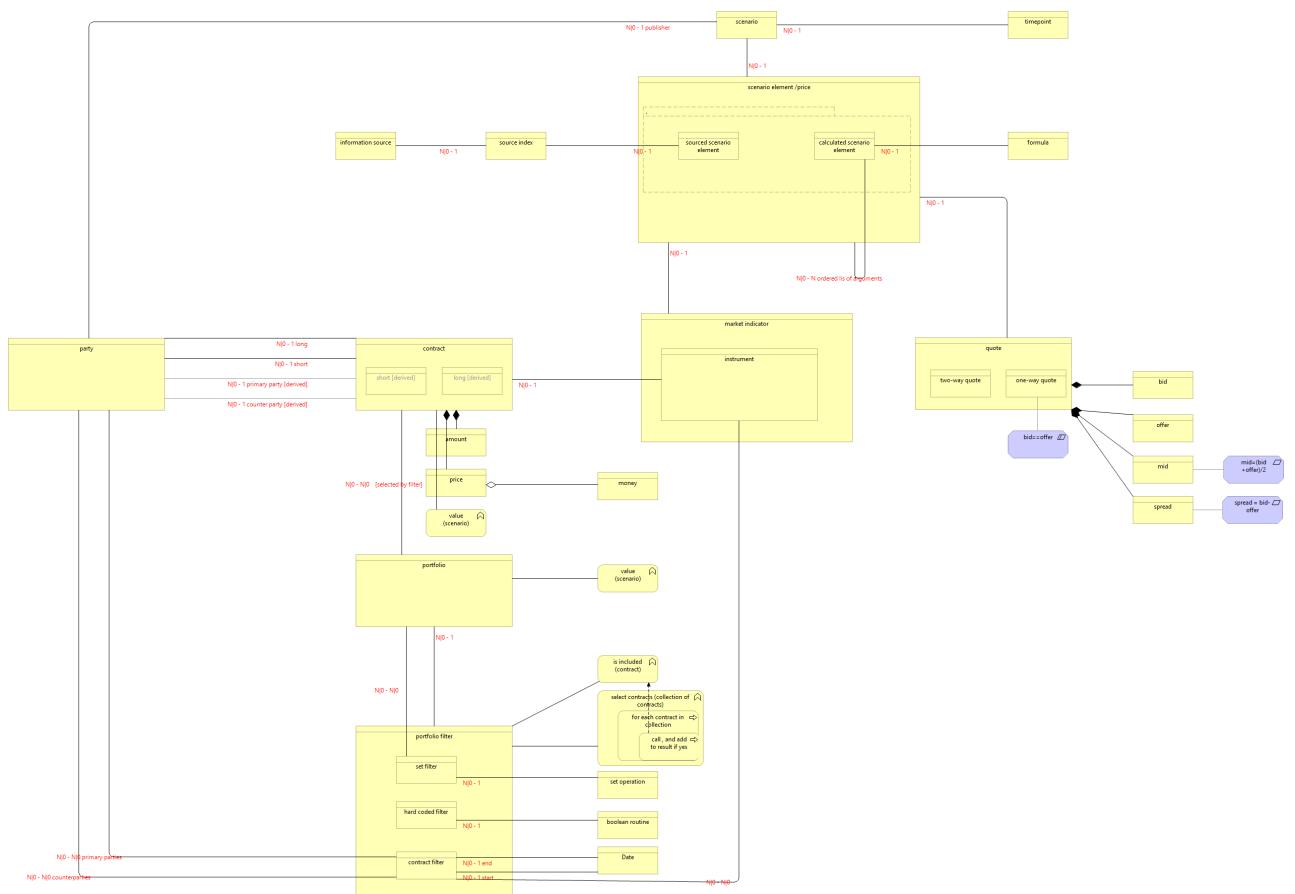
QUOTE



SCENARIO



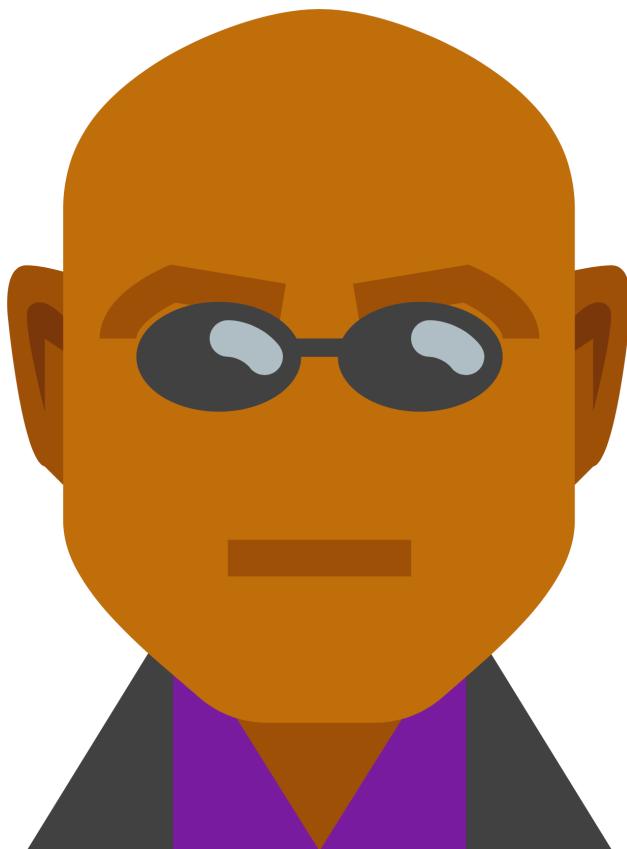
TRADING



DERIVATIVE CONTRACTS

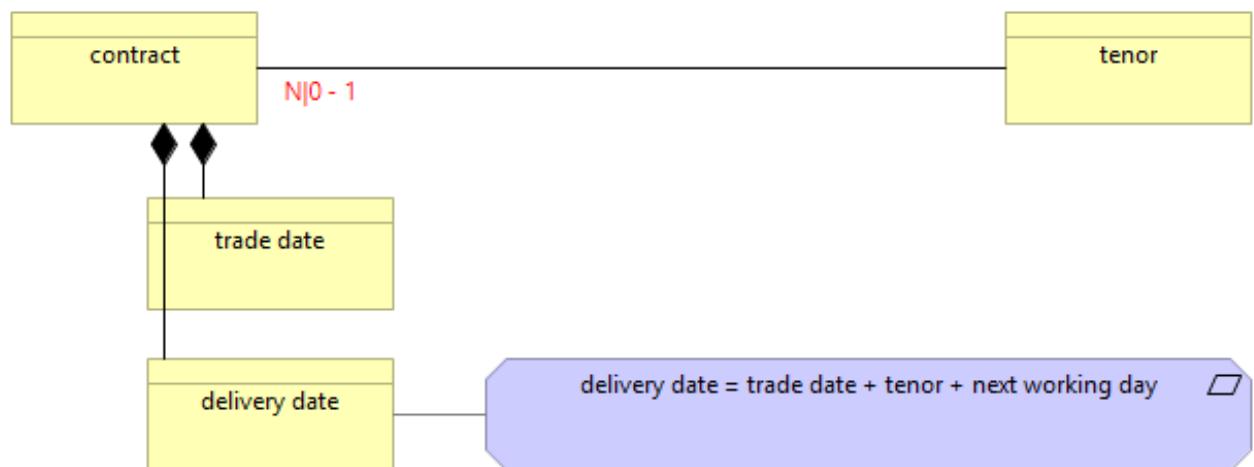
ANALYSIS PATTERNS

Martin Fowler



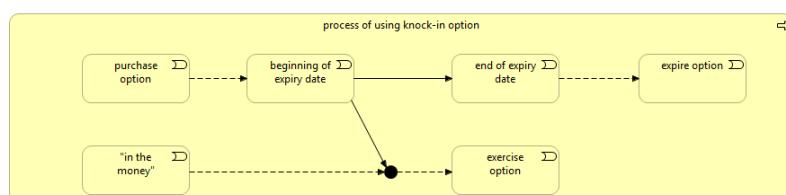
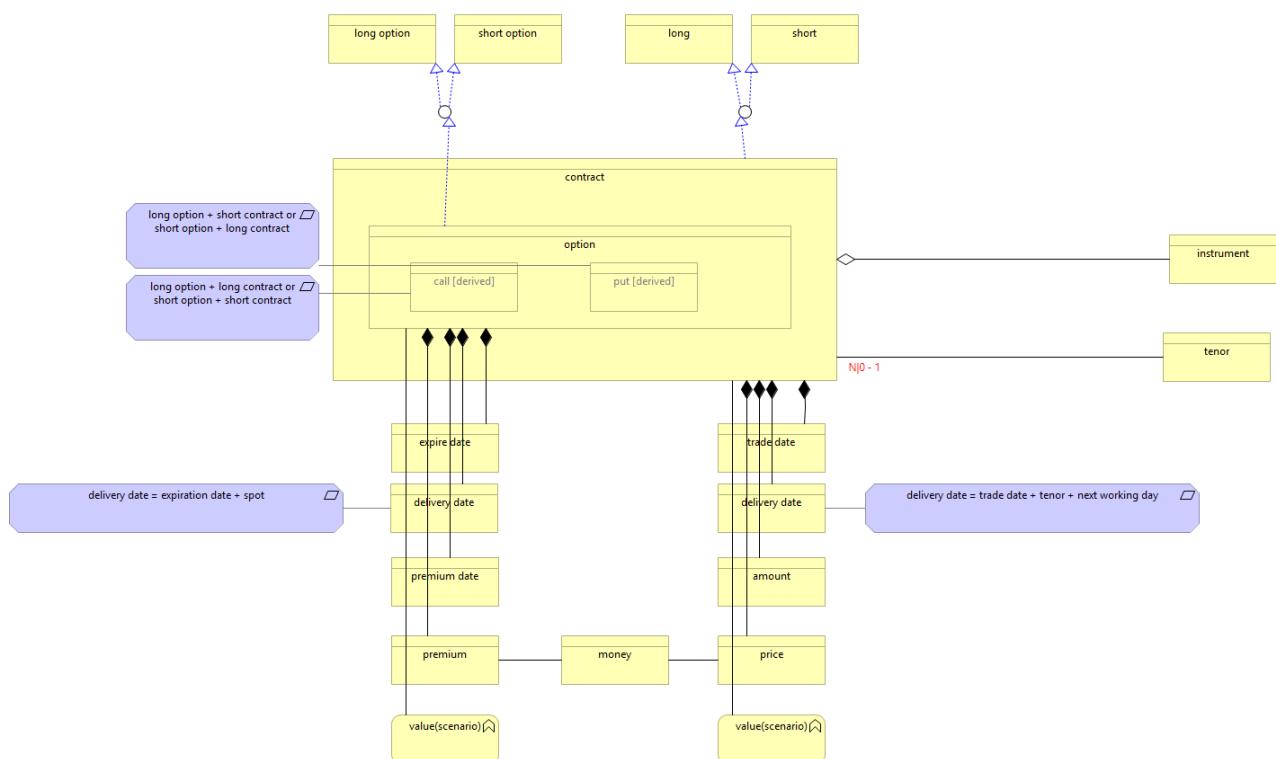
FORWARD CONTRACTS

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



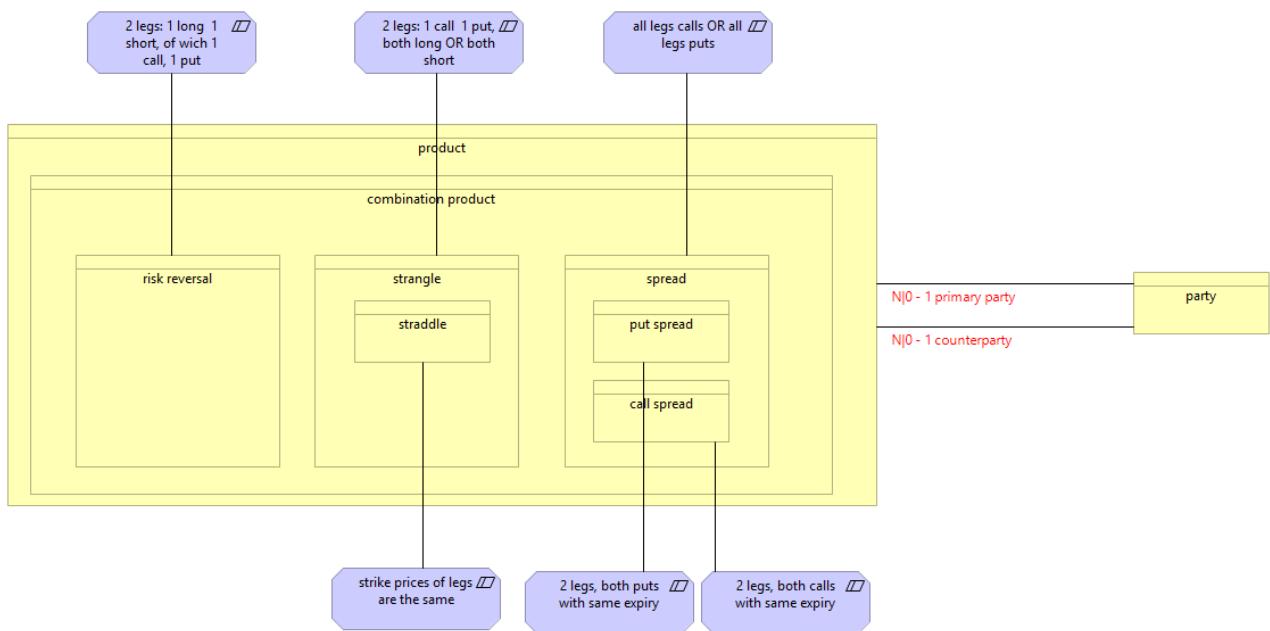
OPTIONS

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



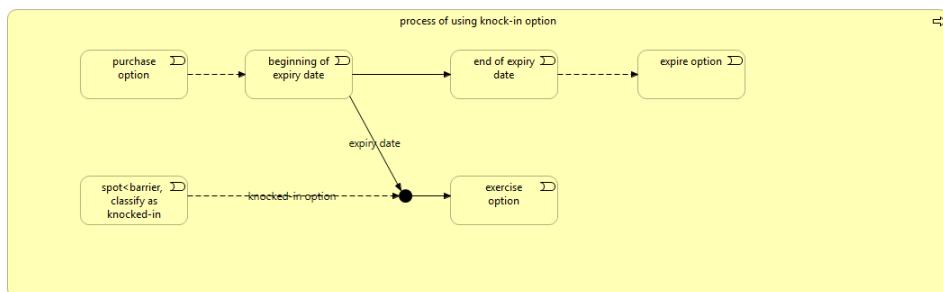
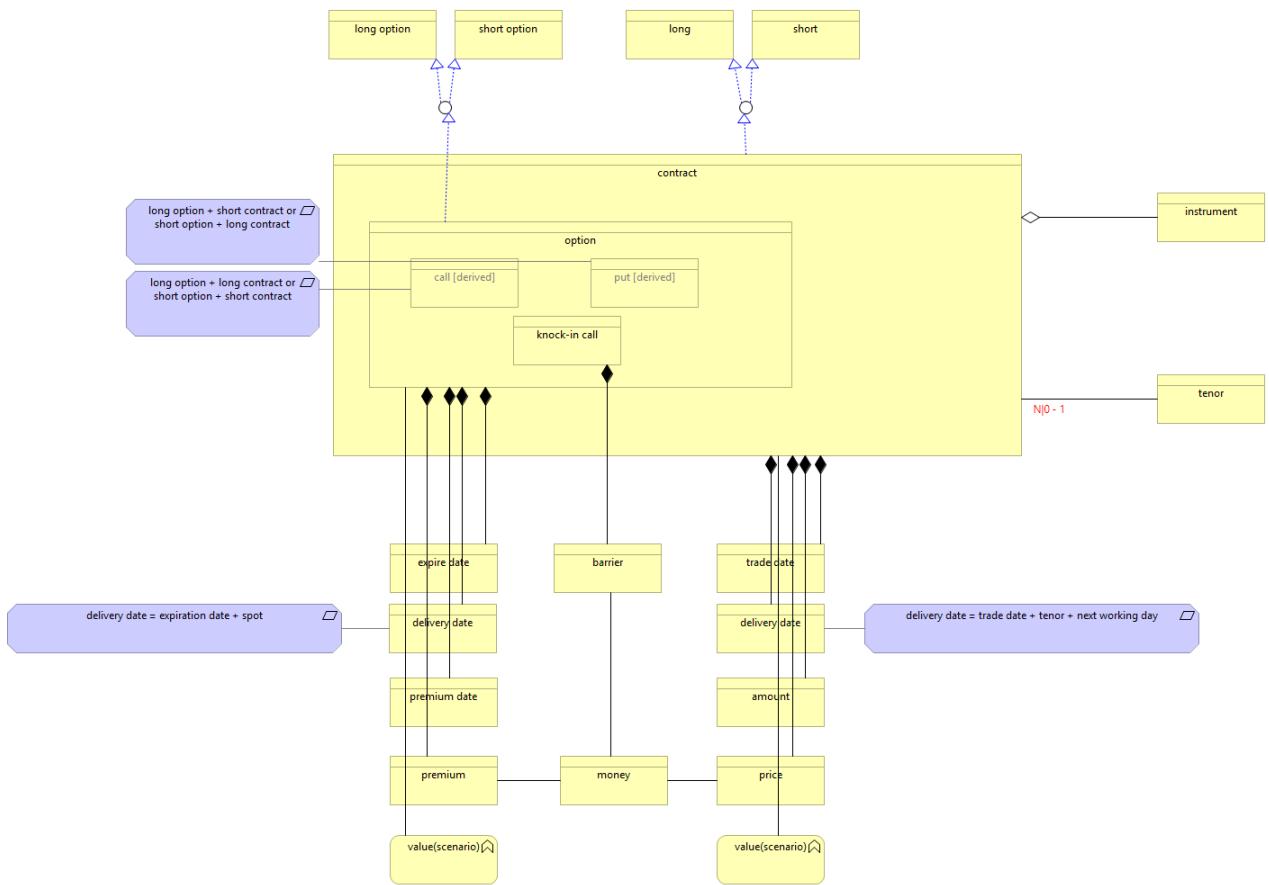
PRODUCT

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



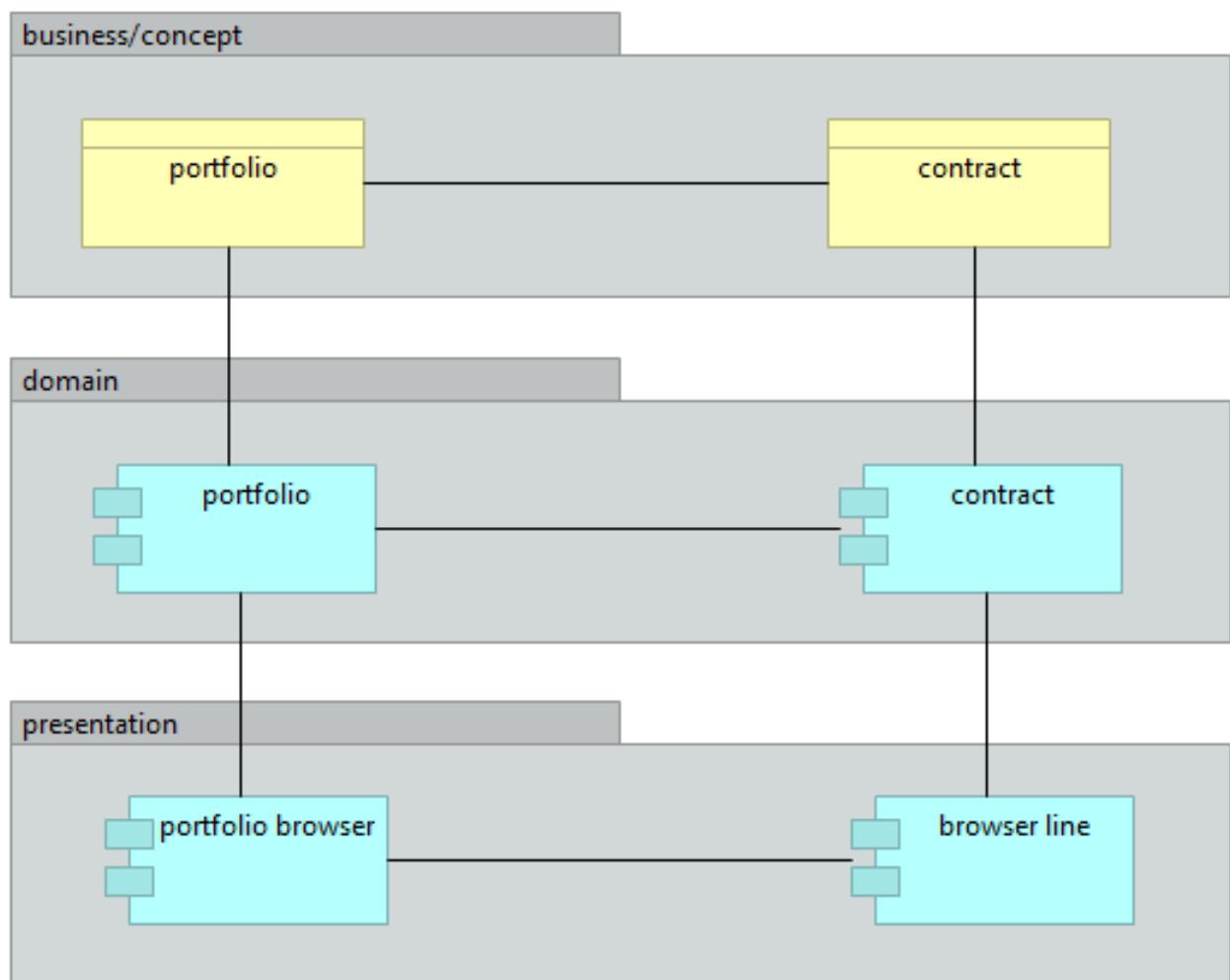
SUBTYPE STATE MACHINES

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.

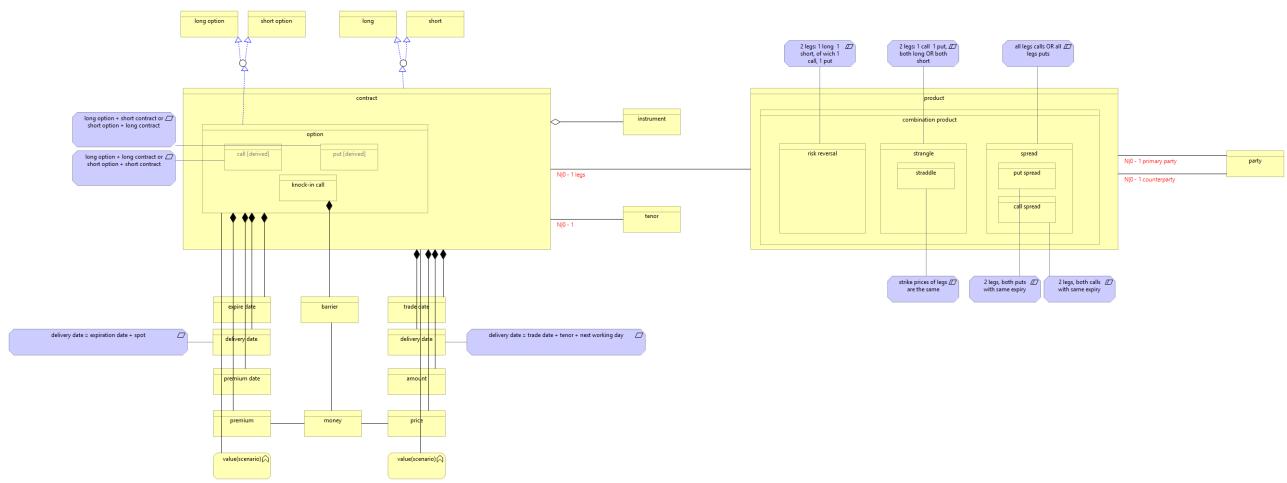


PARALLEL APPLICATION AND DOMAIN HIERARCHIES

example, a report containing a list of contracts



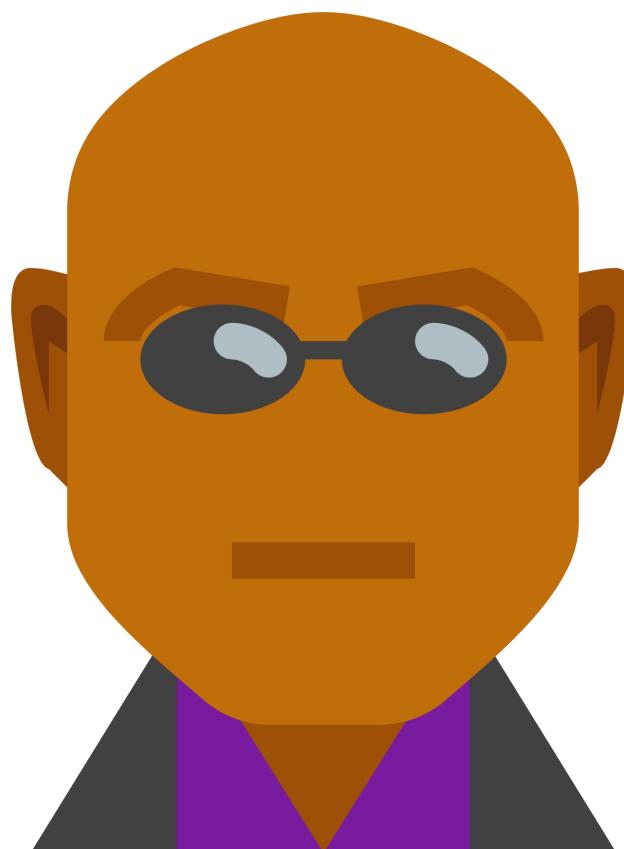
DERIVATIVE CONTRACTS



TRADING PACKAGES

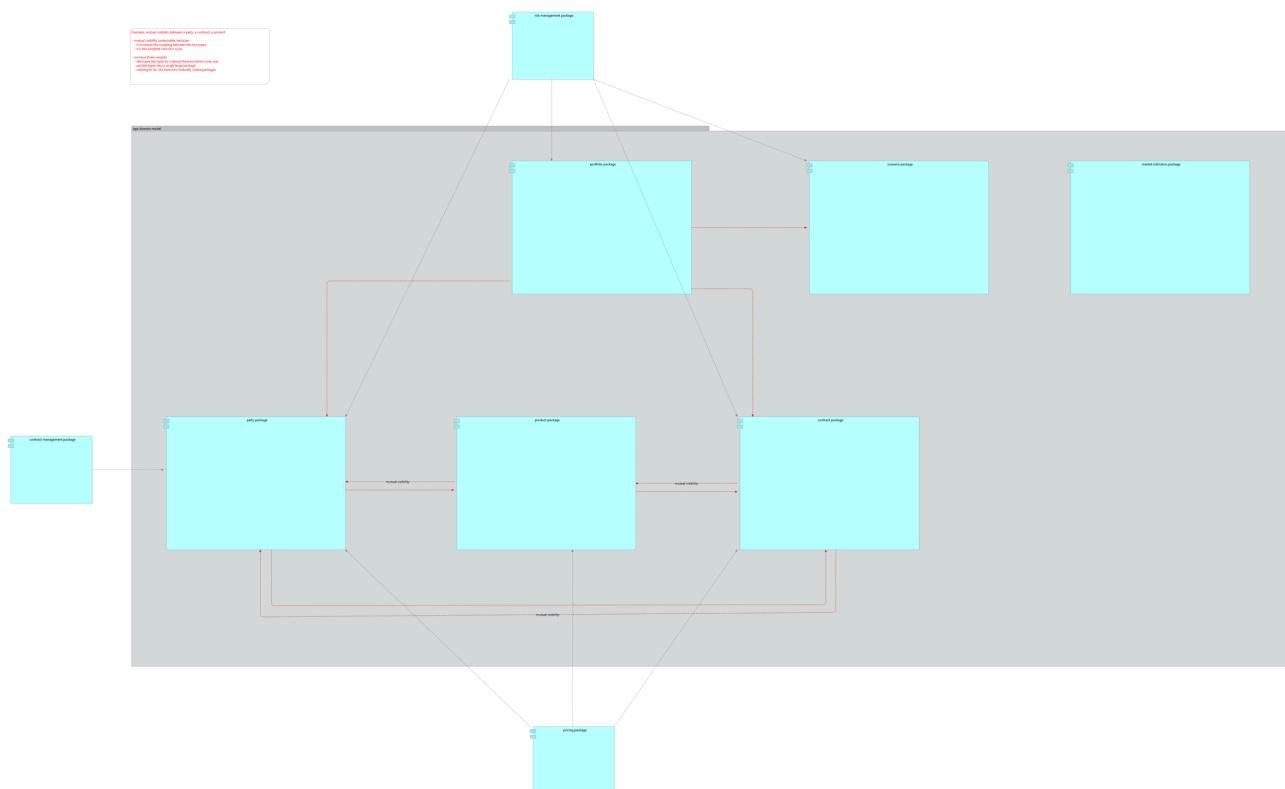
ANALYSIS PATTERNS

Martin Fowler

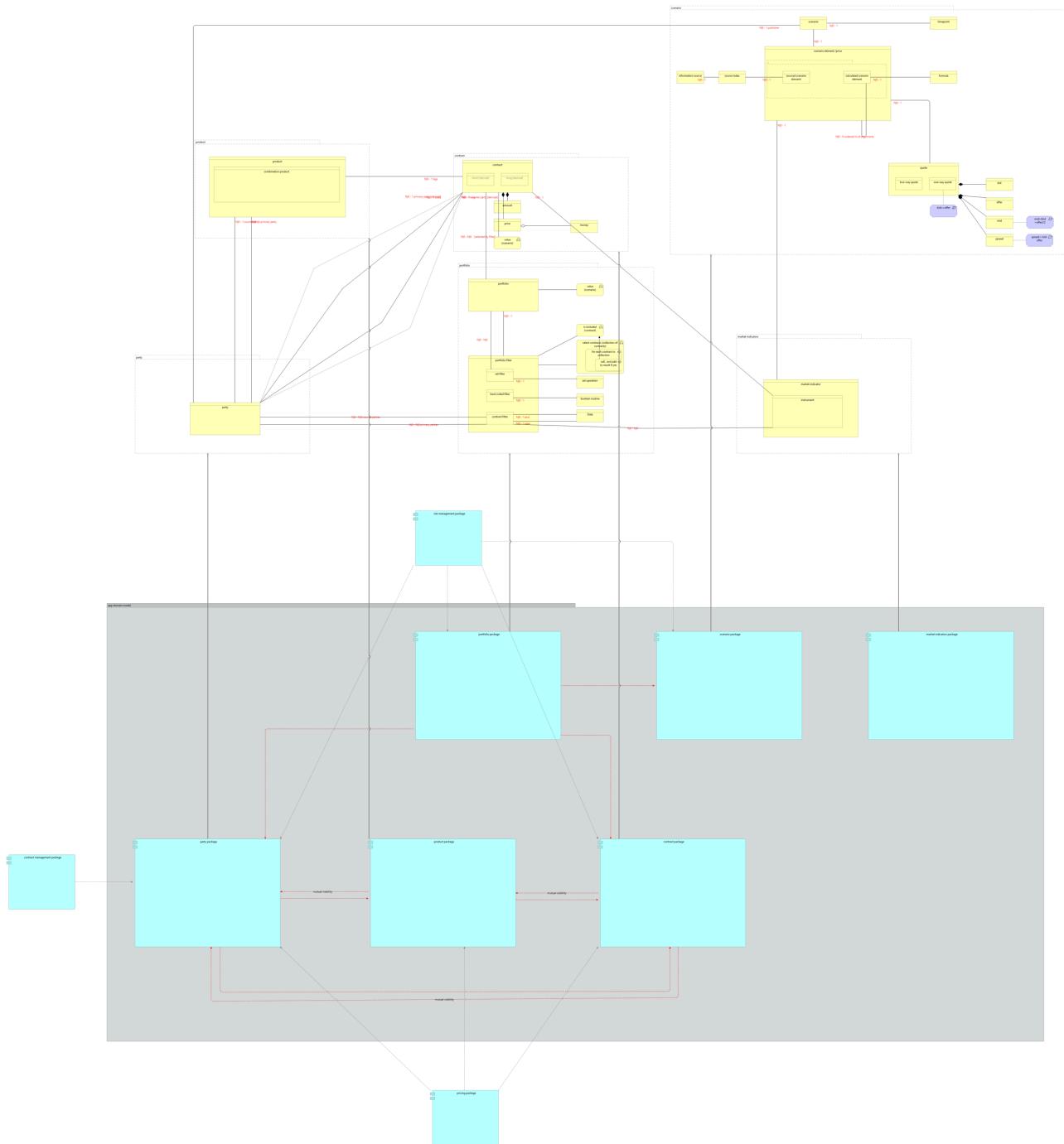


MULTIPLE ACCESS LEVELS TO A PACKAGE

MUTUAL VISIBILITY



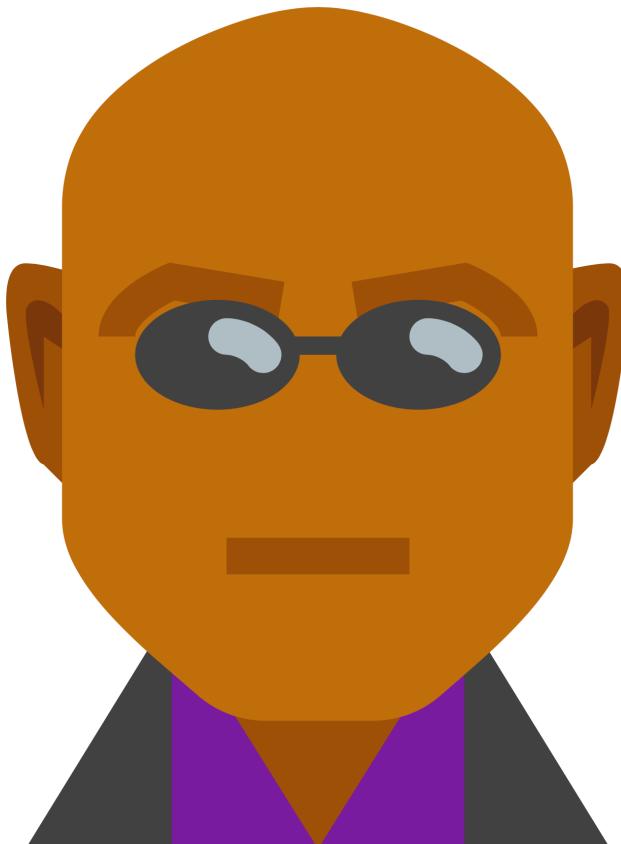
TRADING PACKAGES



LAYERED ARCHITECTURE

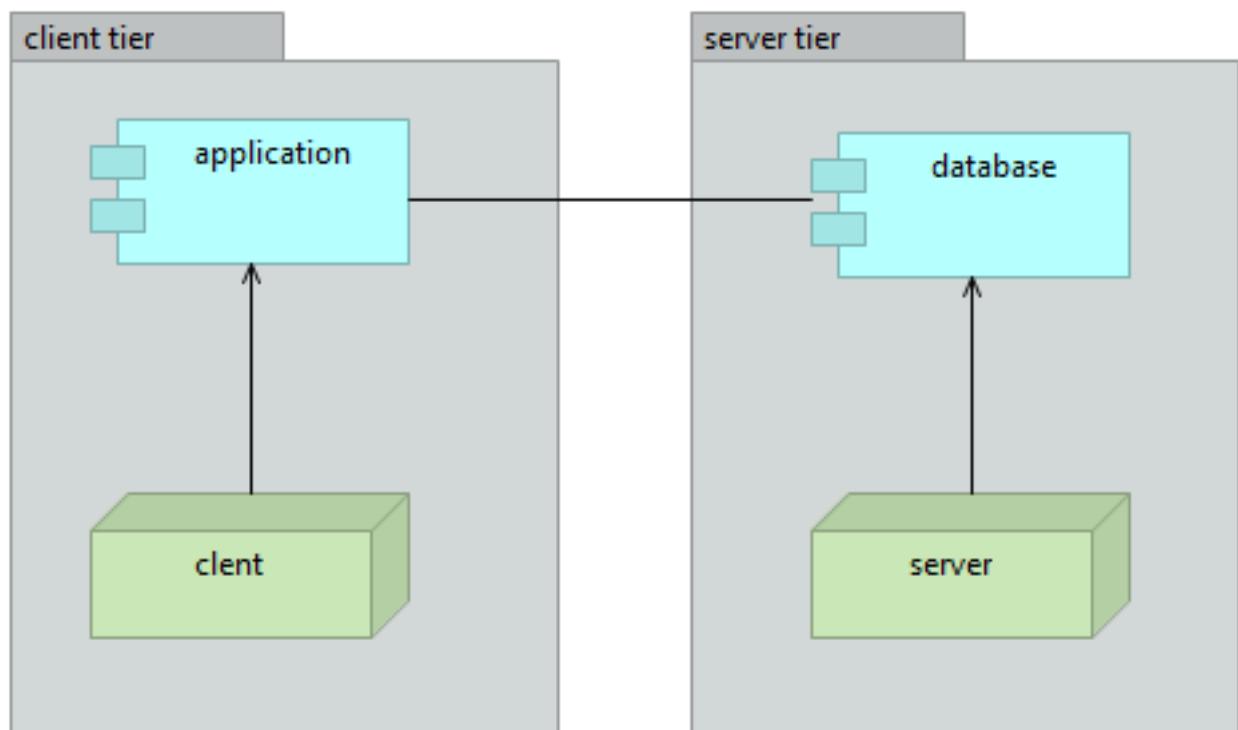
ANALYSIS PATTERNS

Martin Fowler

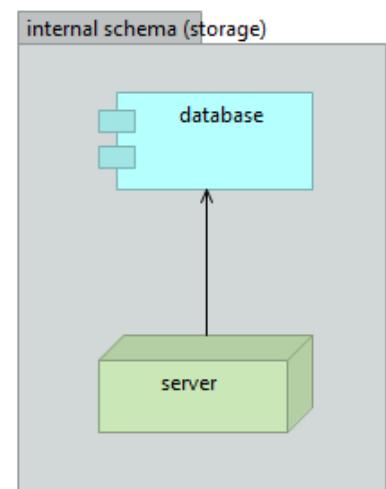
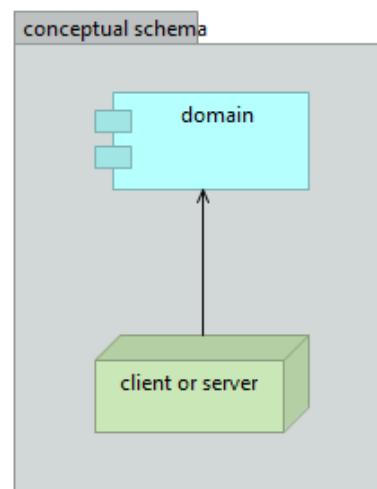
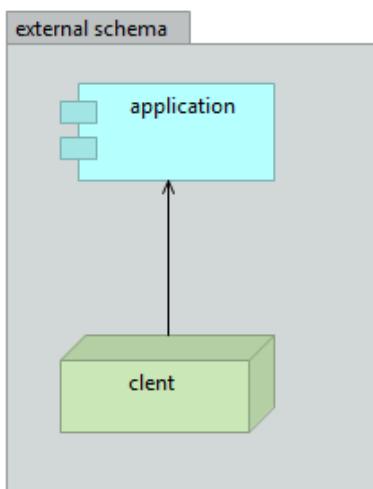


TWO-TIER ARCHITECTURE

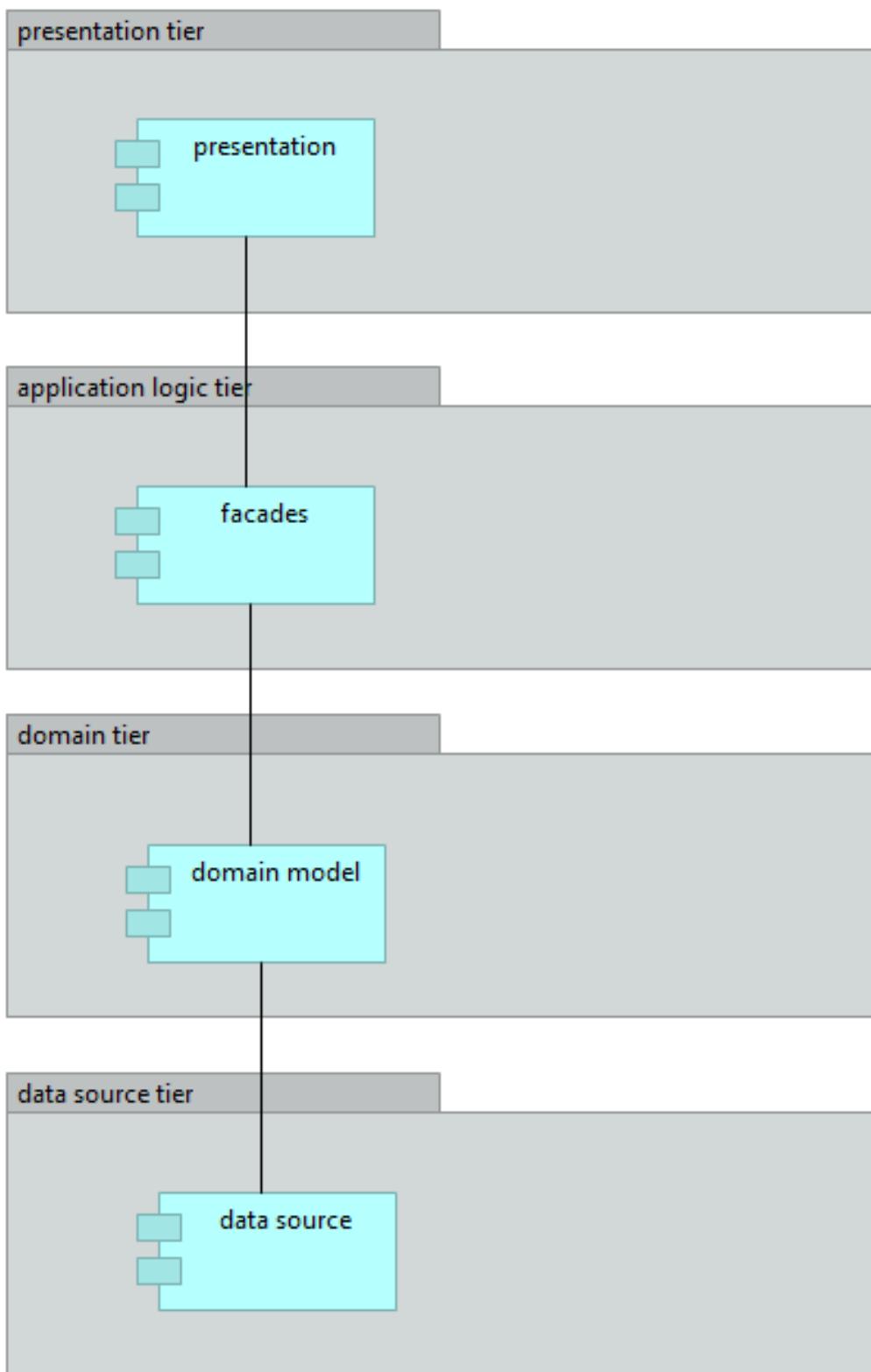
two-tier architecture



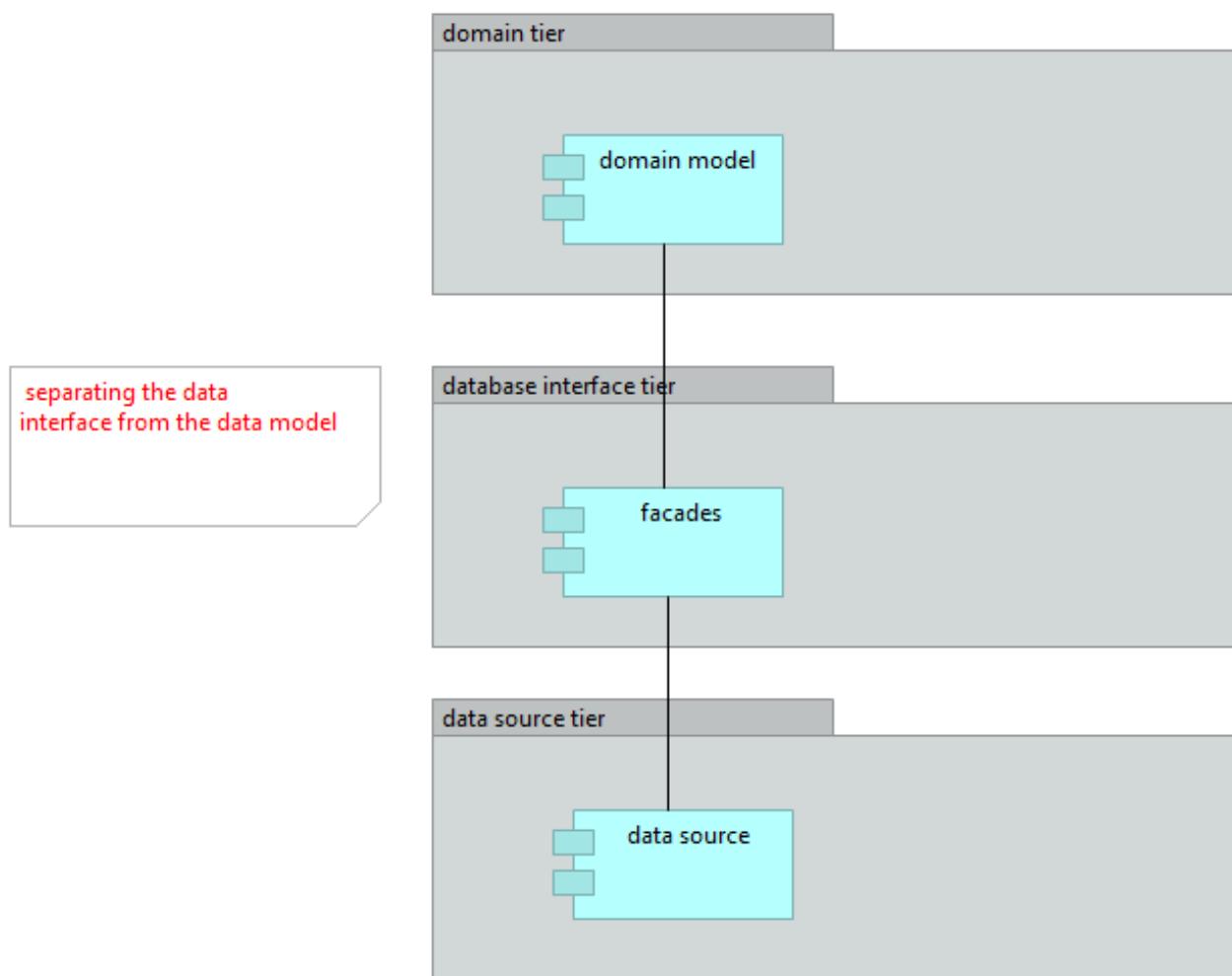
THREE-TIER ARCHITECTURE



PRESENTATION AND APPLICATION LOGIC



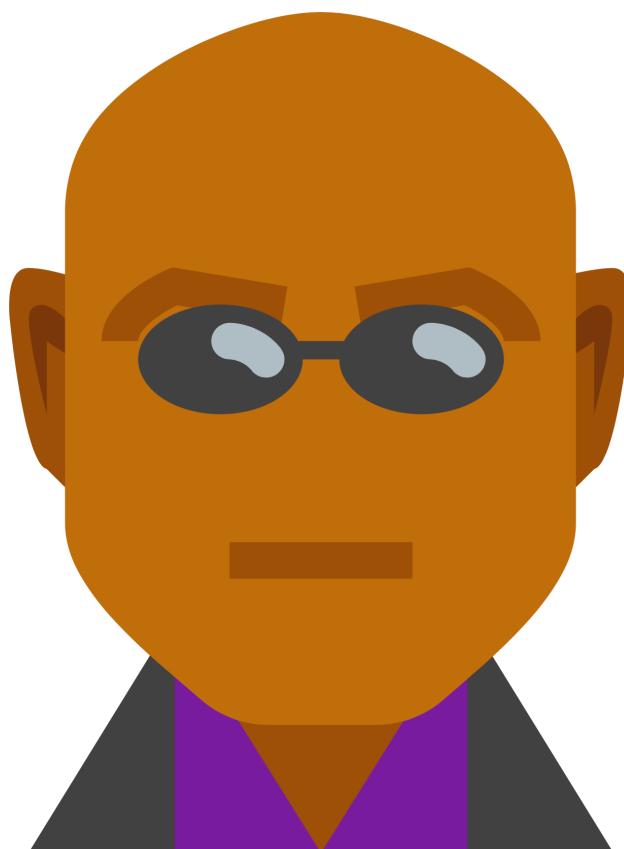
DATABASE INTERACTION



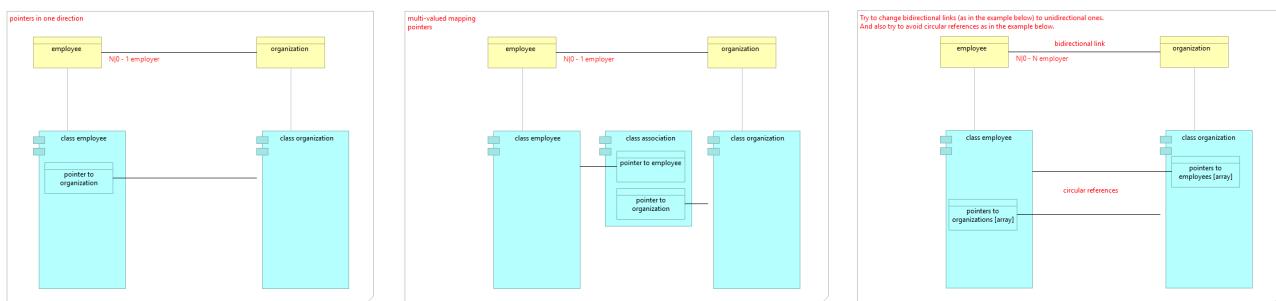
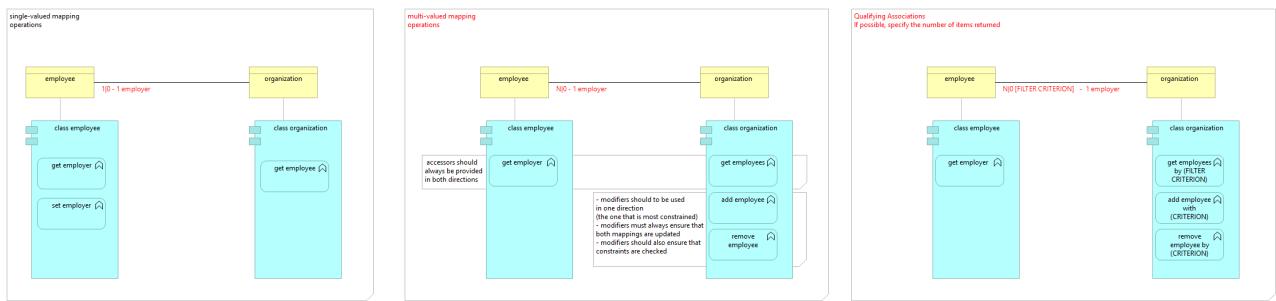
TYPE MODEL DESIGN

ANALYSIS PATTERNS

Martin Fowler

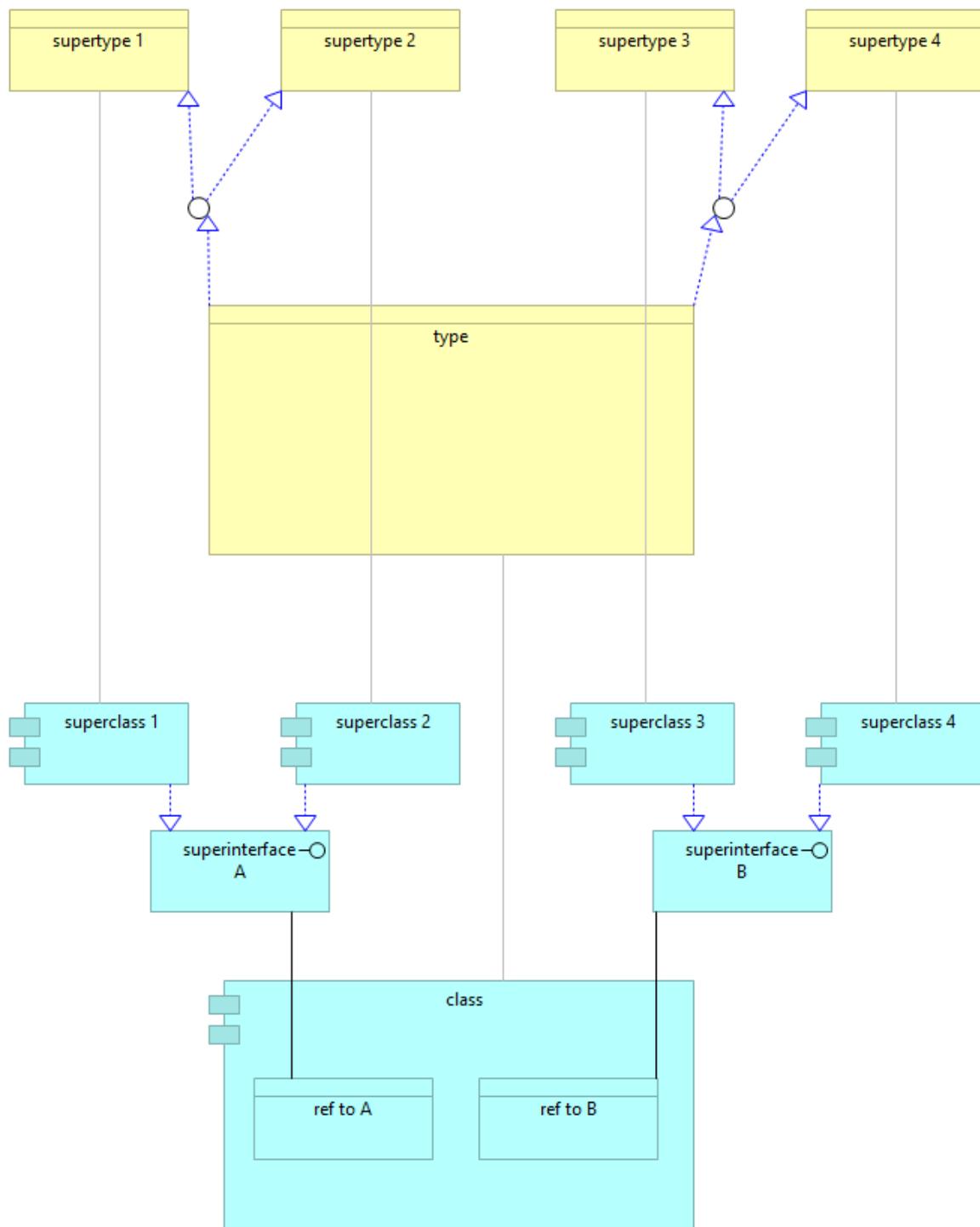


IMPLEMENTING ASSOCIATIONS

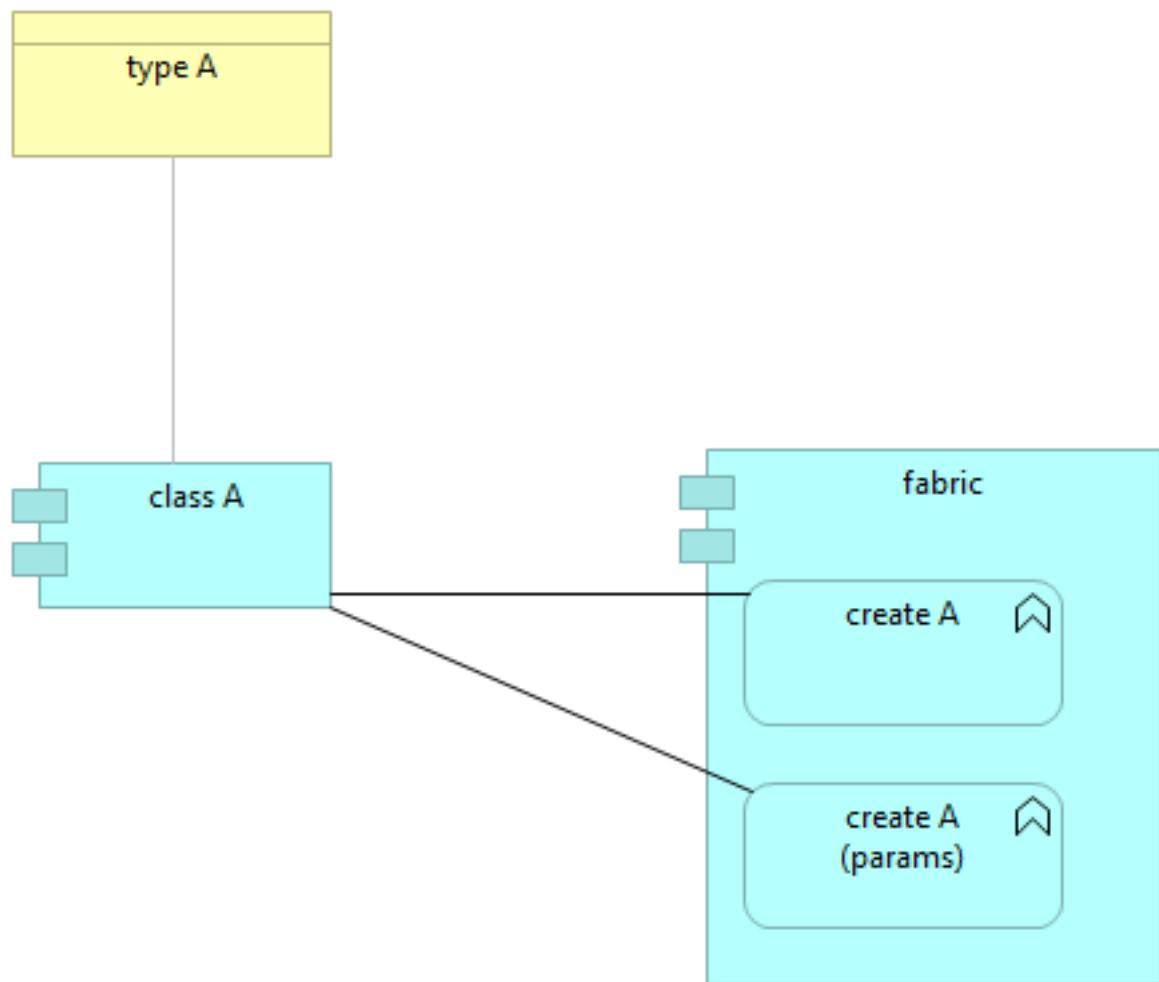


IMPLEMENTING GENERALIZATION

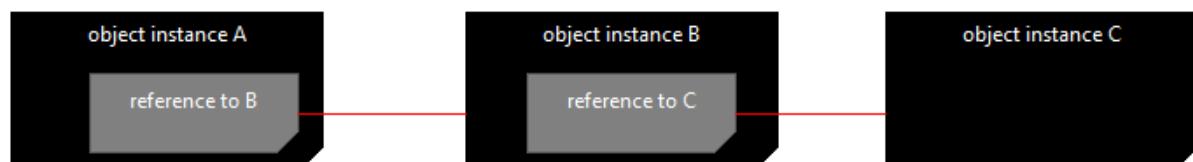
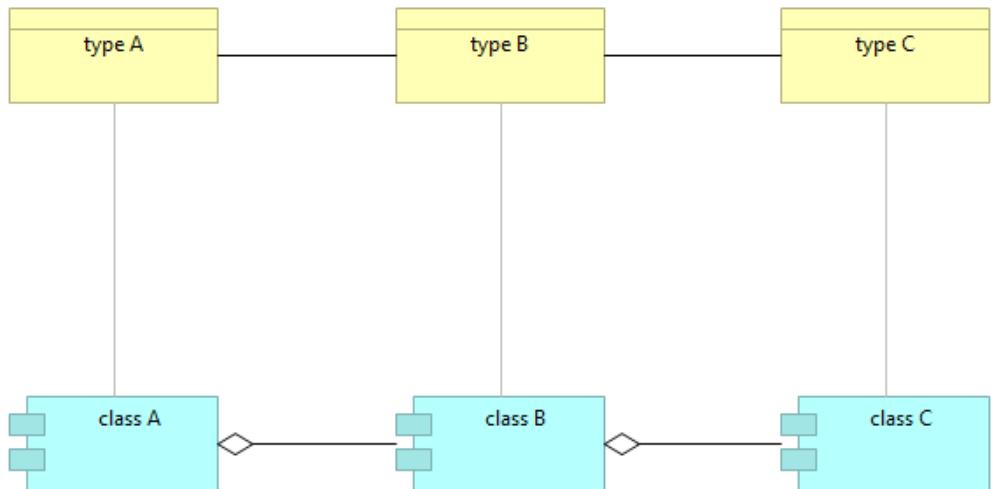
- example of implementation of multiple inheritance
- for example, the composition is applicable instead of inheritance



OBJECT CREATION



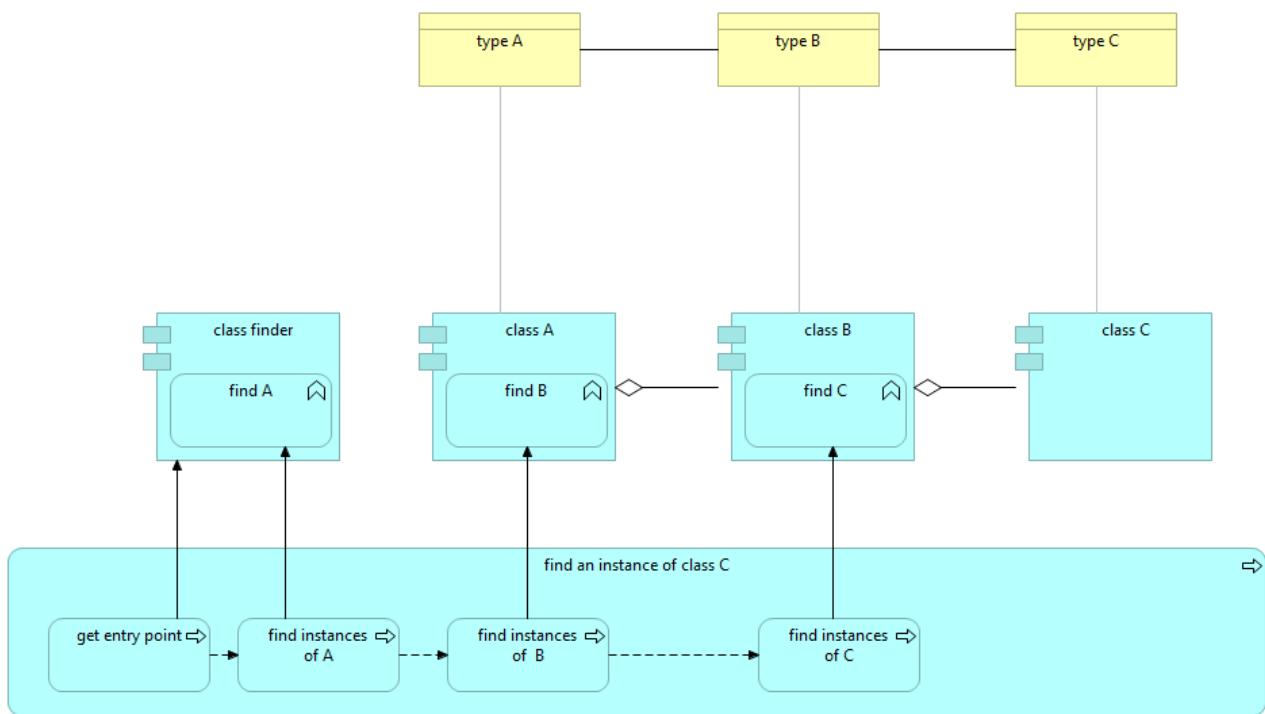
OBJECT DESTRUCTION



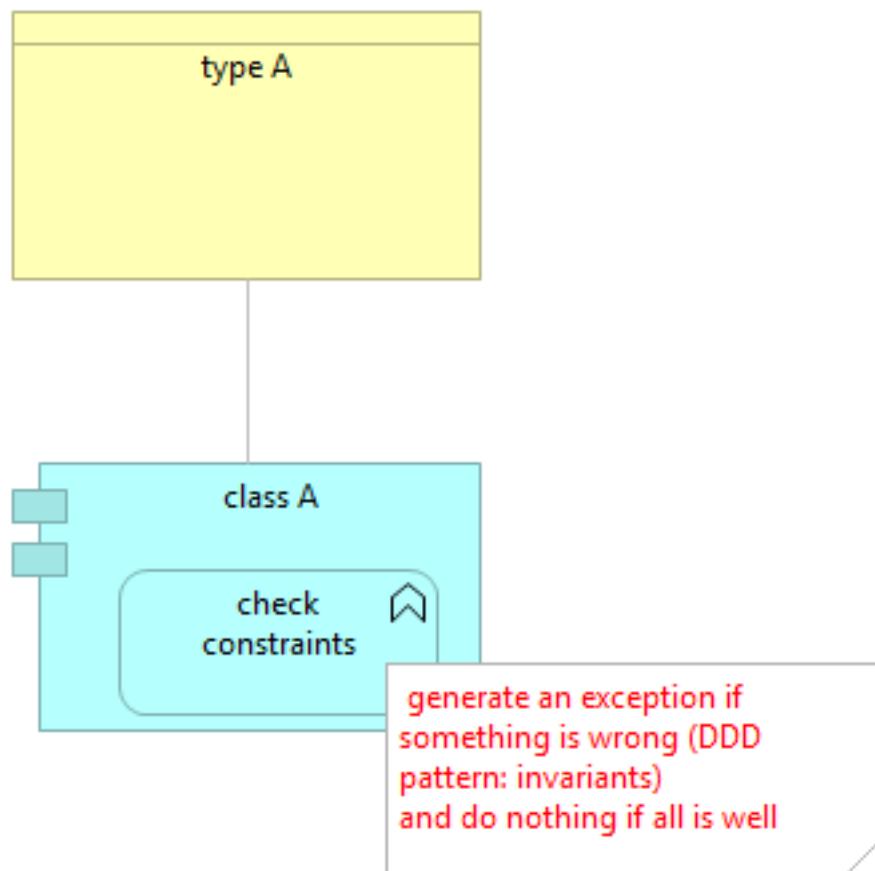
for example, delete object B
- all constraints are met
- no dangling references

object instance A

ENTRY POINT.



IMPLEMENTING CONSTRAINTS

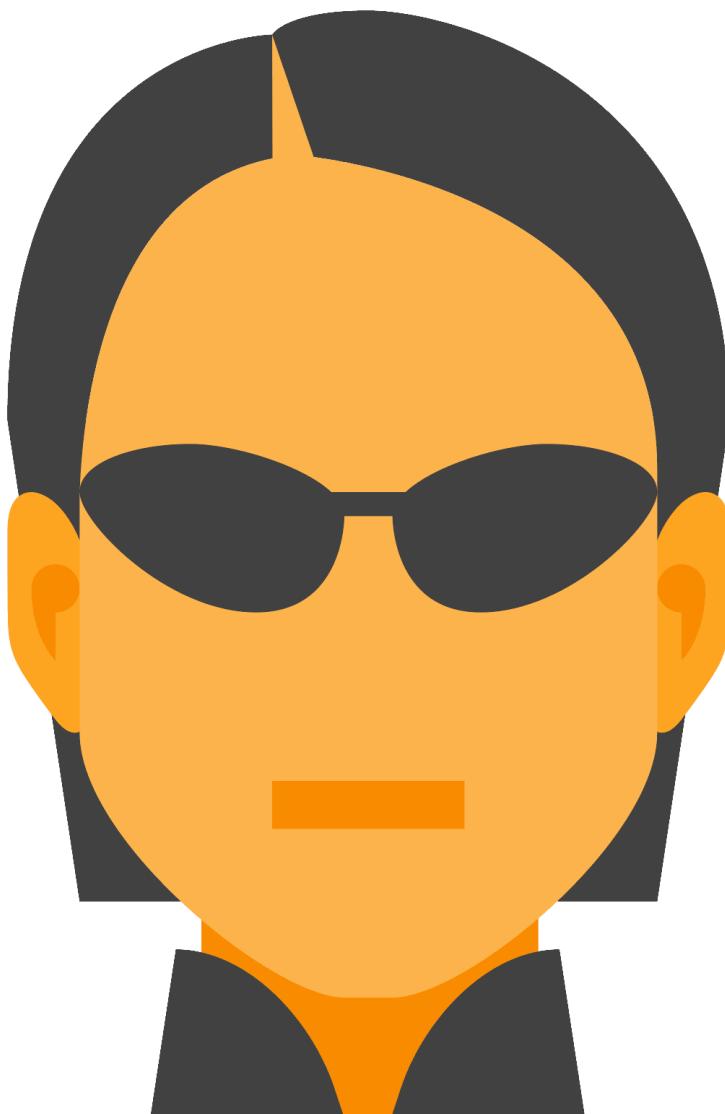


04

Domain Driven Design

Tackling Complexity in the Heart
of Software.

ERIC EVANS



MODEL AND STRUCTURAL ELEMENTS

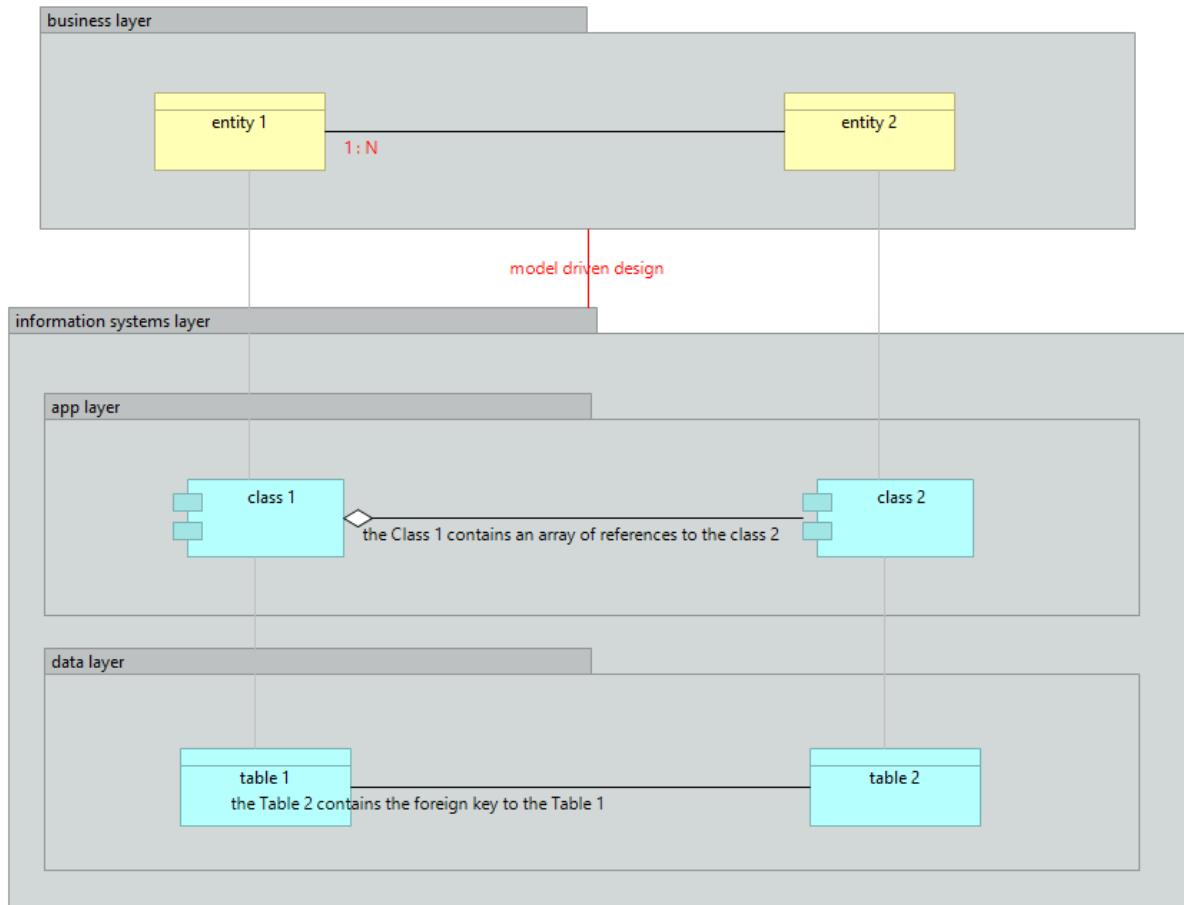
DOMAIN DRIVEN DESIGN

Eric Evans

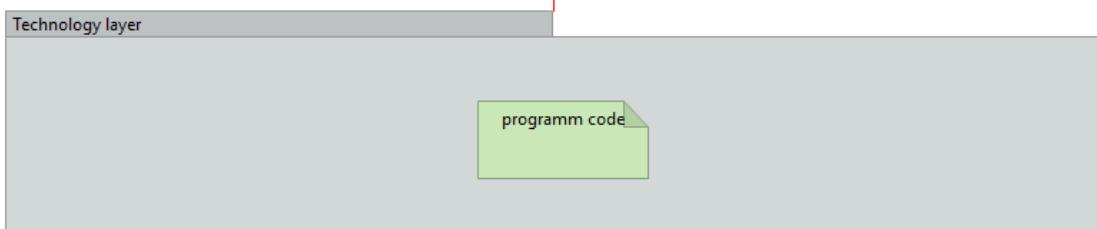


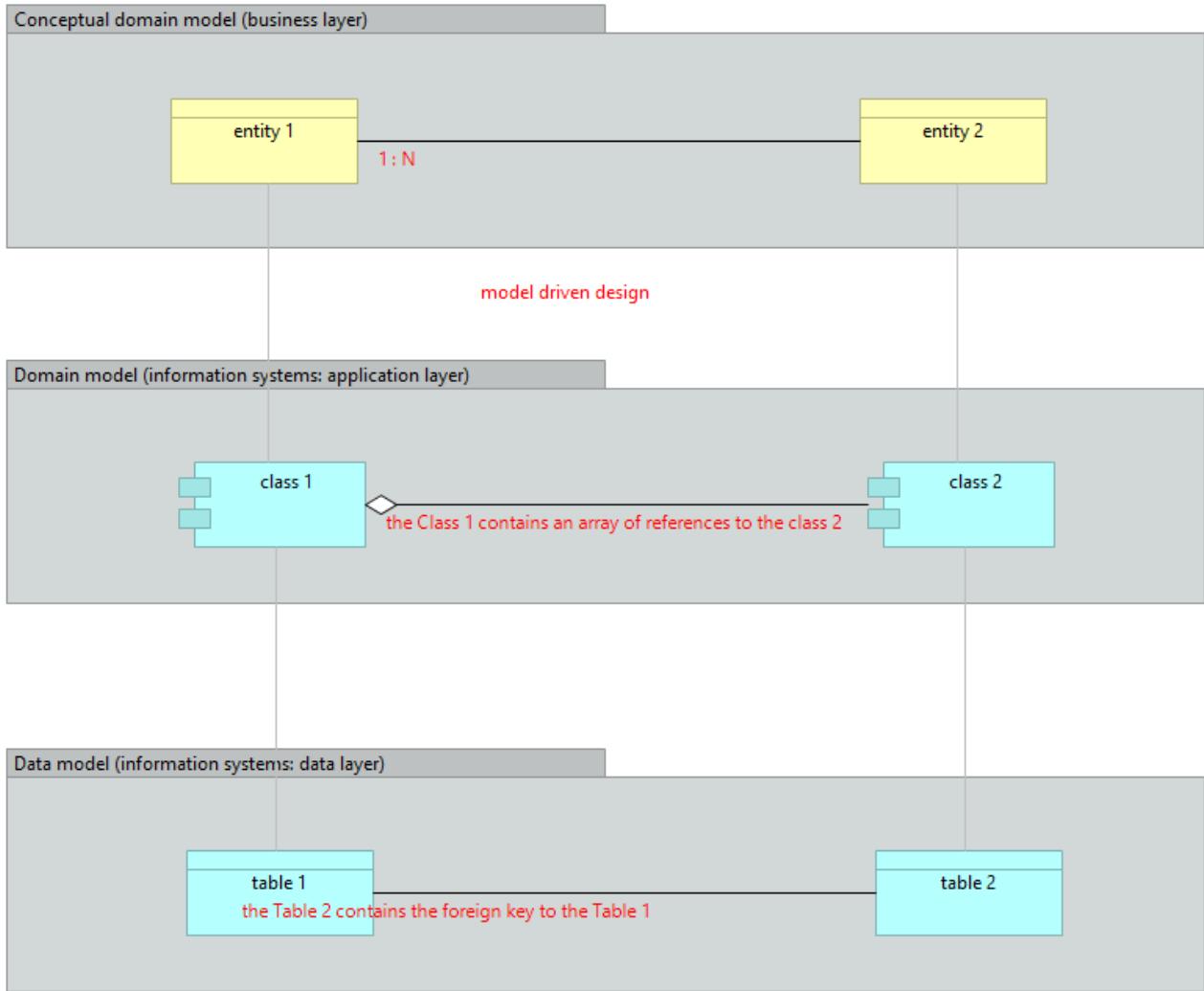
MODEL-DRIVEN DESIGN

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.

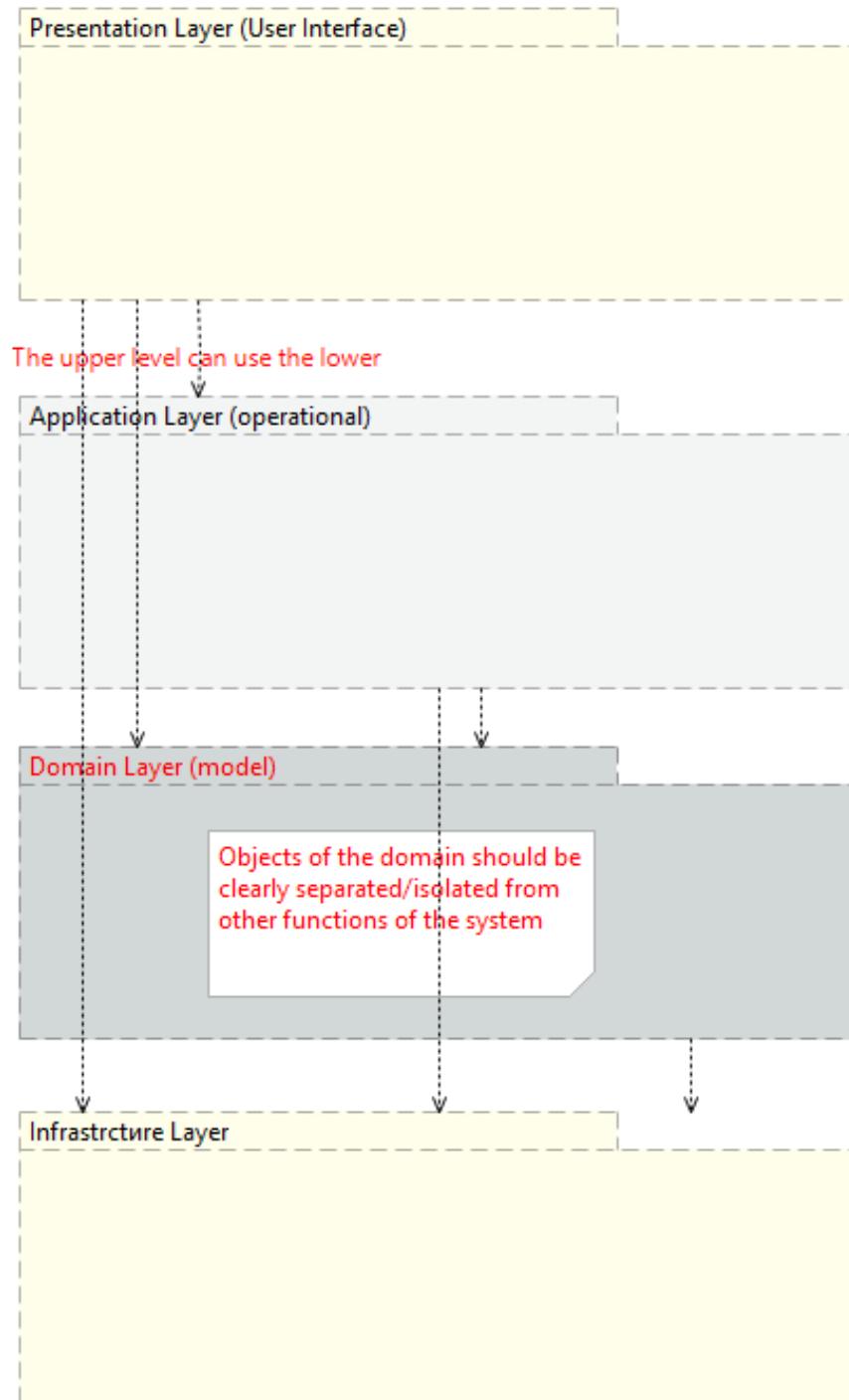
Domain model (information systems: application layer)

class 1

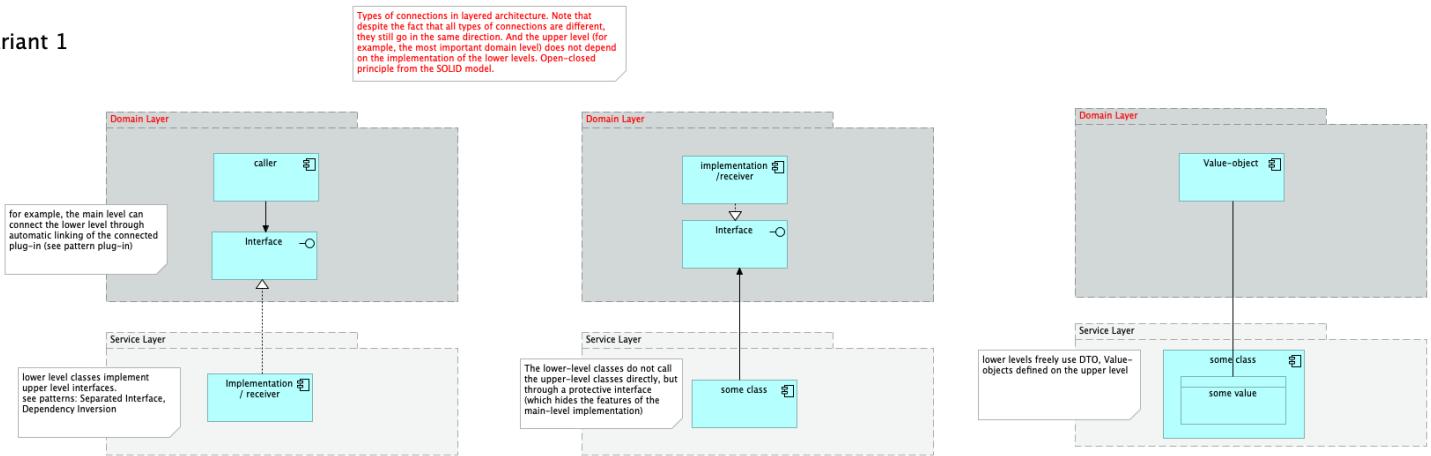
class 2

LAYERED ARCHITECTURE (ASYMMETRIC)

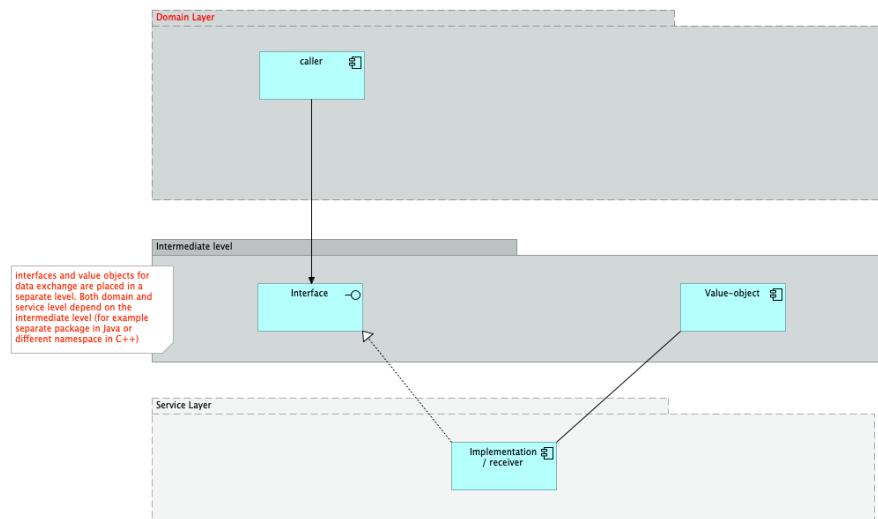
layered architecture
Only domain layer is required



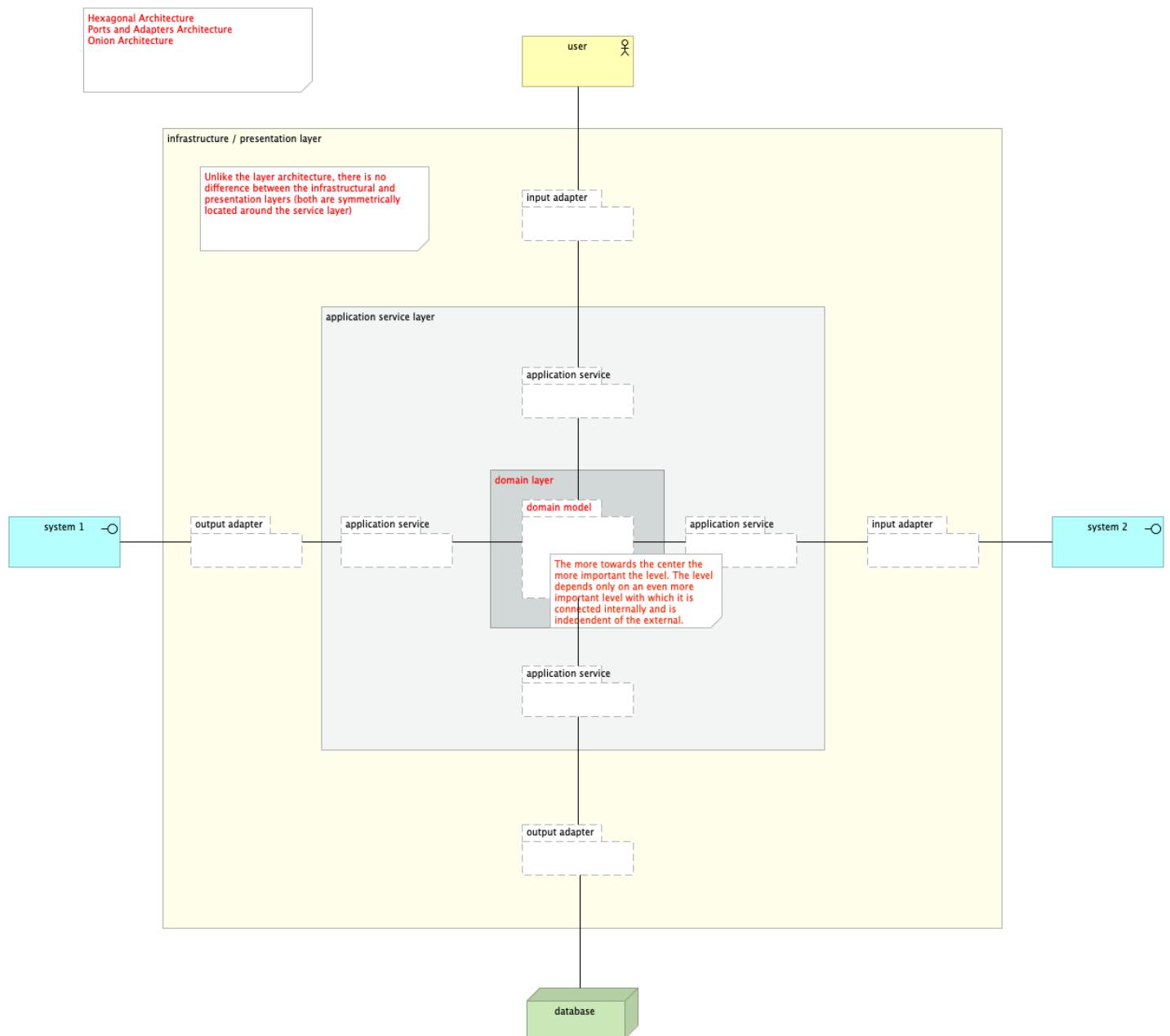
variant 1



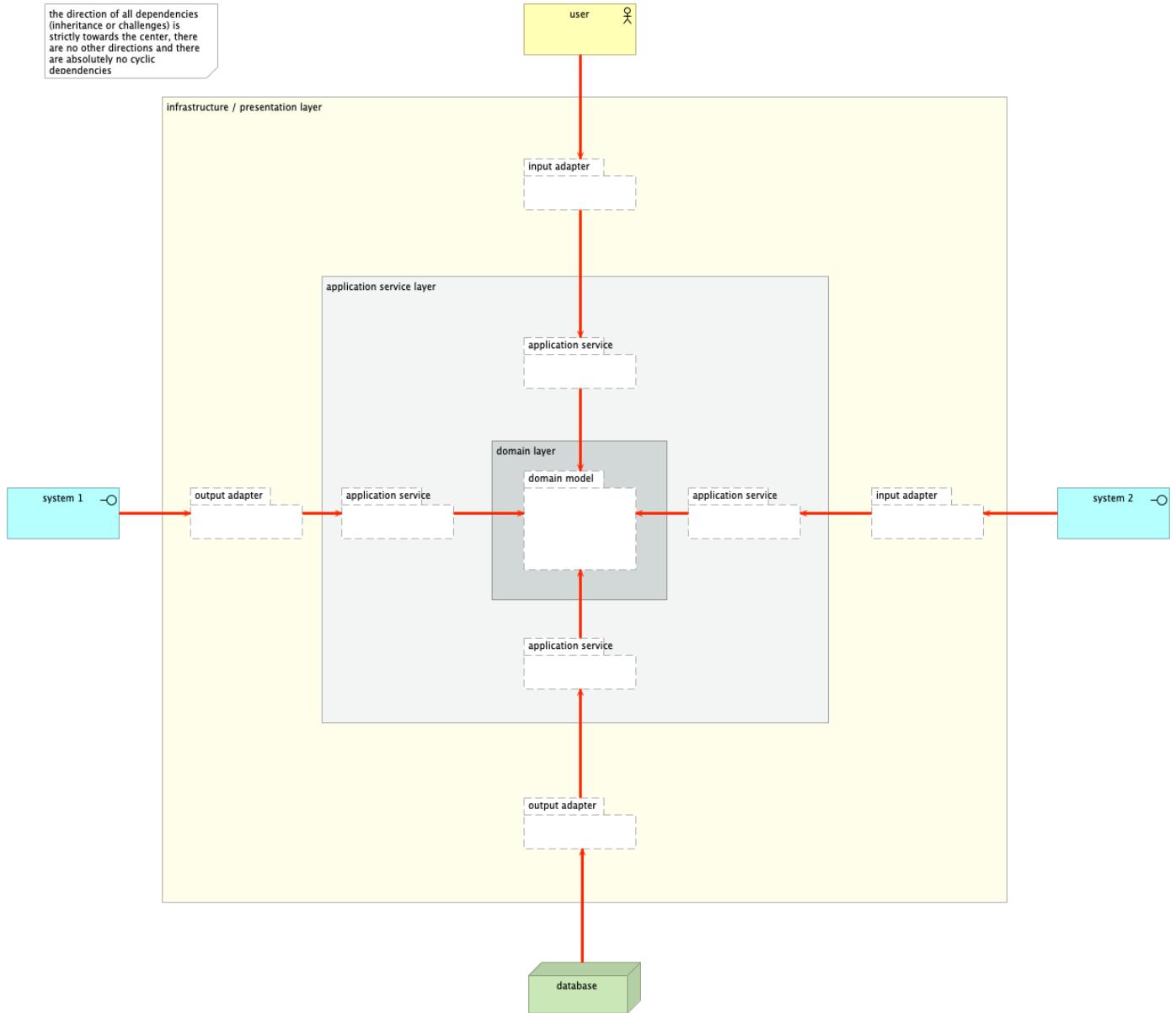
variant 2

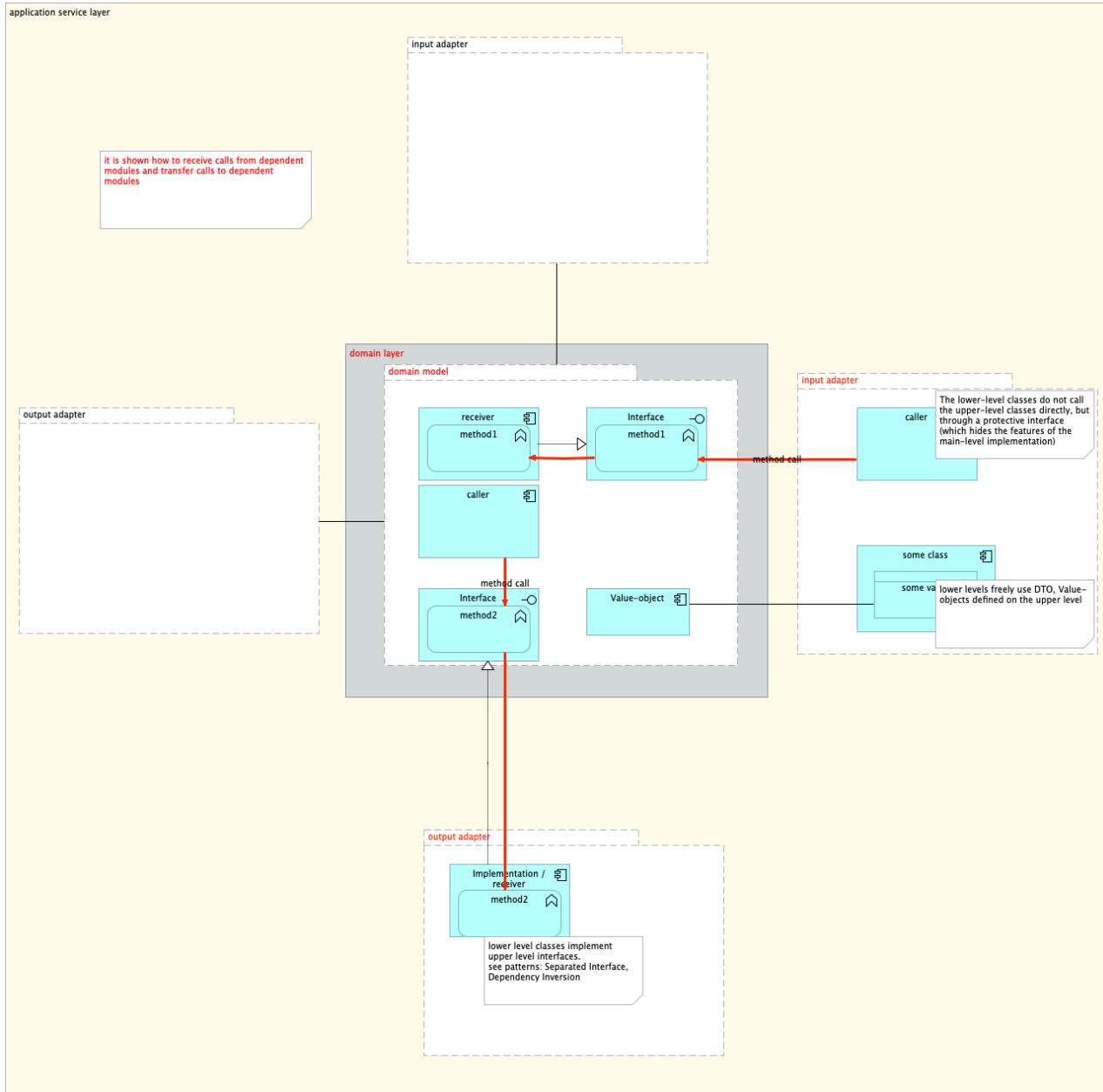


HEXAGONAL ARCHITECTURE (SYMMETRIC)

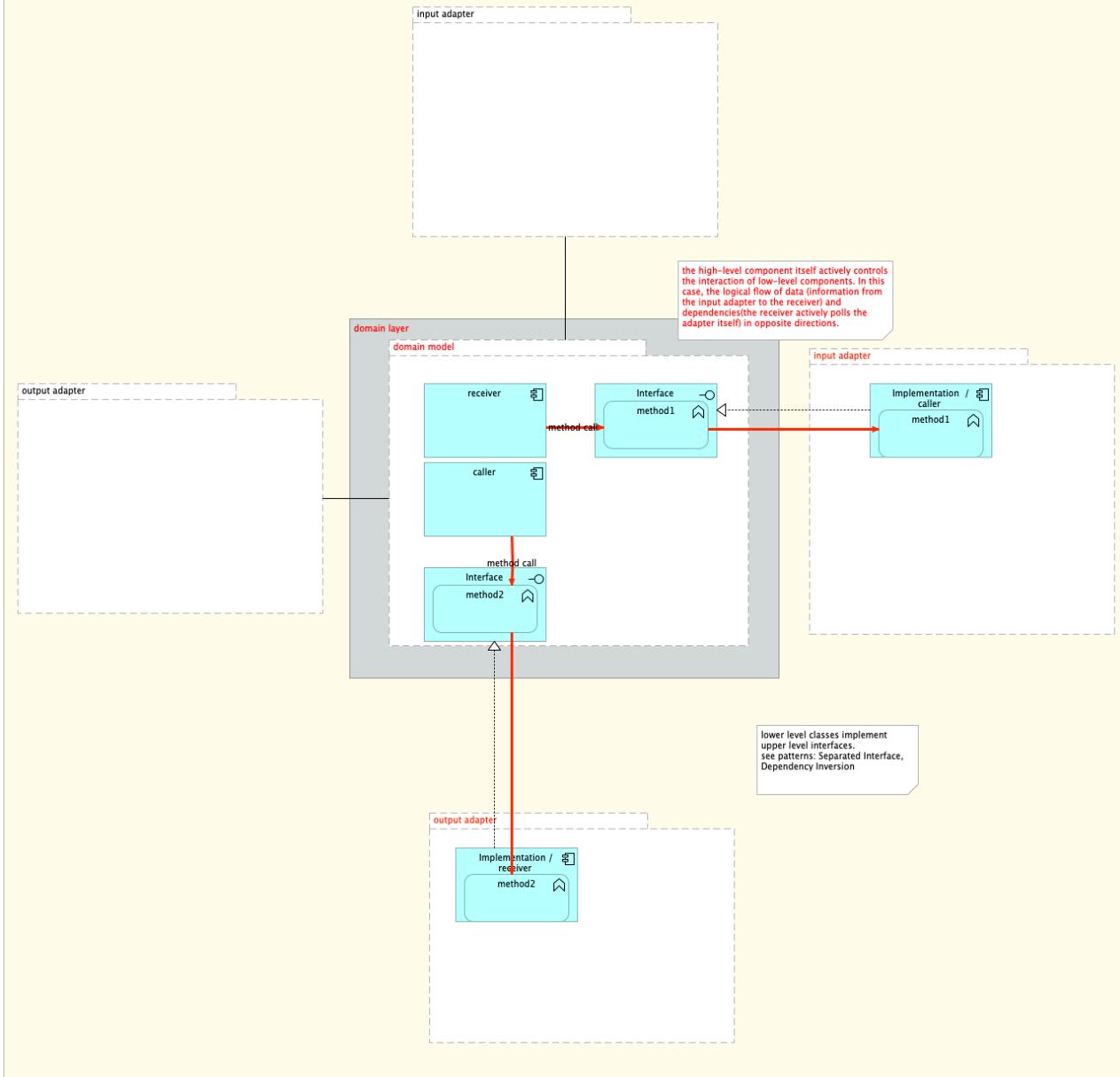


the direction of all dependencies
(inheritance or challenges) is
strictly towards the center, there
are no other directions and there
are absolutely no cyclic
dependencies

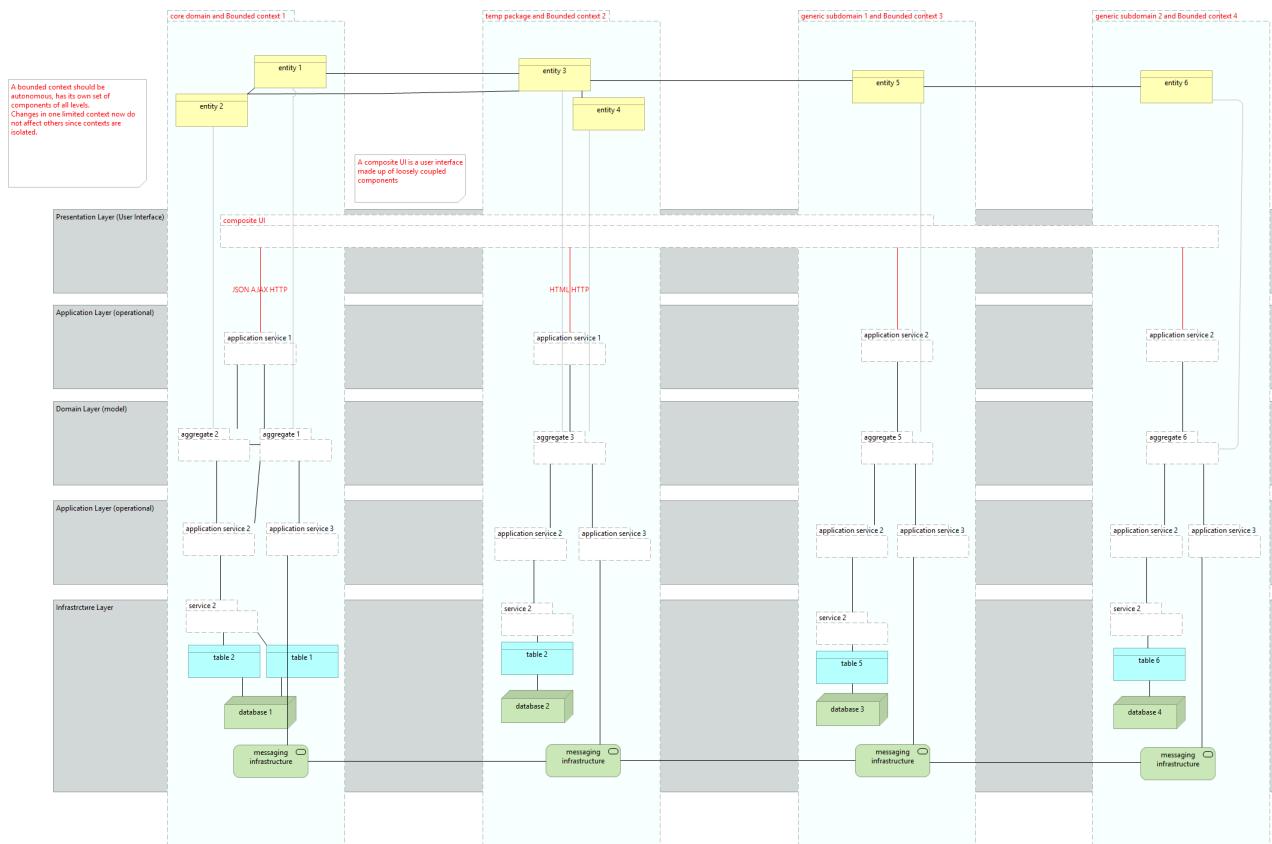




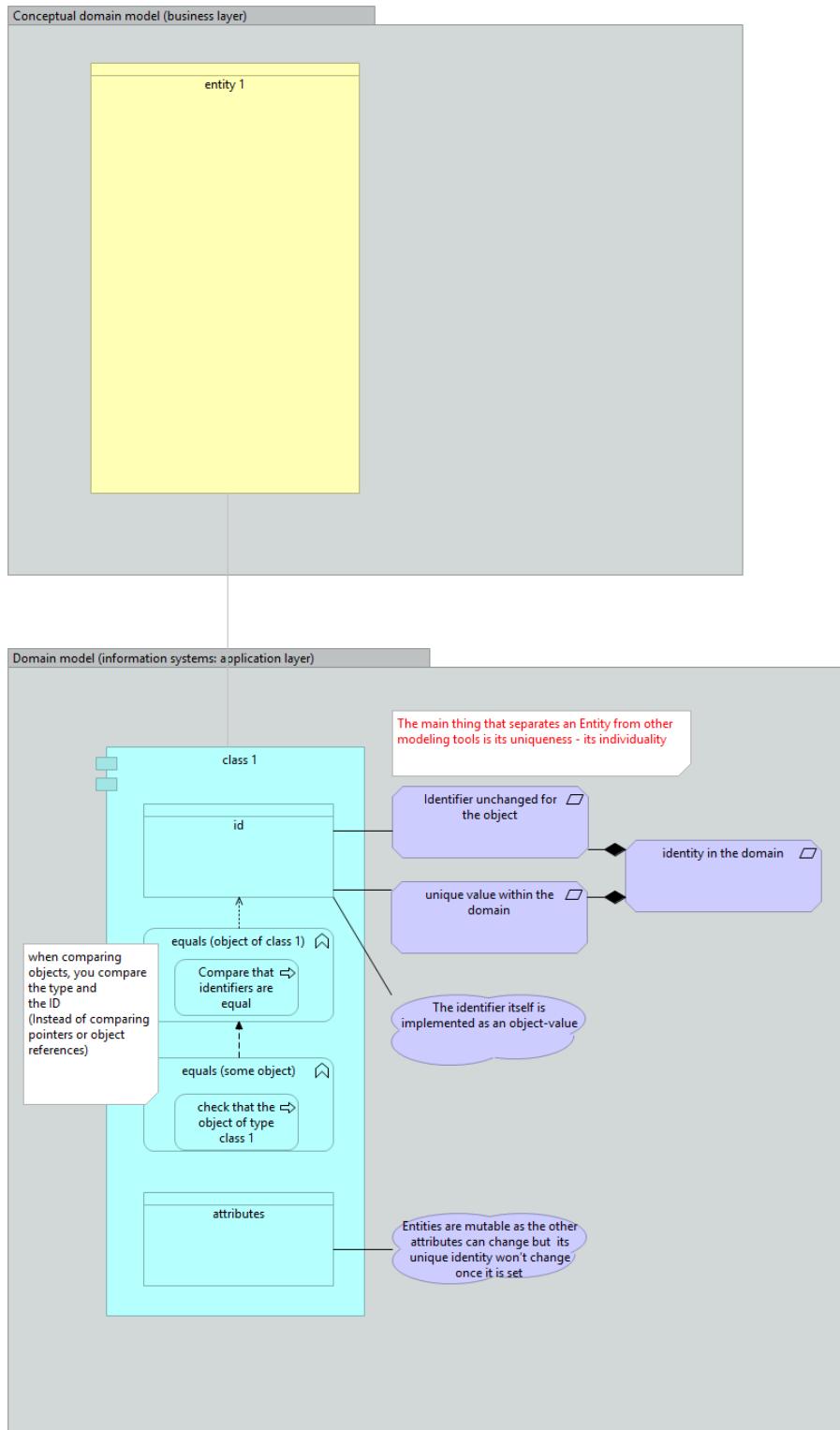
application service layer

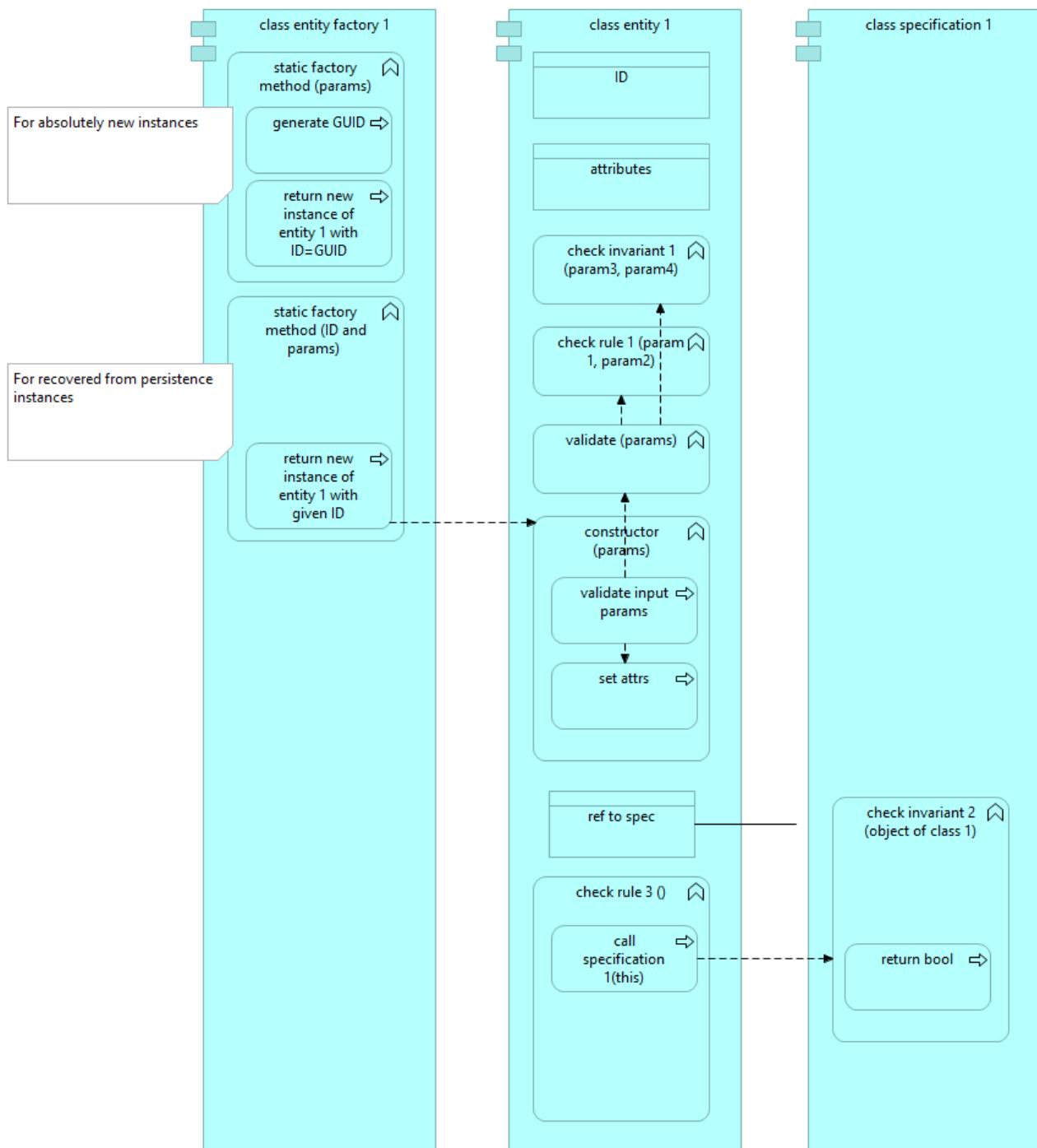


COMPOSITE UI

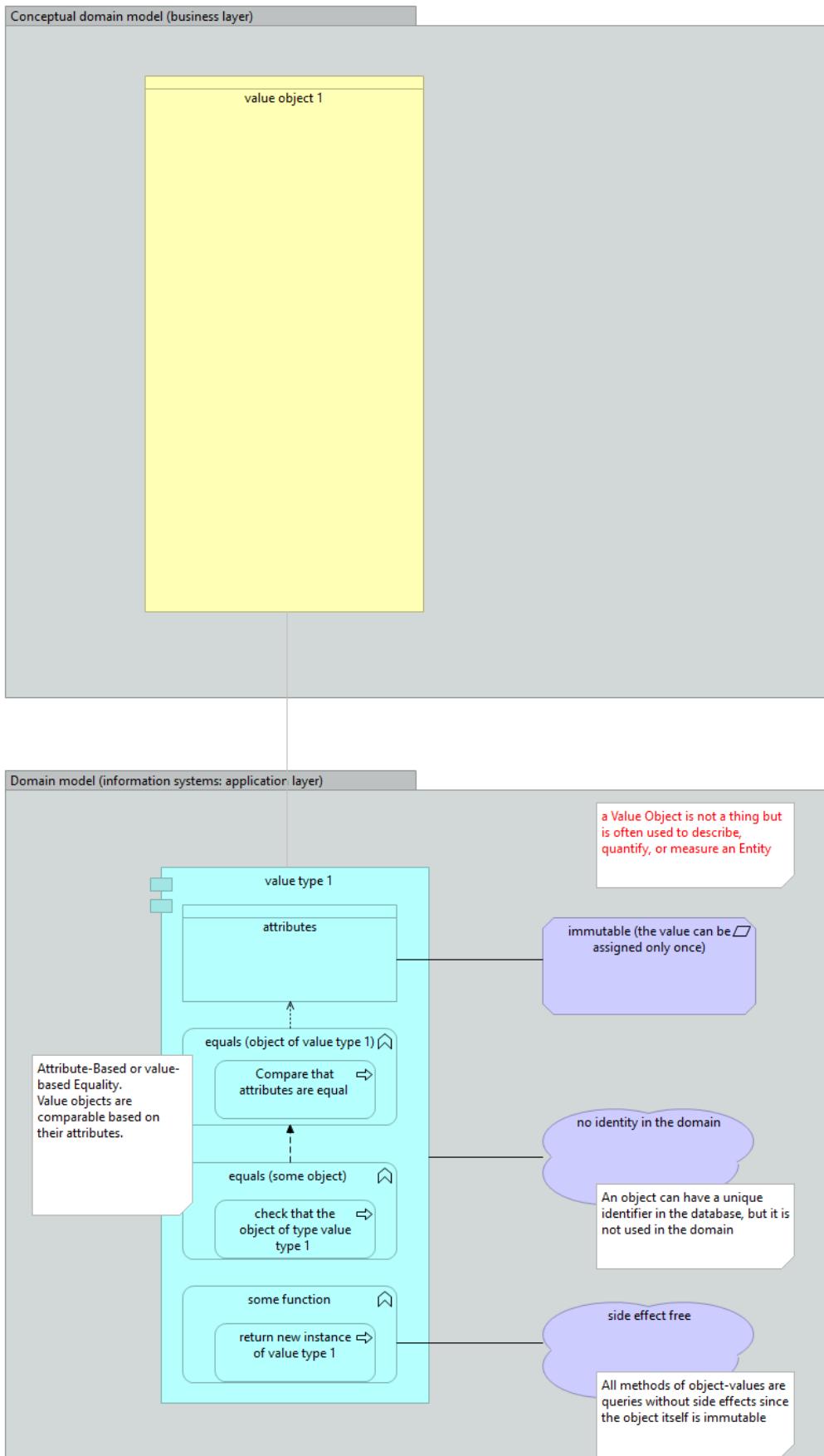


ENTITIES

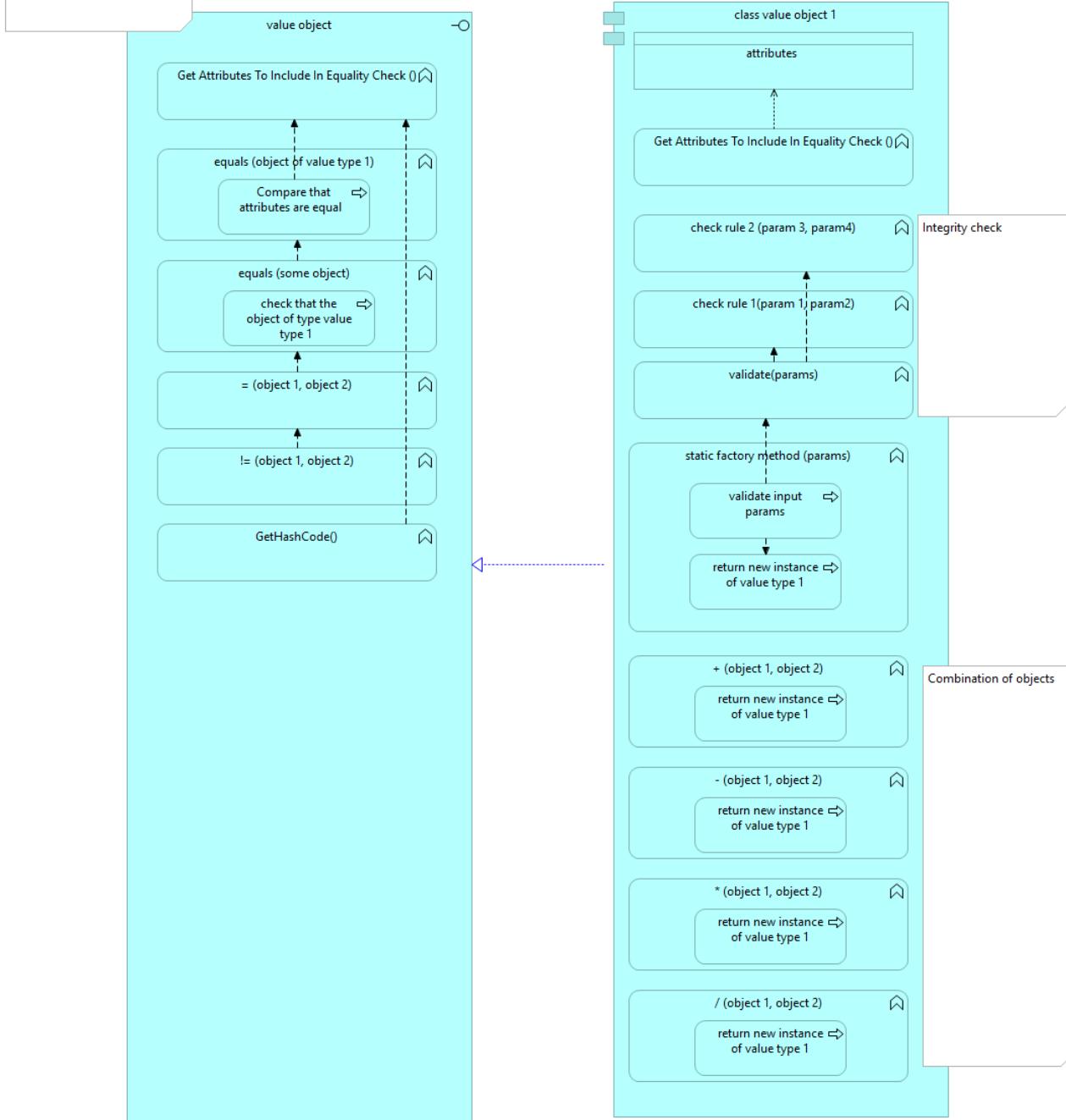




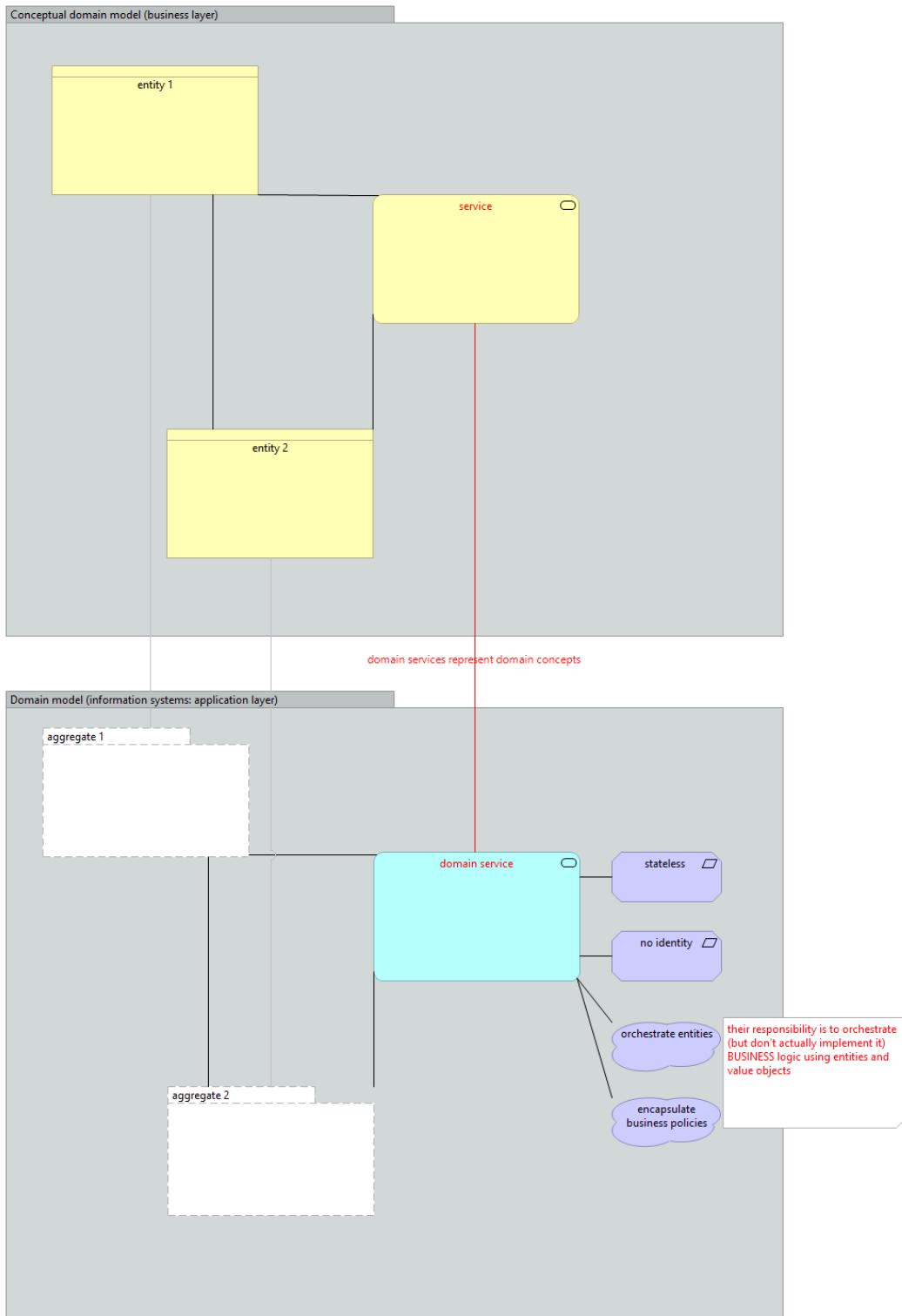
VALUE-OBJECTS



See example: Fowler's Money

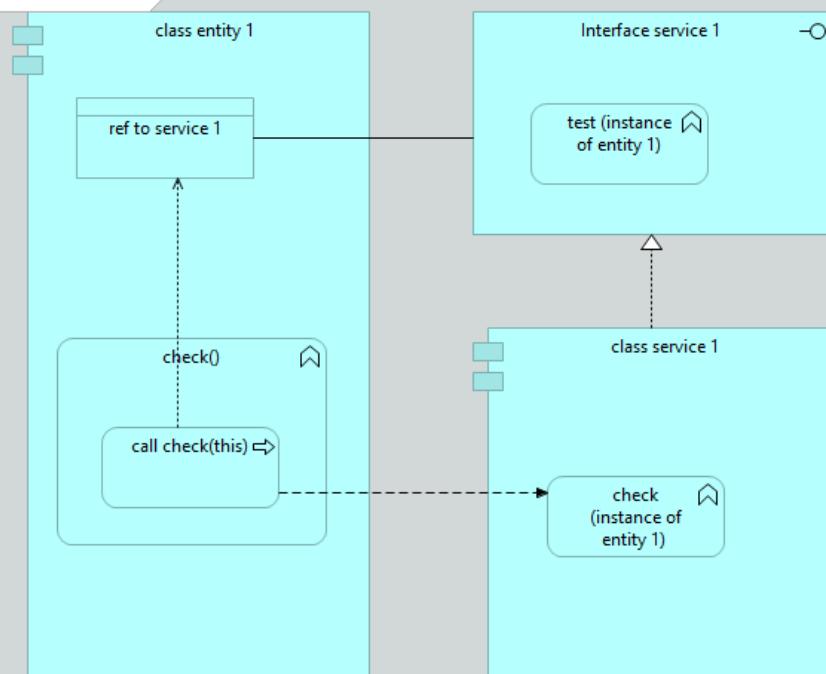


DOMAIN SERVICES

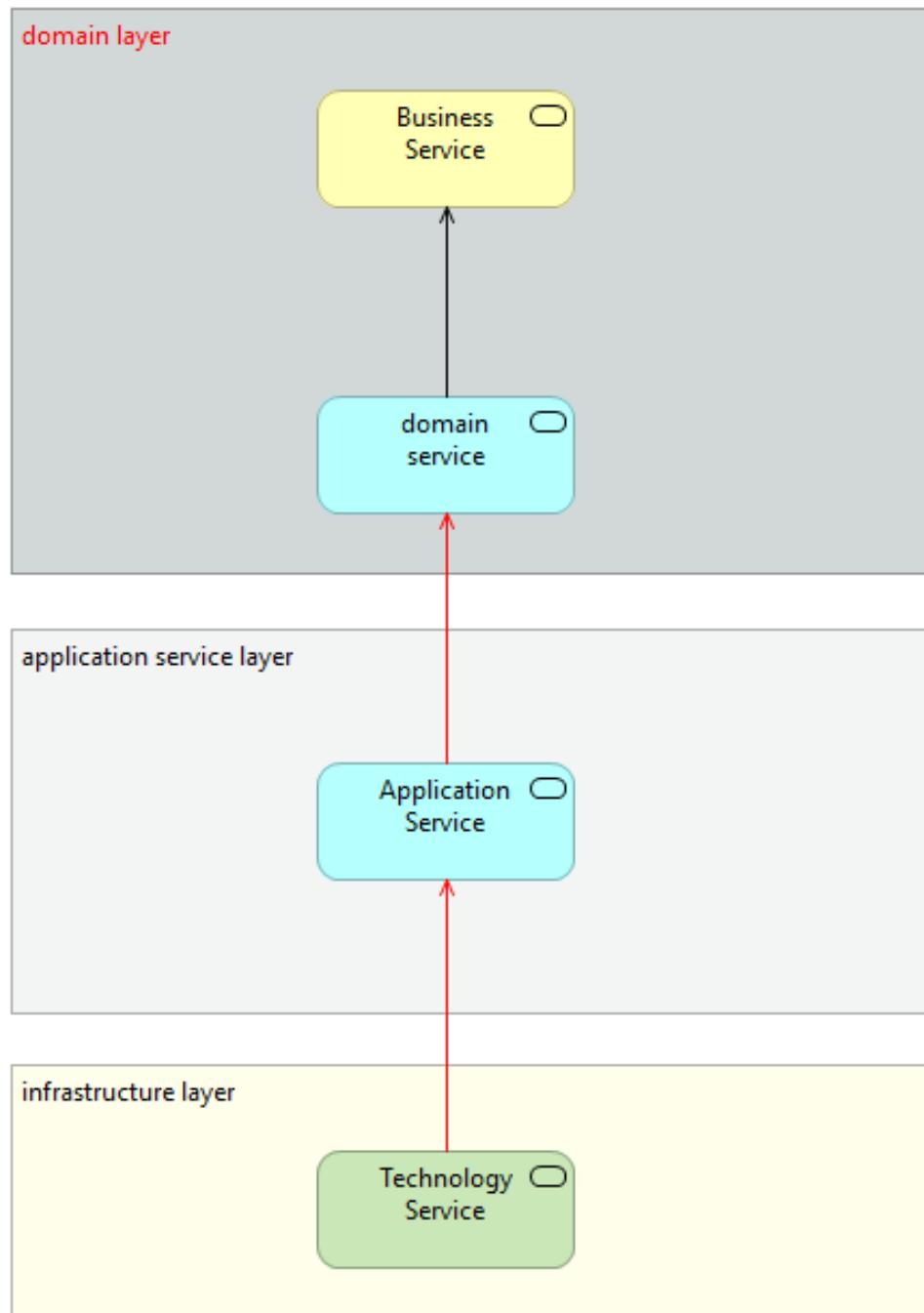


Domain model (information systems: application layer)

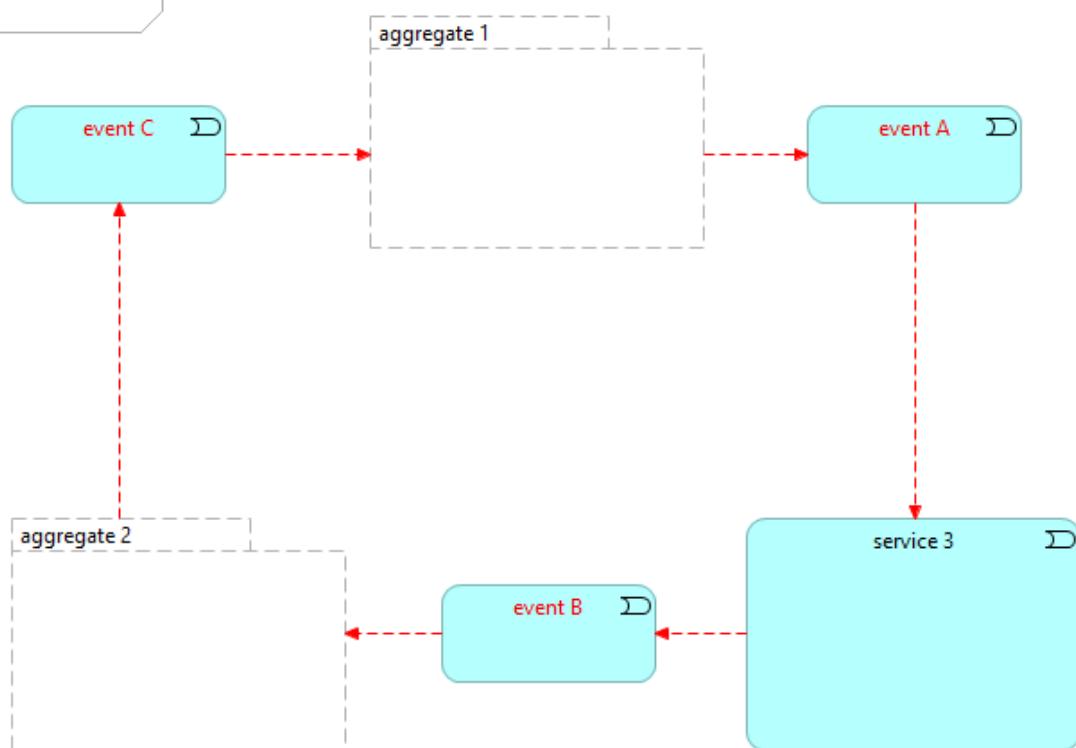
Example of communicating services with aggregates via direct service call

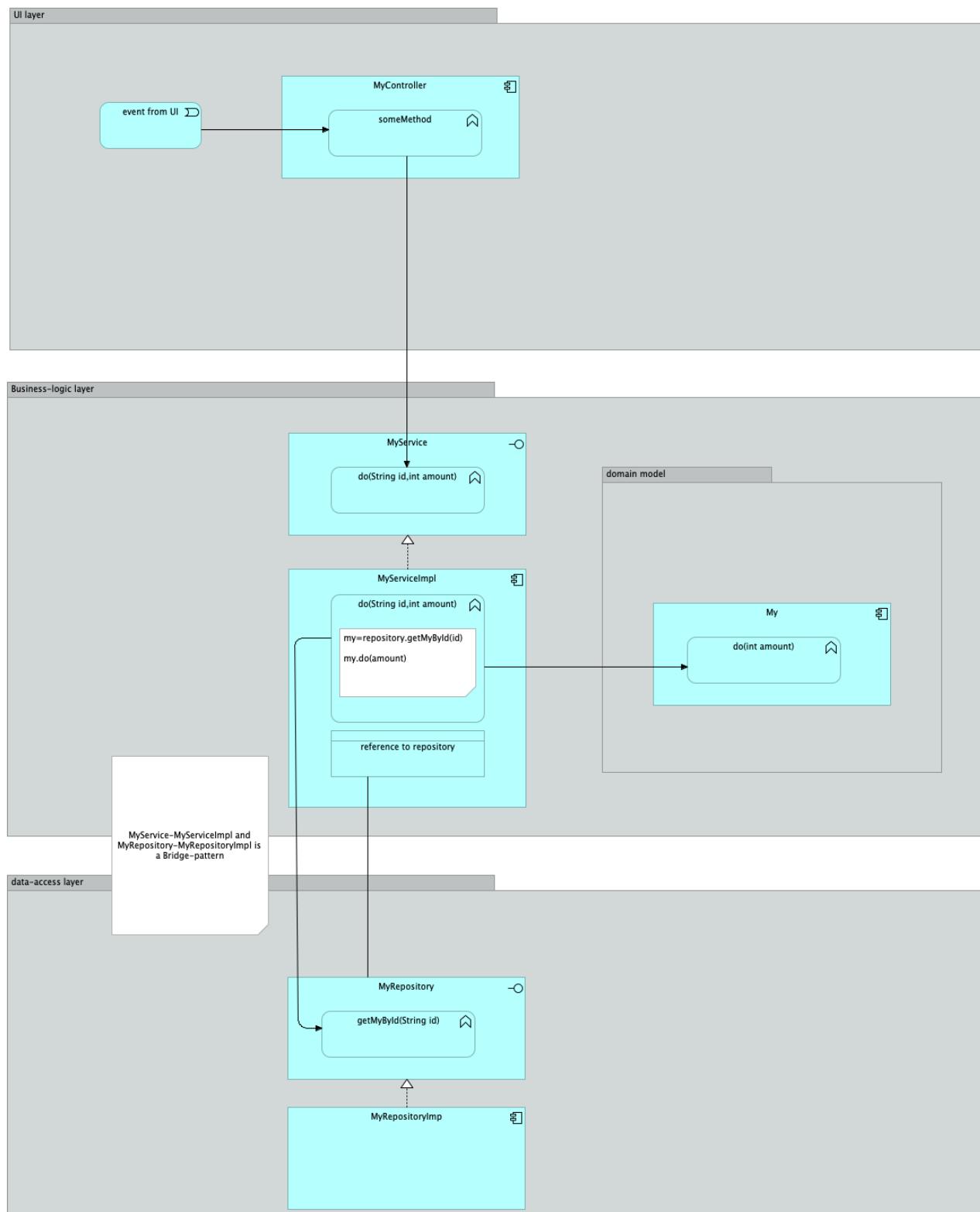


Services support each other

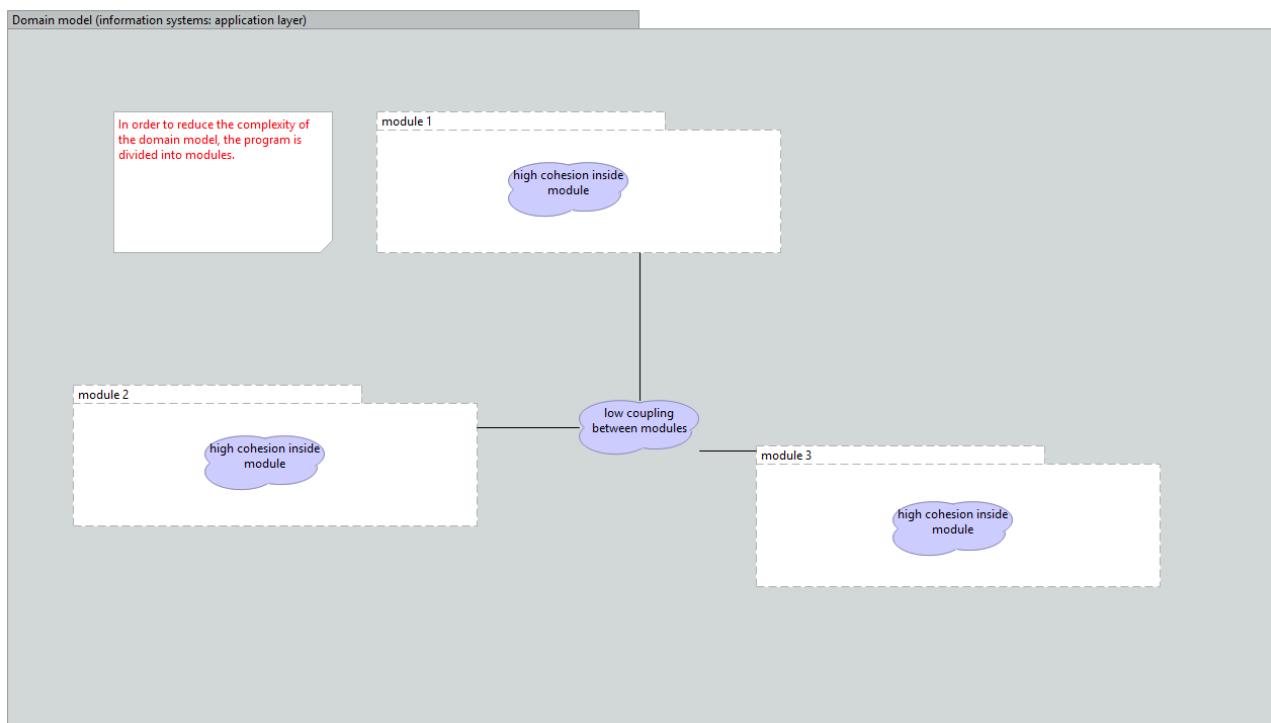


Example of communicating services with aggregates through events

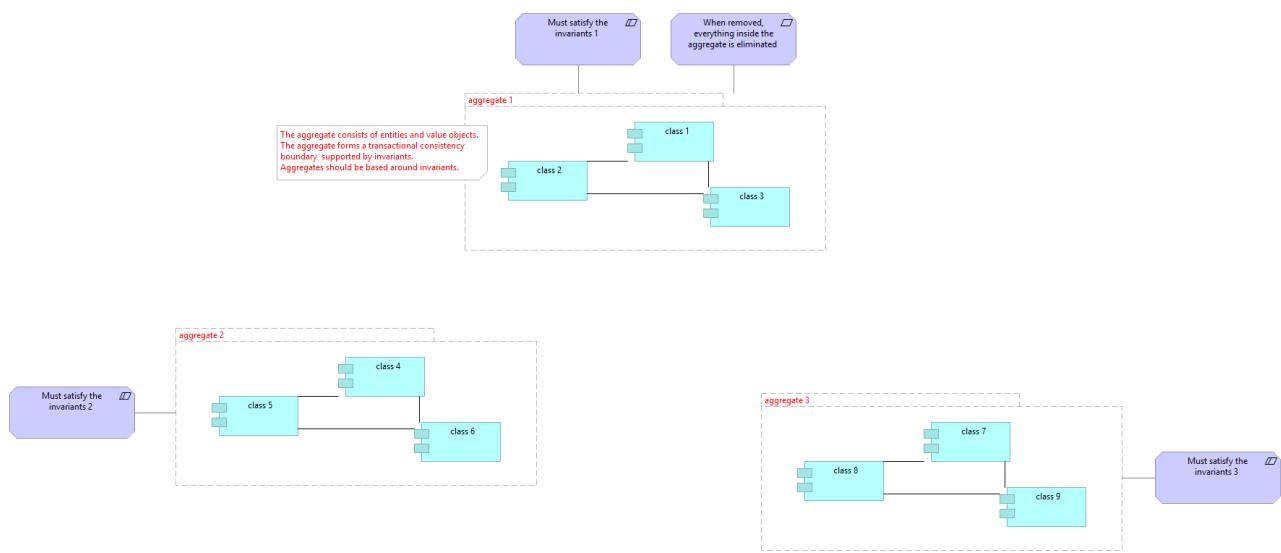




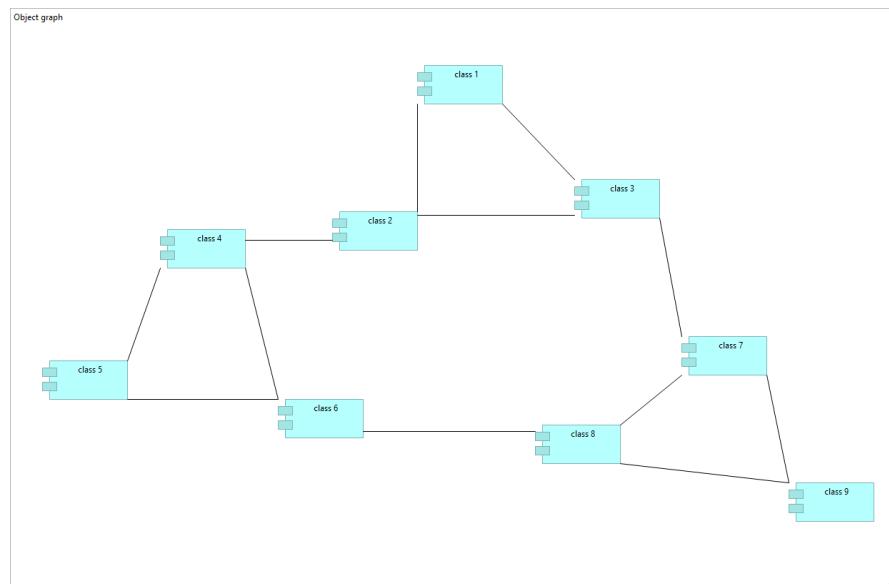
MODULES



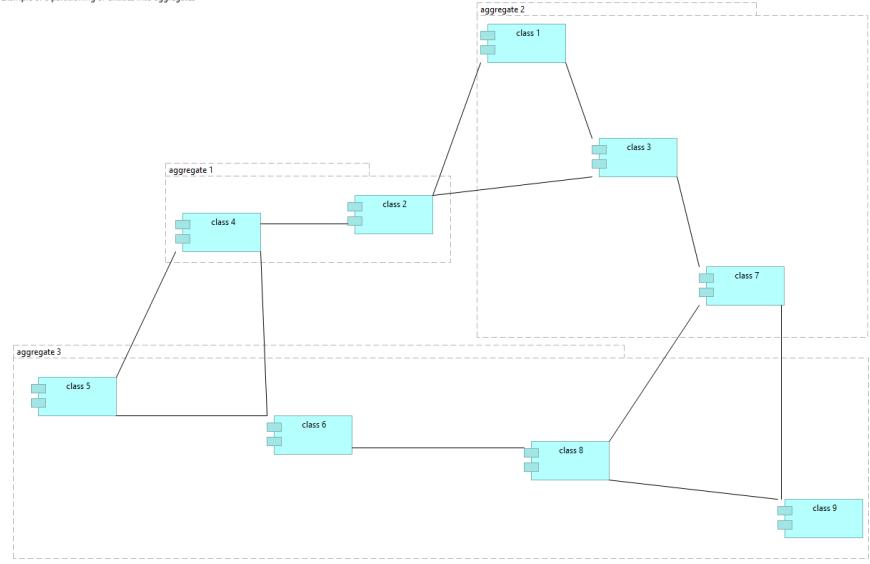
AGGREGATES



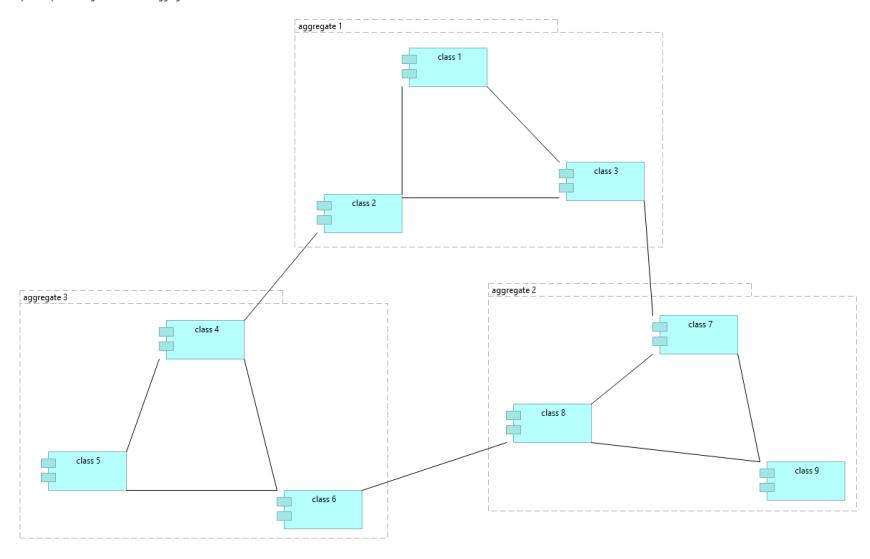
In the course of modeling, think about different ways of splitting the model into aggregates



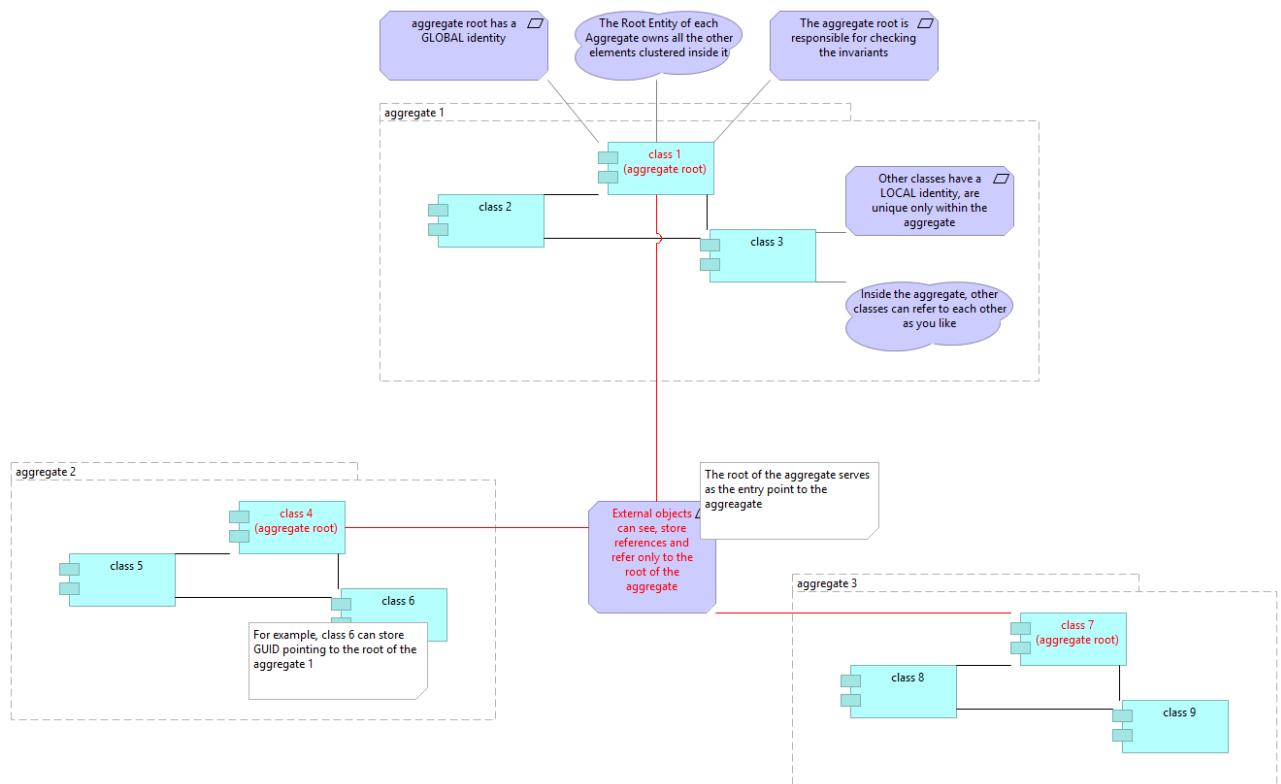
Example of a partitioning of entities into aggregates



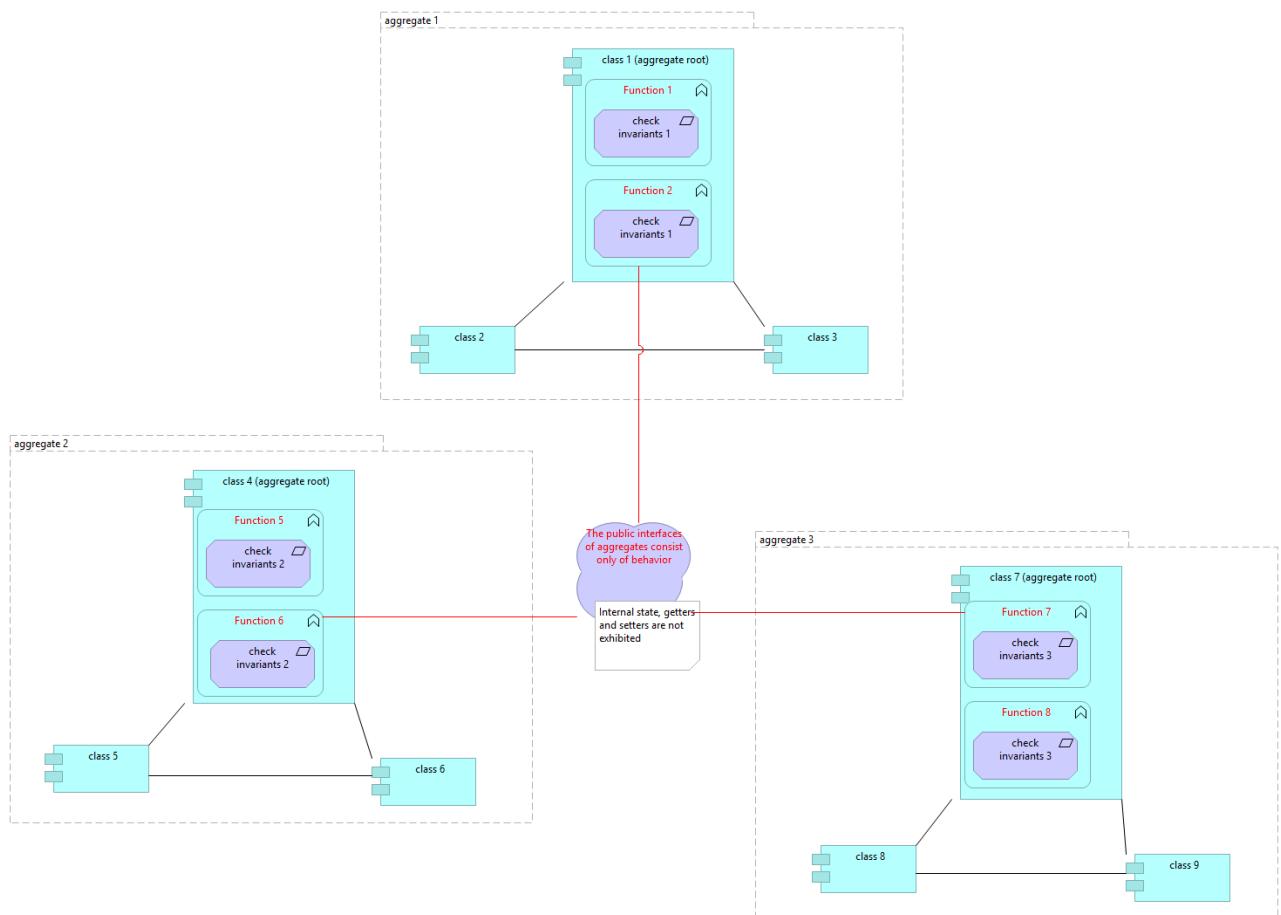
Example of a partitioning of entities into aggregates



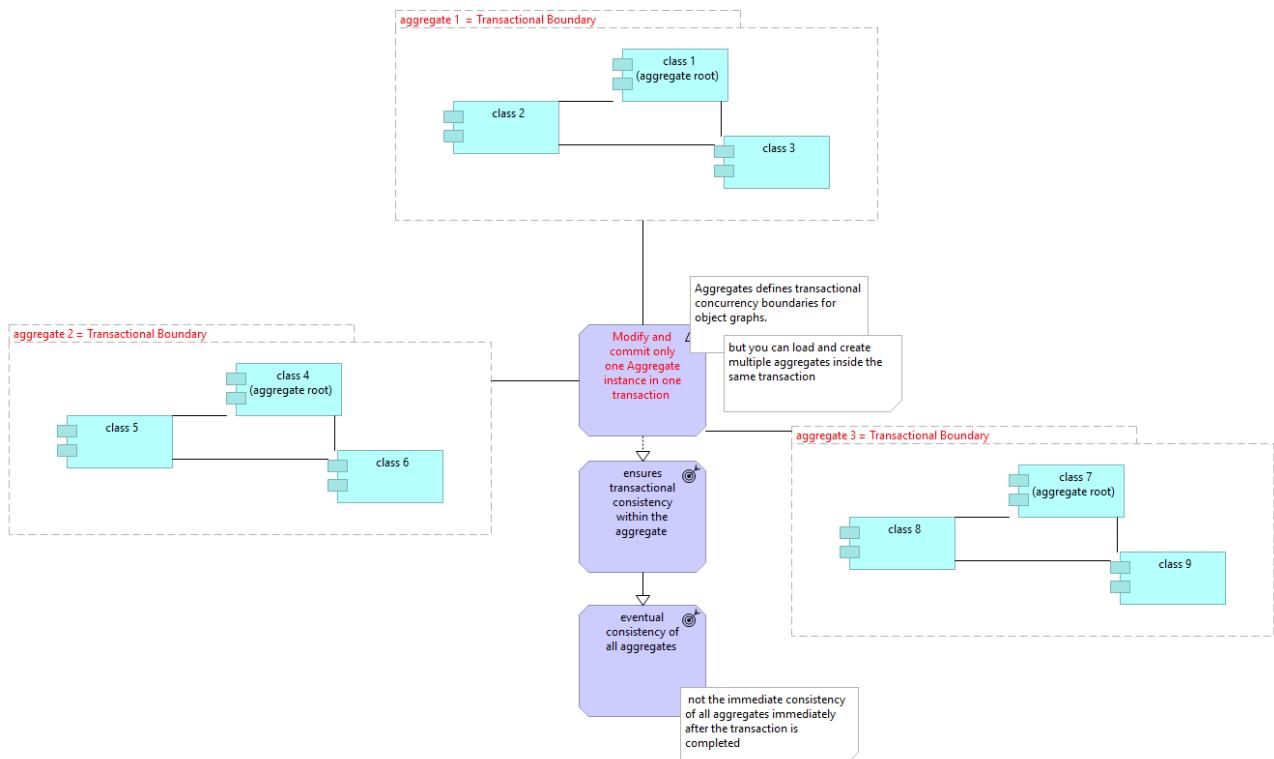
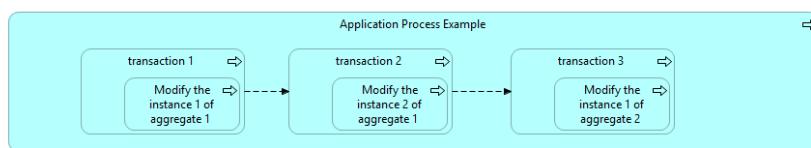
AGGREGATE ROOT



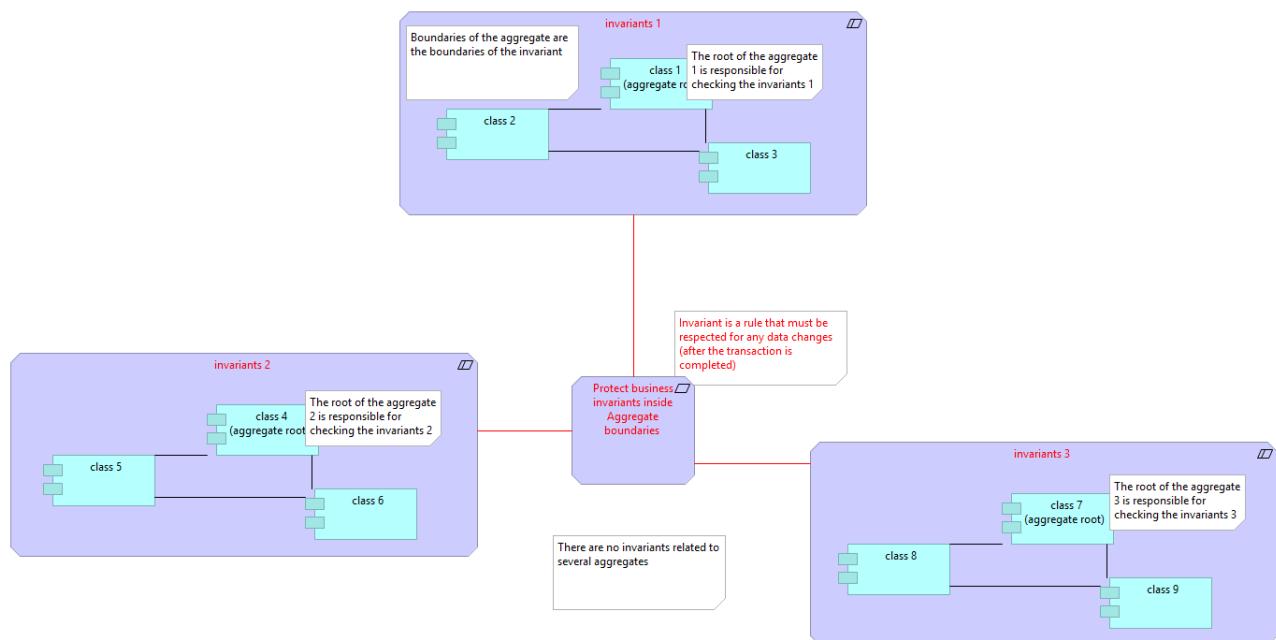
BEHAVIOR-FOCUSED AGGREGATE ROOT



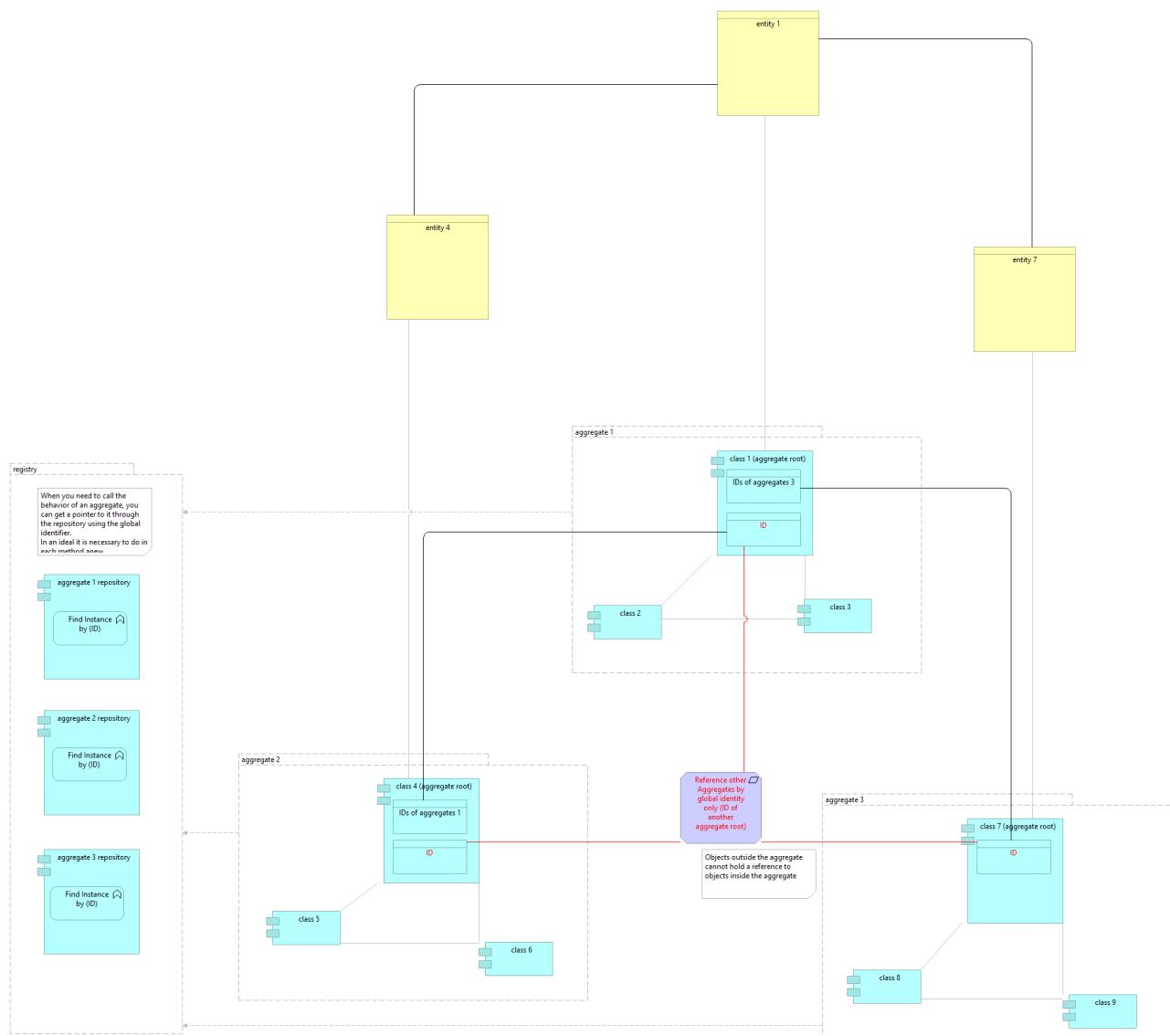
MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION



PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES



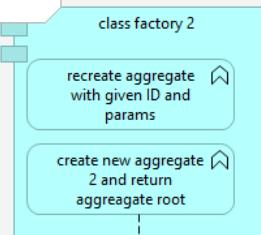
REFERENCE OTHER AGGREGATES BY IDENTITY ONLY



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

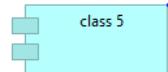


domain layer

Must satisfy the invariants 2 after creation

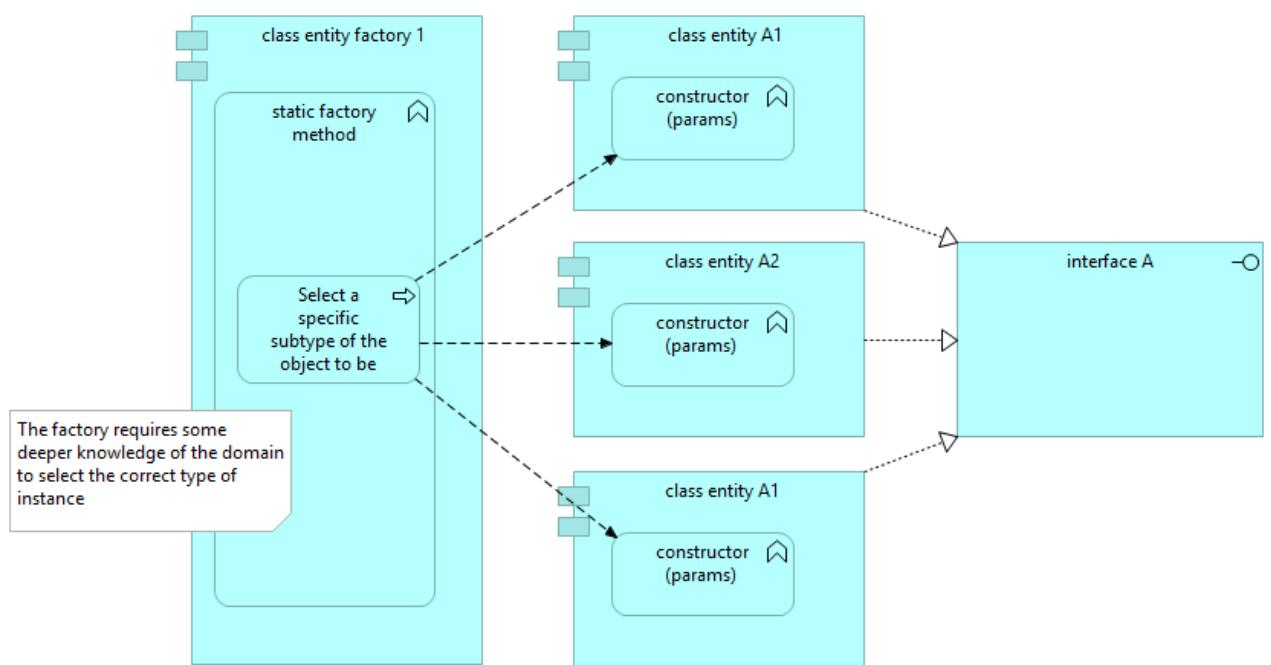
aggregate 2

The root of the aggregate itself creates all internal objects

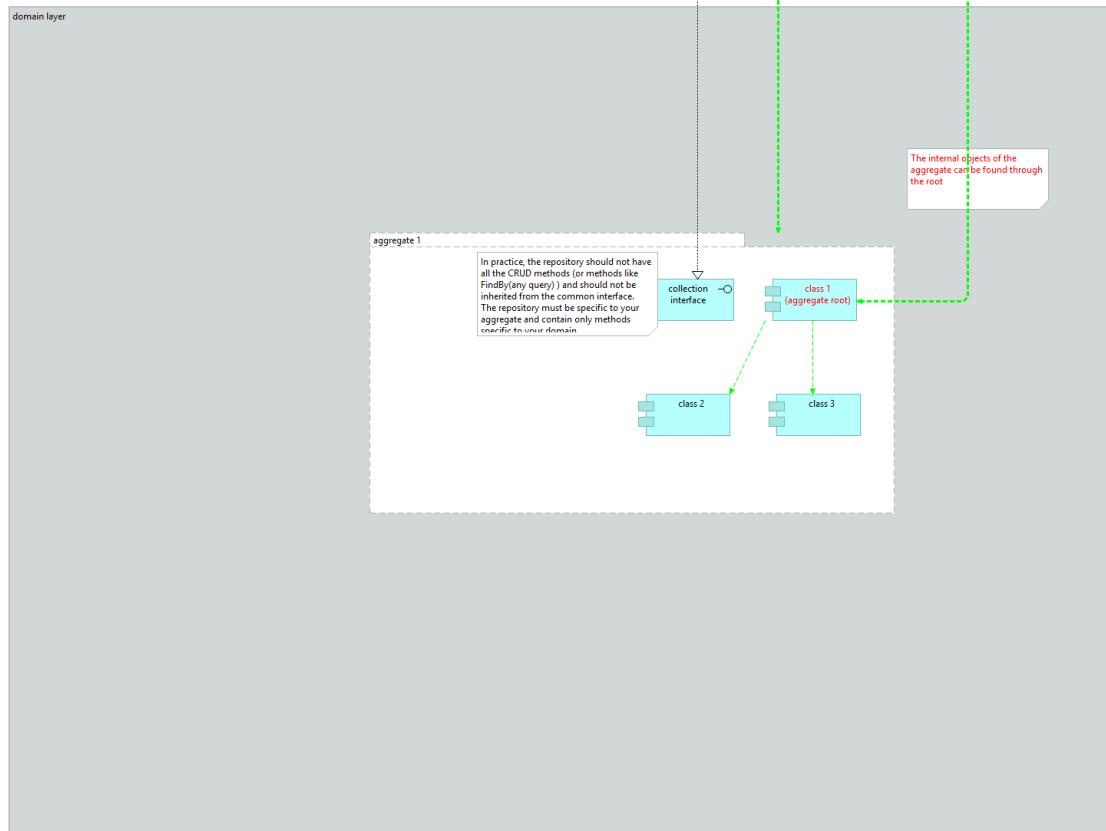
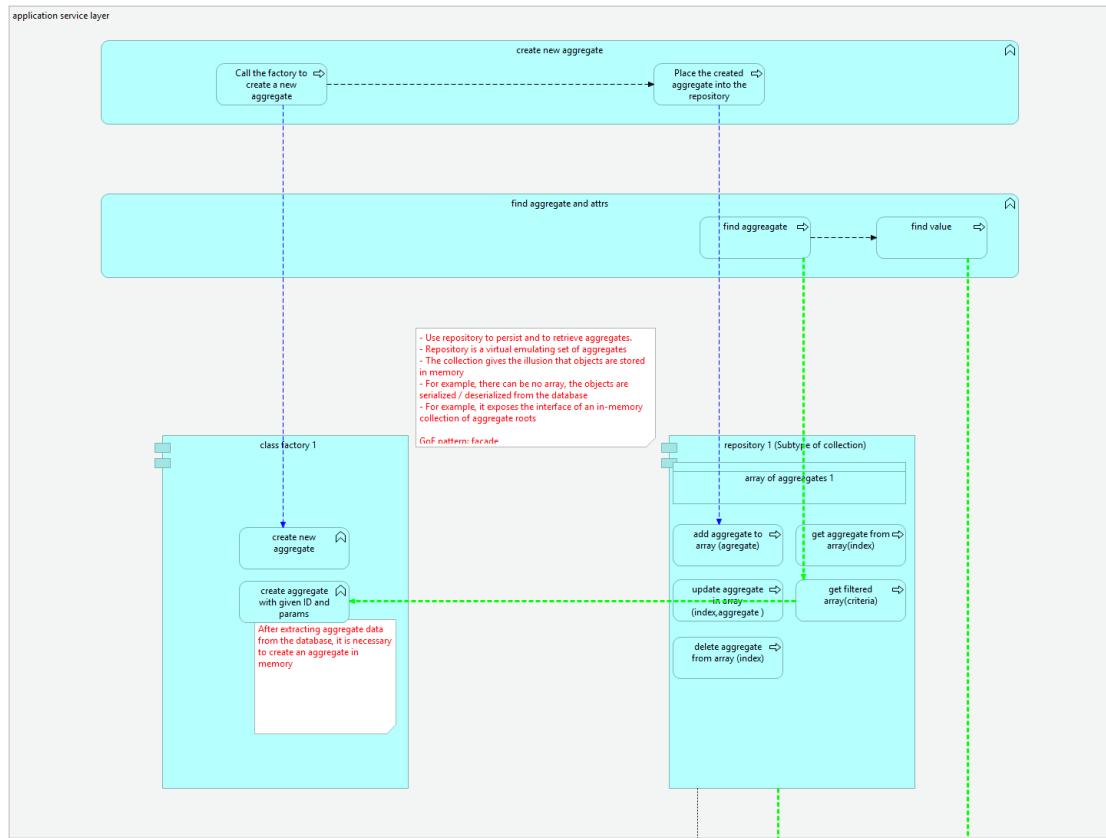


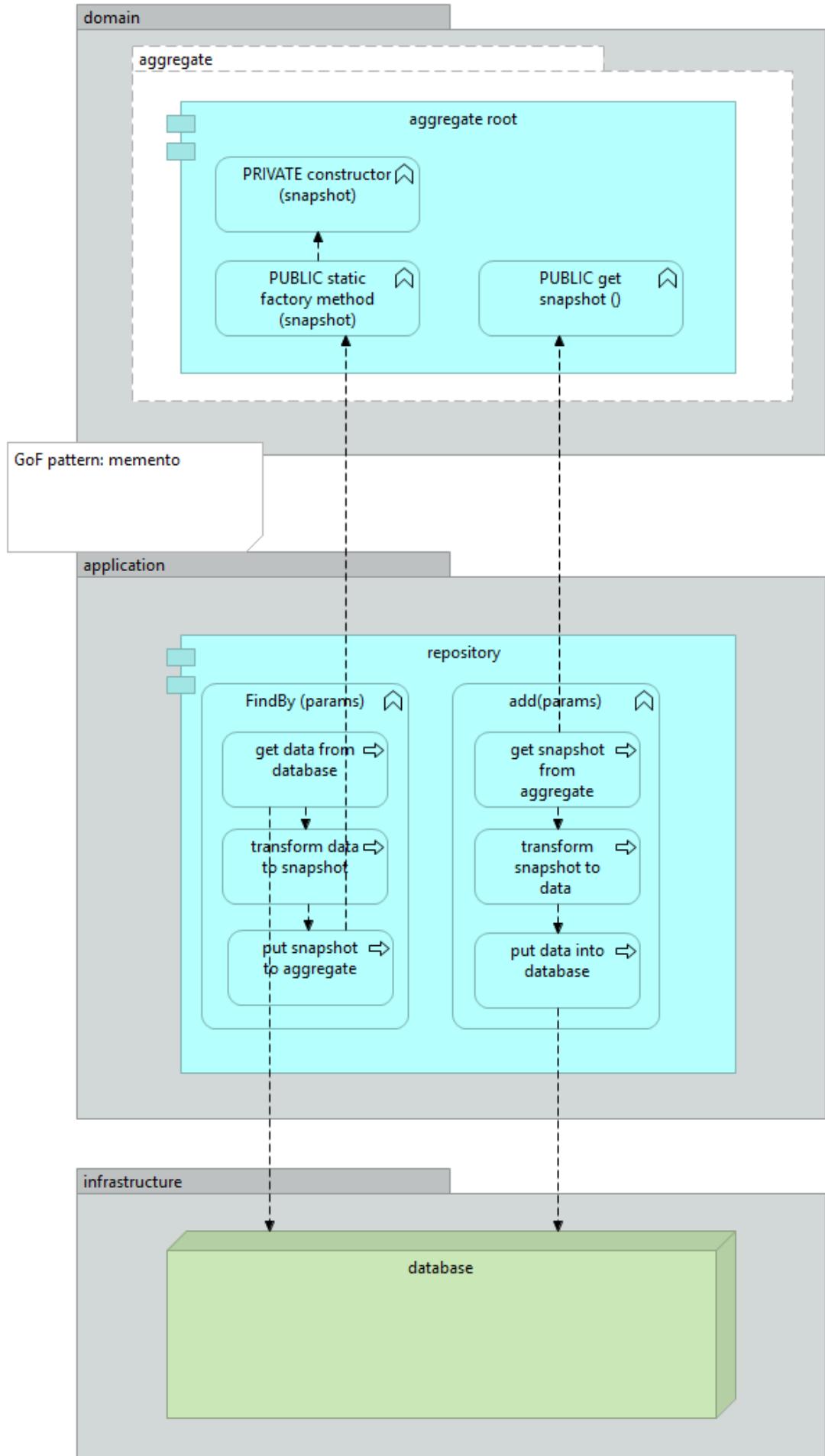
class 6

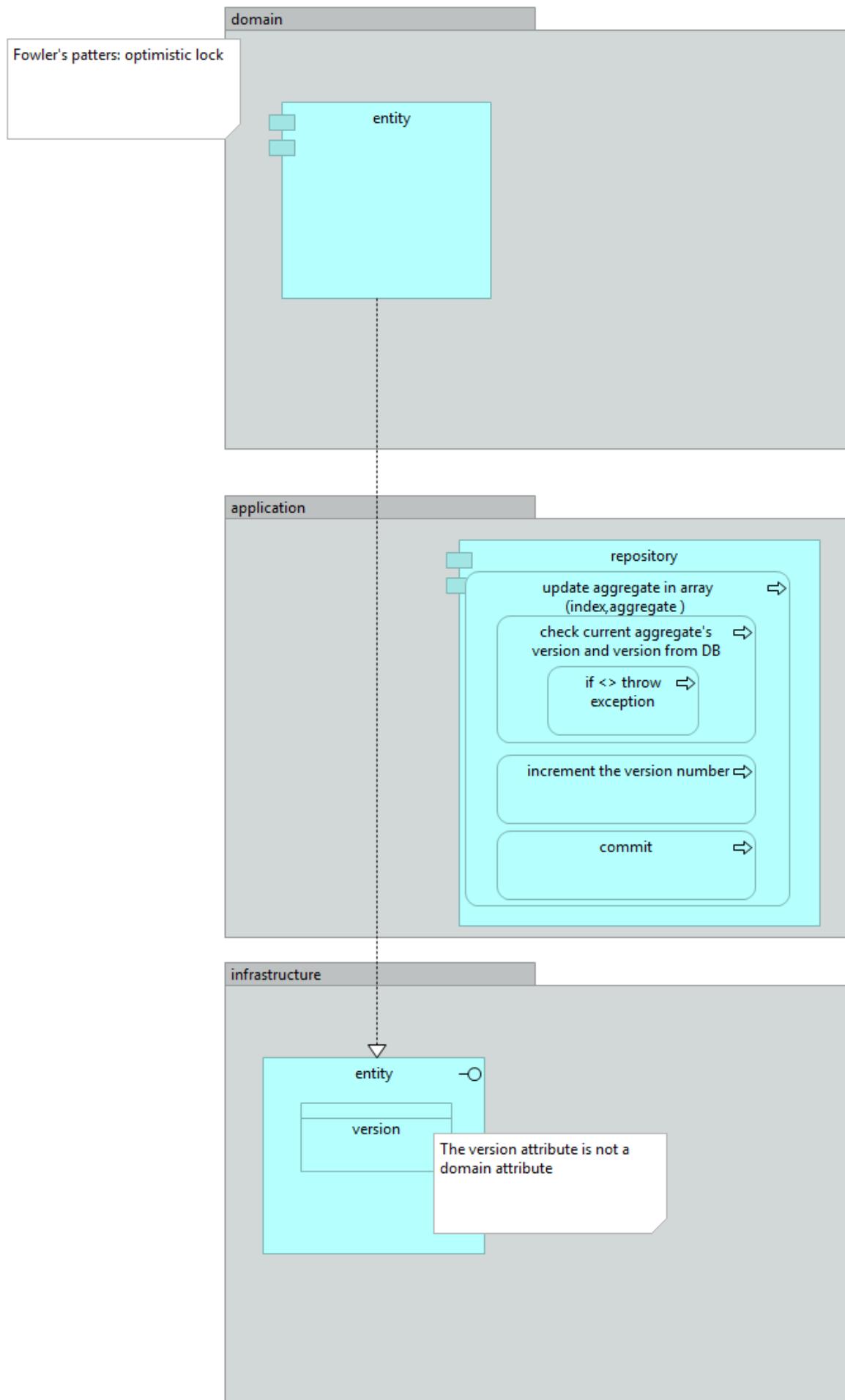


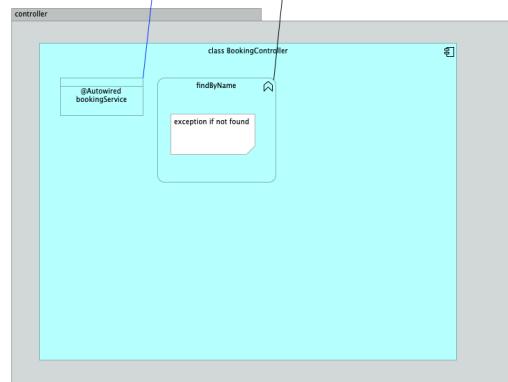
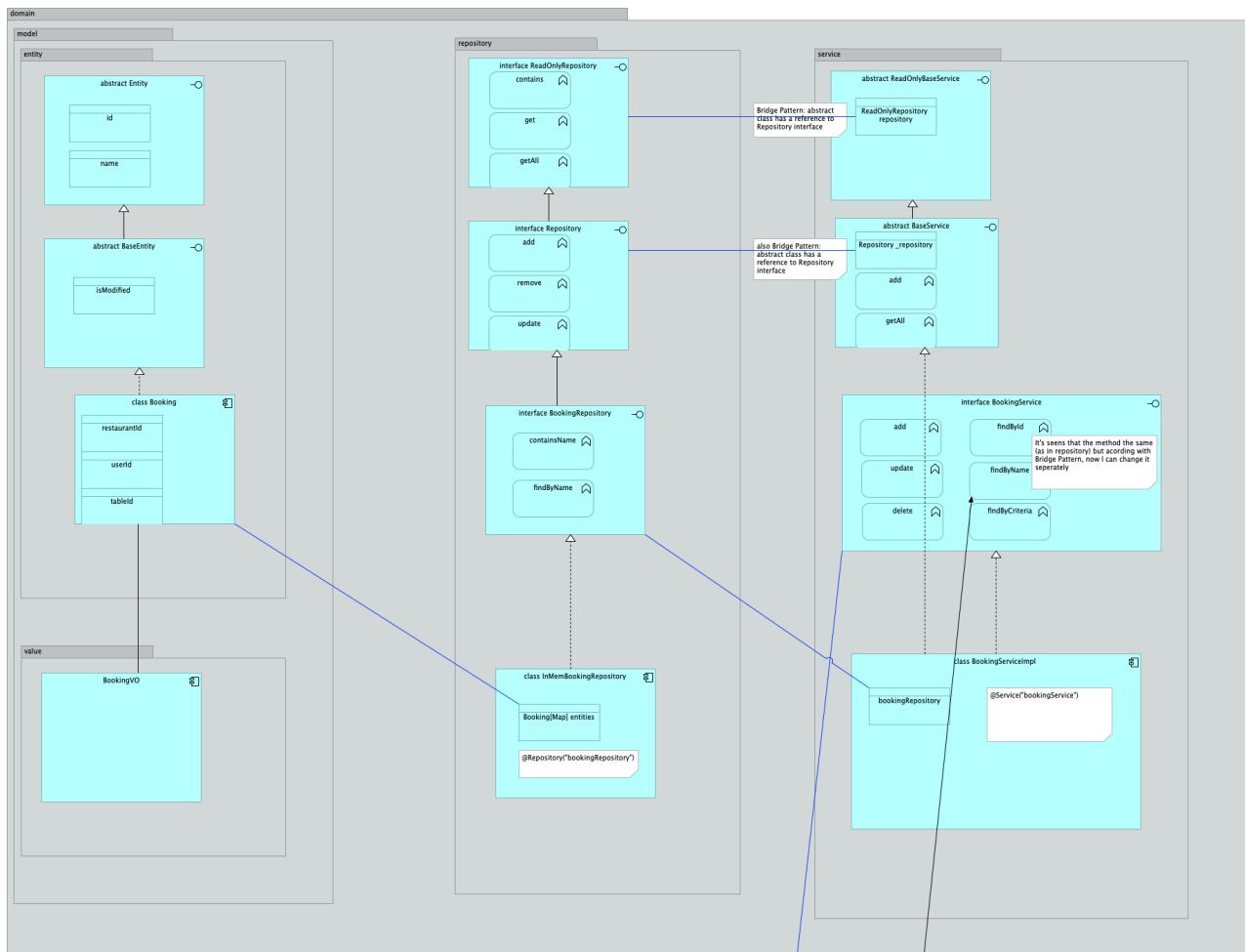


REPOSITORIES









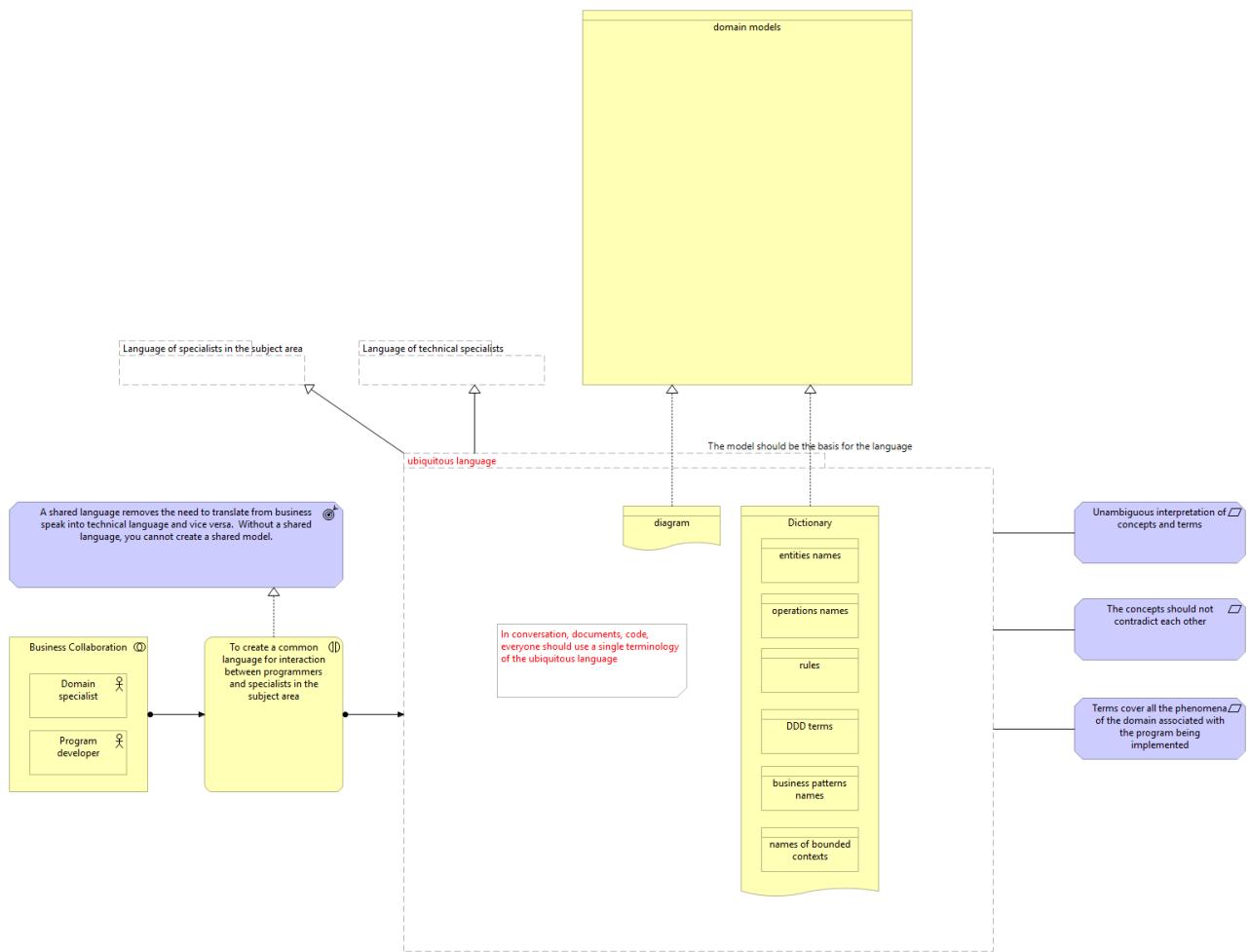
SUPPLE DESIGN

DOMAIN DRIVEN DESIGN

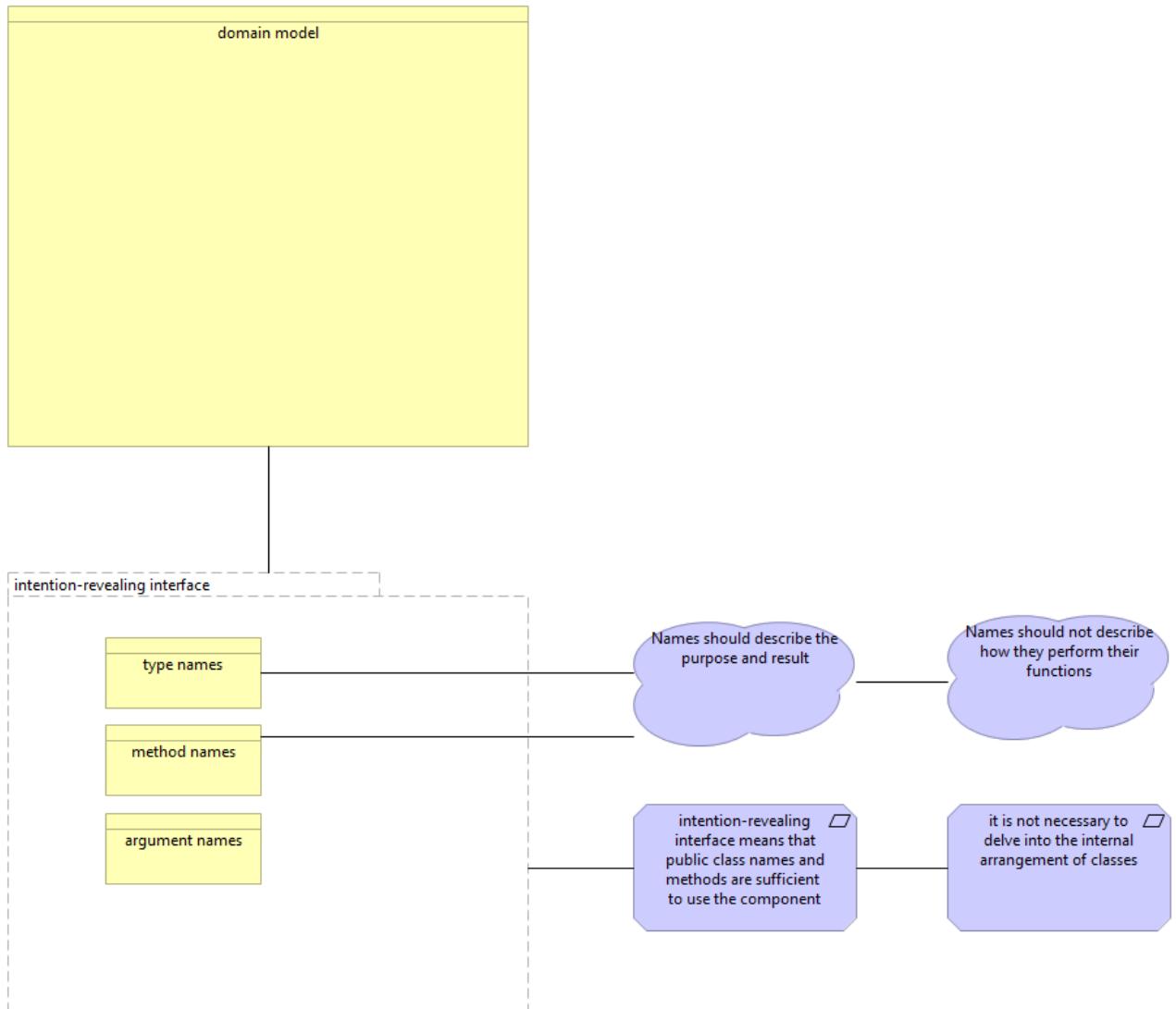
Eric Evans



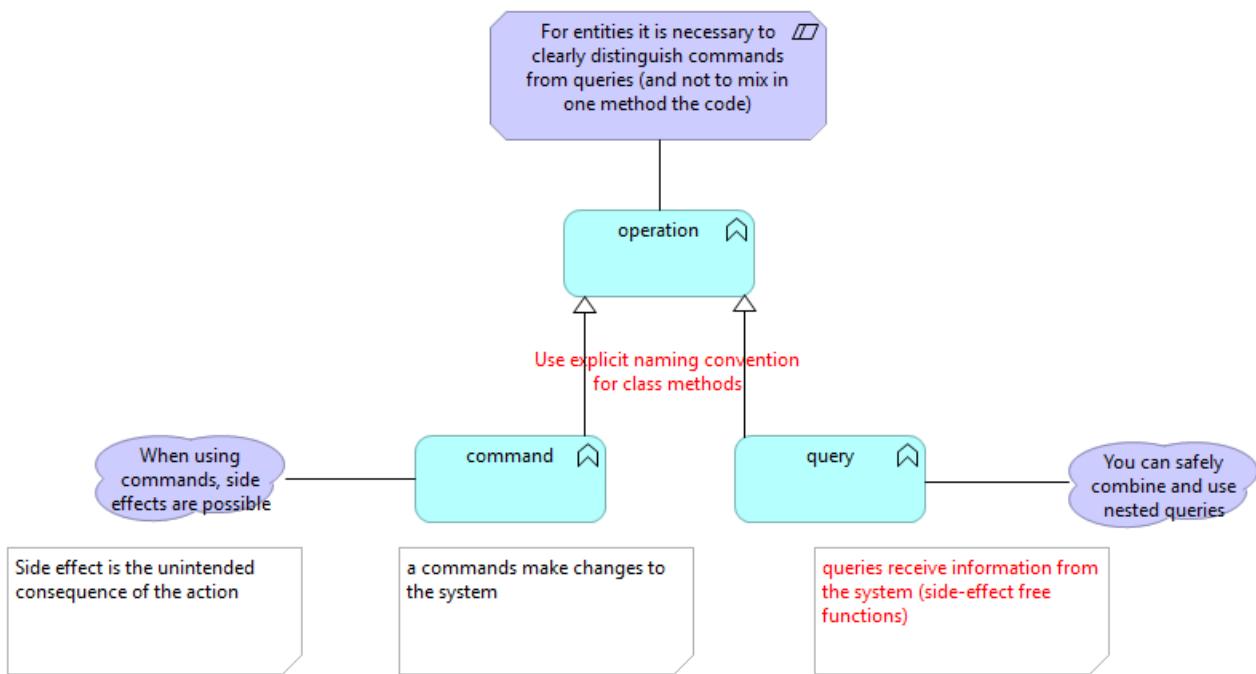
UBIQUITOUS LANGUAGE



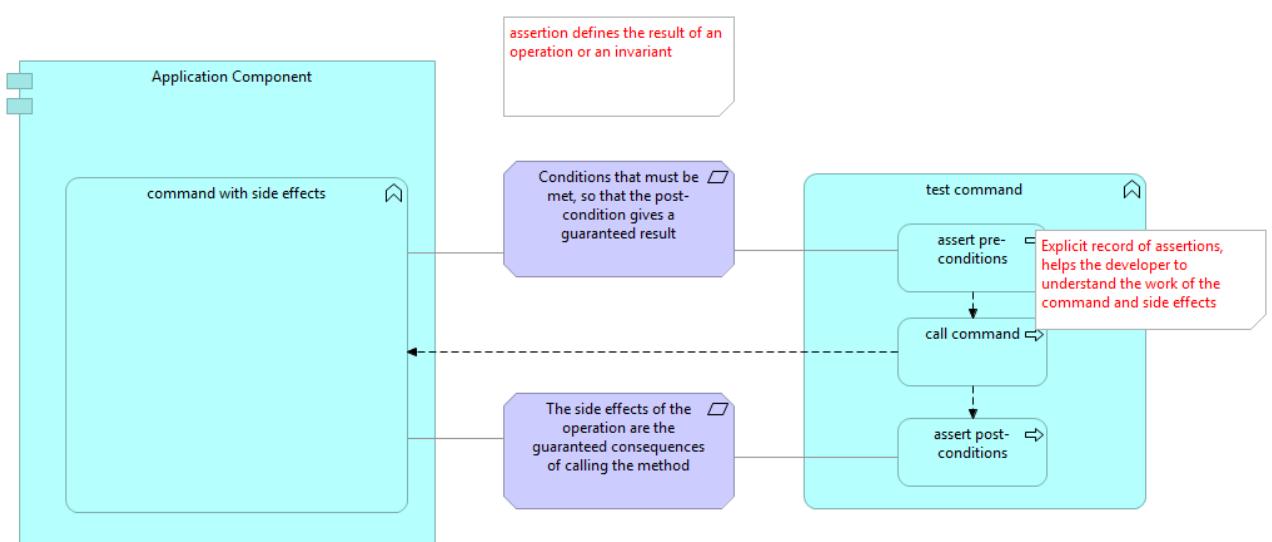
INTENTION-REVEALING INTERFACES



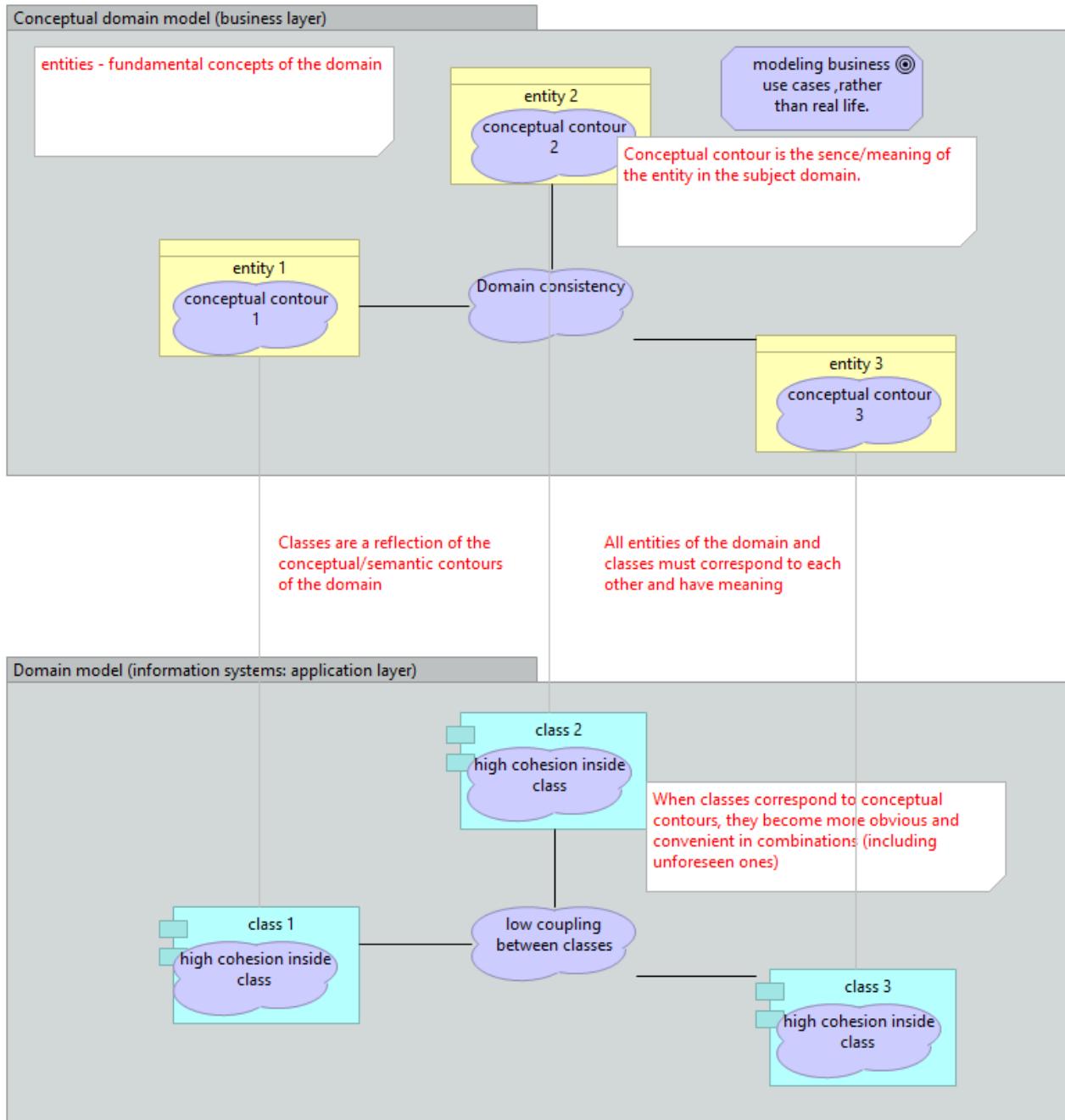
SIDE-EFFECT FREE FUNCTIONS



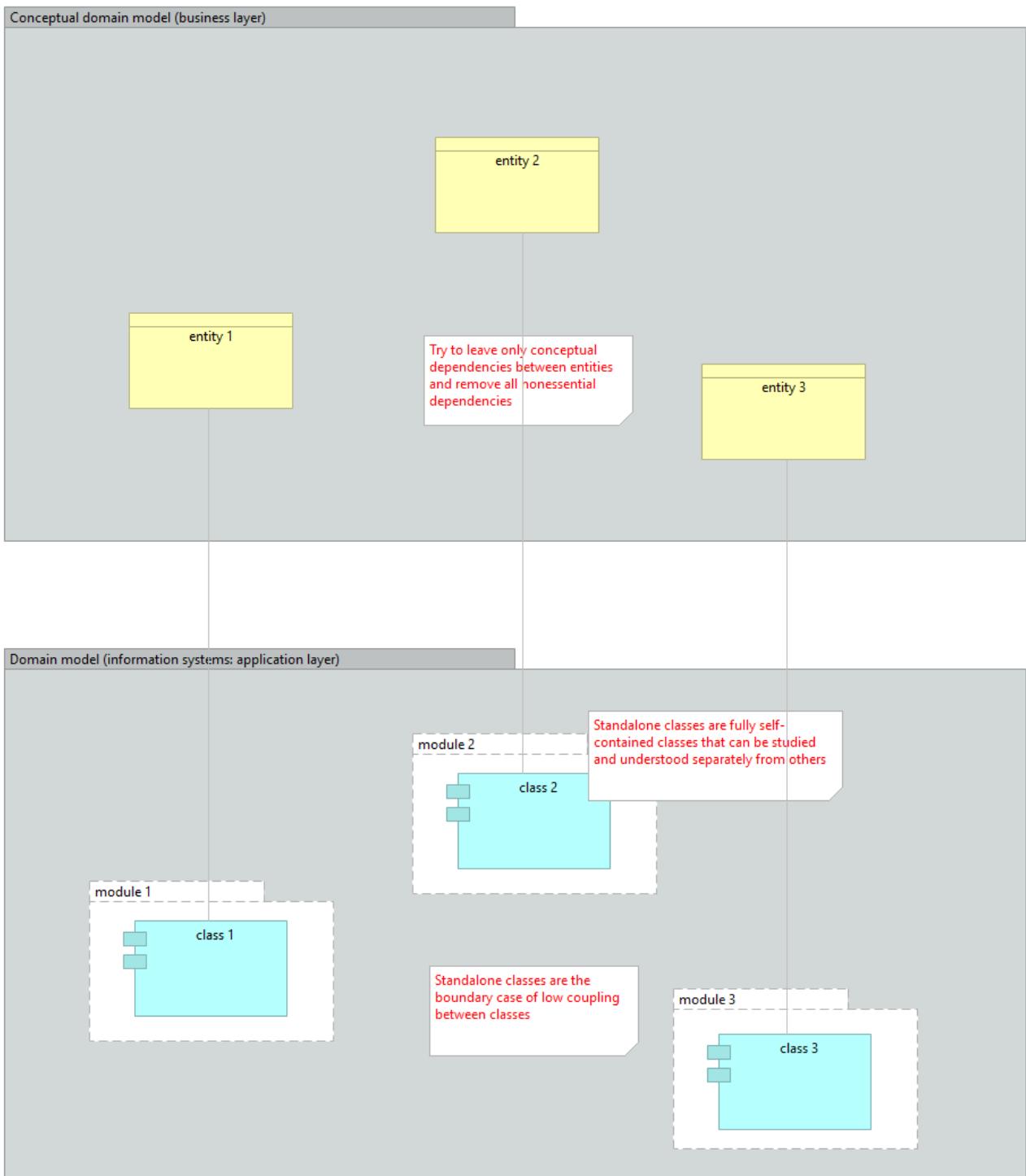
ASSERTIONS



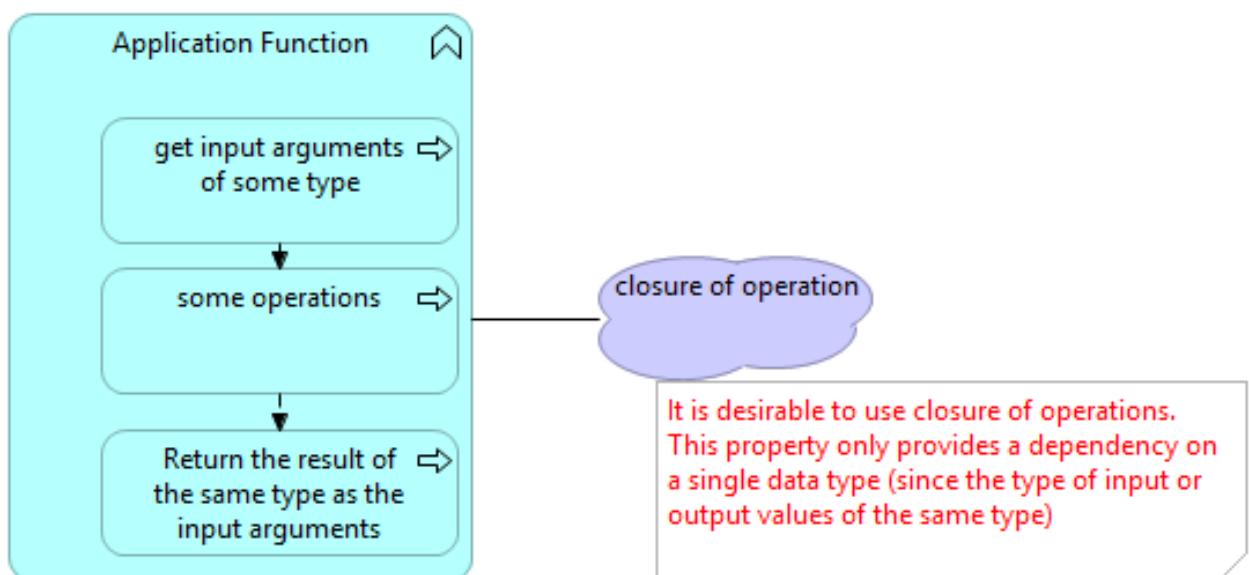
CONCEPTUAL CONTOURS



STANDALONE CLASSES



CLOSURE OF OPERATIONS



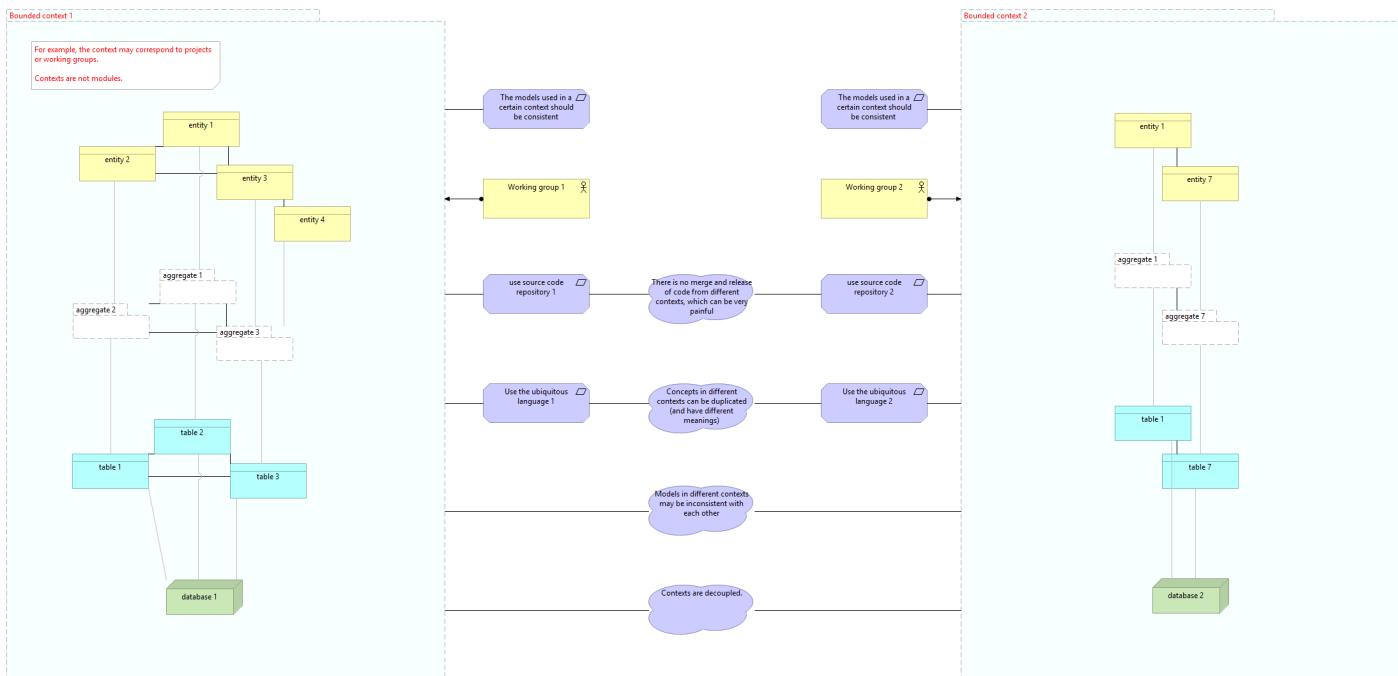
MODEL INTEGRITY AND CONTEXT

DOMAIN DRIVEN DESIGN

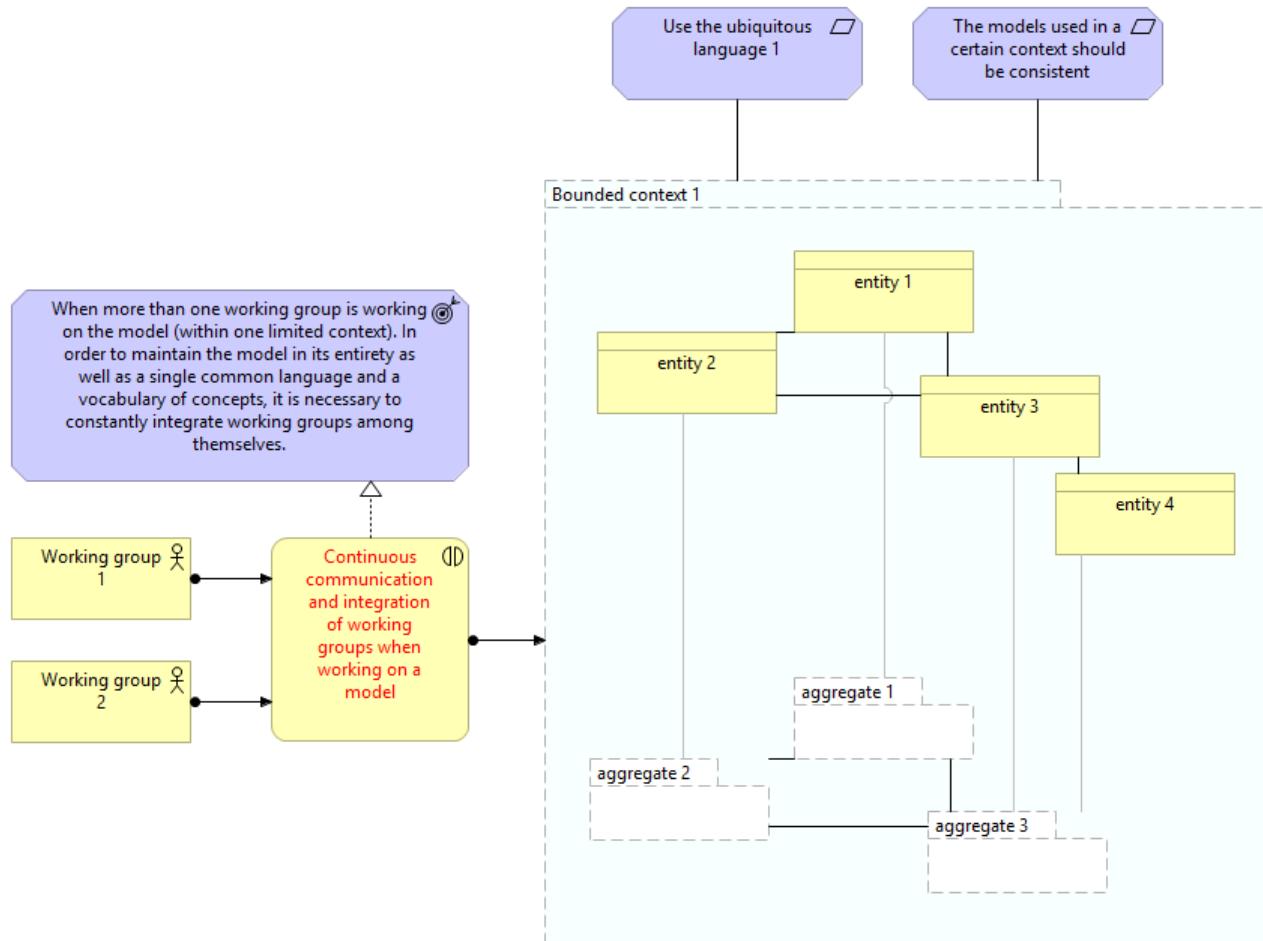
Eric Evans



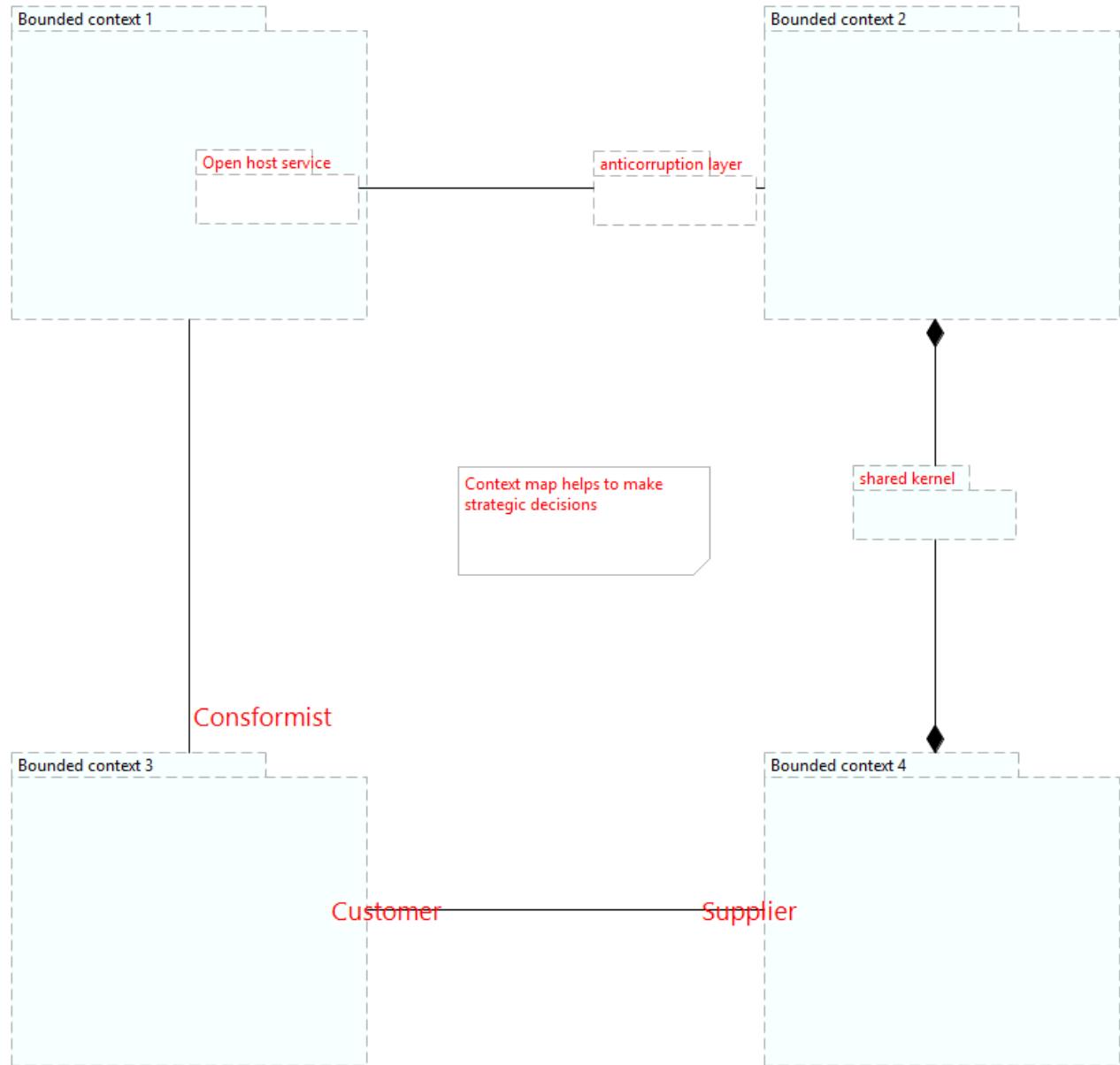
BOUNDED CONTEXT



CONTINUOUS INTEGRATION

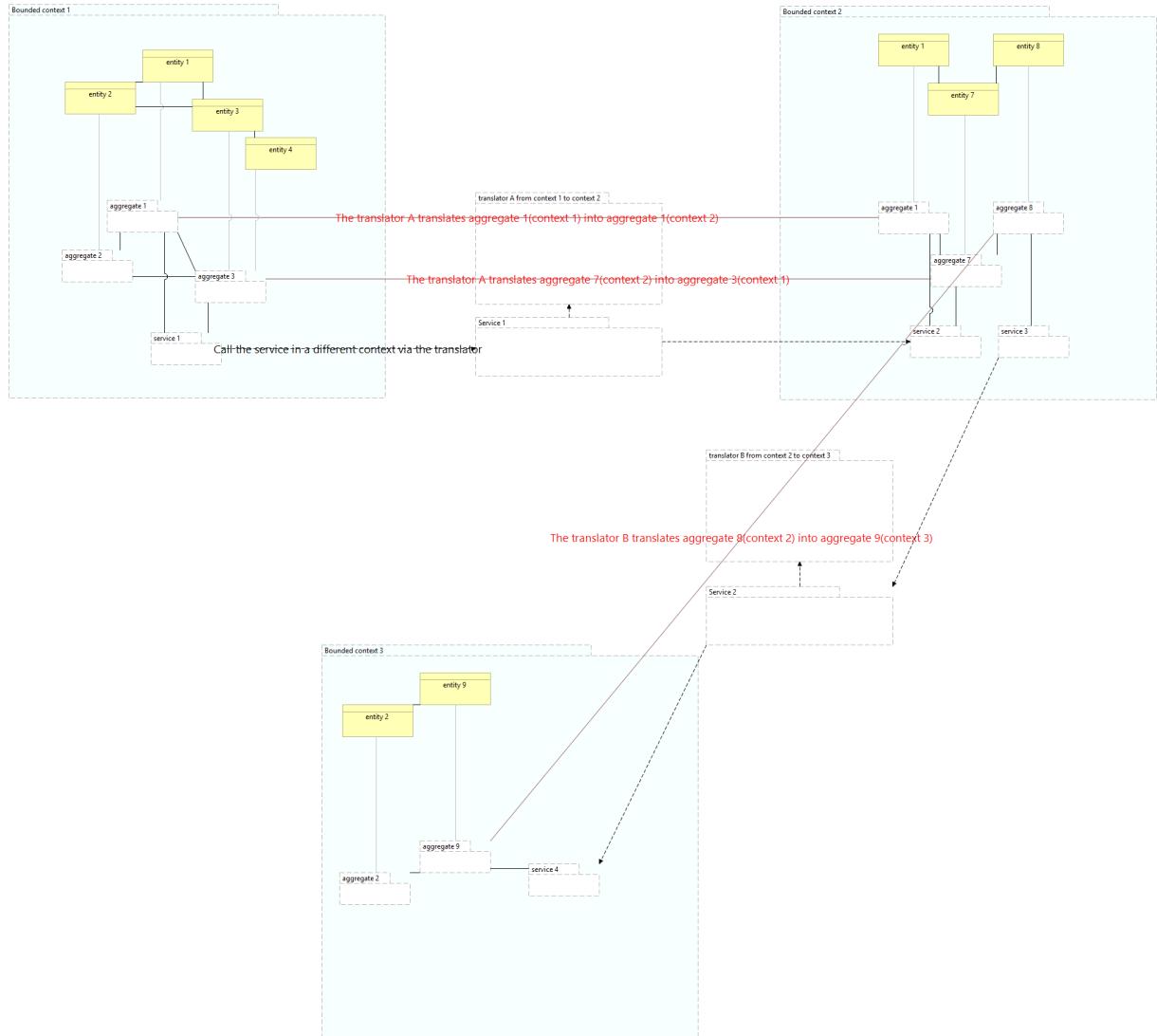


STRATEGIC CONTEXT MAP

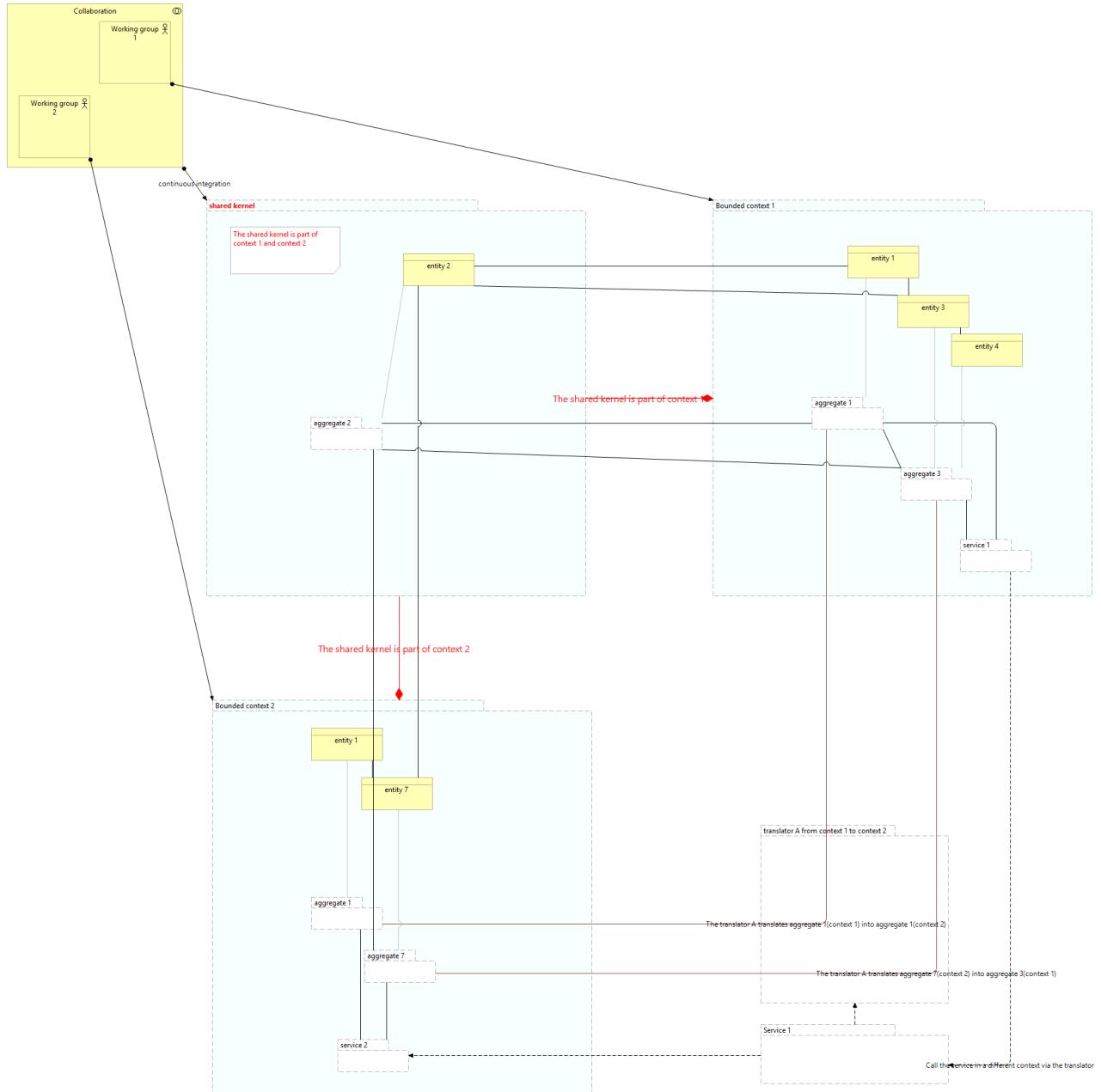


CONTEXTUAL MAP

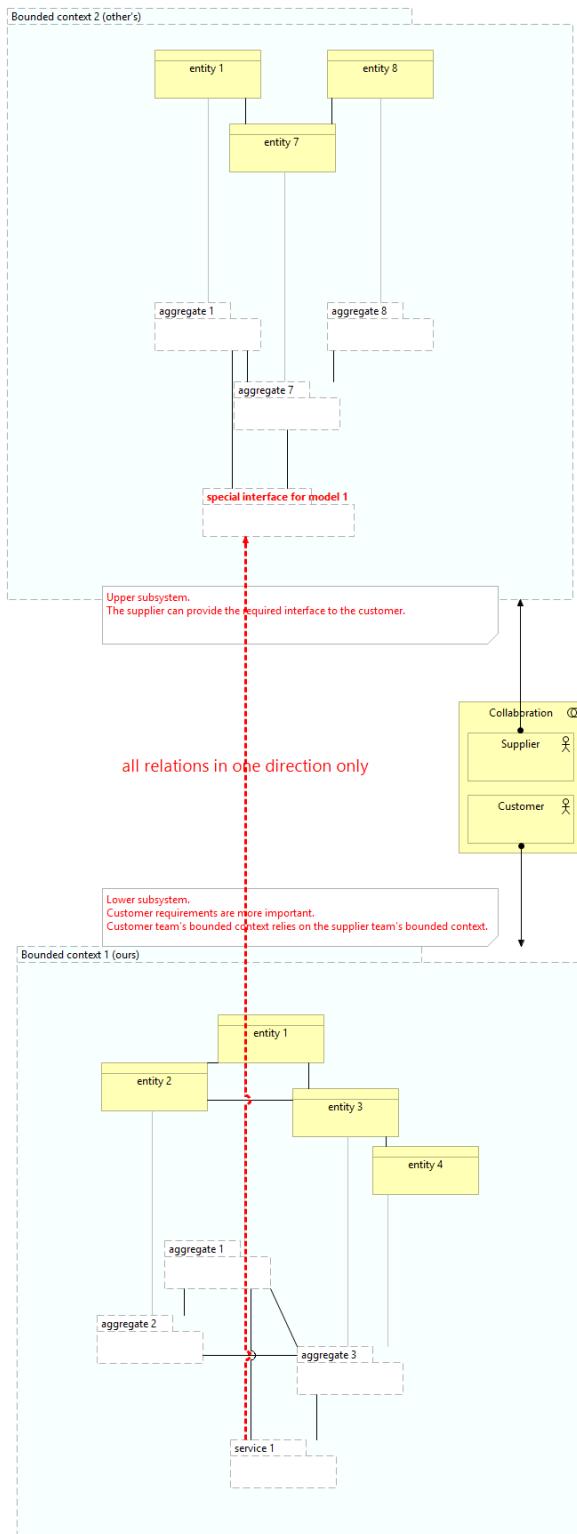
A single map of all contexts.
Integration of all bounded contexts.



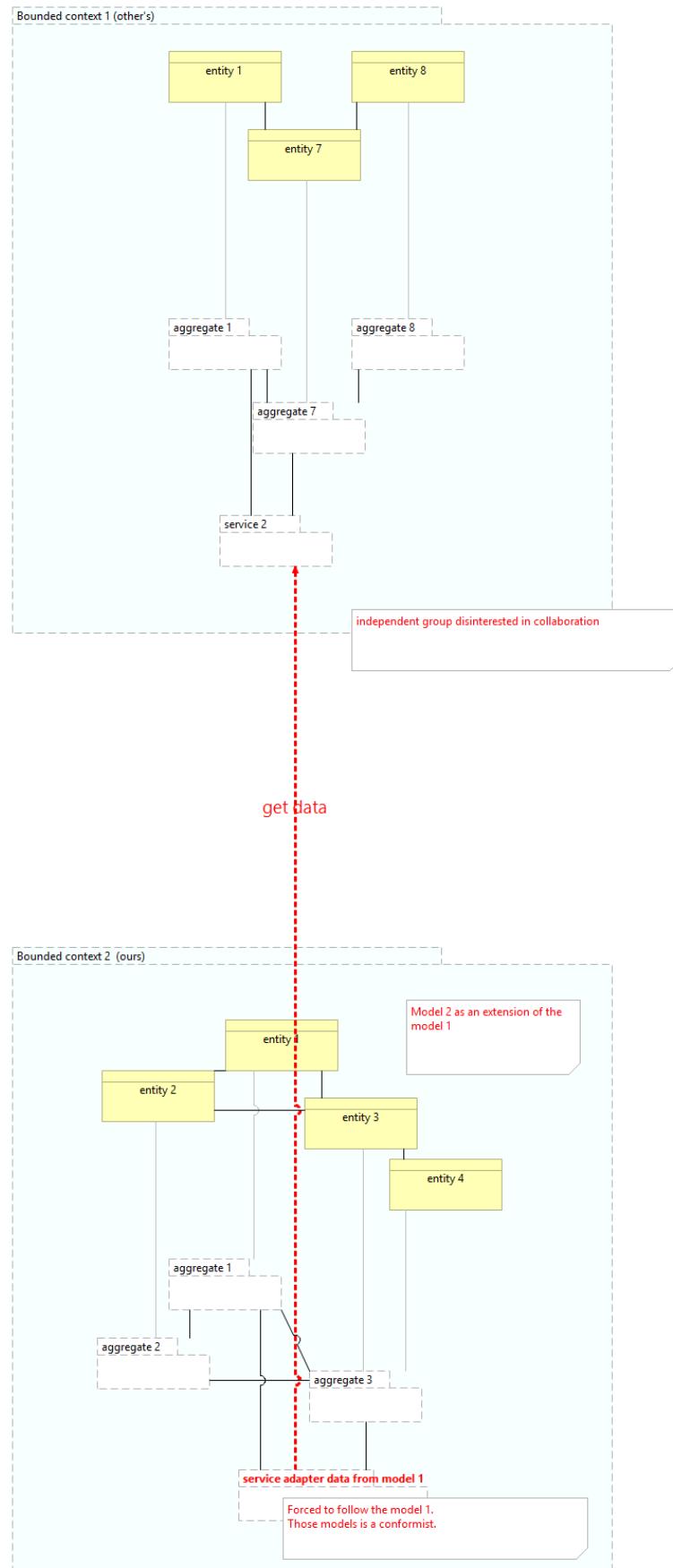
SHARED KERNEL



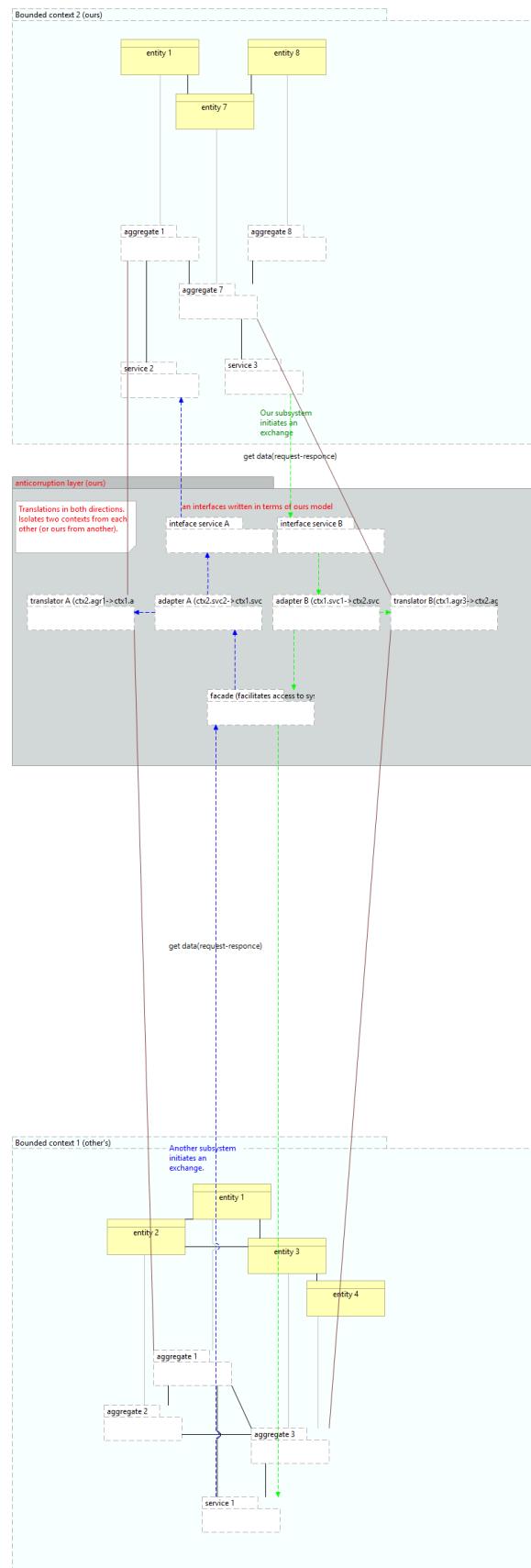
CUSTOMER-SUPPLIER TEAMS



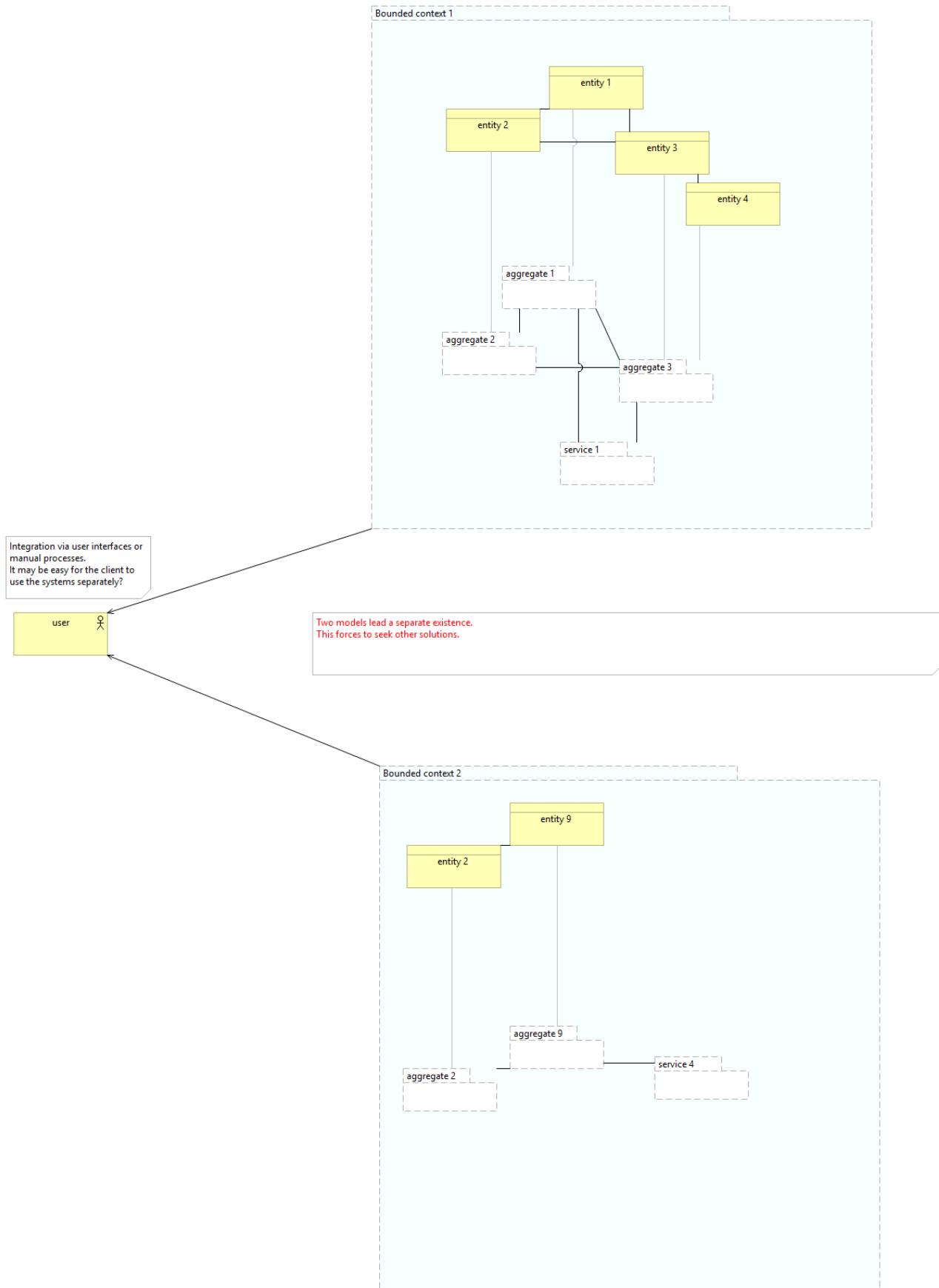
CONFORMIST



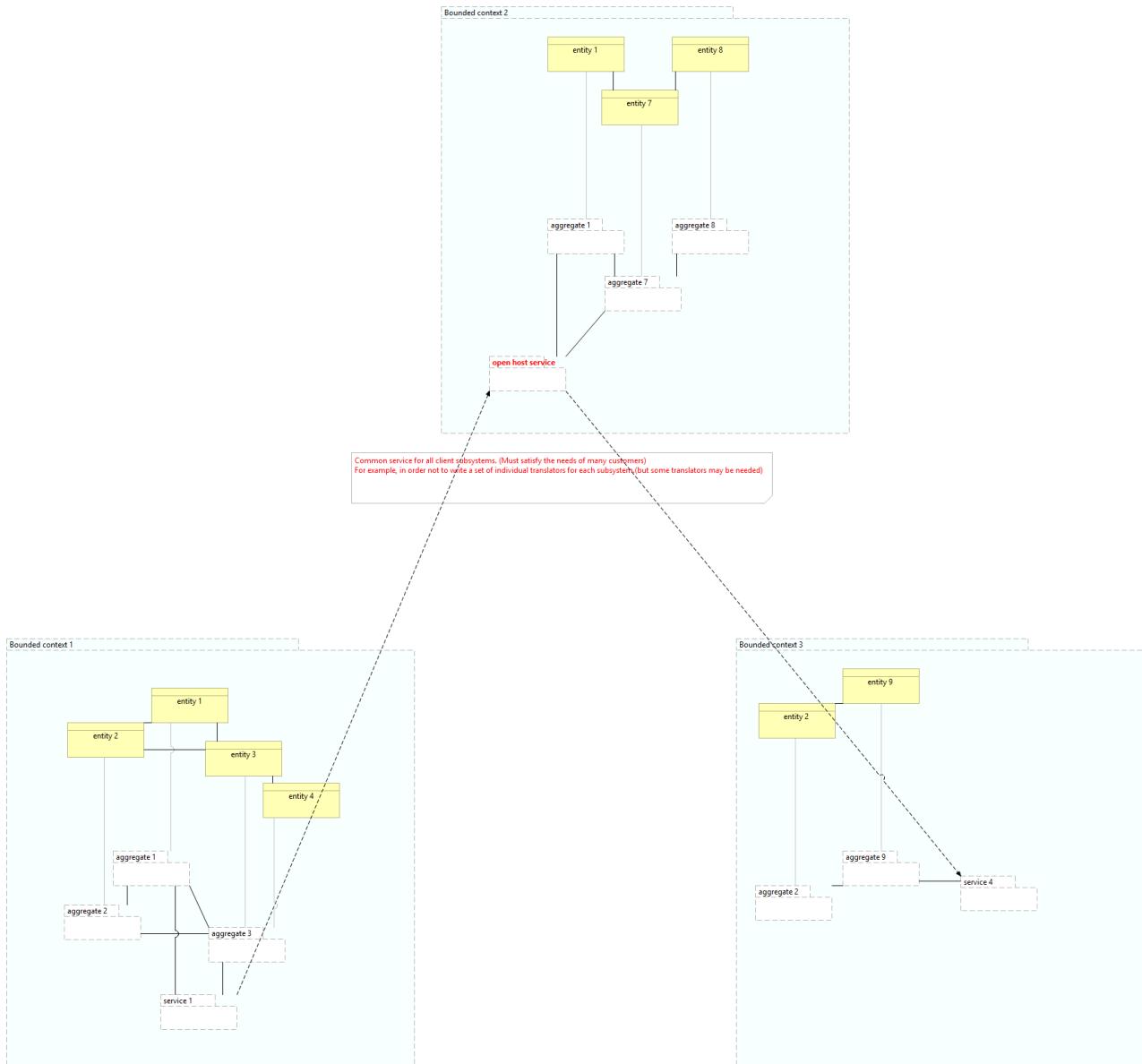
ANTICORRUPTION LAYER



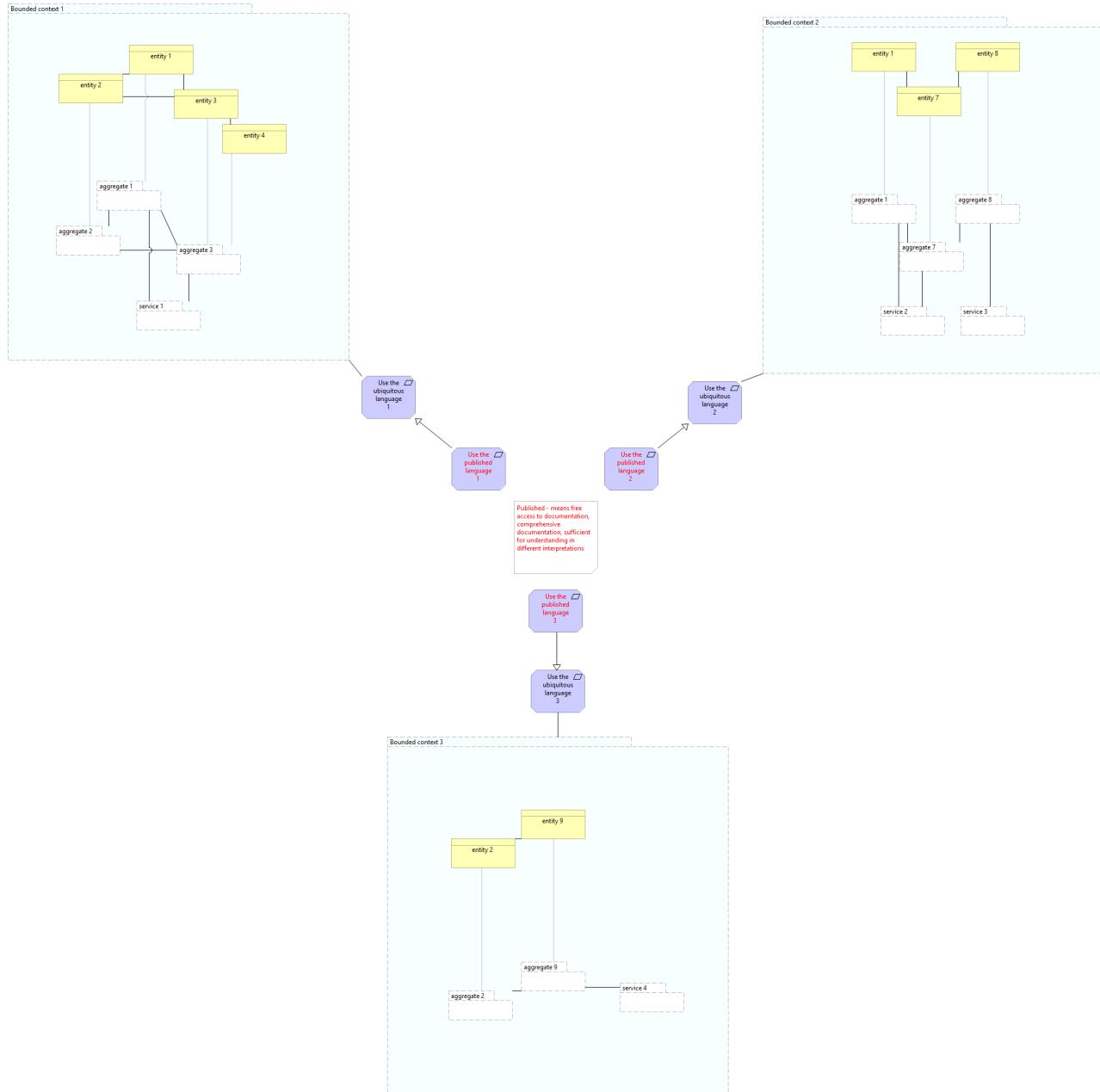
SEPARATE WAYS



OPEN HOST SERVICE



PUBLISHED LANGUAGE



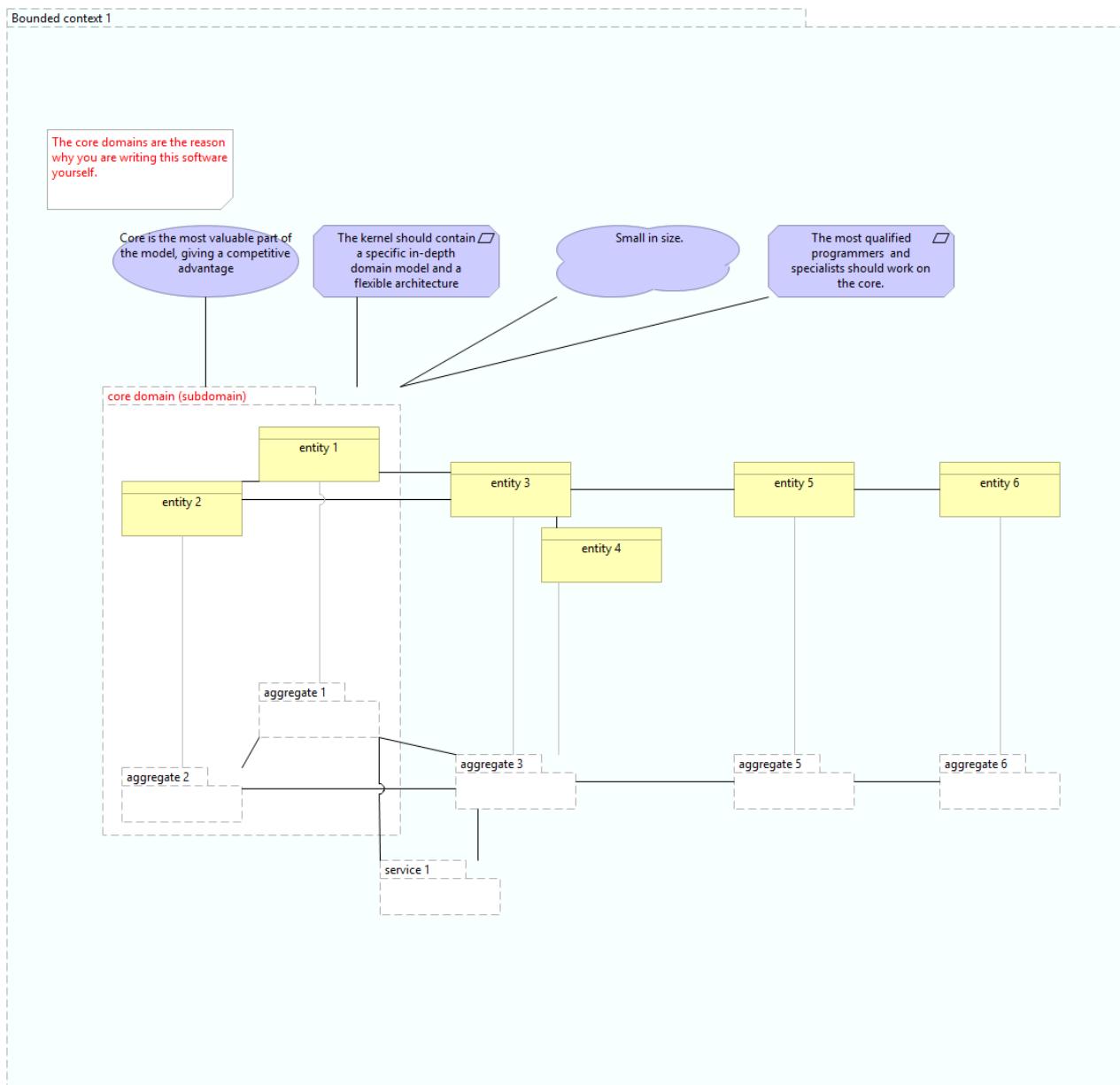
DISTILLATION

DOMAIN DRIVEN DESIGN

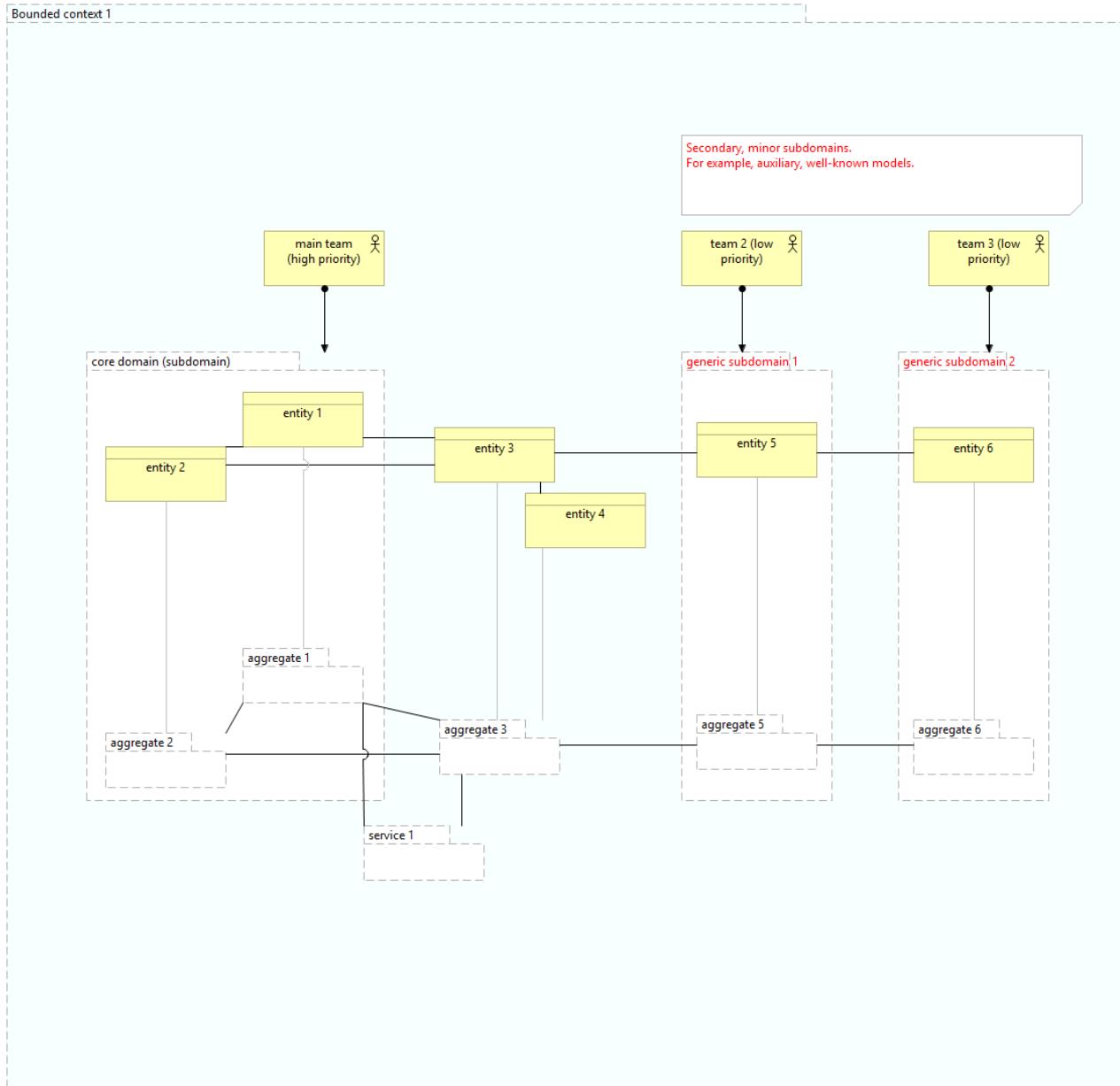
Eric Evans



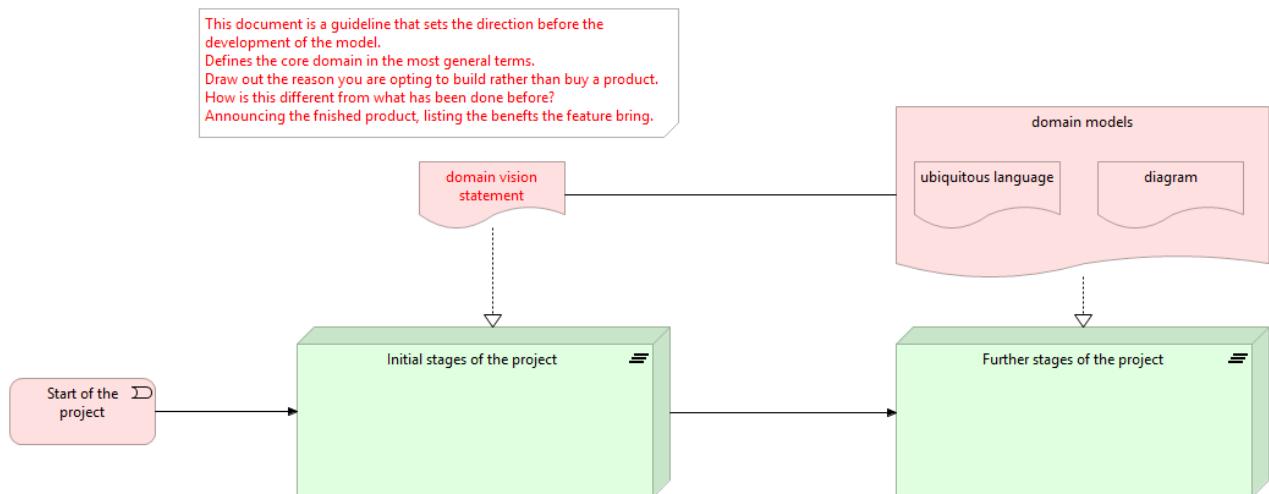
CORE DOMAIN



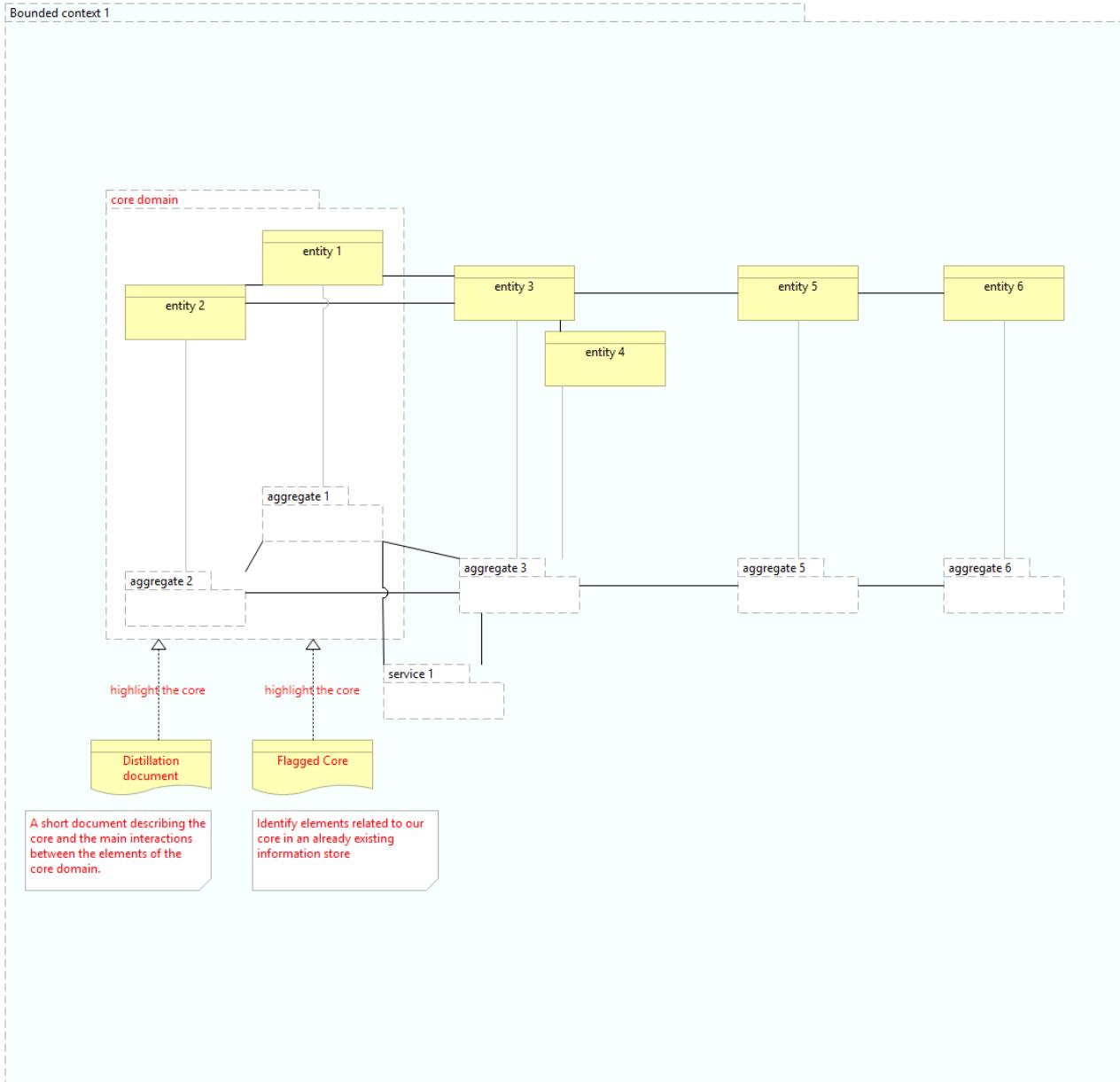
GENERIC SUBDOMAINS



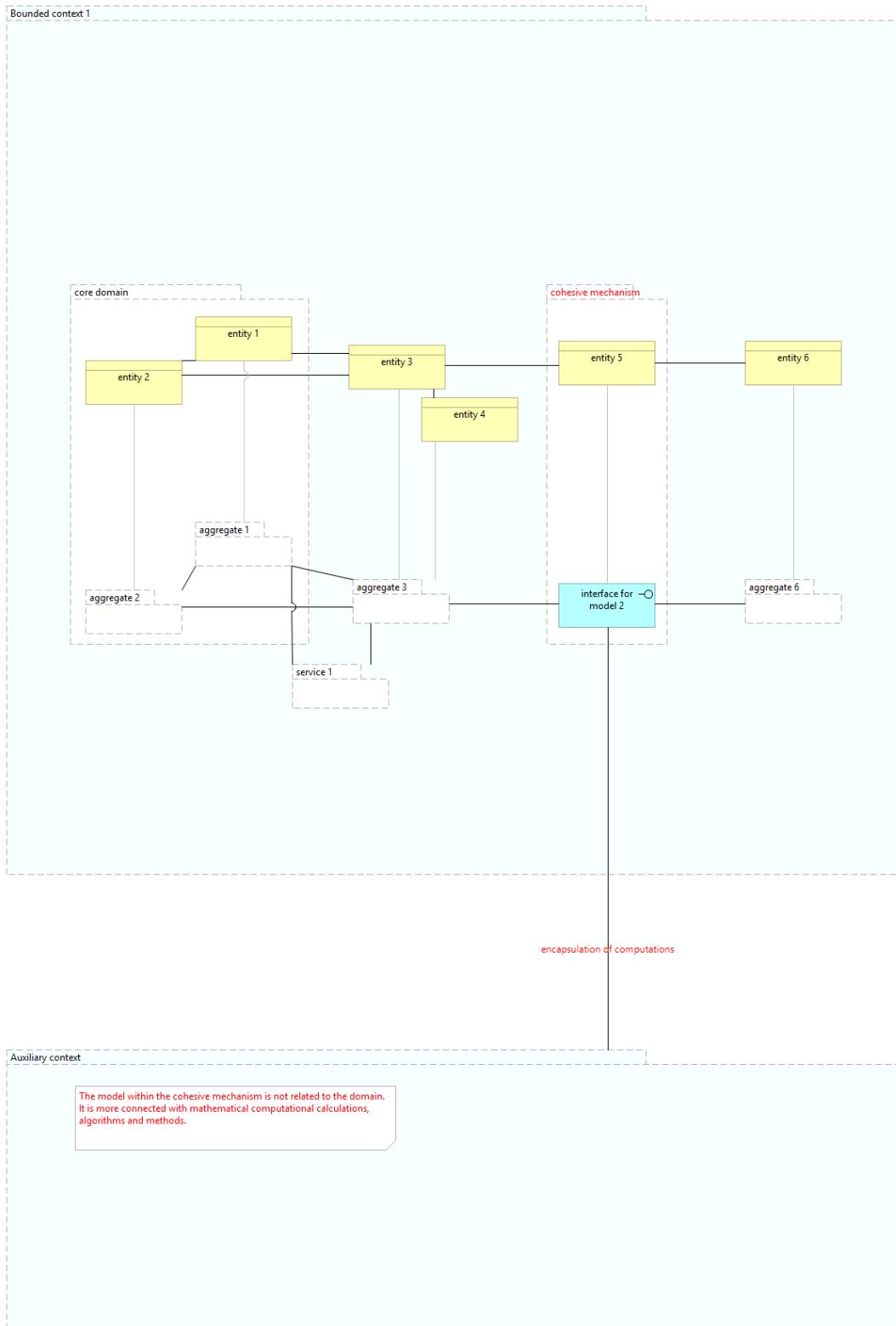
DOMAIN VISION STATEMENT



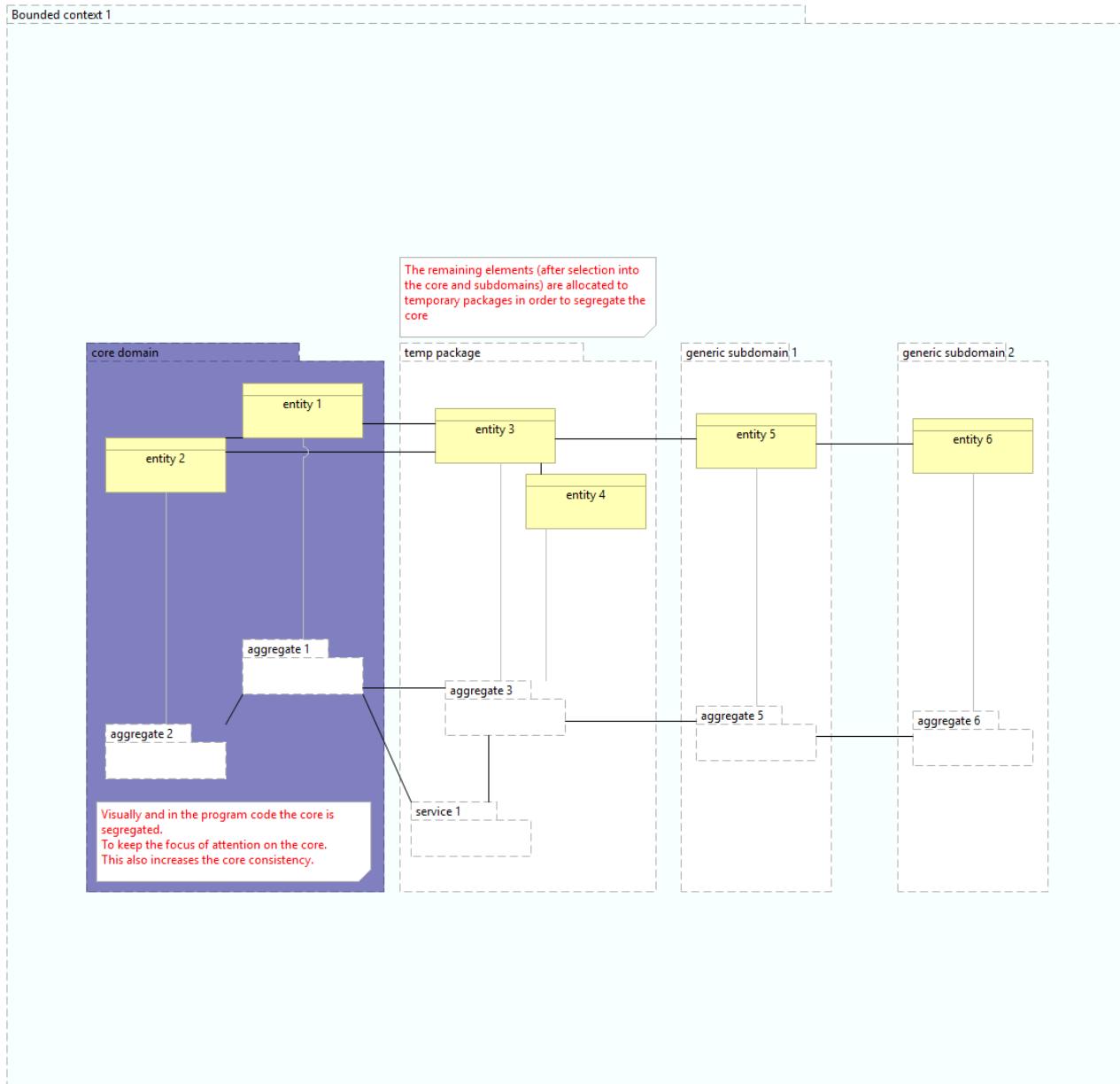
HIGHLIGHTED CORE



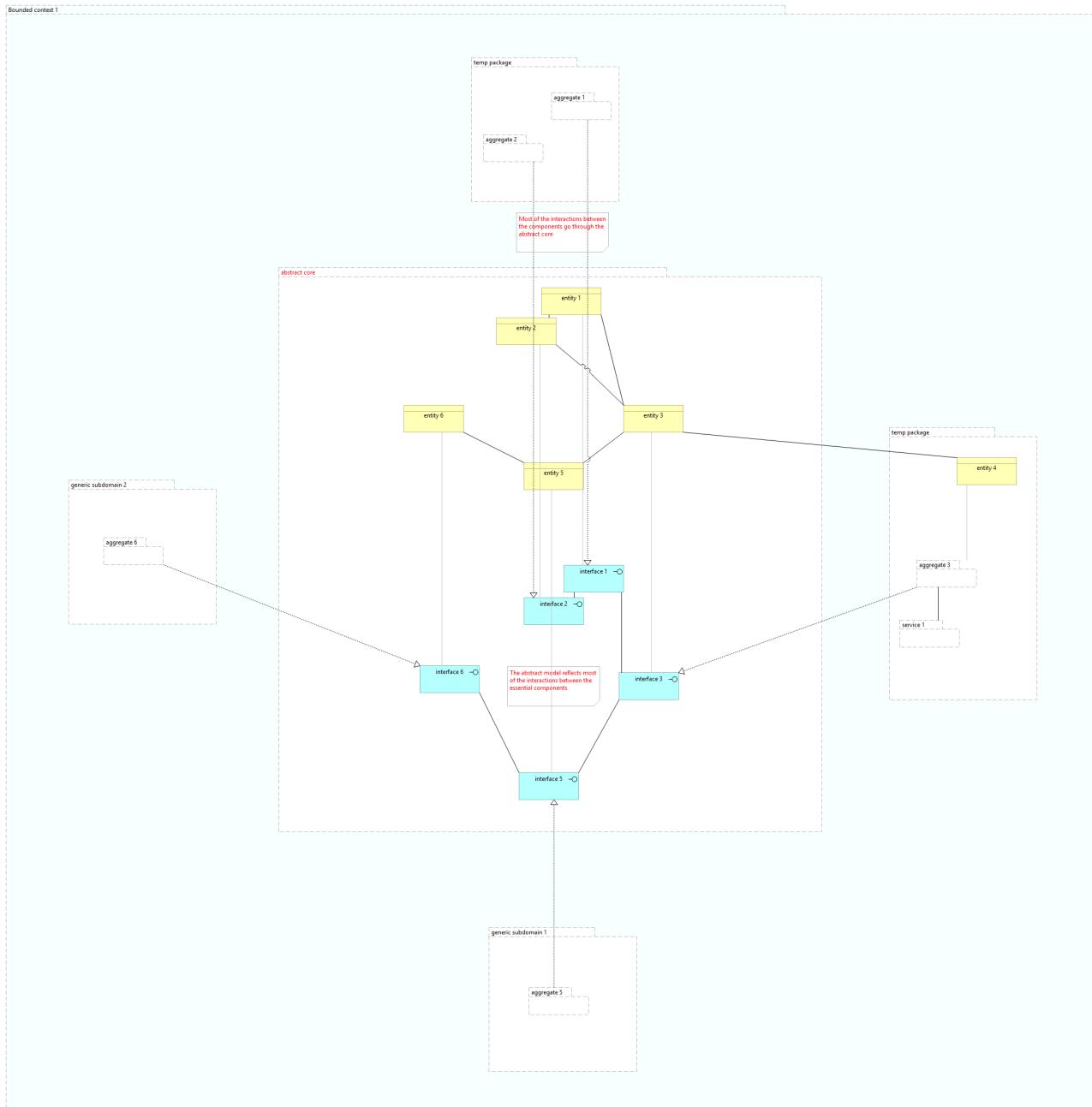
COHESIVE MECHANISMS



SEGREGATED CORE



ABSTRACT CORE



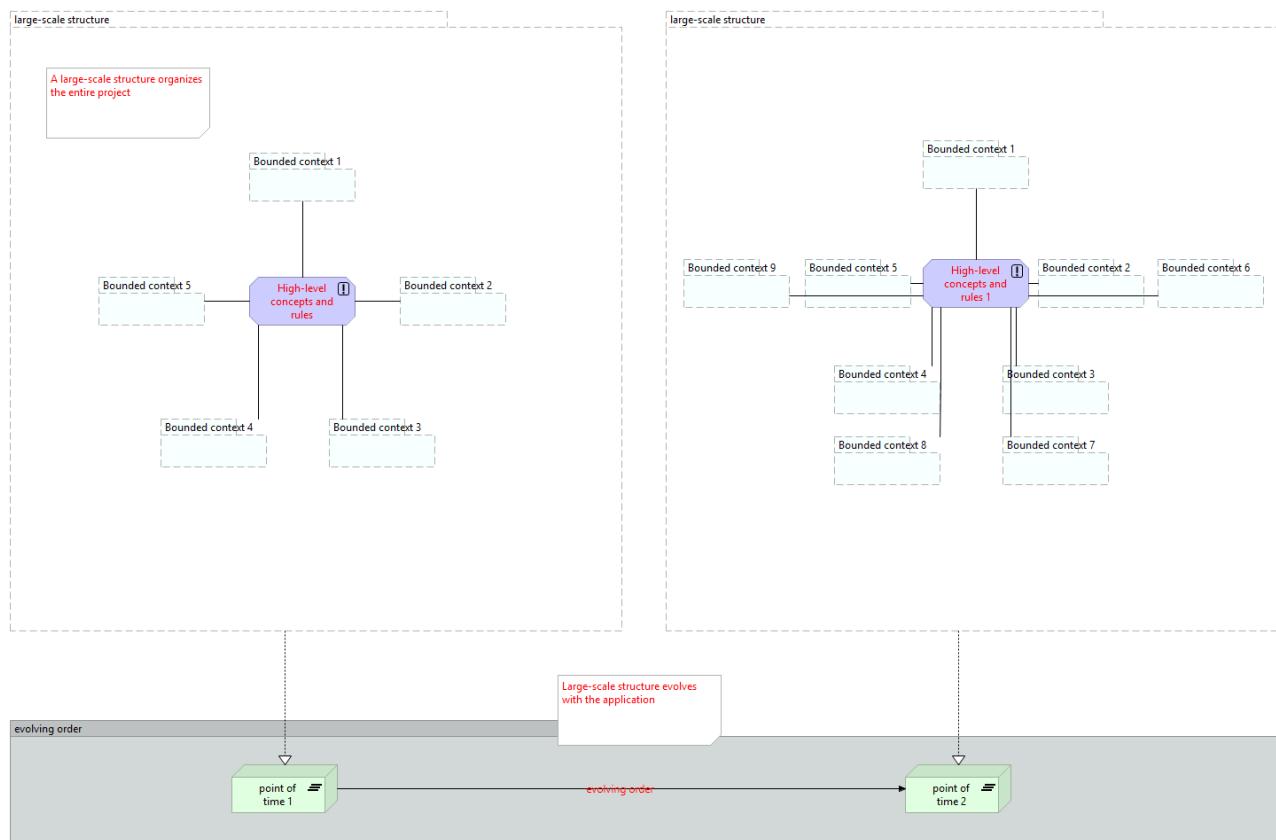
LARGE-SCALE STRUCTURE

DOMAIN DRIVEN DESIGN

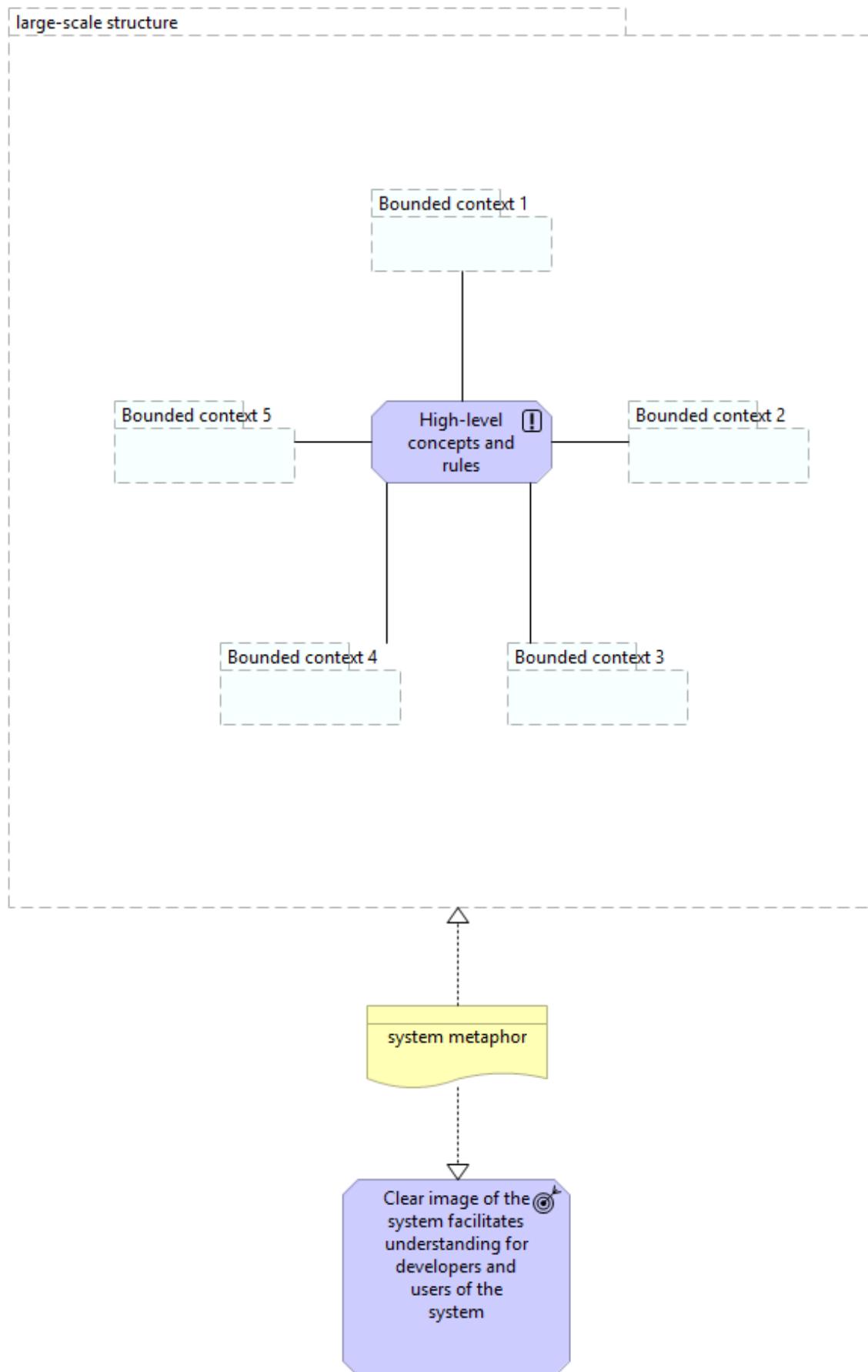
Eric Evans



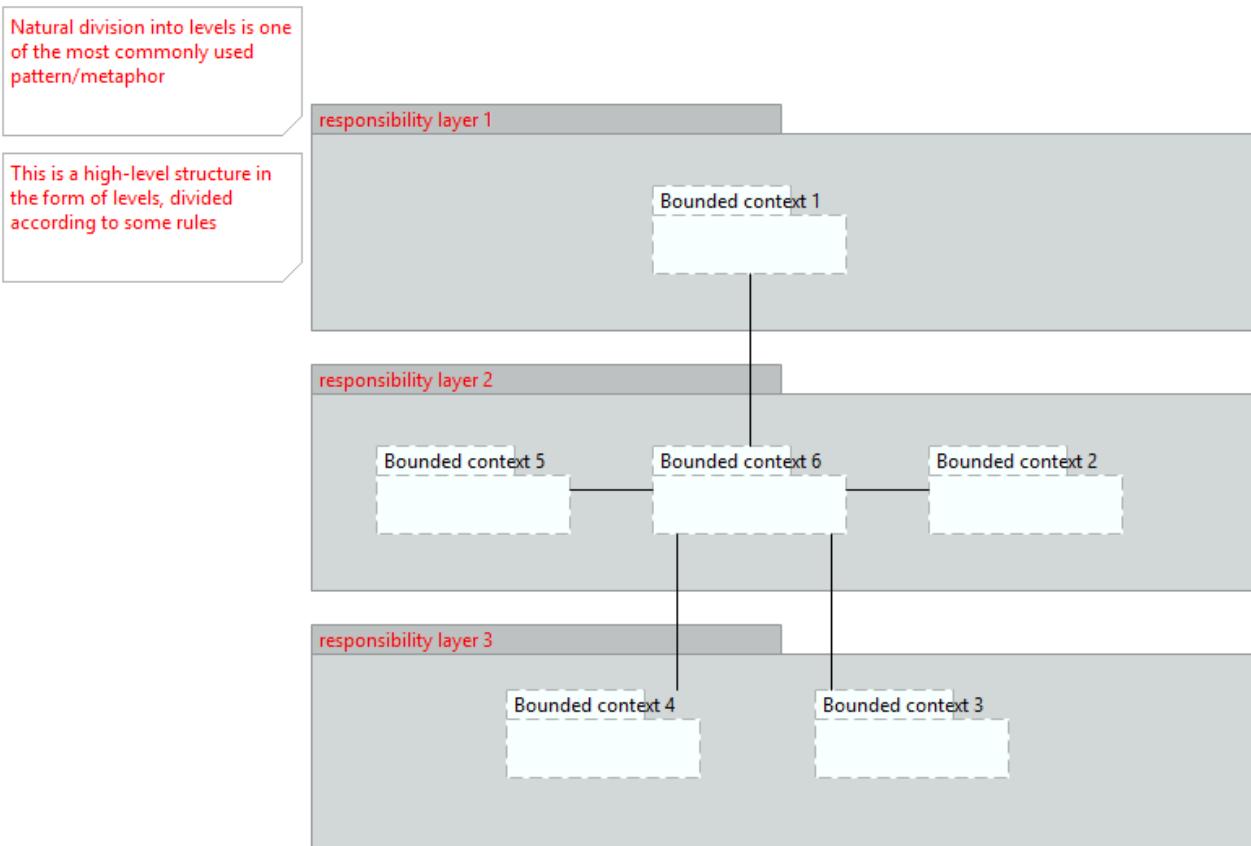
EVOLVING ORDER



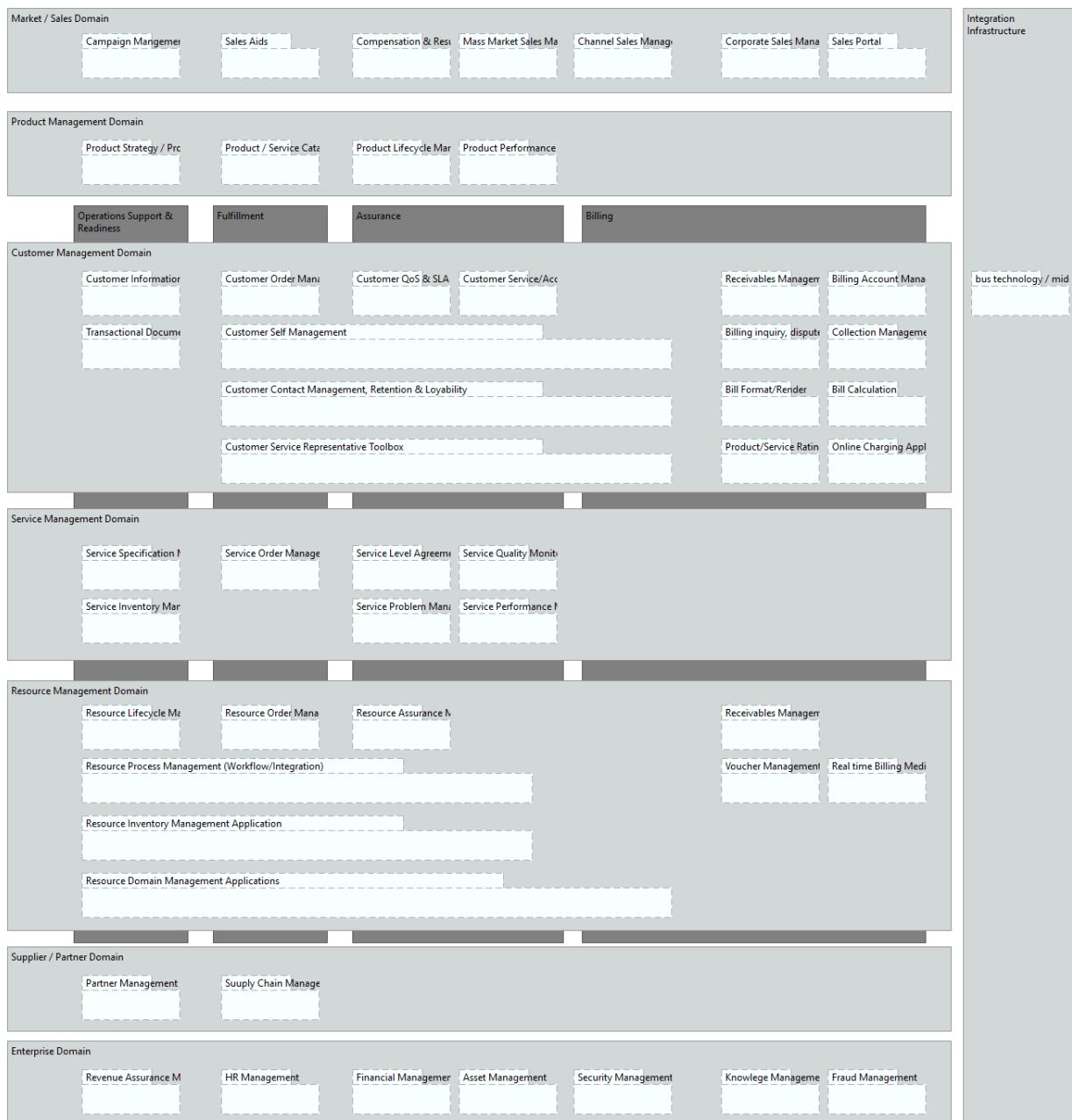
SYSTEM METAPHOR



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

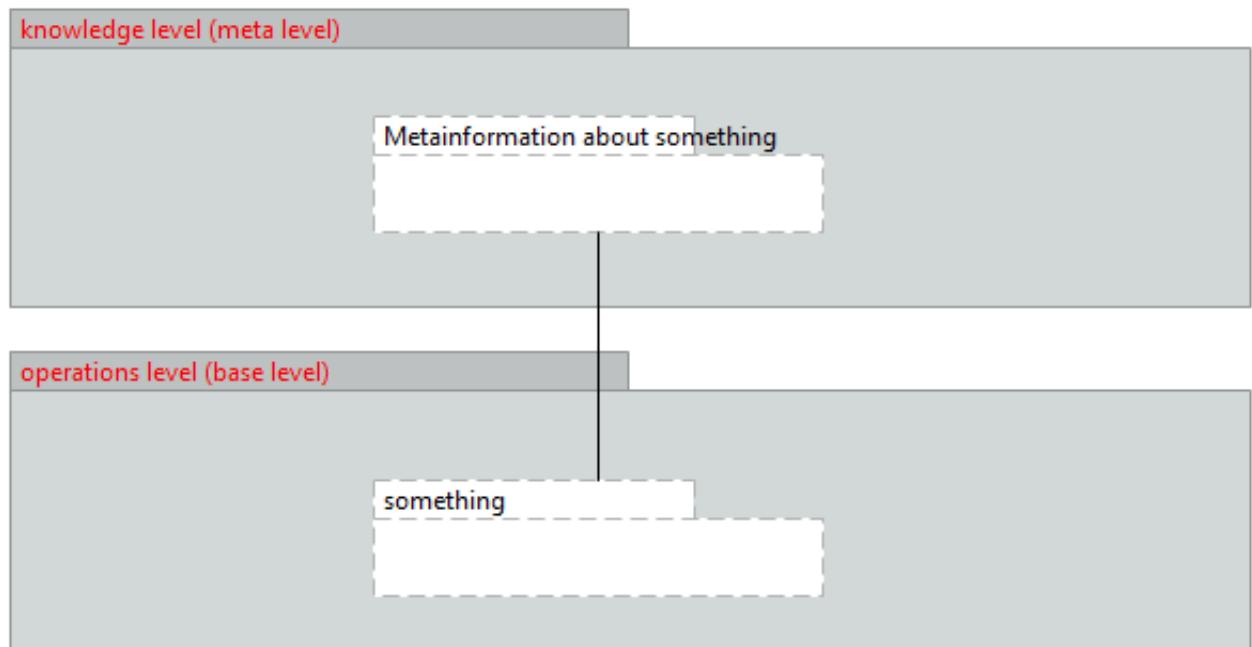
Bounded context 10

Service

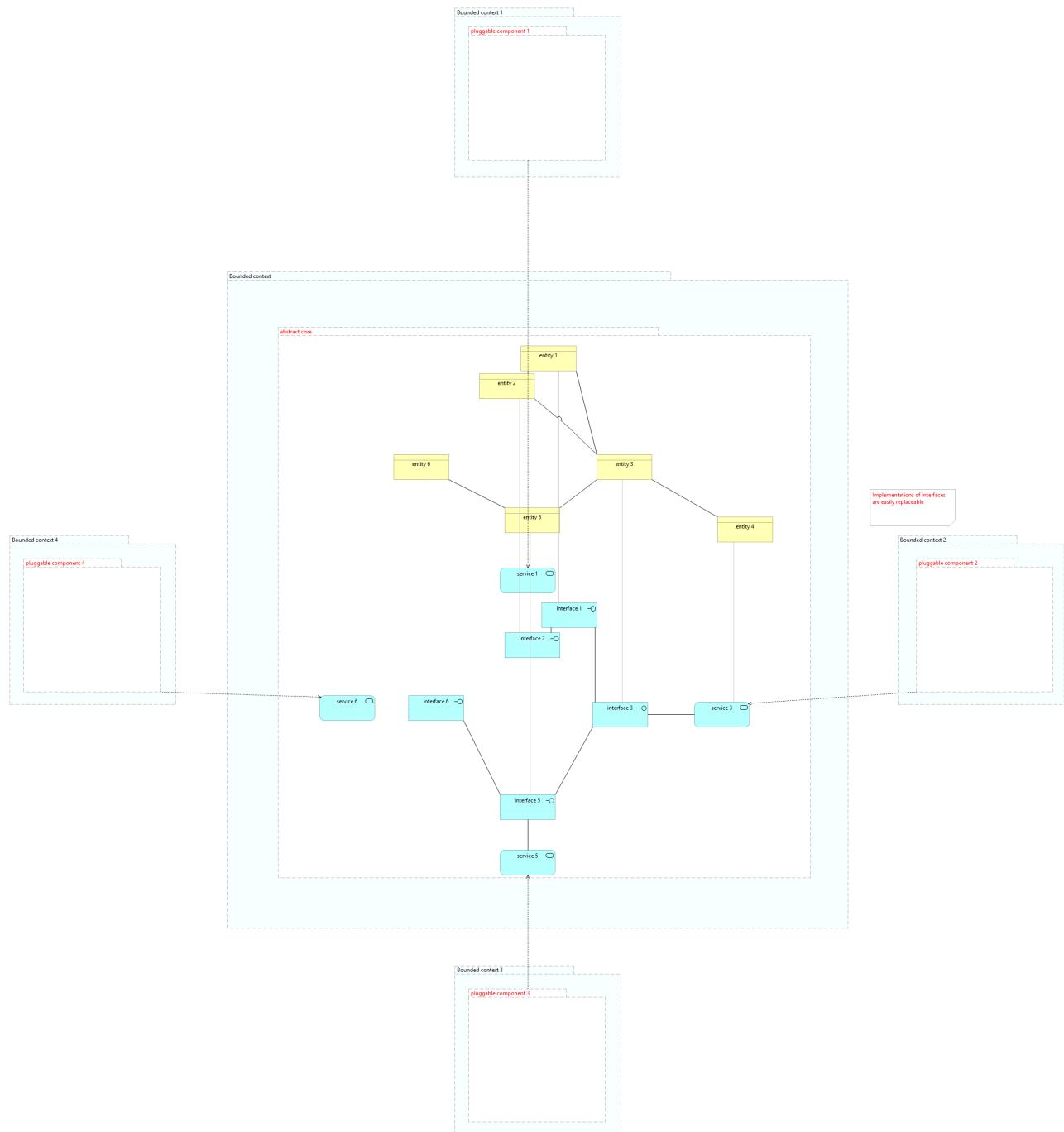
Bounded context 11



KNOWLEDGE LEVEL



PLUGGABLE COMPONENT FRAMEWORK



ADDITIONAL PATTERNS

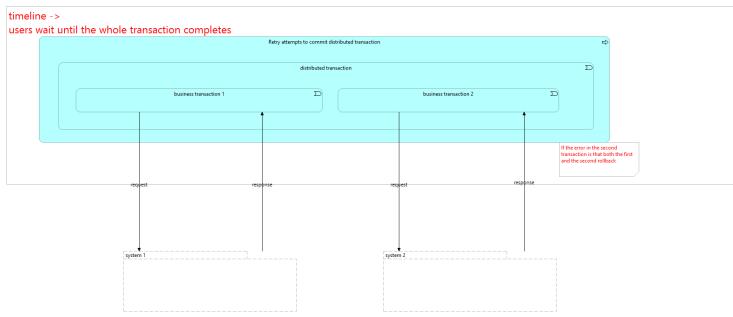
DOMAIN DRIVEN DESIGN

Eric Evans

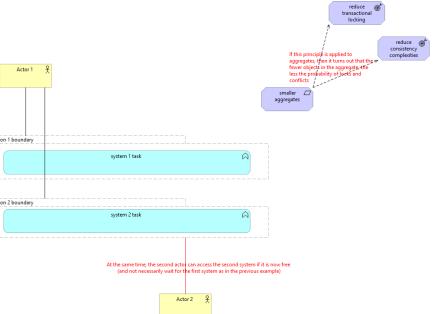
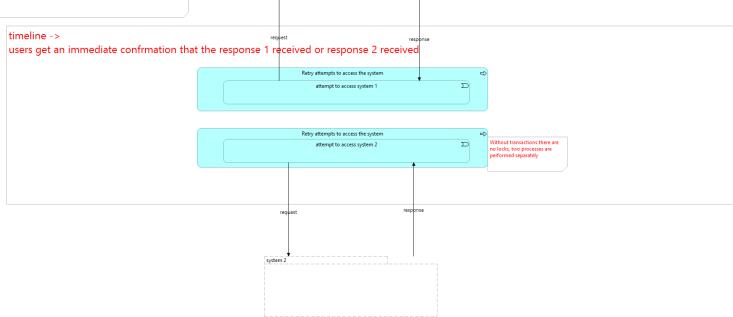


TYPES OF CONSISTENCY

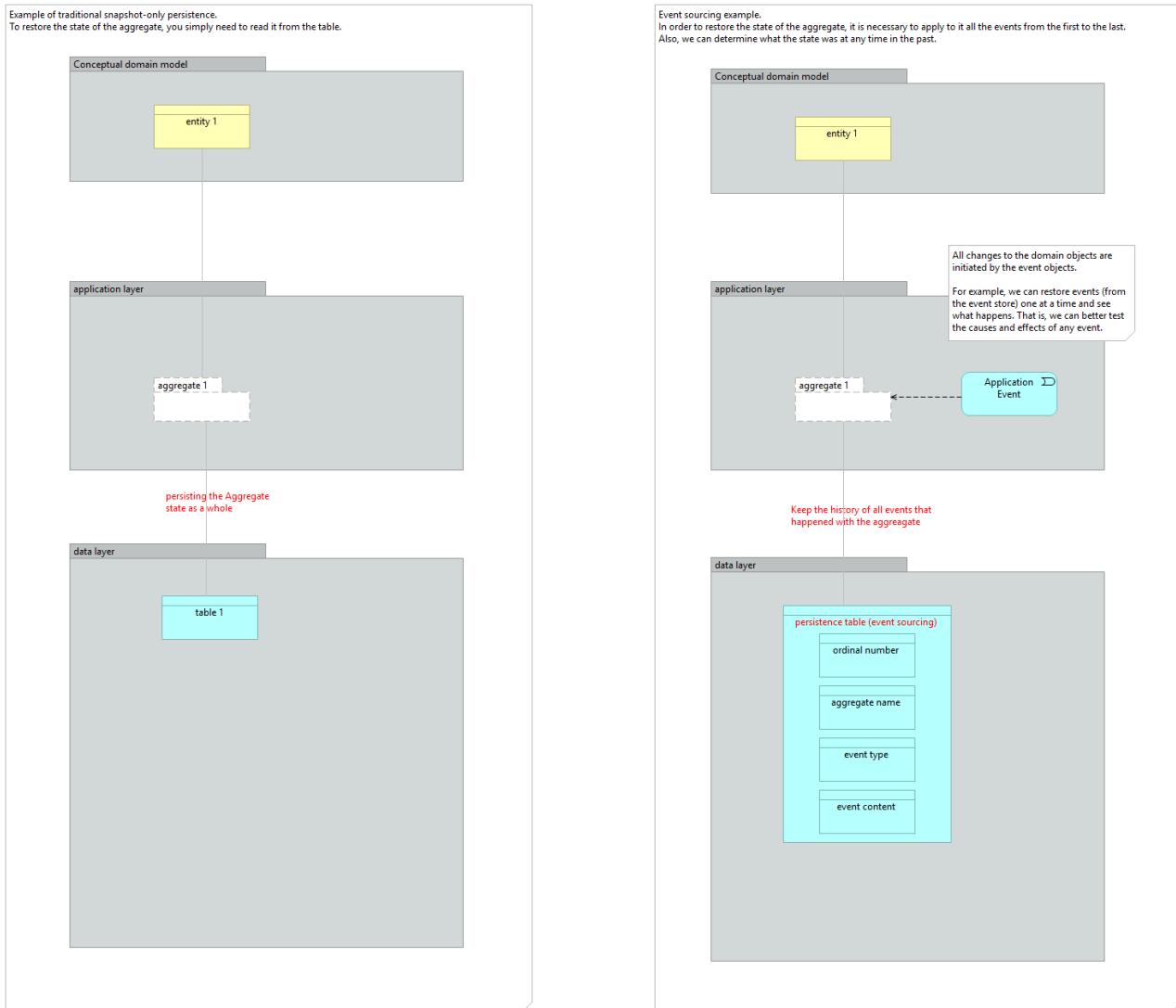
Transactional consistency:
When a transaction is finished the system is in a consistent state.
C� A distributed transaction is a workflow.
The more objects involved in a distributed transaction, the greater the risk of potential loss and conflicts.



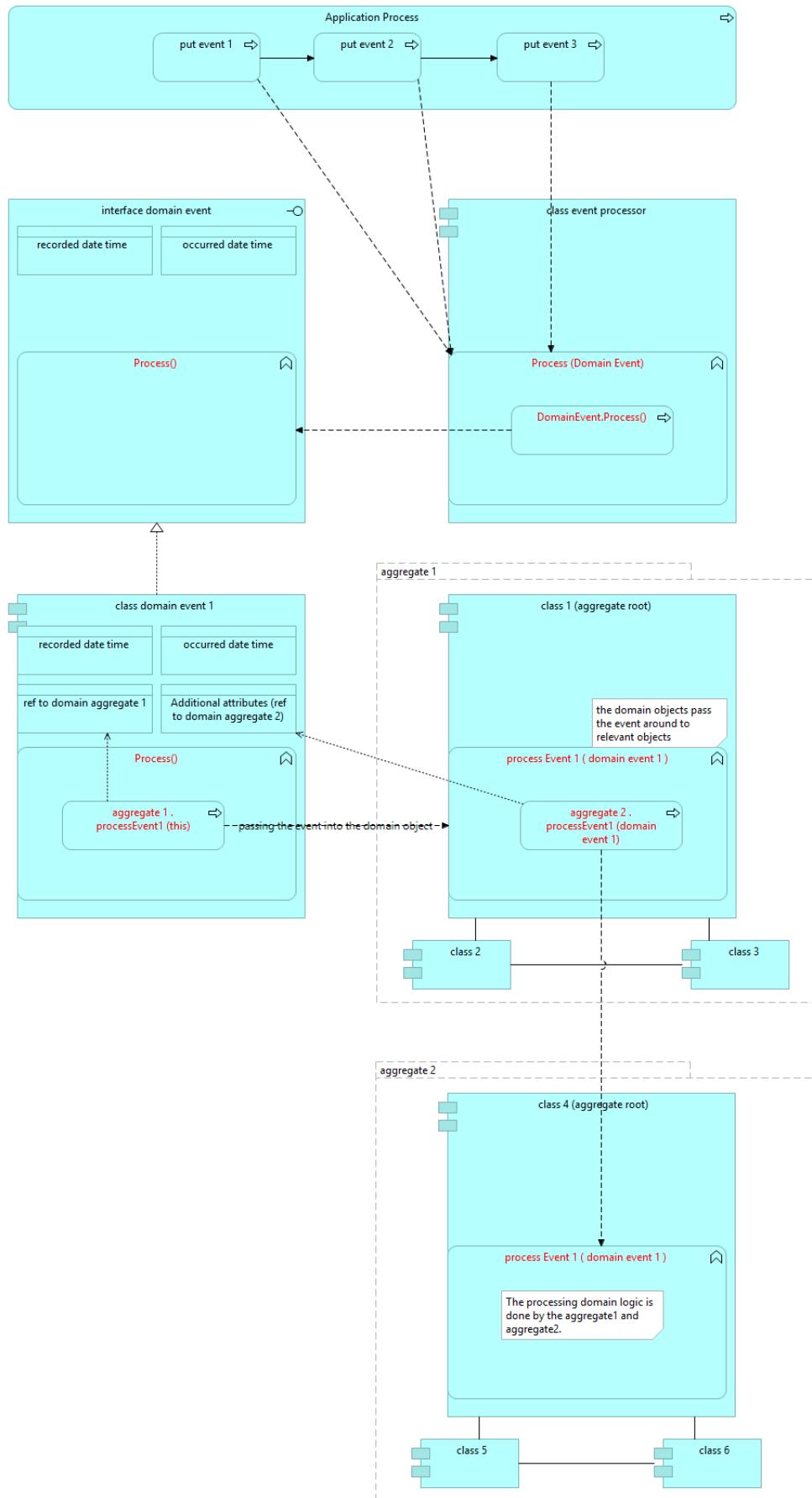
Eventual Consistency:
B&D (Best Available, Soft state, Eventual consistency)



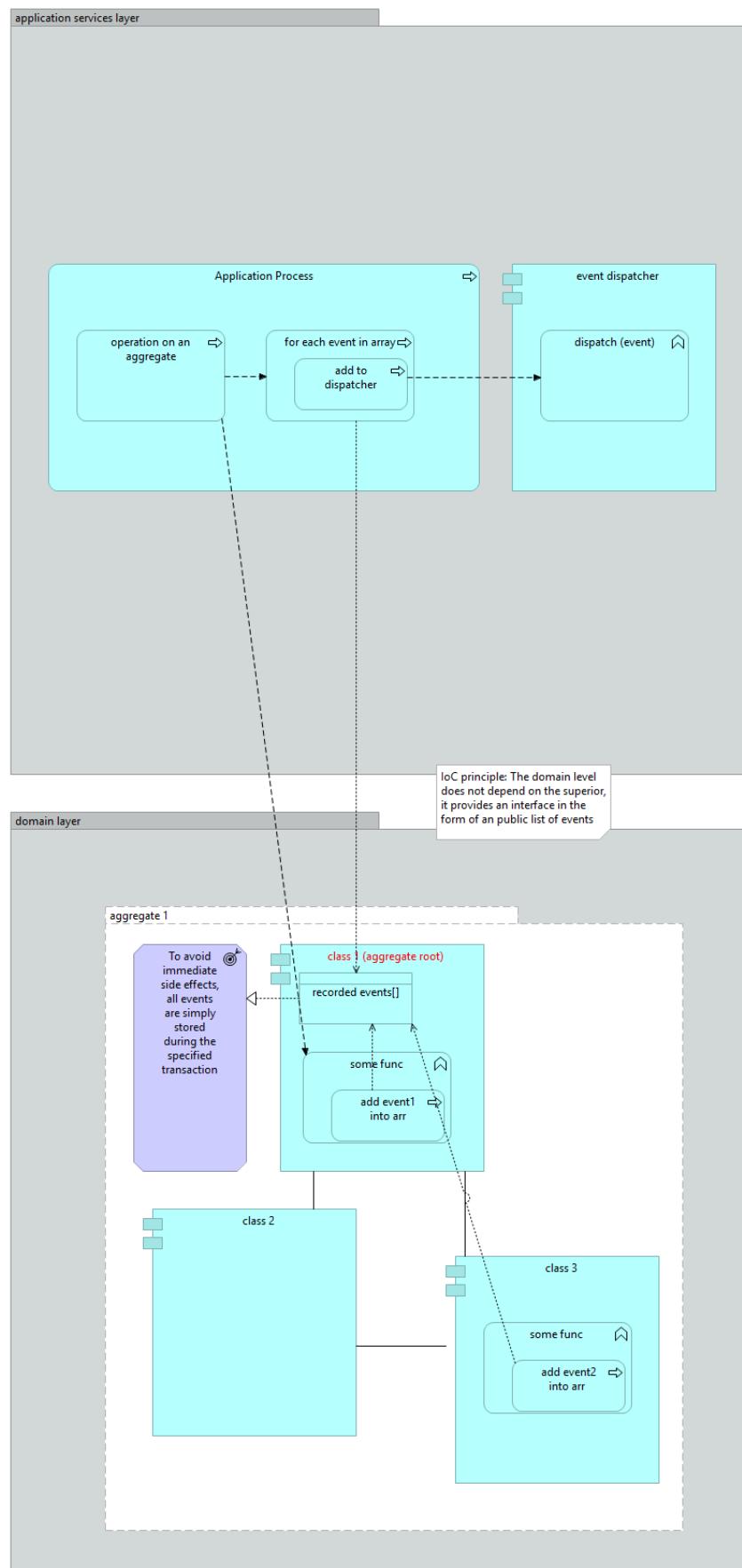
EVENT SOURCING



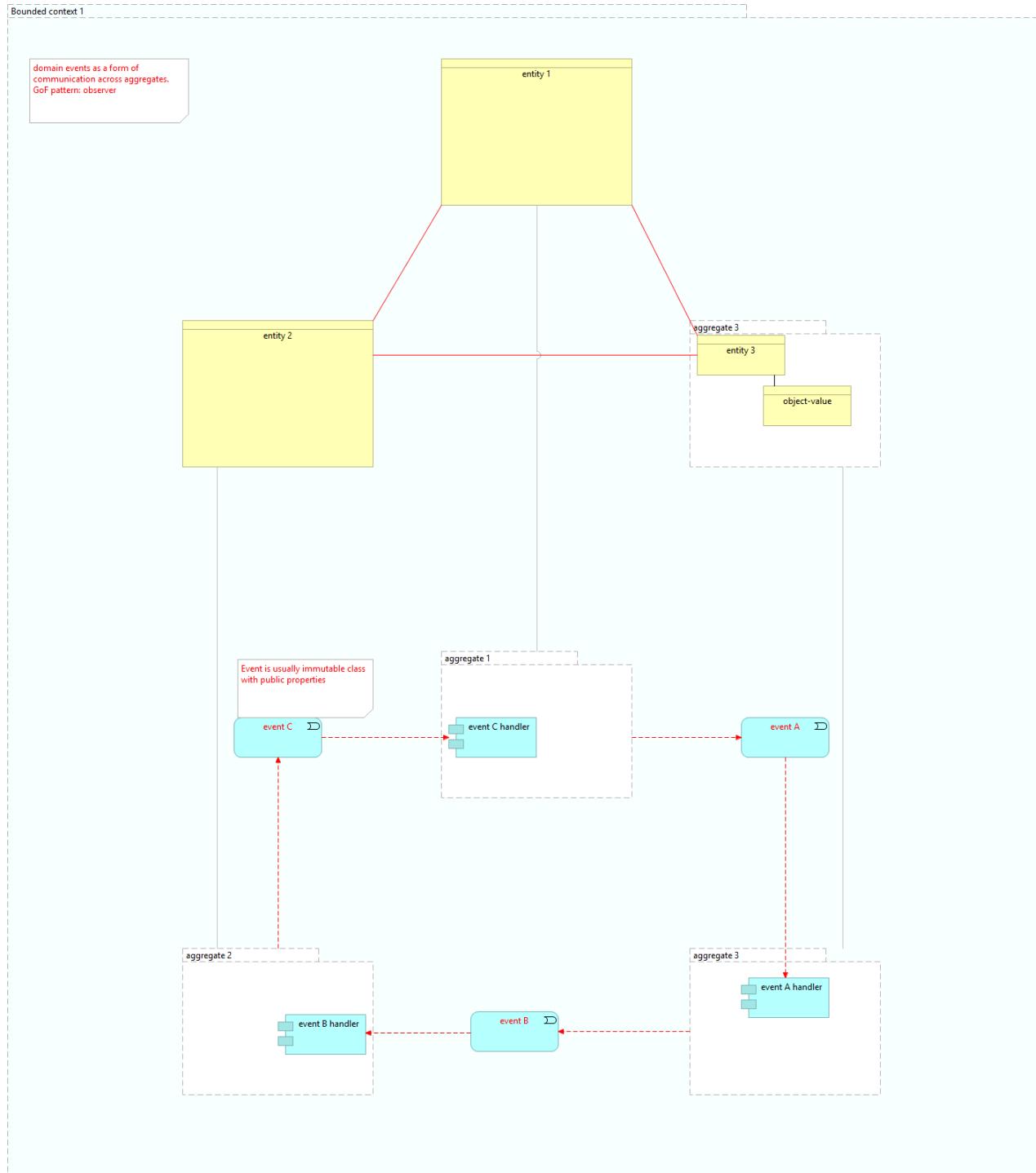
EVENT PROCESSOR



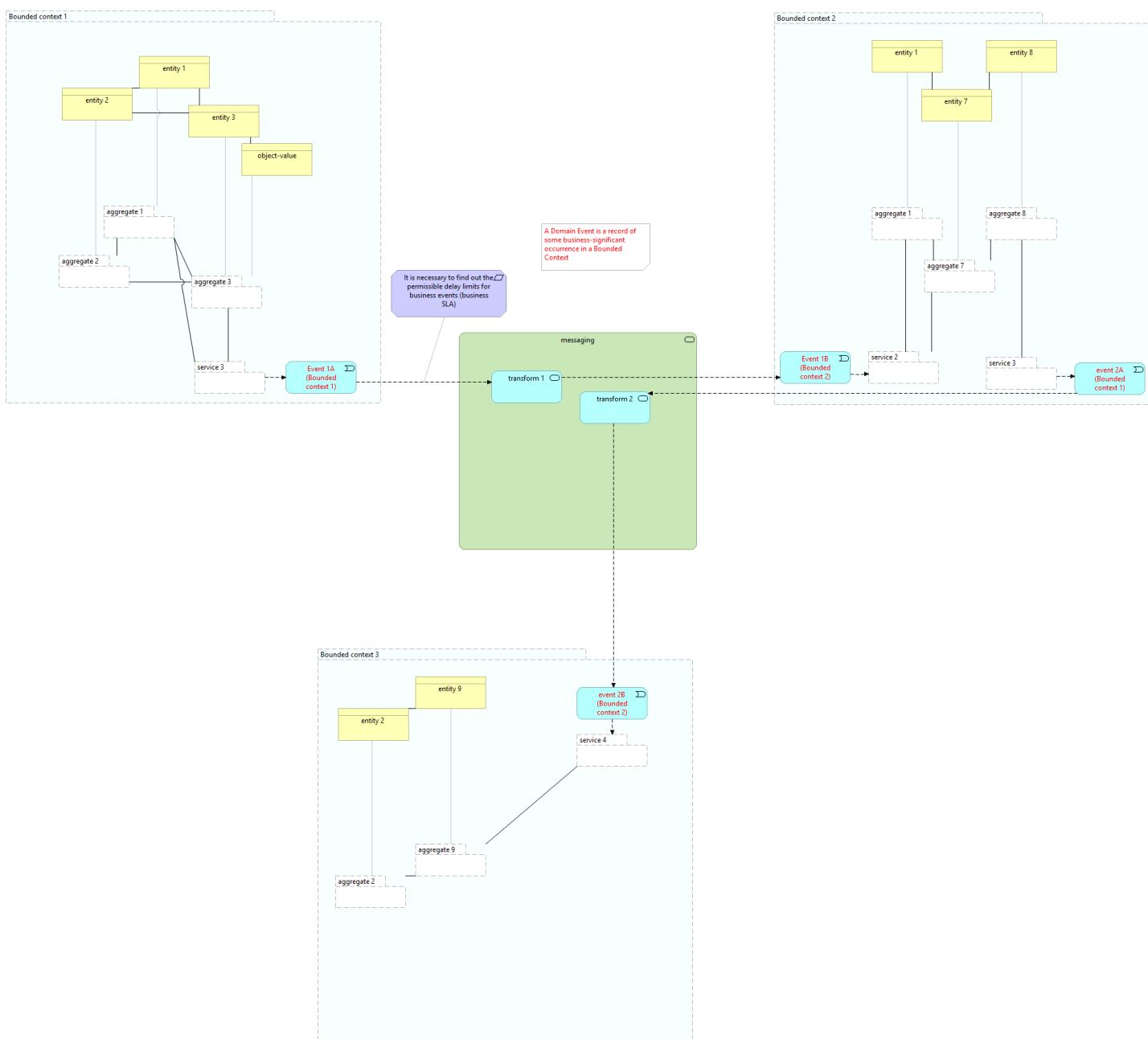
EVENT DISPATCHER



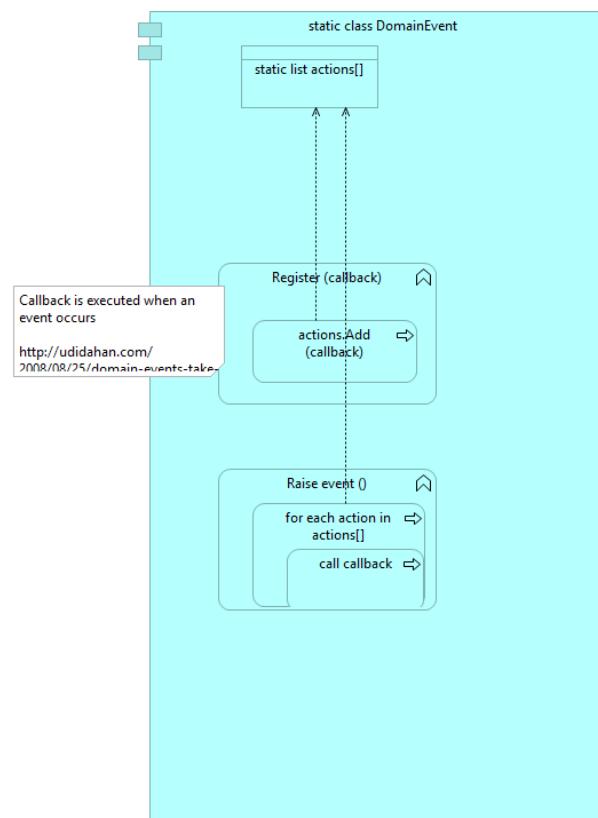
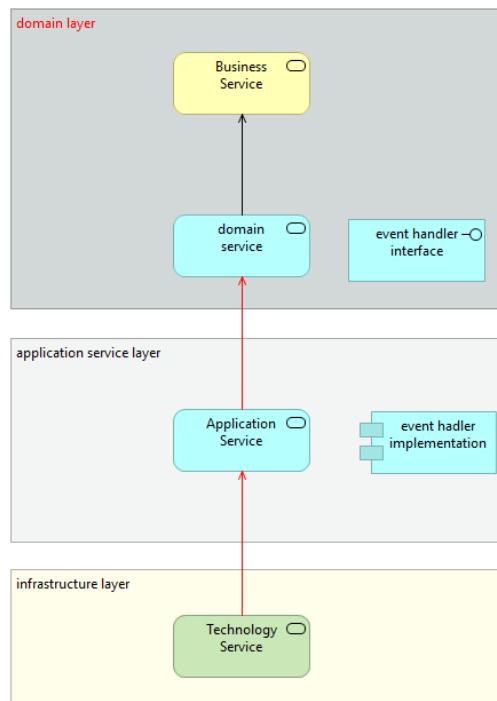
INTERNAL DOMAIN EVENTS



EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS

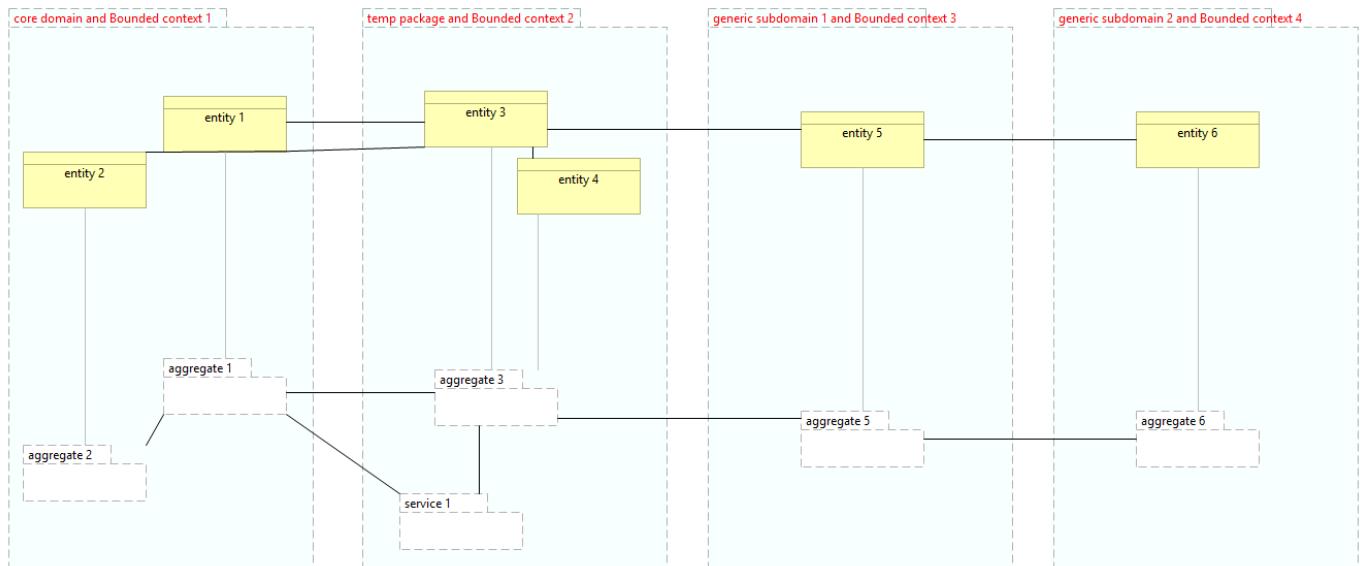


STATIC DOMAIN EVENTS CLASS

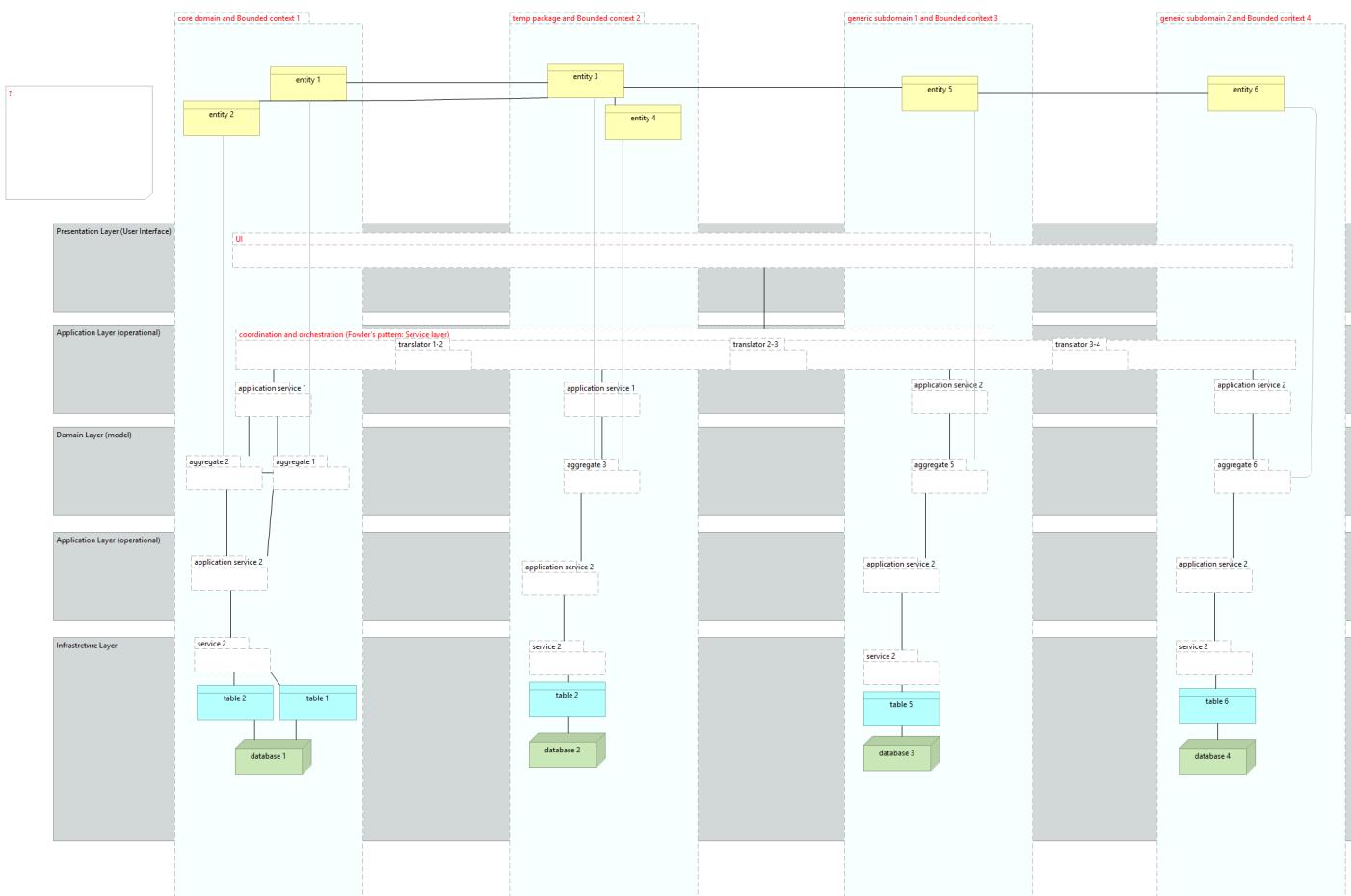


ONE SUBDOMAIN PER BOUNDED CONTEXT

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

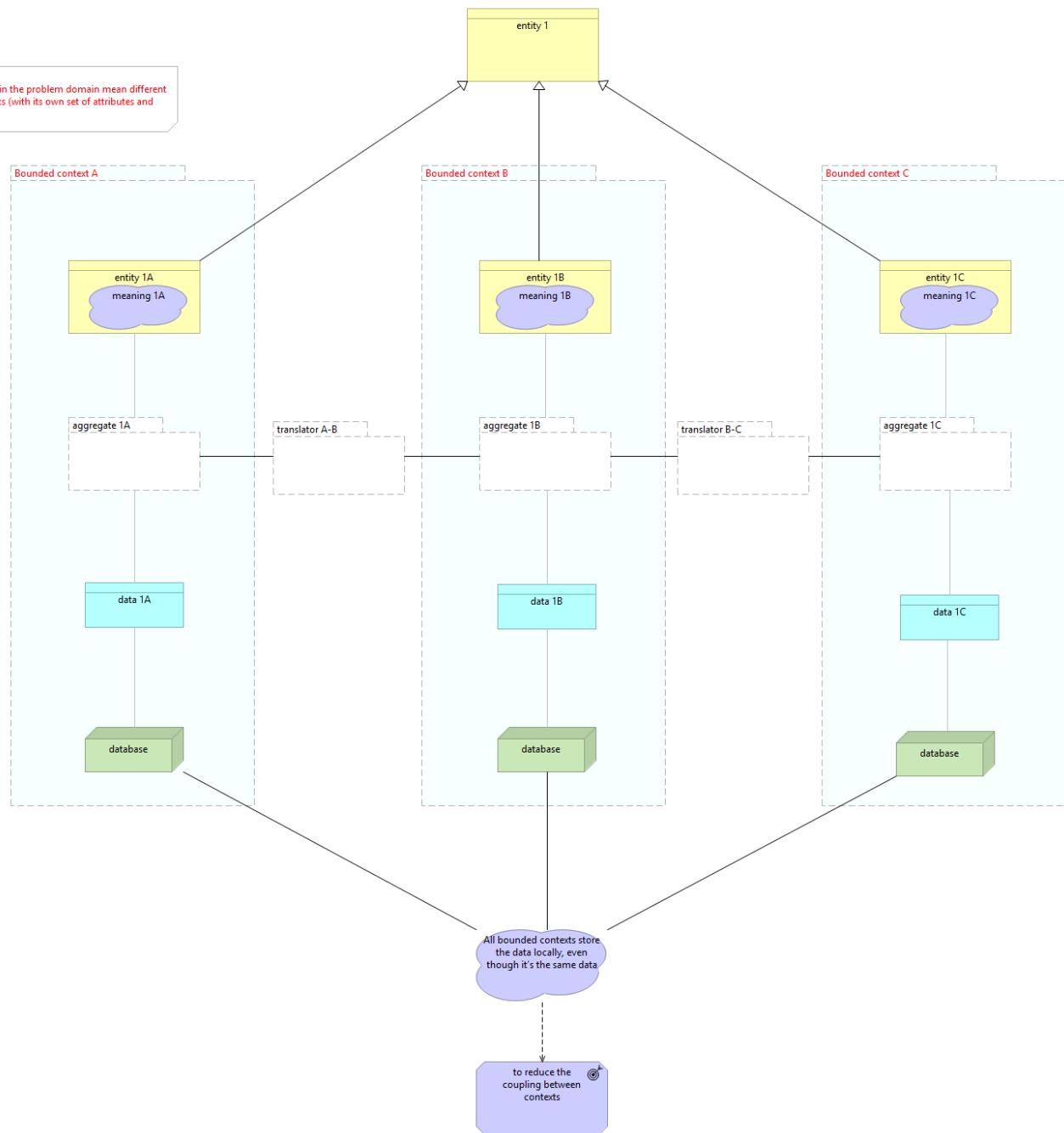


THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS

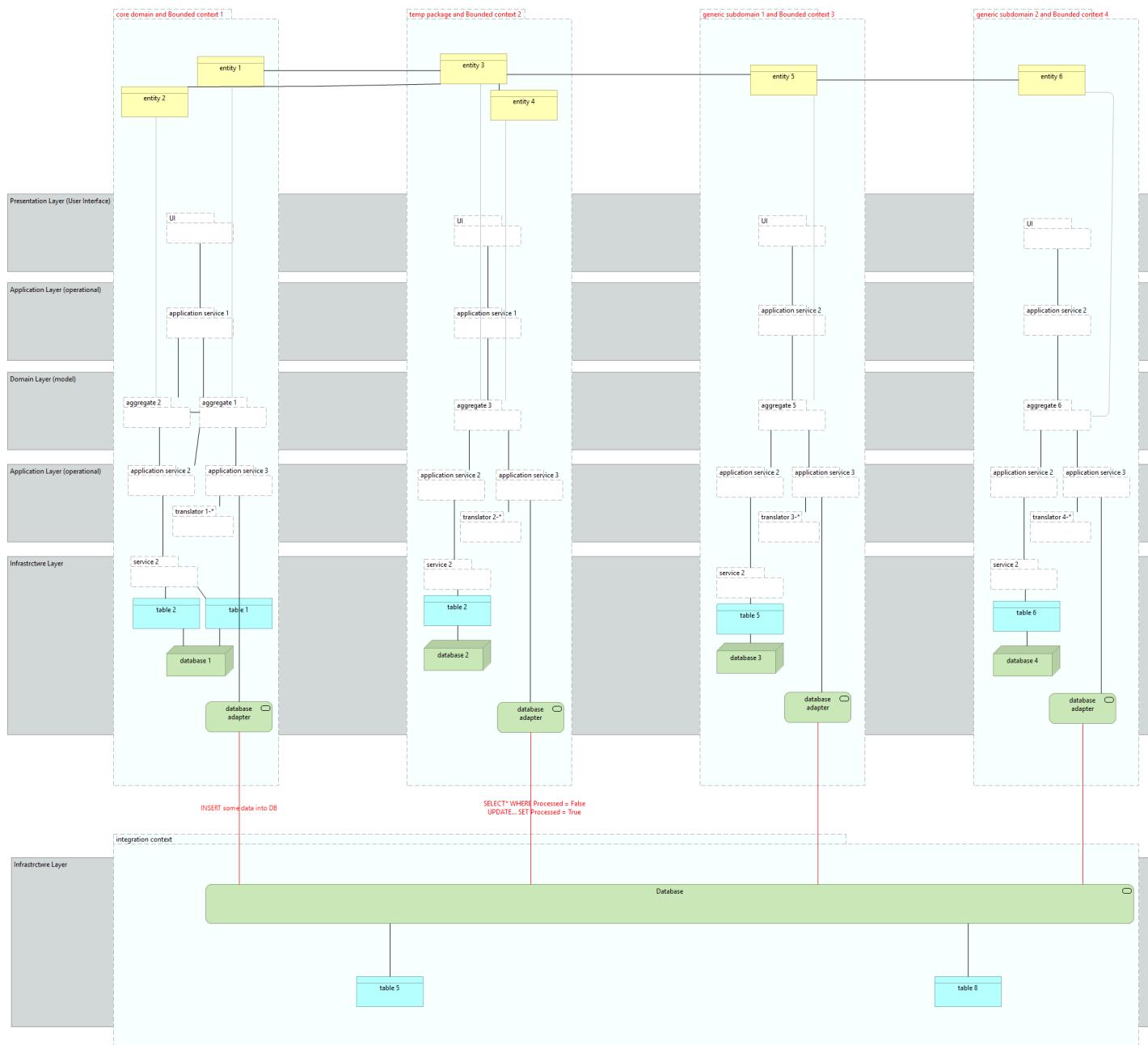


THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS

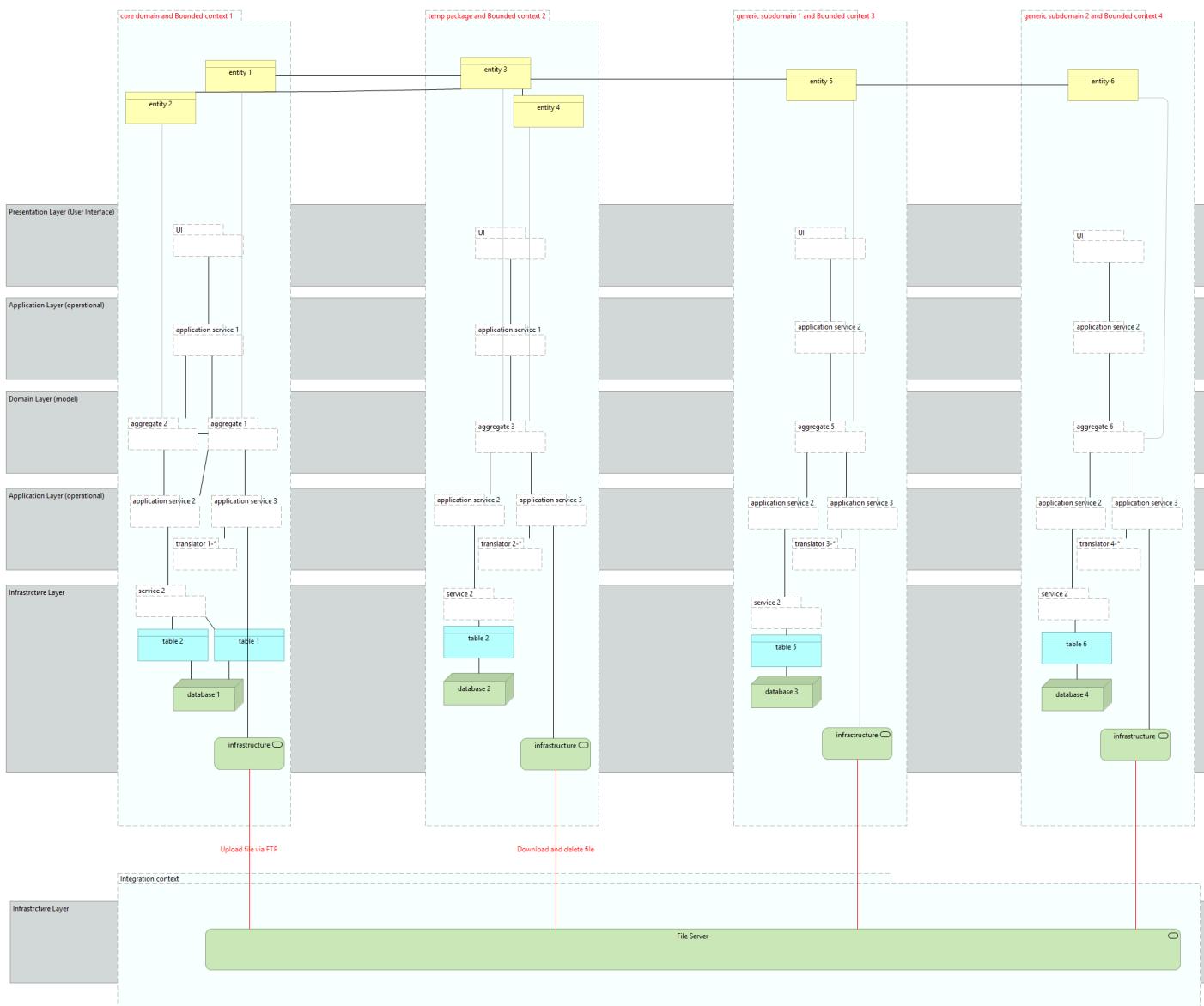
Example
The same physical entity in the problem domain mean different things in different contexts (with its own set of attributes and aspects of behavior).



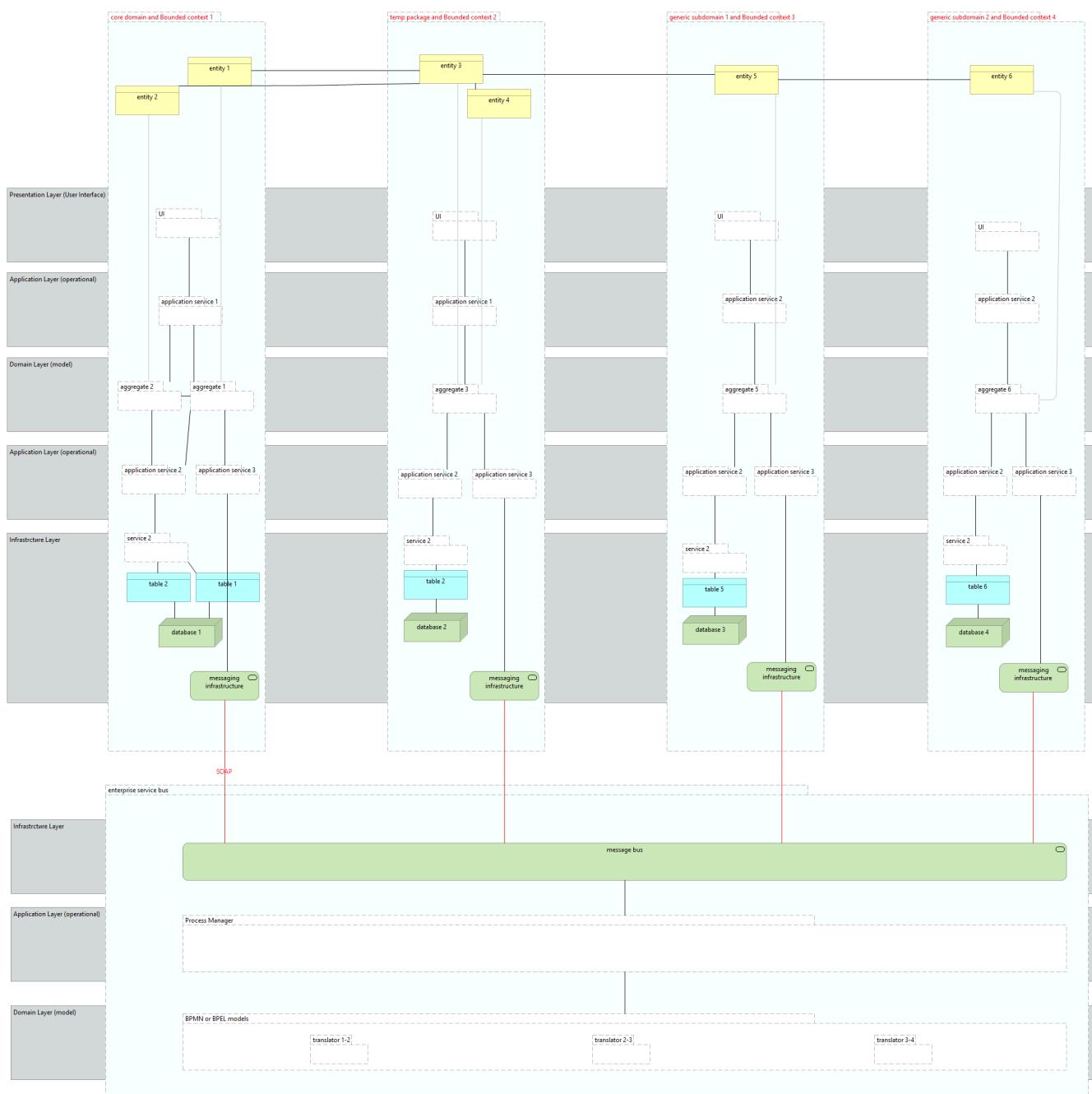
INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE



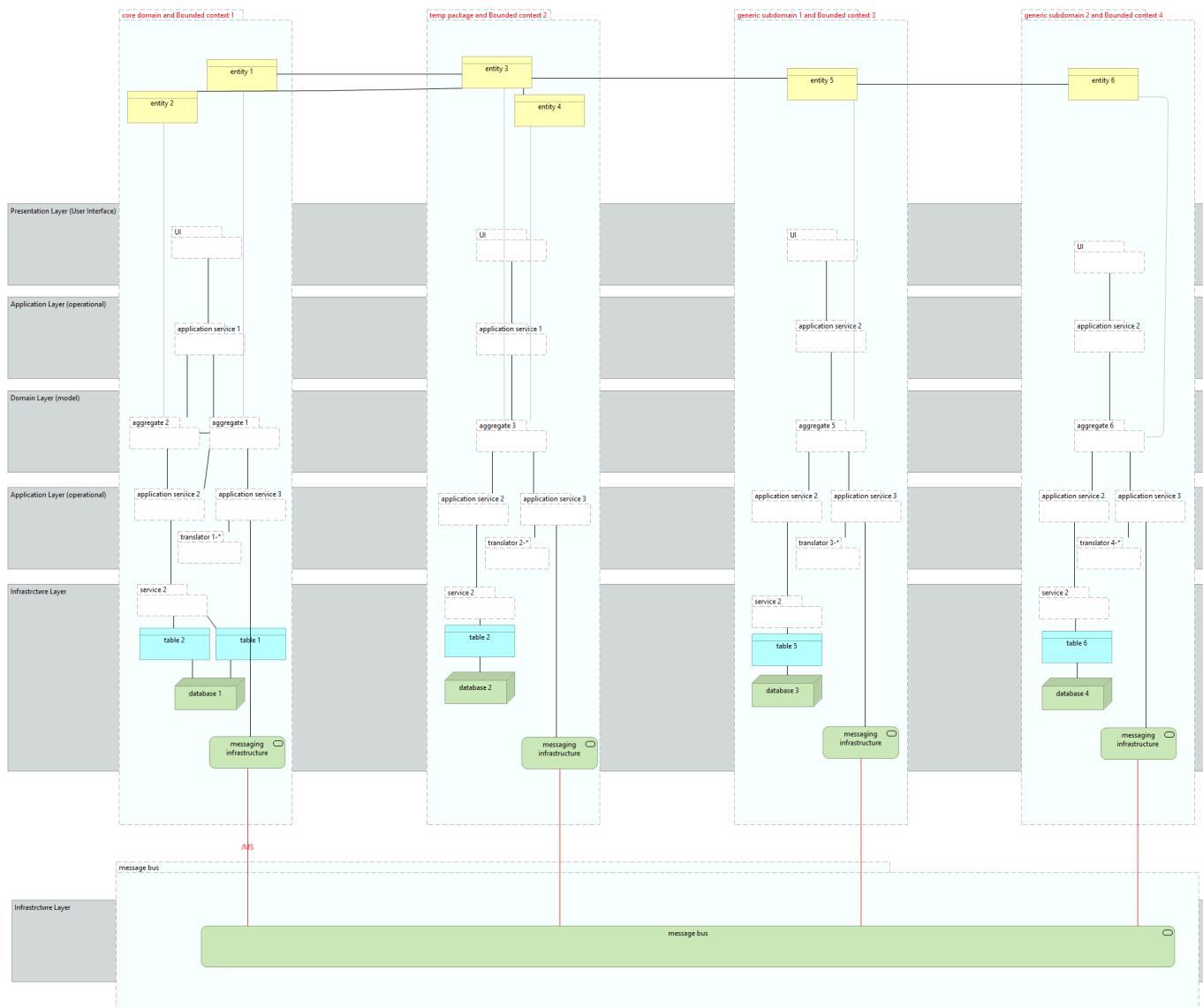
INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES



INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS



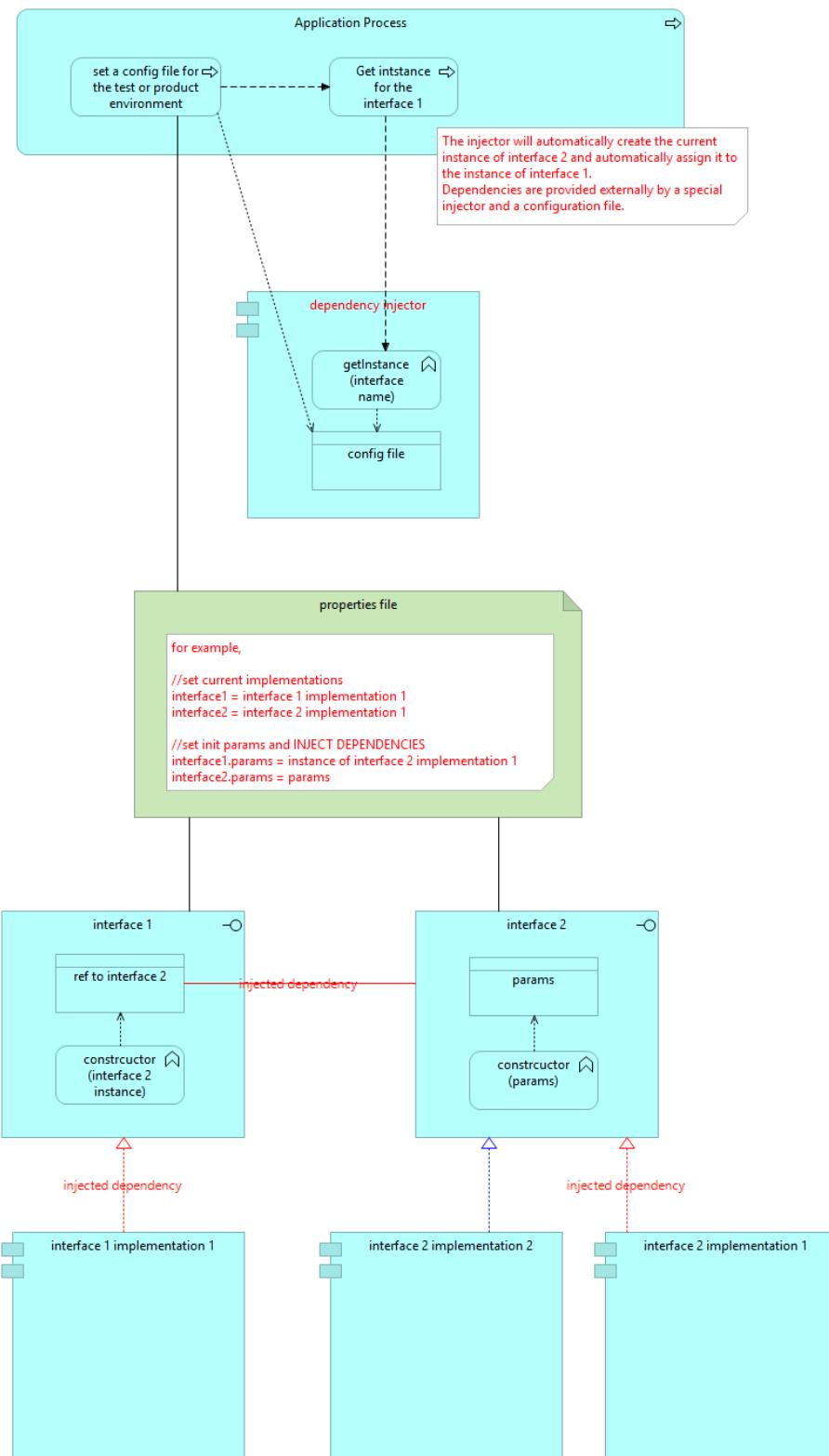
INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE



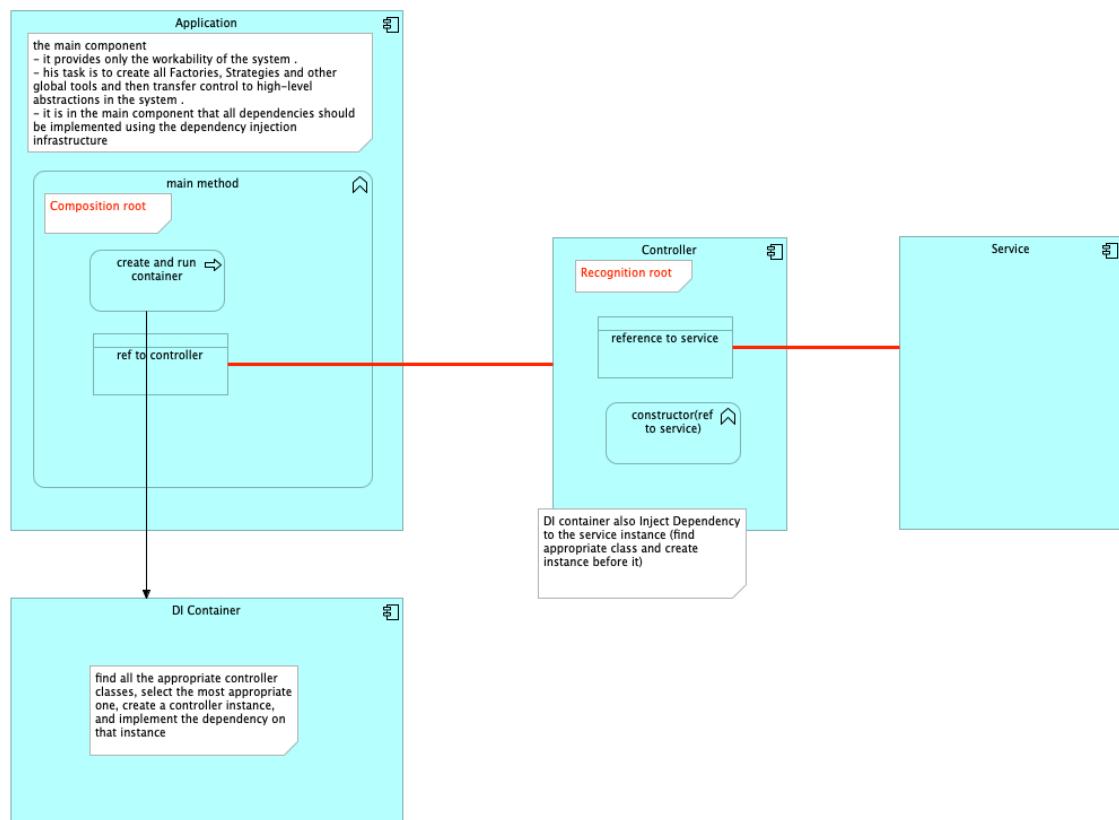
DEPENDENCY INJECTION

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

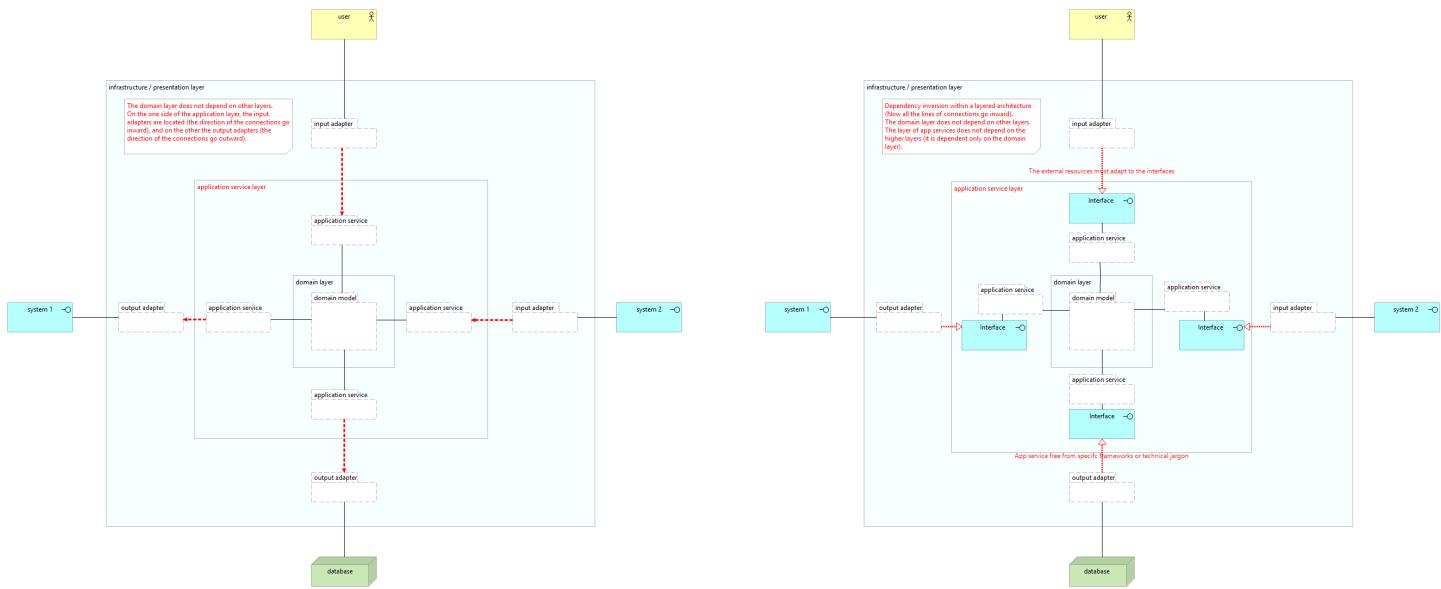
- inject the implementation into the application class
- removing the dependency from the application class to the plugin implementation



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

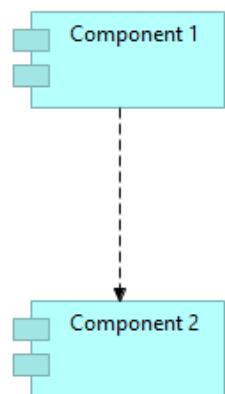


INVERSION OF CONTROL

IoC

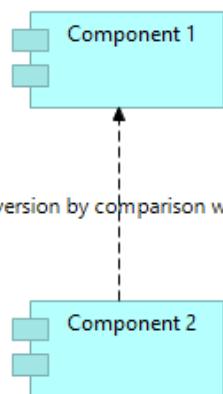
Inversion of control is about who initiates messages.

Variant 1



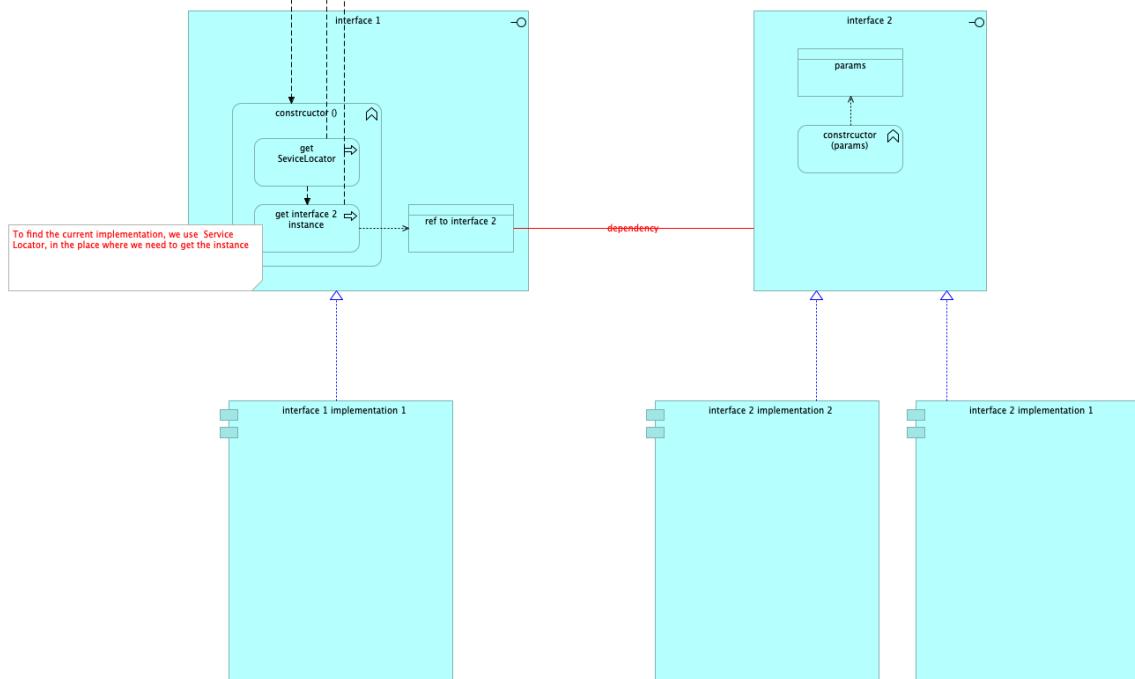
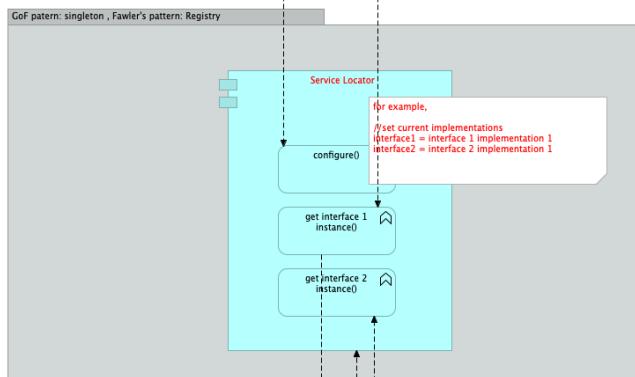
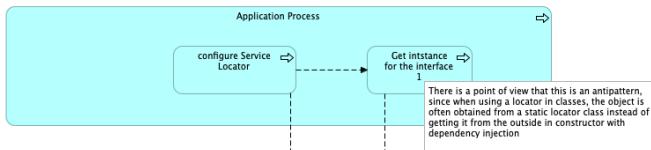
Variant 2

Here the control inversion by comparison with the previous case



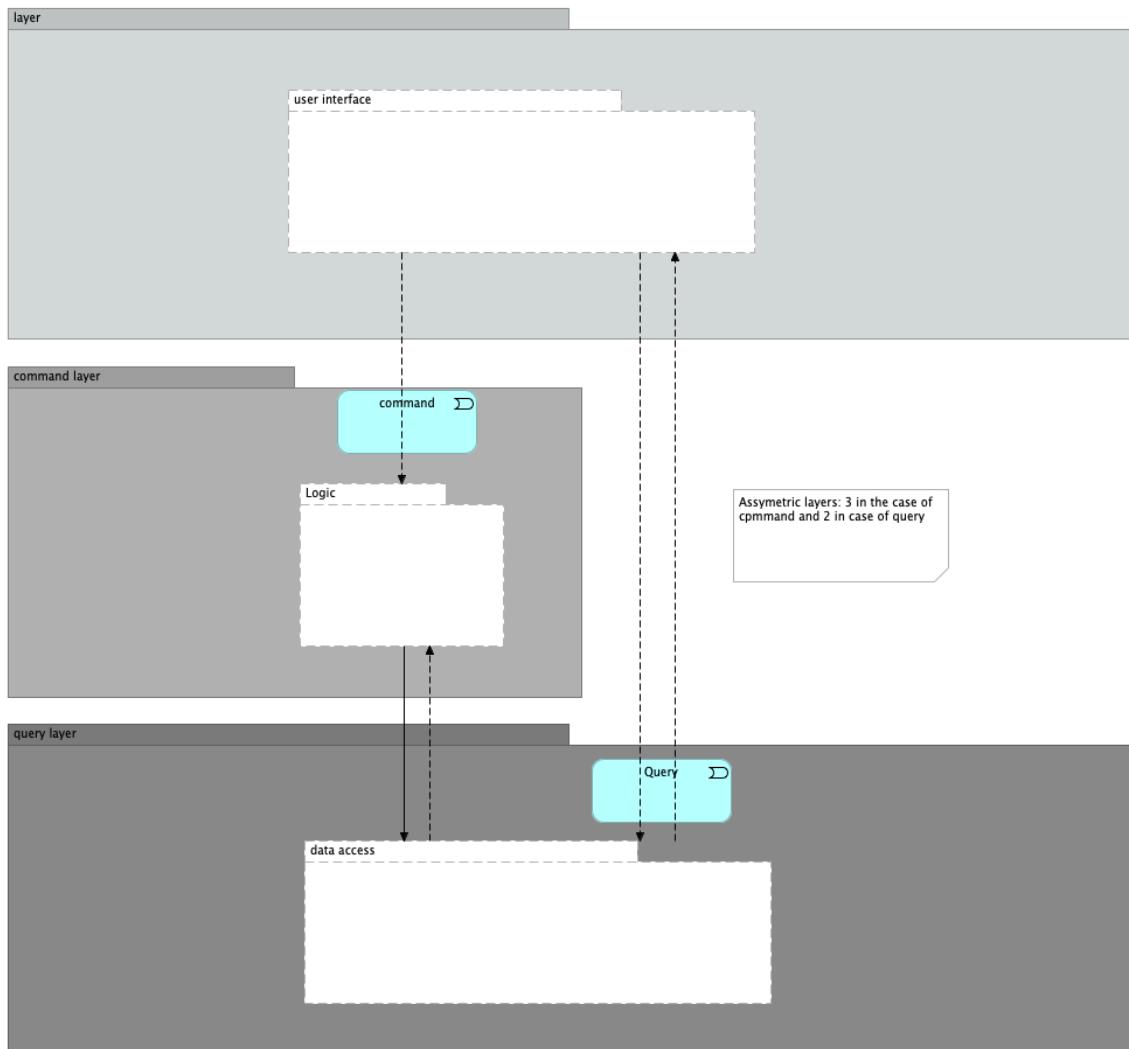
SERVICE LOCATOR

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



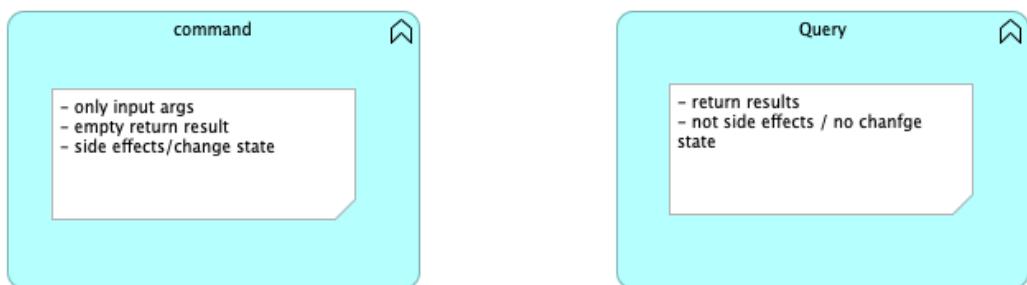
CQRS

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

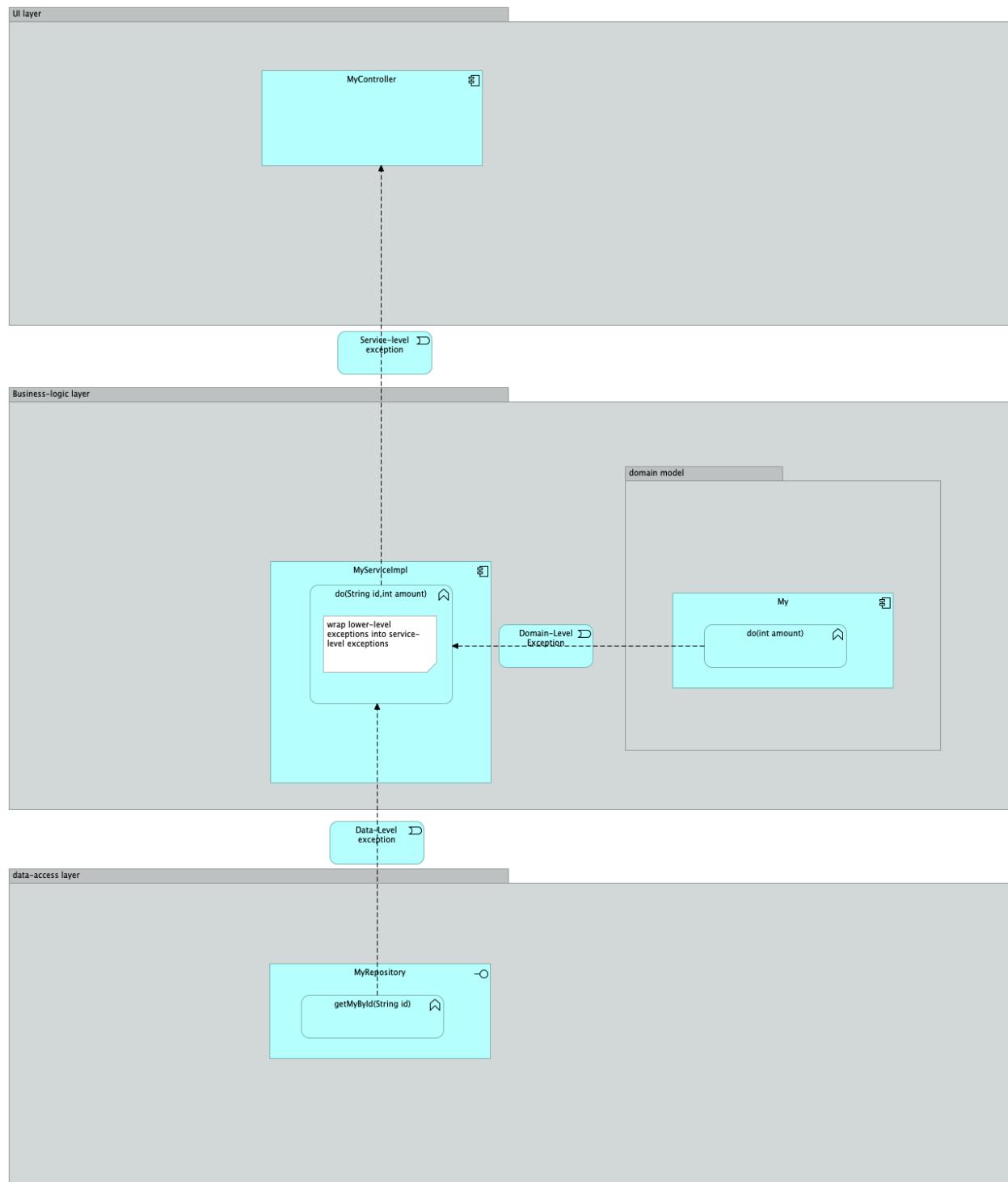


CQS

CQS Command/Query Separation

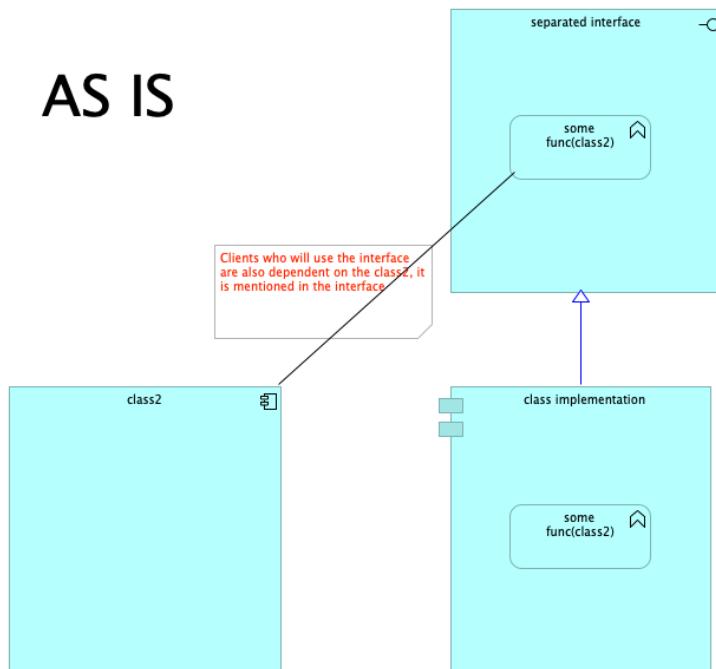


WRAP LOW-LEVEL EXCEPTIONS

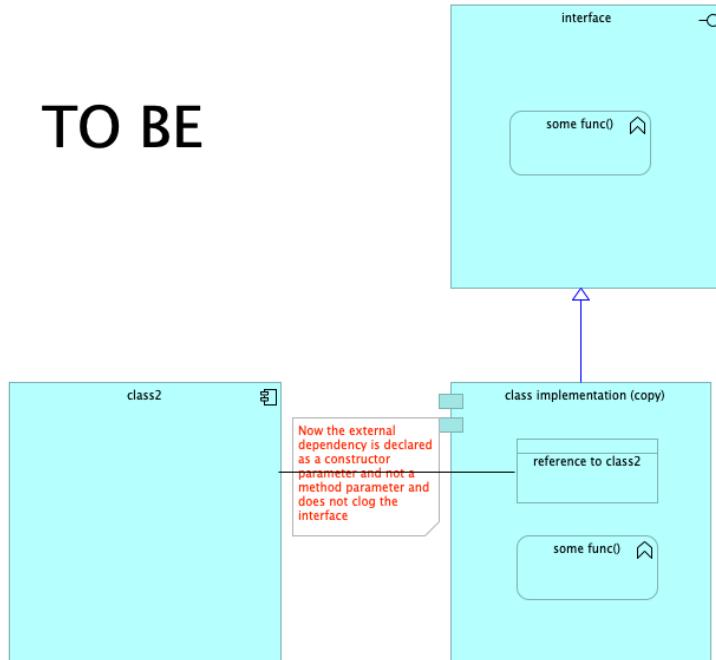


EXTRACT DEPENDENCY FROM INTERFACE TO CONSTRUCTOR

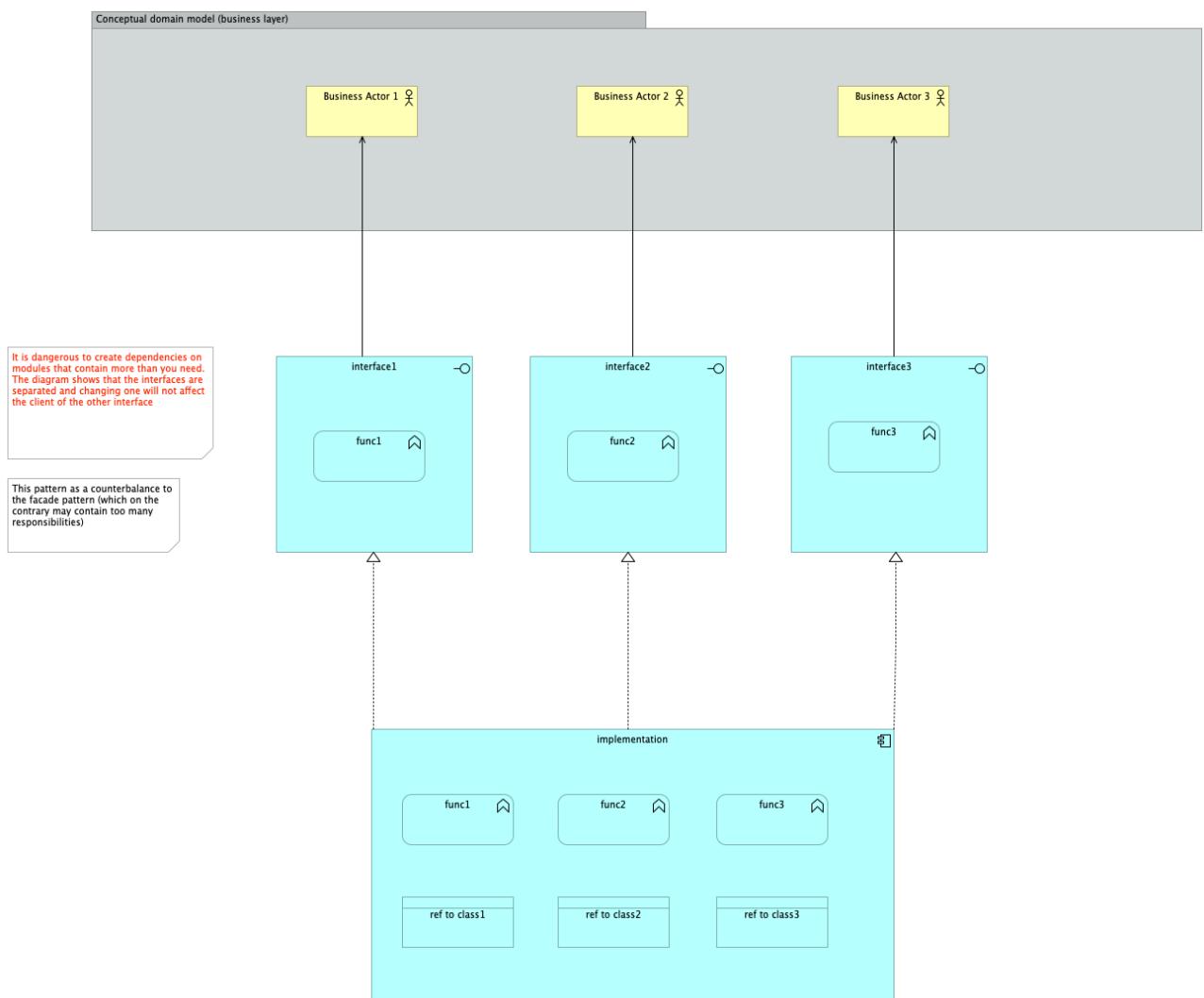
AS IS



TO BE

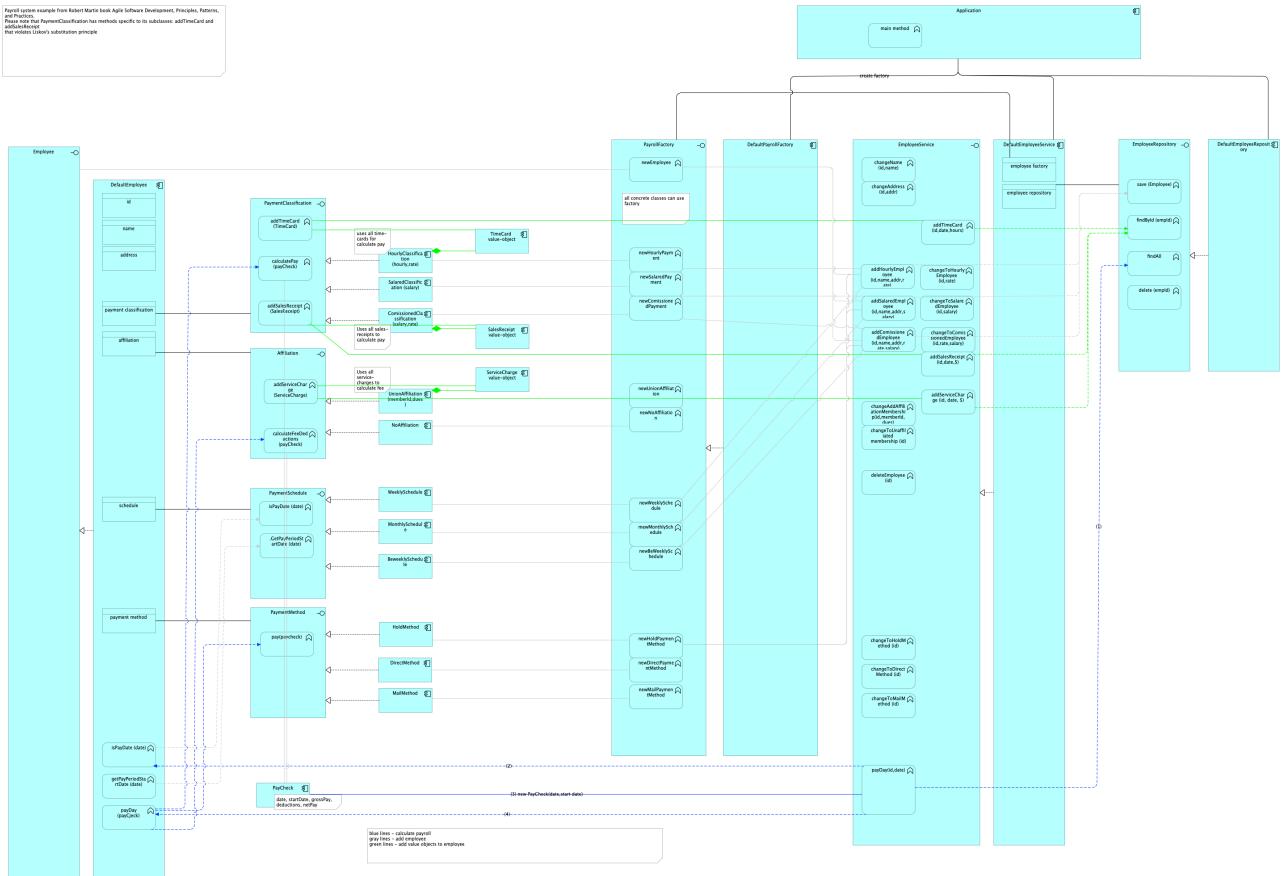


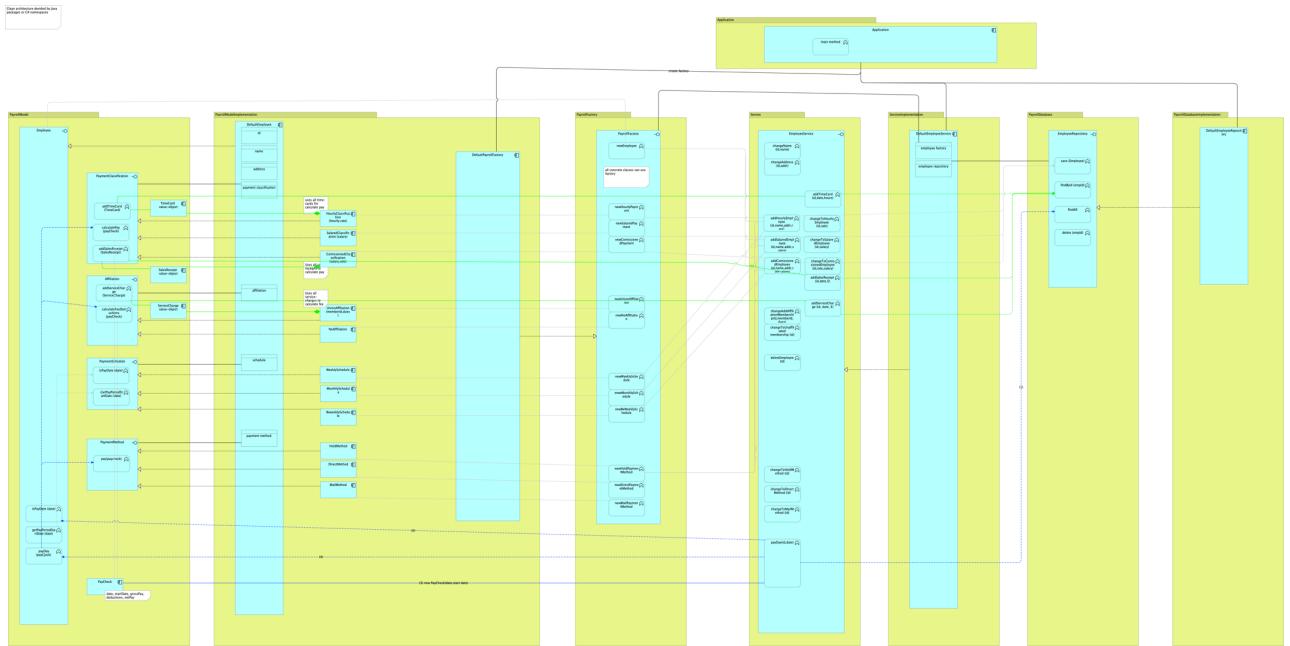
INTERFACE SEGREGATION

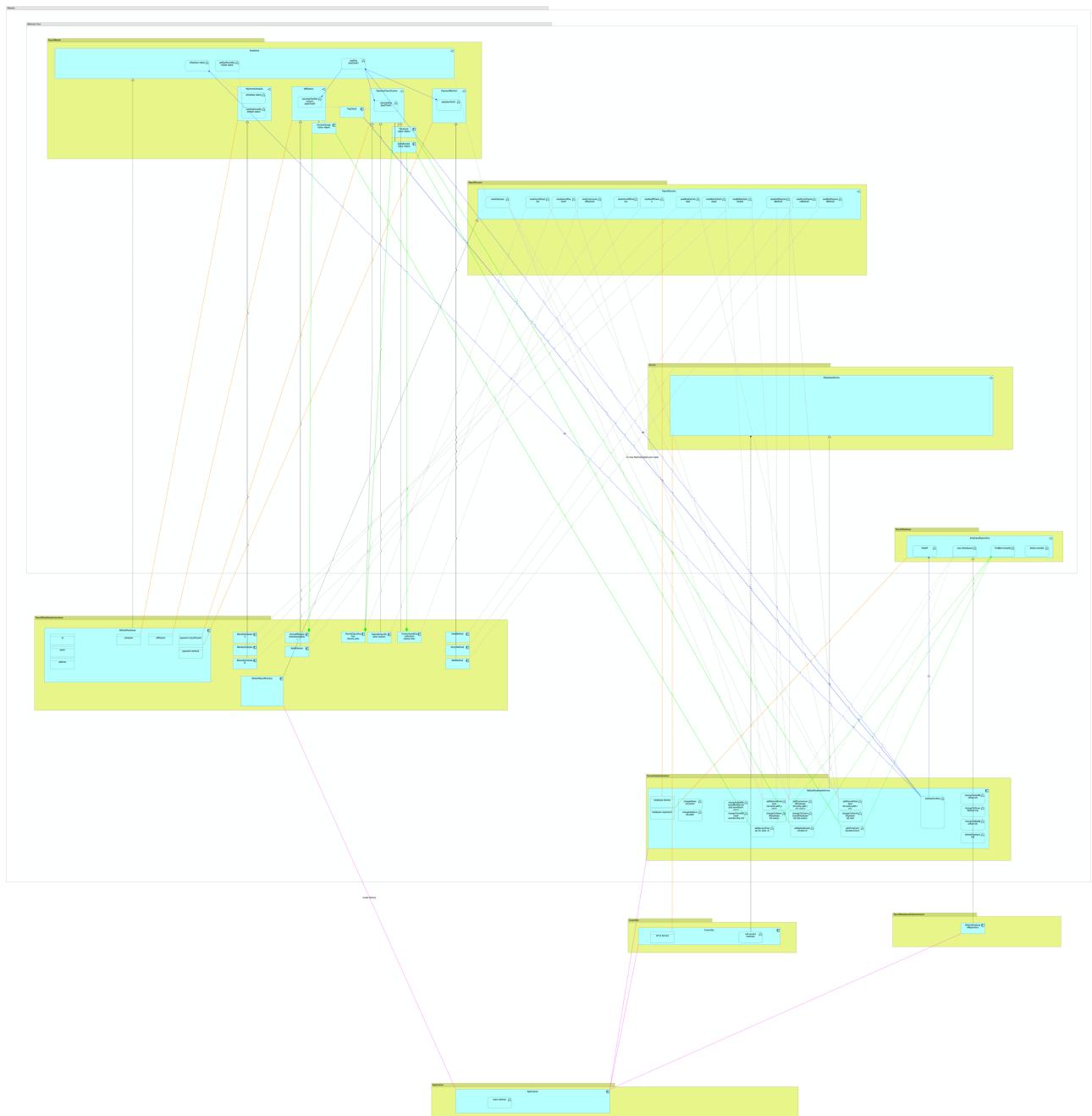


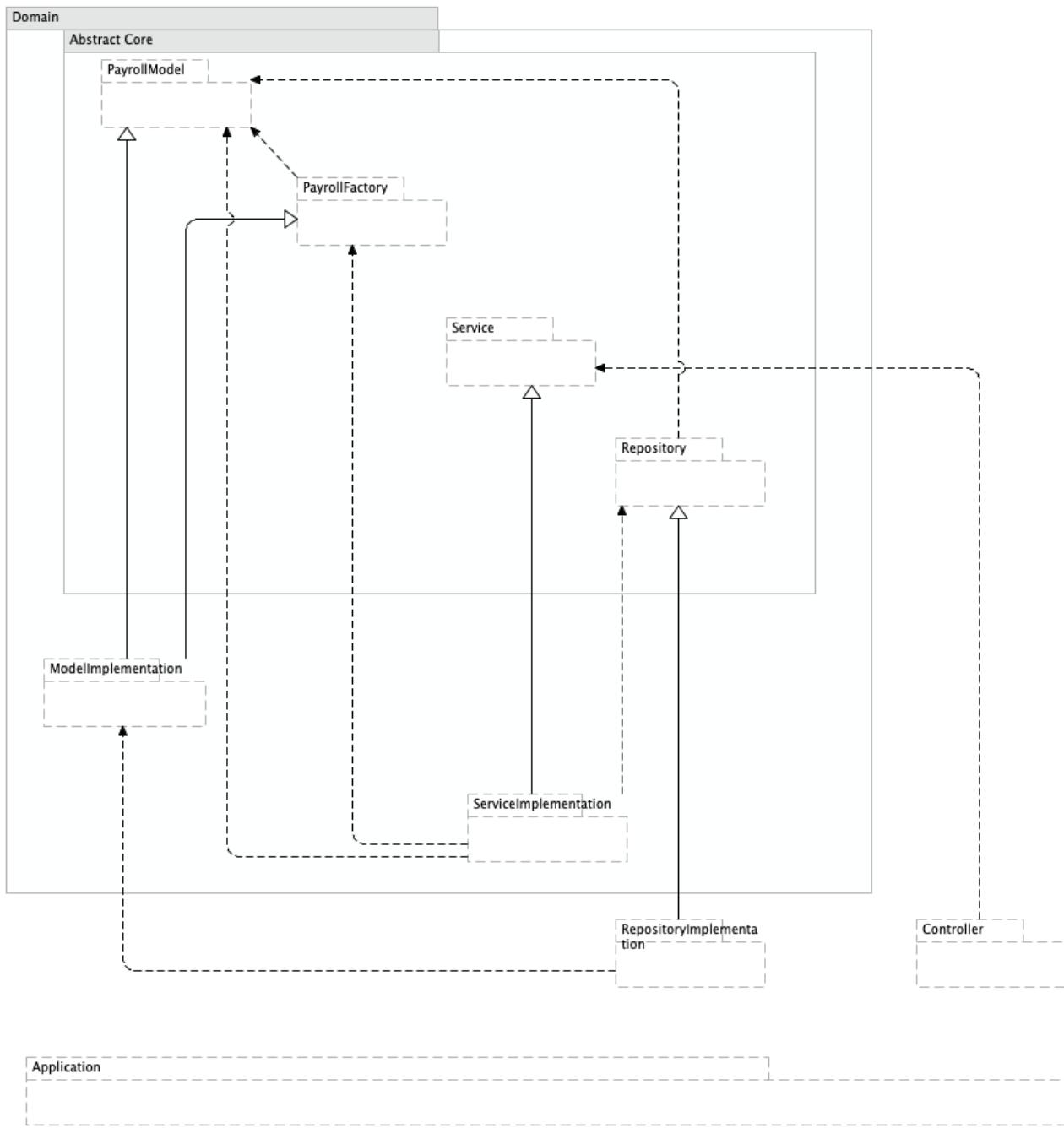
CLEAN ARCHITECTURE

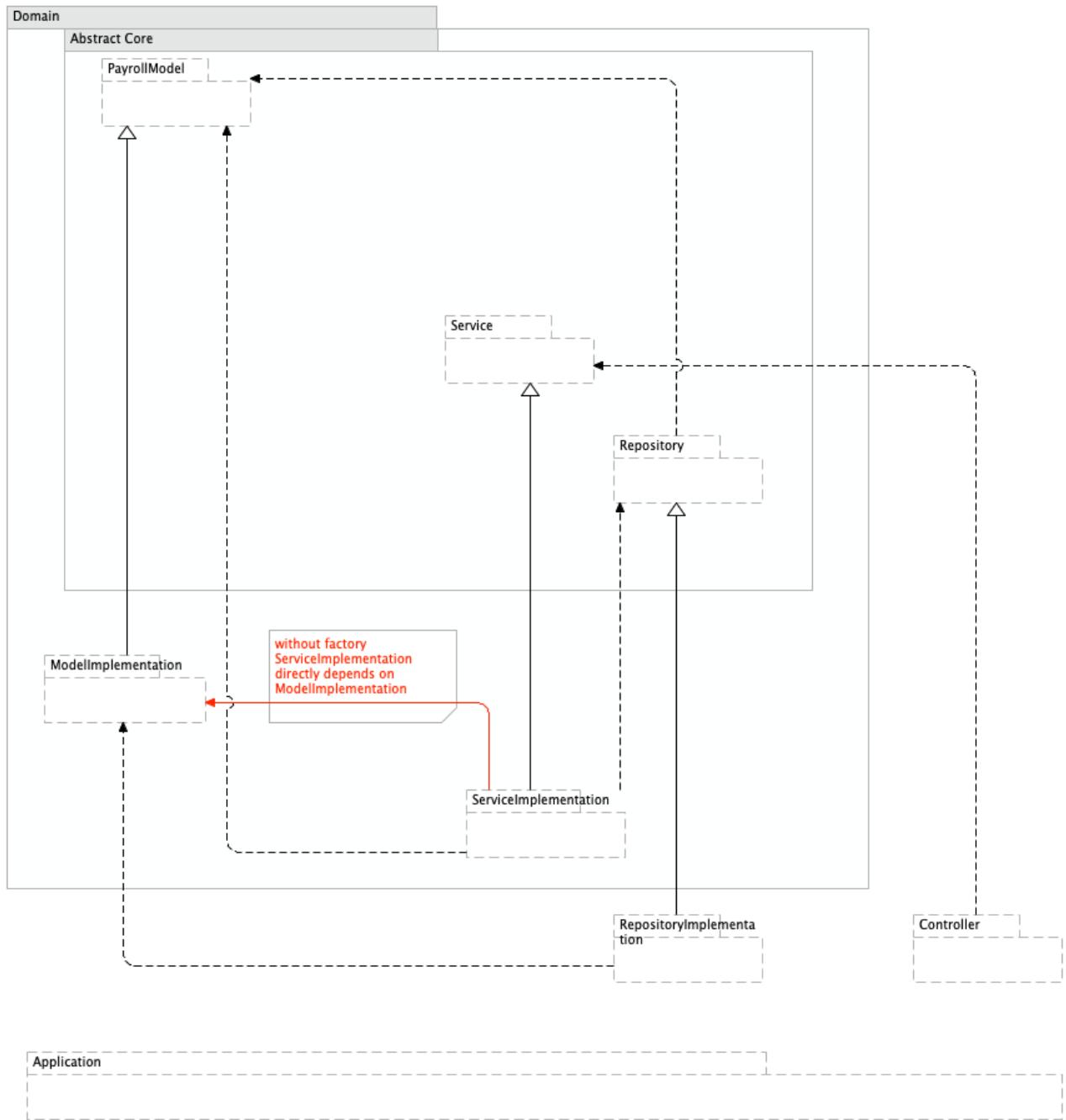
Payment example from Robert Martin's Agile Software Development, Principles, Patterns, and Practices. Note that the PaymentCalculator has methods specific to its subclasses: addTimeCard and addCheck.

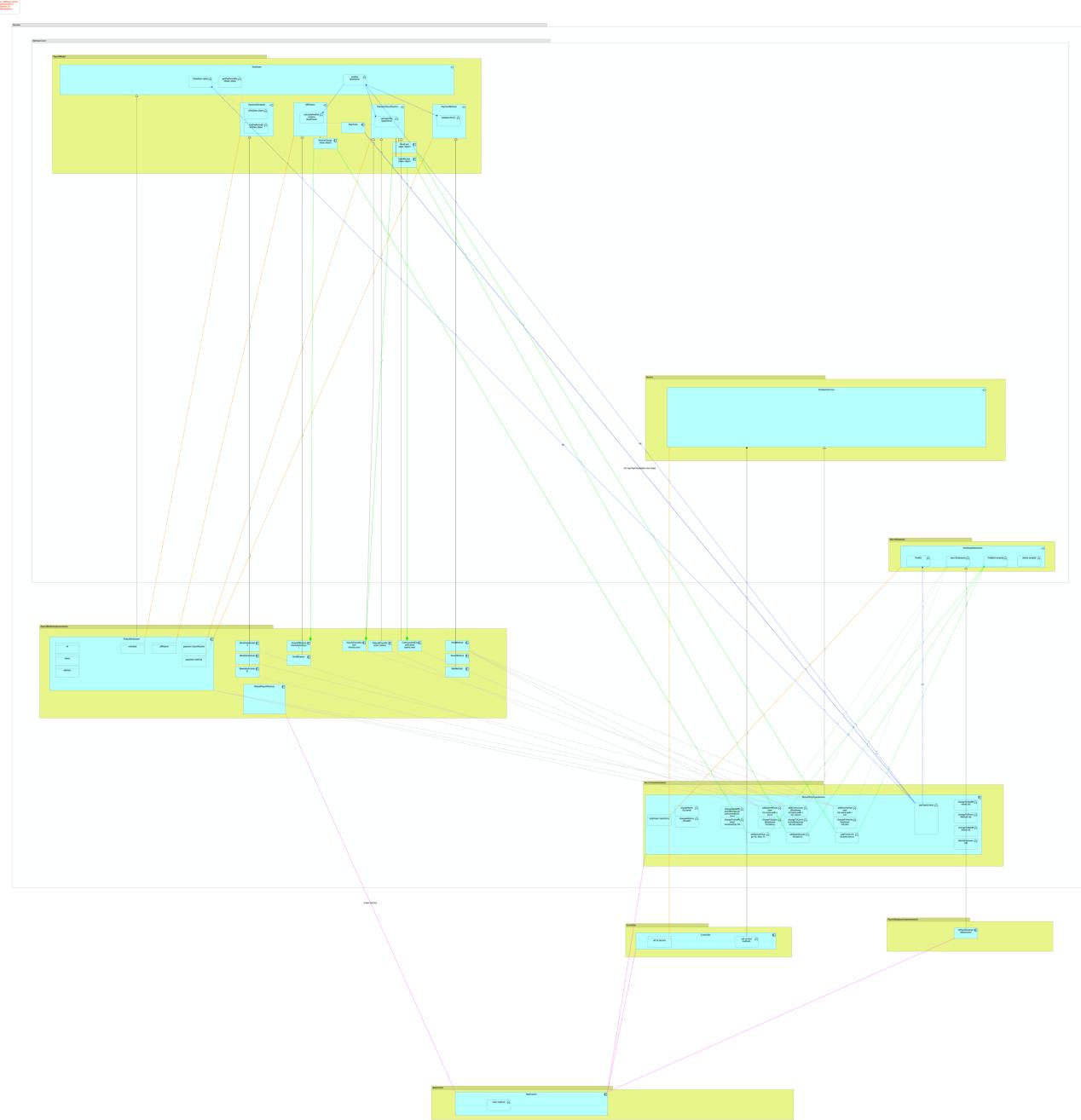














SOME ARCHITECTURAL DIAGRAMS
ARE CONSTRUCTED USING
STRUCTURE101 STUDIO,
COURTESY OF STRUCTURE101

structure101



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

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