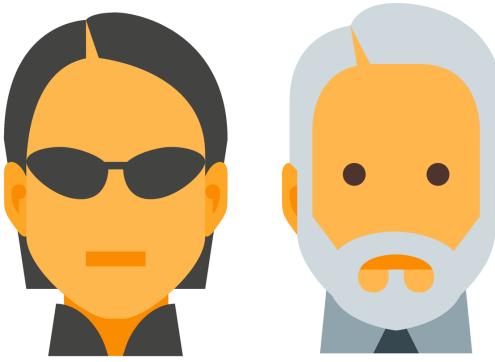


# 155 PATTERNS



## 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

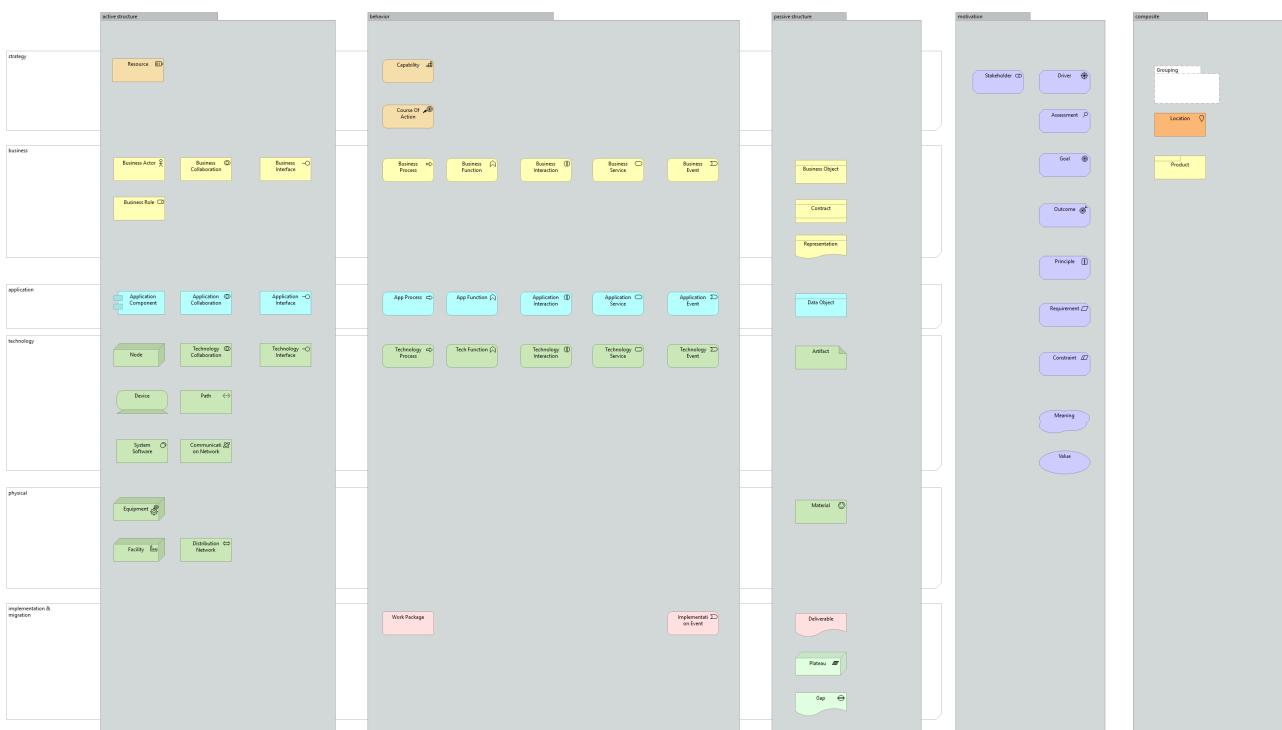
Tackling Complexity in the Heart of Software.

ERIC EVANS

# USED NOTATION

ARCHIMATE METAMODEL

The Open Group



DESIGN PATTERNS	11
CREATIONAL PATTERNS	12
ABSTRACT FACTORY	13
BUILDER	16
FACTORY METHOD	18
PROTOTYPE	20
SINGLETON	22
STRUCTURAL PATTERNS	24
ADAPTER OF CLASS	25
ADAPTER OF OBJECT	27
BRIDGE	30
COMPOSITE	32
DECORATOR	34
FAÇADE	36
FLYWEIGHT	39
FLYWEIGHT + COMPOSITE	41
PROXY	42
BEHAVIORAL PATTERNS	44
CHAIN OF RESPONSIBILITY	45
COMMAND	47
INTERPRETER	49
ITERATOR	51
MEDIATOR	53
MEMENTO	55
OBSERVER	57
STATE	59
STRATEGY	61
TEMPLATE METHOD	63
VISITOR	66
ENTERPRISE PATTERNS	68
BUSINESS LOGIC	69
DOMAIN MODEL	70
SERVICE LAYER	71
TRANSACTION SCRIPT	72
TABLE MODULE	73
DATA SOURCES	74
ACTIVE RECORD	75
DATA MAPPER	76
ROW DATA GATEWAY	77
TABLE DATA GATEWAY	78
MODELING BEHAVIOR	79

IDENTITY MAP	80
LAZY LOAD	81
UNIT OF WORK.	82
MODELING STRUCTURE HIERARCHY	83
CLASS TABLE INHERITANCE	84
CONCRETE TABLE INHERITANCE	85
INHERITANCE MAPPERS	86
SINGLE TABLE INHERITANCE	87
MODELING STRUCTURE RELATIONS	88
ASSOCIATION TABLE MAPPING	89
DEPENDENT MAPPING	90
EMBEDDED VALUE	91
FOREIGN KEY MAPPING	92
IDENTITY FIELD	93
SERIALIZED LOB	94
METADATA	95
METADATA MAPPING	96
QUERY OBJECT	97
REPOSITORY	98
WEB REPRESENTATION CONTROLLER	99
MODEL VIEW CONTROLLER	100
APPLICATION CONTROLLER	101
FRONT CONTROLLER	102
PAGE CONTROLLER	103
WEB REPRESENTATION VIEW	104
TEMPLATE VIEW	105
TRANSFORM VIEW	106
TWO STEP VIEW	107
DISTRIBUTED PROCESSING	108
DATA TRANSFER OBJECT	109
REMOTE FAÇADE	110
PARALLEL PROCESSING	111
COARSE-GRAINED LOCK	112
IMPLICIT LOCK	113
OPTIMISTIC OFFLINE LOCK	114
PESSIMISTIC OFFLINE LOCK	115
SESSION STATE	116
CLIENT SESSION STATE	117
DATABASE SESSION STATE	118
SERVER SESSION STATE	119
COMMON PATTERNS	120
GATEWAY	121
LAYER SUPERTYPE	122

MAPPER	123
MONEY	124
PLUGIN	125
RECORD SET	126
REGISTRY	127
SEPARATED INTERFACE	128
SERVICE STUB	129
SPECIAL CASE	130
VALUE OBJECT	131
ANALYSIS PATTERNS	132
ACCOUNTABILITY	134
PARTY	135
ACCOUNTABILITY	136
ORGANIZATION HIERARCHIES	138
ORGANIZATION STRUCTURE	139
ACCOUNTABILITY KNOWLEDGE LEVEL	140
PARTY TYPE GENERALIZATIONS	141
HIERARCHIC ACCOUNTABILITY	142
OPERATING SCOPES	143
POST	144
OBSERVATIONS AND MEASUREMENTS	145
QUANTITY	146
CONVERSION RATIO	147
OBSERVATIONS AND MEASUREMENTS	148
COMPOUND UNITS	149
MEASUREMENT	150
OBSERVATION	151
SUBTYPING OBSERVATION CONCEPTS	152
PROTOCOL	153
DUAL TIME RECORD	154
REJECTED OBSERVATION	155
ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION	156
ASSOCIATED OBSERVATION	157
PROCESS OF OBSERVATION	158
OBSERVATIONS FOR CORPORATE FINANCE	159
ENTERPRISE SEGMENT	160
MEASUREMENT PROTOCOL	161
RANGE	162
OBSERVATIONS FOR CORPORATE FINANCE	163
PHENOMENON WITH RANGE	166
REFERRING TO OBJECTS	167
NAME	168
IDENTIFICATION SCHEME	169

OBJECT MERGE	170
OBJECT EQUIVALENCE	171
REFERRING TO OBJECTS	172
INVENTORY AND ACCOUNTING	173
ACCOUNT	174
TRANSACTIONS	175
SUMMARY ACCOUNT	176
MEMO ACCOUNT	177
POSTING RULES	178
INVENTORY AND ACCOUNTING	179
INDIVIDUAL INSTANCE METHOD	180
POSTING RULE EXECUTION	181
POSTING RULES FOR MANY ACCOUNTS	182
CHOOSING ENTRIES	183
ACCOUNTING PRACTICE	184
SOURCES OF AN ENTRY	185
BALANCE SHEET AND INCOME STATEMENT	186
CORRESPONDING ACCOUNT	187
SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)	188
SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)	189
BOOKING ENTRIES TO MULTIPLE ACCOUNTS	190
PLANNING	191
PROPOSED AND IMPLEMENTED ACTION	192
COMPLETED AND ABANDONED ACTIONS	193
SUSPENSION	194
PLAN	195
PROTOCOL	196
RESOURCE ALLOCATION	197
PLANNING	198
PLANNING (NO OUTCOME)	199
OUTCOME AND START FUNCTIONS	200
TRADING	201
CONTRACT	202
PORTFOLIO	203
QUOTE	204
SCENARIO	205
TRADING	206
DERIVATIVE CONTRACTS	207
FORWARD CONTRACTS	208
OPTIONS	209
PRODUCT	210
SUBTYPE STATE MACHINES	211
PARALLEL APPLICATION AND DOMAIN HIERARCHIES	212

DERIVATIVE CONTRACTS	213
TRADING PACKAGES	214
MULTIPLE ACCESS LEVELS TO A PACKAGE	215
MUTUAL VISIBILITY	216
TRADING PACKAGES	217
LAYERED ARCHITECTURE FOR INFORMATION SYSTEMS	218
TWO-TIER ARCHITECTURE	219
THREE-TIER ARCHITECTURE	220
PRESENTATION AND APPLICATION LOGIC	221
DATABASE INTERACTION	222
TYPE MODEL DESIGN	223
IMPLEMENTING ASSOCIATIONS	224
IMPLEMENTING GENERALIZATION	225
OBJECT CREATION	226
OBJECT DESTRUCTION	227
ENTRY POINT.	228
IMPLEMENTING CONSTRAINTS	229
DOMAIN DRIVEN DESIGN	230
MODEL AND STRUCTURAL ELEMENTS	231
MODEL-DRIVEN DESIGN	232
LAYERED ARCHITECTURE (ASYMMETRIC )	235
HEXAGONAL ARCHITECTURE (SYMMETRIC)	237
	238
COMPOSITE UI	241
ENTITIES	242
VALUE-OBJECTS	244
DOMAIN SERVICES	246
MODULES	251
AGGREGATES	252
AGGREGATE ROOT	254
BEHAVIOR-FOCUSED AGGREGATE ROOT	255
MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION	256
PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES	257
REFERENCE OTHER AGGREGATES BY IDENTITY ONLY	258
FACTORIES	259
REPOSITORIES	261
SUPPLE DESIGN	265
UBIQUITOUS LANGUAGE	266
INTENTION-REVEALING INTERFACES	267
SIDE-EFFECT FREE FUNCTIONS	268
ASSERTIONS	269
CONCEPTUAL CONTOURS	270
STANDALONE CLASSES	271

CLOSURE OF OPERATIONS	272
MODEL INTEGRITY AND CONTEXT	273
BOUNDED CONTEXT	274
CONTINUOUS INTEGRATION	275
STRATEGIC CONTEXT MAP	276
CONTEXTUAL MAP	277
SHARED KERNEL	278
CUSTOMER-SUPPLIER TEAMS	279
CONFORMIST	280
ANTICORRUPTION LAYER	281
SEPARATE WAYS	282
OPEN HOST SERVICE	283
PUBLISHED LANGUAGE	284
DISTILLATION	285
CORE DOMAIN	286
GENERIC SUBDOMAINS	287
DOMAIN VISION STATEMENT	288
HIGHLIGHTED CORE	289
COHESIVE MECHANISMS	290
SEGREGATED CORE	291
ABSTRACT CORE	292
LARGE-SCALE STRUCTURE	293
EVOLVING ORDER	294
SYSTEM METAPHOR	295
RESPONSIBILITY LAYERS	296
KNOWLEDGE LEVEL	300
PLUGGABLE COMPONENT FRAMEWORK	301
ADDITIONAL PATTERNS	302
TYPES OF CONSISTENCY	303
EVENT SOURCING	304
EVENT PROCESSOR	305
EVENT DISPATCHER	306
INTERNAL DOMAIN EVENTS	307
EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS	308
STATIC DOMAIN EVENTS CLASS	309
ONE SUBDOMAIN PER BOUNDED CONTEXT	310
THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS	311
THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS	312
INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE	313
INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES	314
INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS	315
INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE	316
DEPENDENCY INJECTION	317

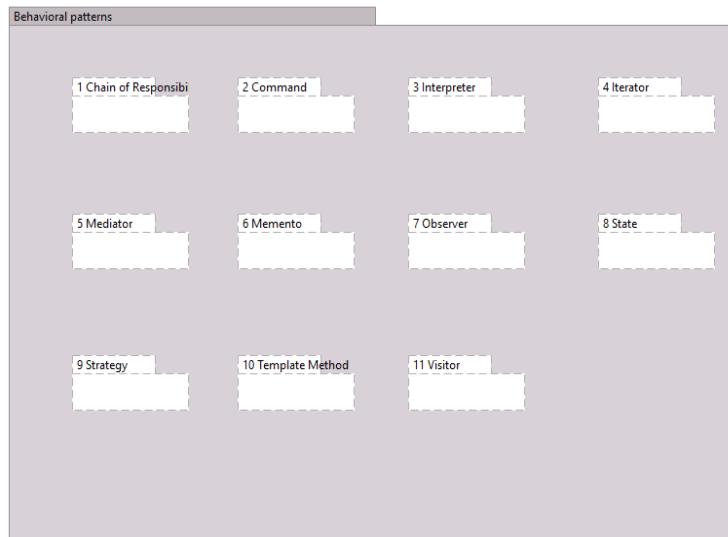
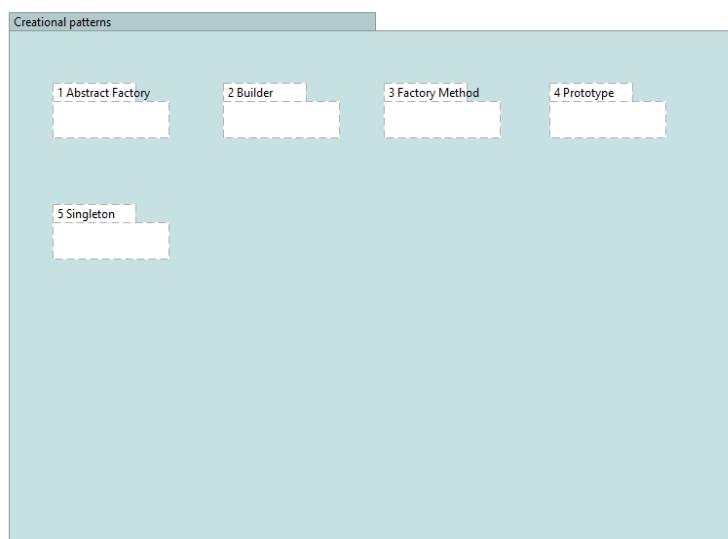
	318
DEPENDENCY INVERSION	319
INVERSION OF CONTROL	320
SERVICE LOCATOR	321
CQRS	322
CQS	323
WRAP LOW-LEVEL EXCEPTIONS	324
EXTRACT DEPENDENCY FROM INTERFACE TO CONSTRUCTOR	325
INTERFACE SEGREGATION	326
CLEAN ARCHITECTURE	327

01

# Design Patterns

Elements of Reusable Object-Oriented Software

GANG OF FOUR



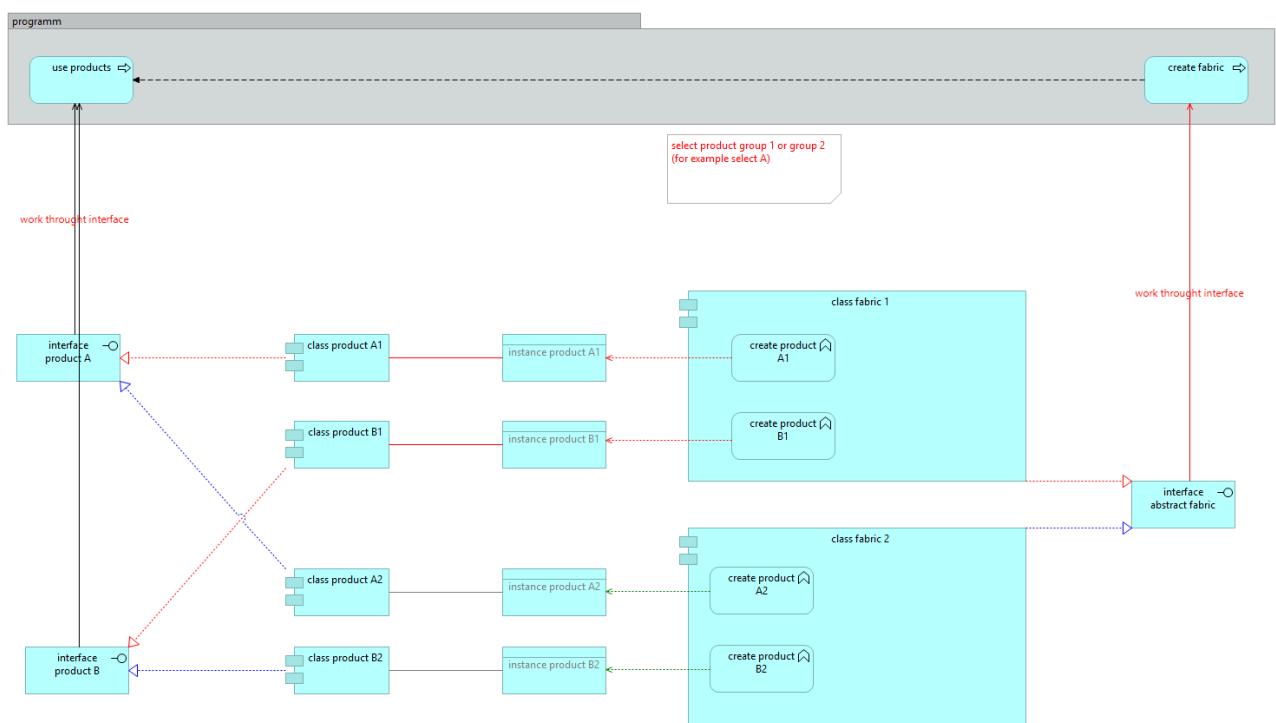
# CREATIONAL PATTERNS

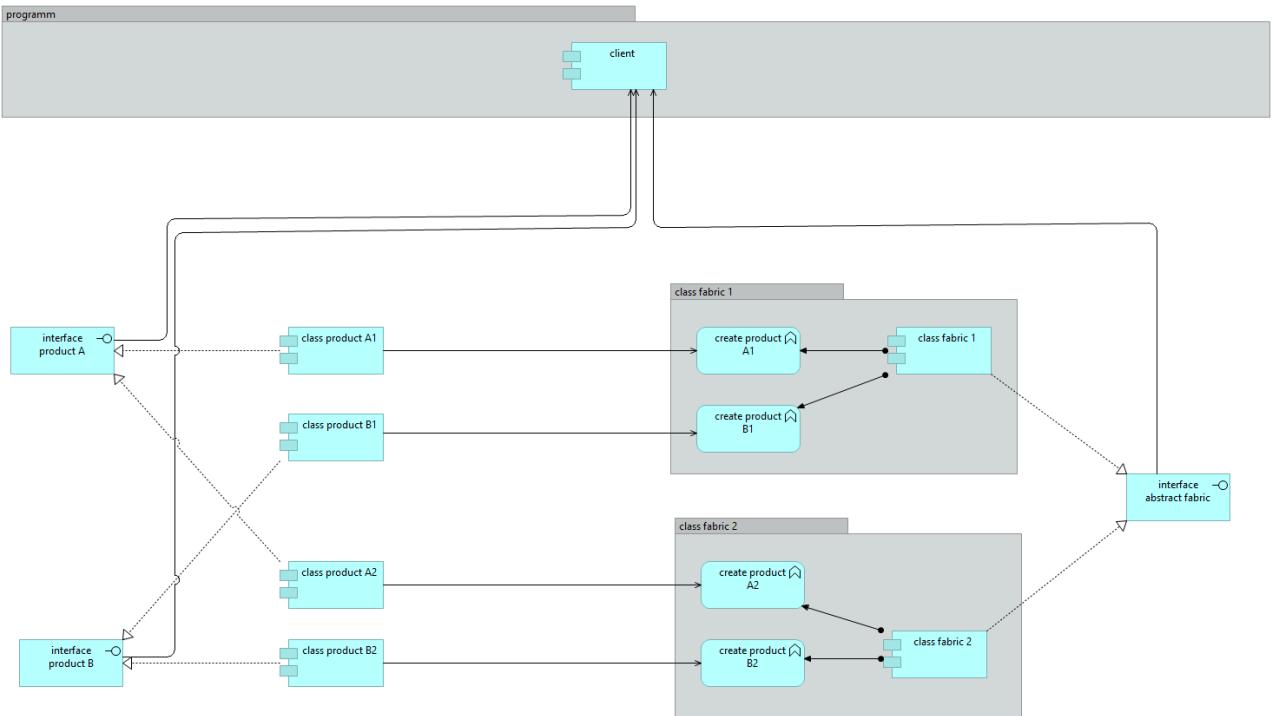
DESIGN PATTERNS

GoF

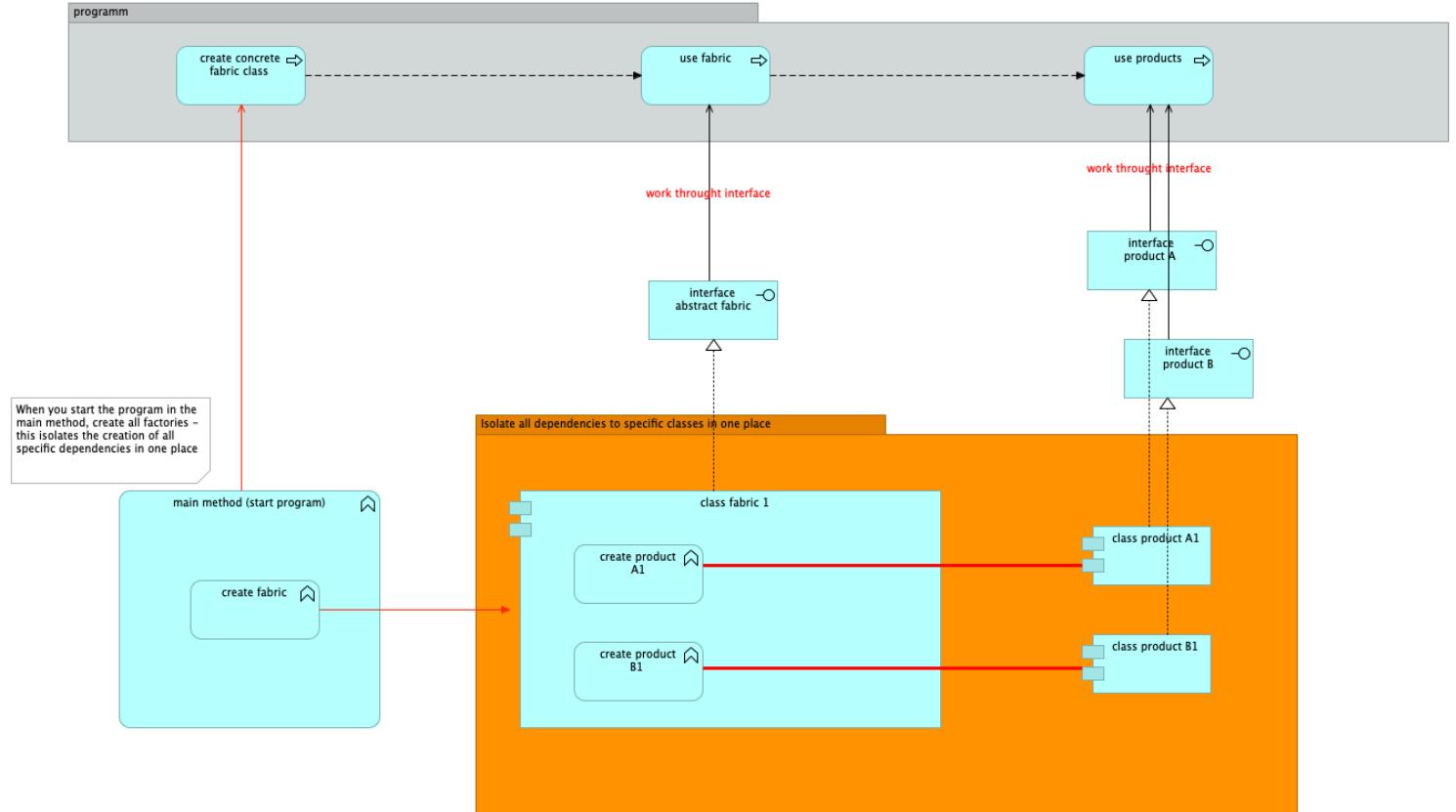


# ABSTRACT FACTORY

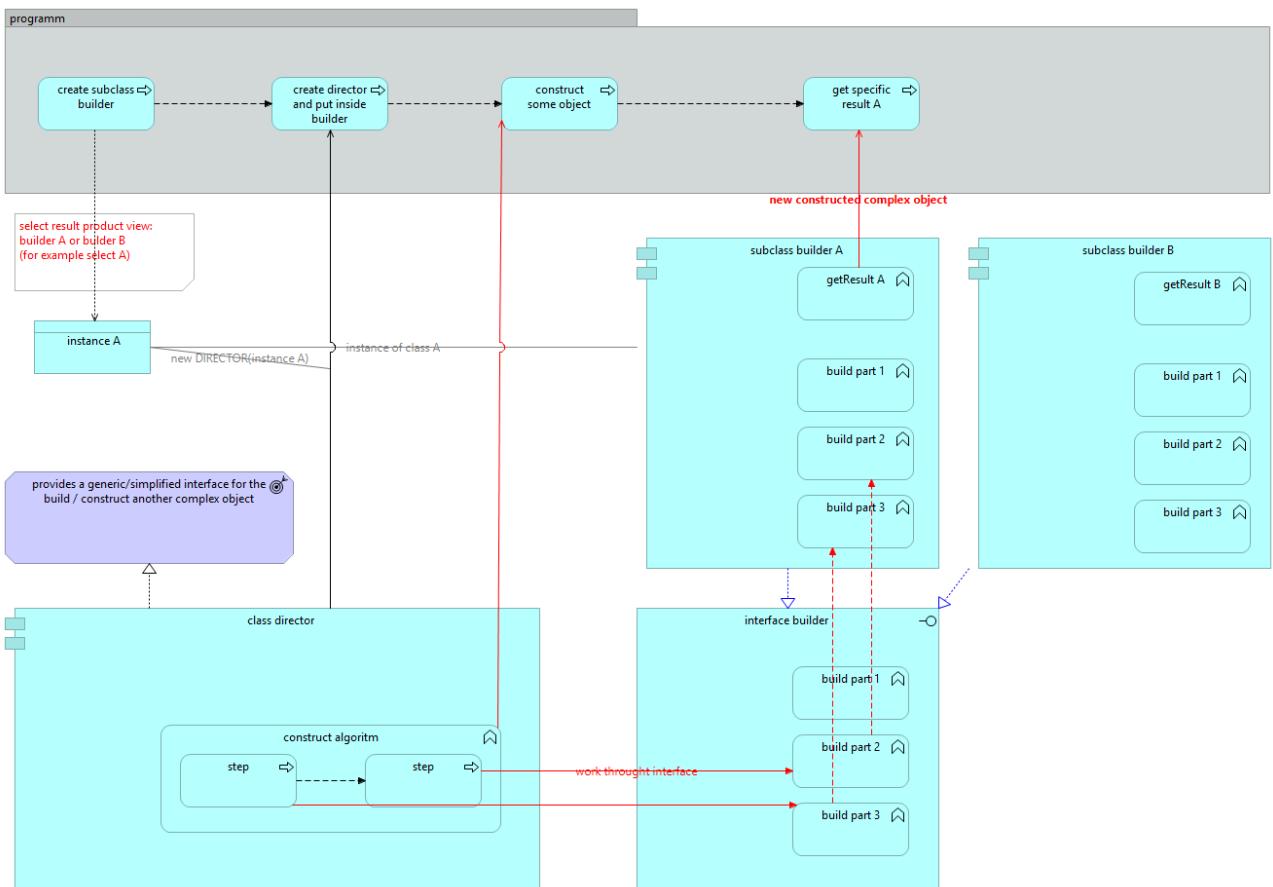


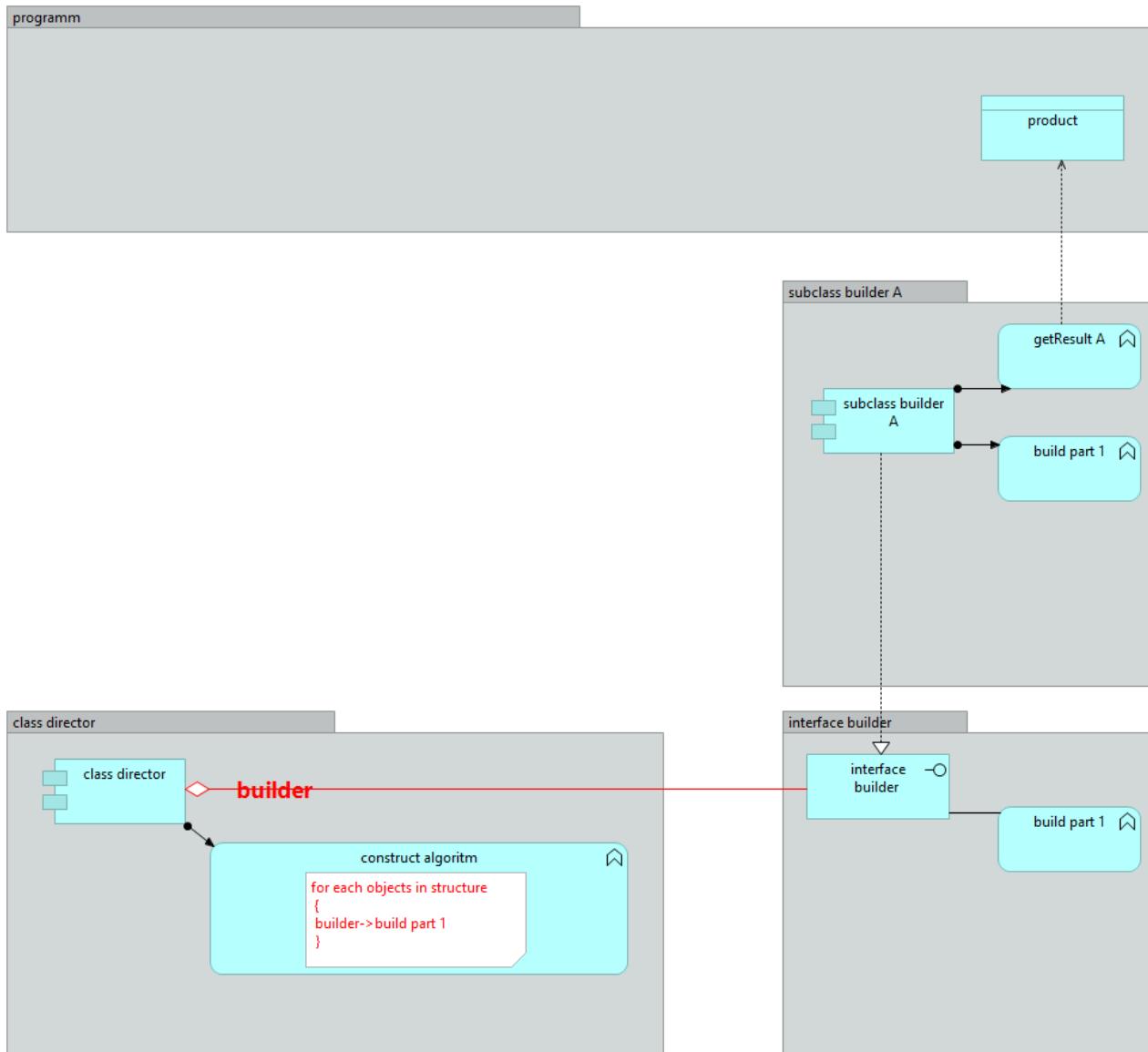


In the whole program (except for the factory inside and one factory creation) we work only with interfaces (and do not refer to specific classes)

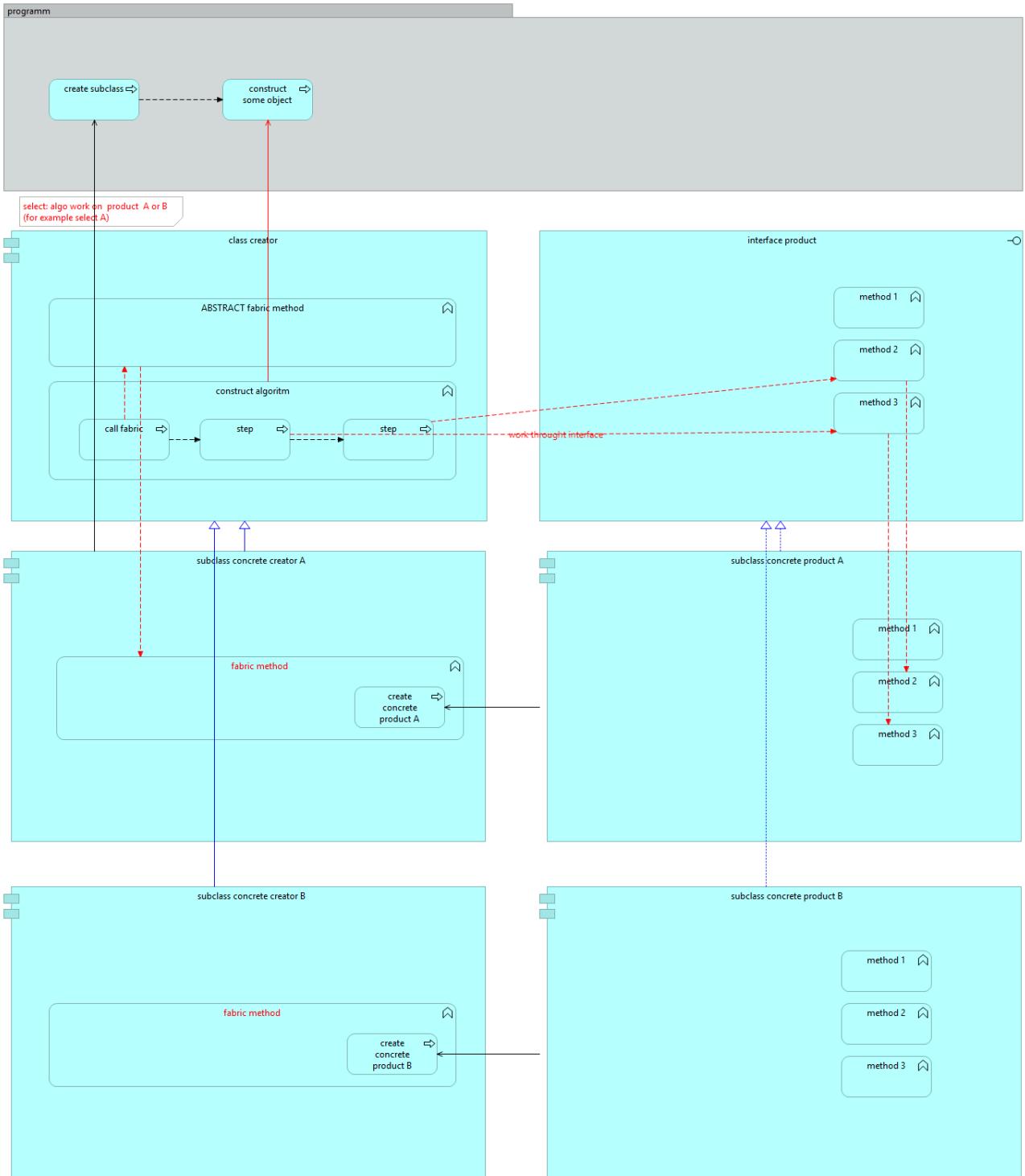


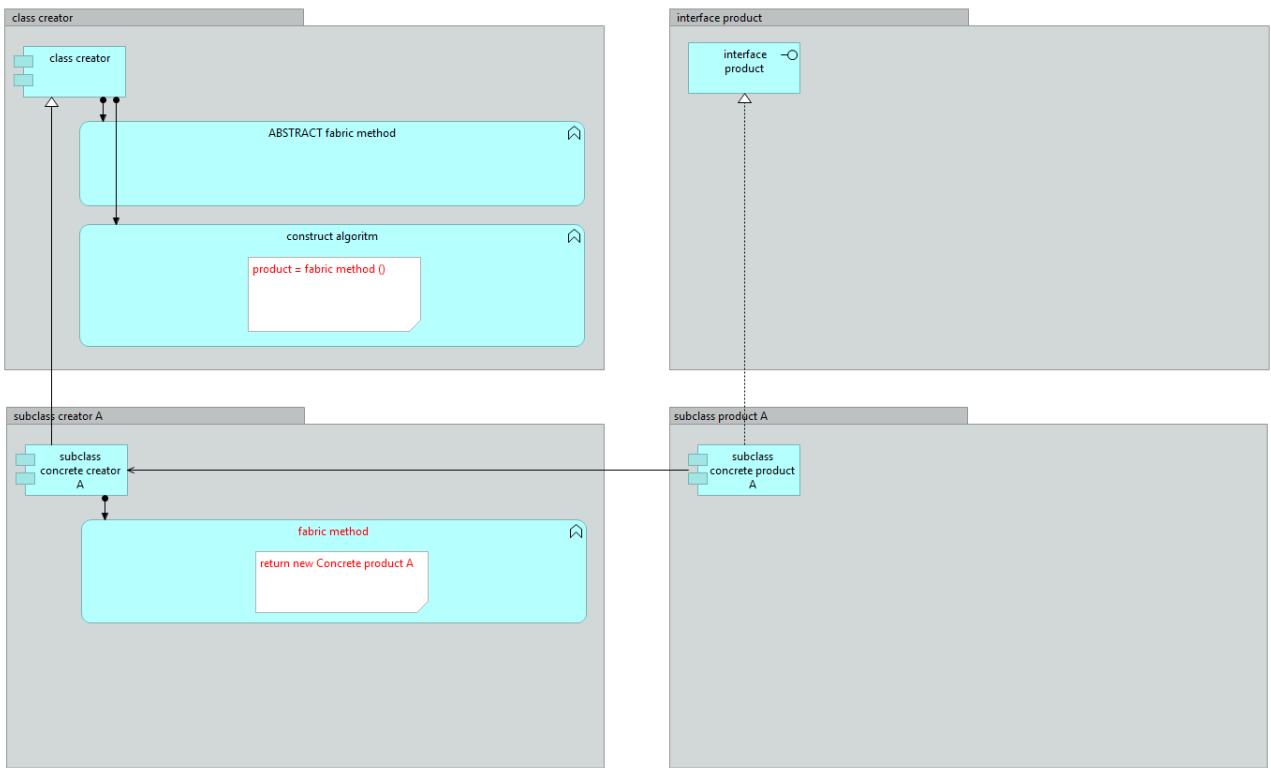
# BUILDER



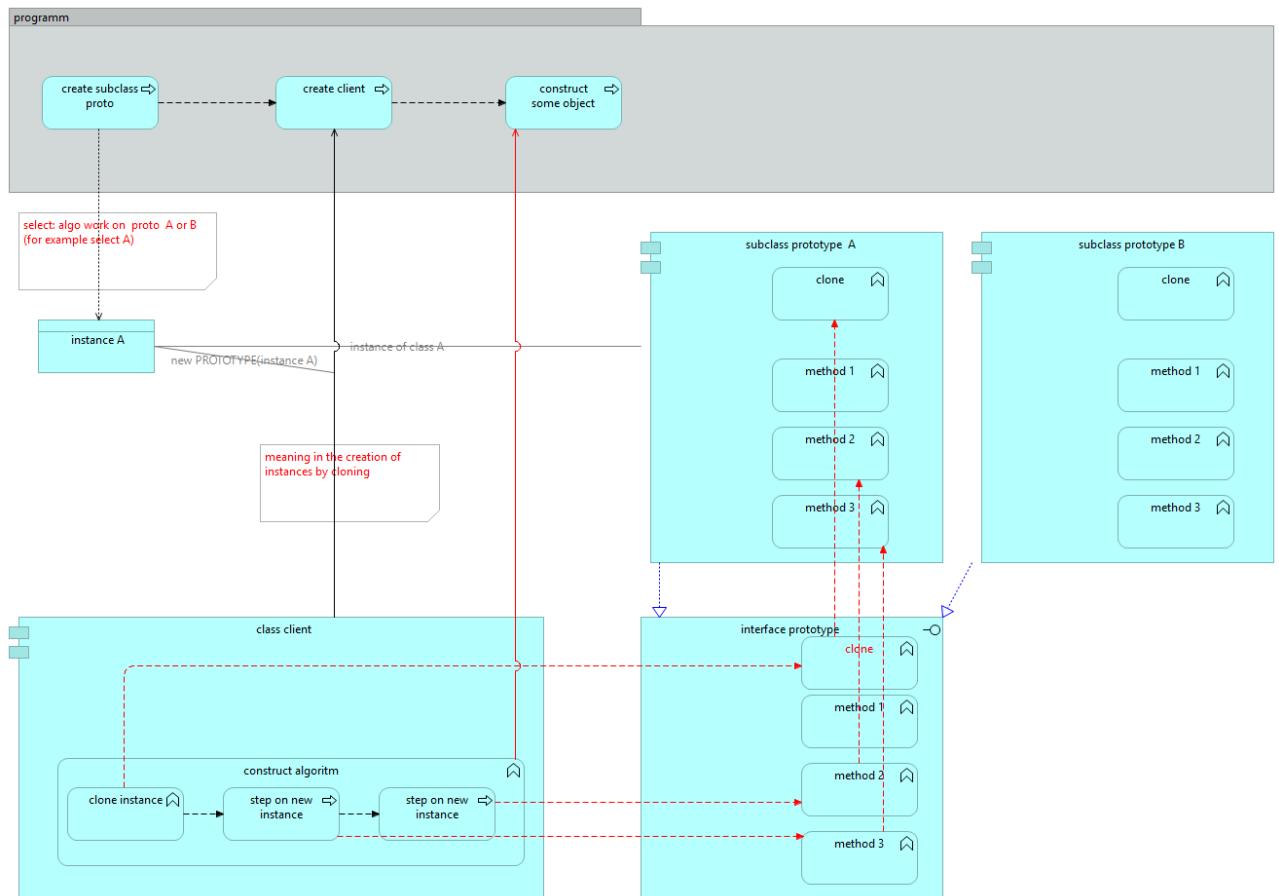


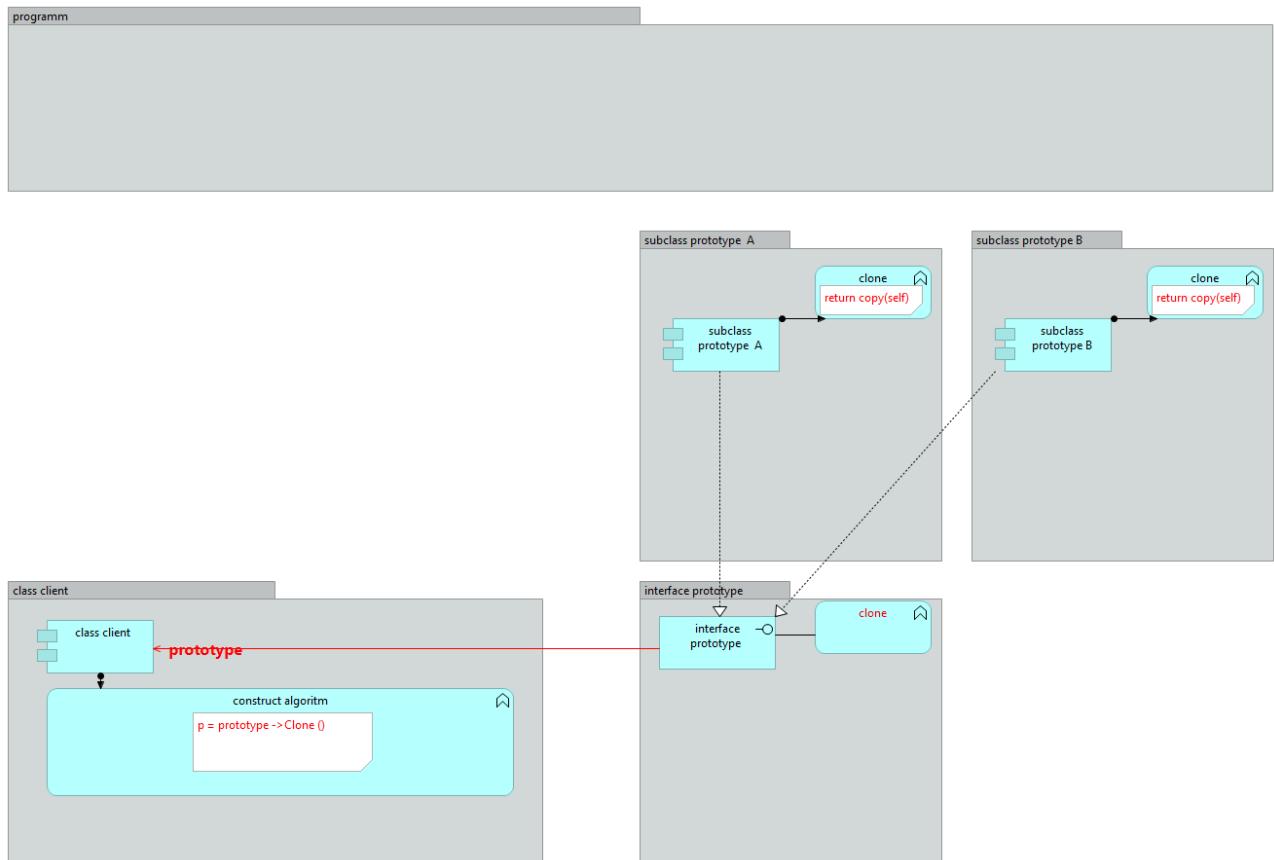
# FACTORY METHOD



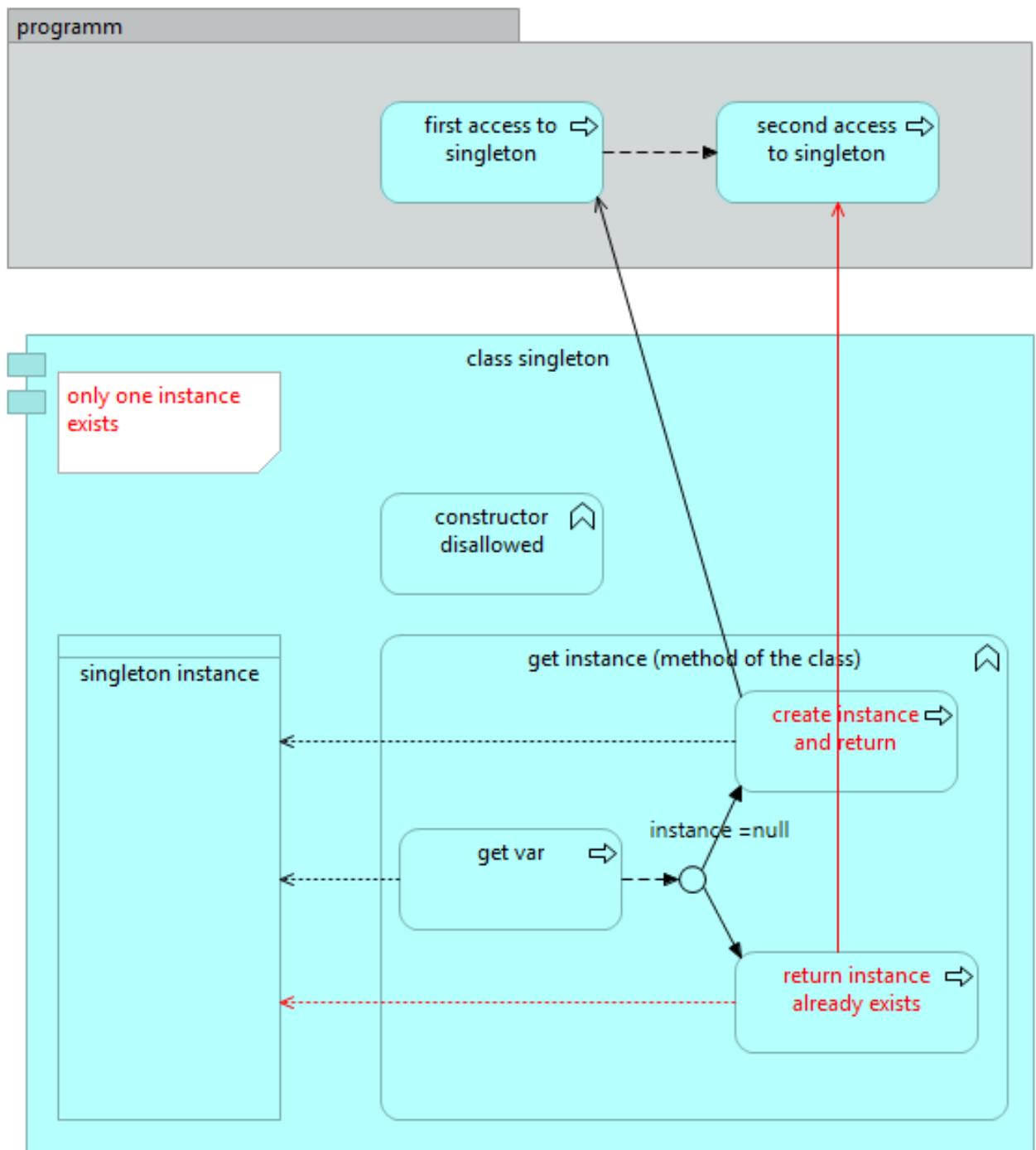


# PROTOTYPE





# SINGLETON



programm

class singleton

  class singleton

  singleton instance

  get instance (method of the class)

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

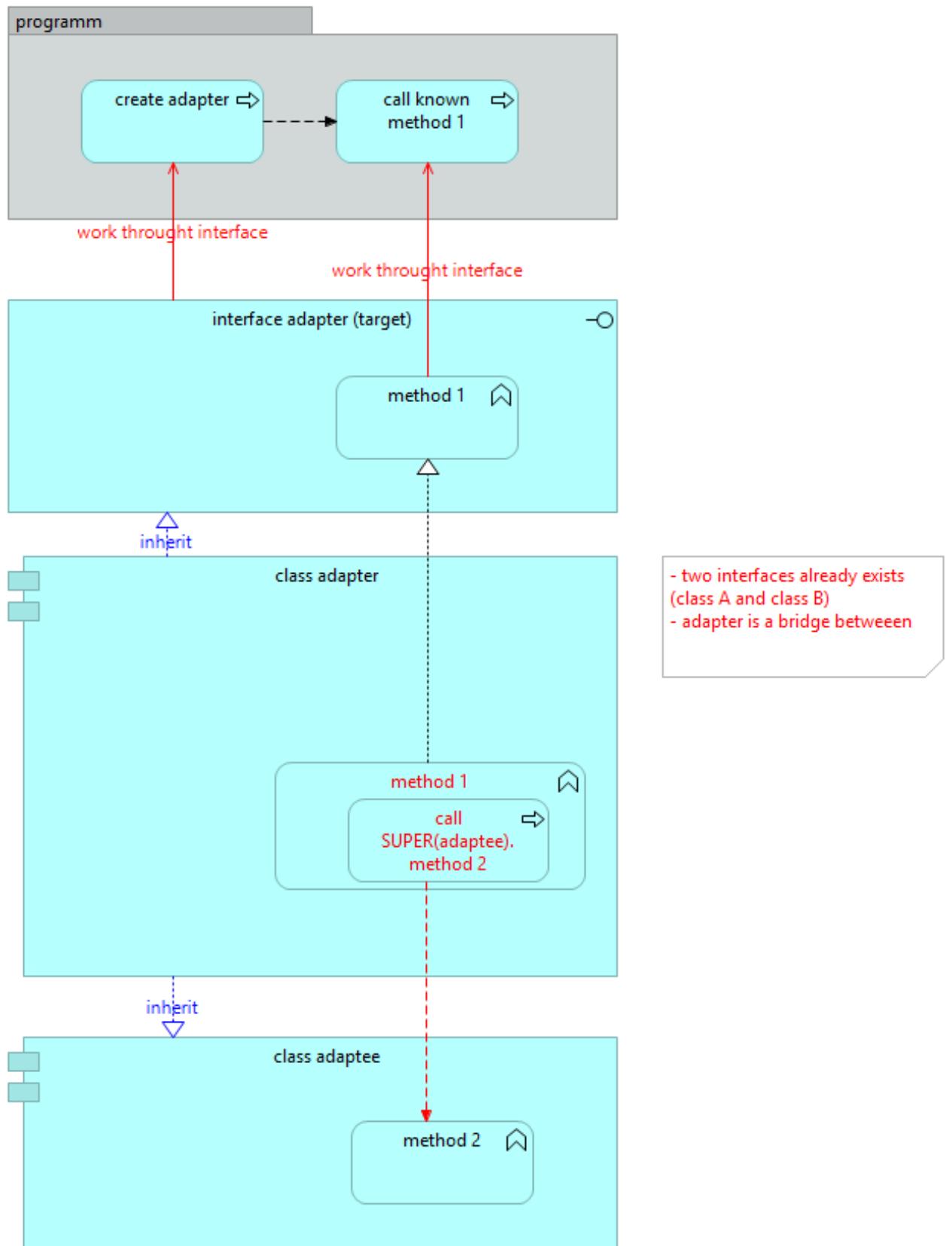
# STRUCTURAL PATTERNS

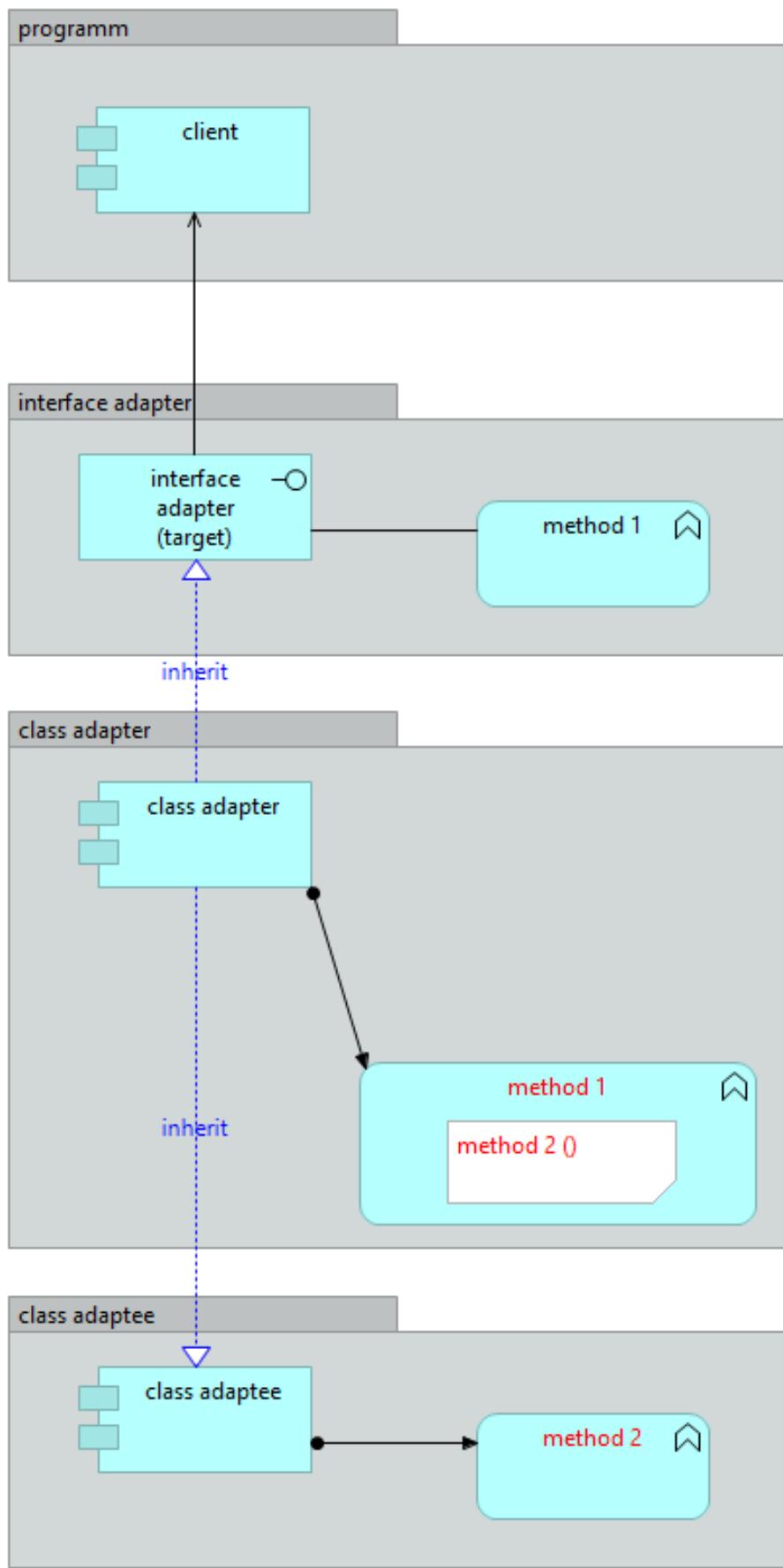
DESIGN PATTERNS

GoF

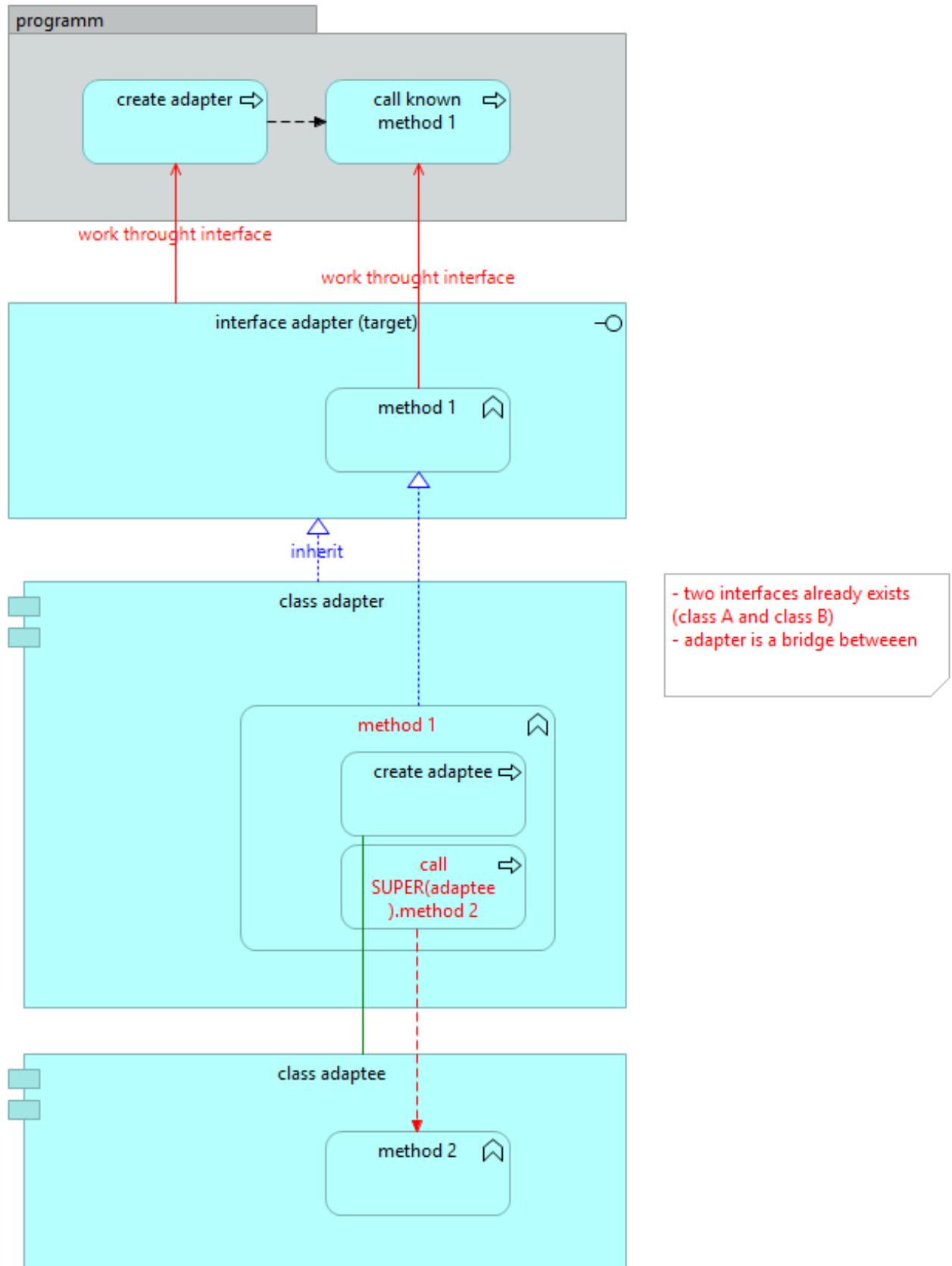


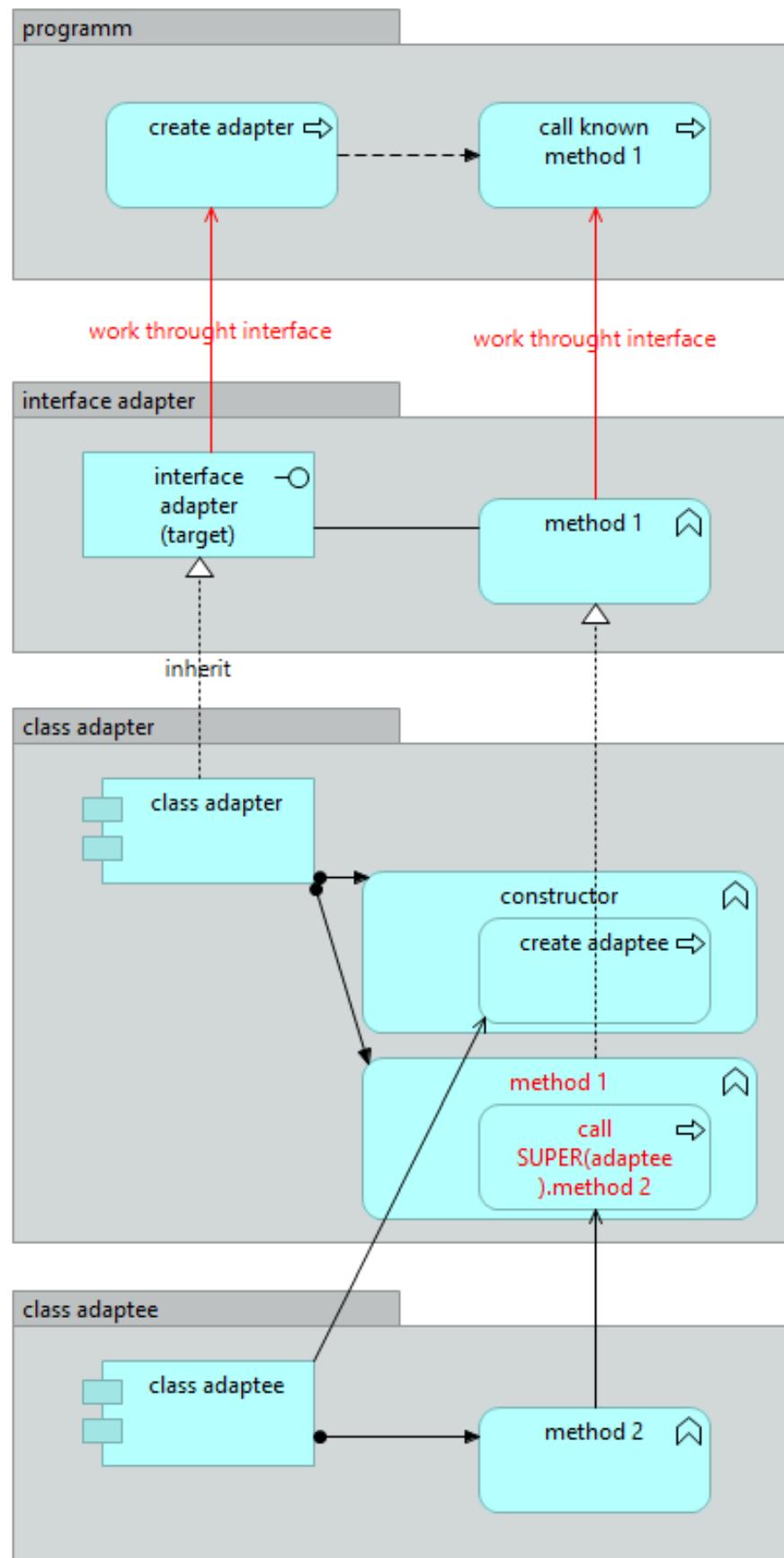
# ADAPTER OF CLASS

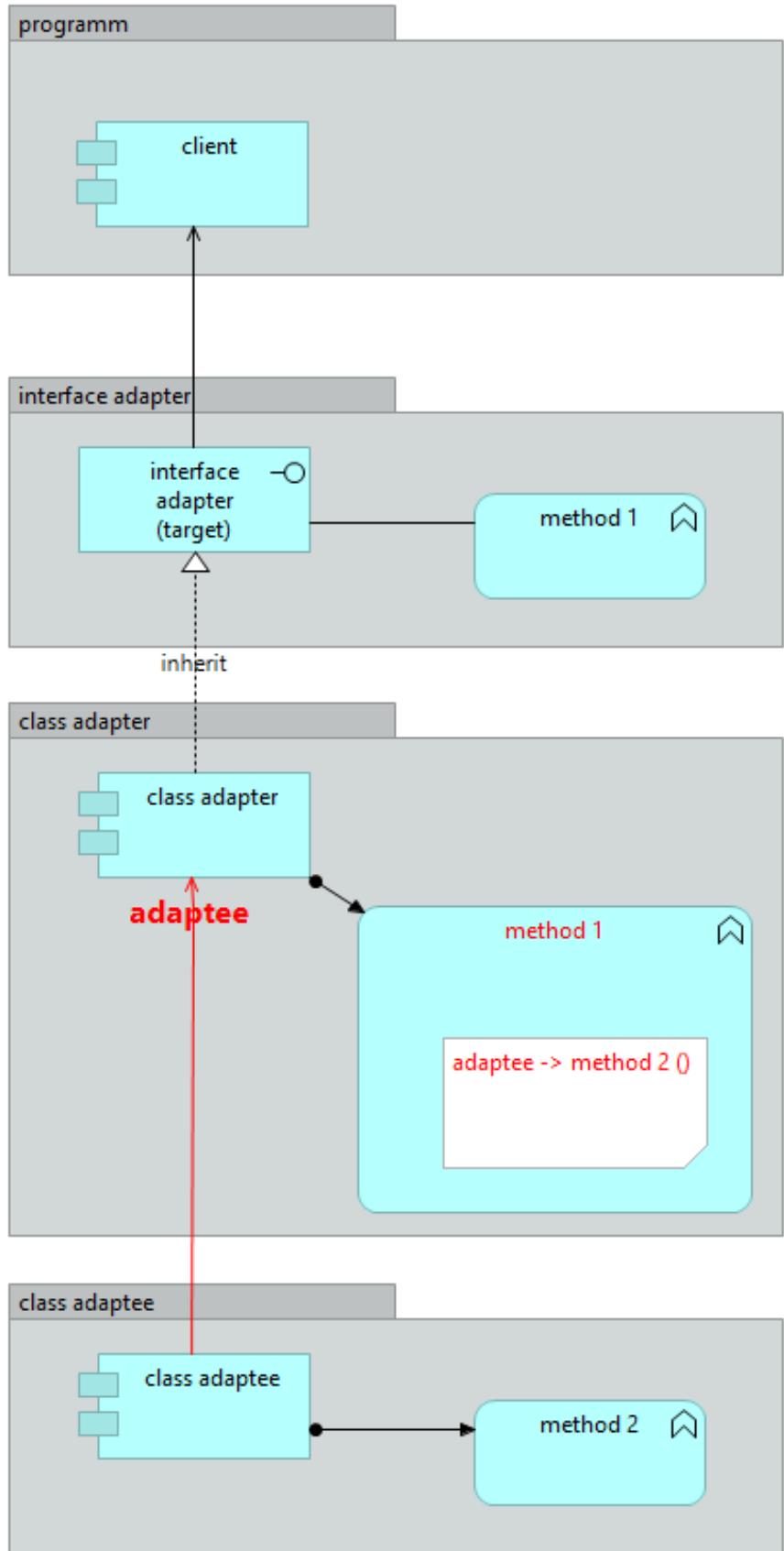




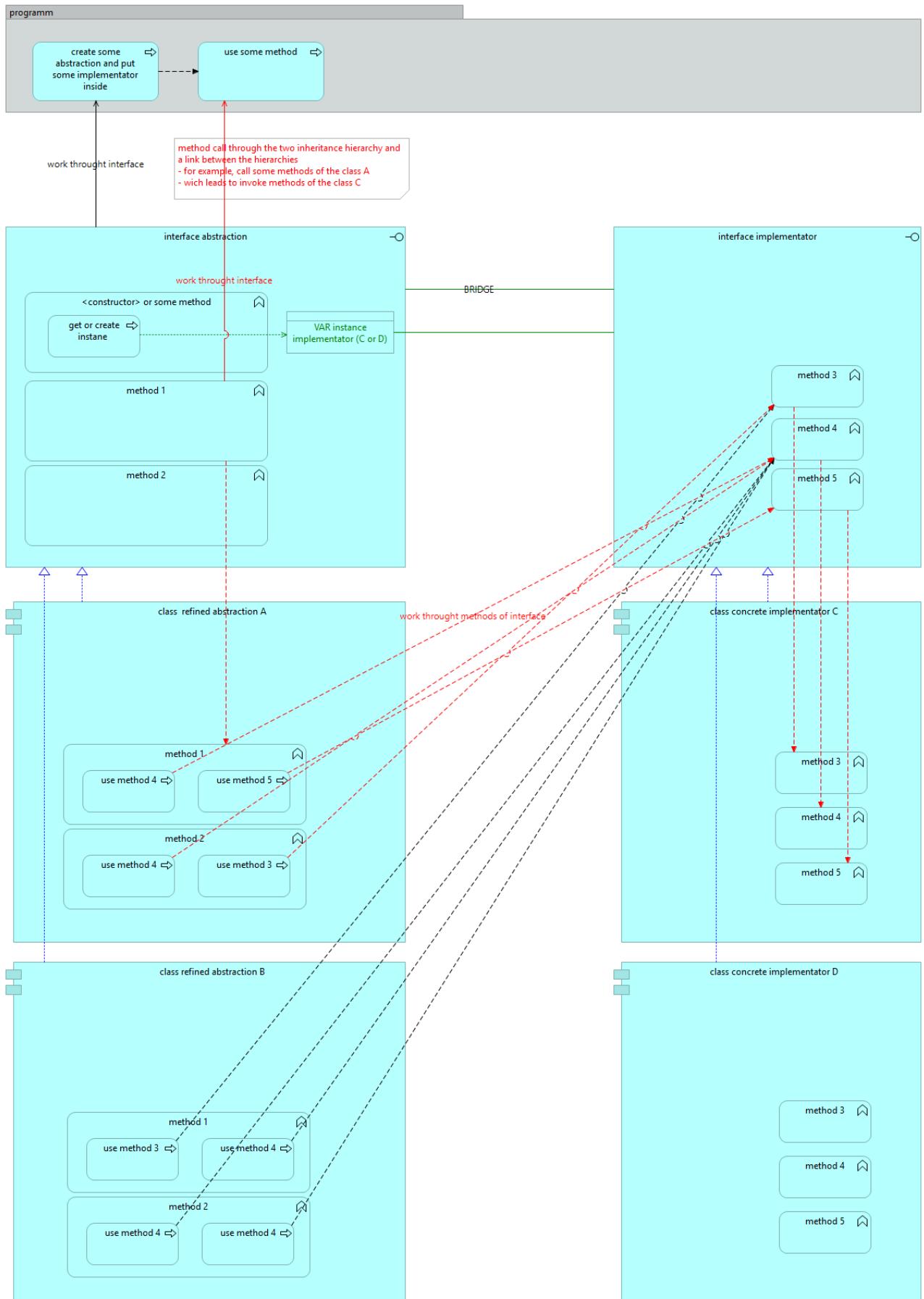
# ADAPTER OF OBJECT

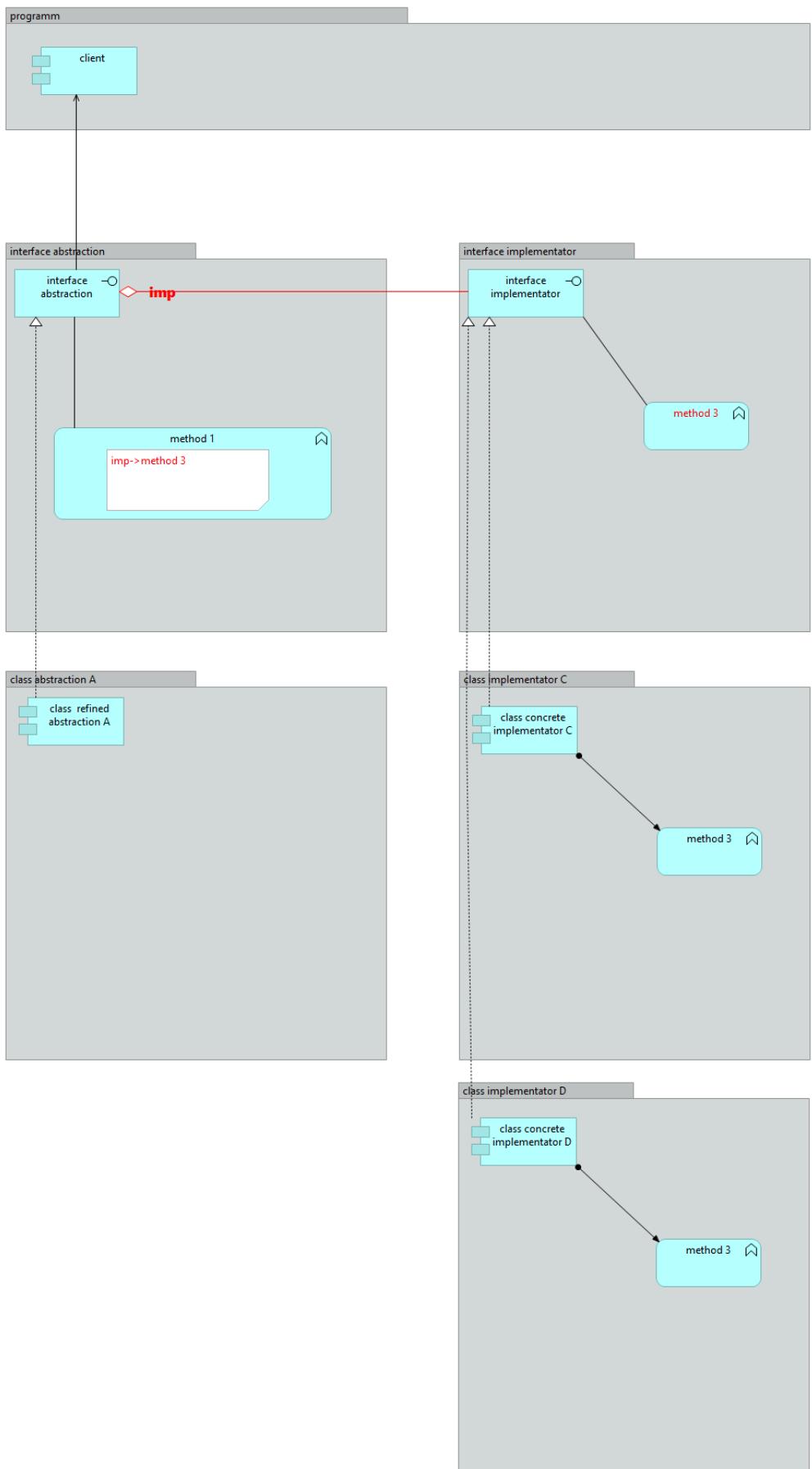




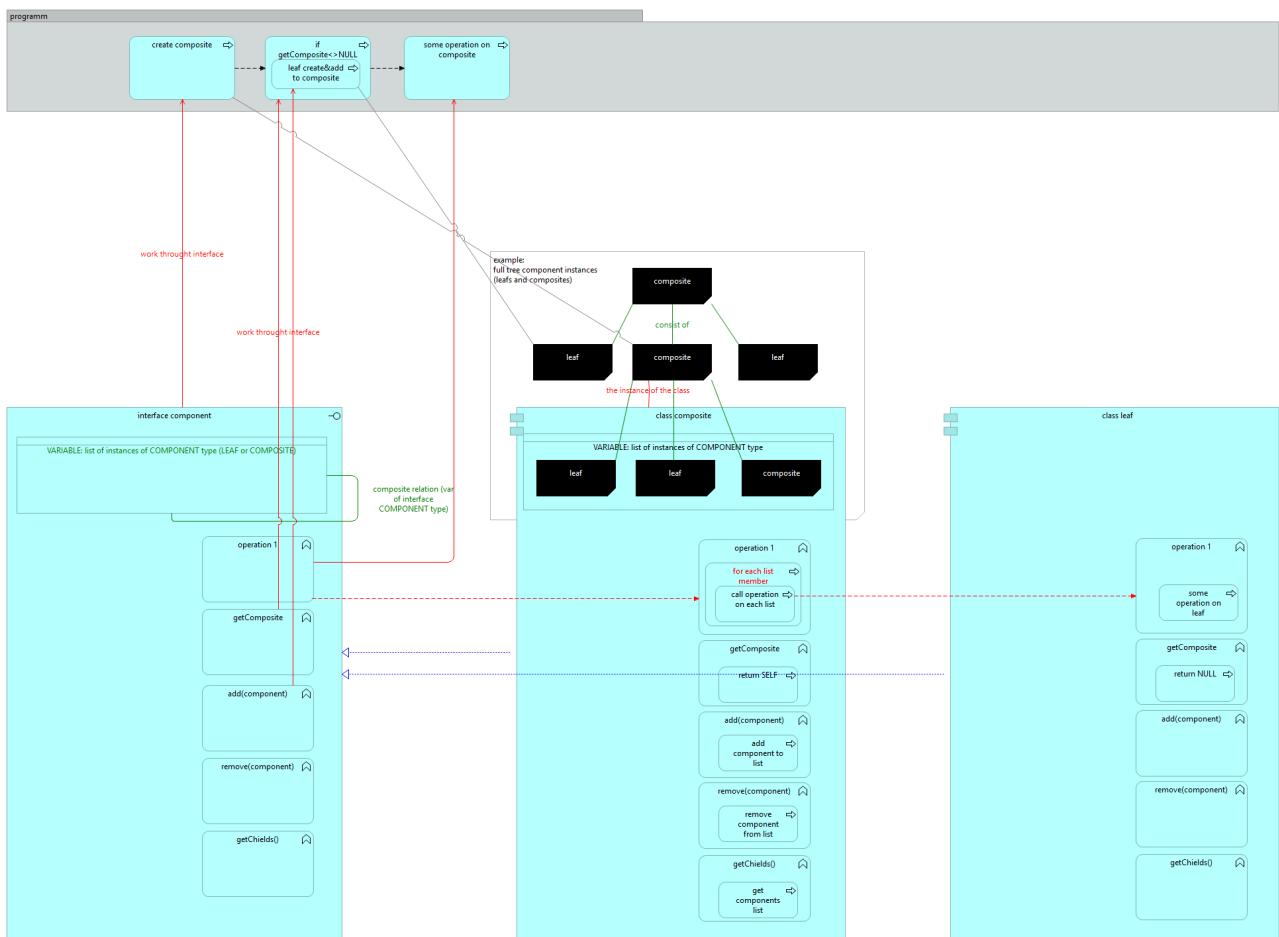


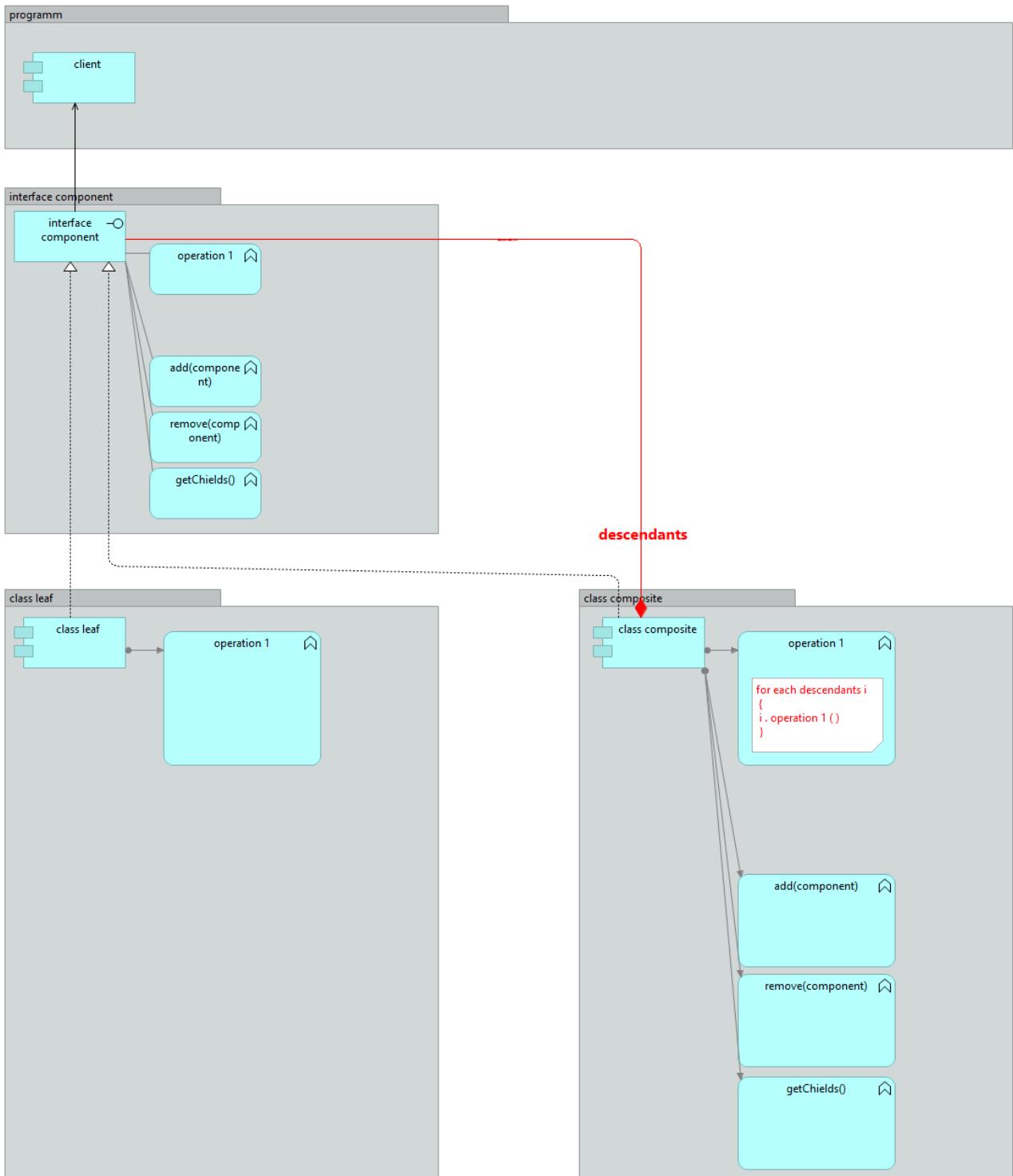
# BRIDGE



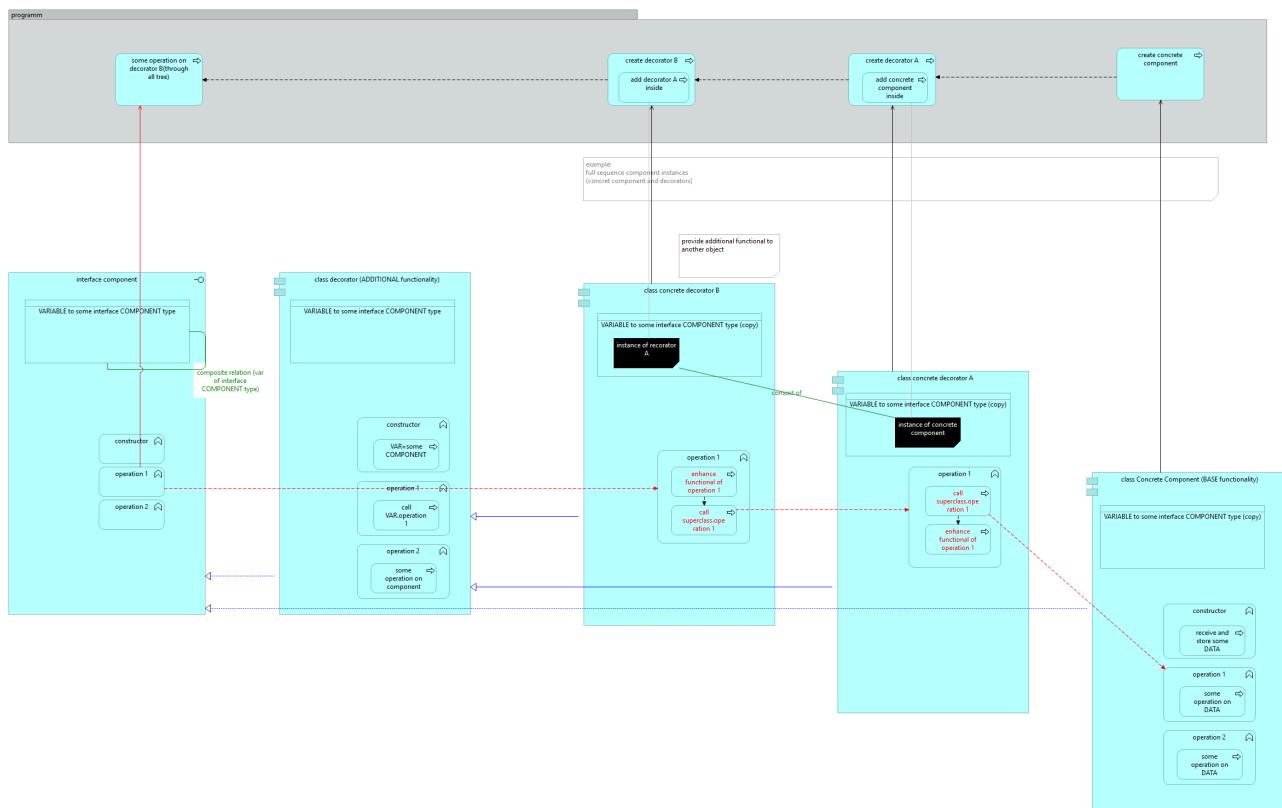


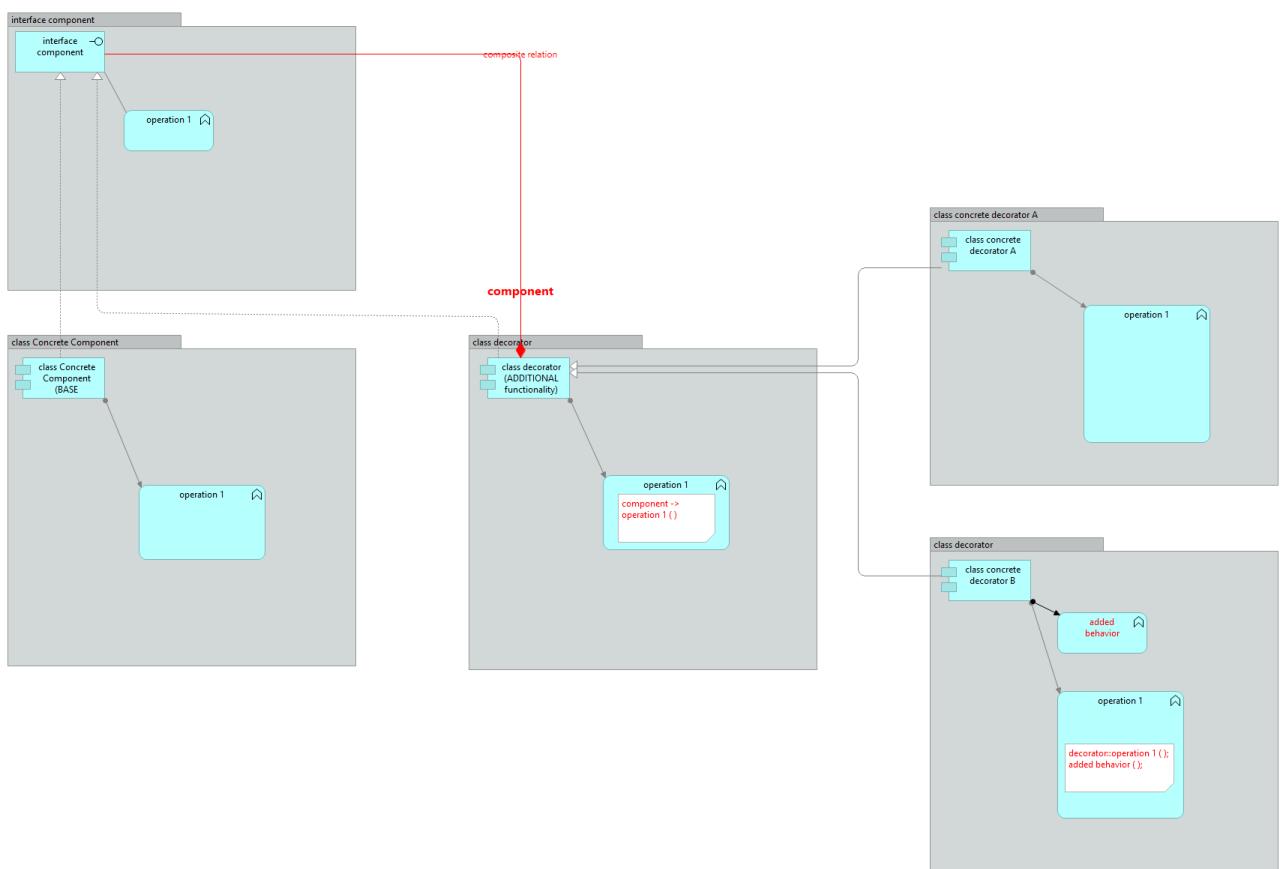
# COMPOSITE



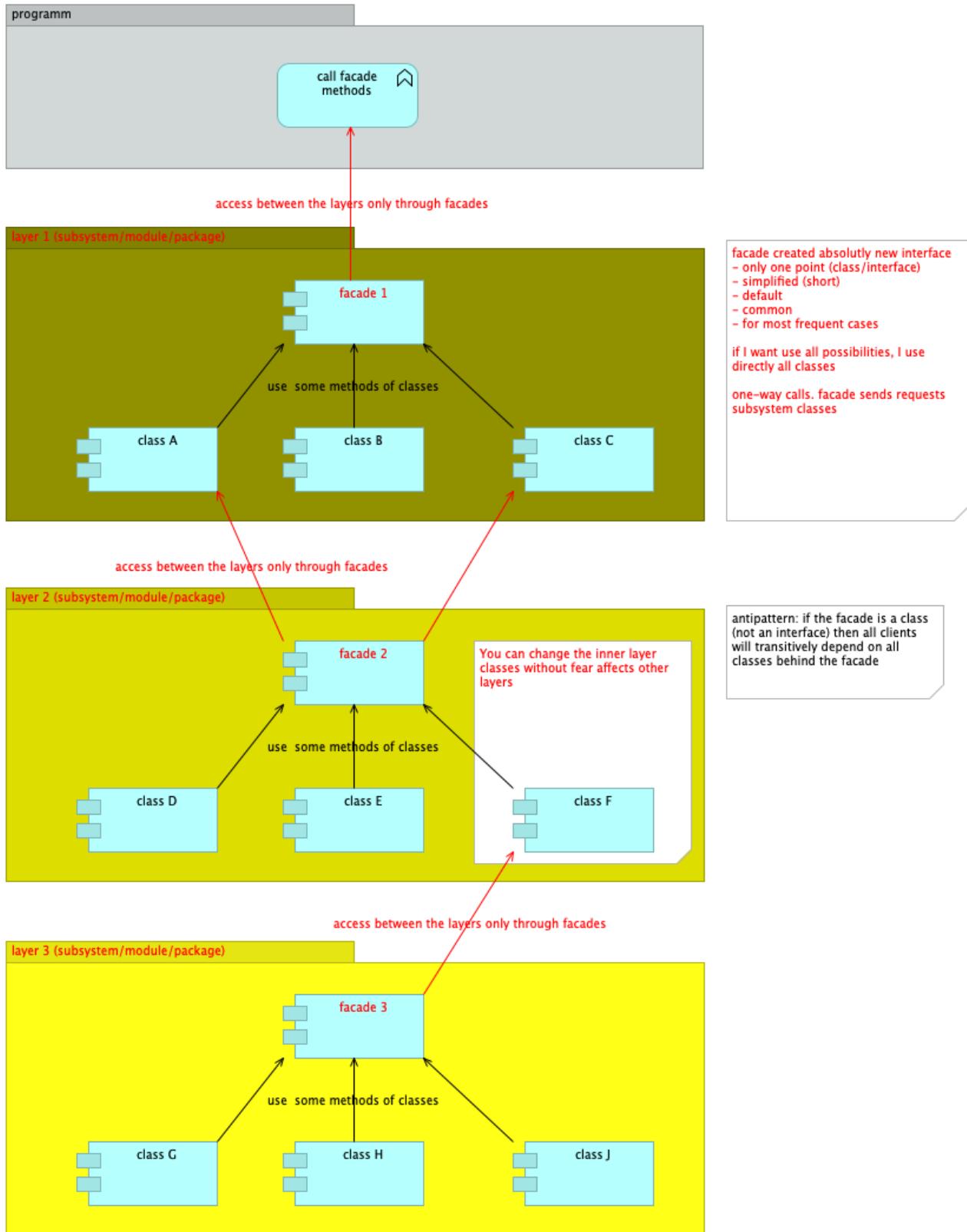


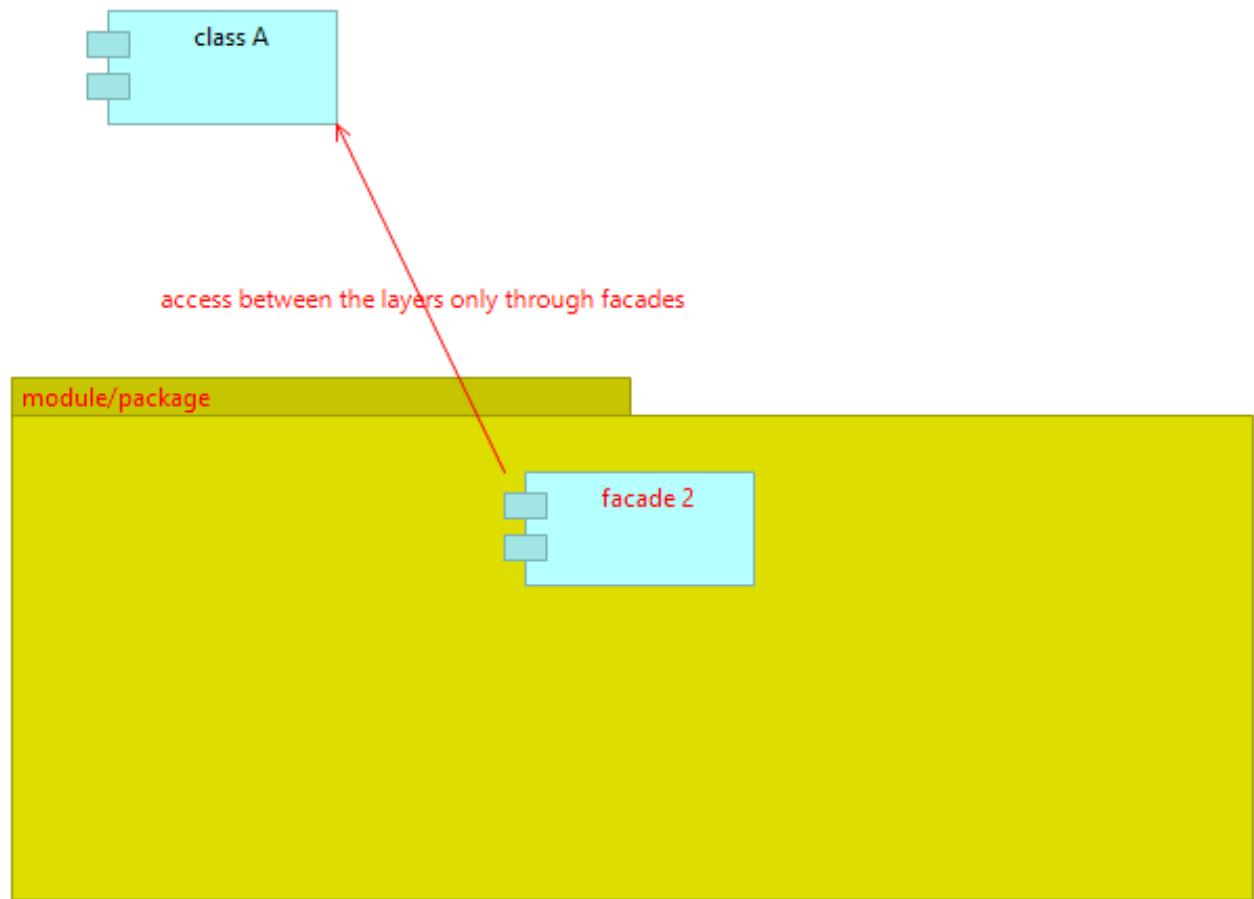
# DECORATOR

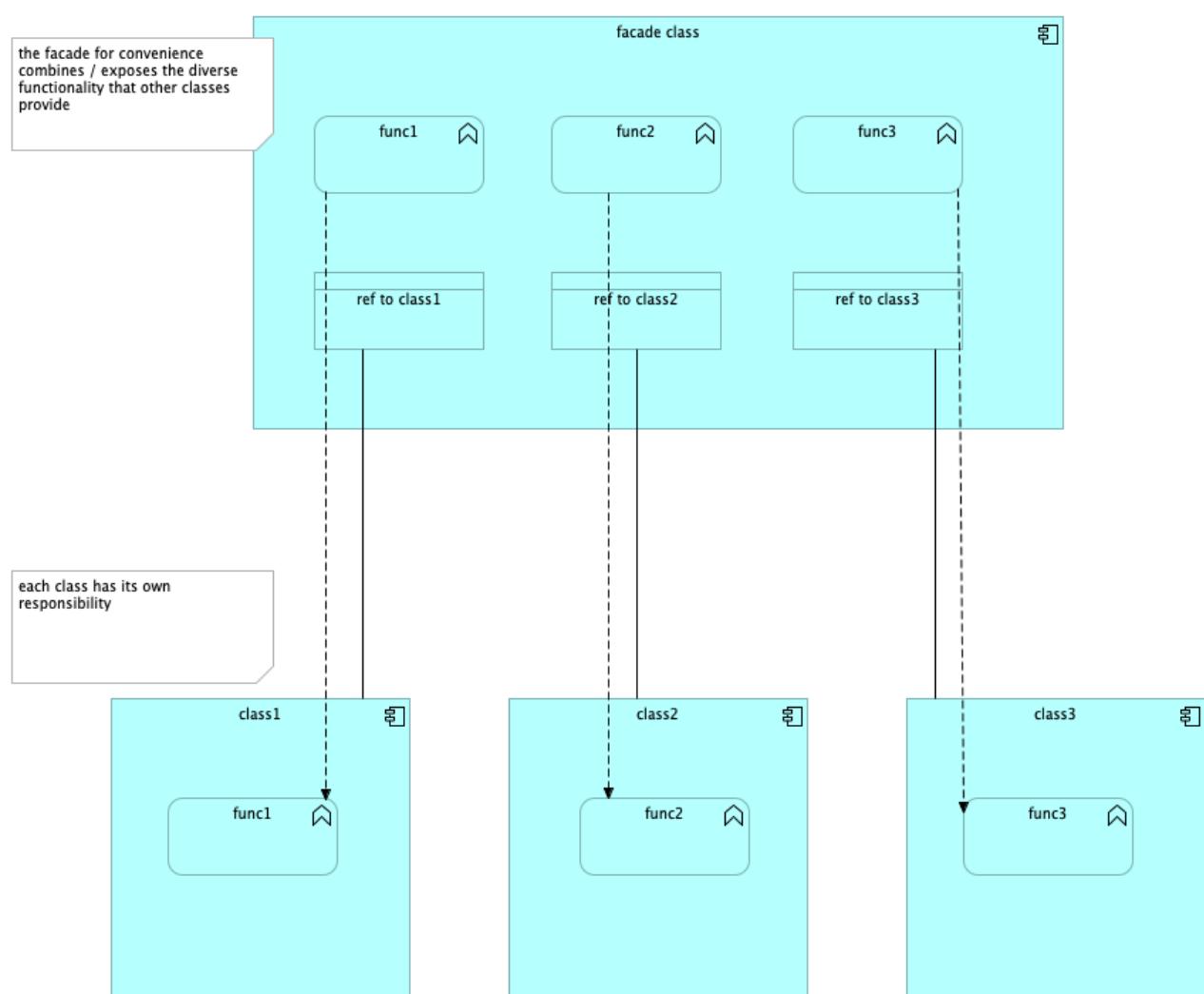




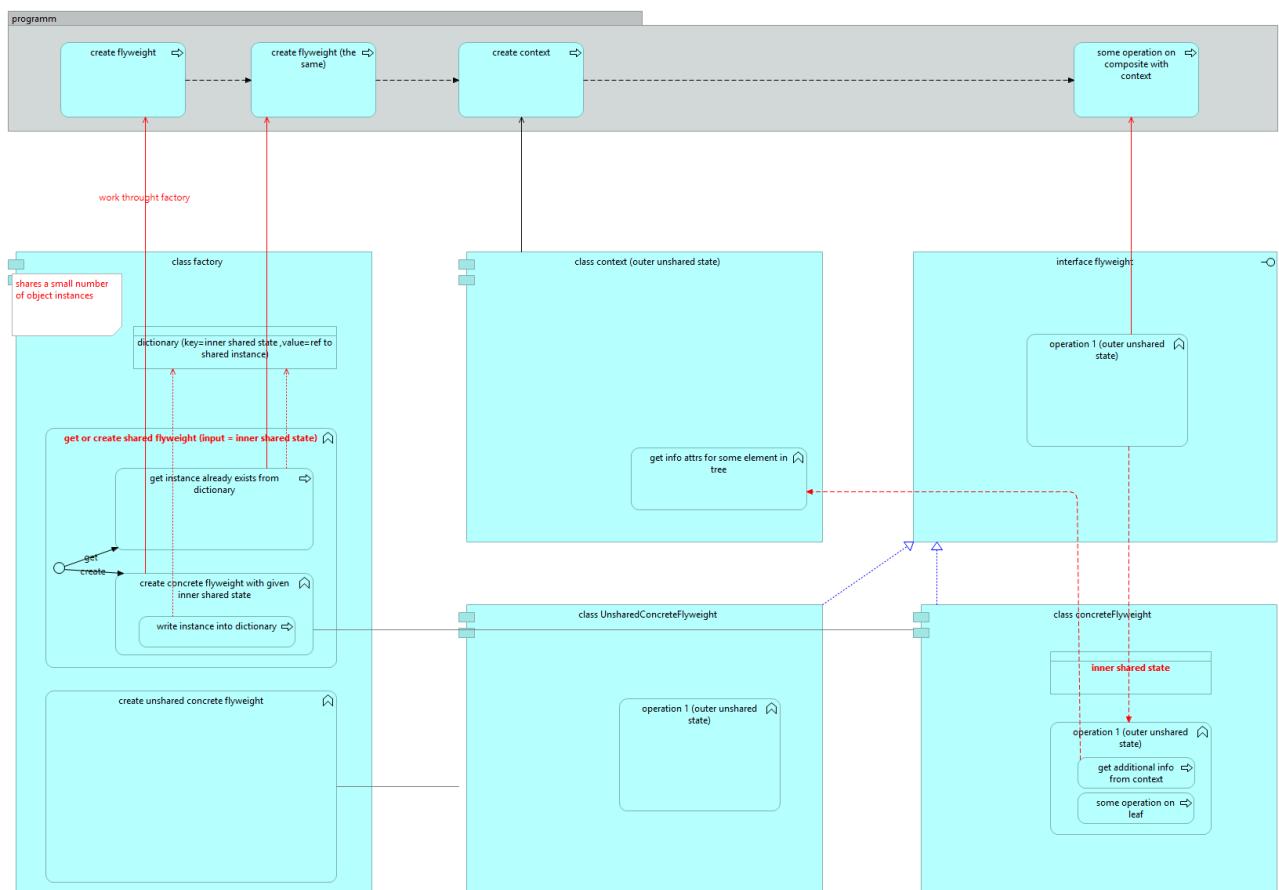
# FACADE

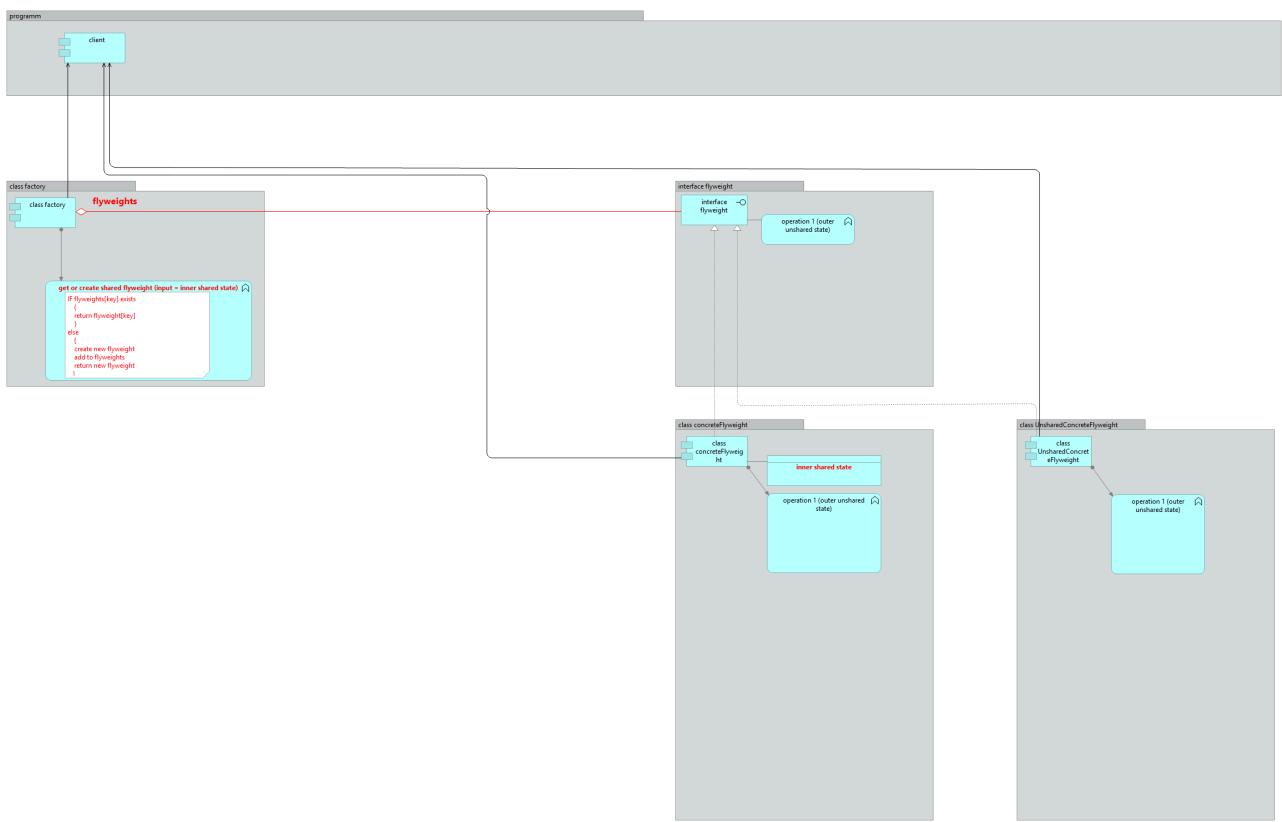




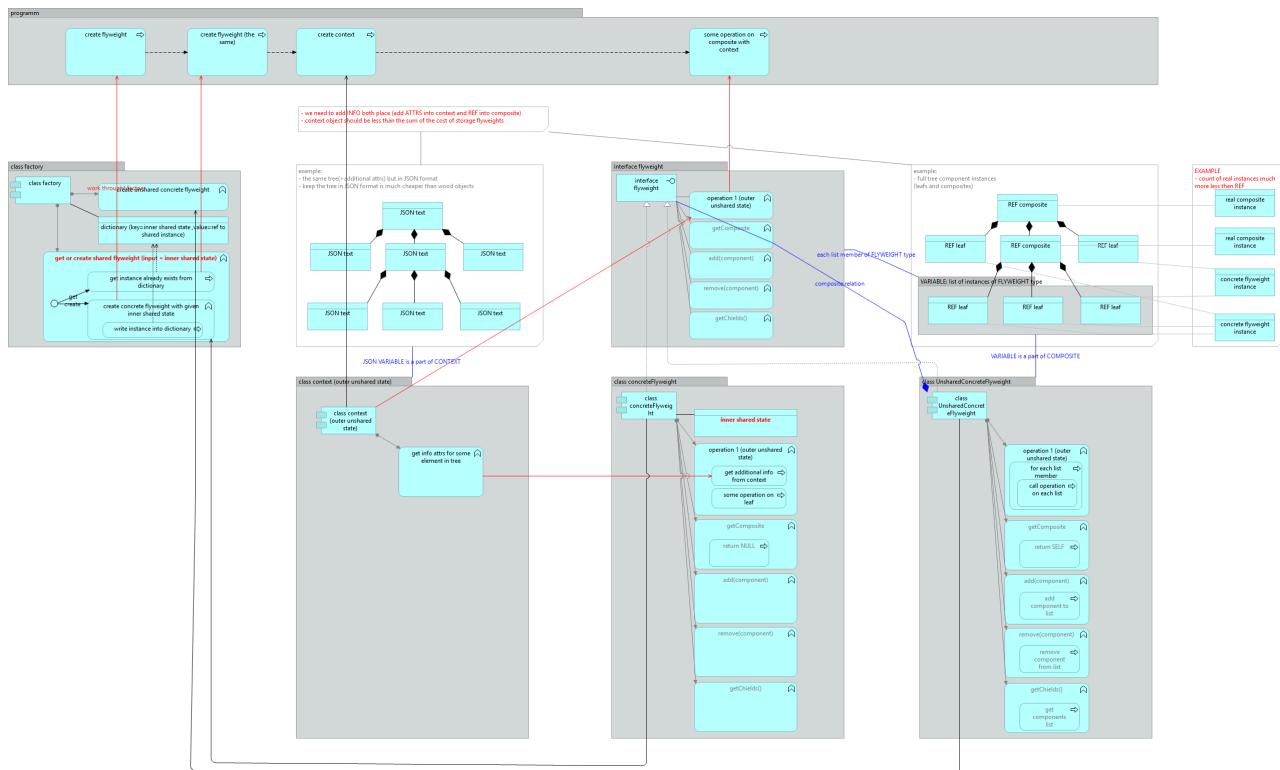


# FLYWEIGHT

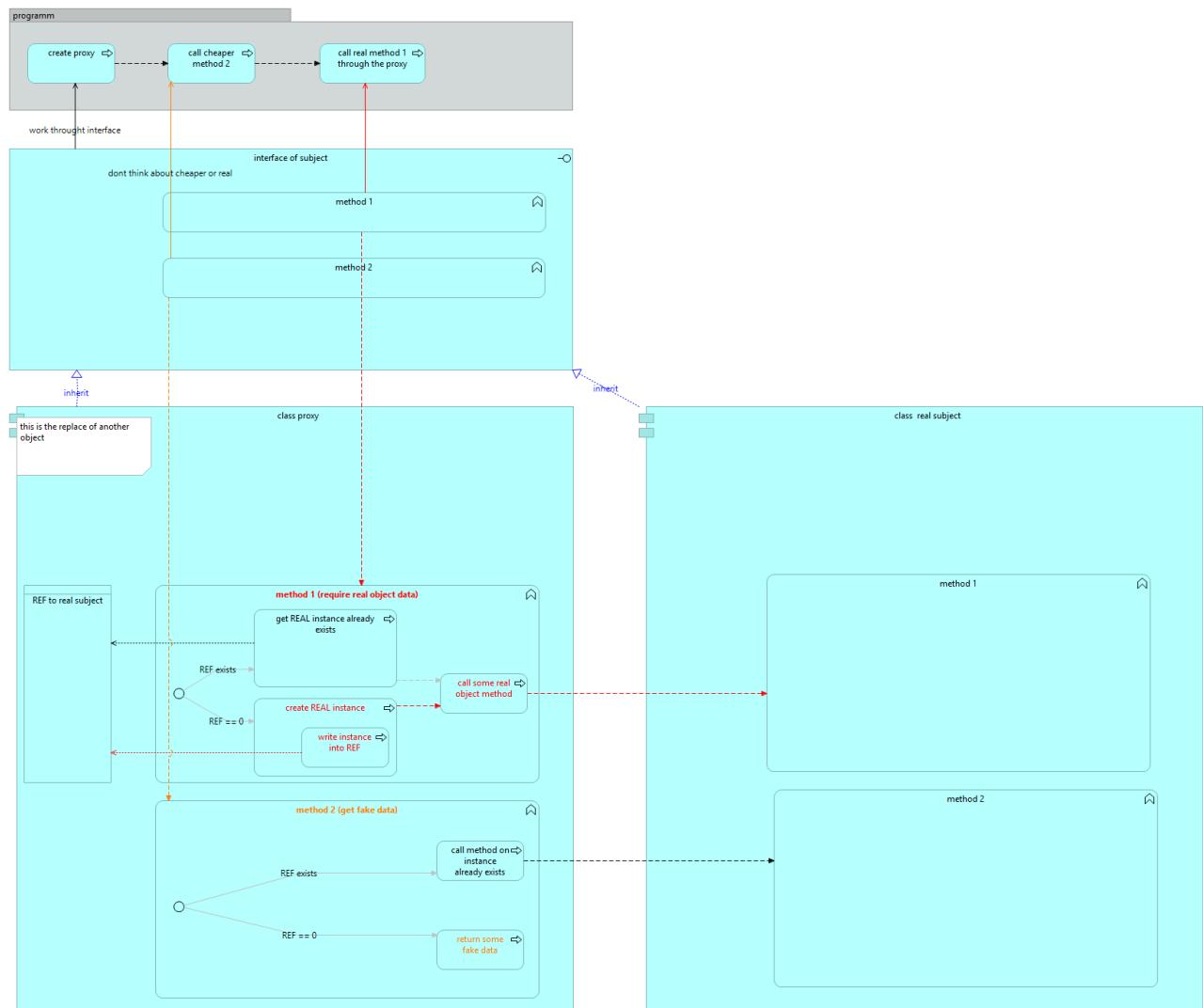


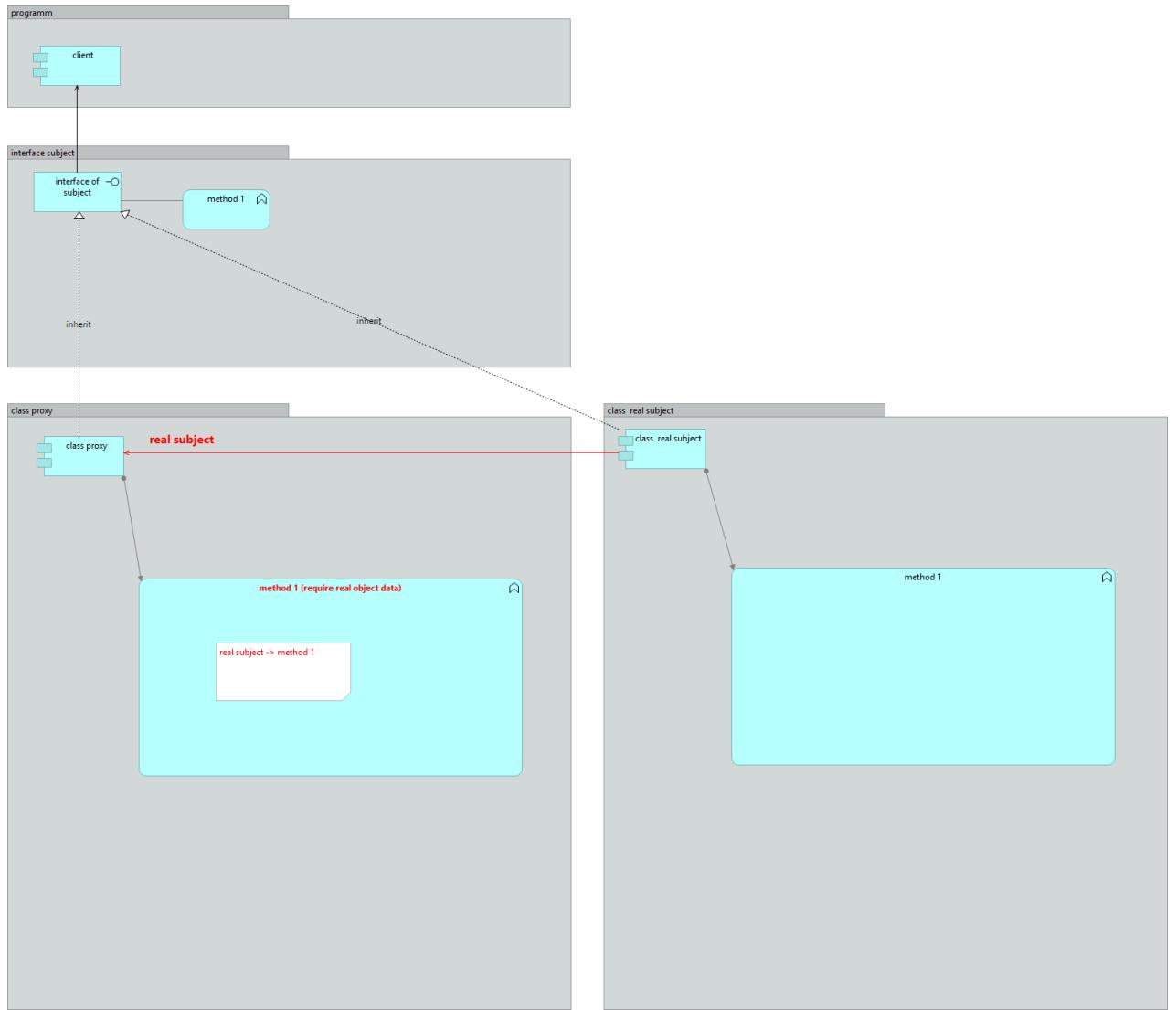


# FLYWEIGHT + COMPOSITE



# PROXY





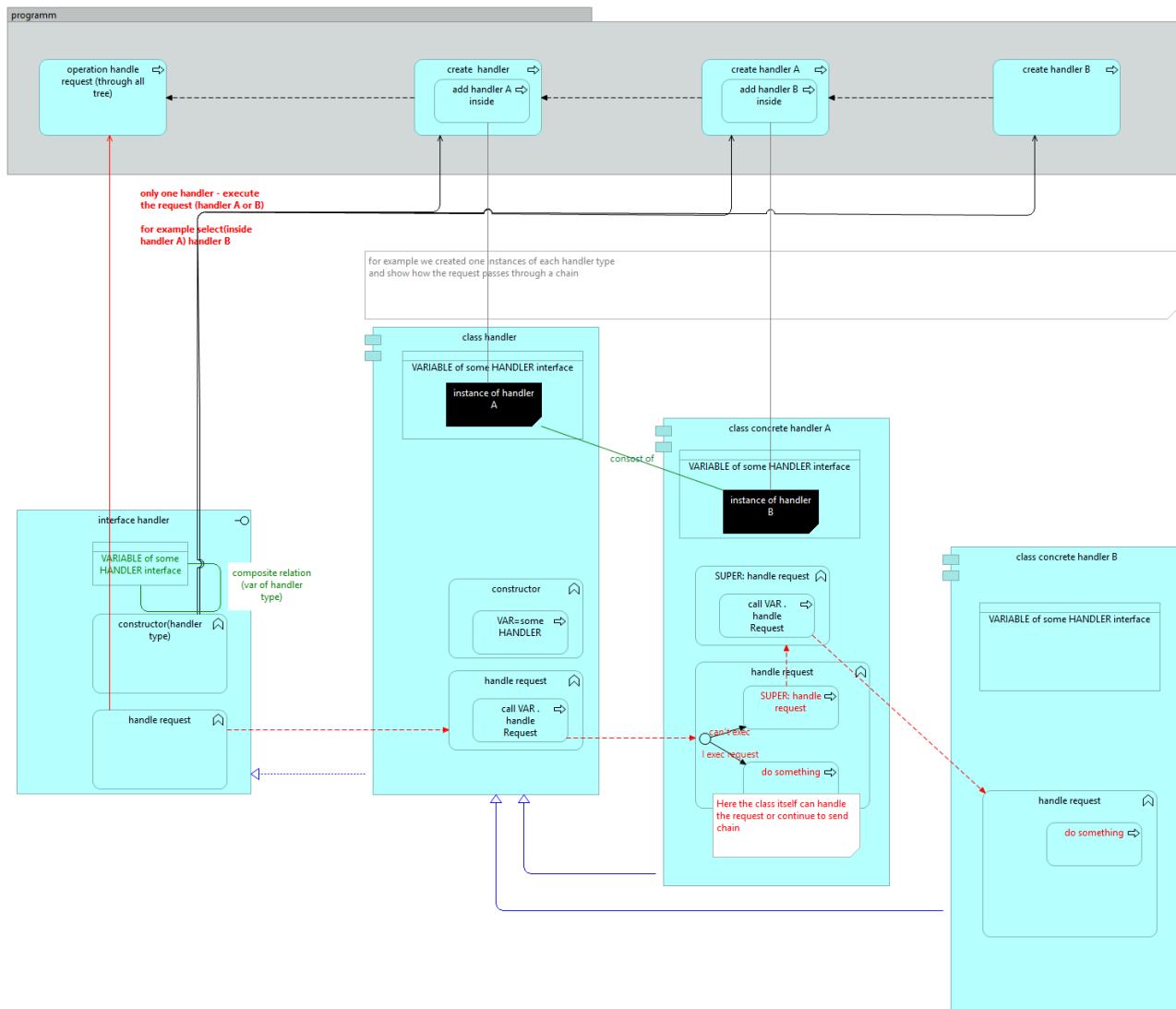
# BEHAVIORAL PATTERNS

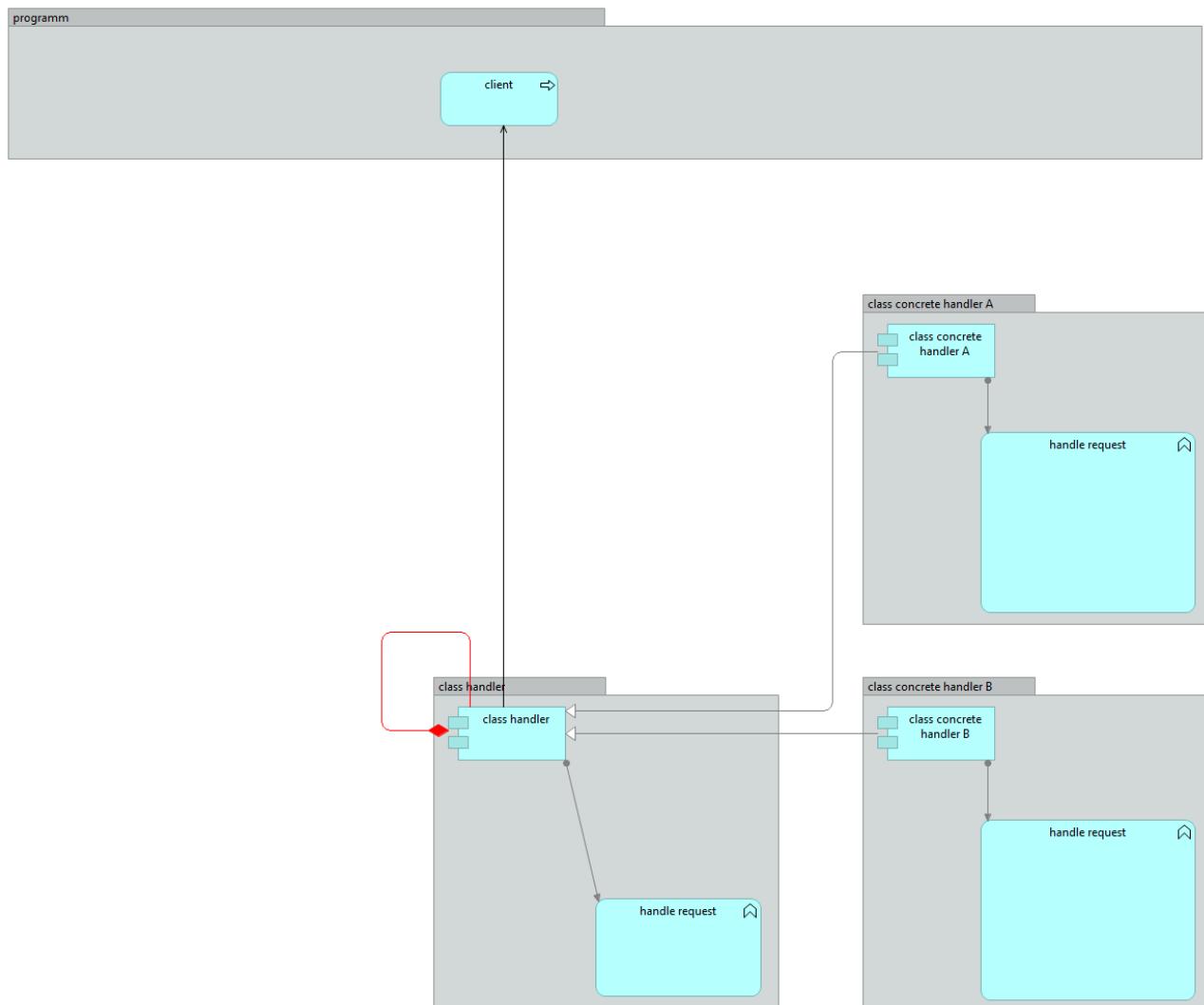
DESIGN PATTERNS

GoF

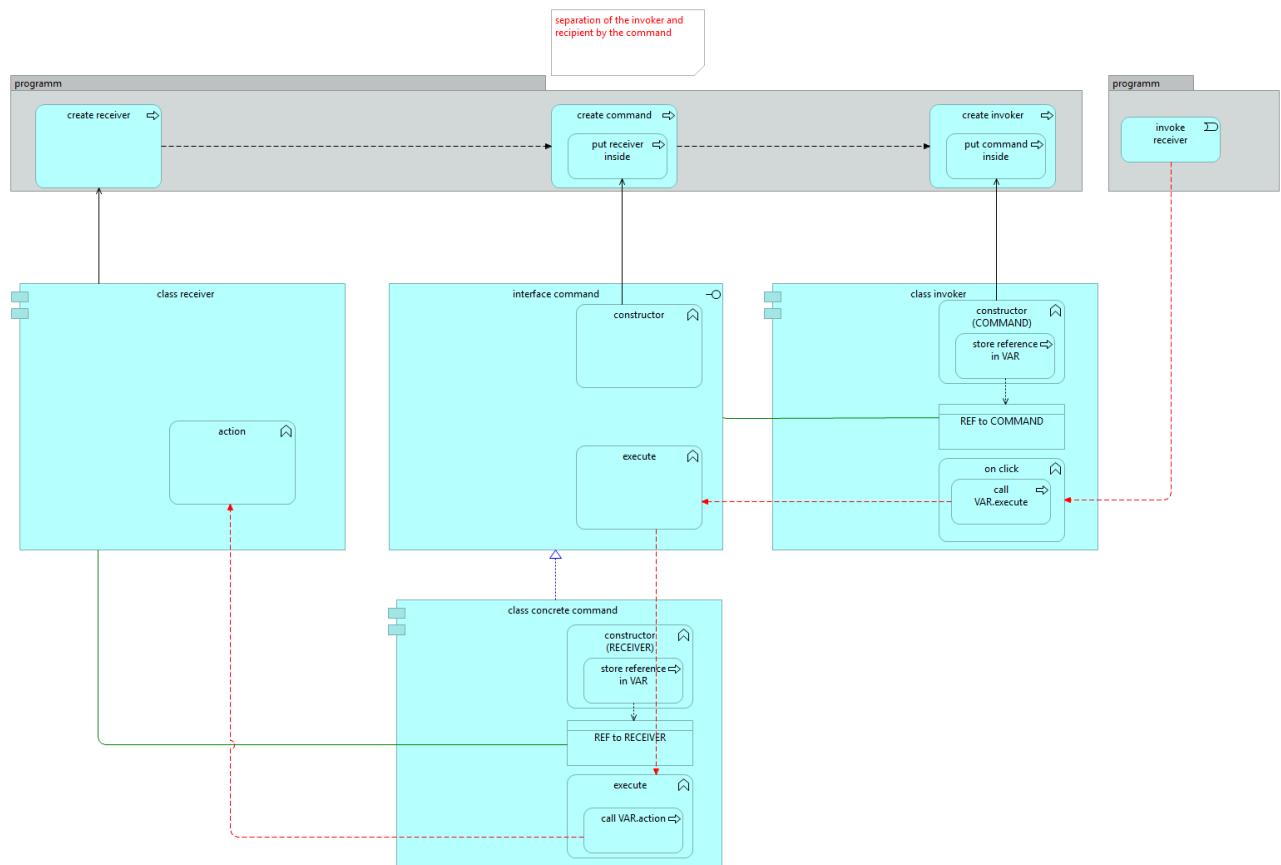


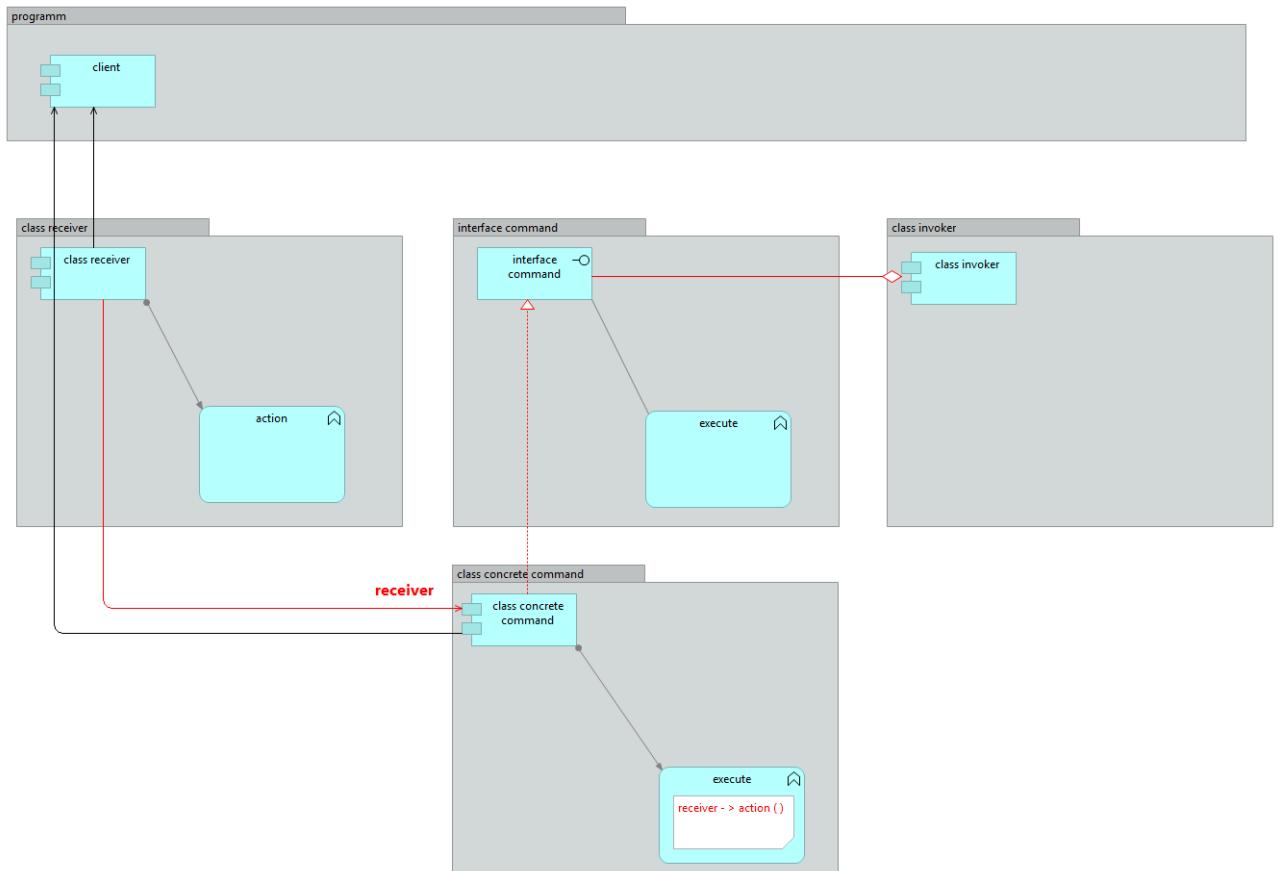
# CHAIN OF RESPONSIBILITY



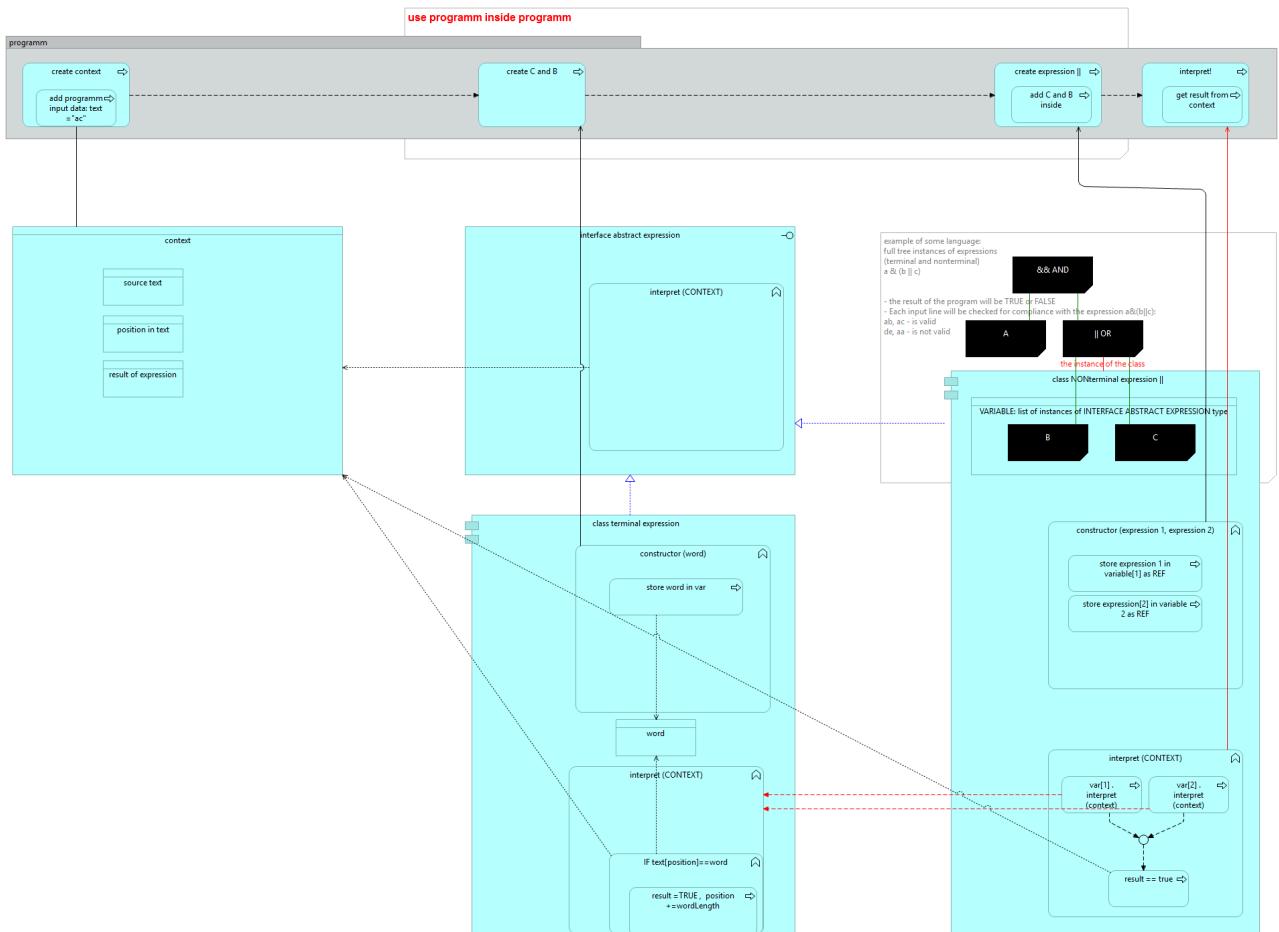


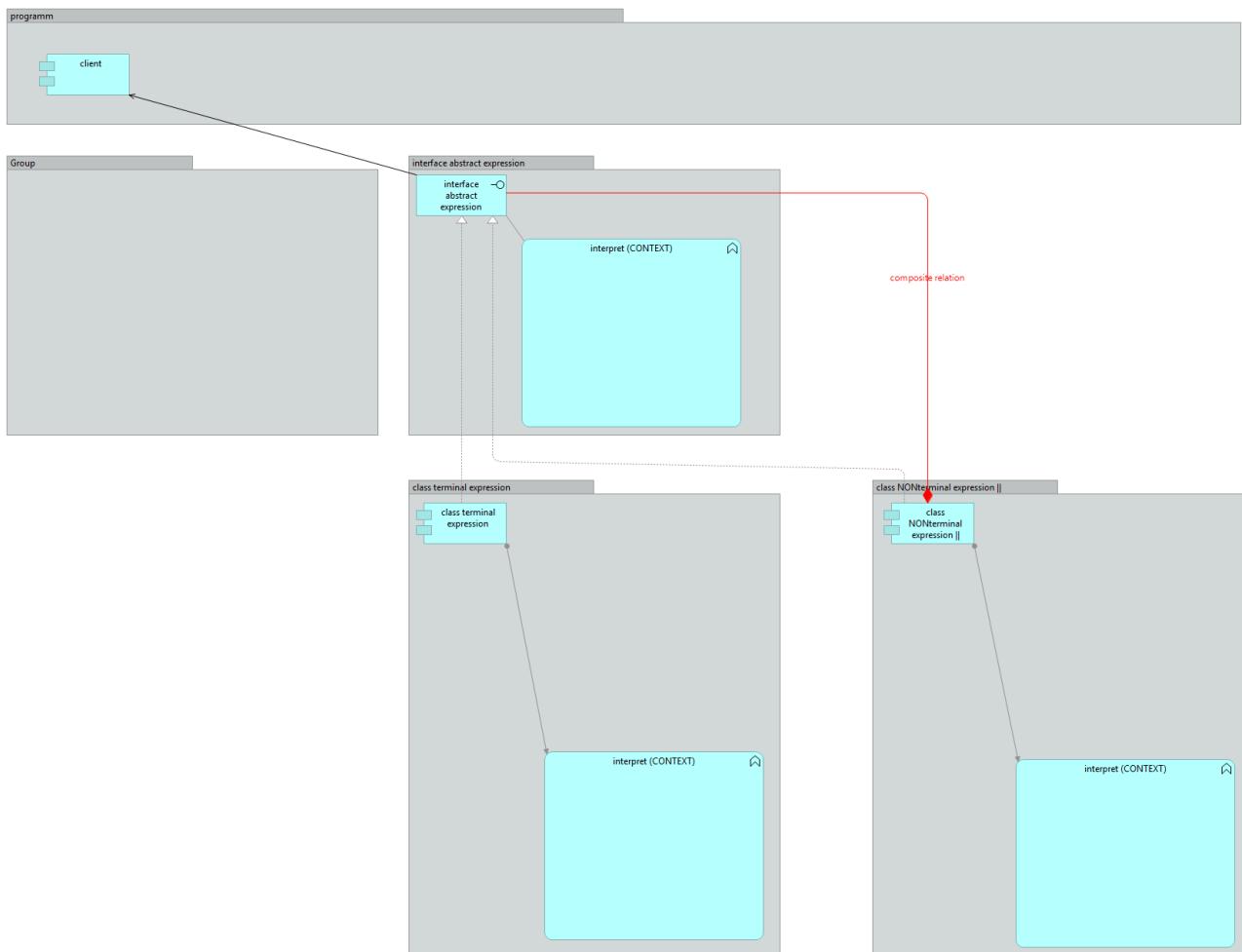
# COMMAND



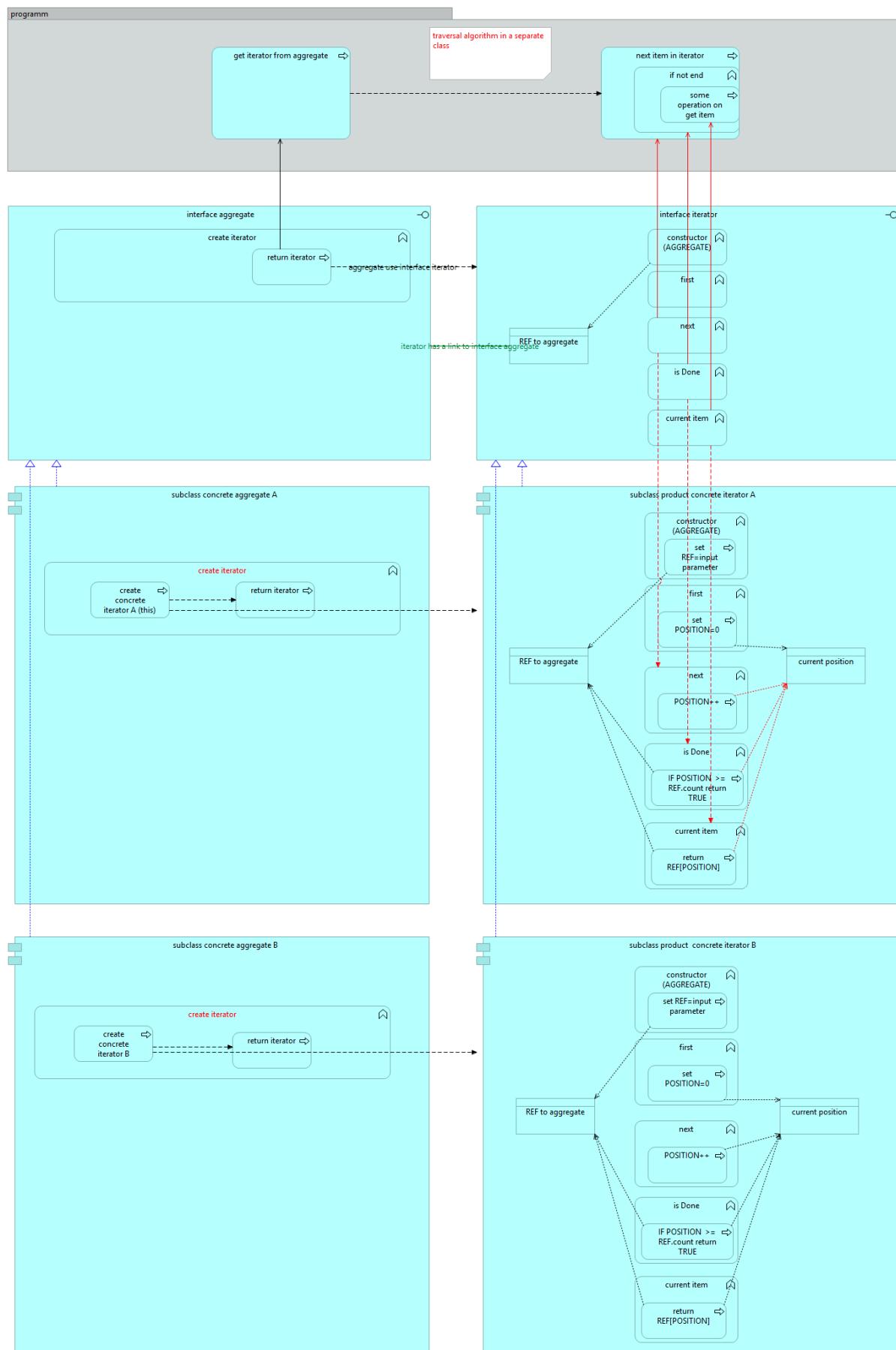


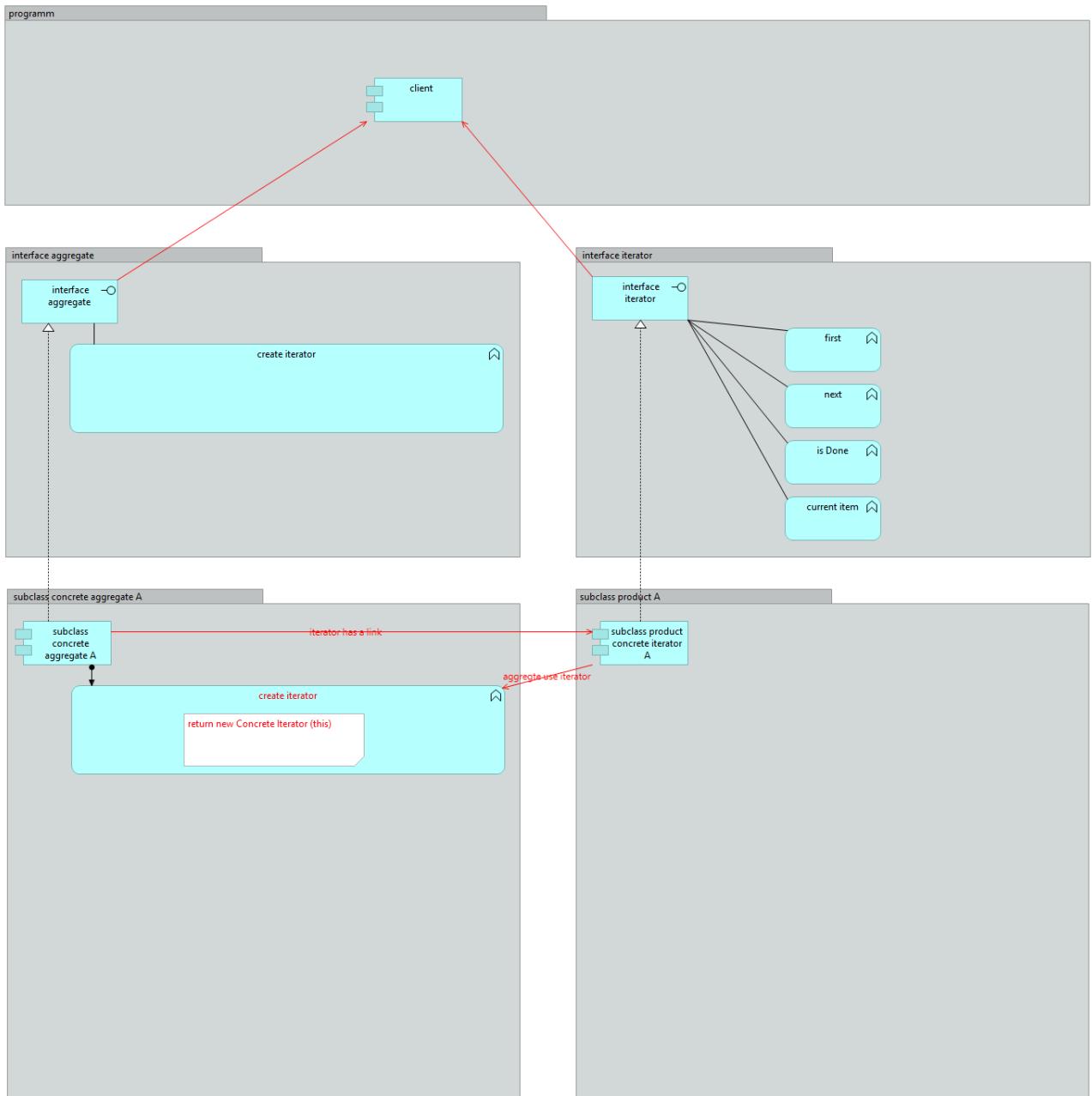
# INTERPRETER



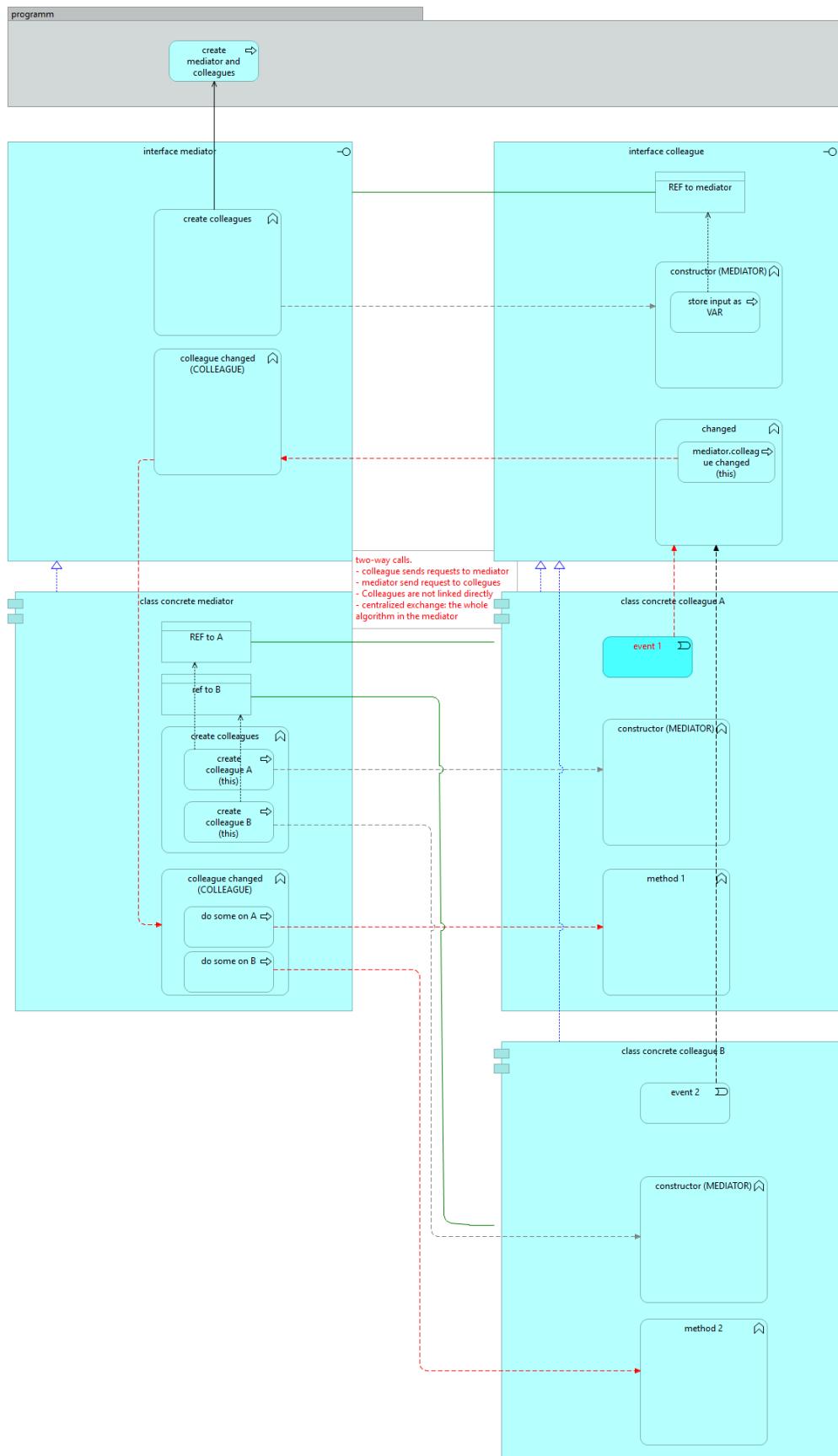


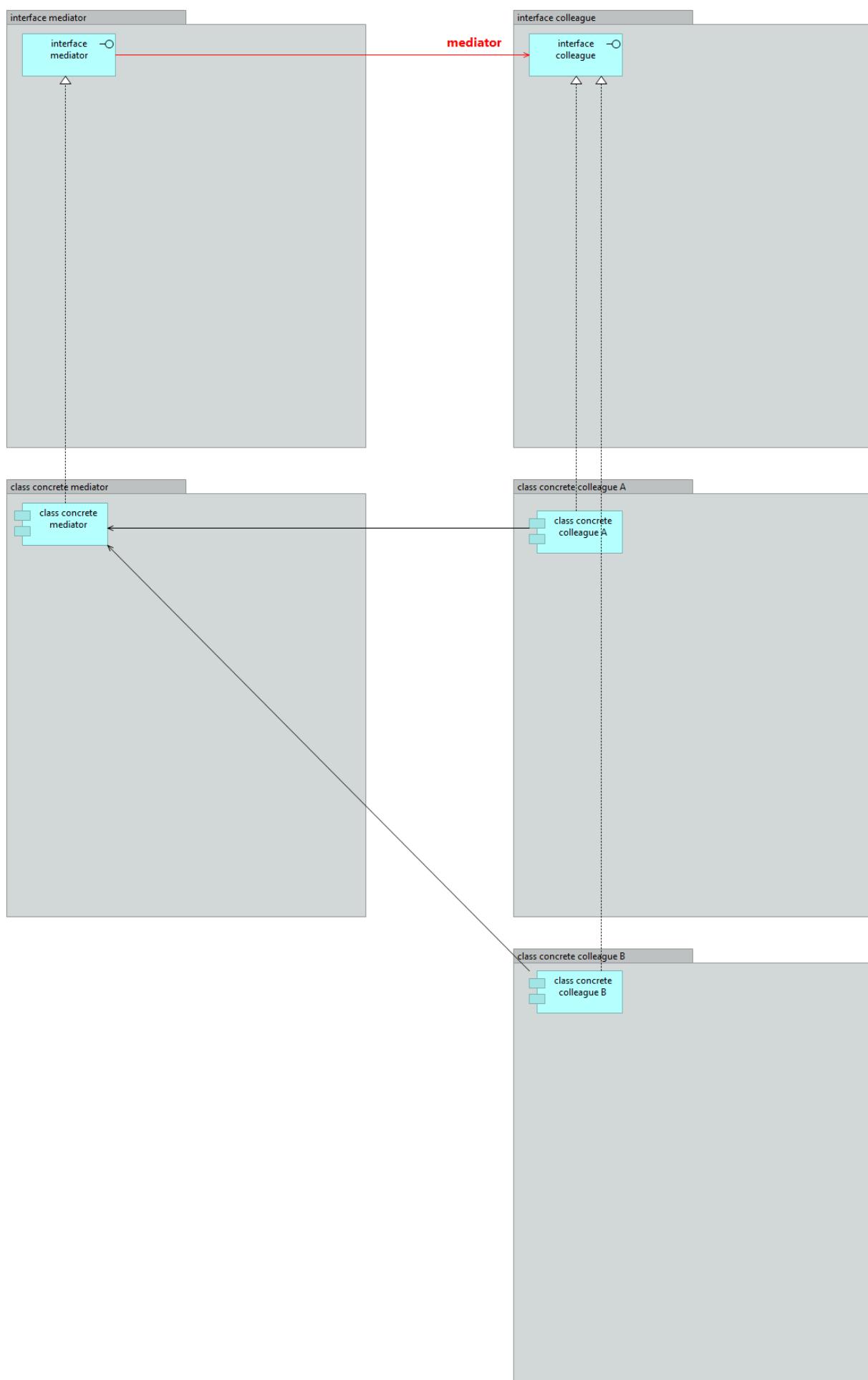
# ITERATOR



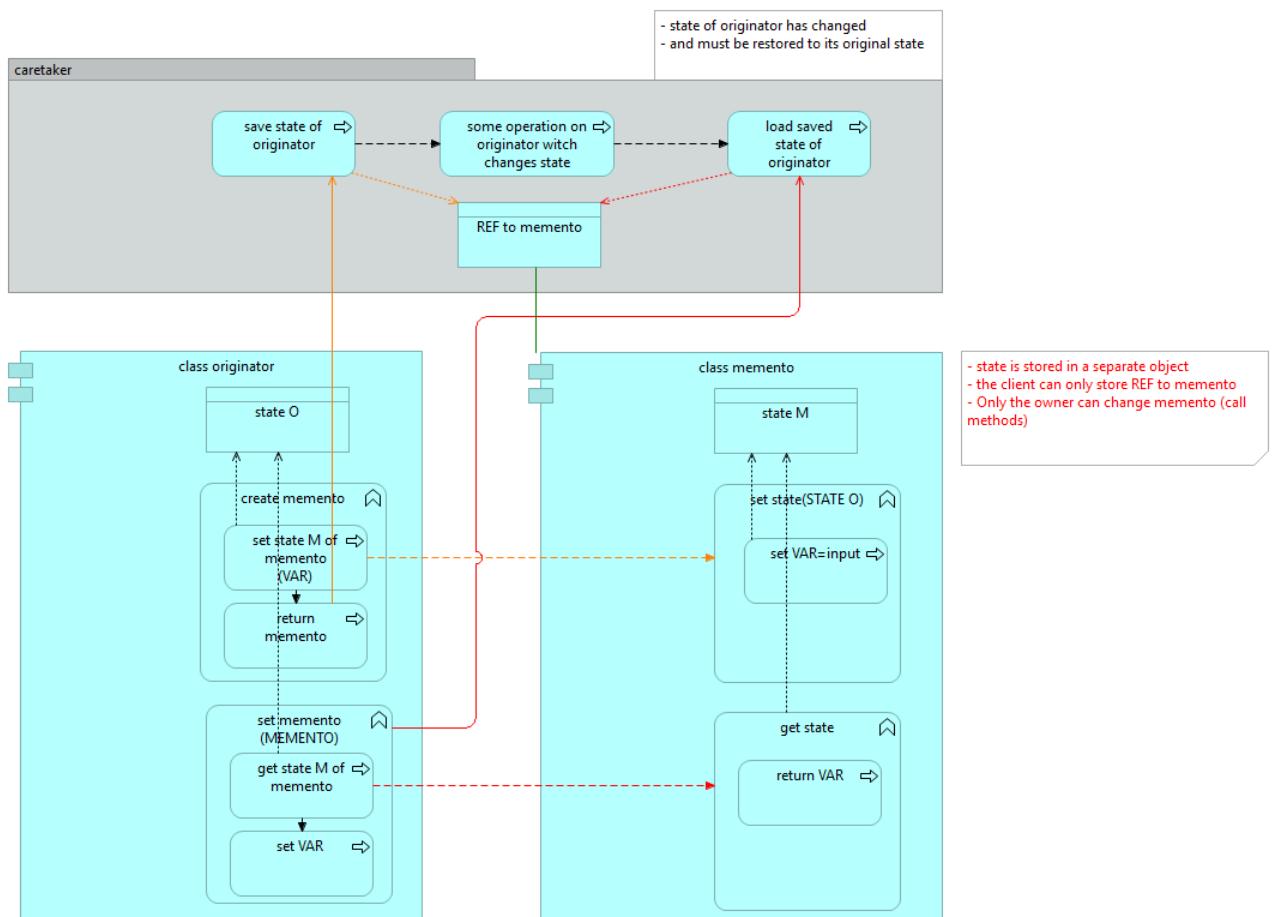


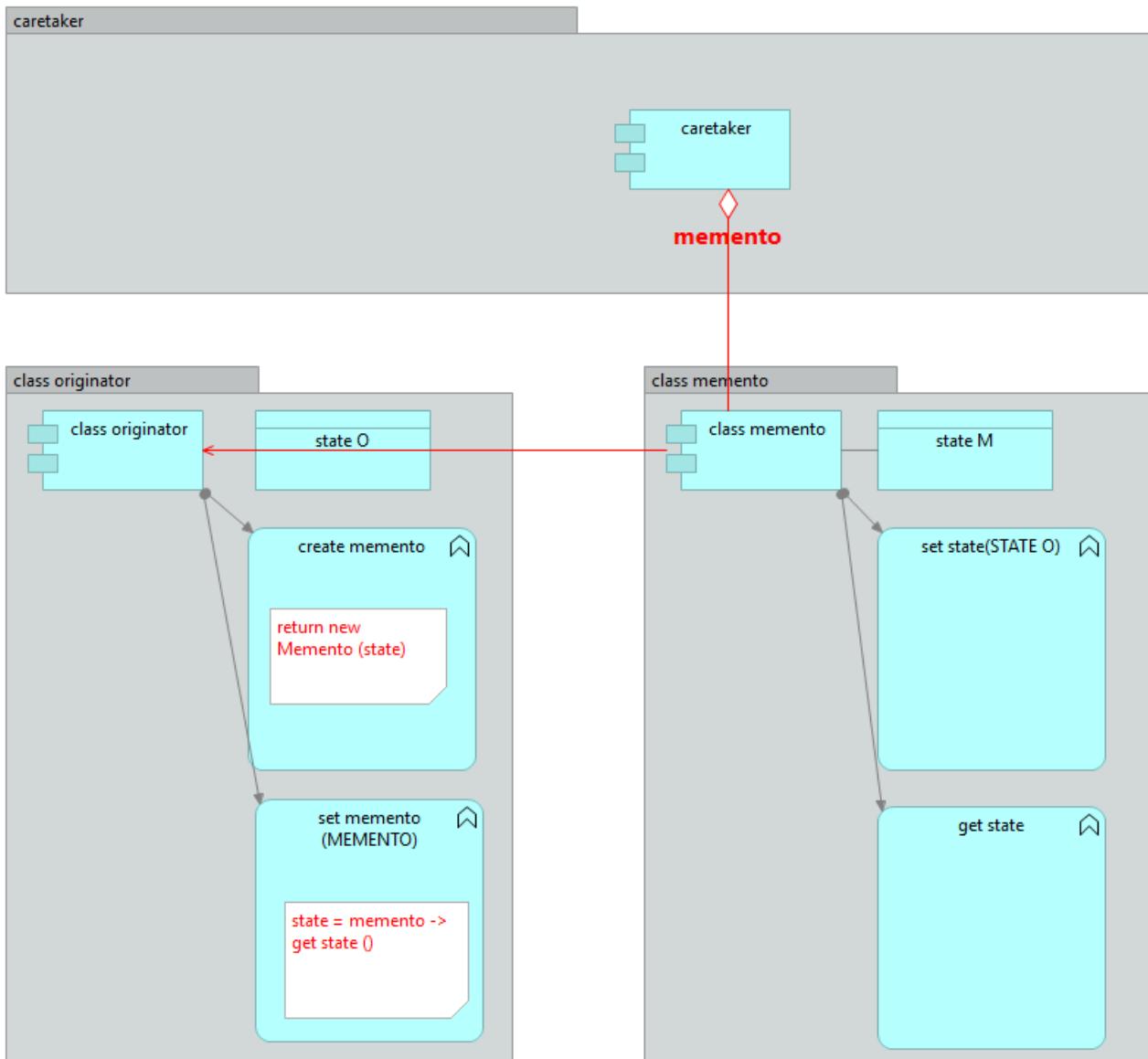
# MEDIATOR



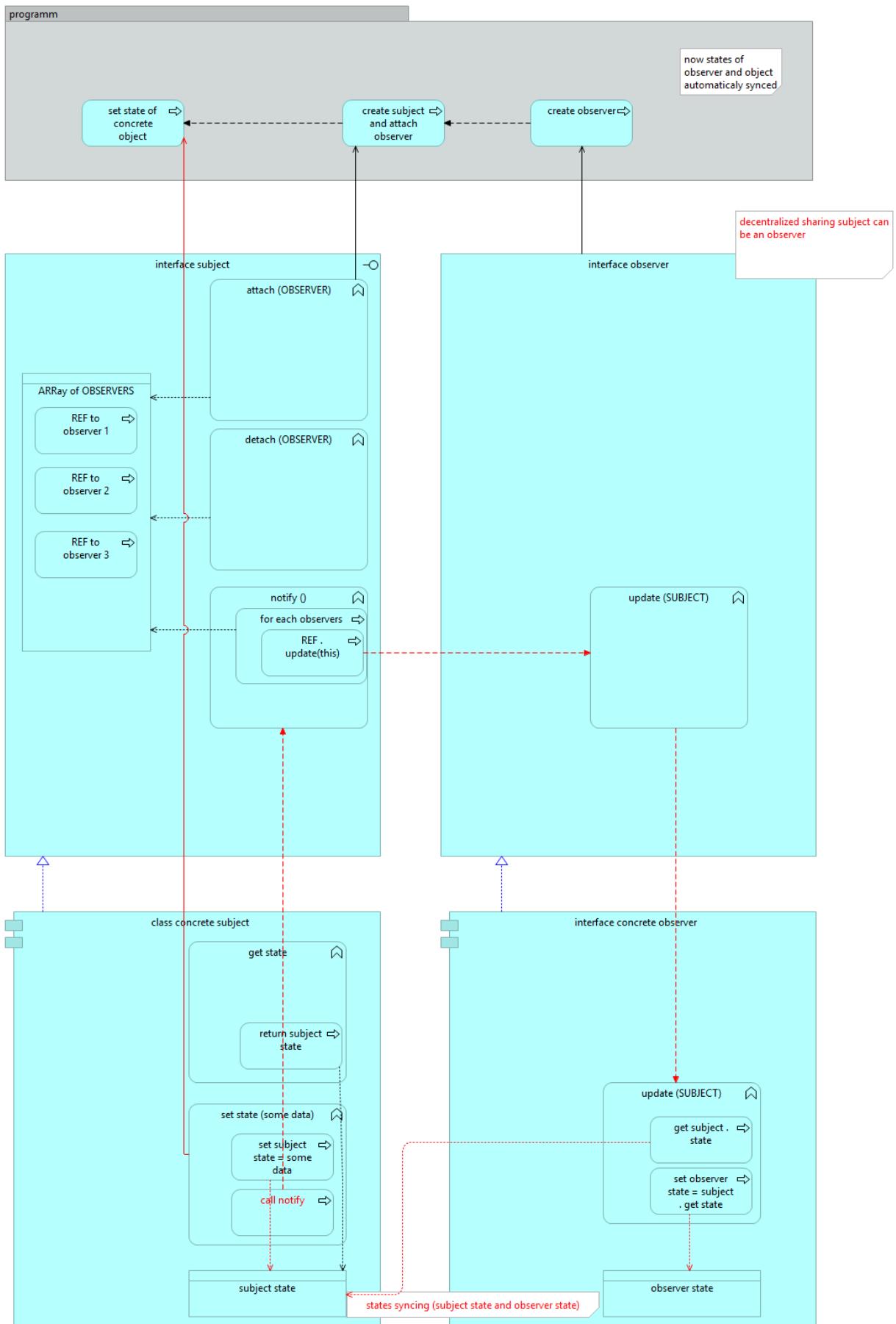


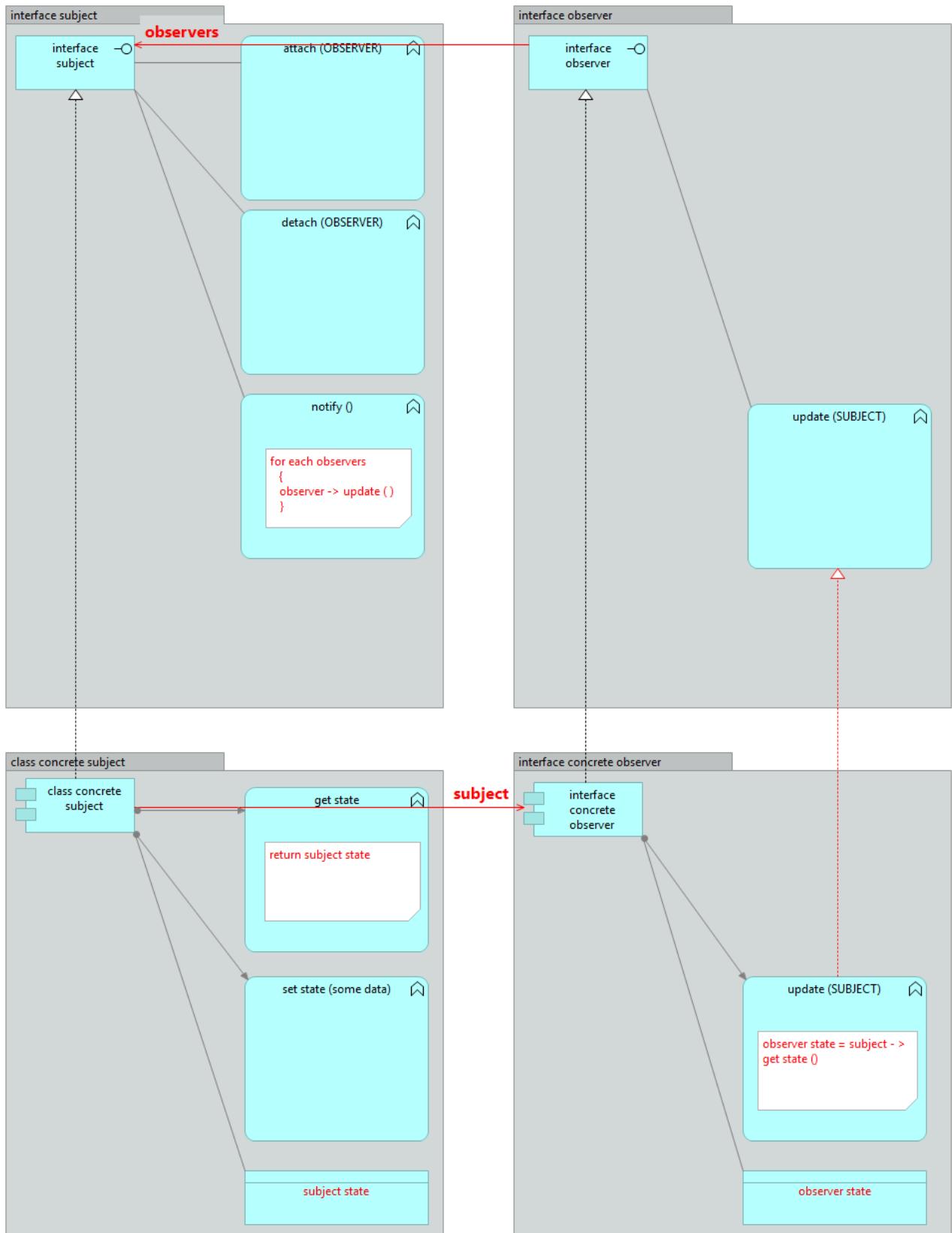
# MEMENTO



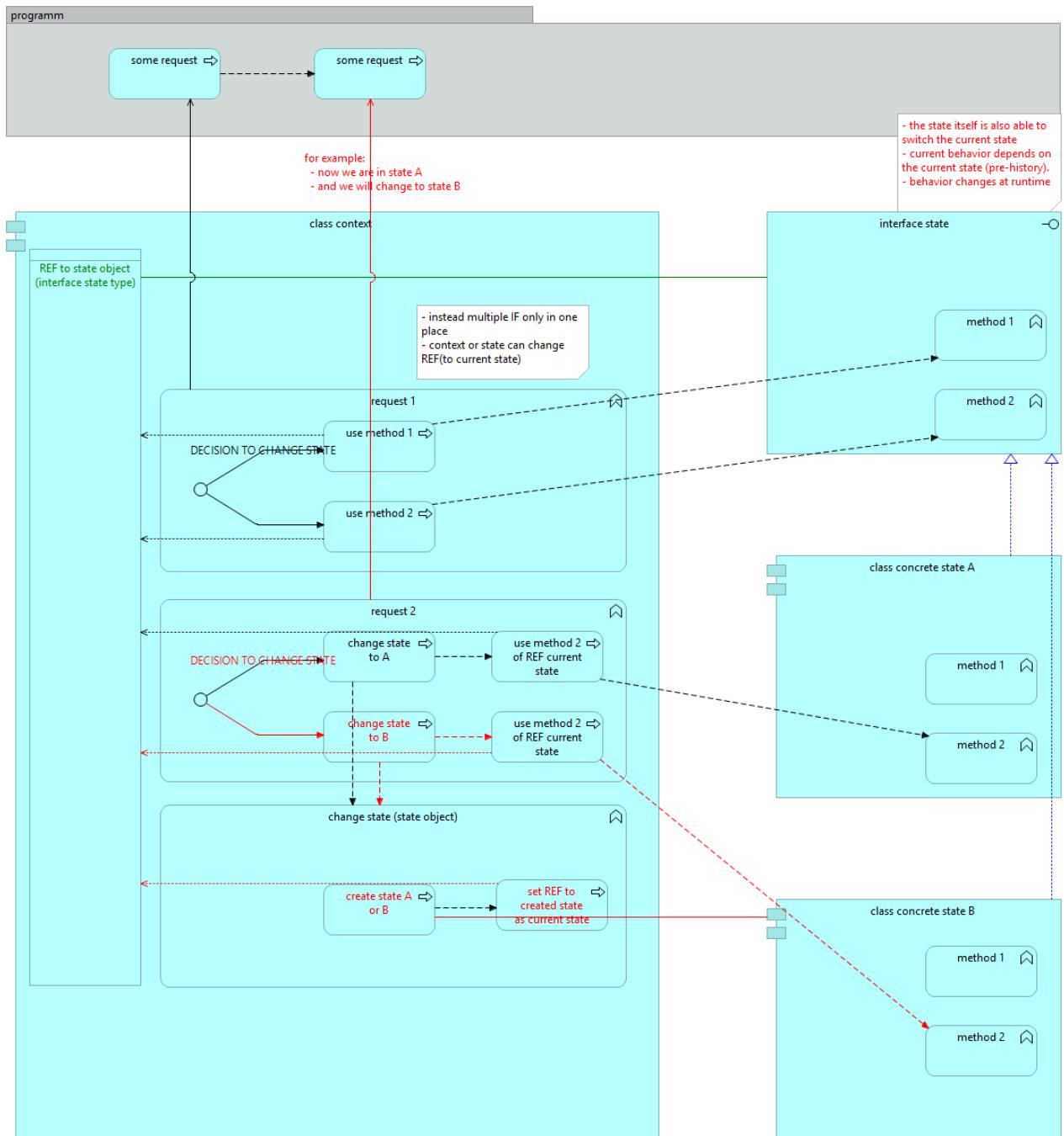


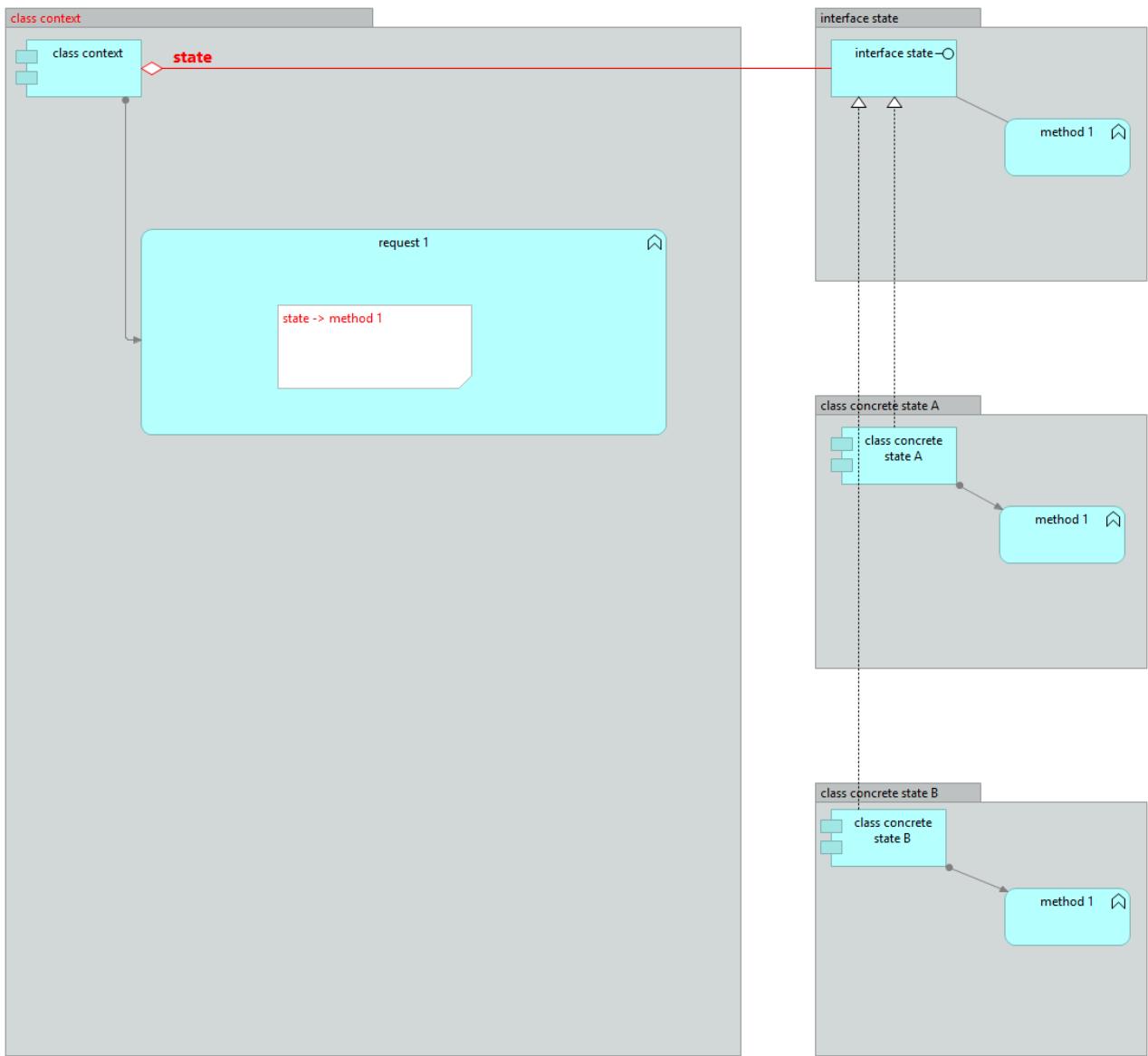
# OBSERVER



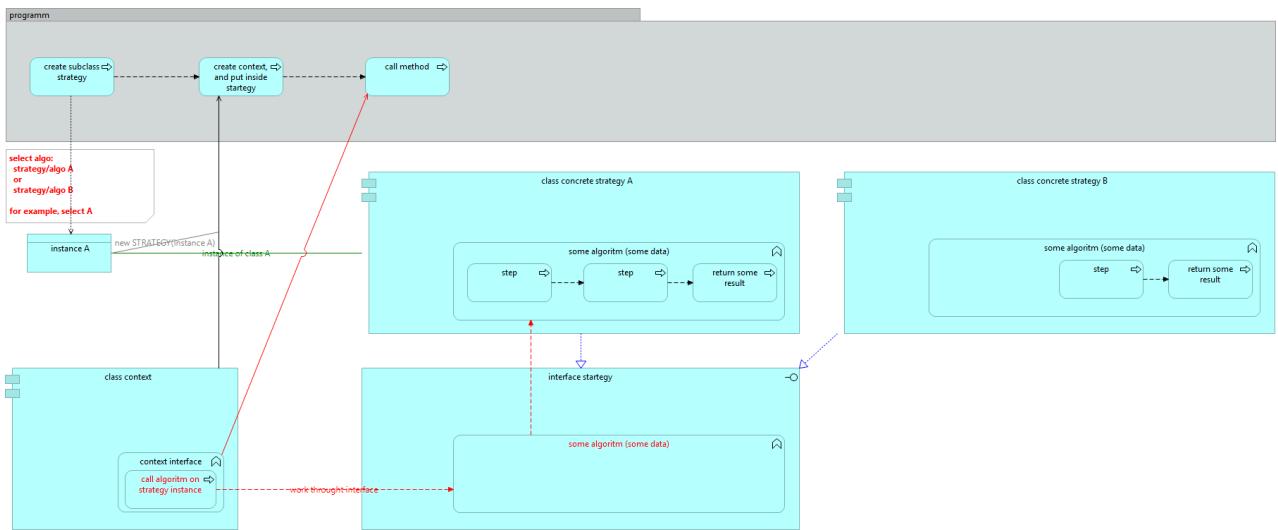


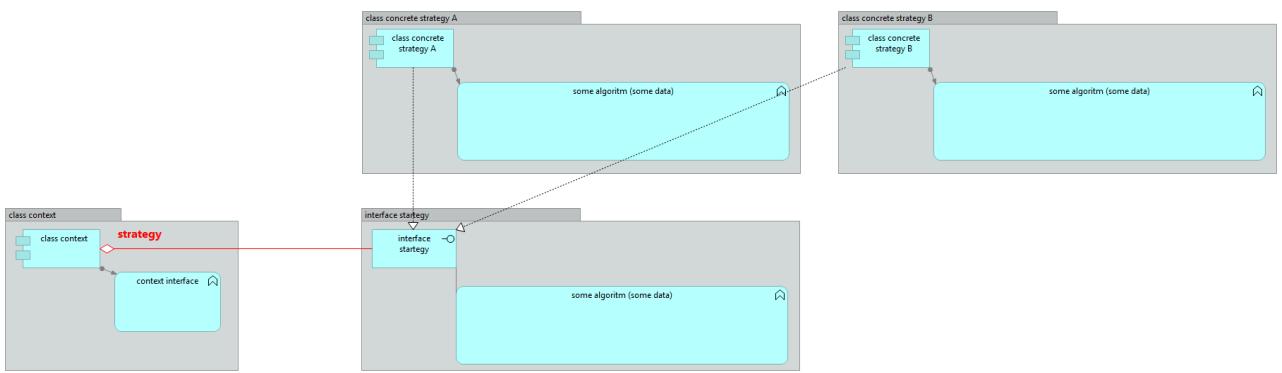
# STATE



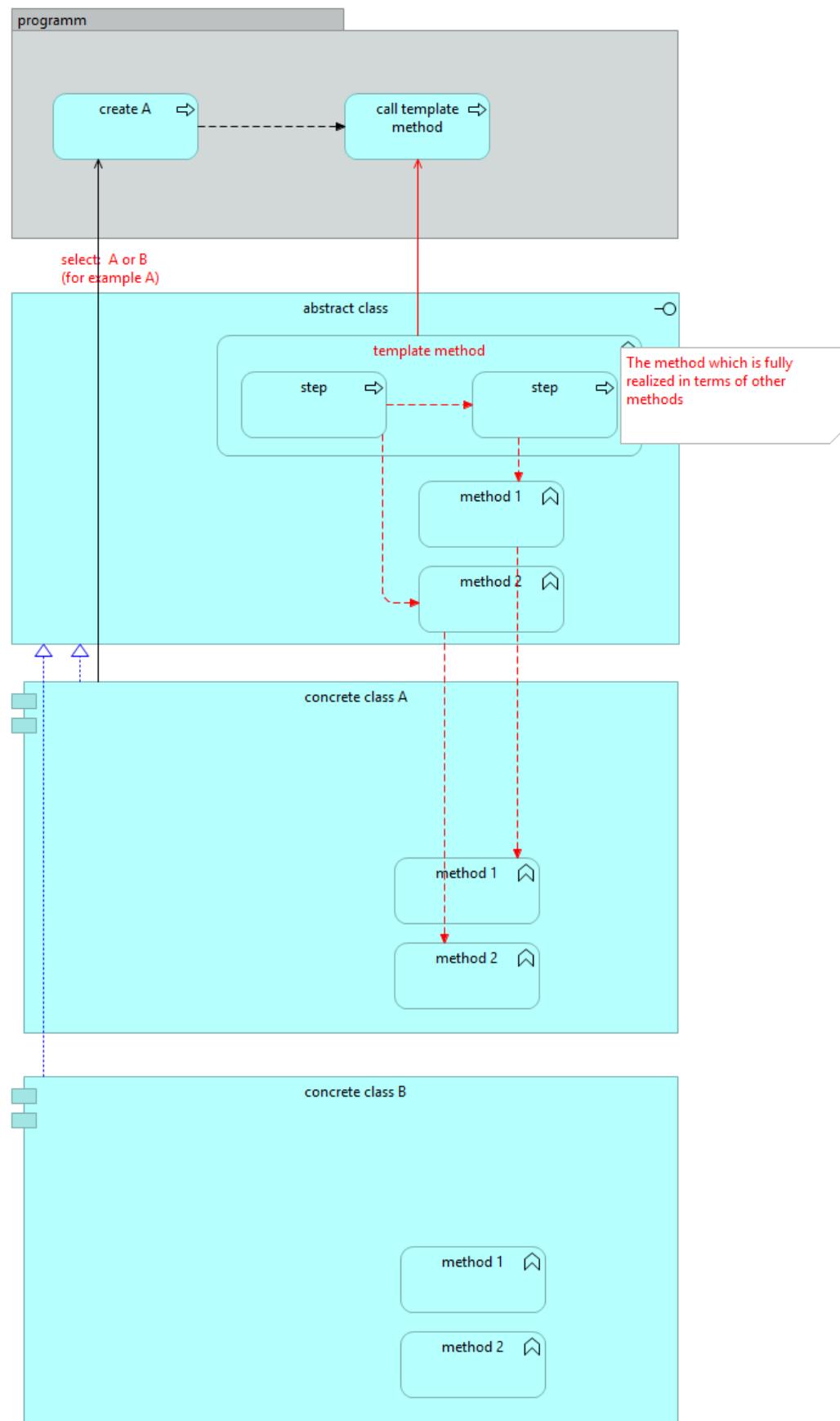


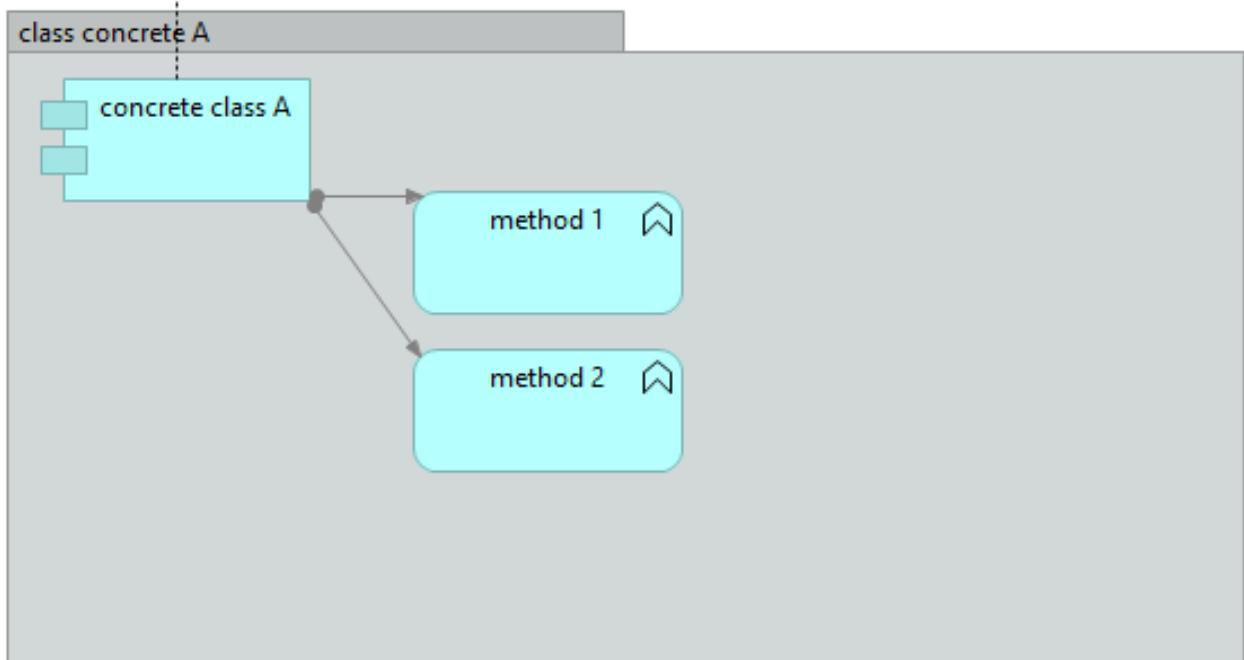
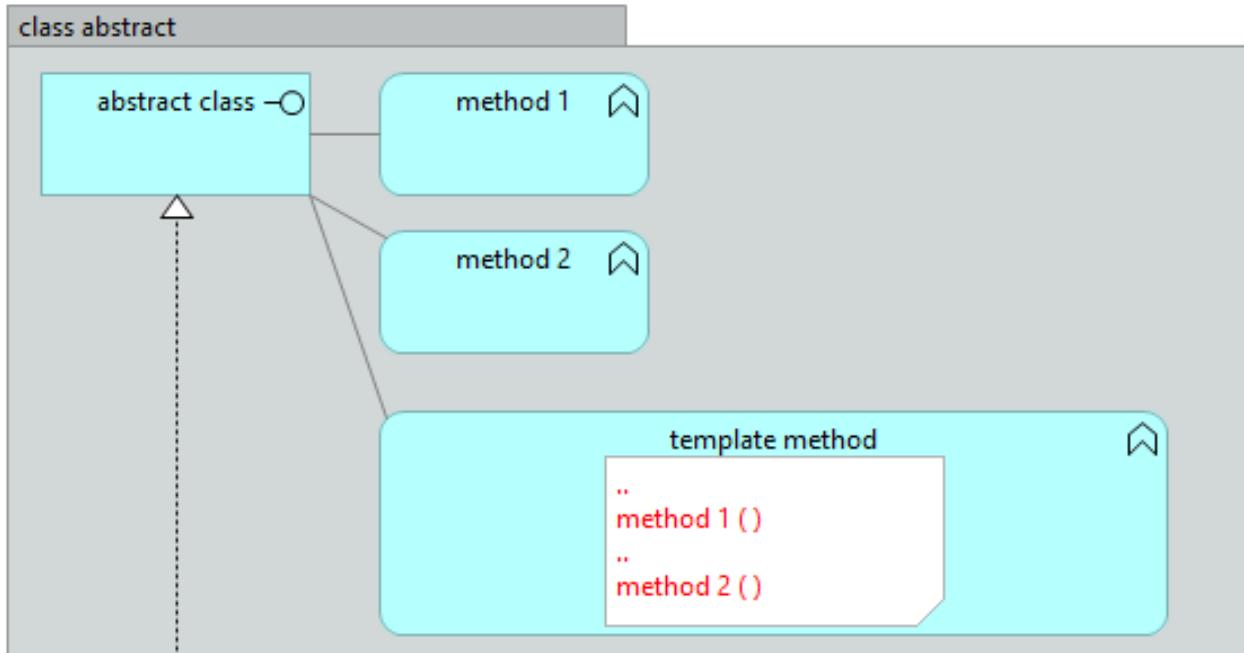
# STRATEGY

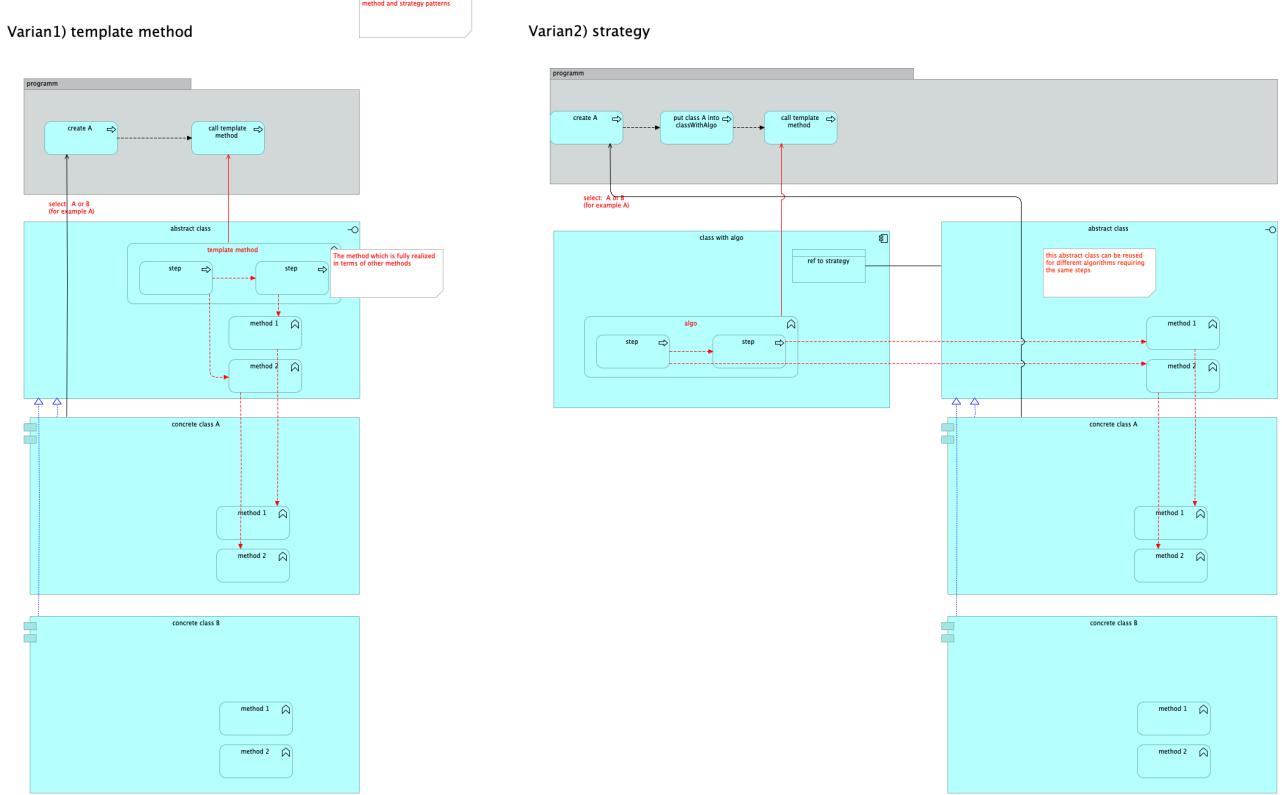




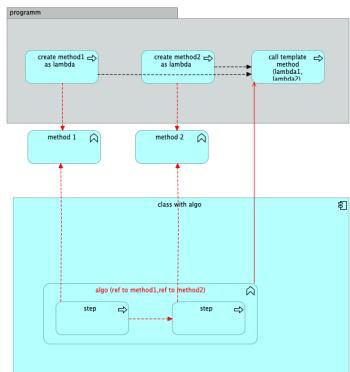
# TEMPLATE METHOD





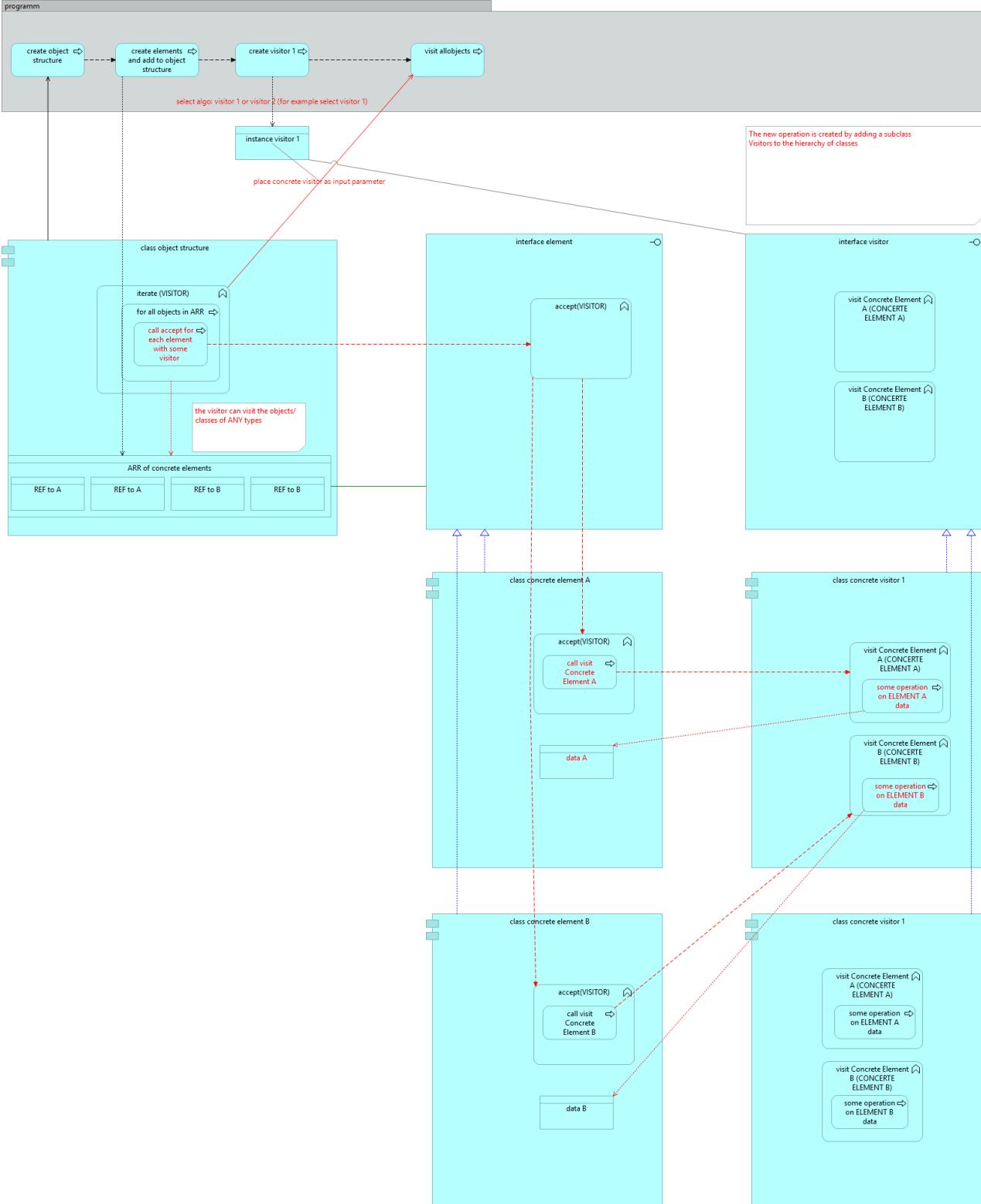


**Varian3) lambdas as input parameters**



# VISITOR





02

# Enterprise Patterns

Catalog of Patterns of Enterprise  
Application Architecture

MARTIN FOWLER



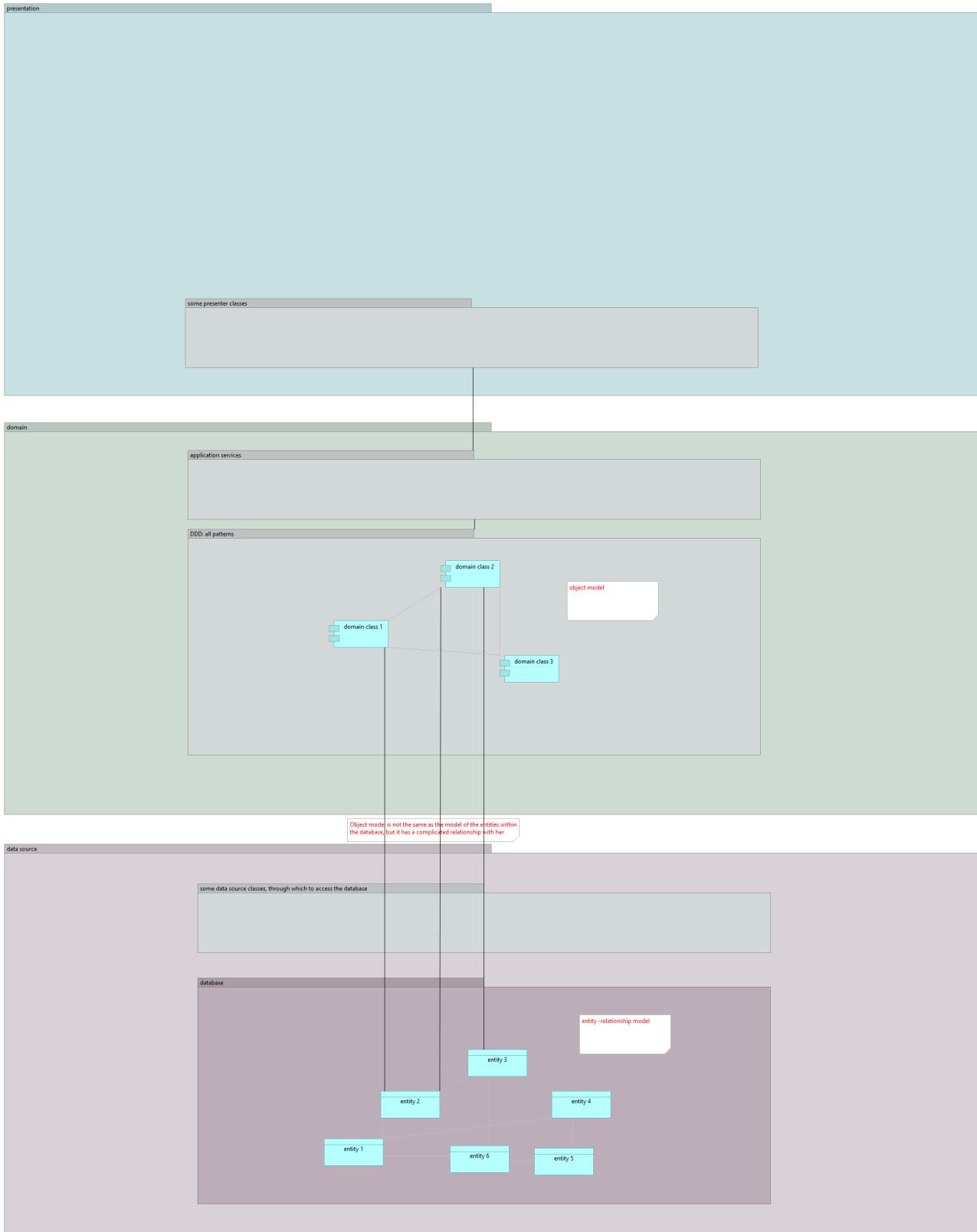
# BUSINESS LOGIC

ENTERPRISE PATTERNS

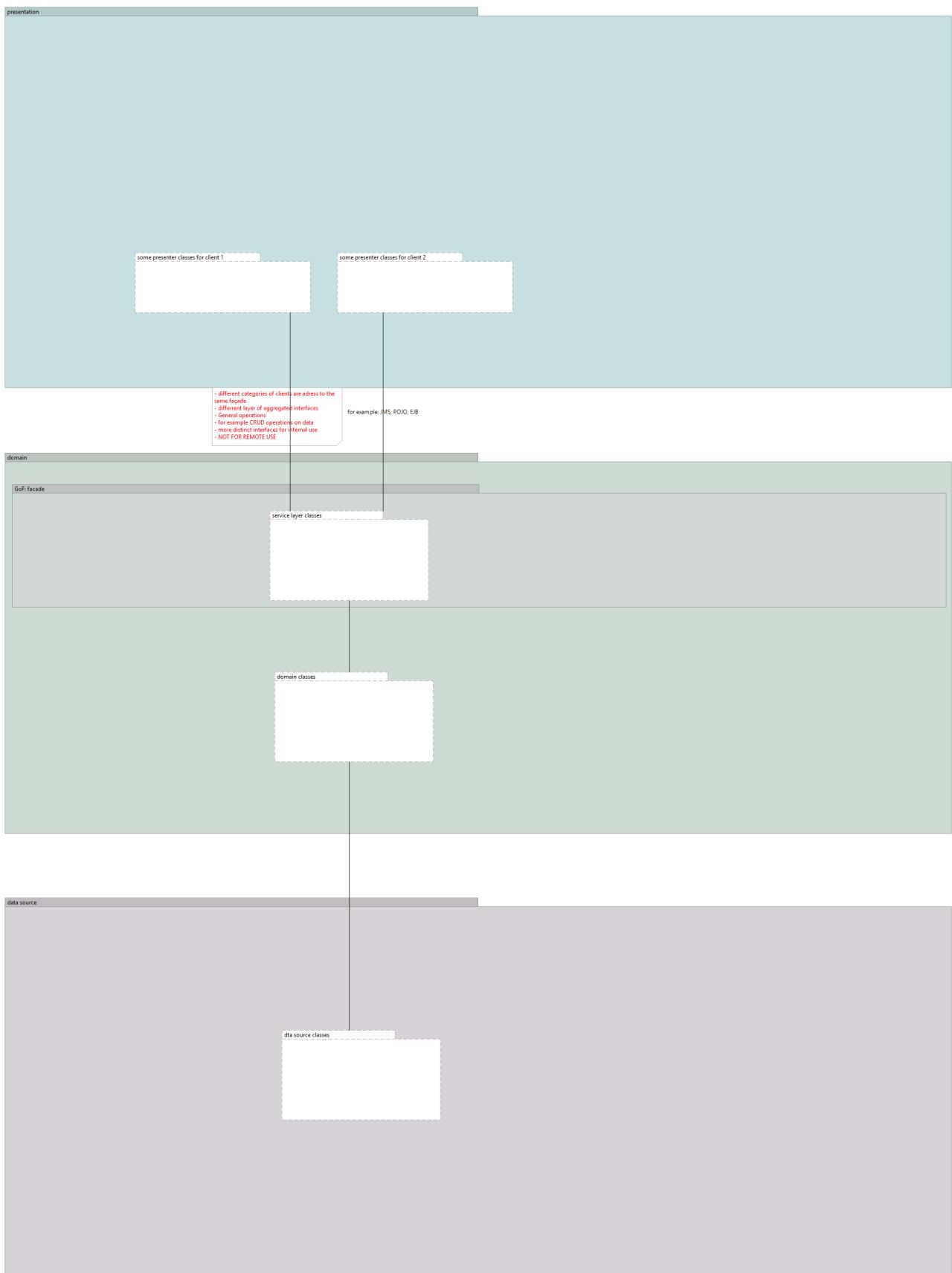
Martin Fowler



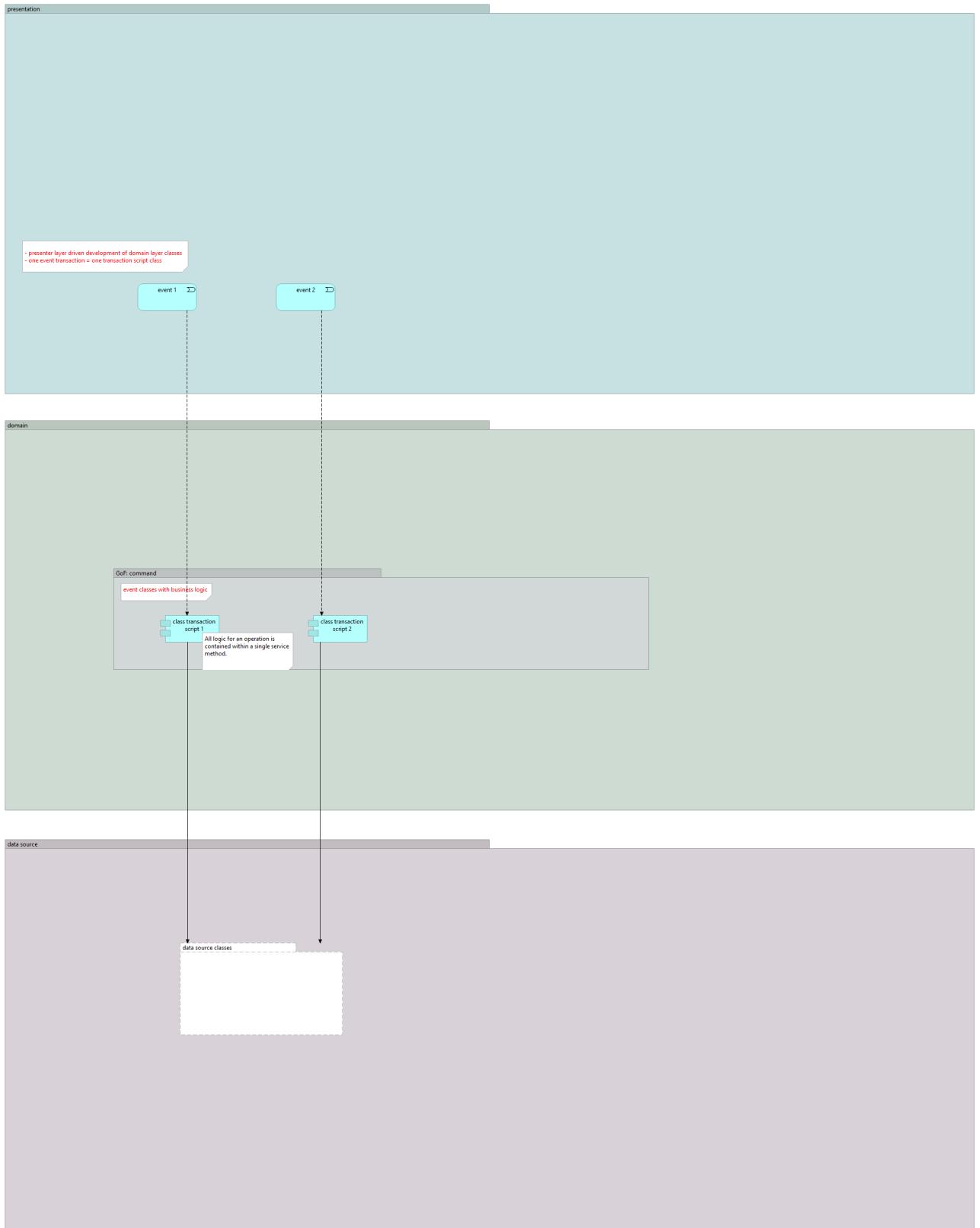
# DOMAIN MODEL



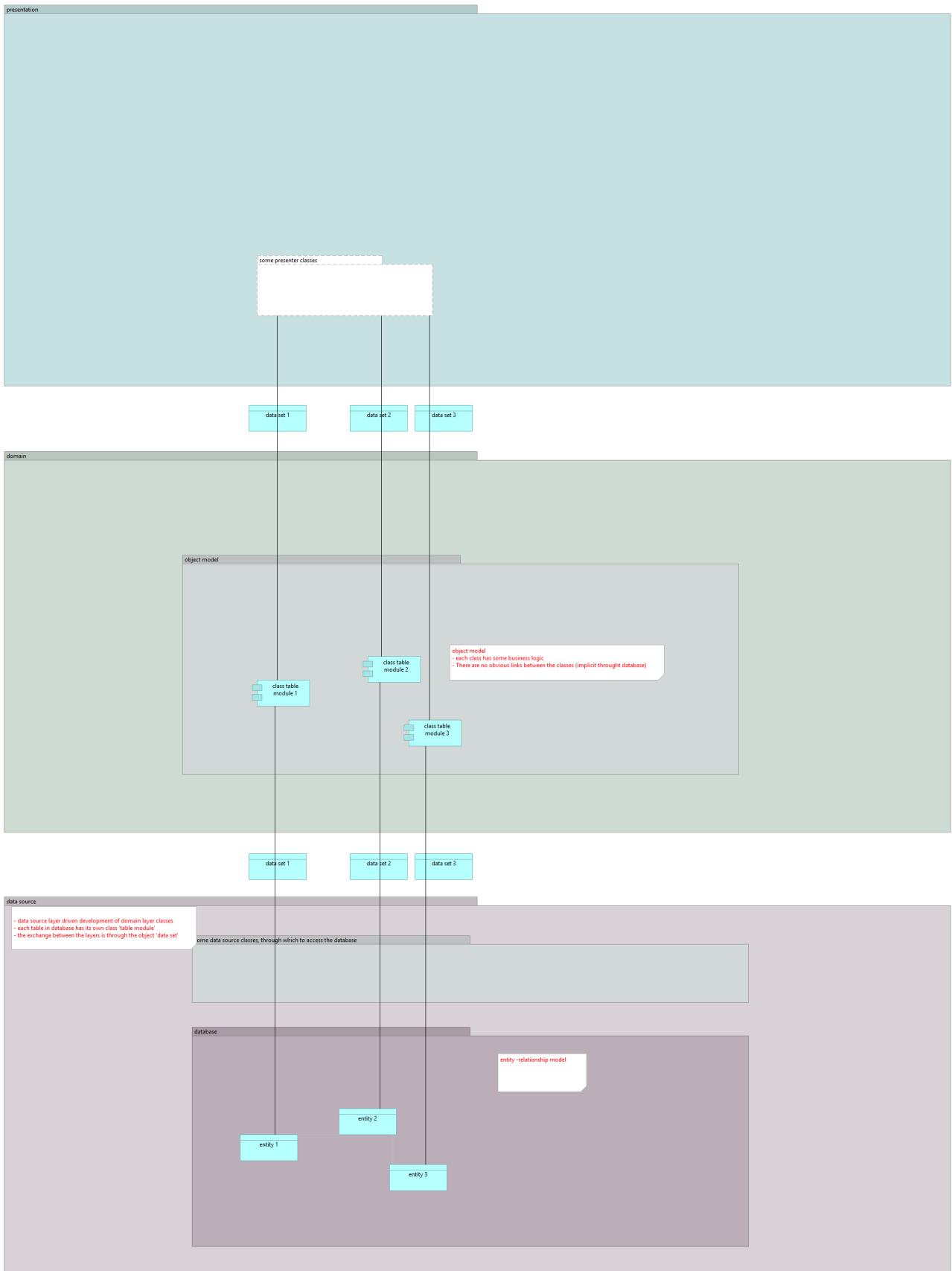
# SERVICE LAYER



# TRANSACTION SCRIPT



# TABLE MODULE



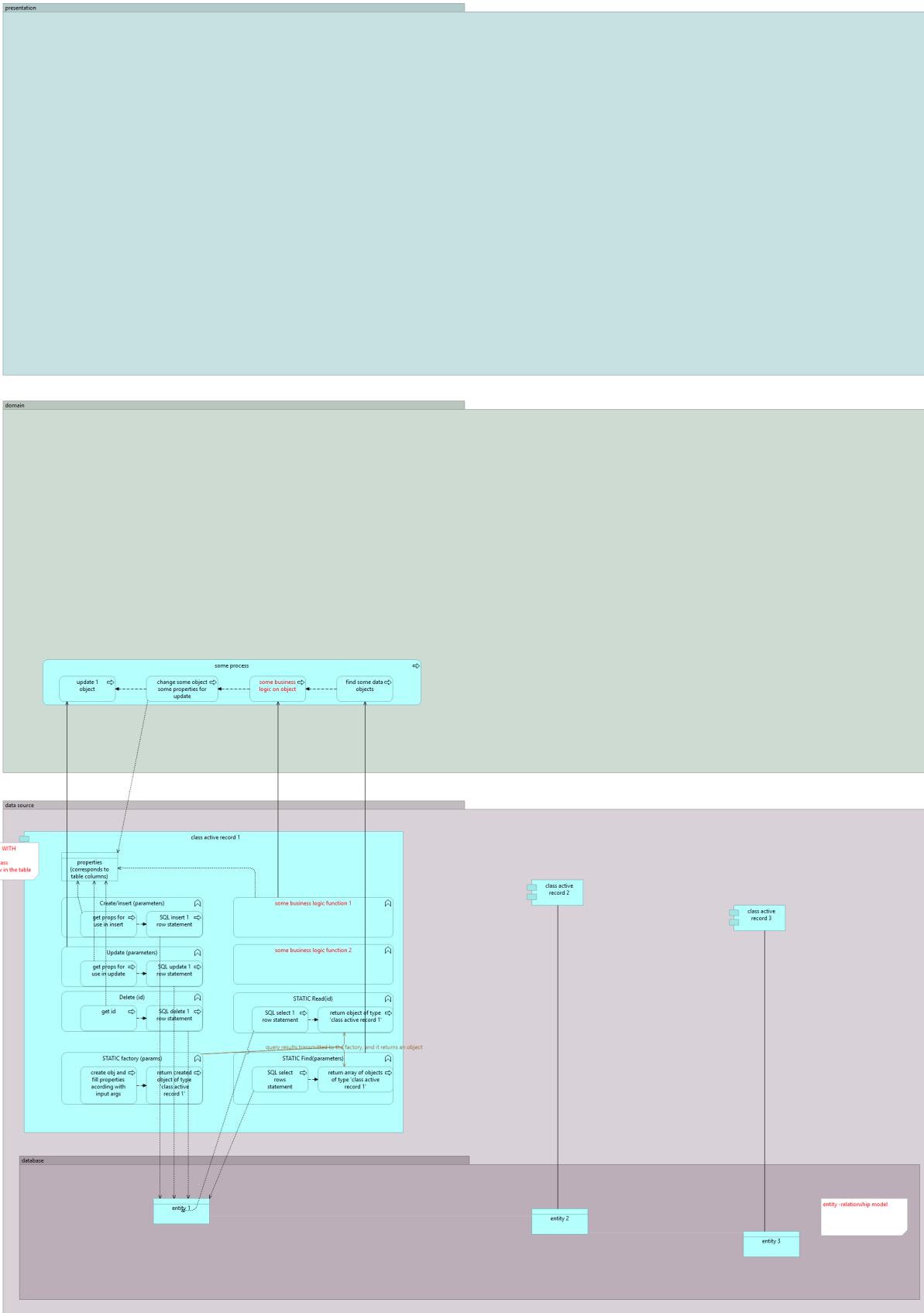
# DATA SOURCES

ENTERPRISE PATTERNS

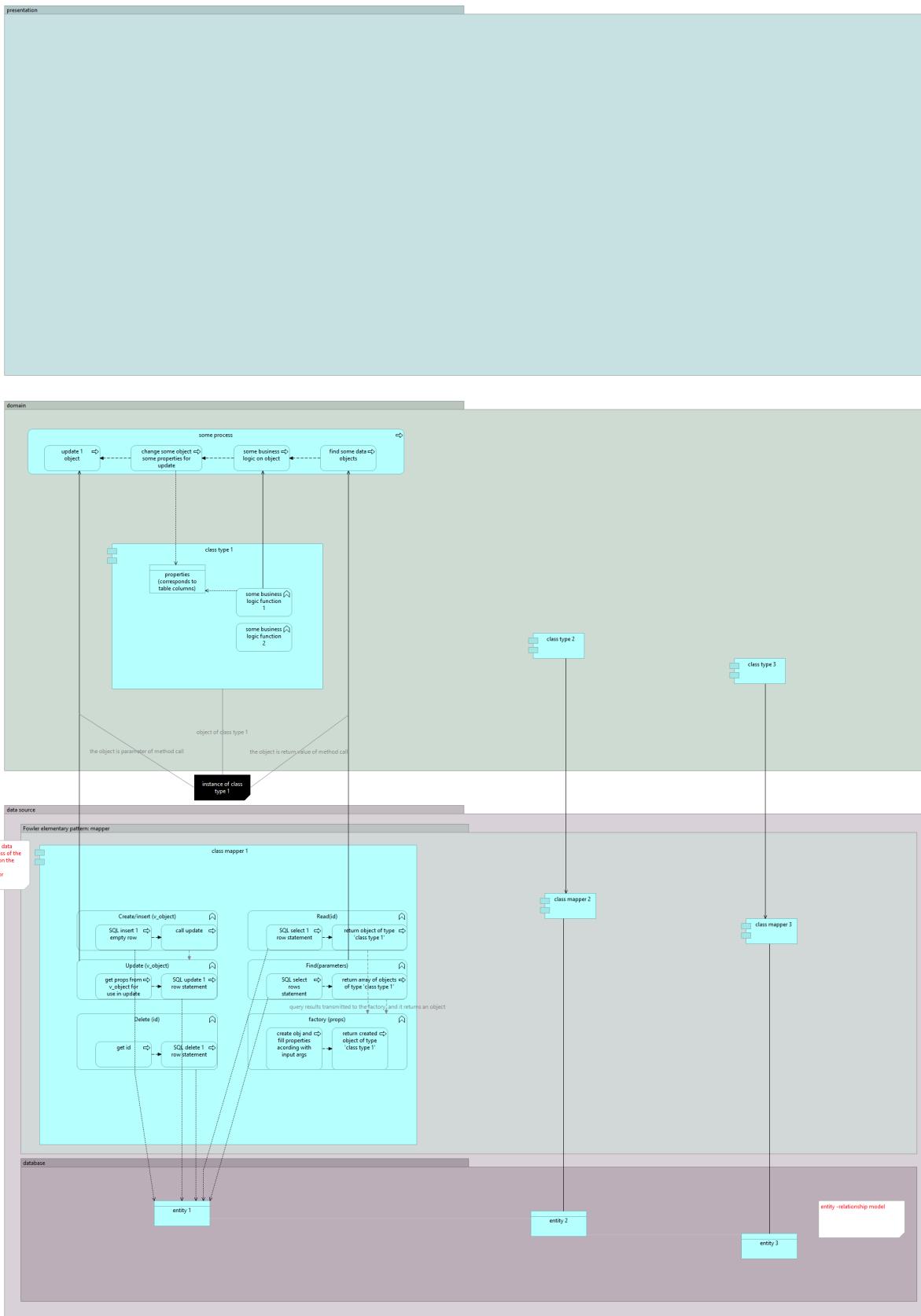
Martin Fowler



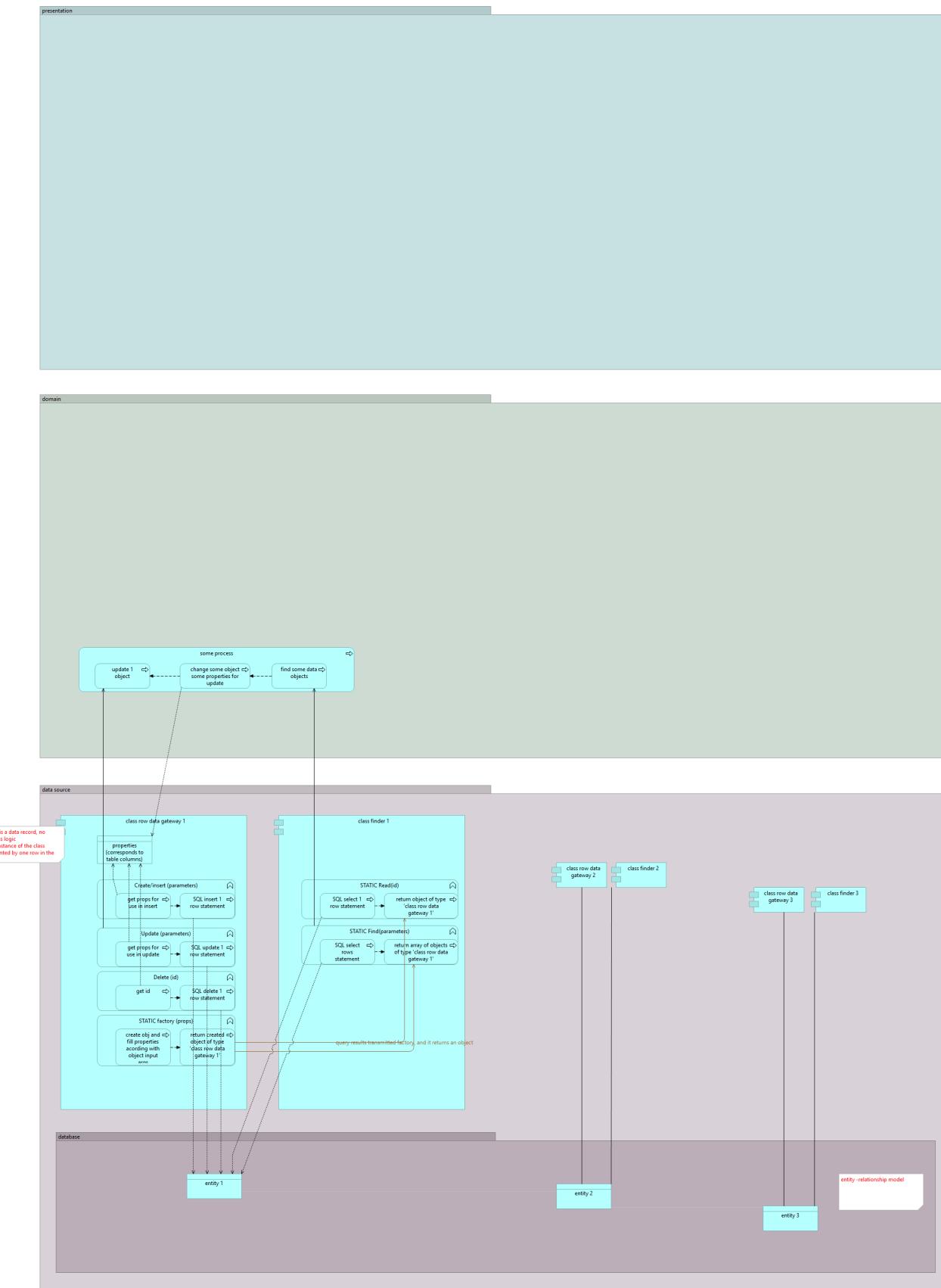
# ACTIVE RECORD



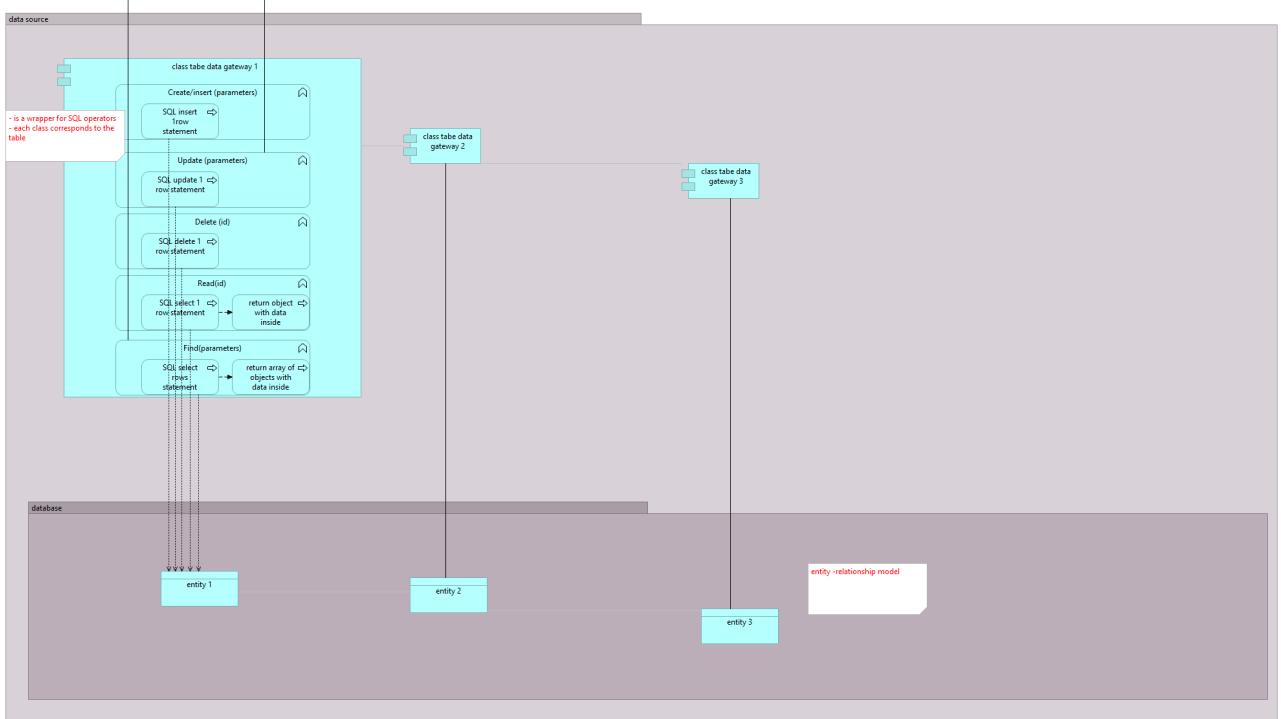
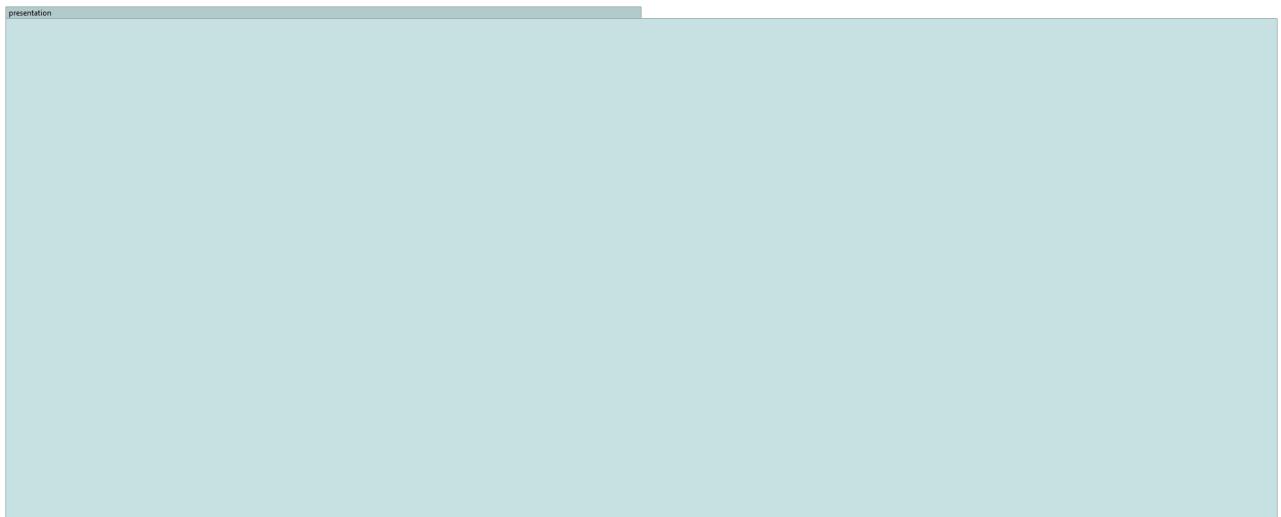
# DATA MAPPER



# ROW DATA GATEWAY



# TABLE DATA GATEWAY



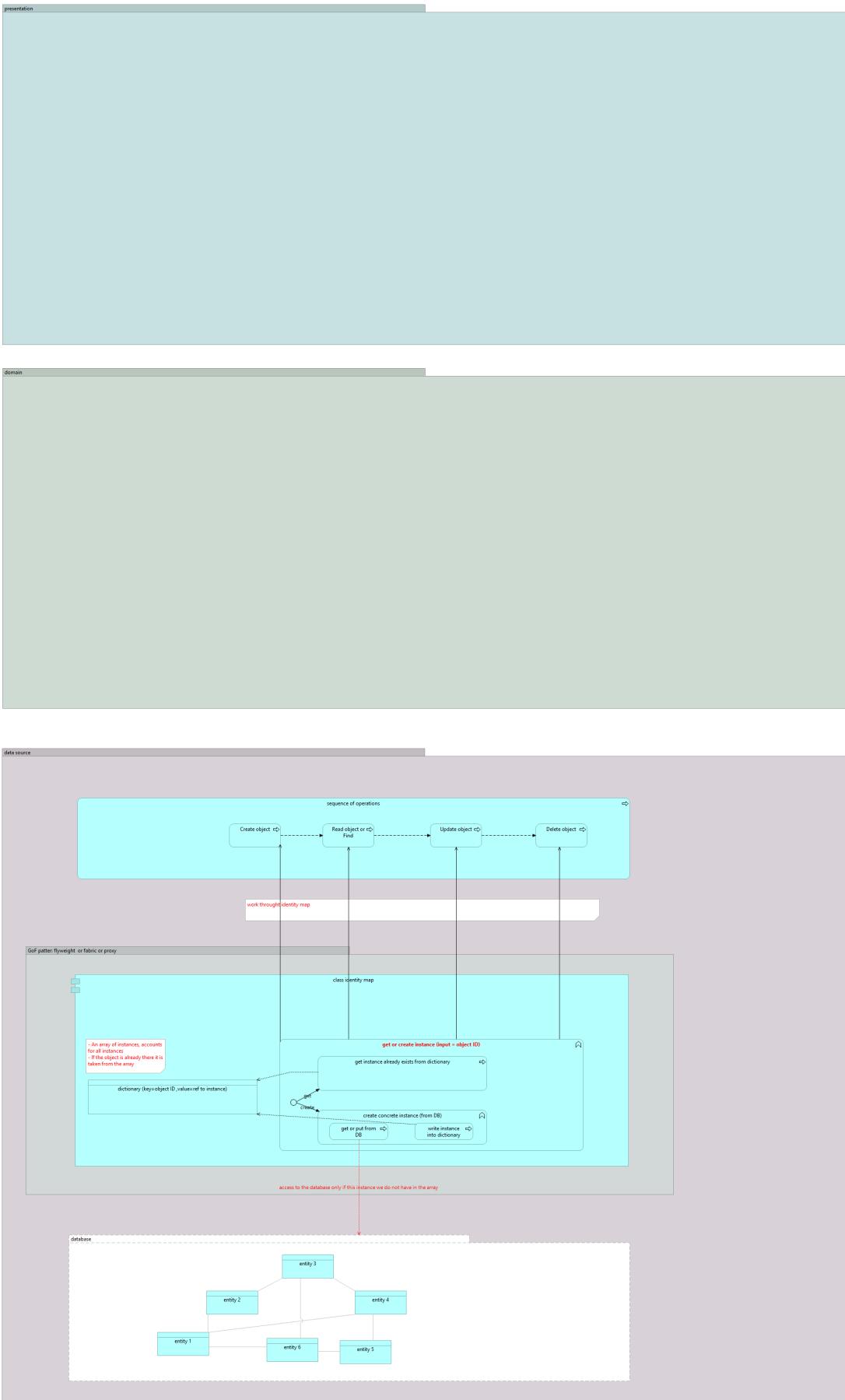
# MODELING BEHAVIOR

ENTERPRISE PATTERNS

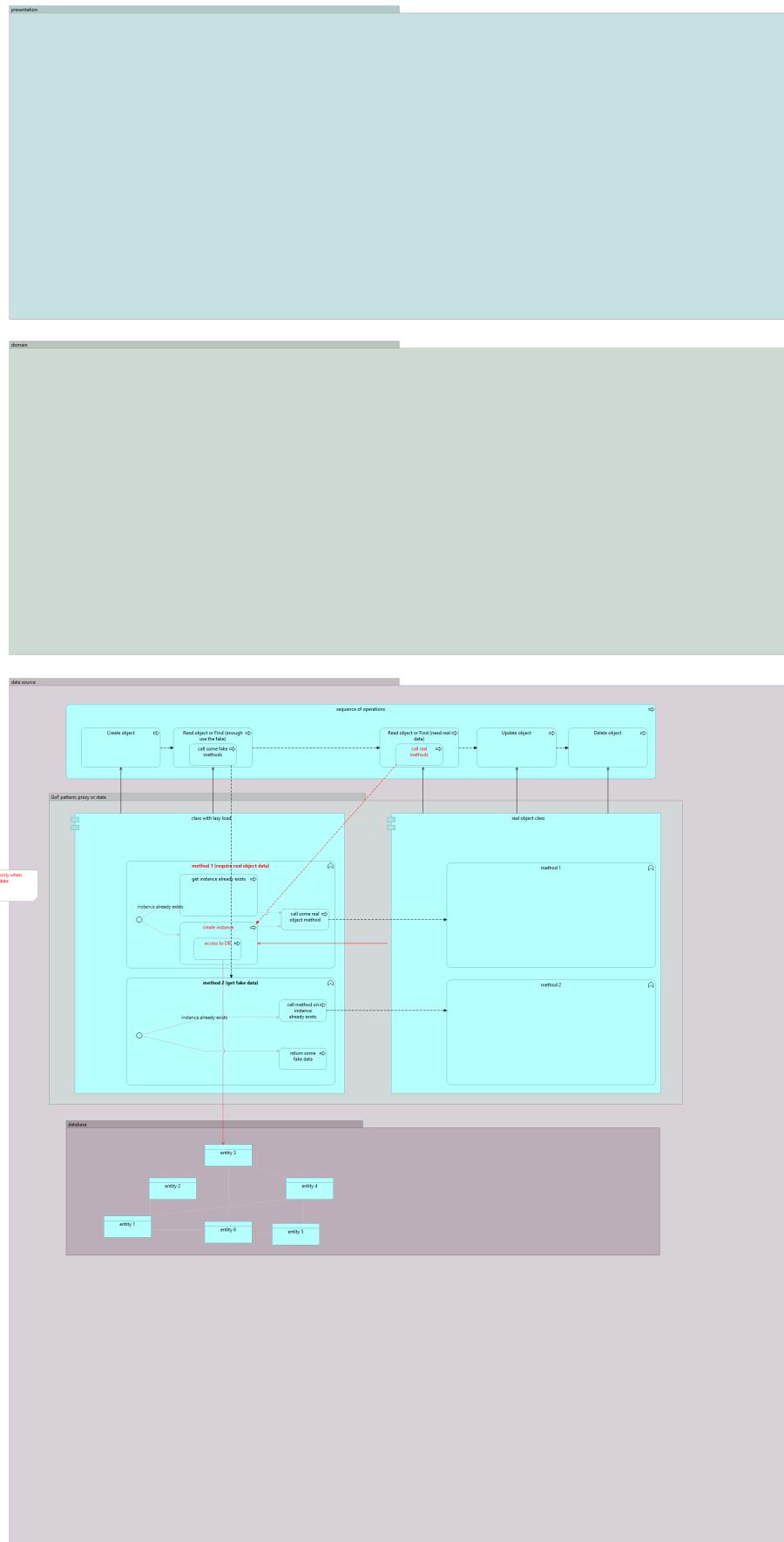
Martin Fowler



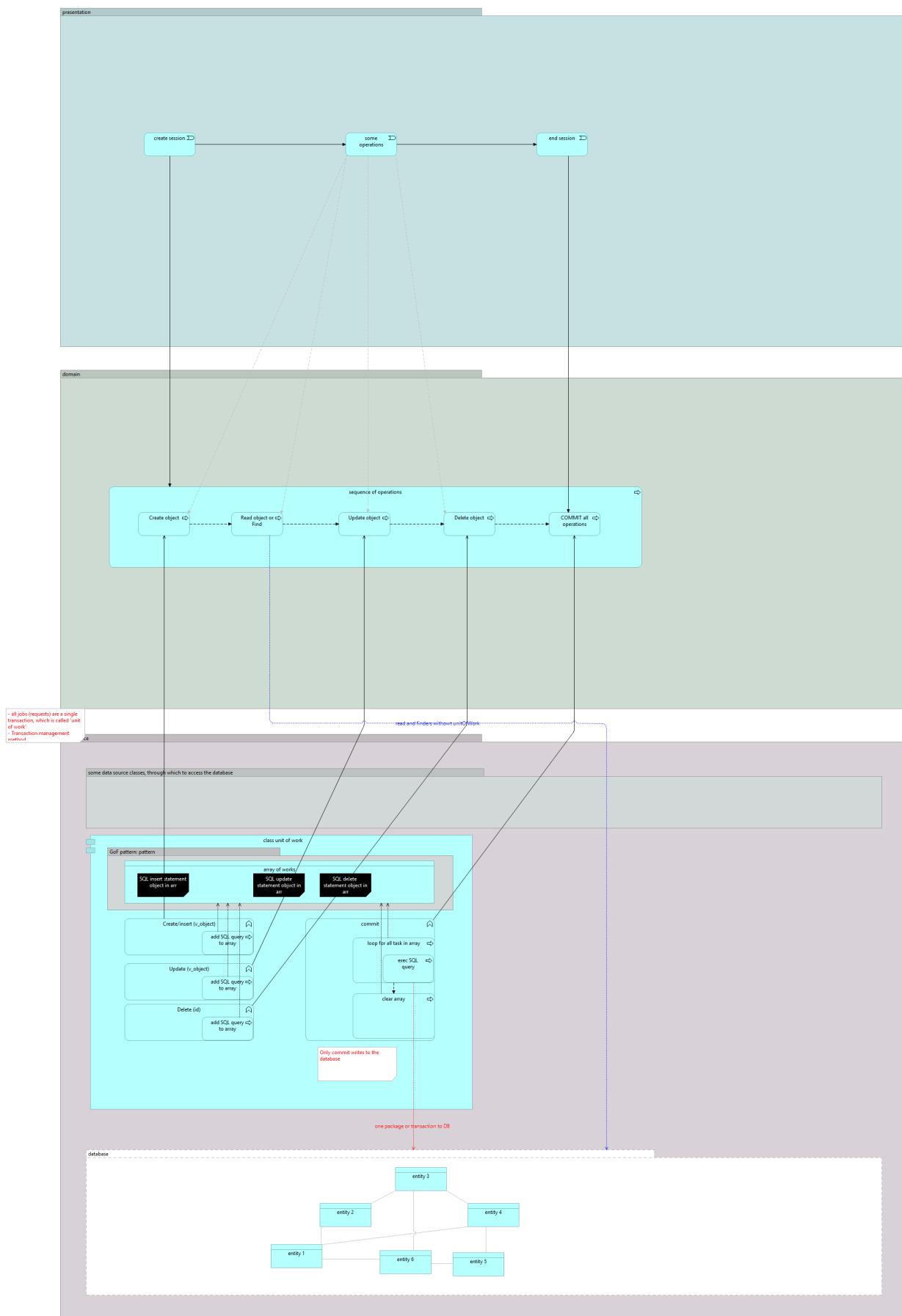
# IDENTITY MAP



# LAZY LOAD



# UNIT OF WORK.



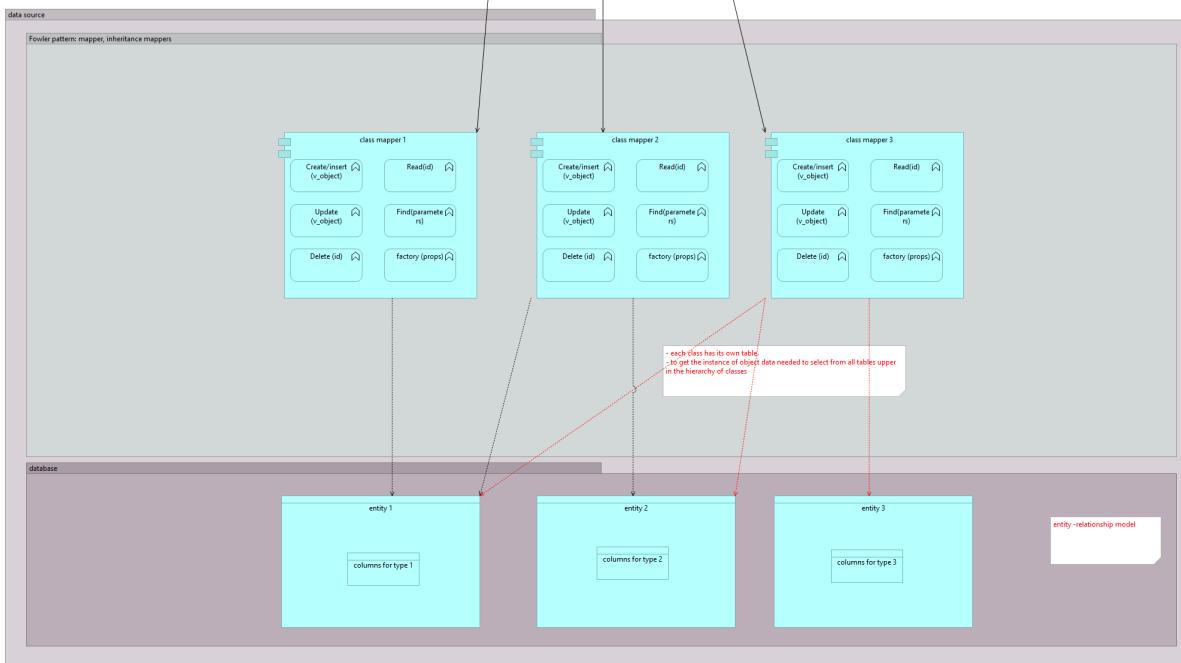
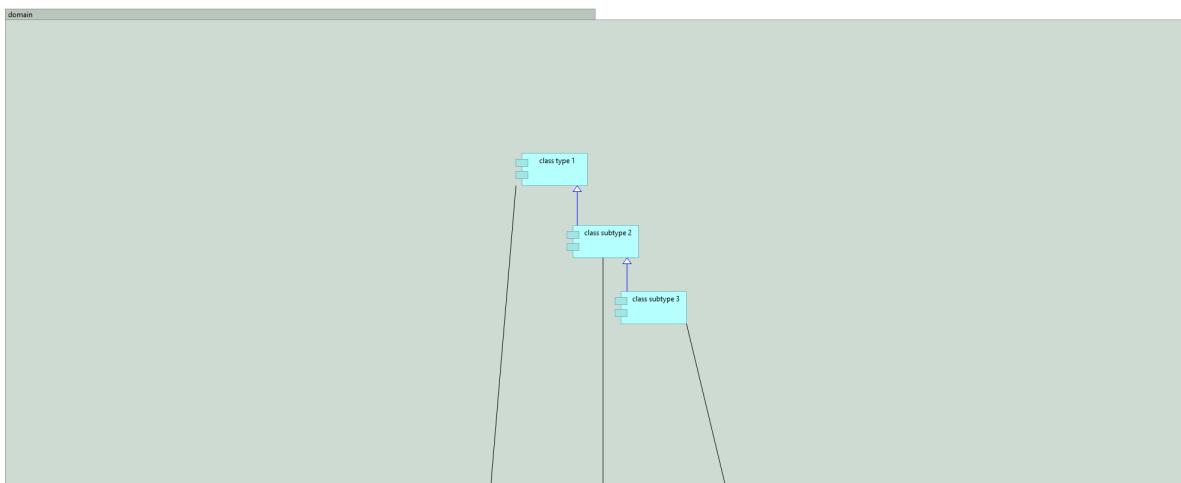
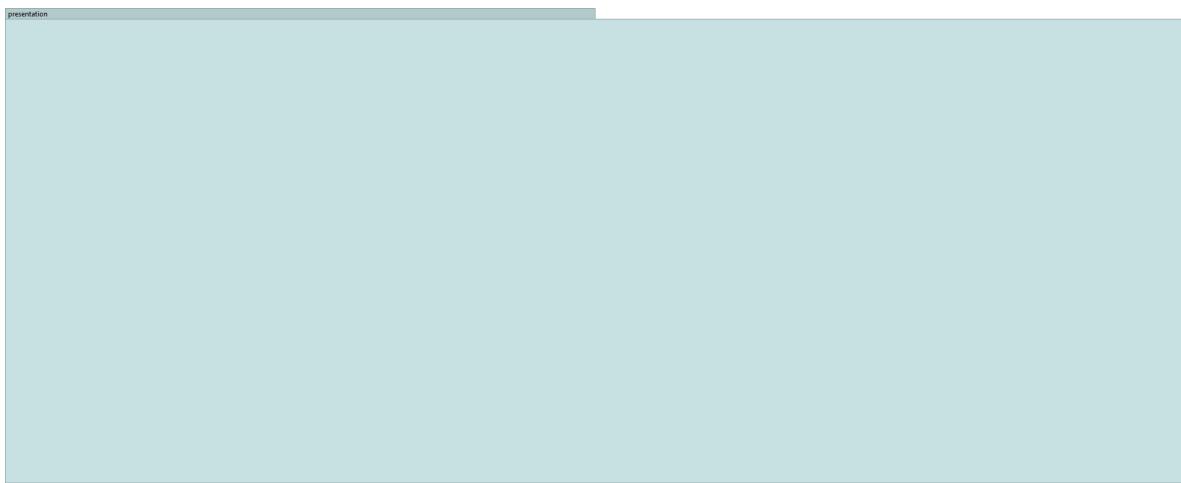
# MODELING STRUCTURE HIERARCHY

ENTERPRISE PATTERNS

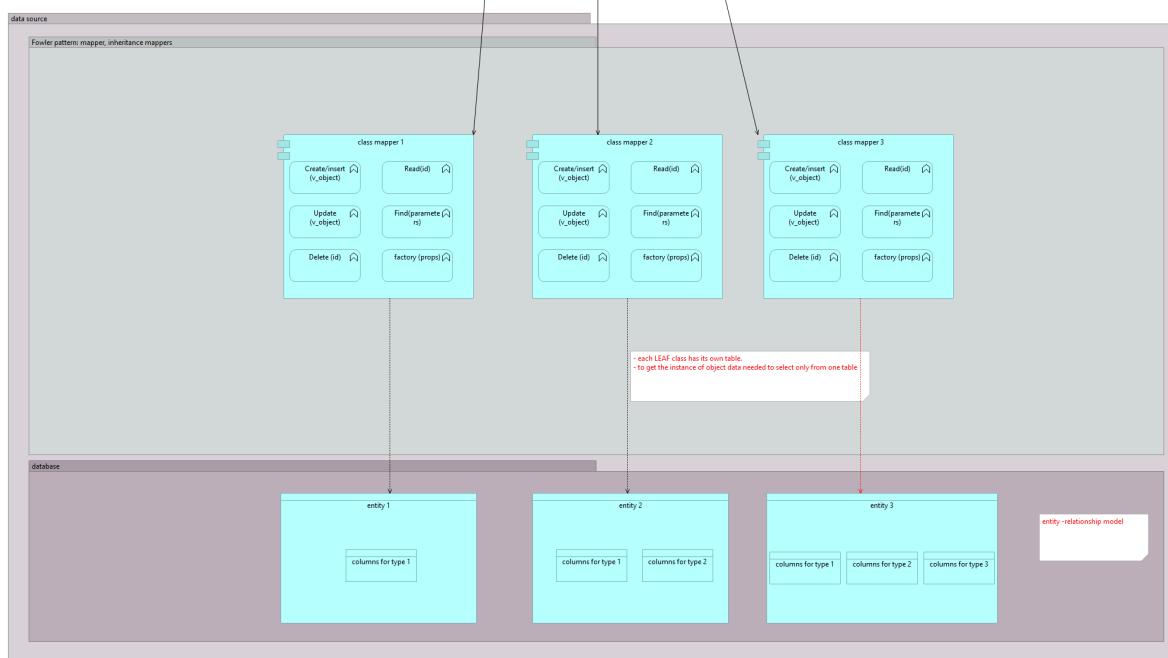
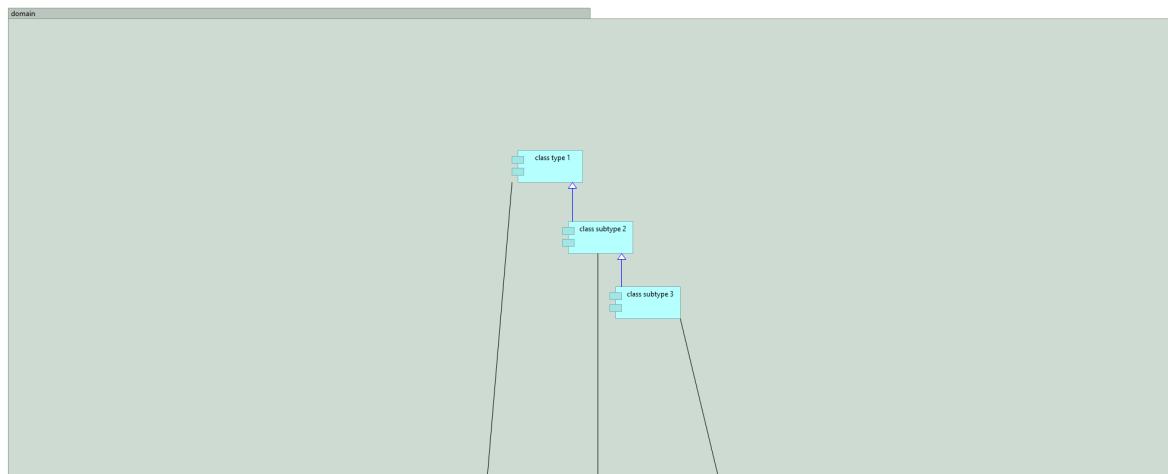
Martin Fowler



# CLASS TABLE INHERITANCE



# CONCRETE TABLE INHERITANCE

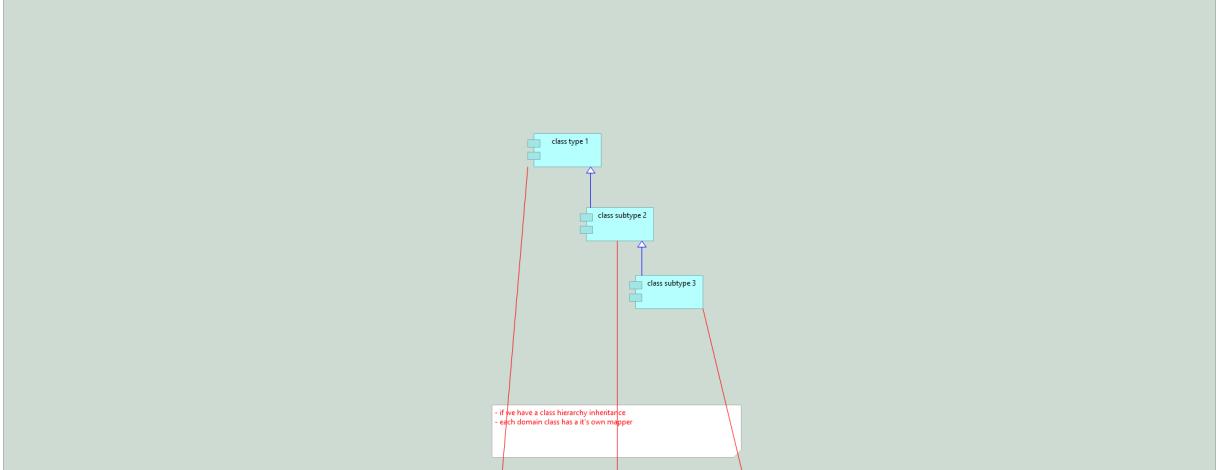


# INHERITANCE MAPPERS

presentation

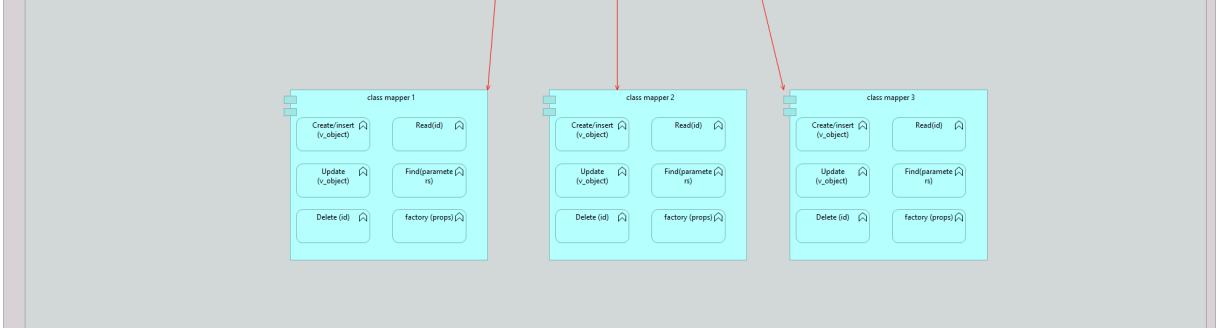


domain



data source

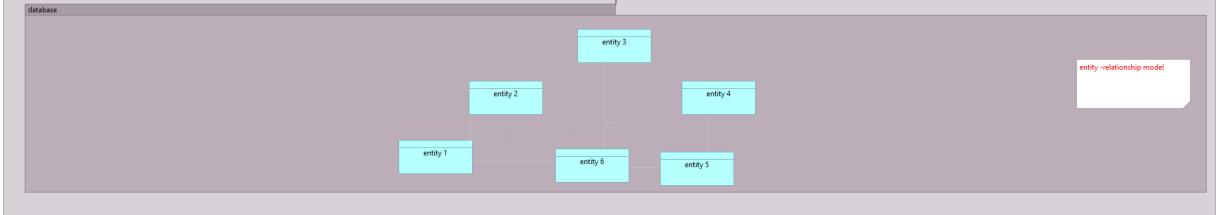
Fowler pattern: mapper



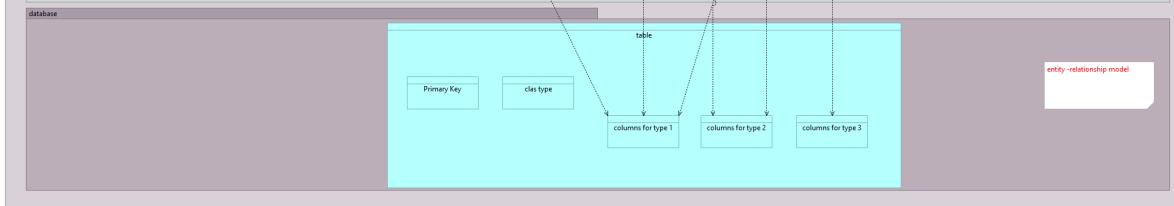
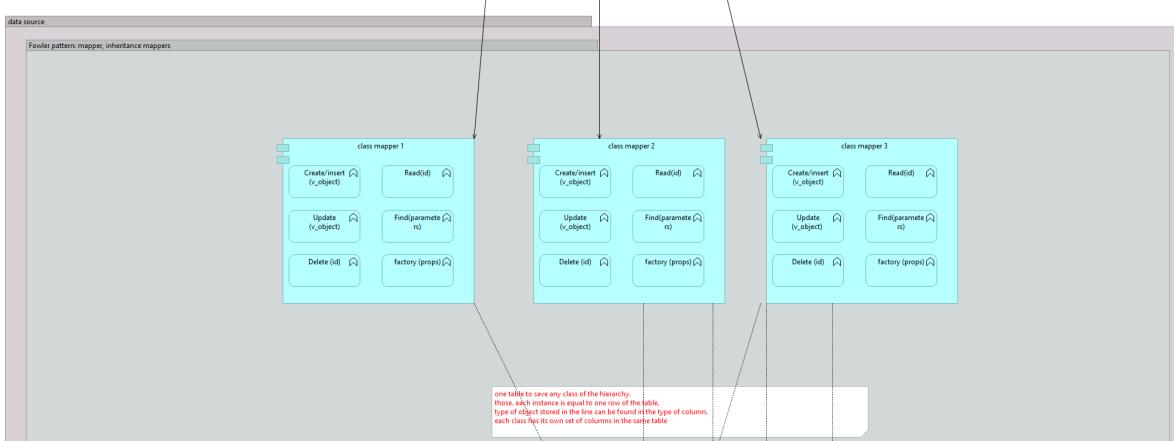
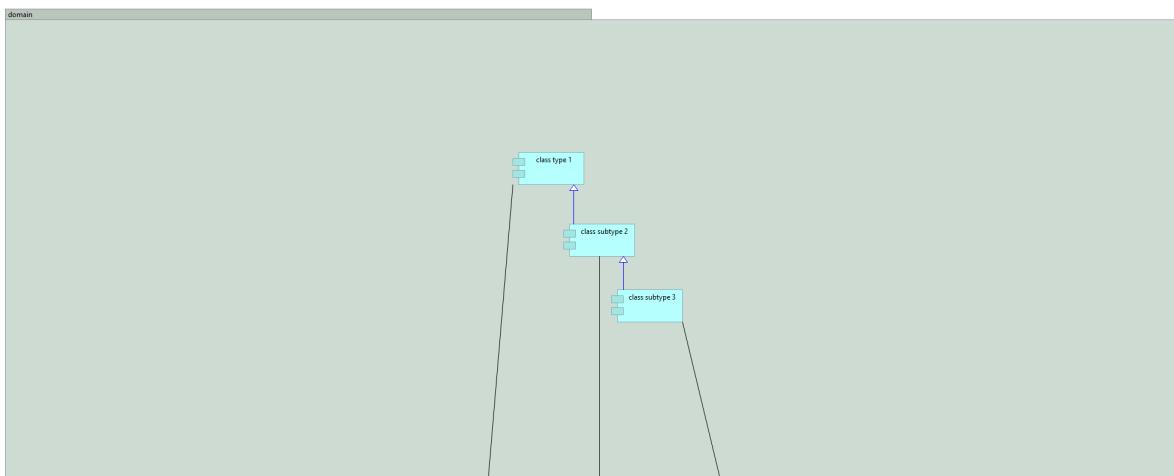
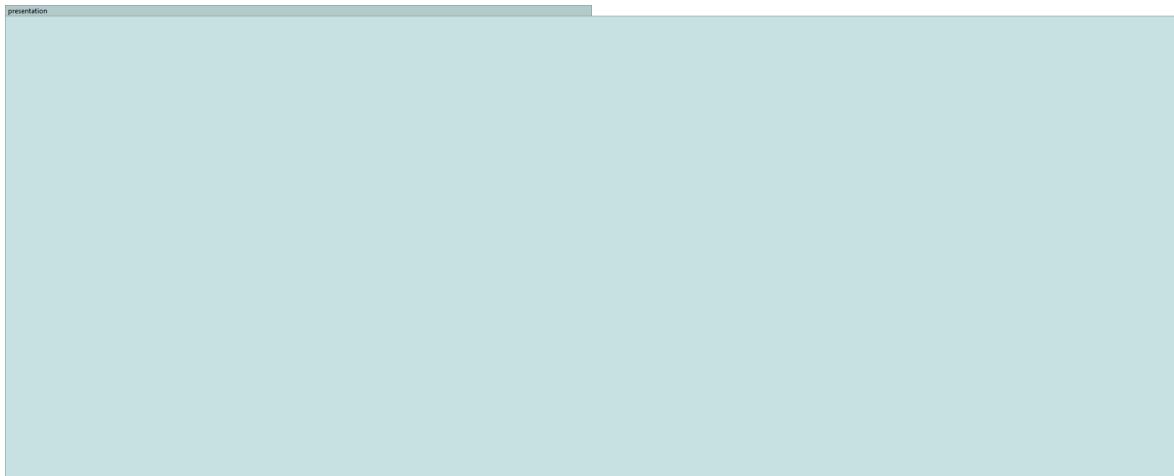
some relations

database

entity -relationship model



# SINGLE TABLE INHERITANCE



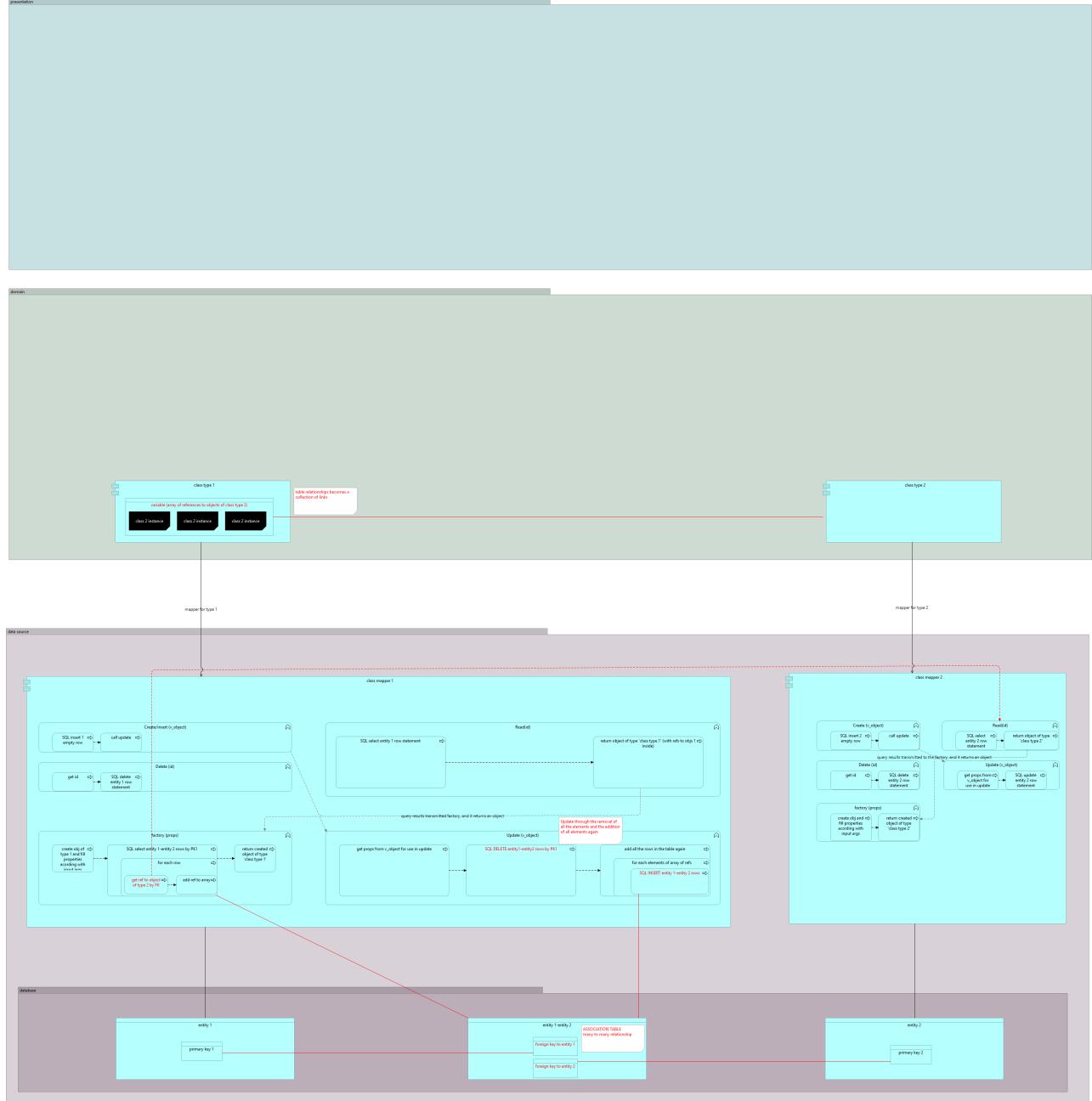
# MODELING STRUCTURE RELATIONS

ENTERPRISE PATTERNS

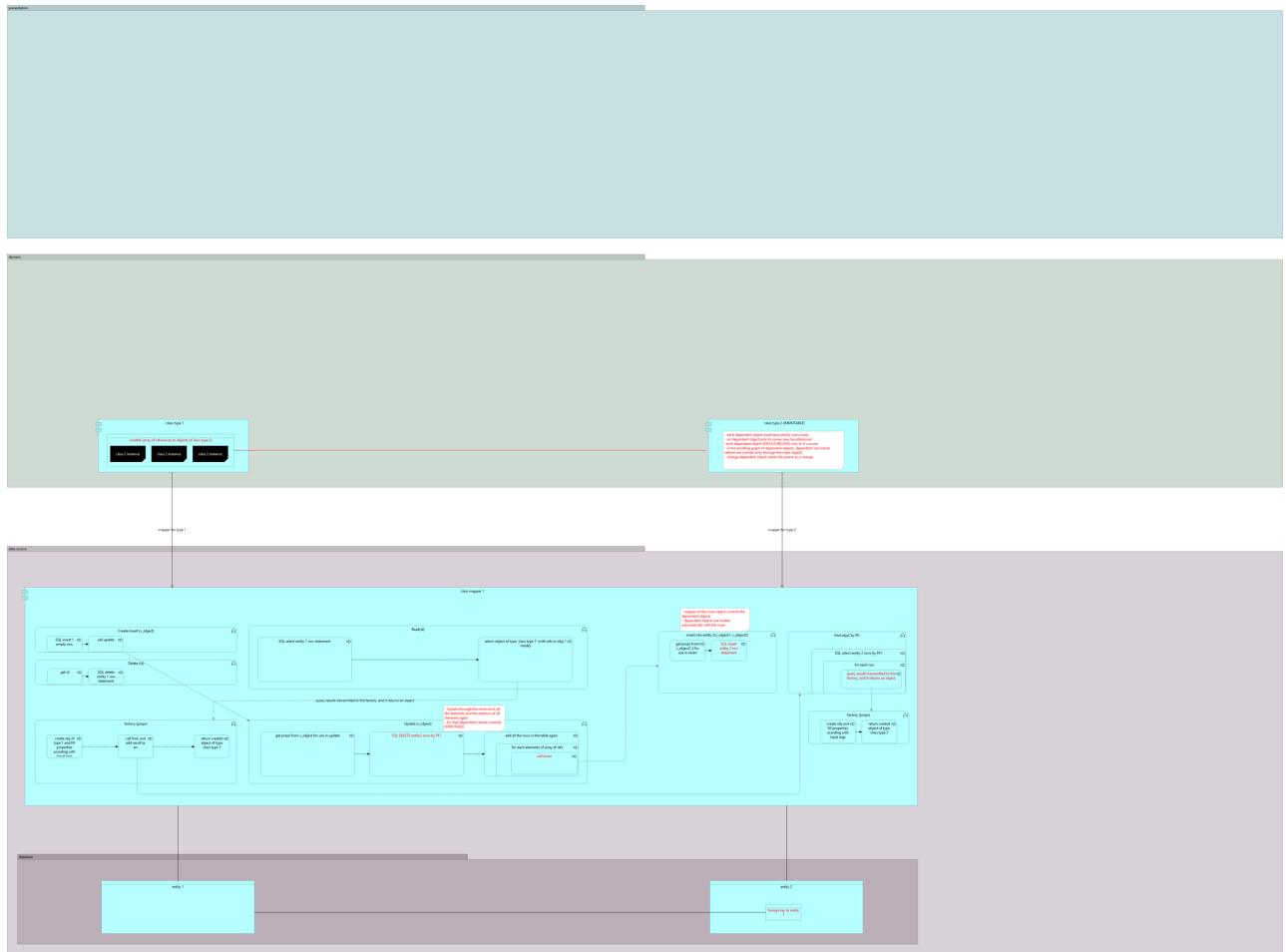
Martin Fowler



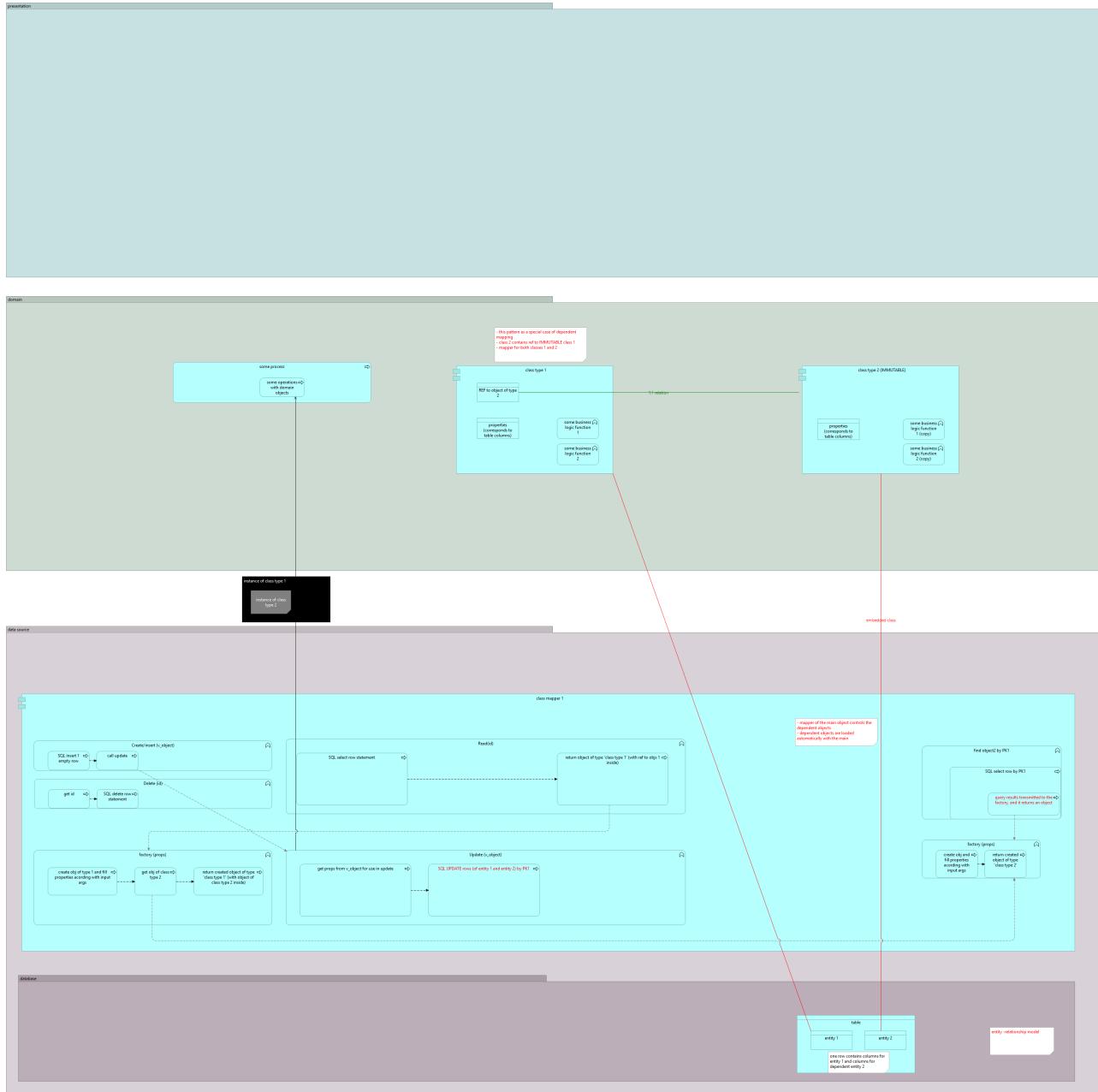
# ASSOCIATION TABLE MAPPING



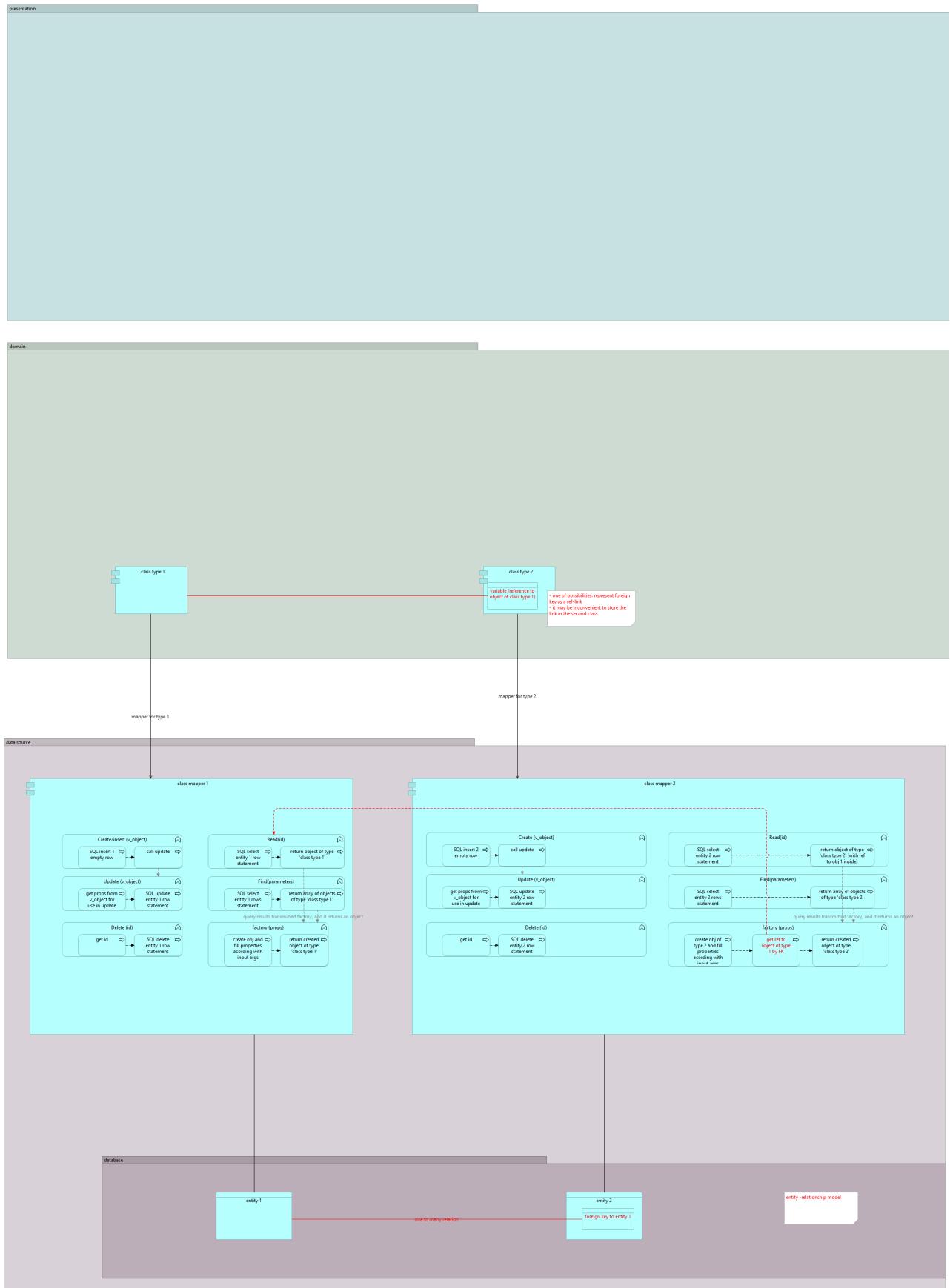
# DEPENDENT MAPPING



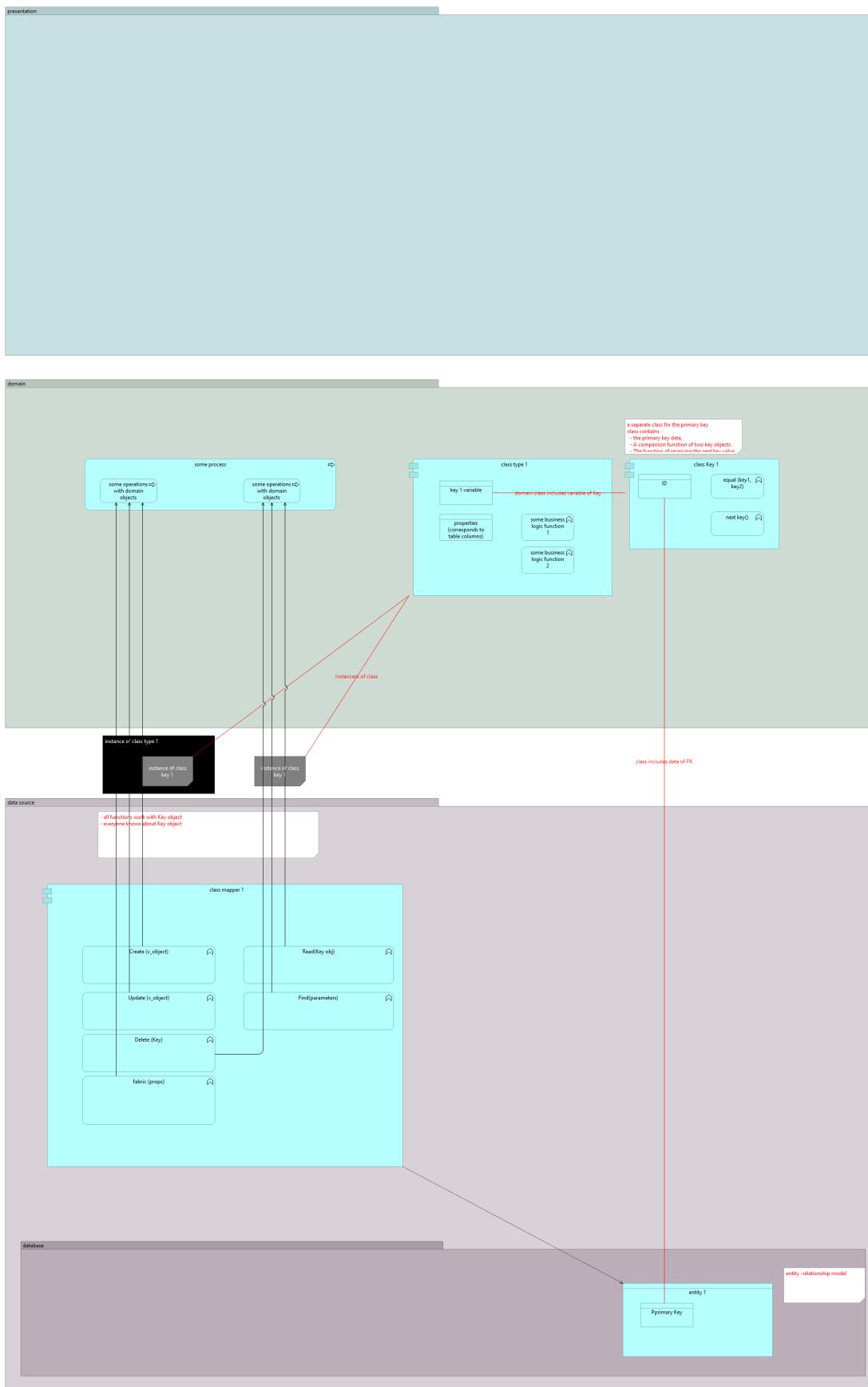
# EMBEDDED VALUE



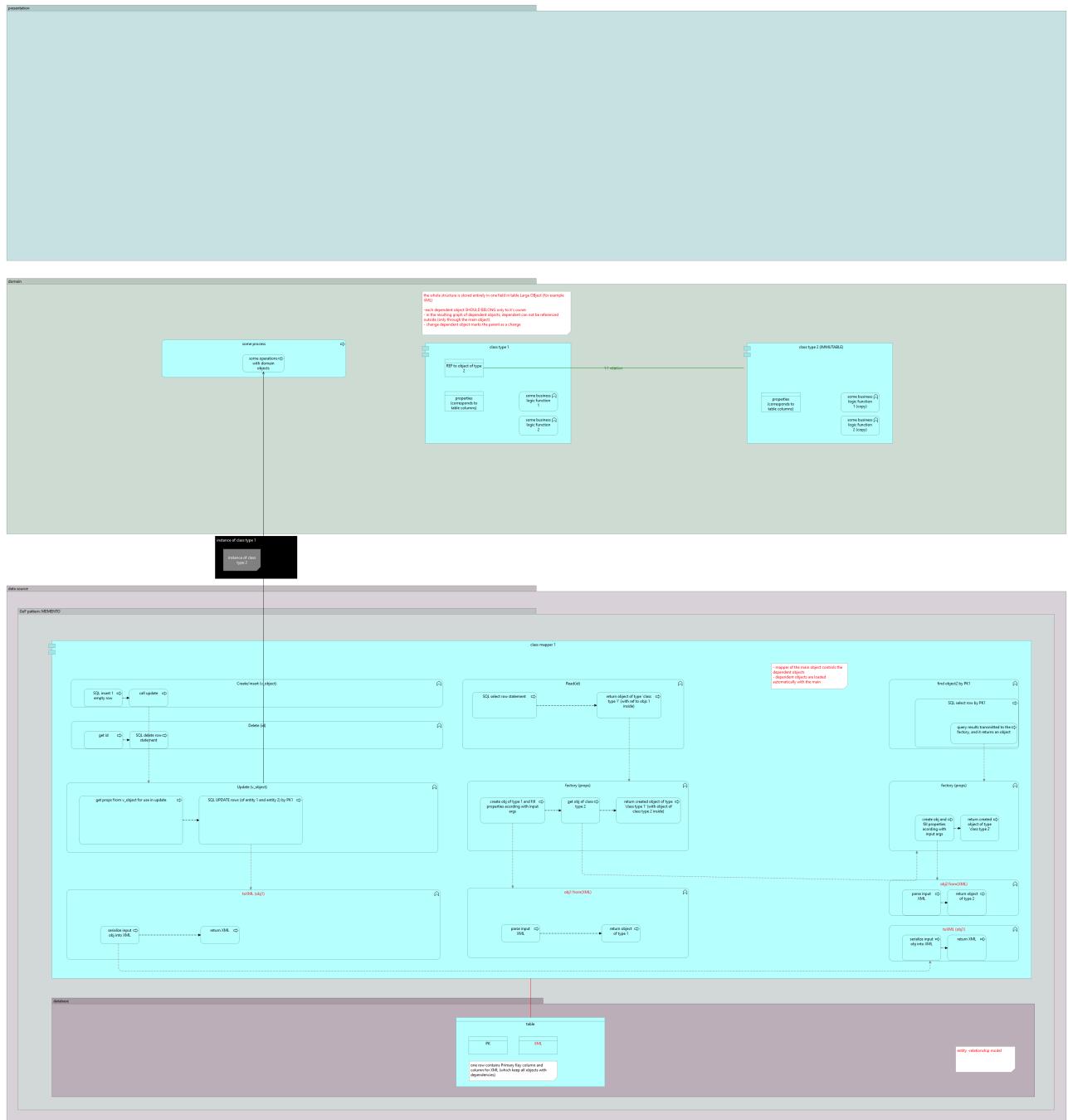
# FOREIGN KEY MAPPING



# IDENTITY FIELD



# SERIALIZED LOB



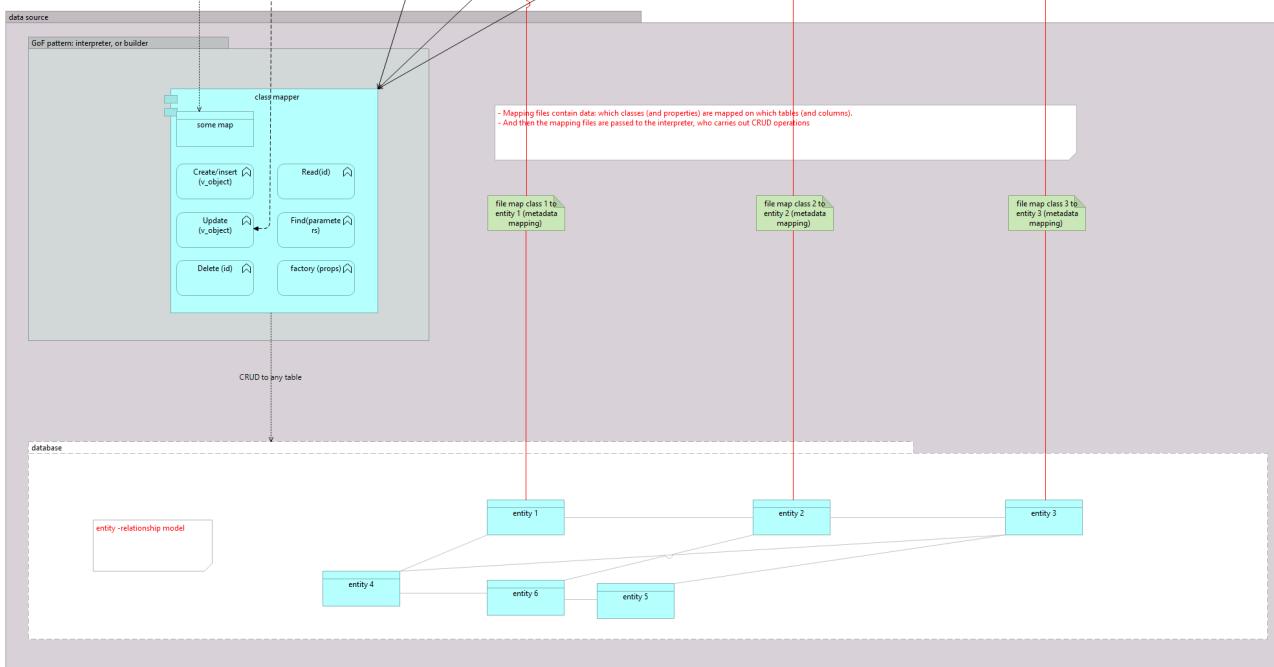
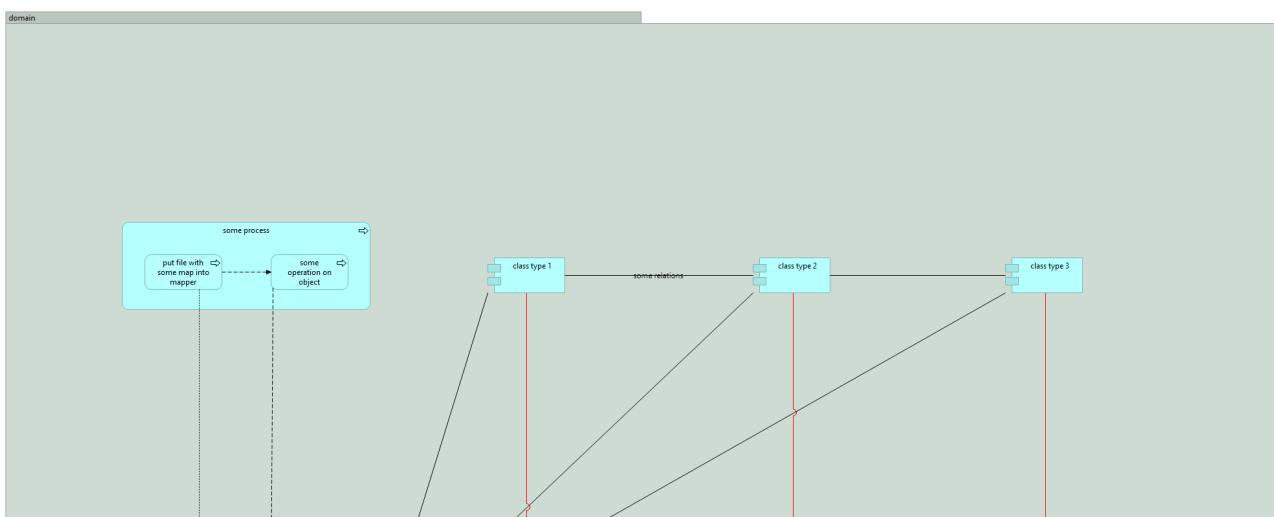
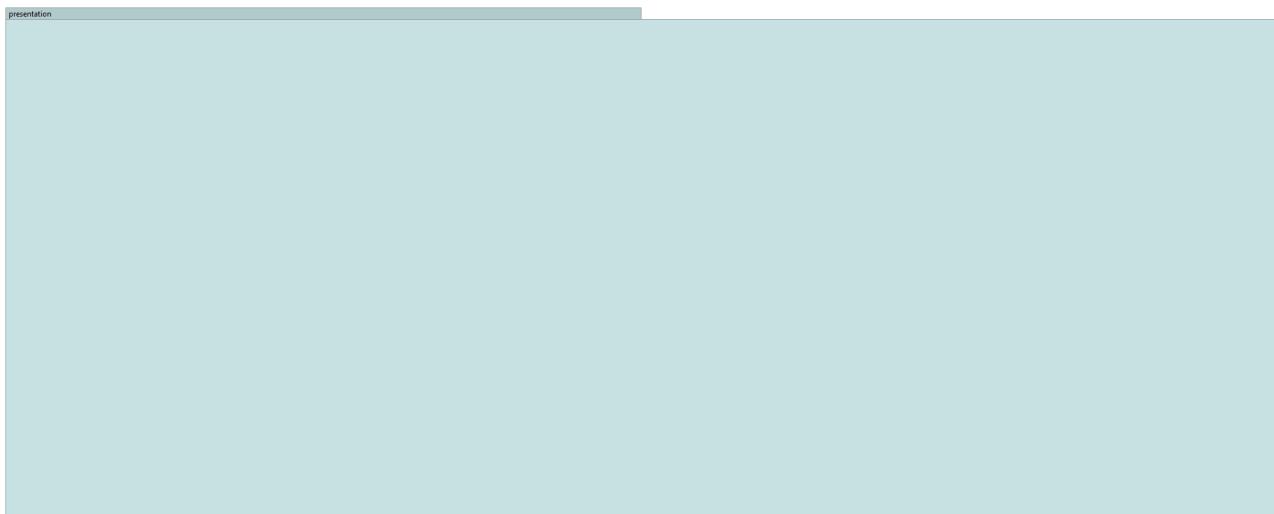
# METADATA

ENTERPRISE PATTERNS

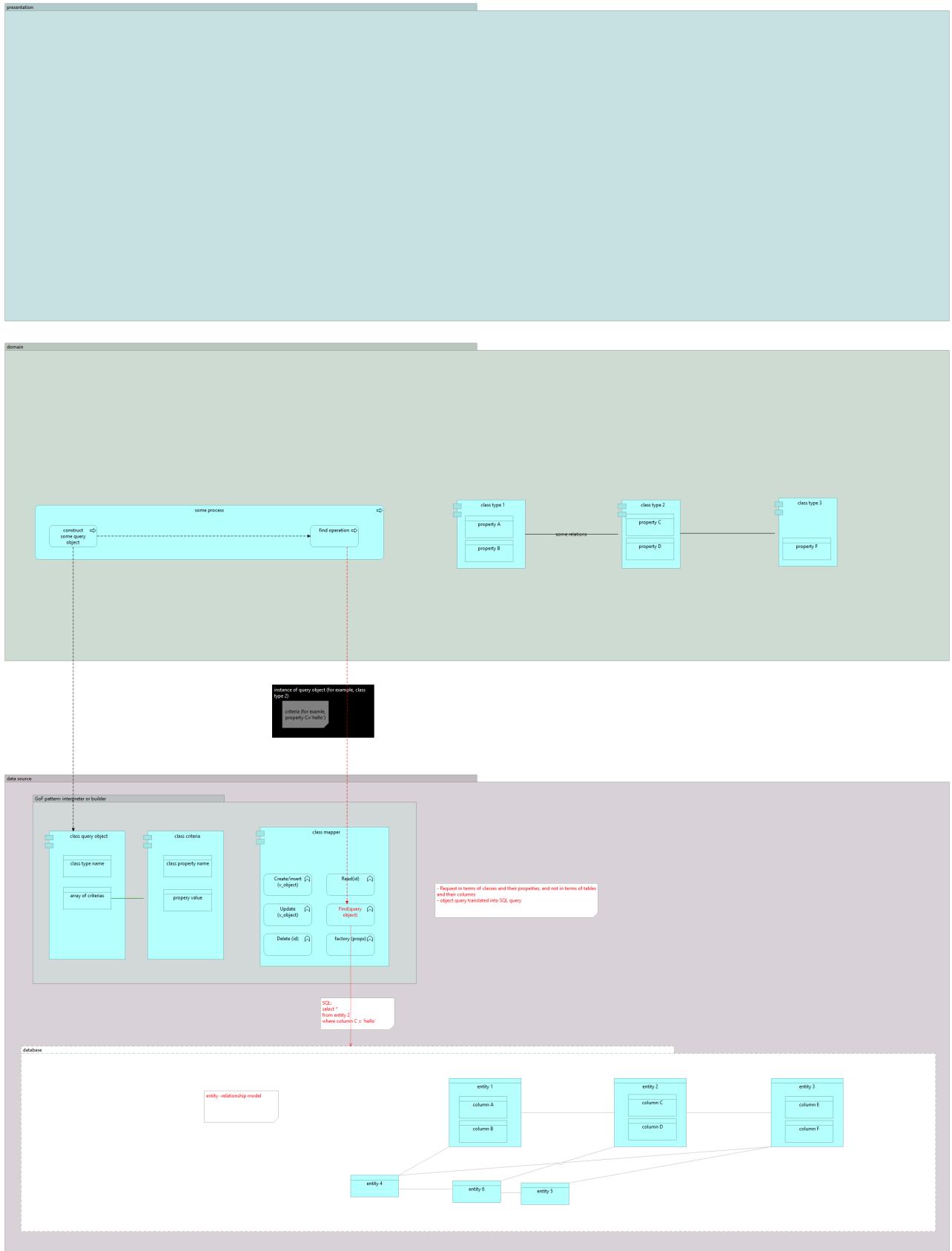
Martin Fowler



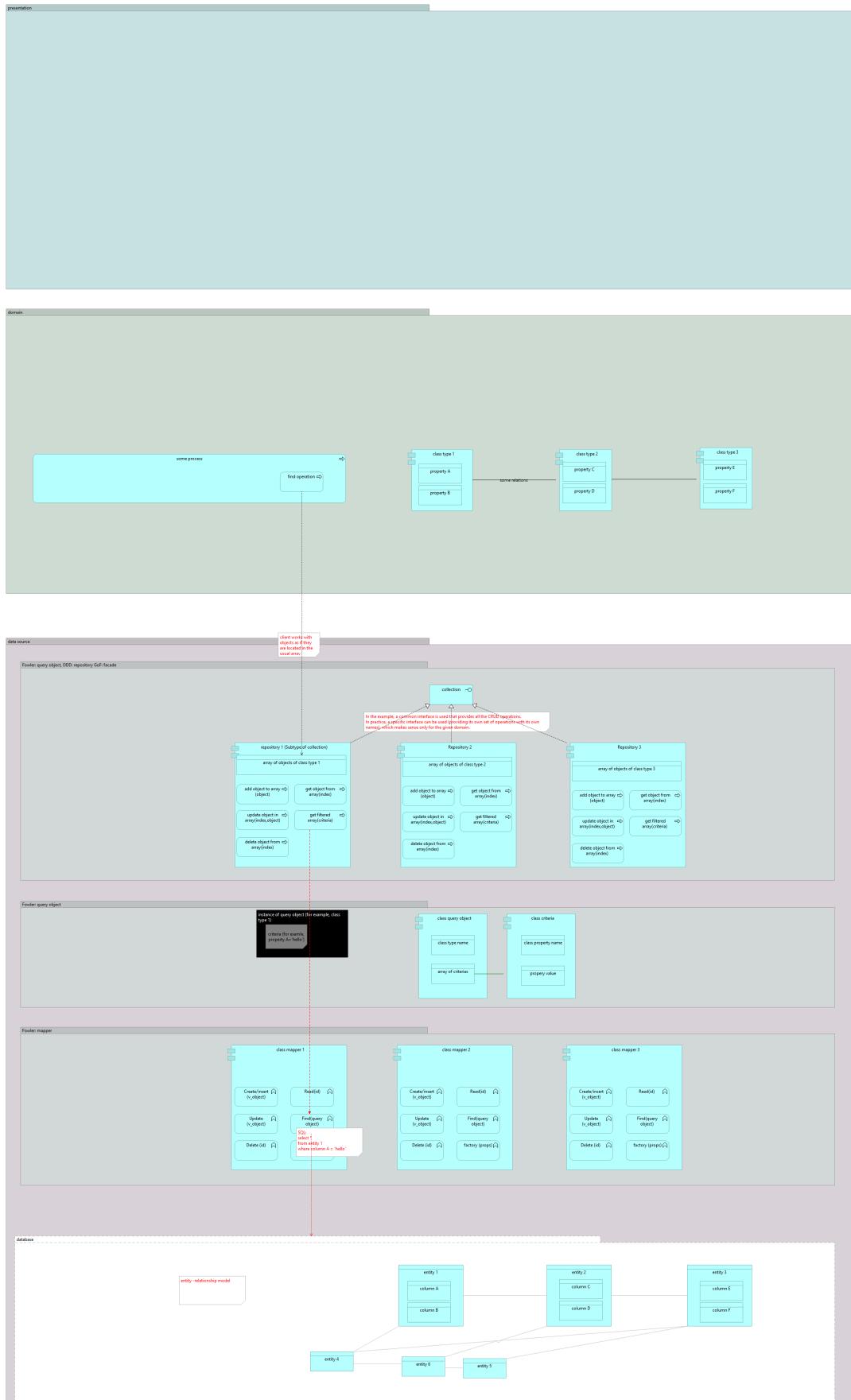
# METADATA MAPPING



# QUERY OBJECT



# REPOSITORY



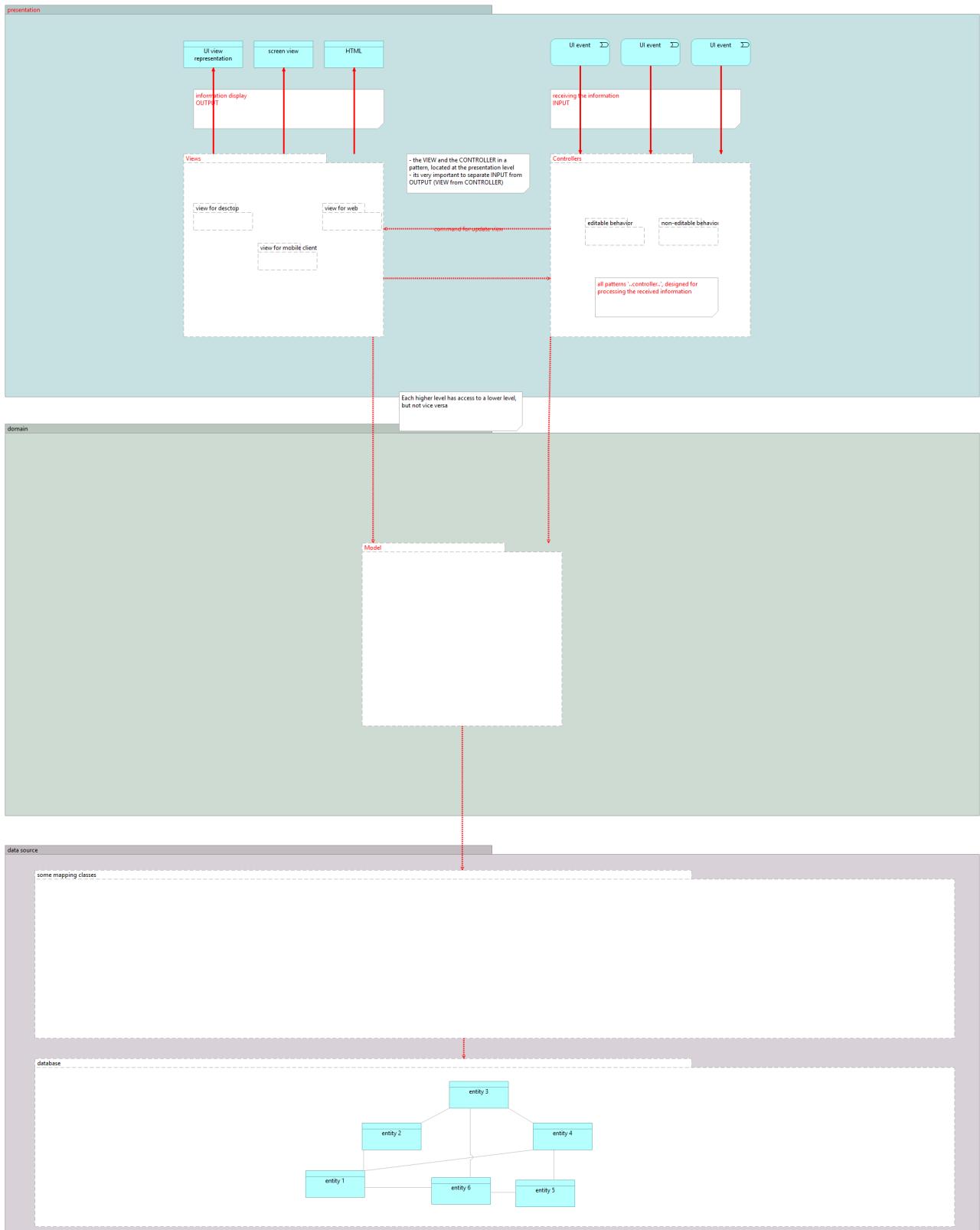
# WEB REPRESENTATION CONTROLLER

ENTERPRISE PATTERNS

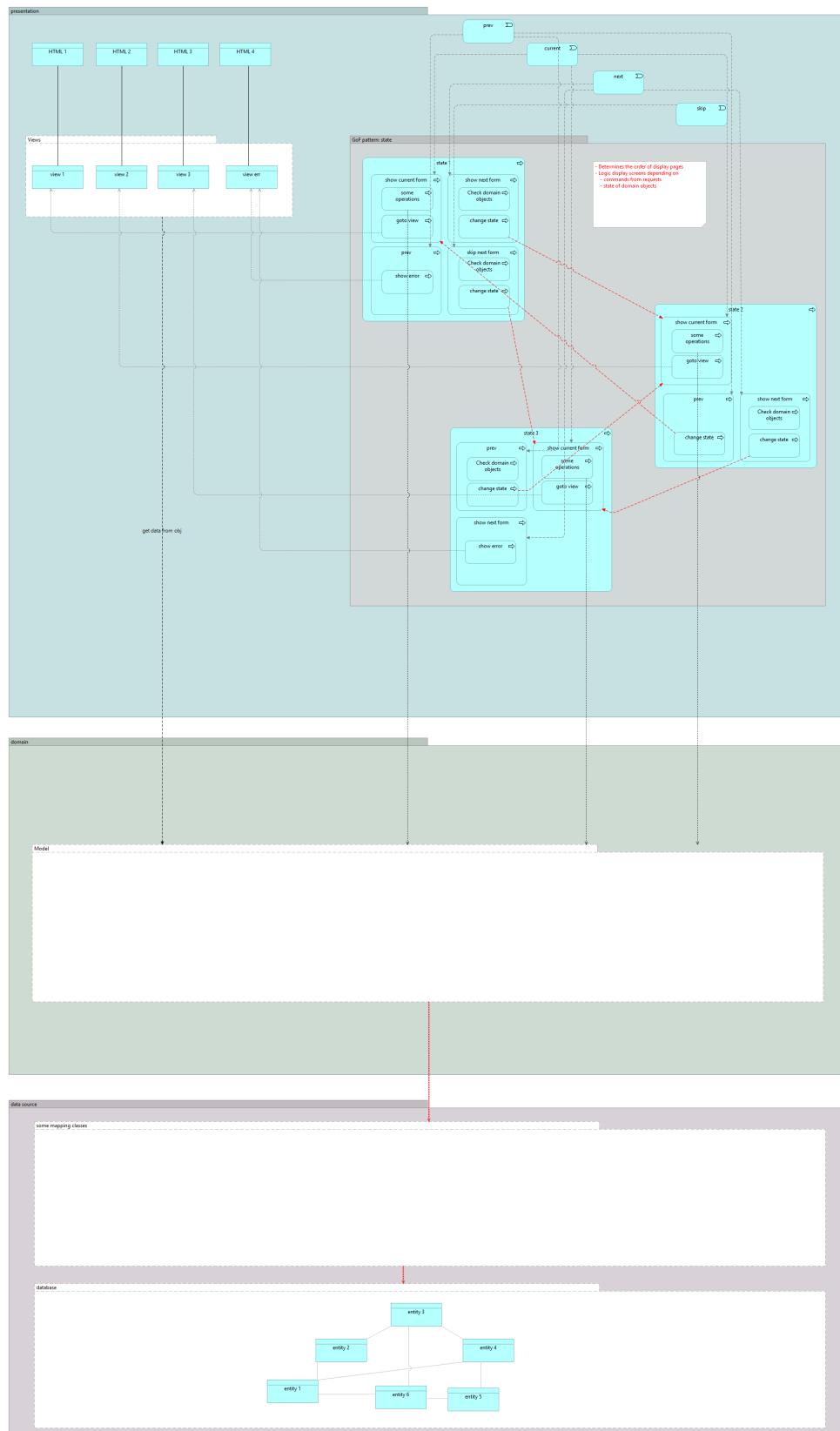
Martin Fowler



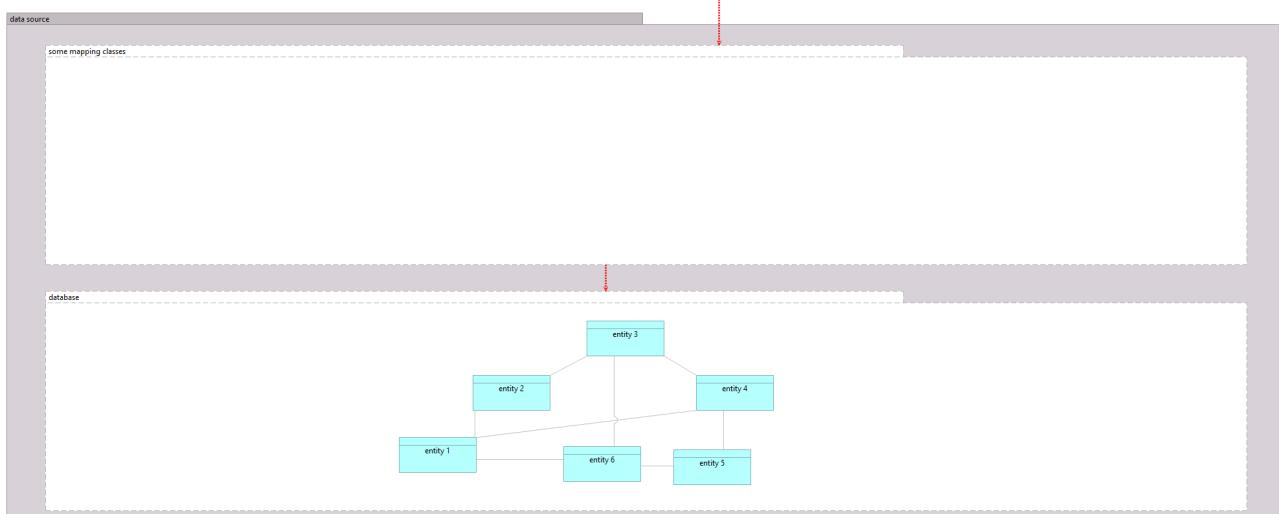
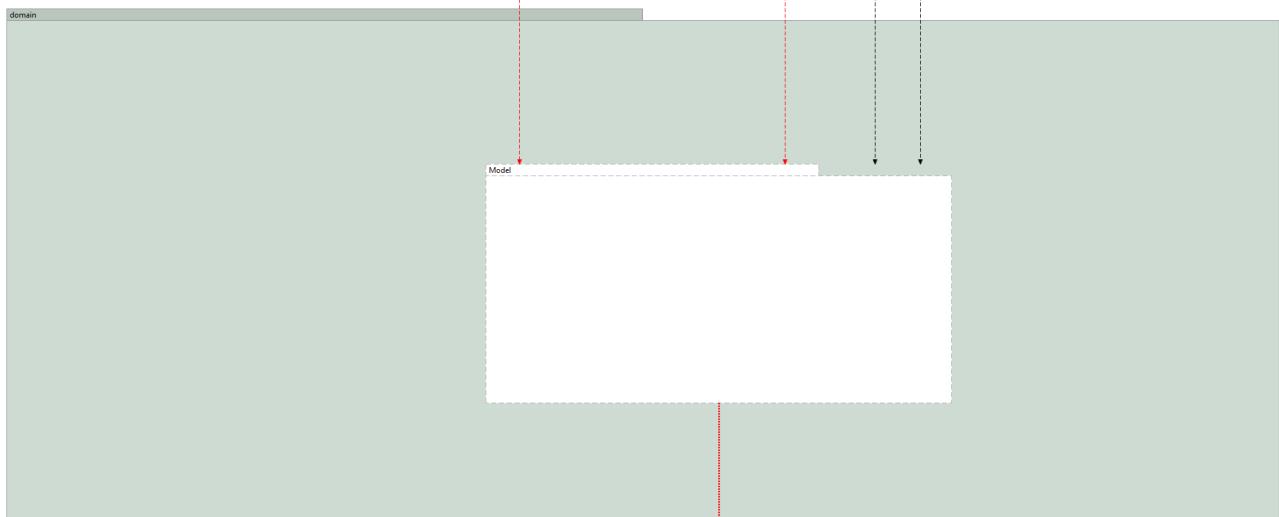
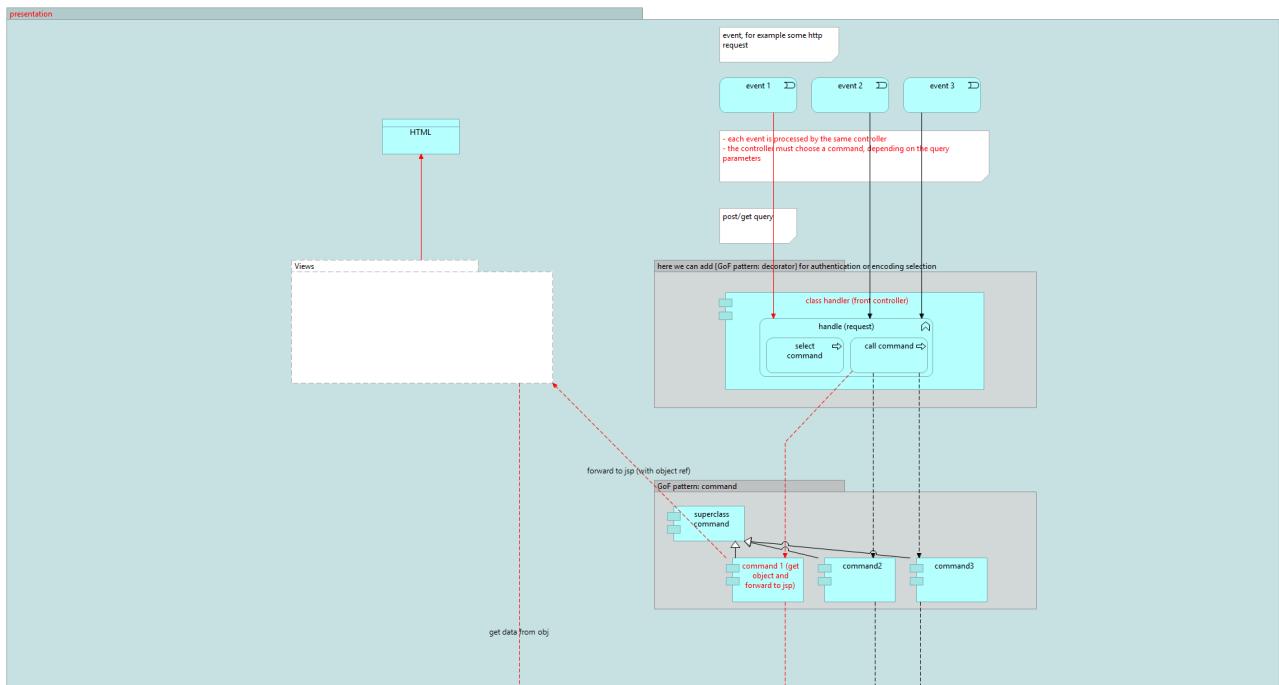
# MODEL VIEW CONTROLLER



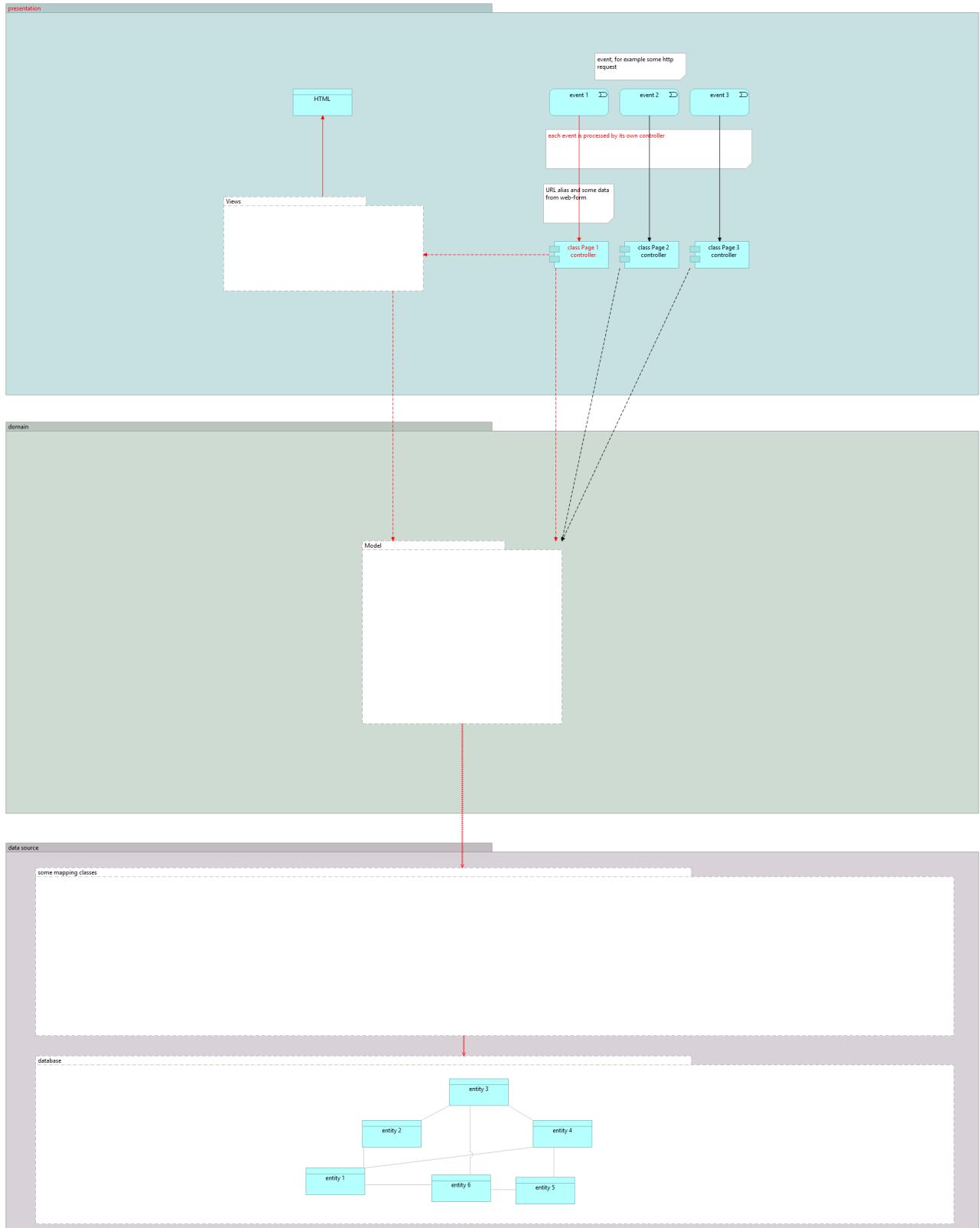
# APPLICATION CONTROLLER



# FRONT CONTROLLER



# PAGE CONTROLLER



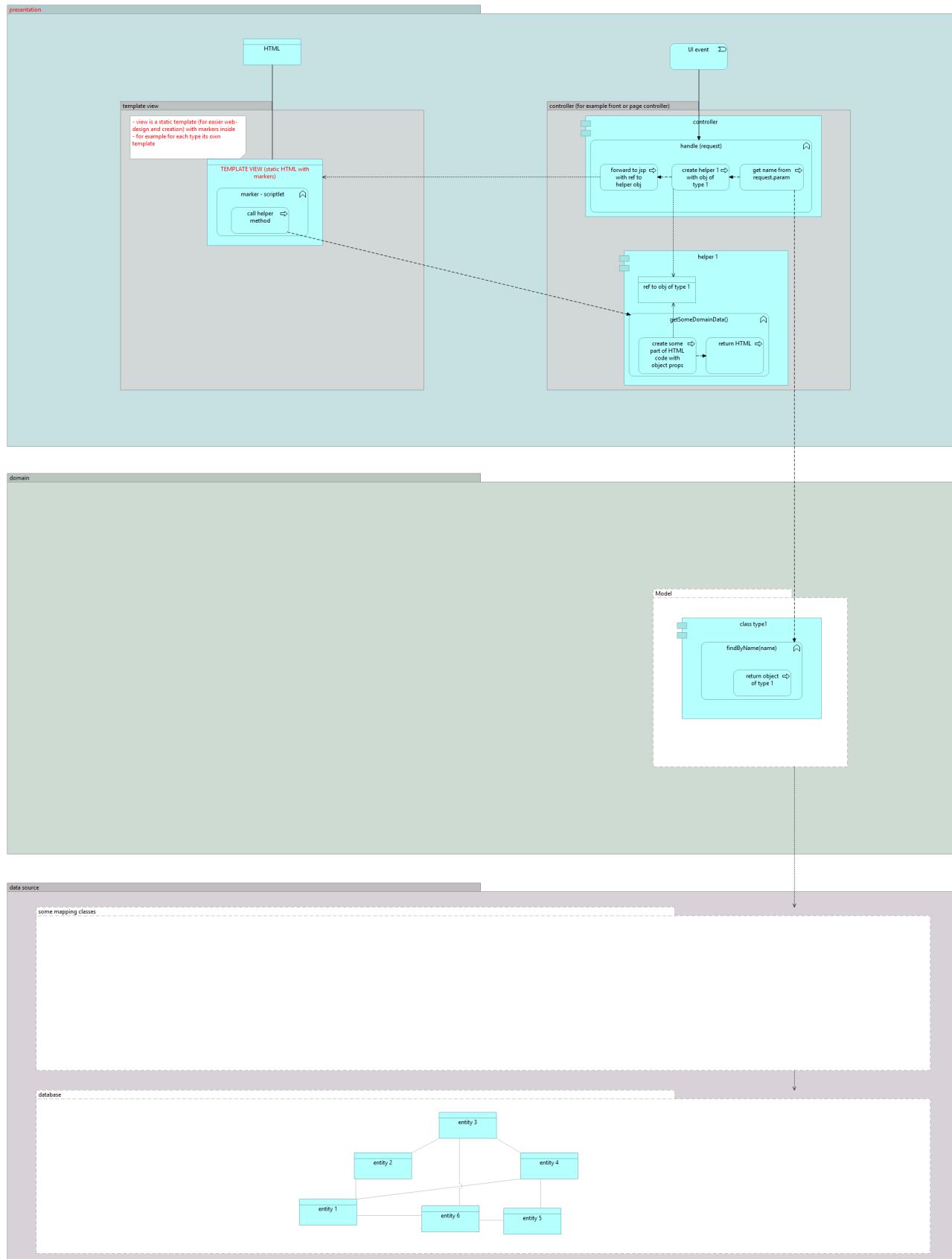
# WEB REPRESENTATION VIEW

ENTERPRISE PATTERNS

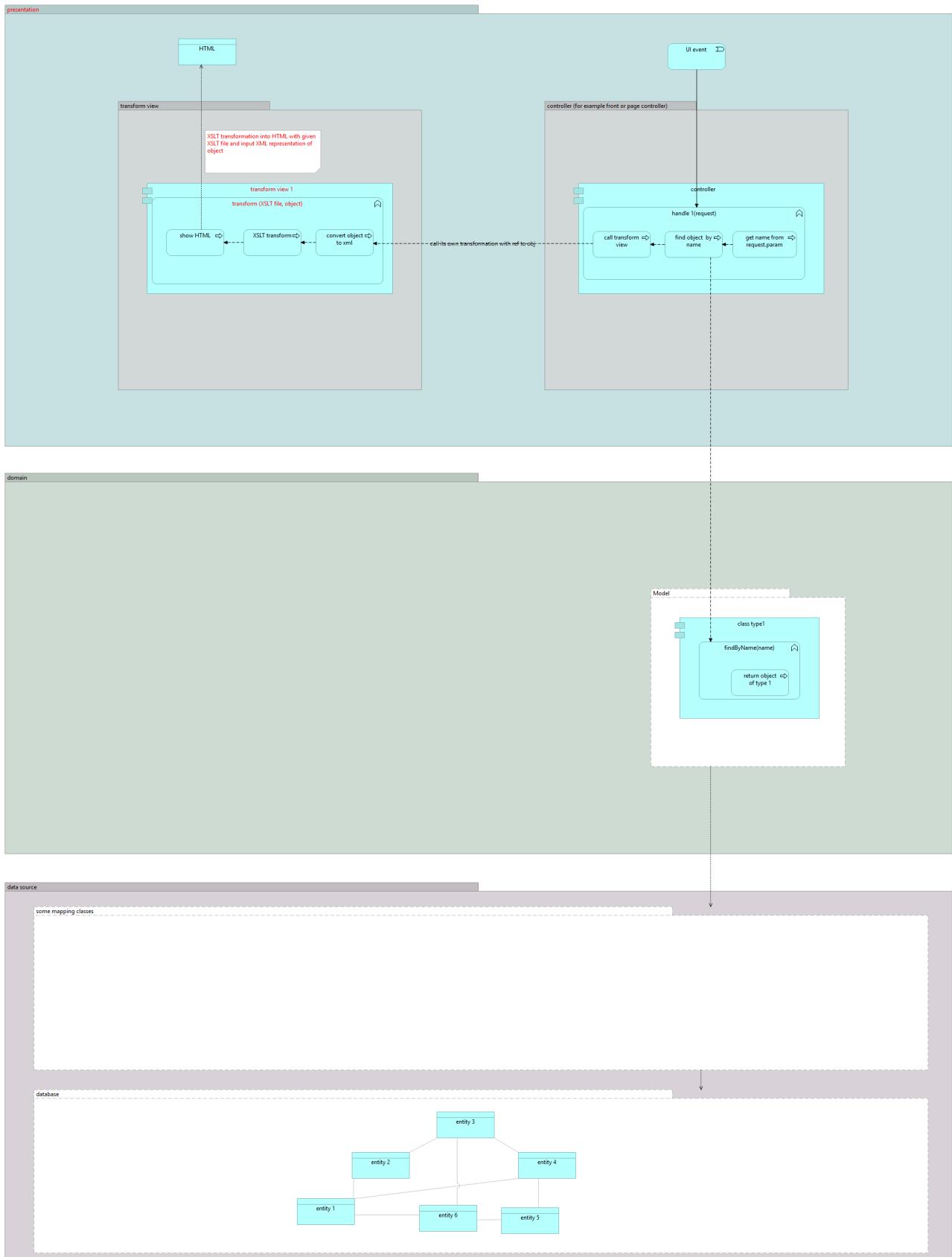
Martin Fowler



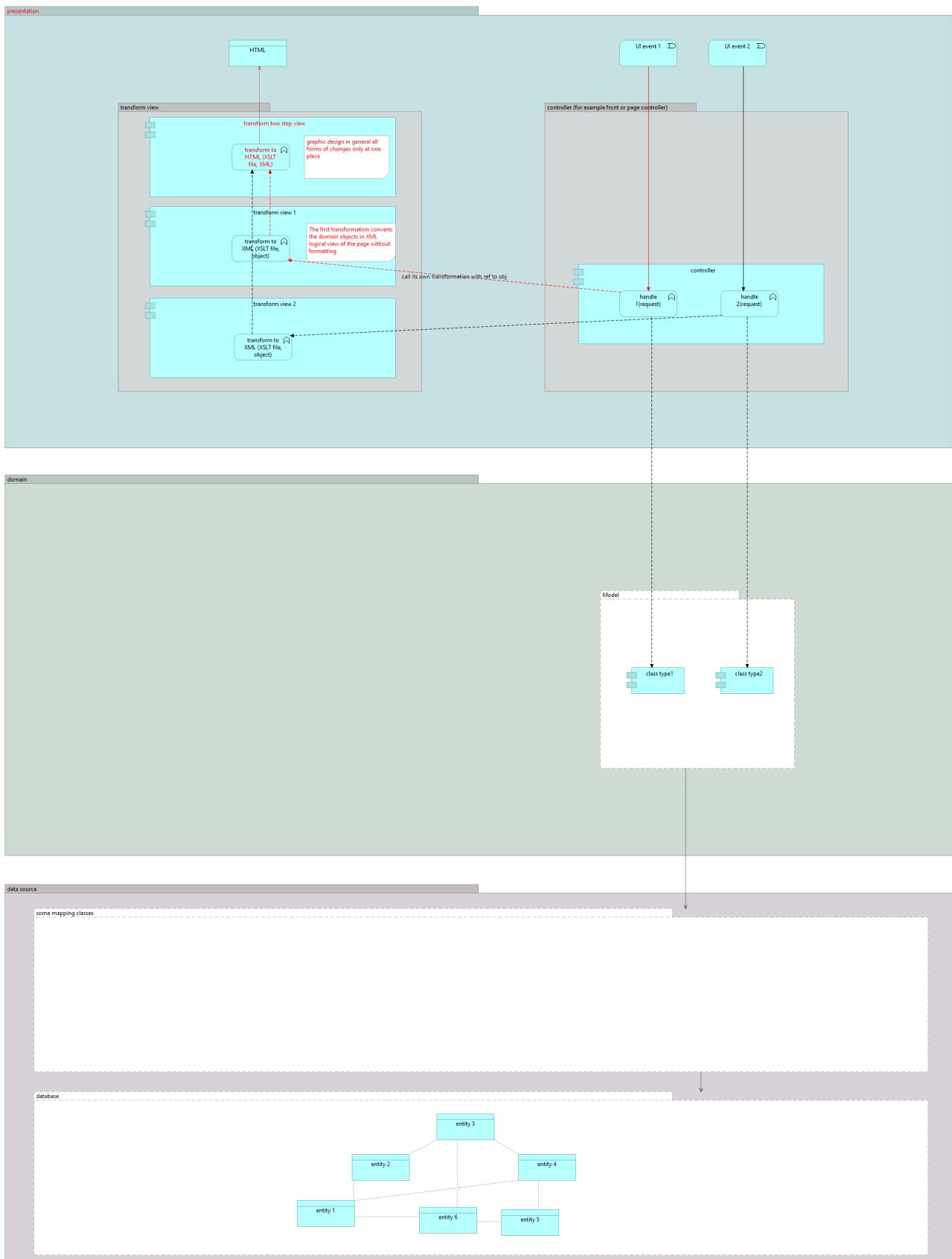
# TEMPLATE VIEW



# TRANSFORM VIEW



# TWO STEP VIEW



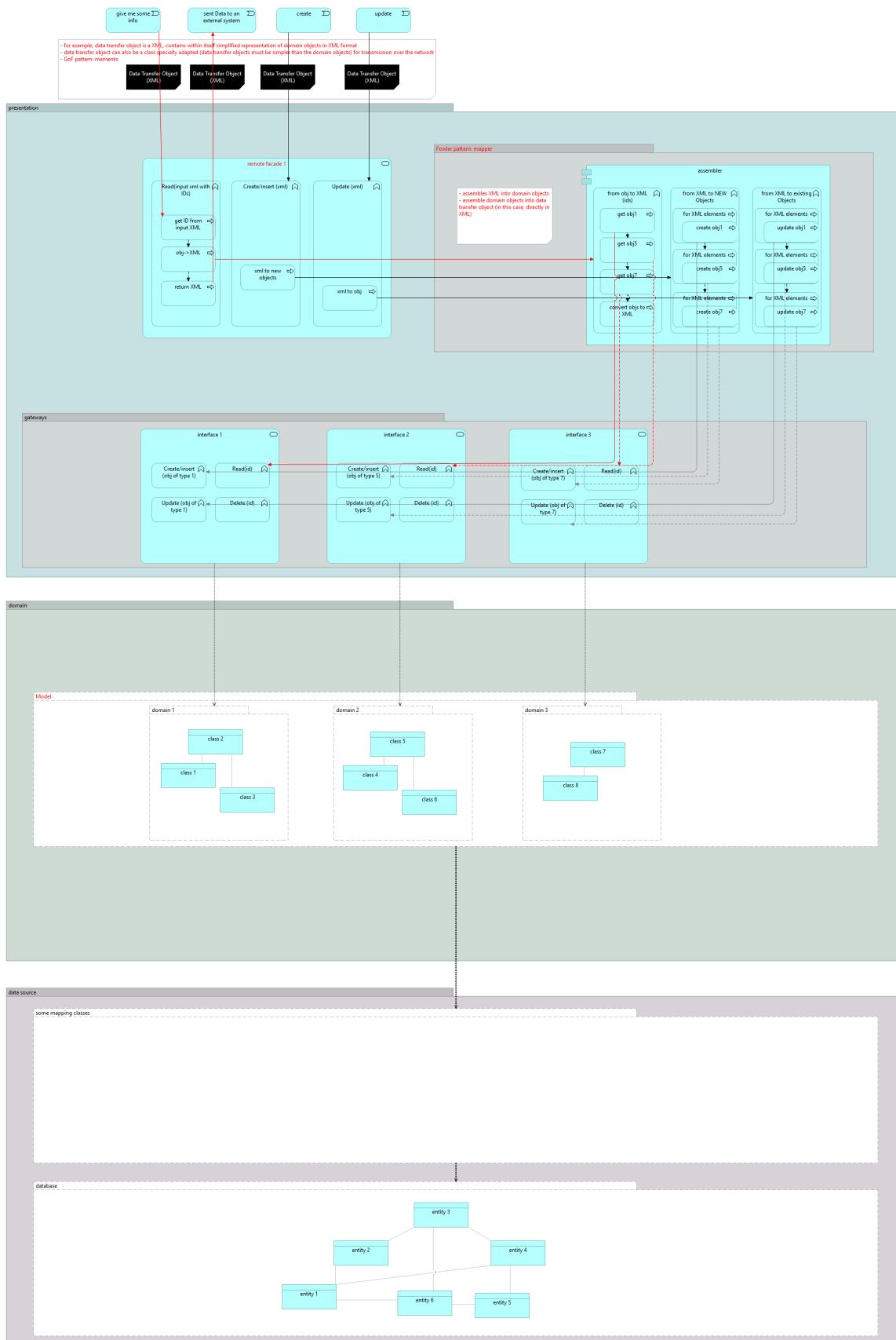
# DISTRIBUTED PROCESSING

ENTERPRISE PATTERNS

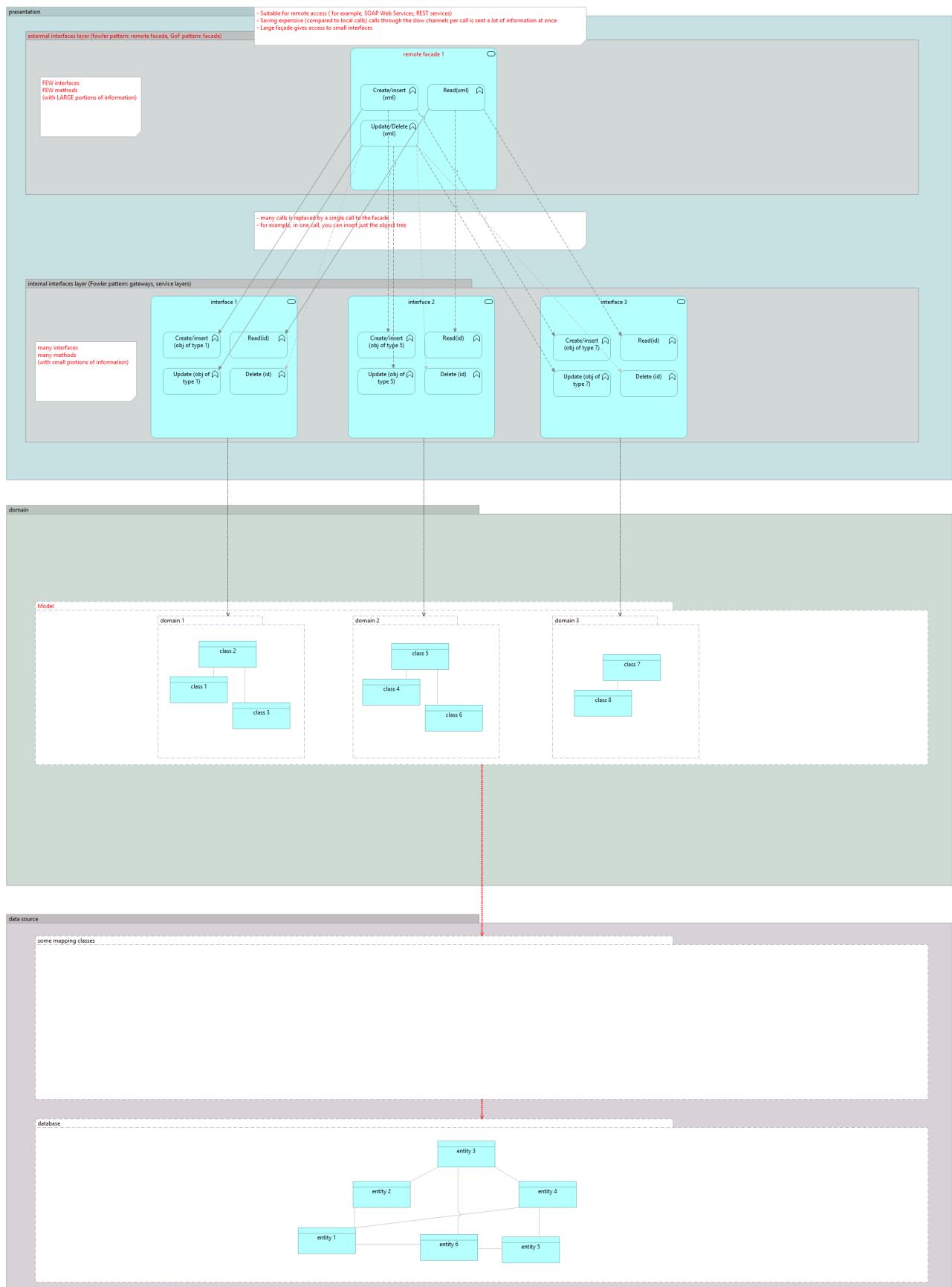
Martin Fowler



# DATA TRANSFER OBJECT



# REMOTE FAÇADE



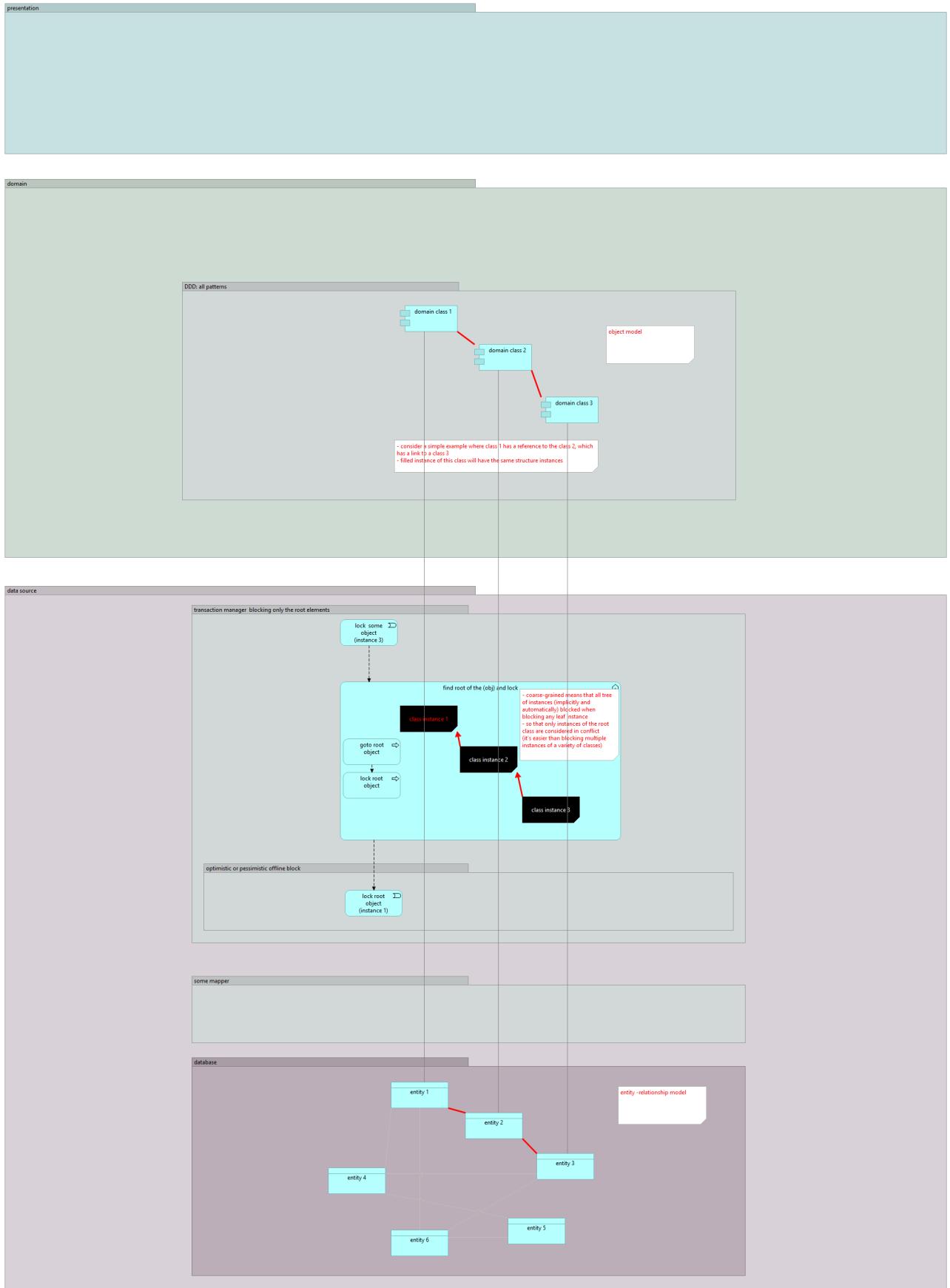
# PARALLEL PROCESSING

ENTERPRISE PATTERNS

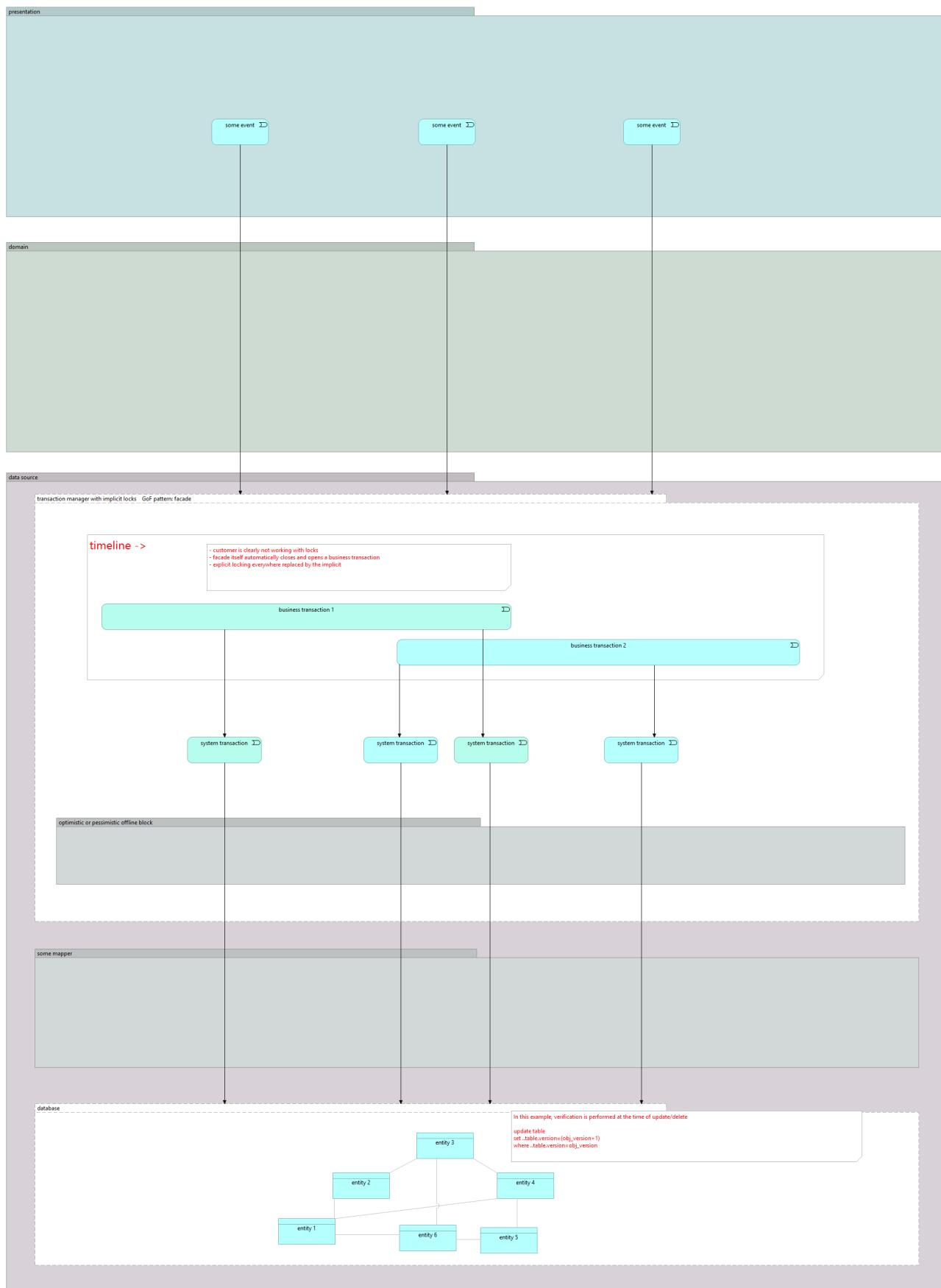
Martin Fowler



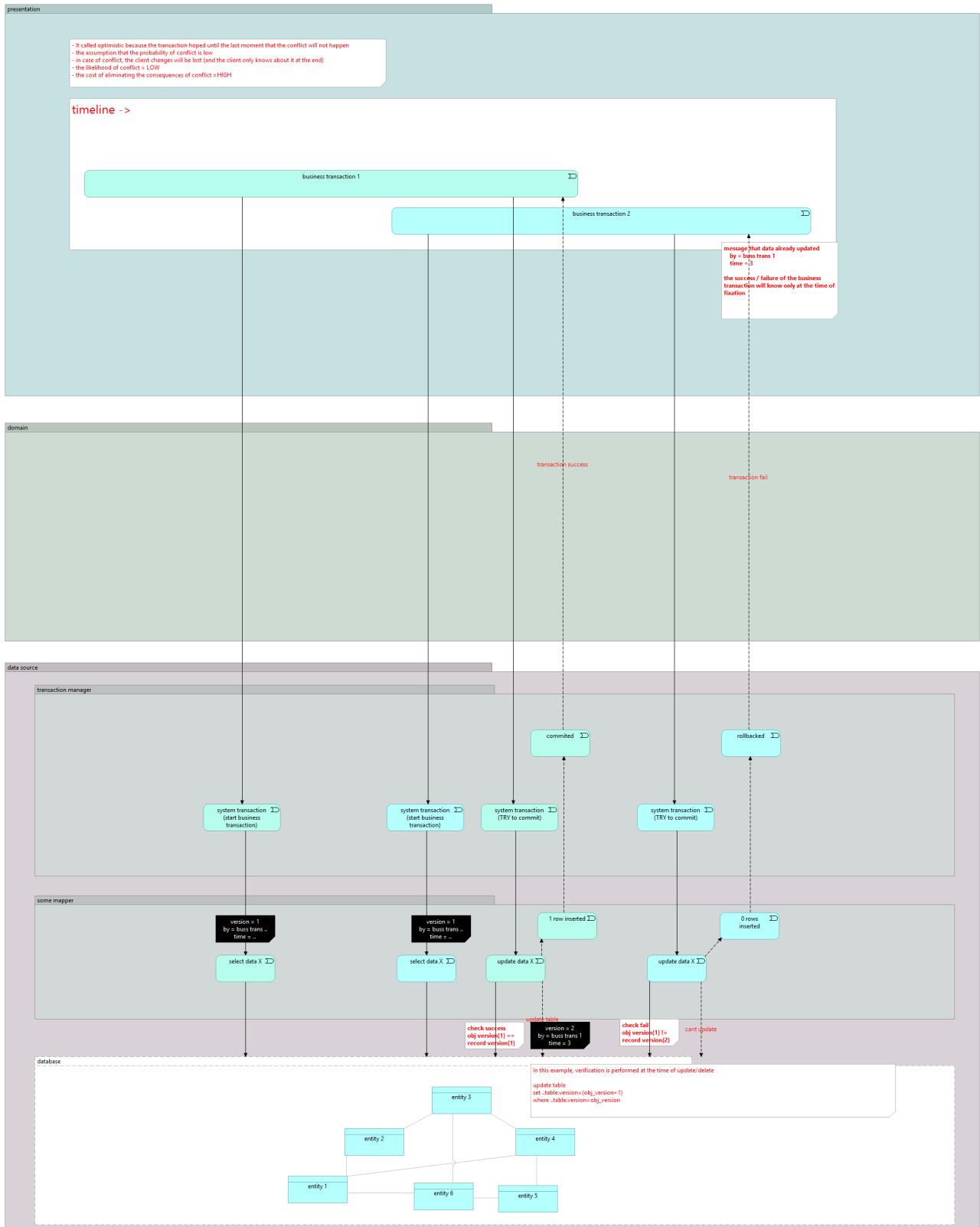
# COARSE-GRAINED LOCK



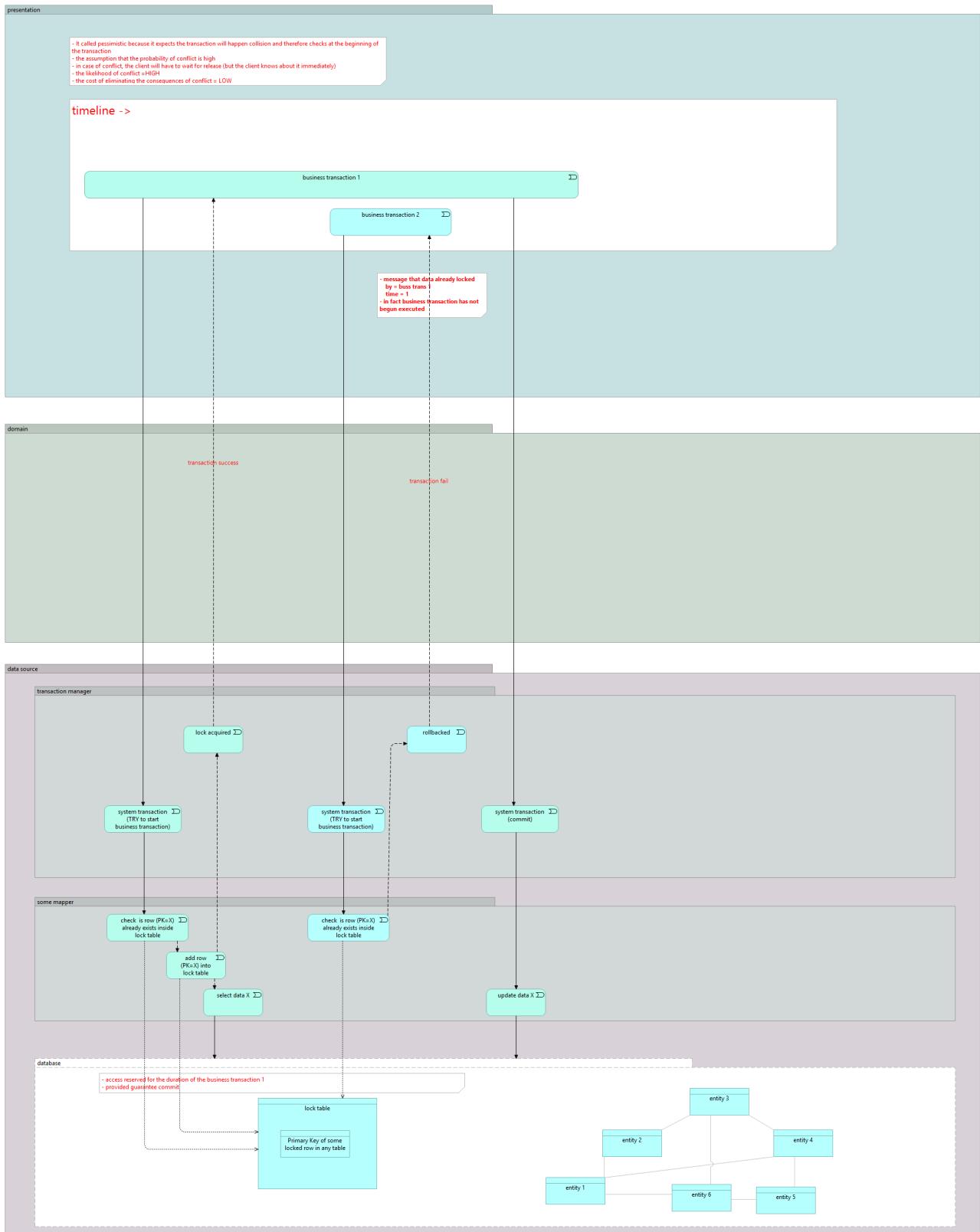
# IMPLICIT LOCK



# OPTIMISTIC OFFLINE LOCK



# PESSIMISTIC OFFLINE LOCK



# SESSION STATE

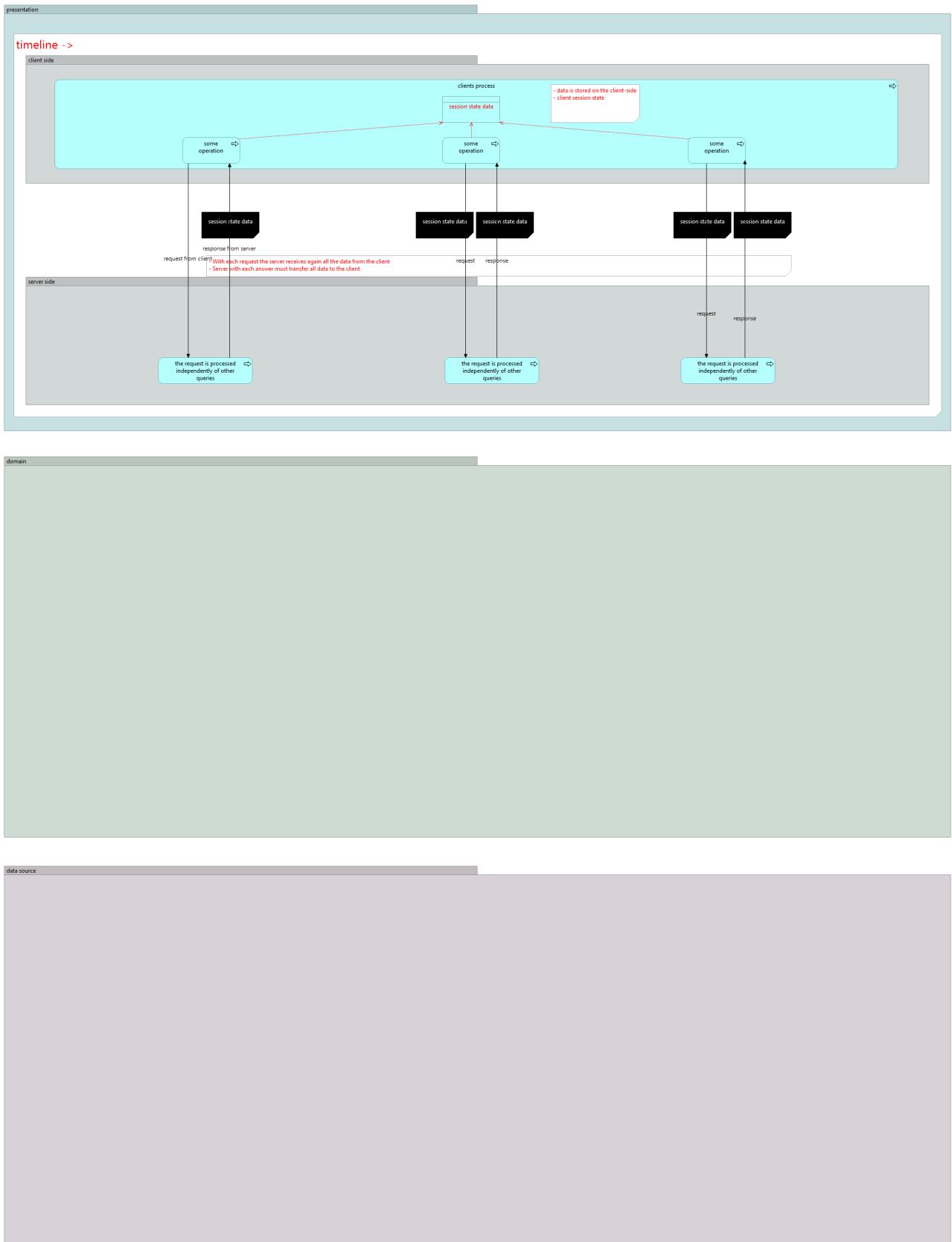
---

ENTERPRISE PATTERNS

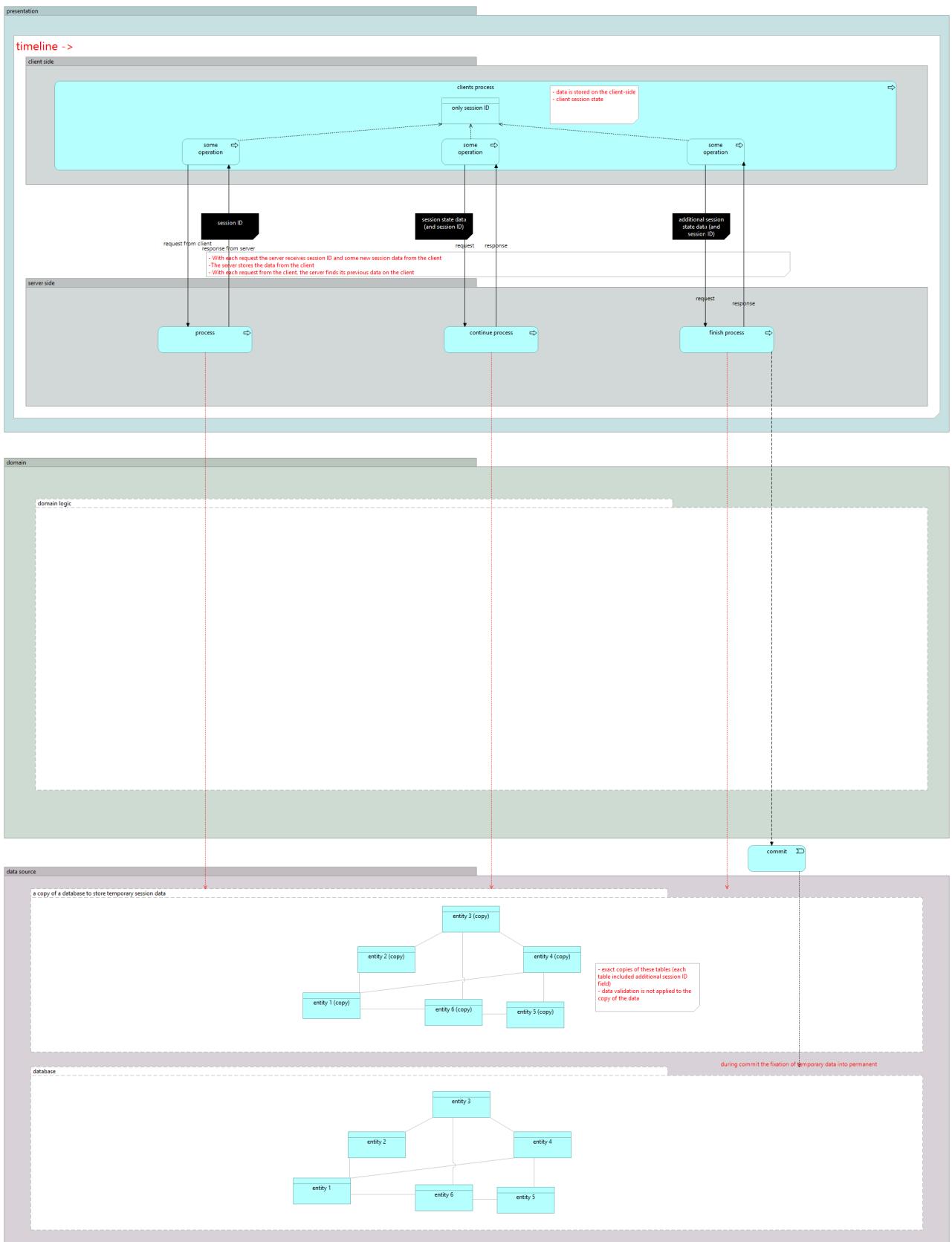
Martin Fowler



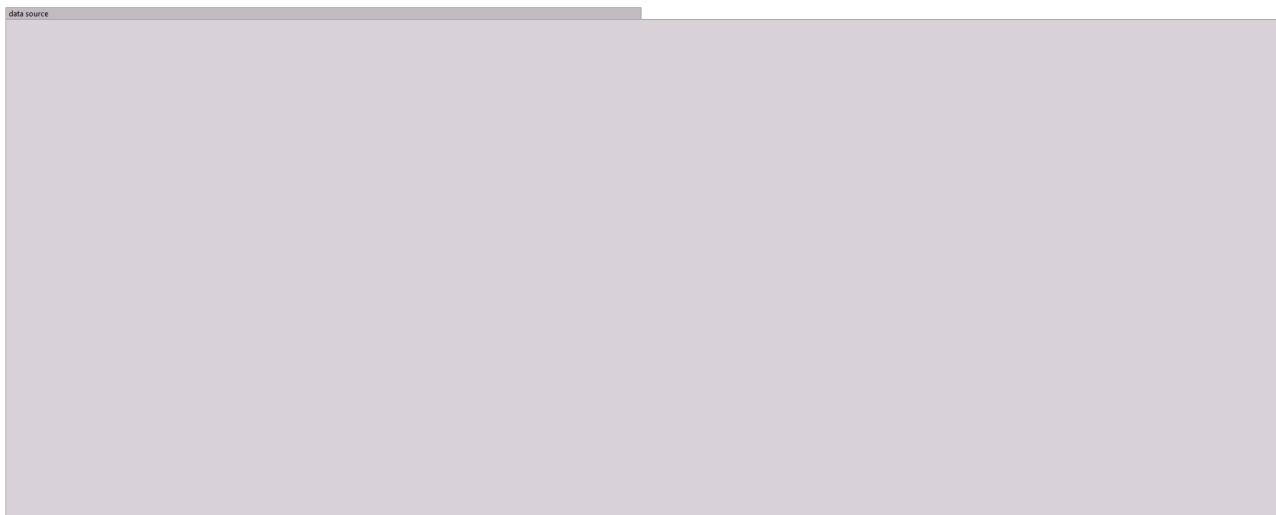
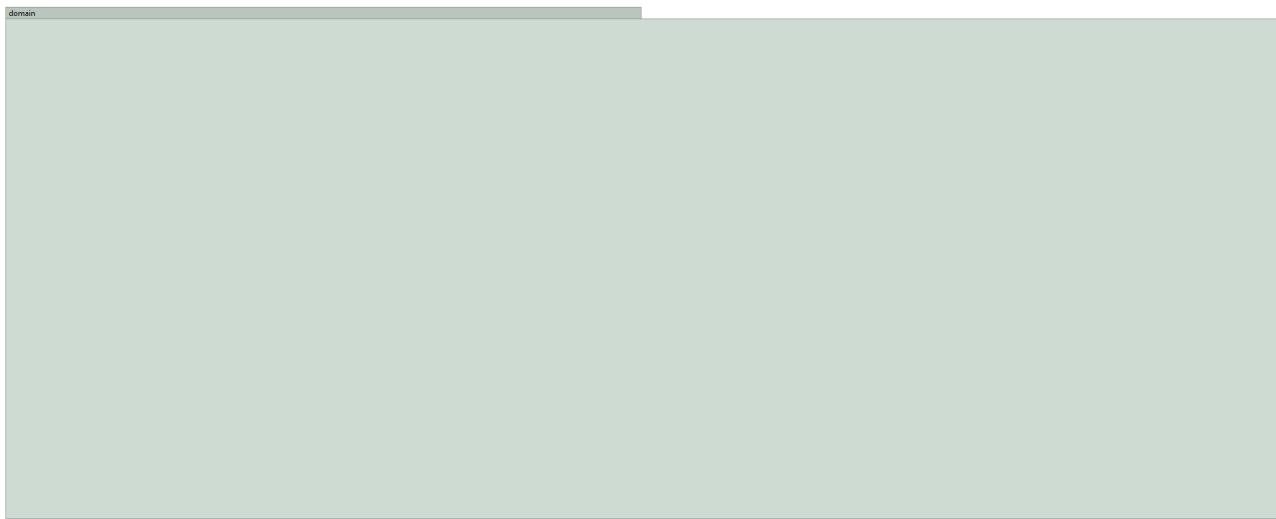
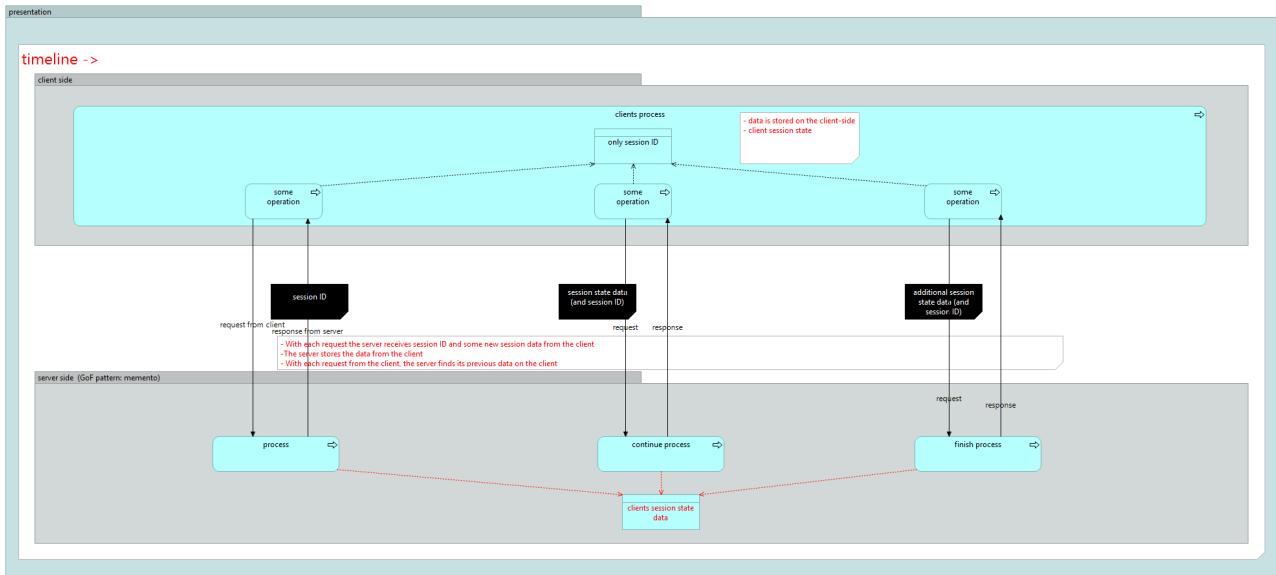
# CLIENT SESSION STATE



# DATABASE SESSION STATE



# SERVER SESSION STATE



# COMMON PATTERNS

---

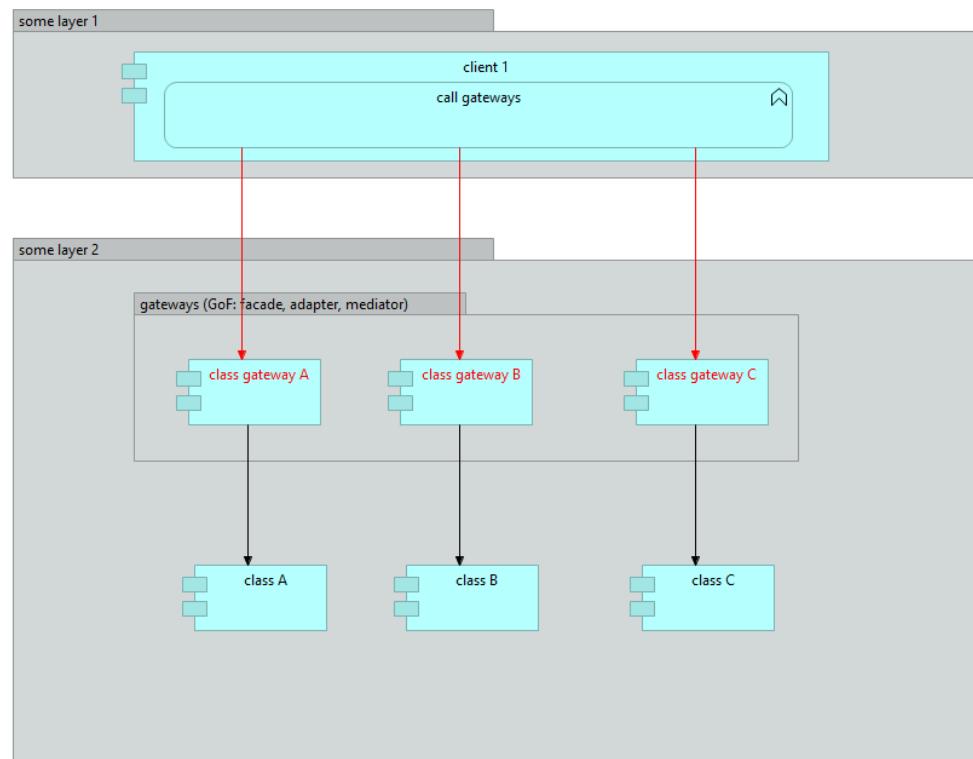
ENTERPRISE PATTERNS

Martin Fowler

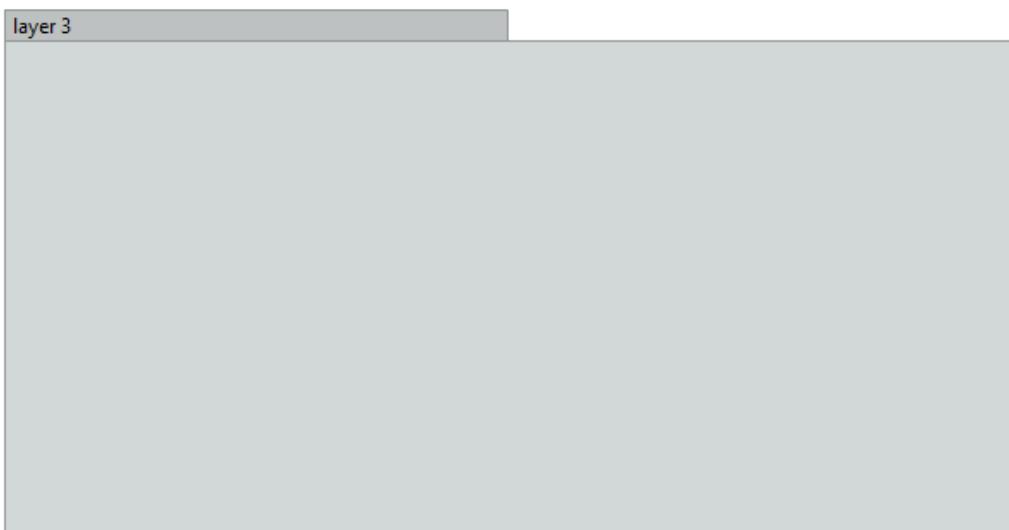
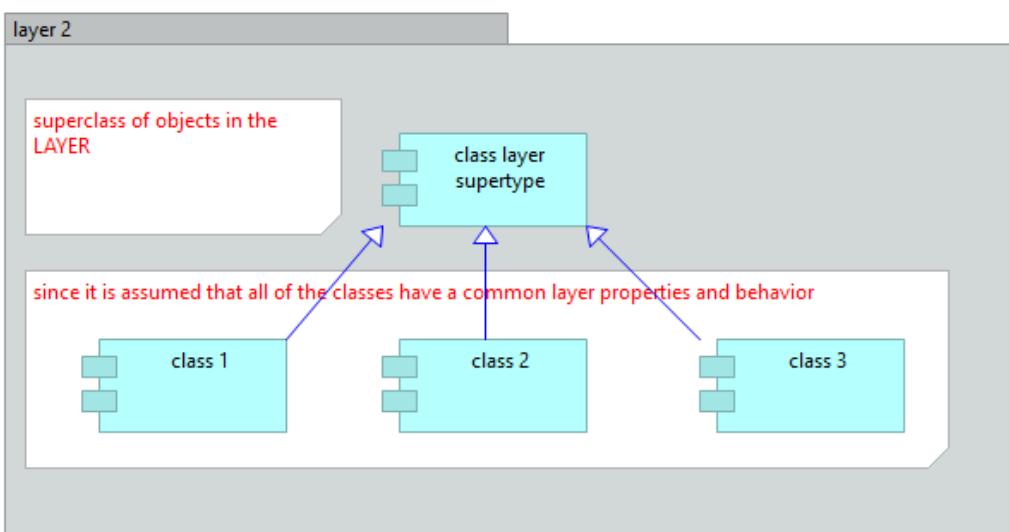


# GATEWAY

- 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



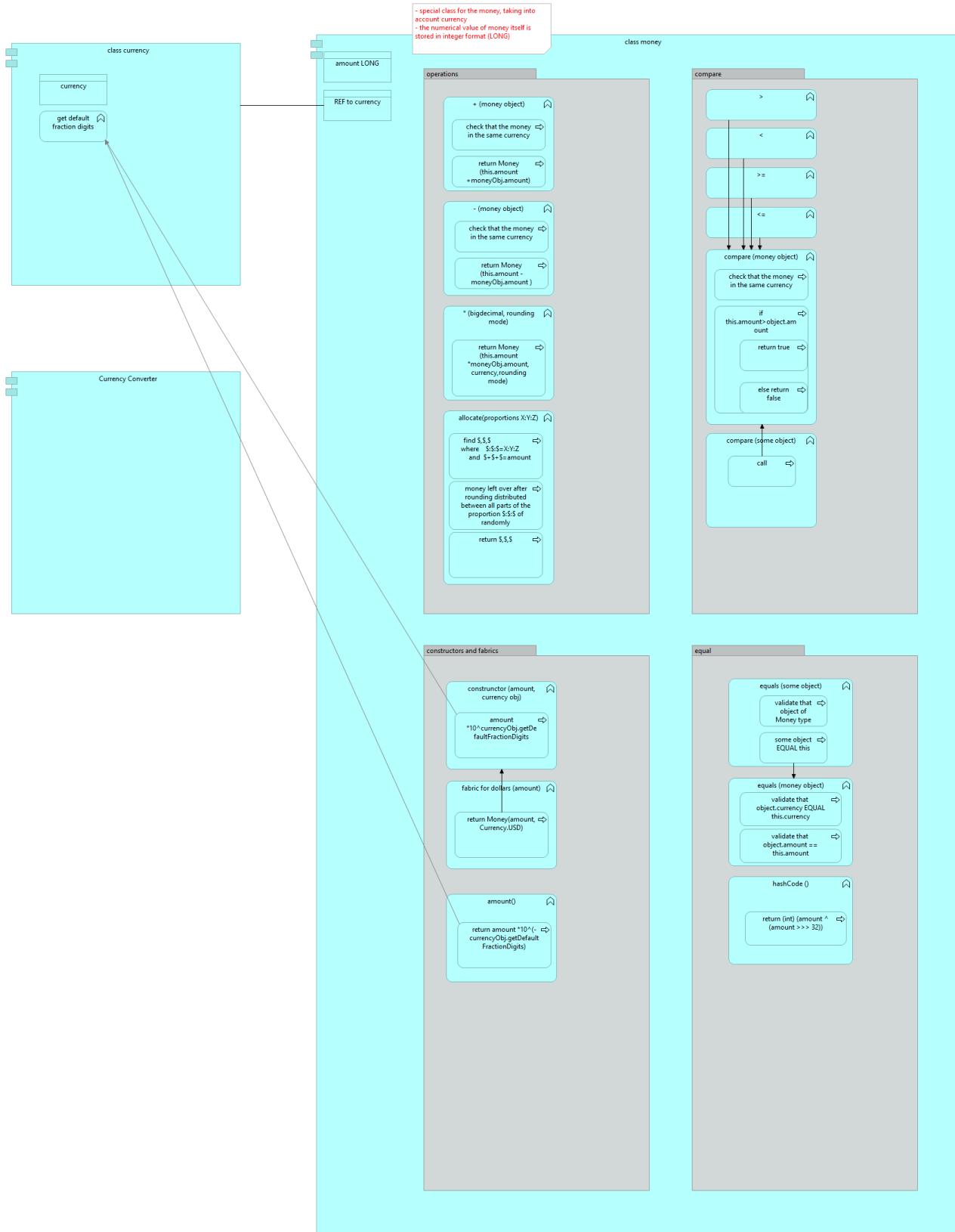
# LAYER SUPERTYPE



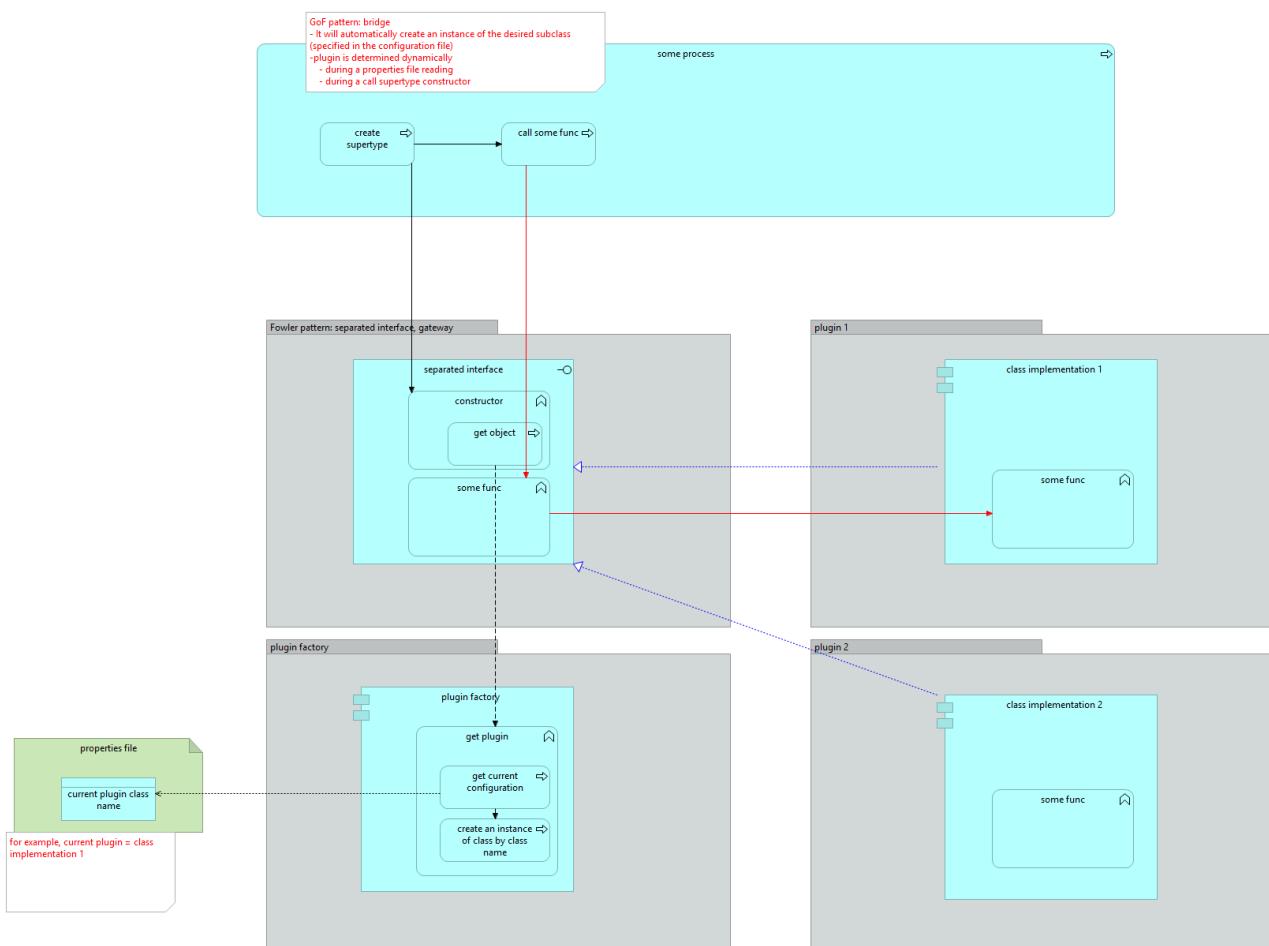
# MAPPER



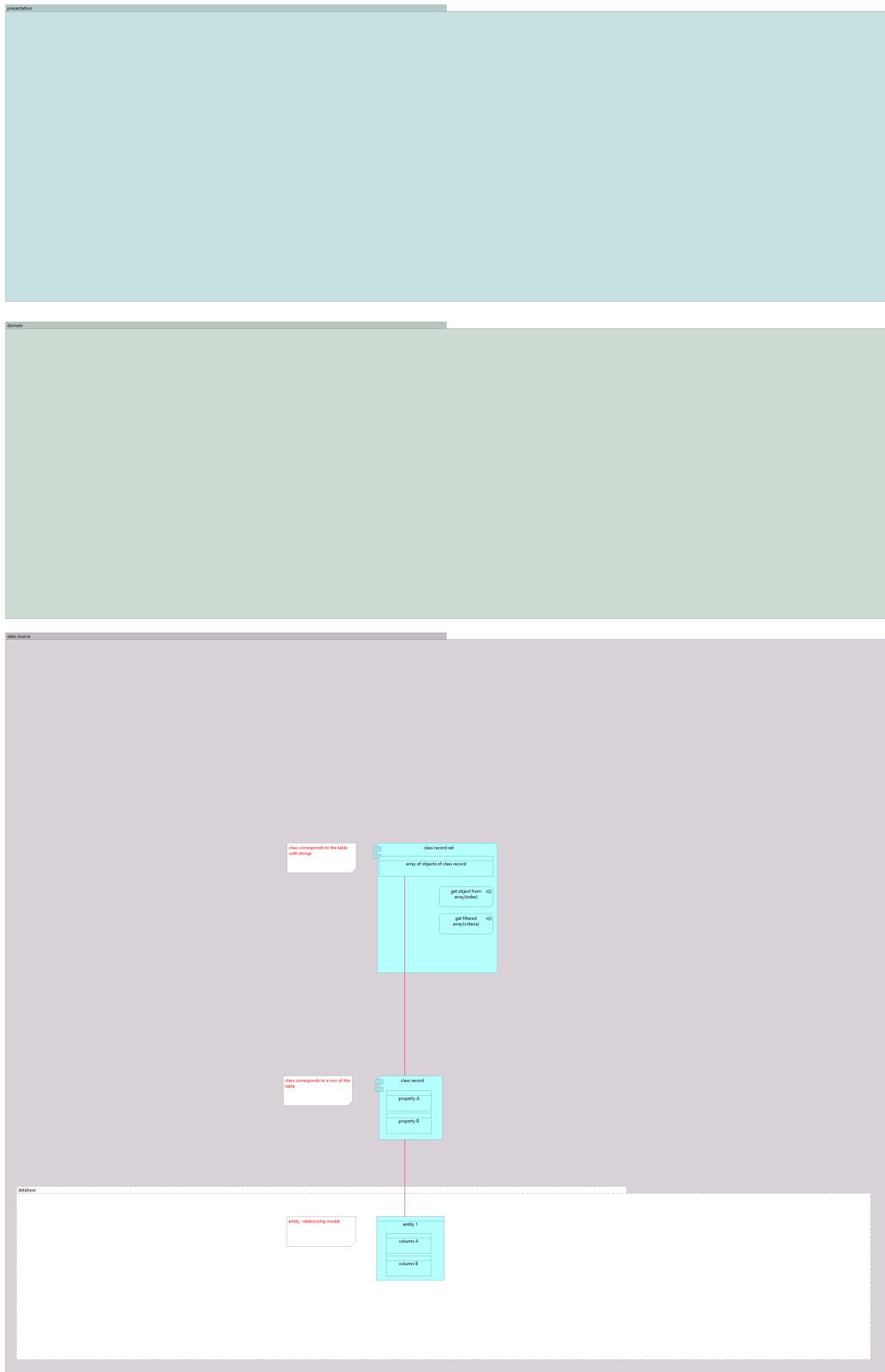
# MONEY



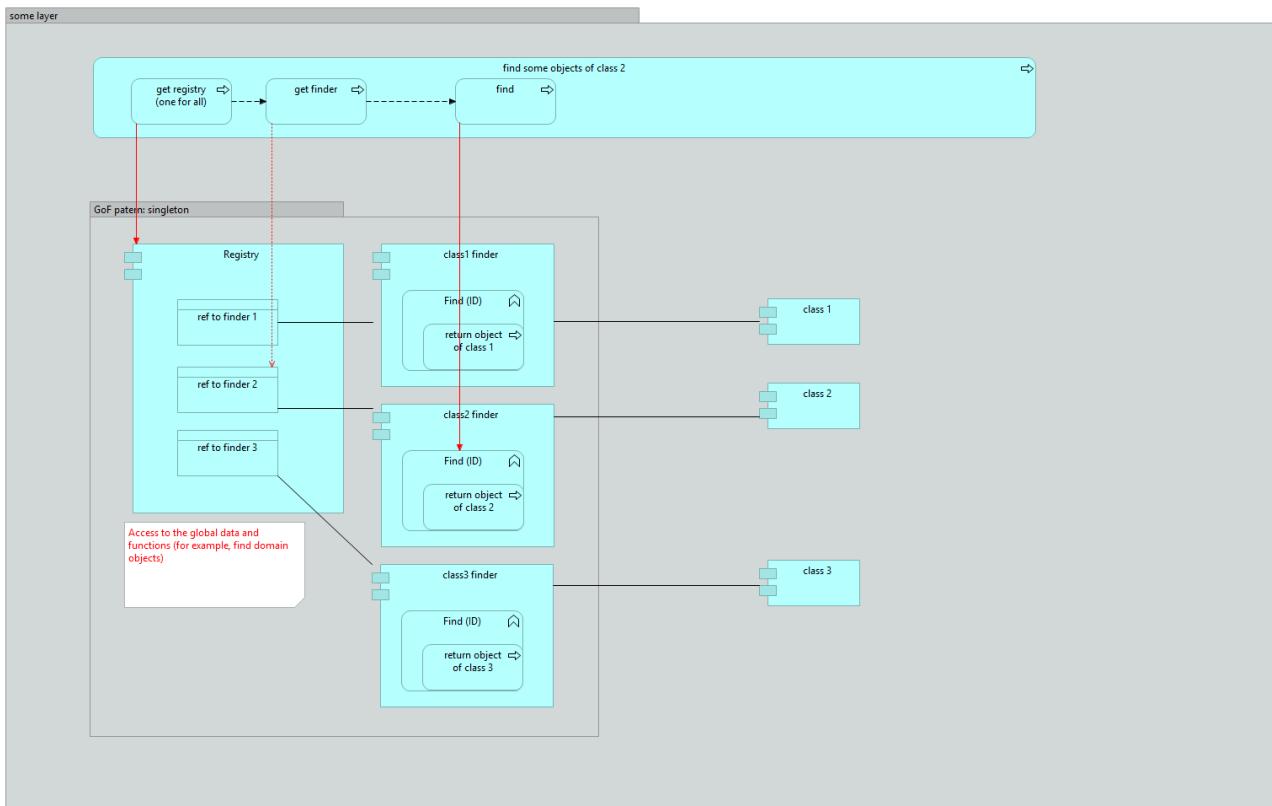
# PLUGIN



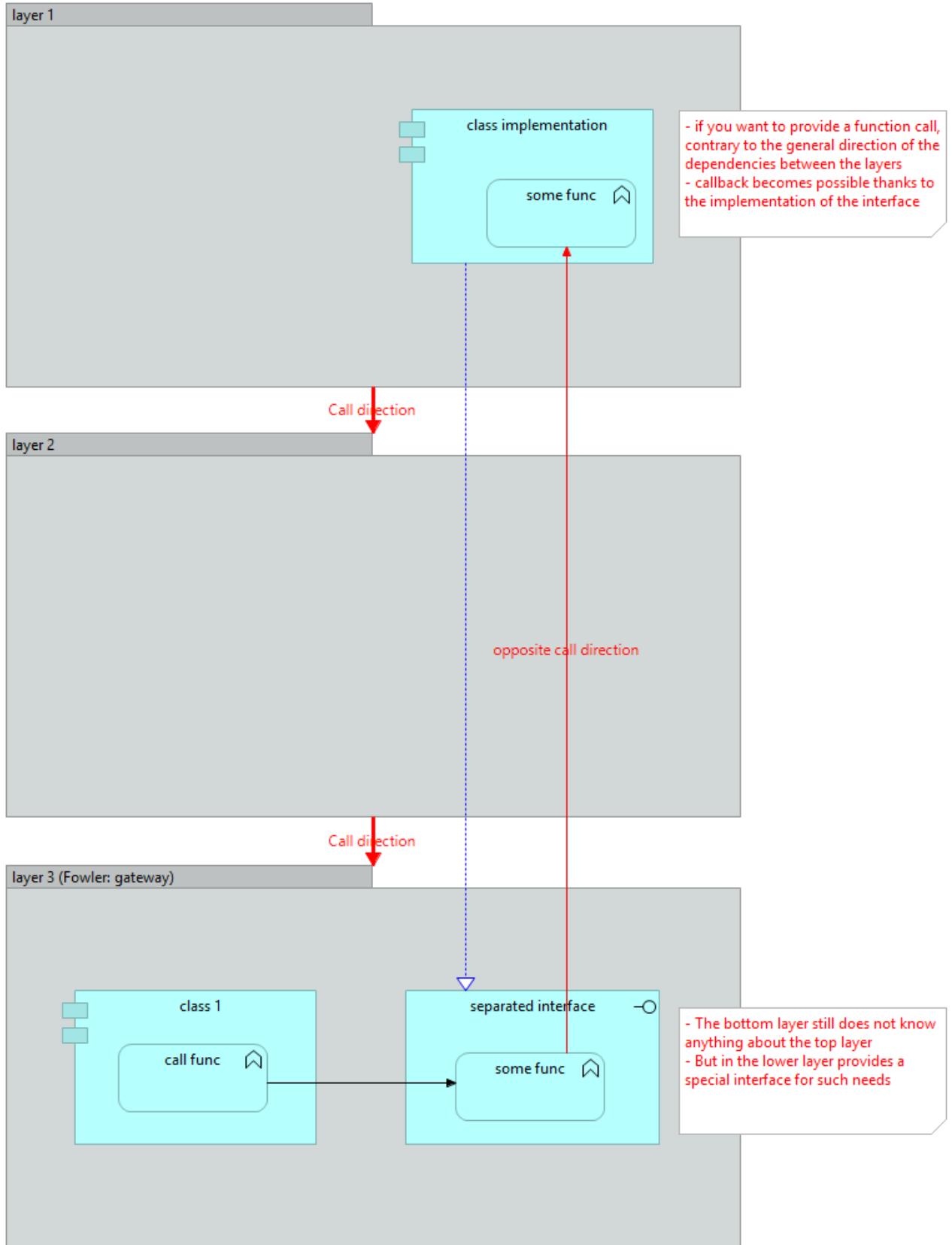
# RECORD SET



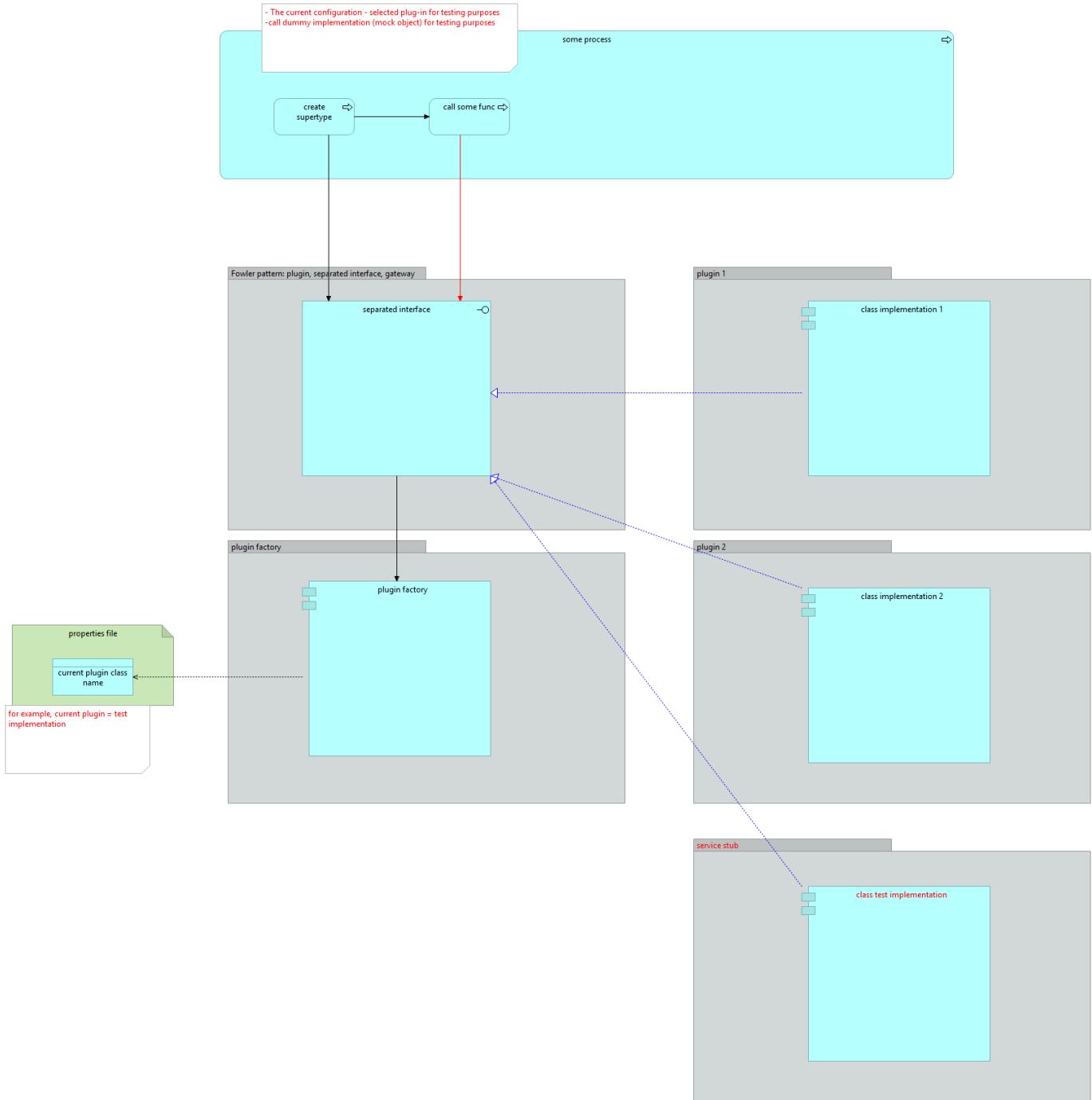
# REGISTRY



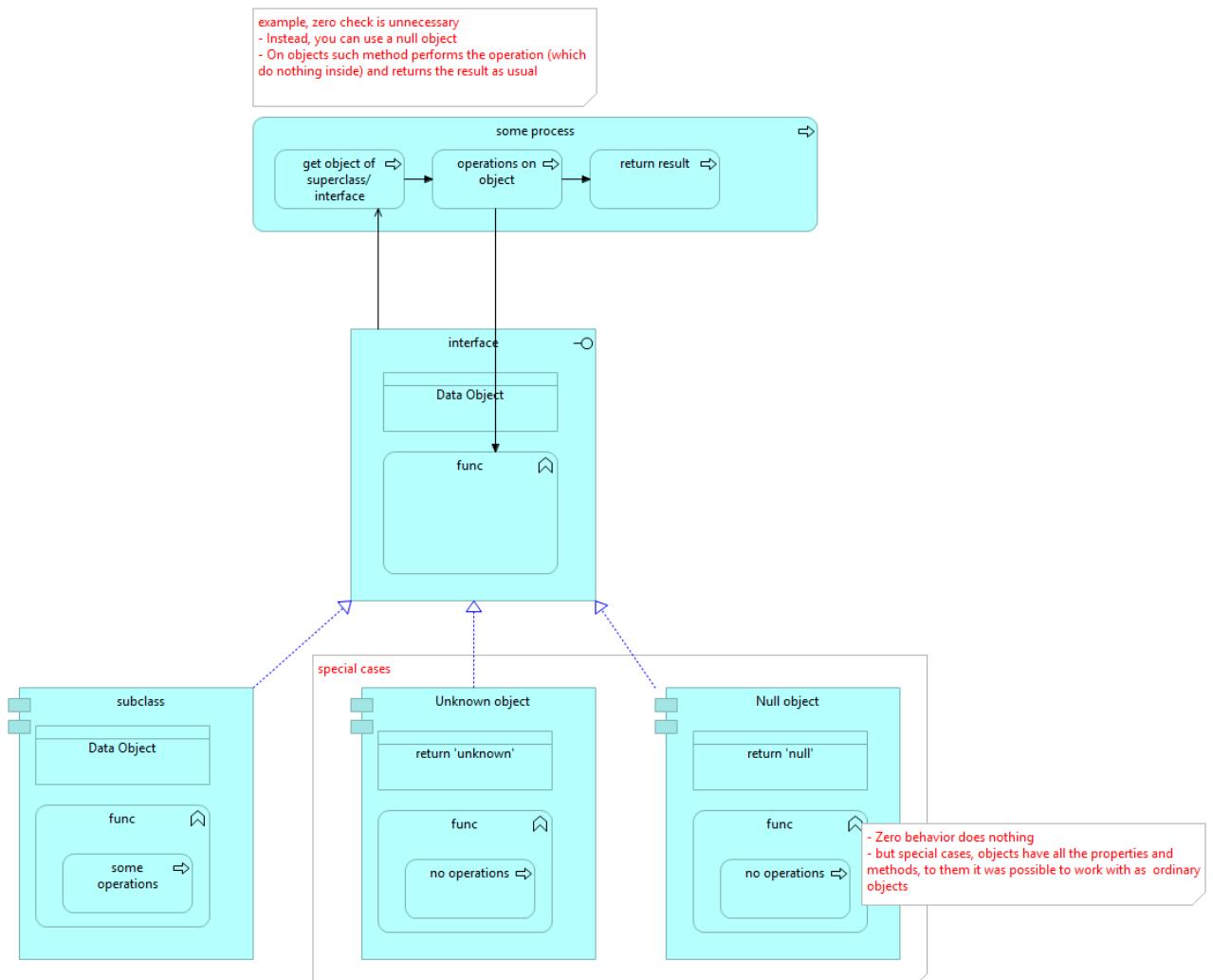
# SEPARATED INTERFACE



# SERVICE STUB

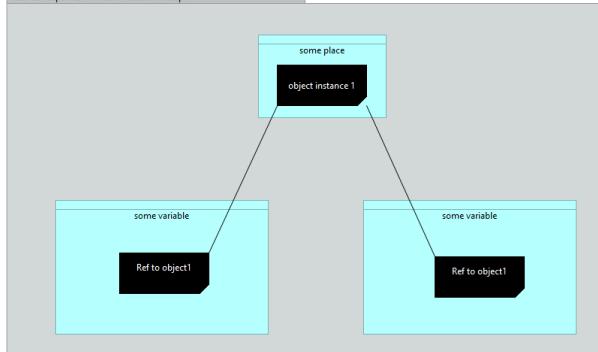


# SPECIAL CASE

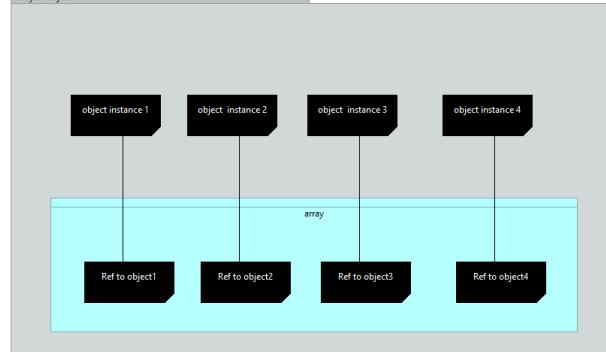


# VALUE OBJECT

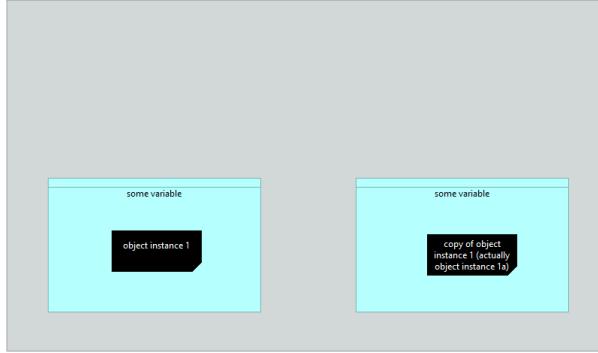
class concept and link transmission concept



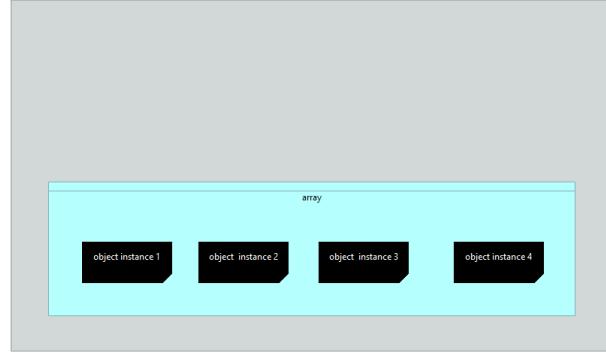
array of objects



value object concept and by value transfer concept



array of value objects

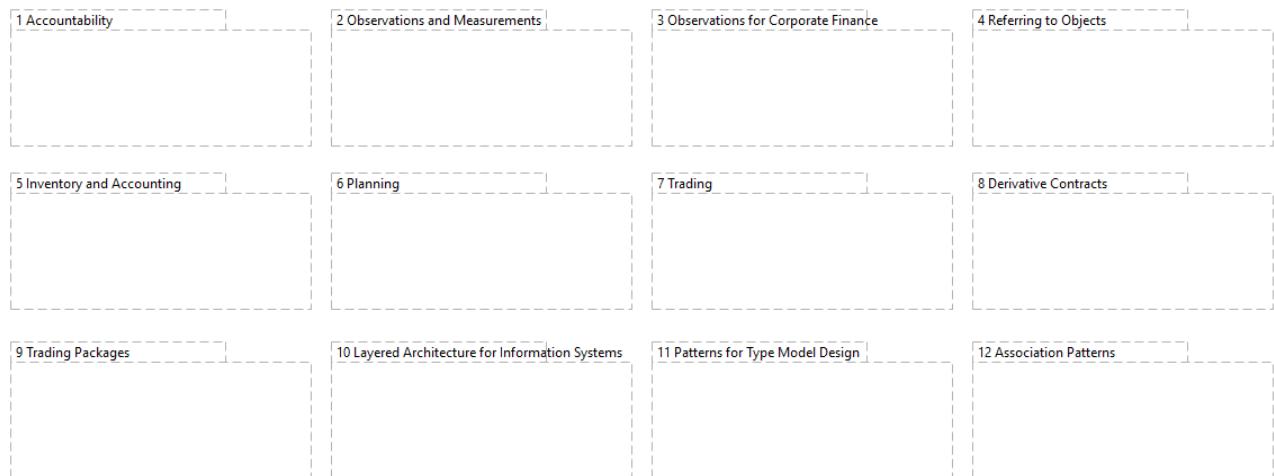


03

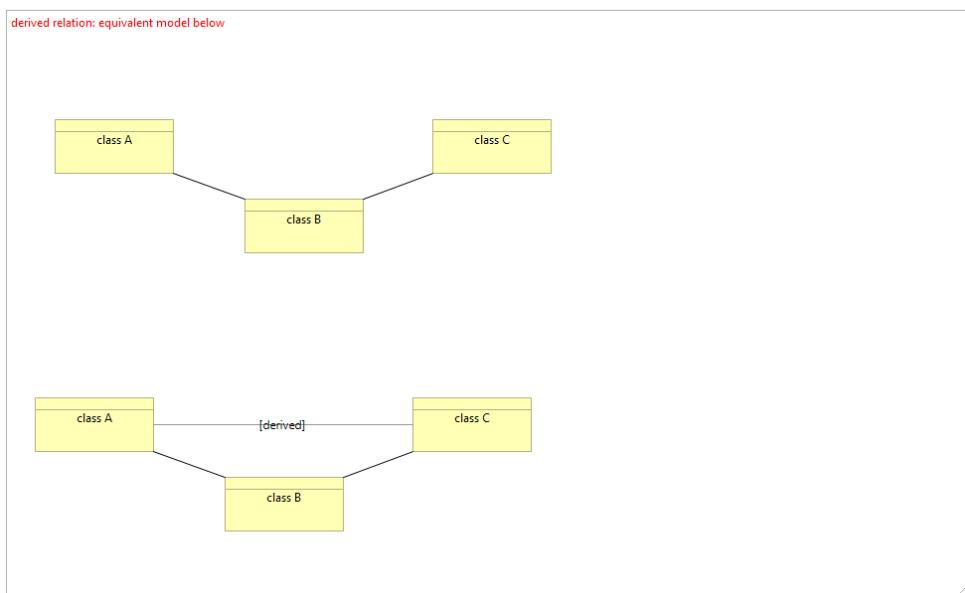
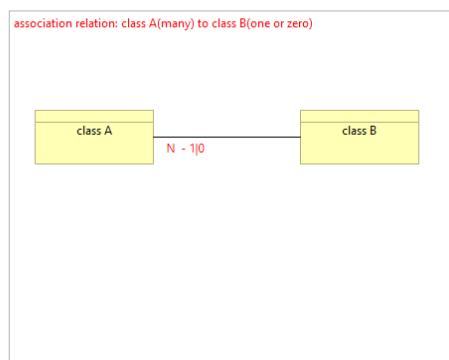
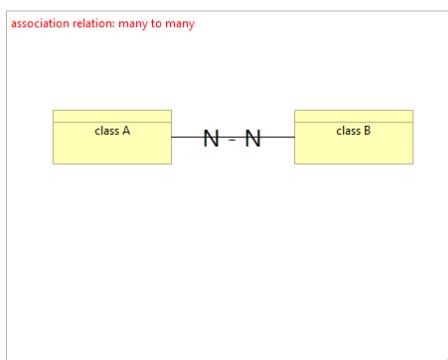
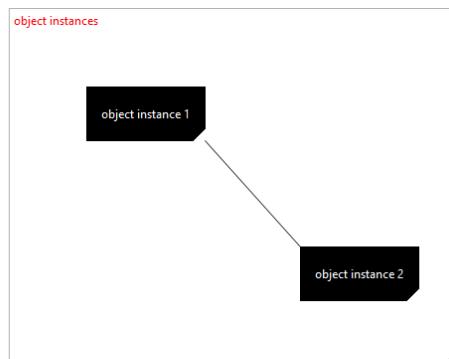
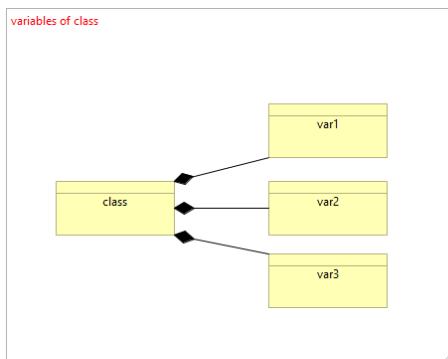
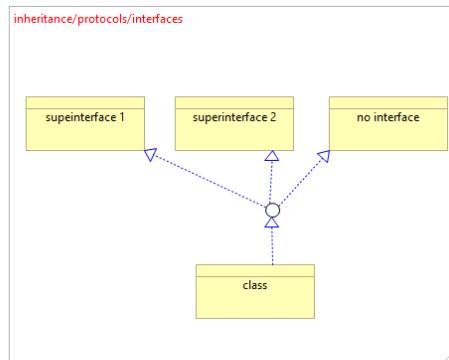
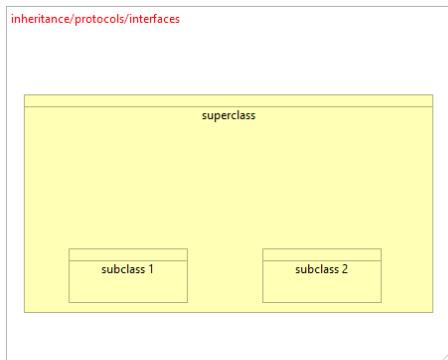
# Analysis Patterns

Reusable Object Models

MARTIN FOWLER



# USED NOTATION

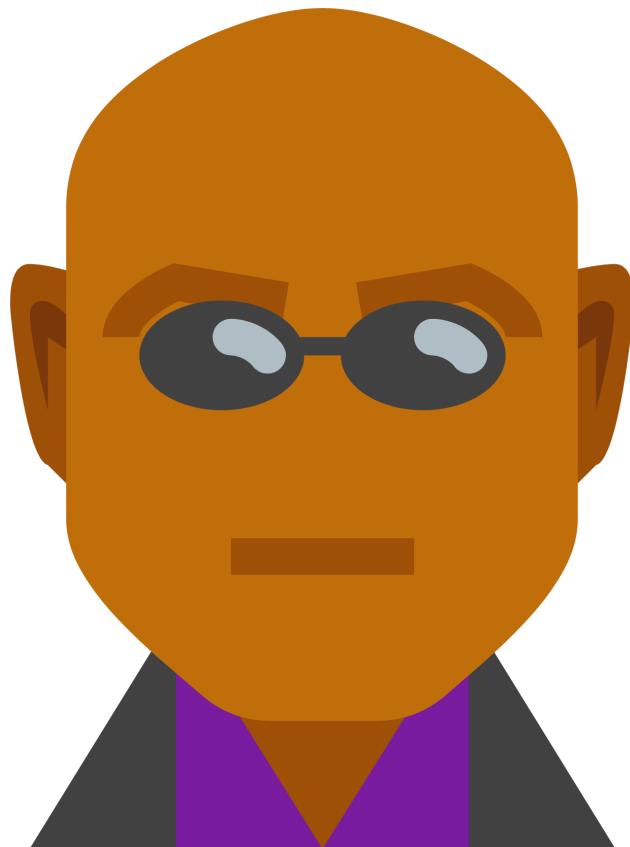


# ACCOUNTABILITY

---

ANALYSIS PATTERNS

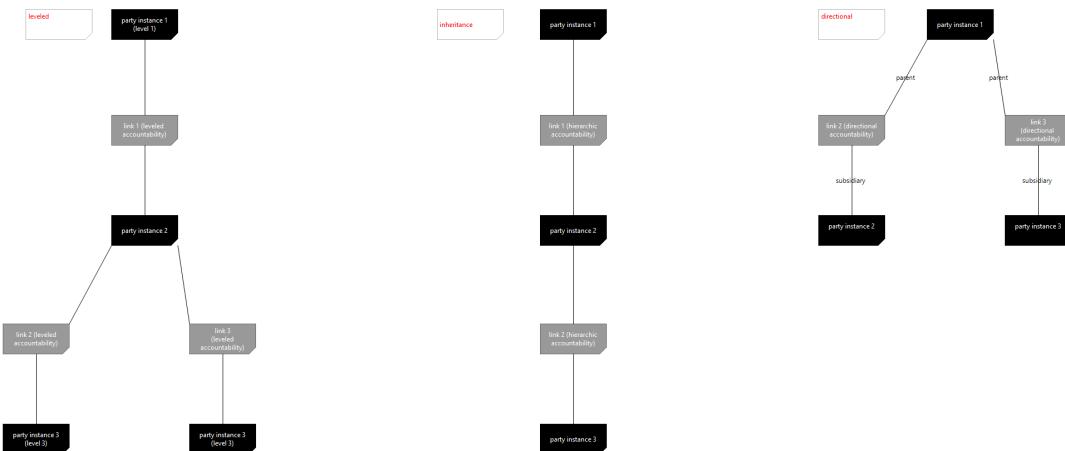
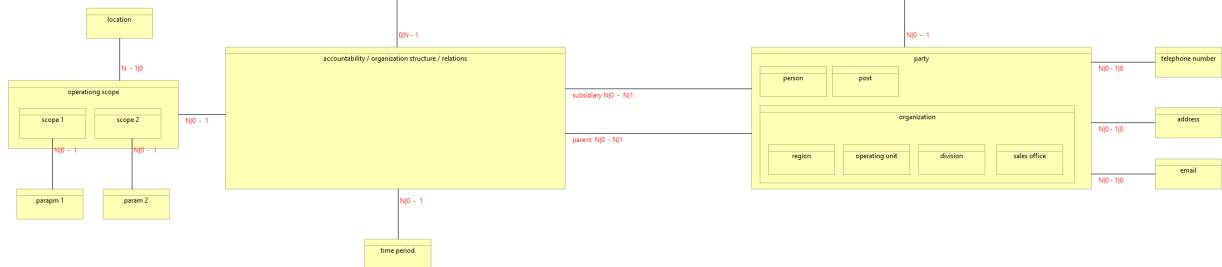
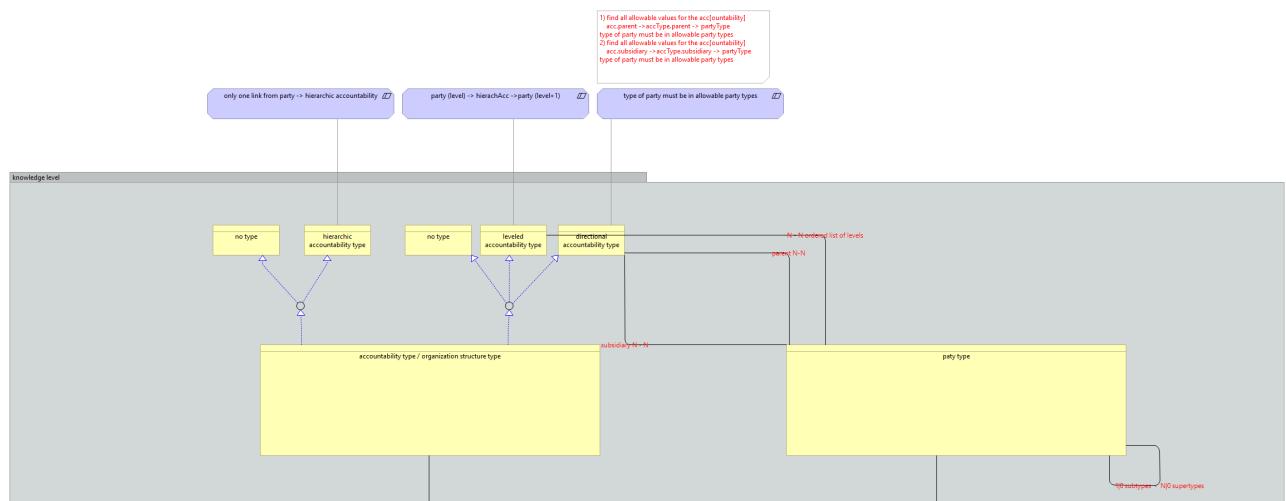
Martin Fowler

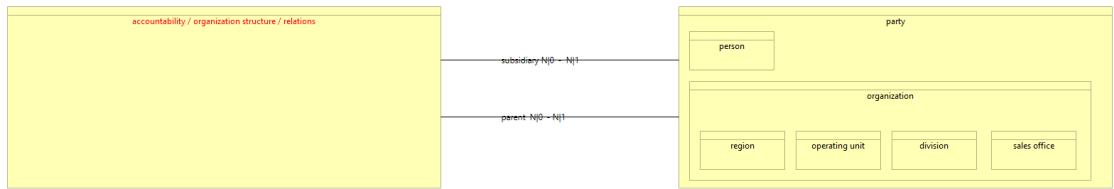


# PARTY

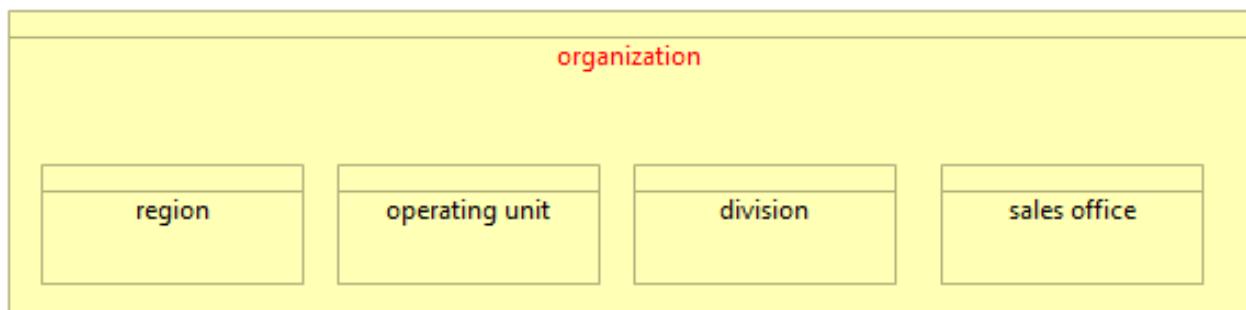


# ACCOUNTABILITY

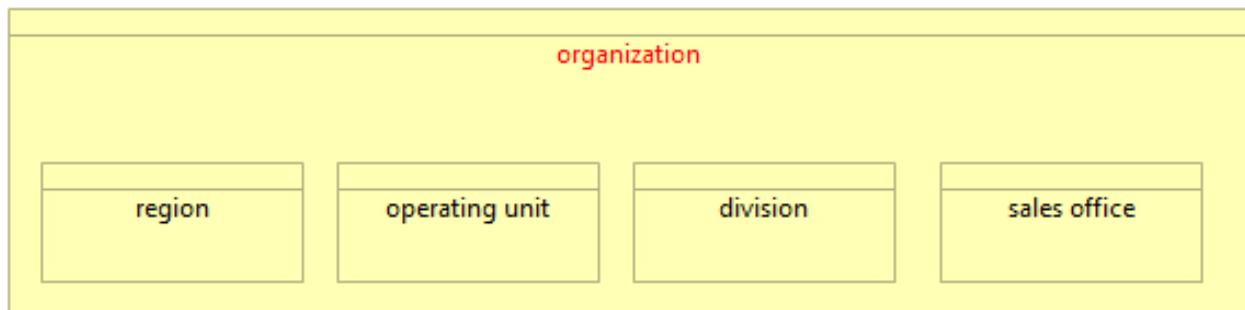




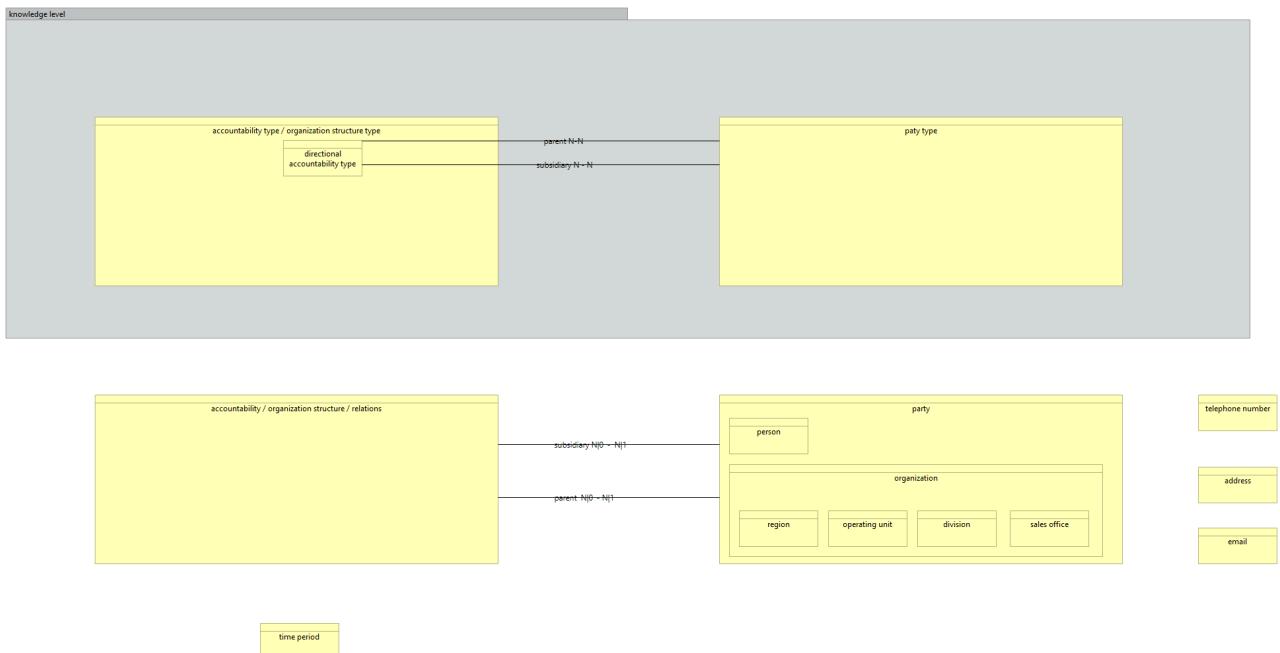
# ORGANIZATION HIERARCHIES



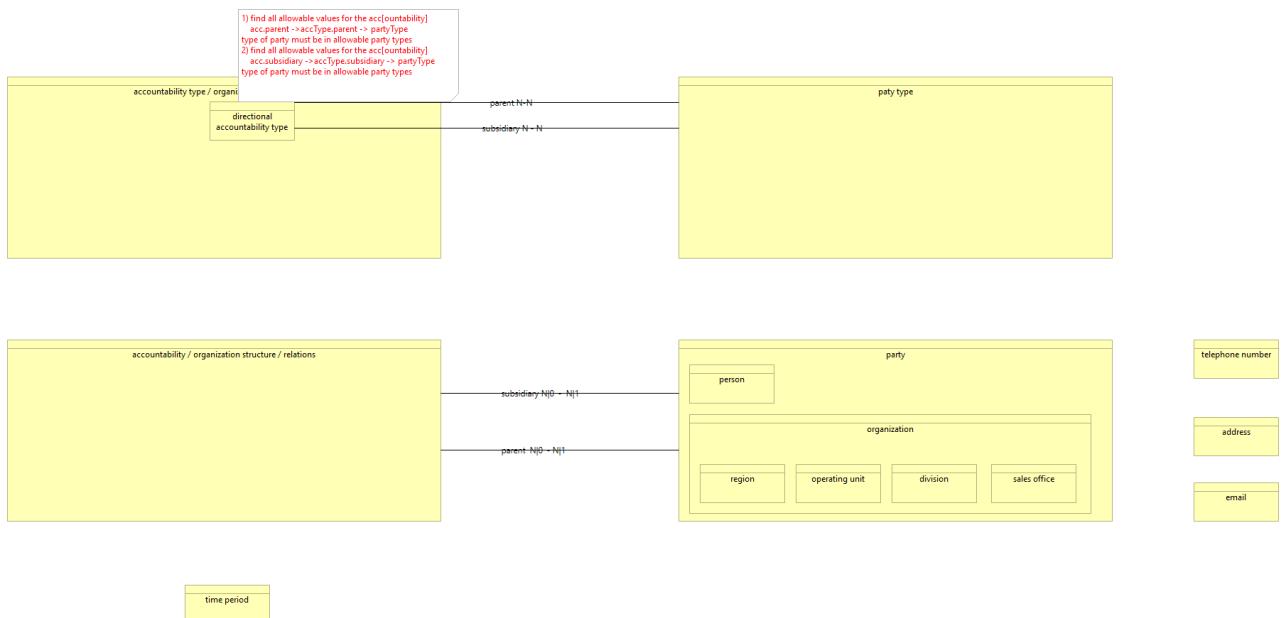
# ORGANIZATION STRUCTURE



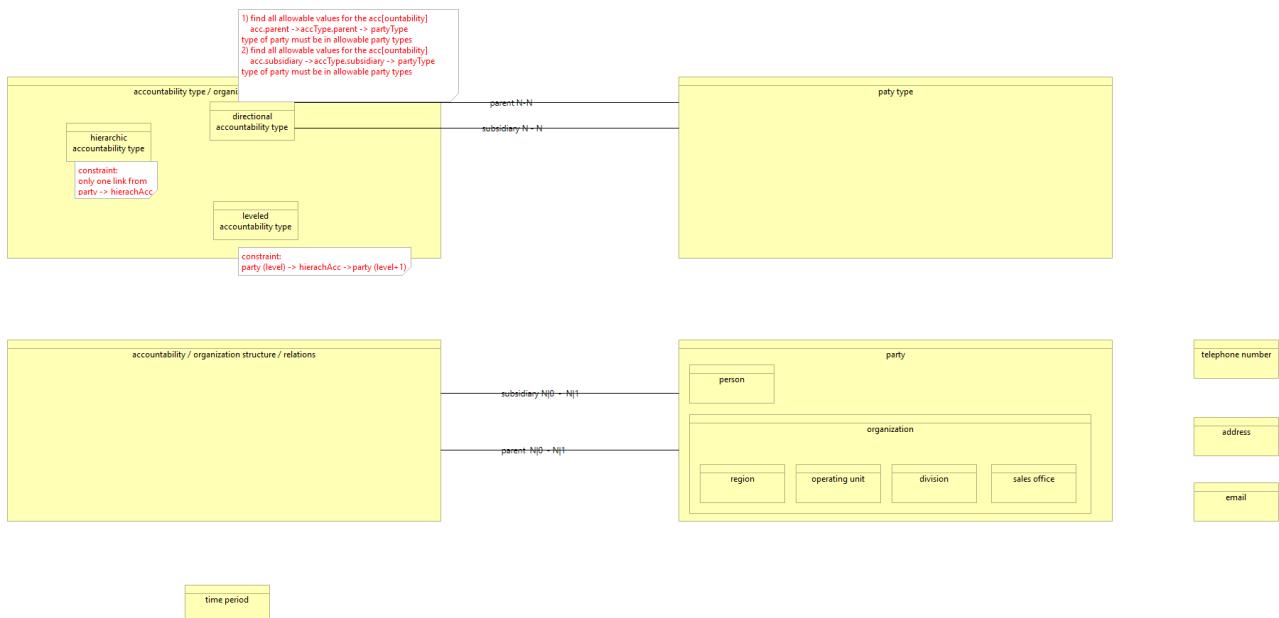
# ACCOUNTABILITY KNOWLEDGE LEVEL



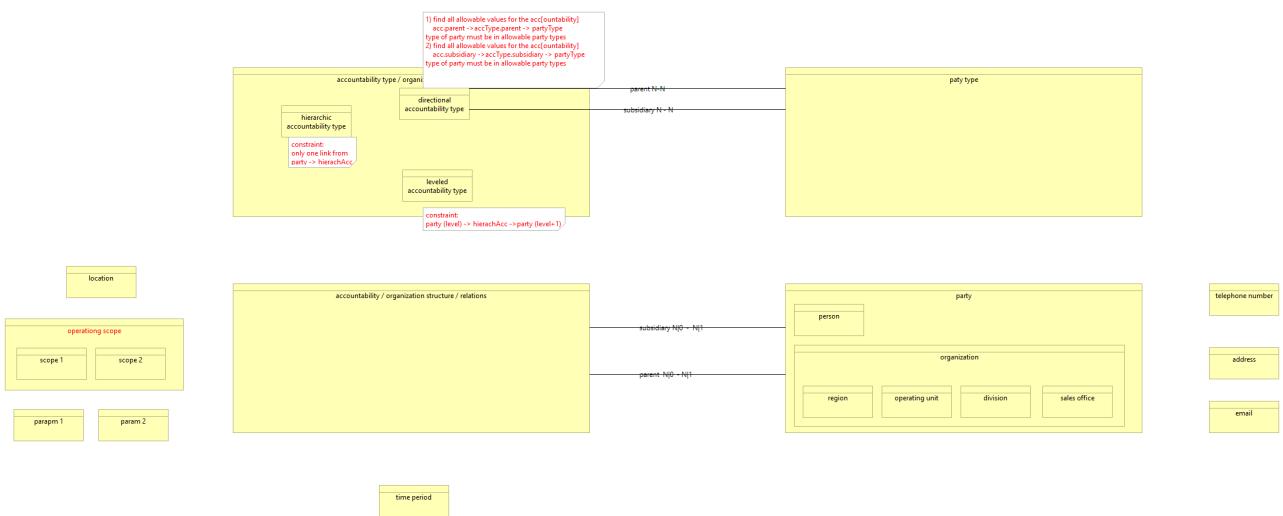
# PARTY TYPE GENERALIZATIONS



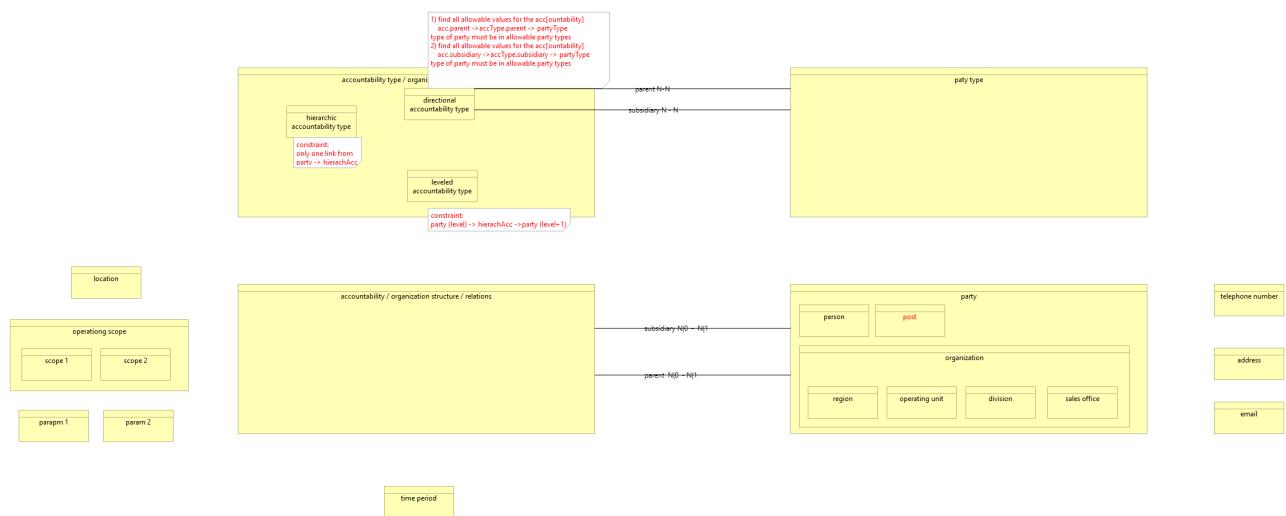
# HIERARCHIC ACCOUNTABILITY



# OPERATING SCOPES



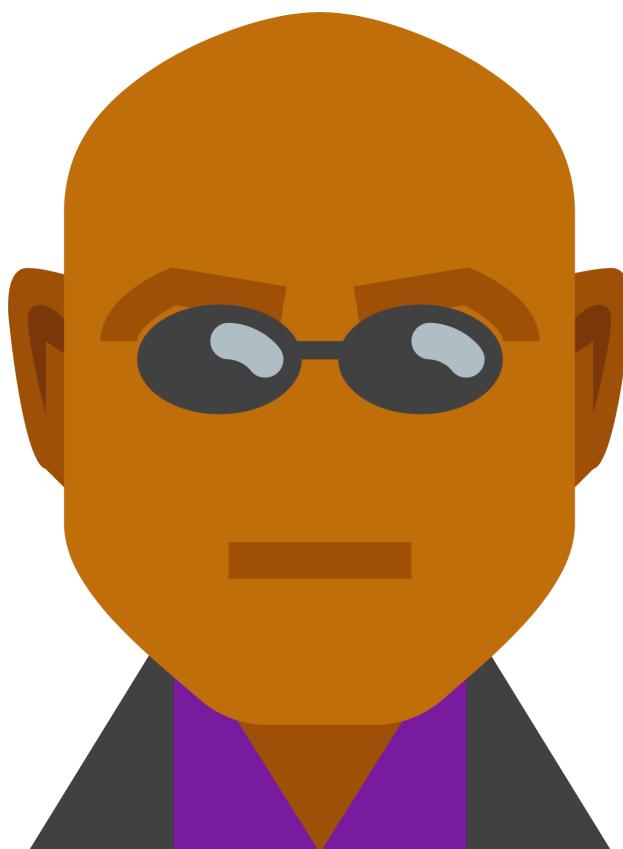
# POST



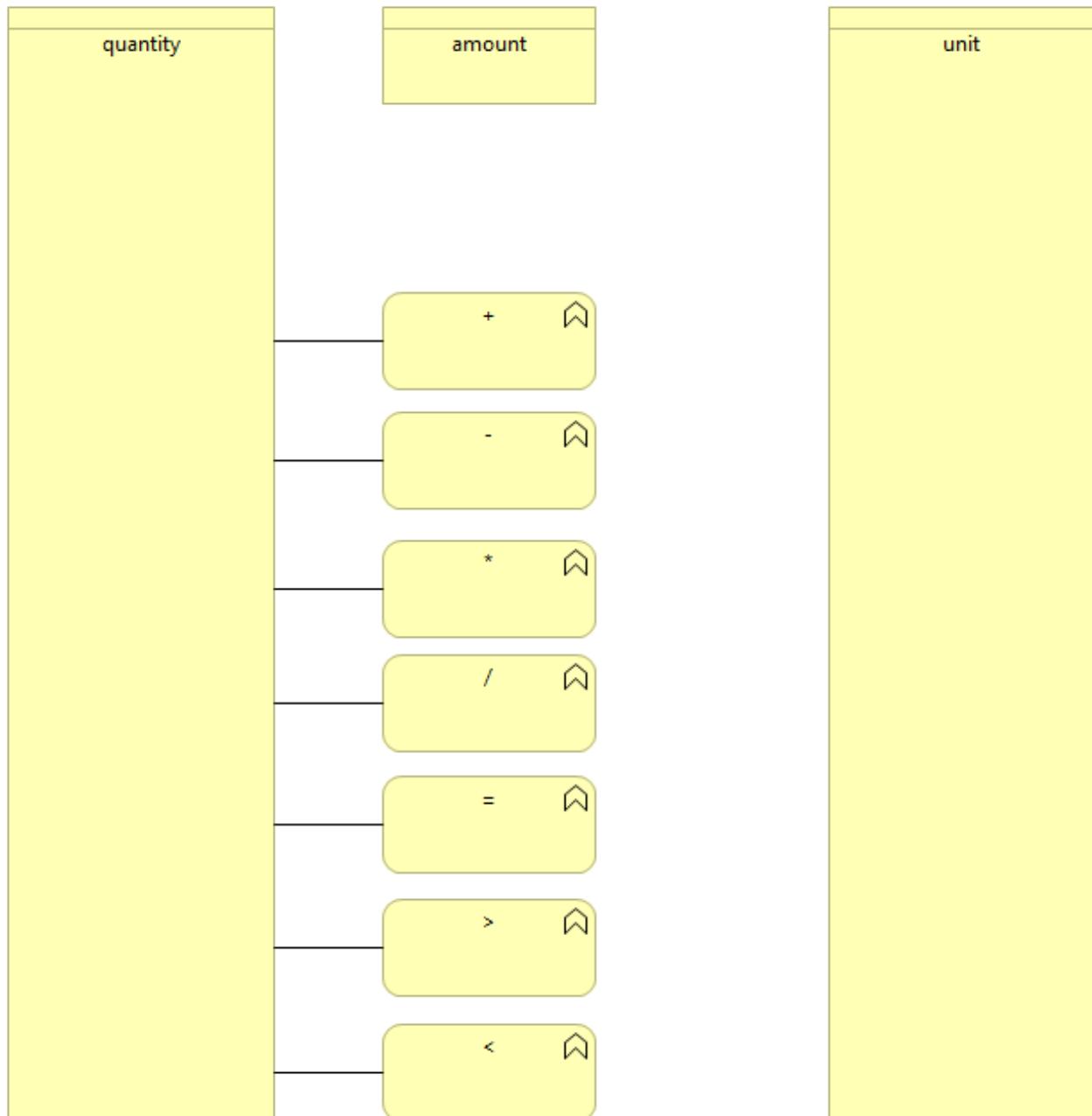
# OBSERVATIONS AND MEASUREMENTS

ANALYSIS PATTERNS

Martin Fowler



# QUANTITY

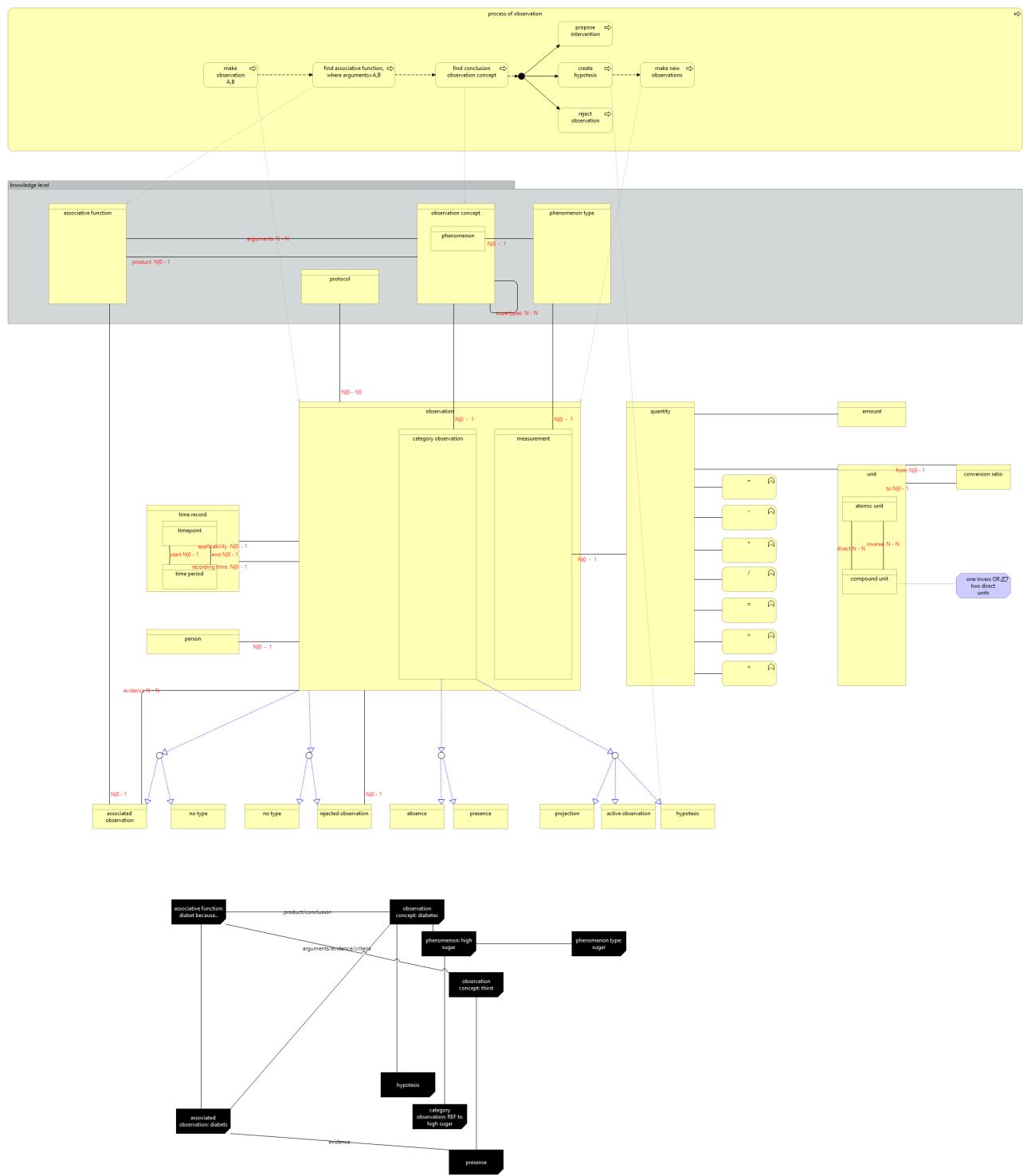


# CONVERSION RATIO

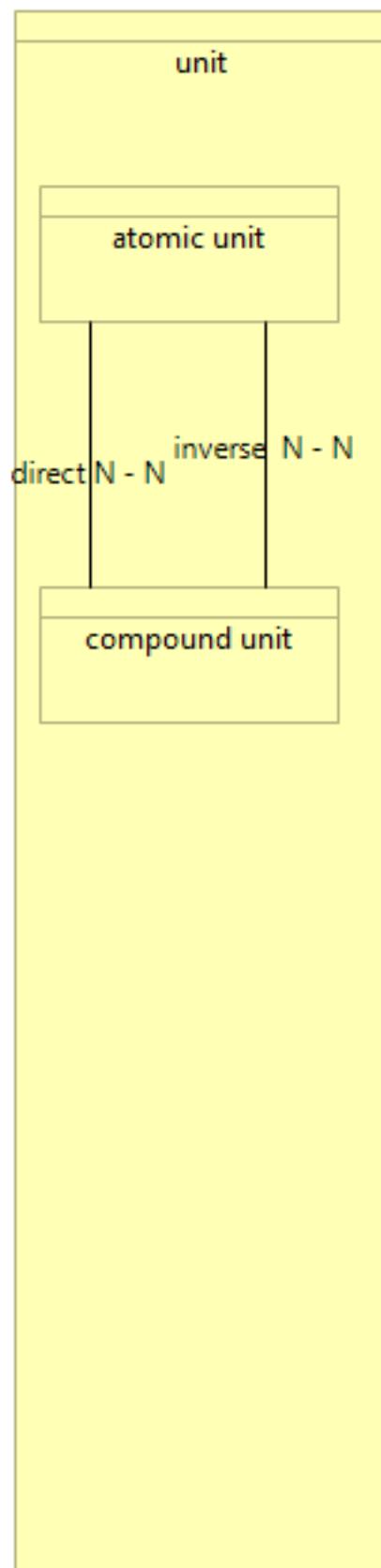
unit

conversion ratio

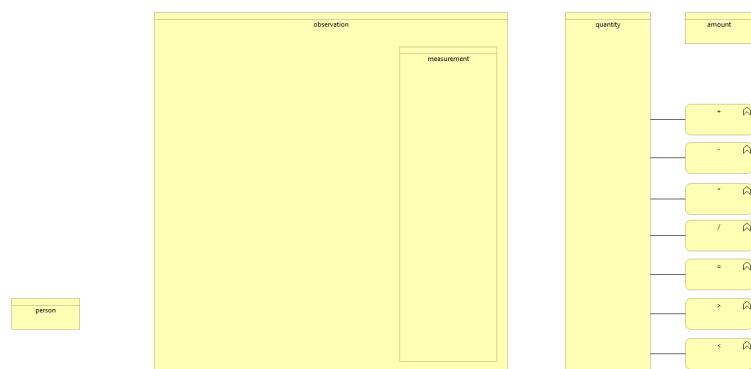
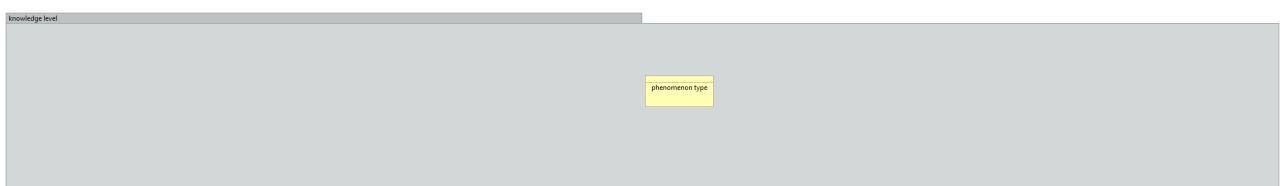
# OBSERVATIONS AND MEASUREMENTS



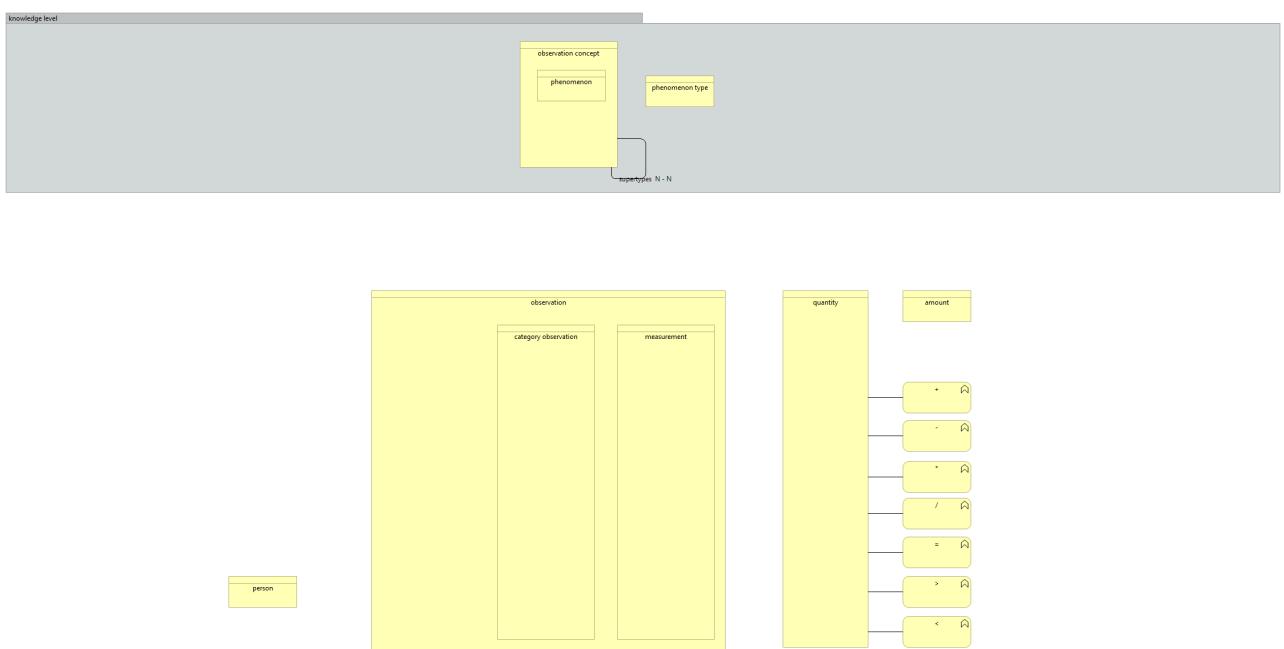
# COMPOUND UNITS



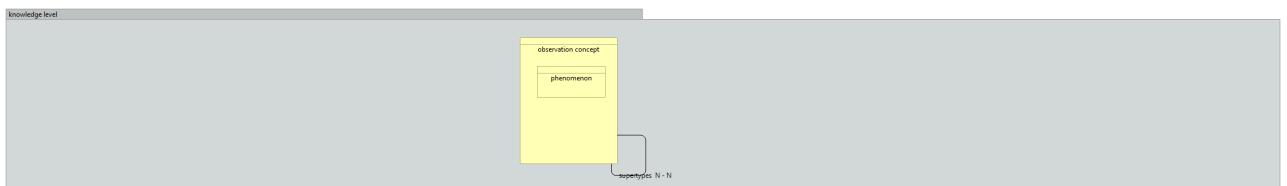
# MEASUREMENT



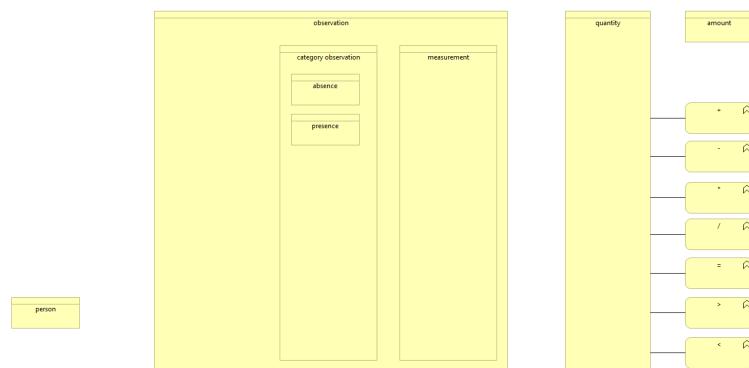
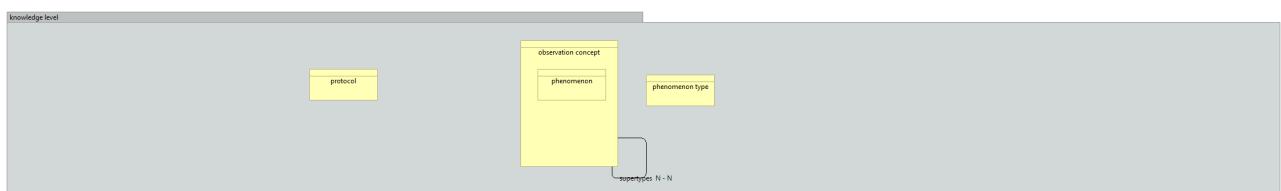
# OBSERVATION



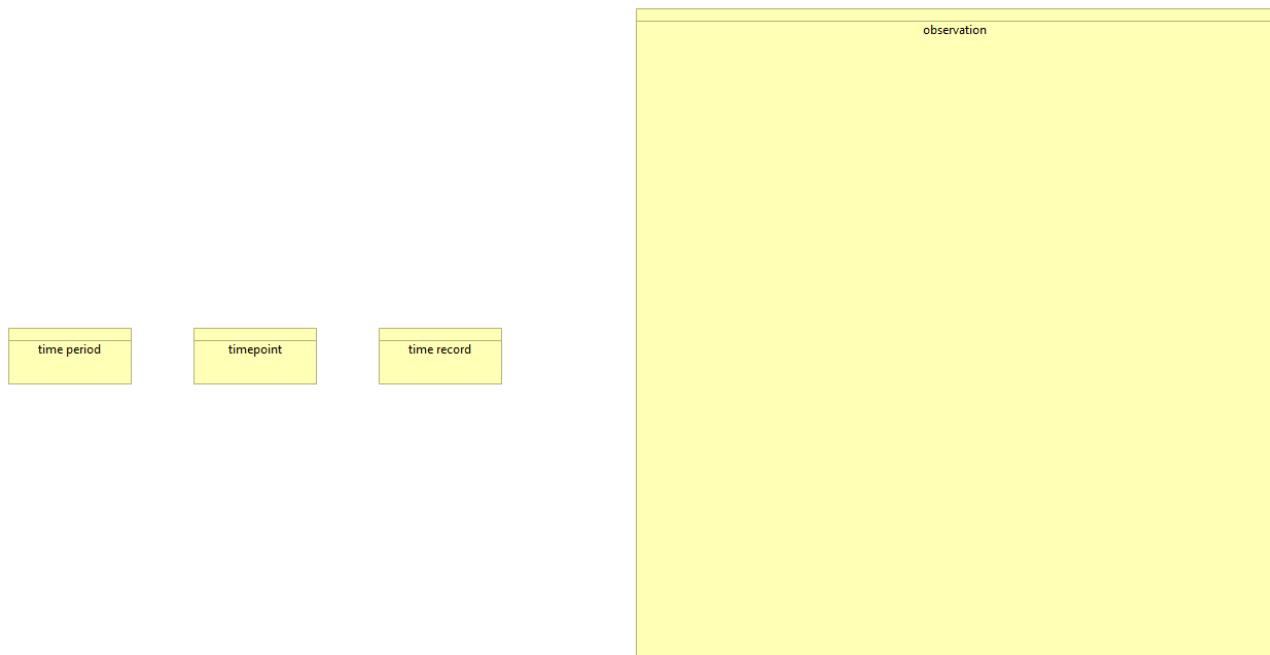
# SUBTYPING OBSERVATION CONCEPTS



# PROTOCOL



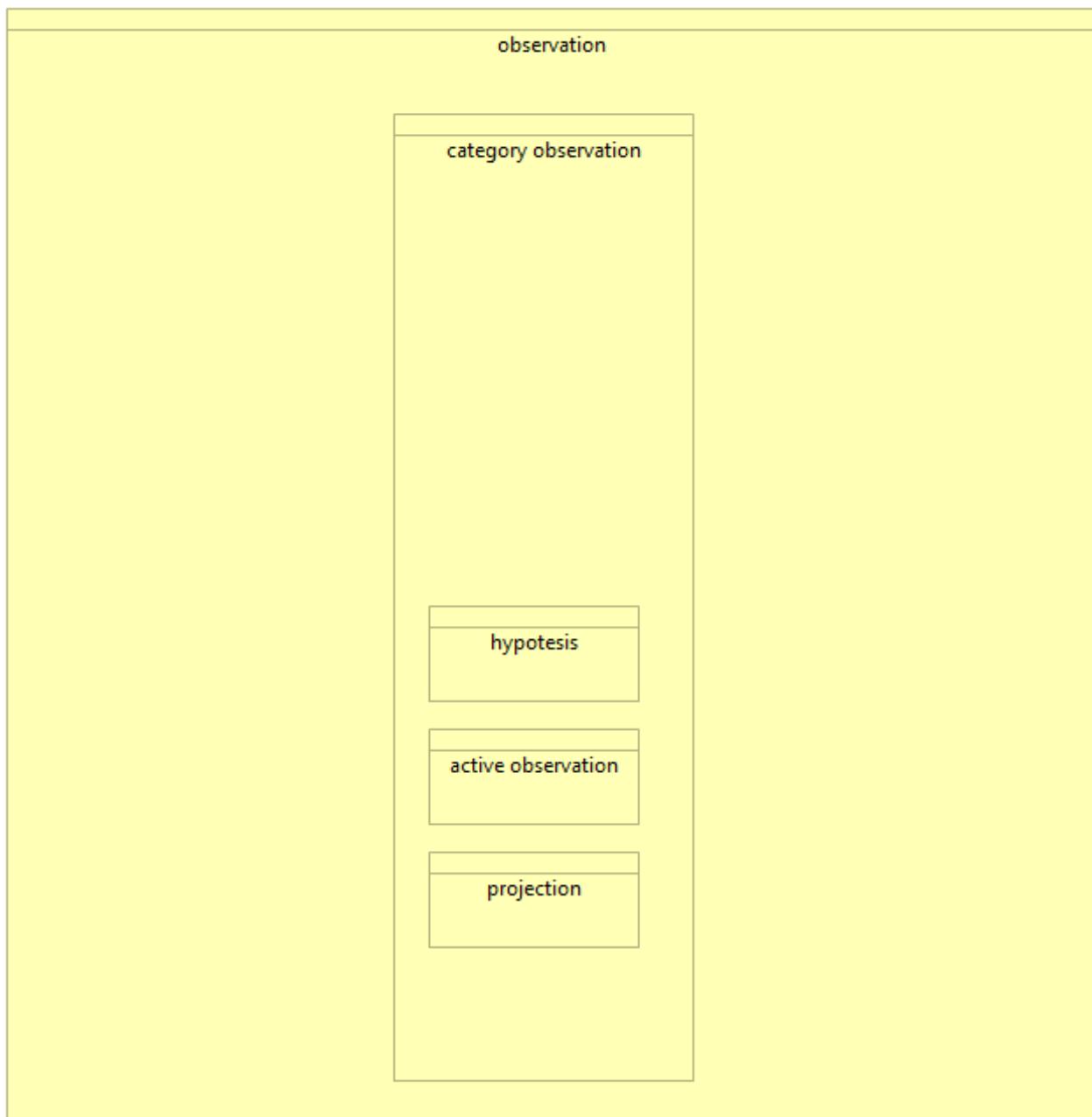
# DUAL TIME RECORD



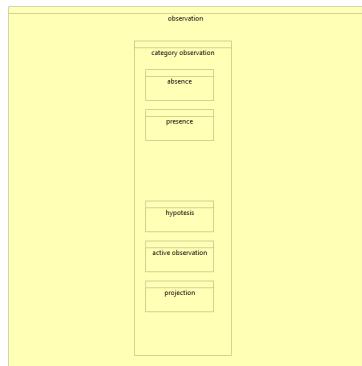
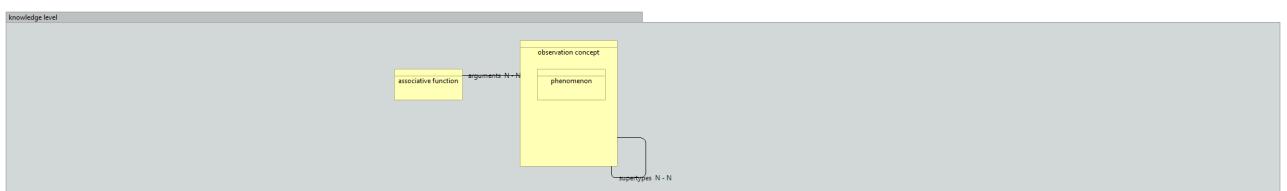
# REJECTED OBSERVATION

observation

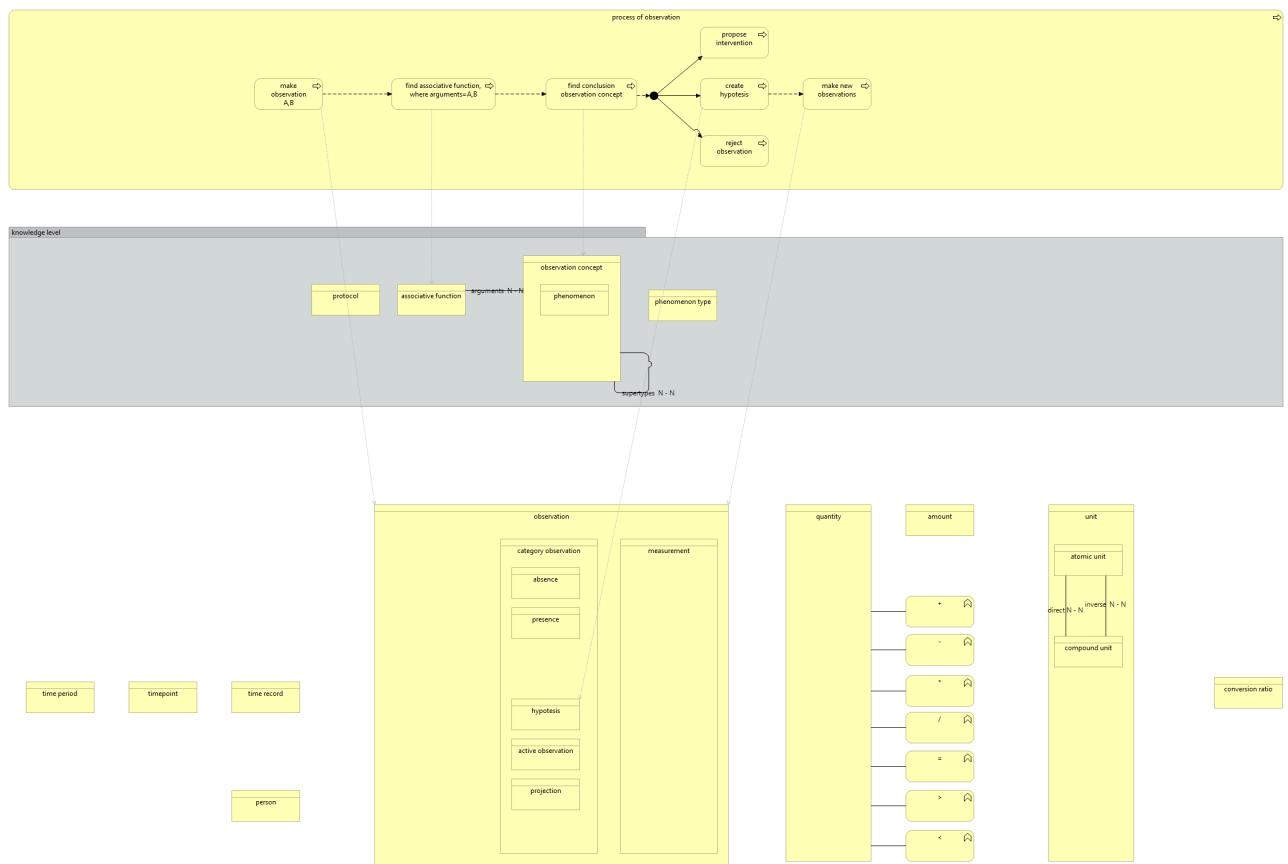
# ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION



# ASSOCIATED OBSERVATION



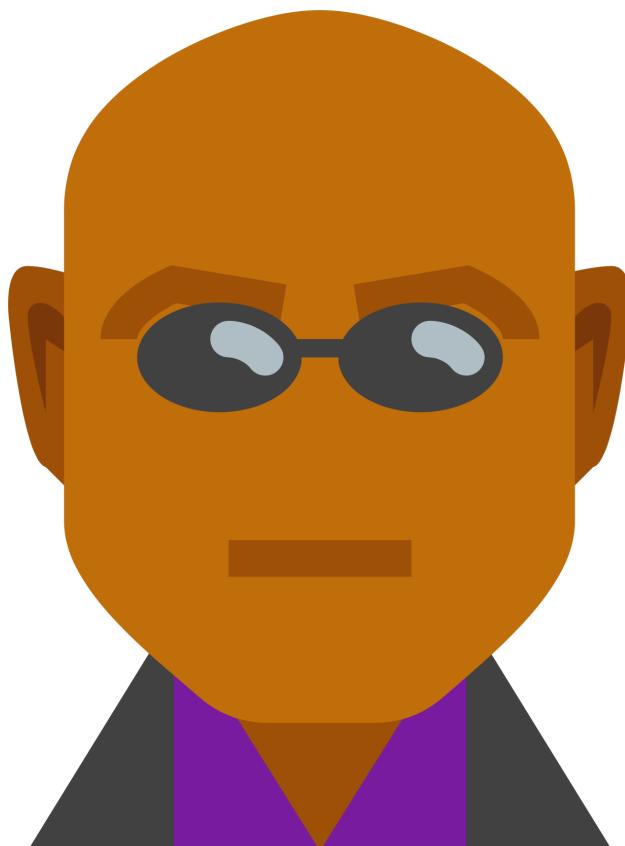
# PROCESS OF OBSERVATION



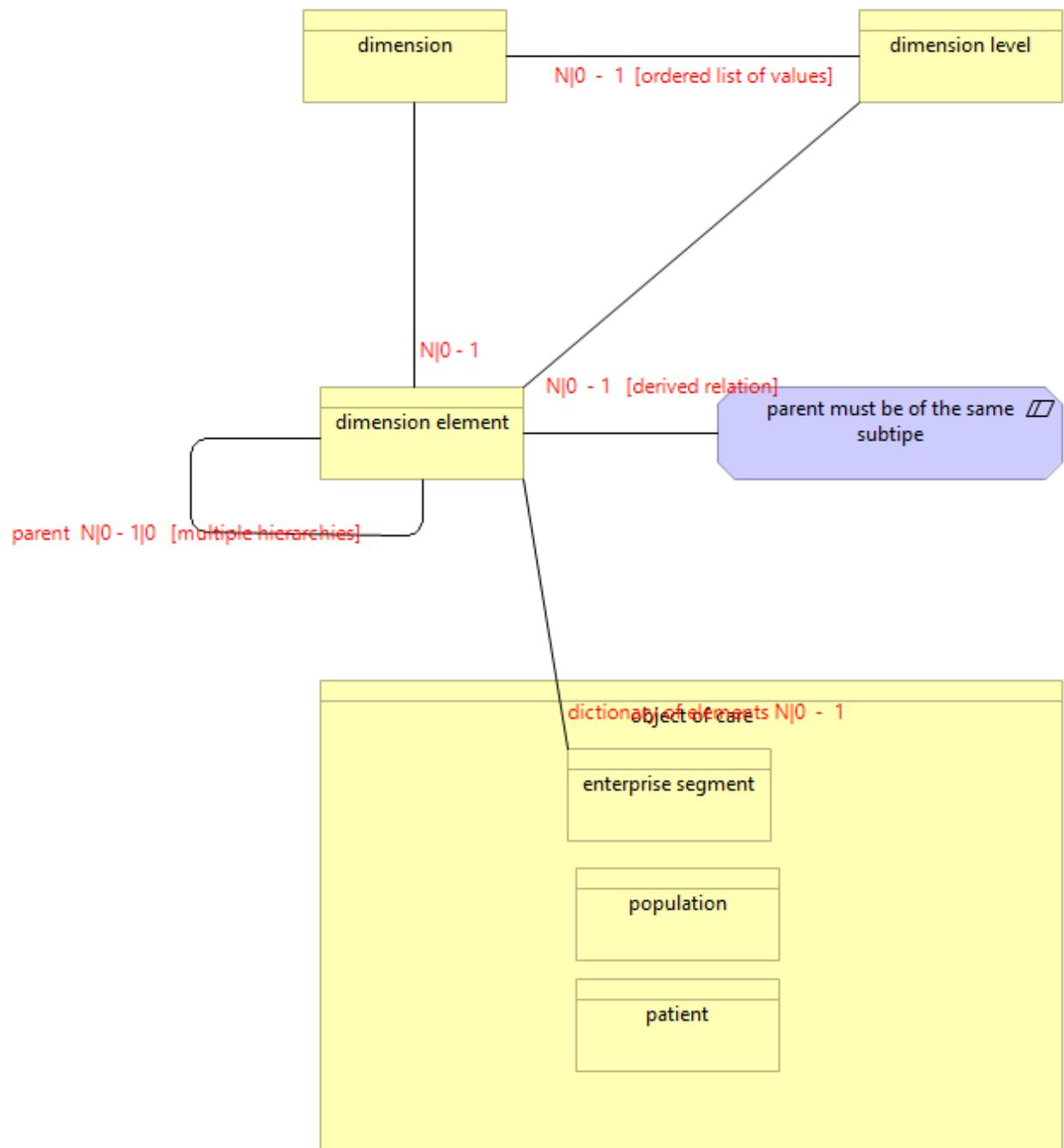
# OBSERVATIONS FOR CORPORATE FINANCE

ANALYSIS PATTERNS

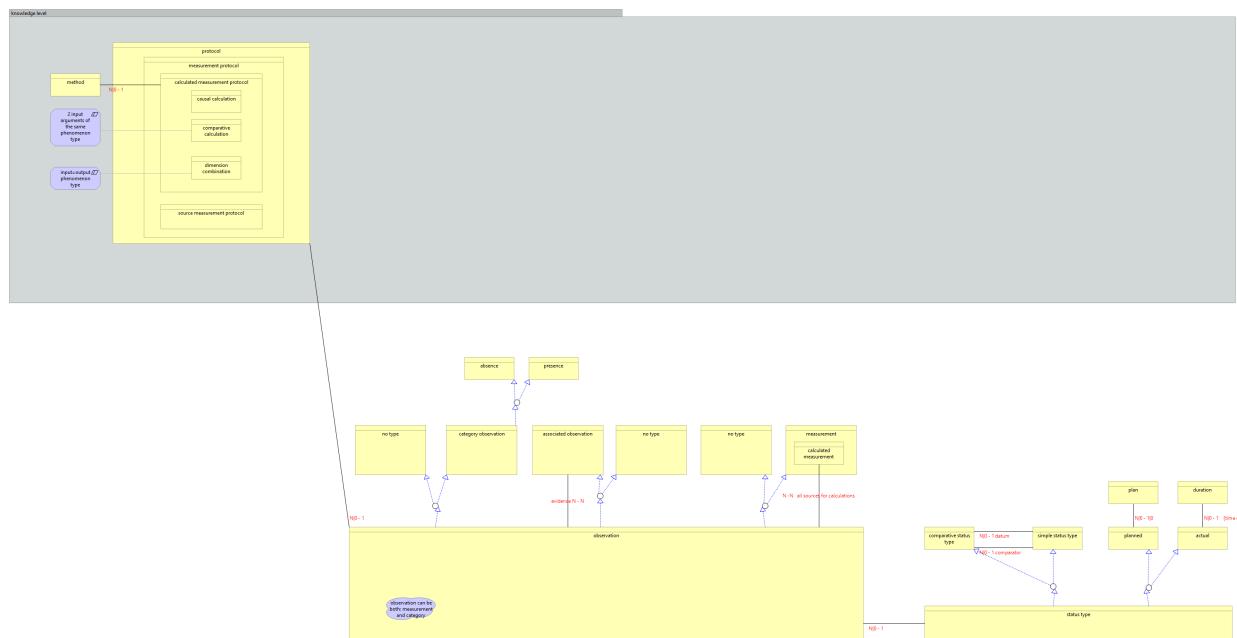
Martin Fowler



# ENTERPRISE SEGMENT



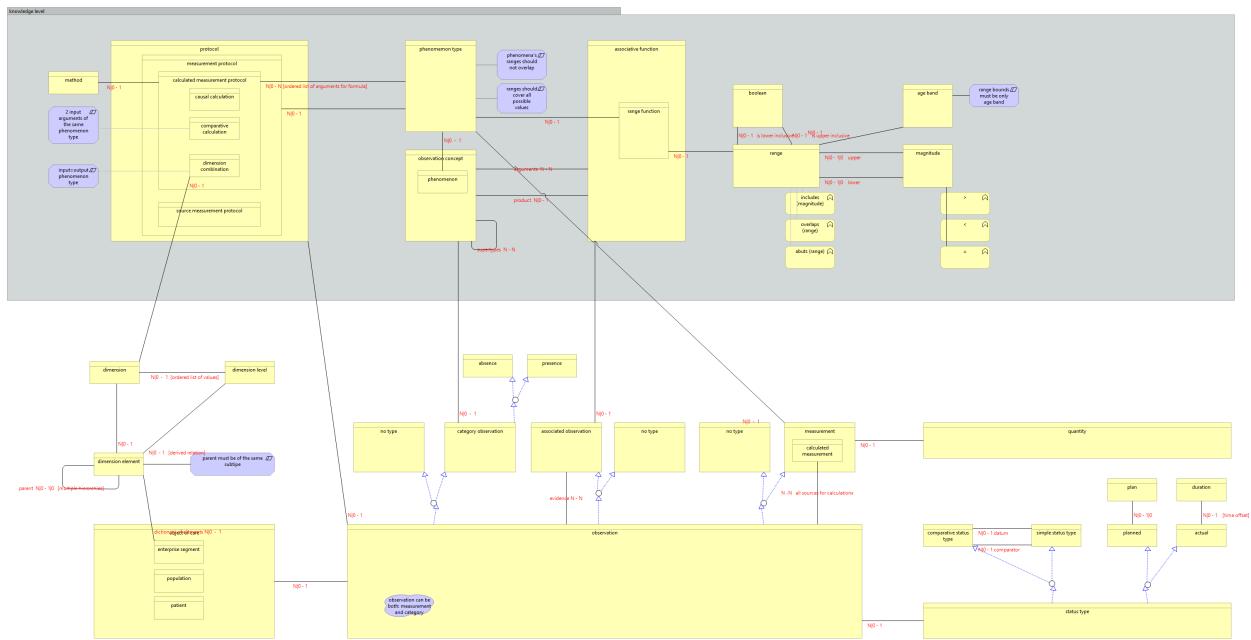
# MEASUREMENT PROTOCOL

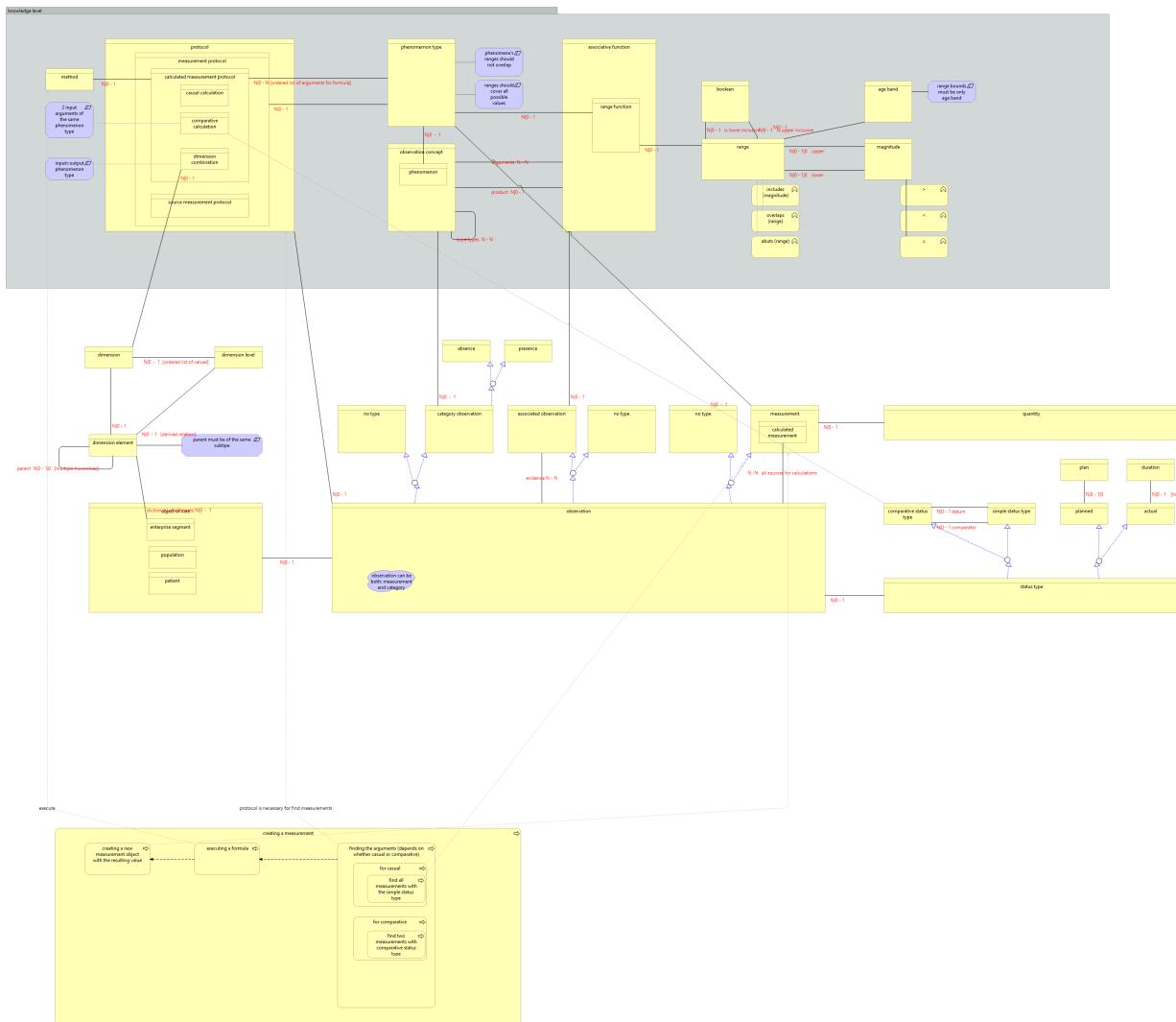


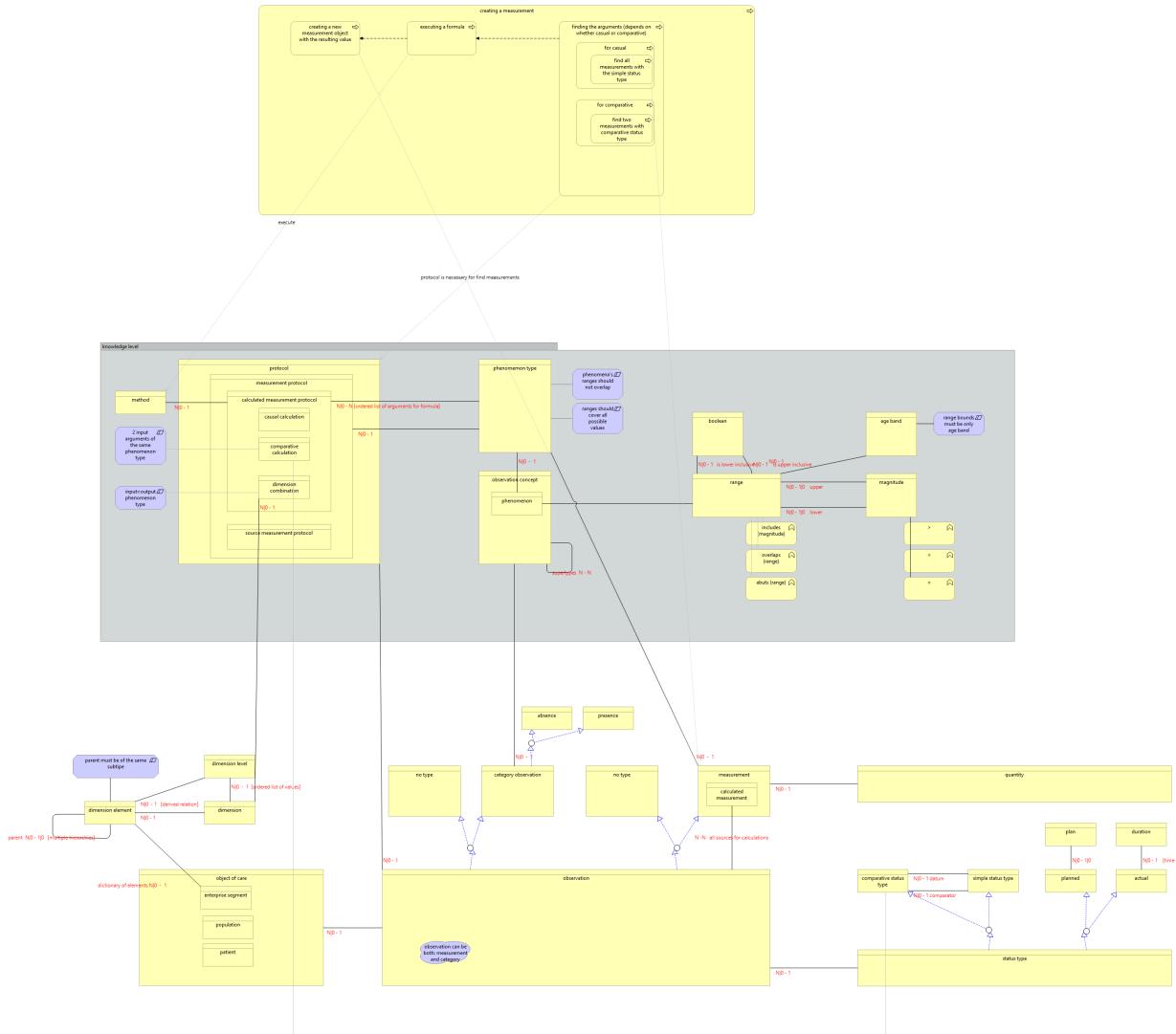
# RANGE



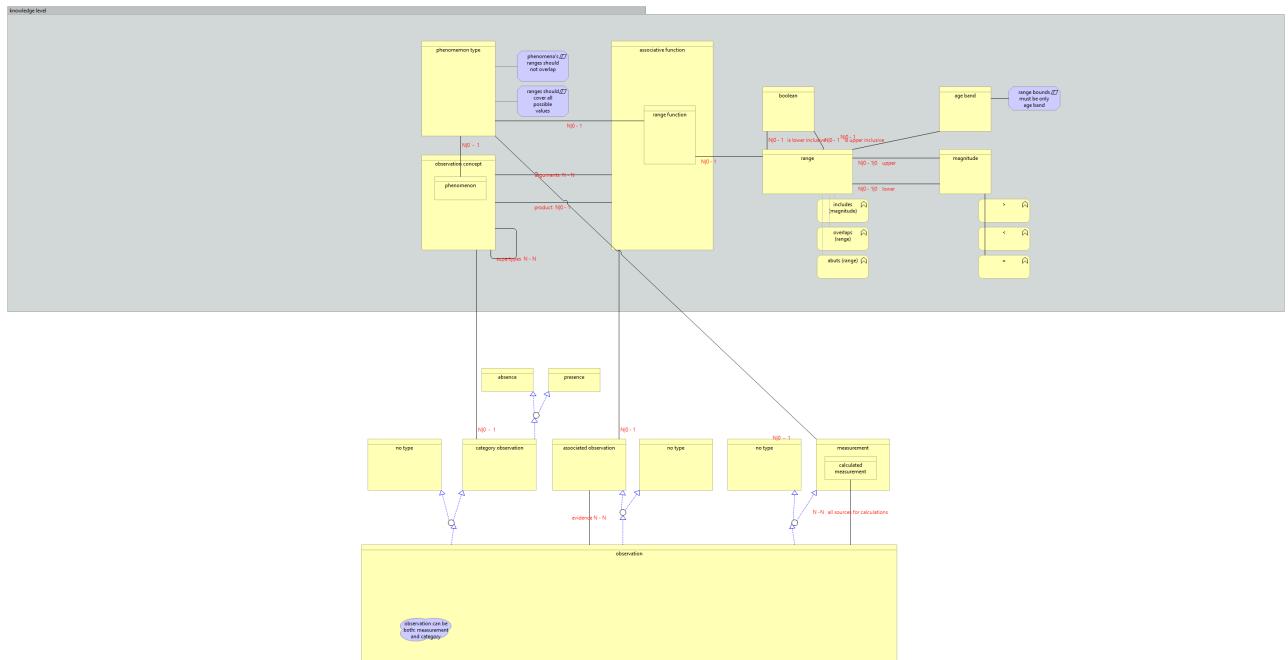
# OBSERVATIONS FOR CORPORATE FINANCE







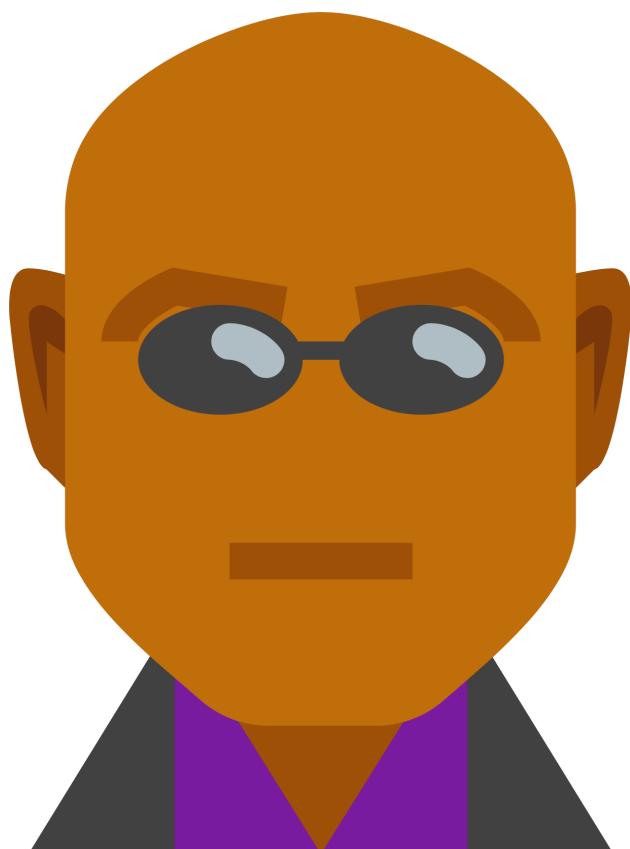
# PHENOMENON WITH RANGE



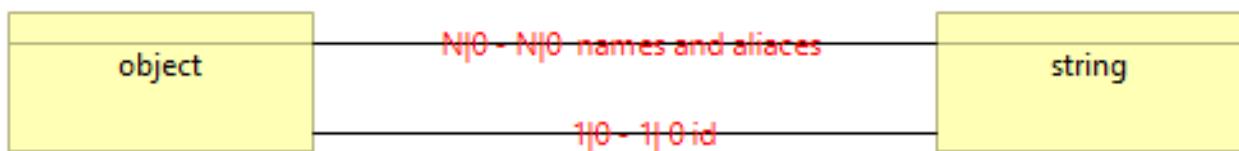
# REFERRING TO OBJECTS

ANALYSIS PATTERNS

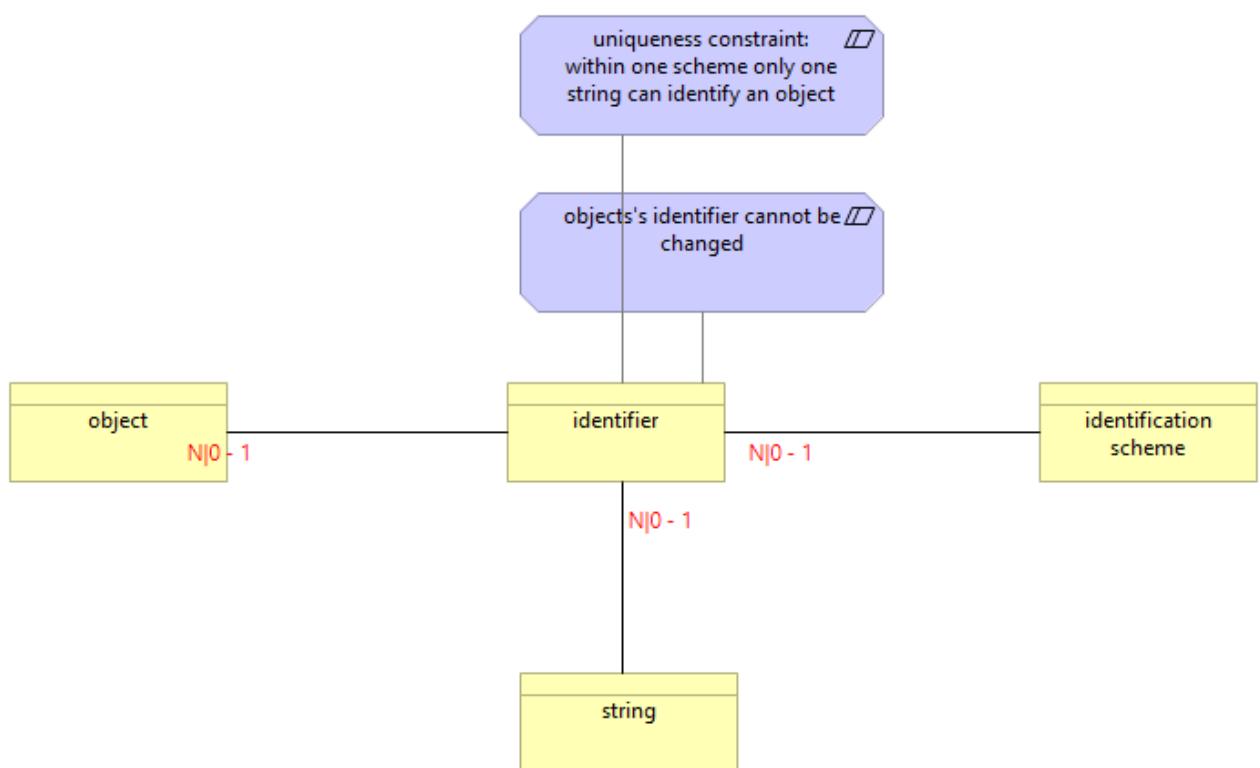
Martin Fowler



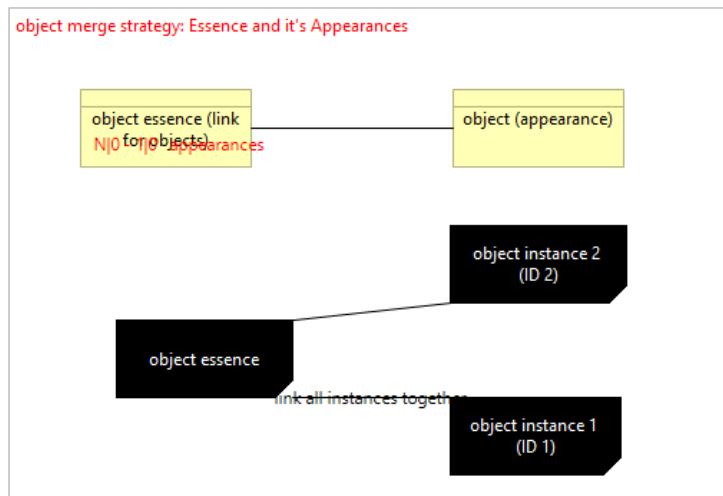
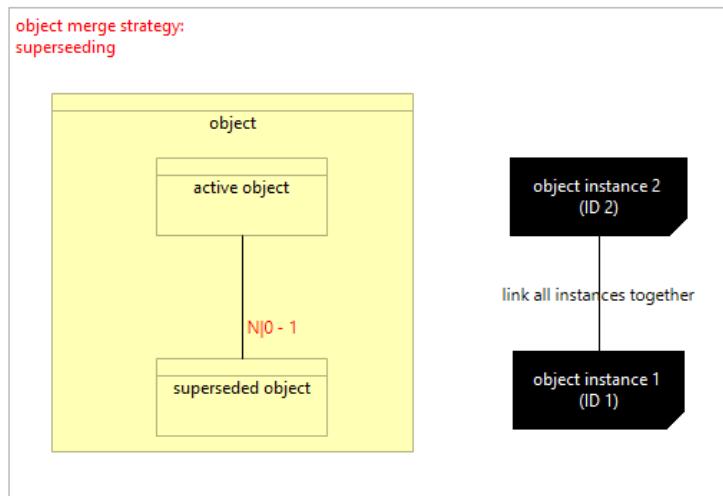
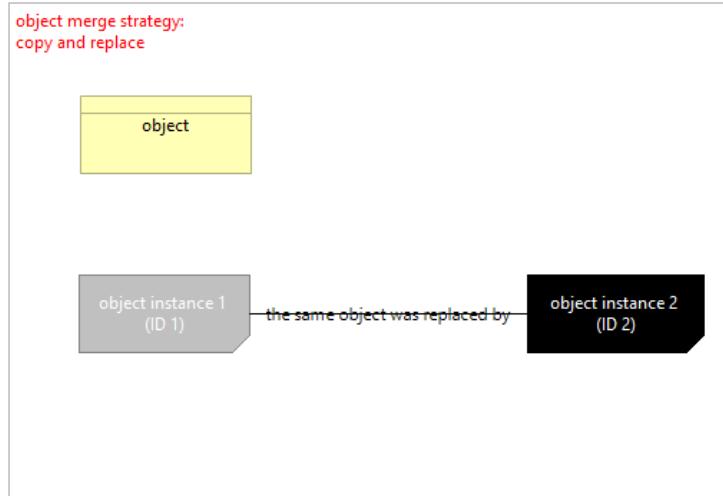
# NAME



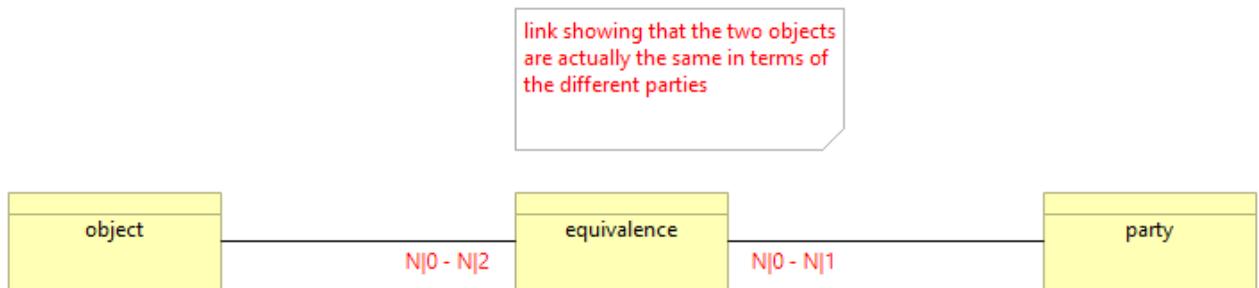
# IDENTIFICATION SCHEME



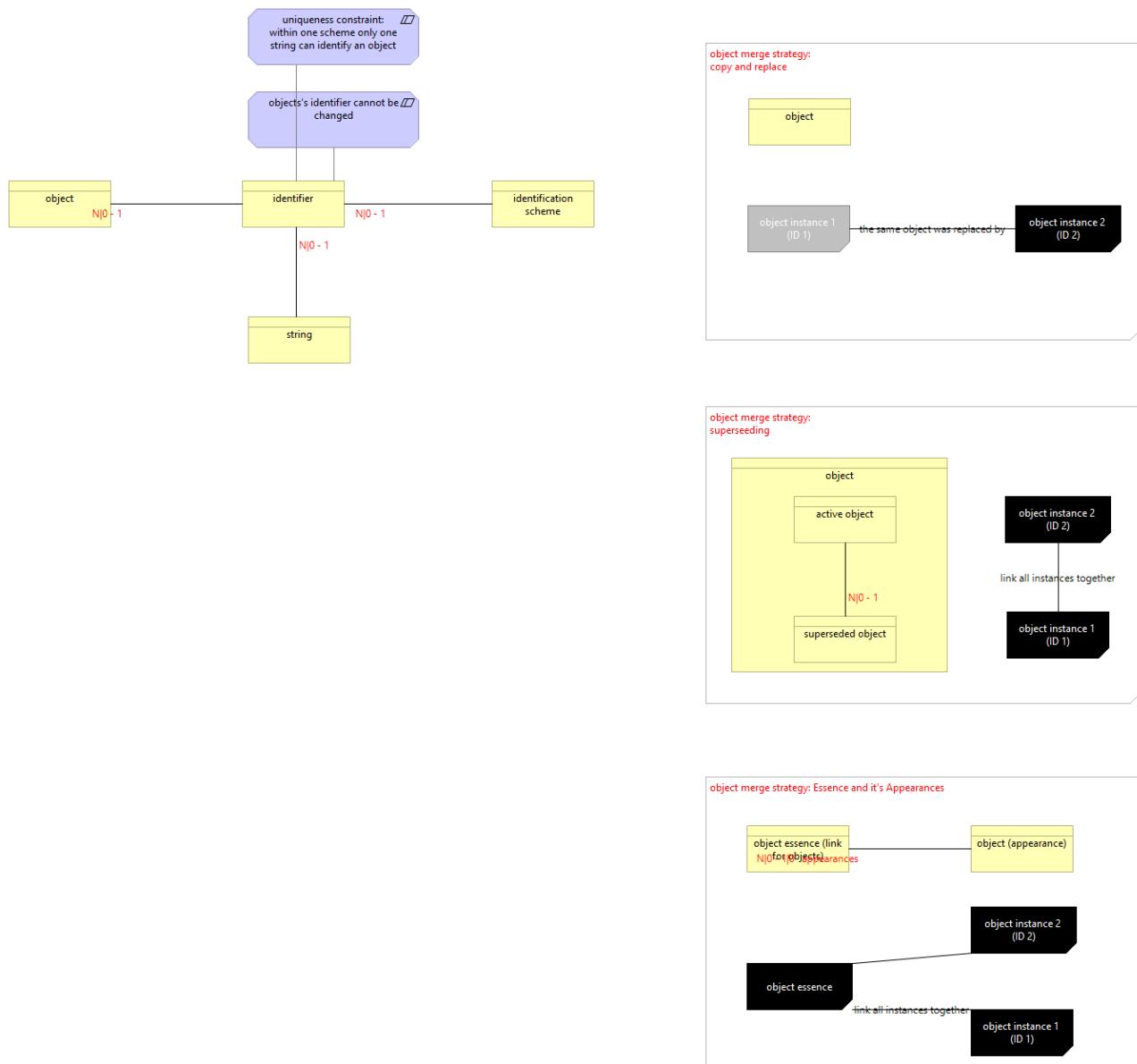
# OBJECT MERGE



# OBJECT EQUIVALENCE



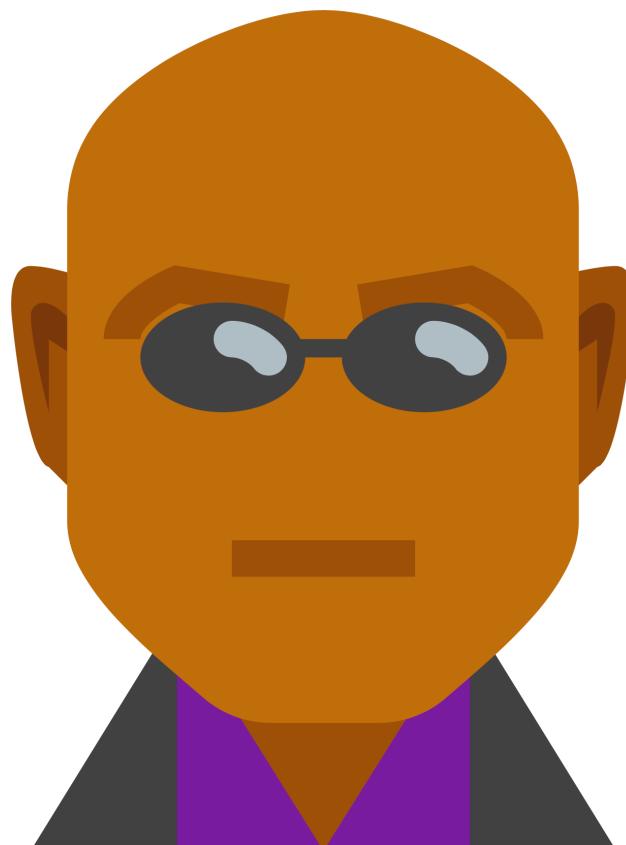
# REFERRING TO OBJECTS



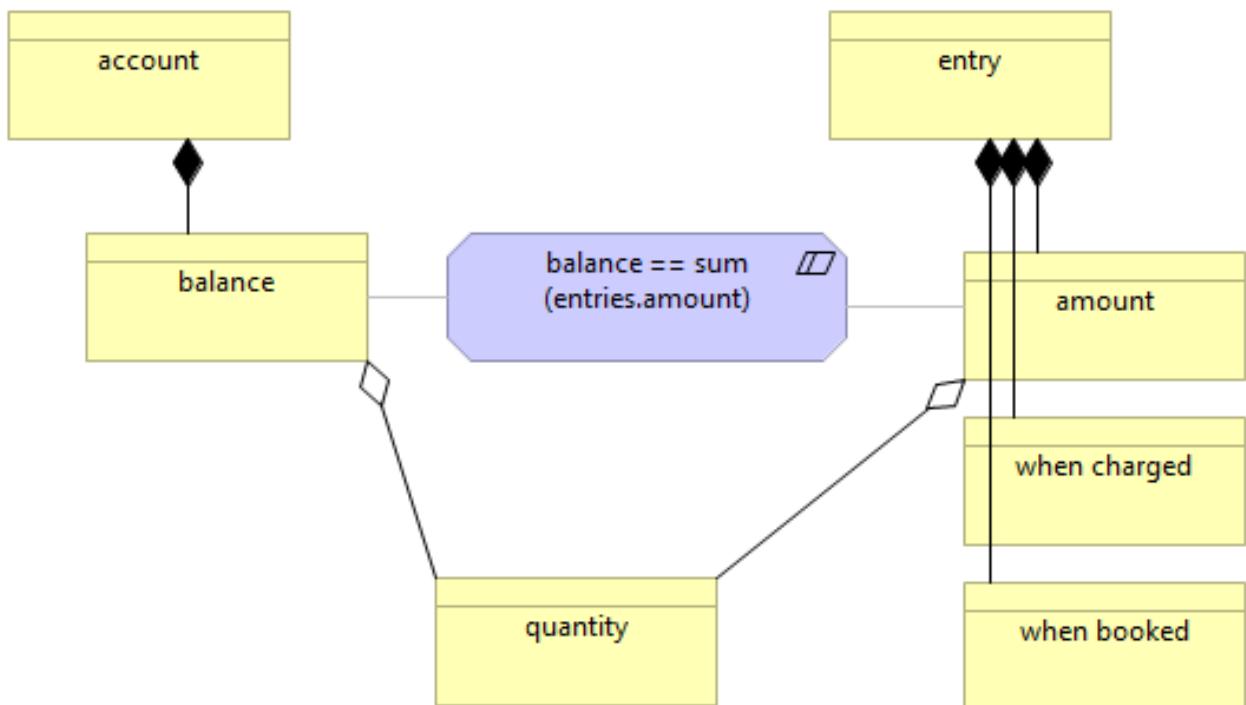
# INVENTORY AND ACCOUNTING

ANALYSIS PATTERNS

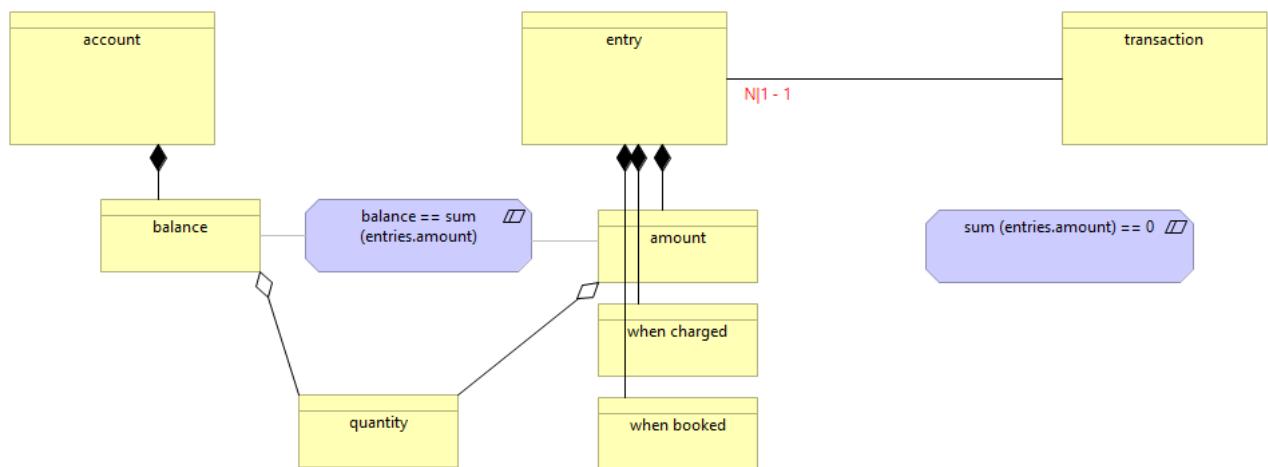
Martin Fowler



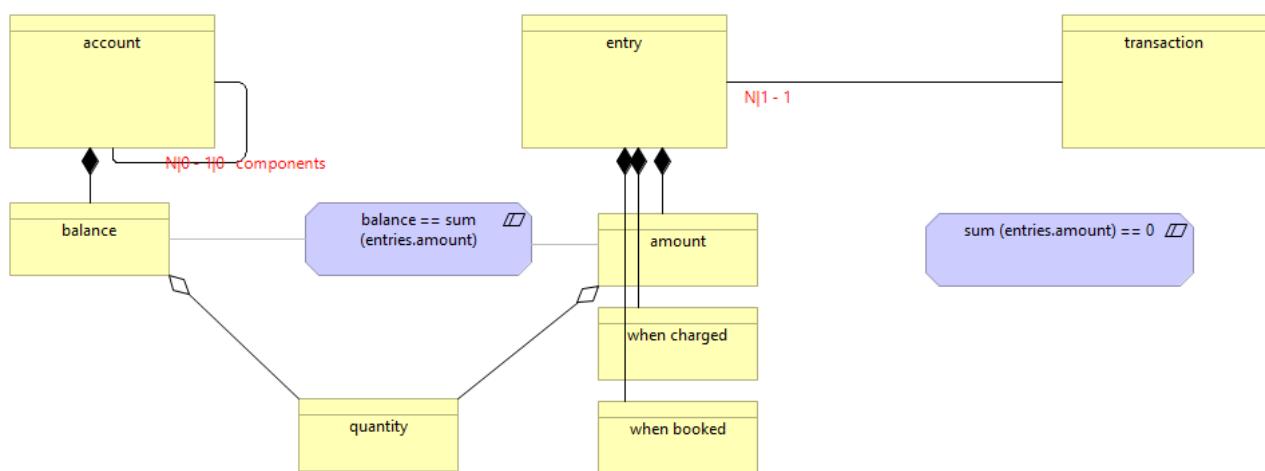
# ACCOUNT



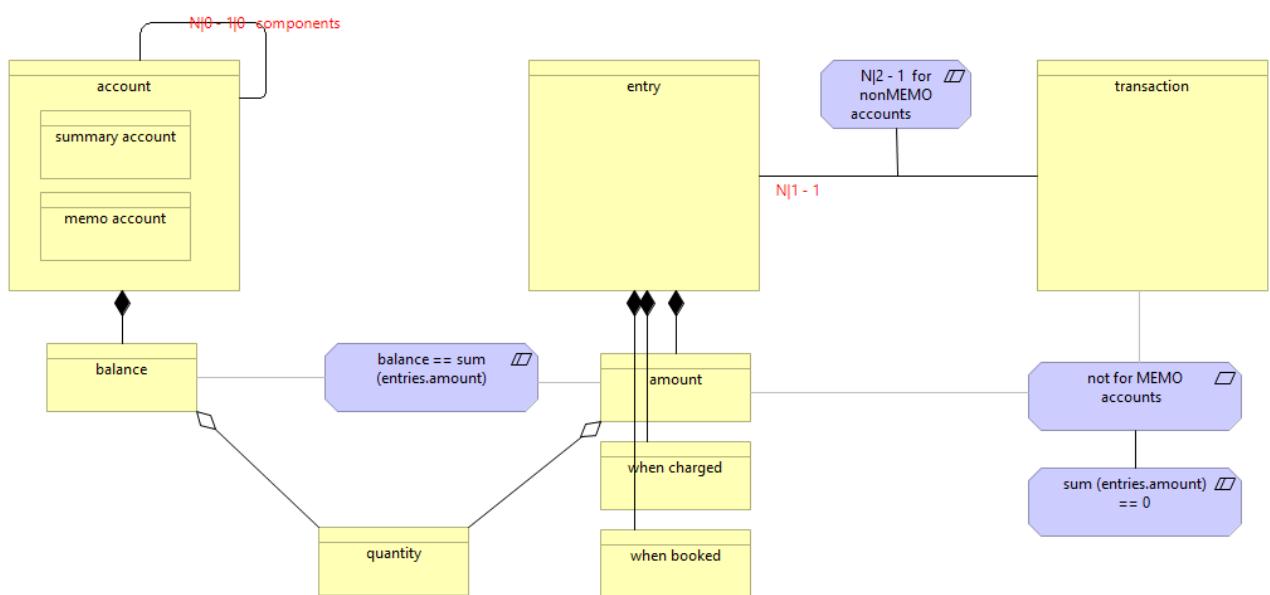
# TRANSACTIONS



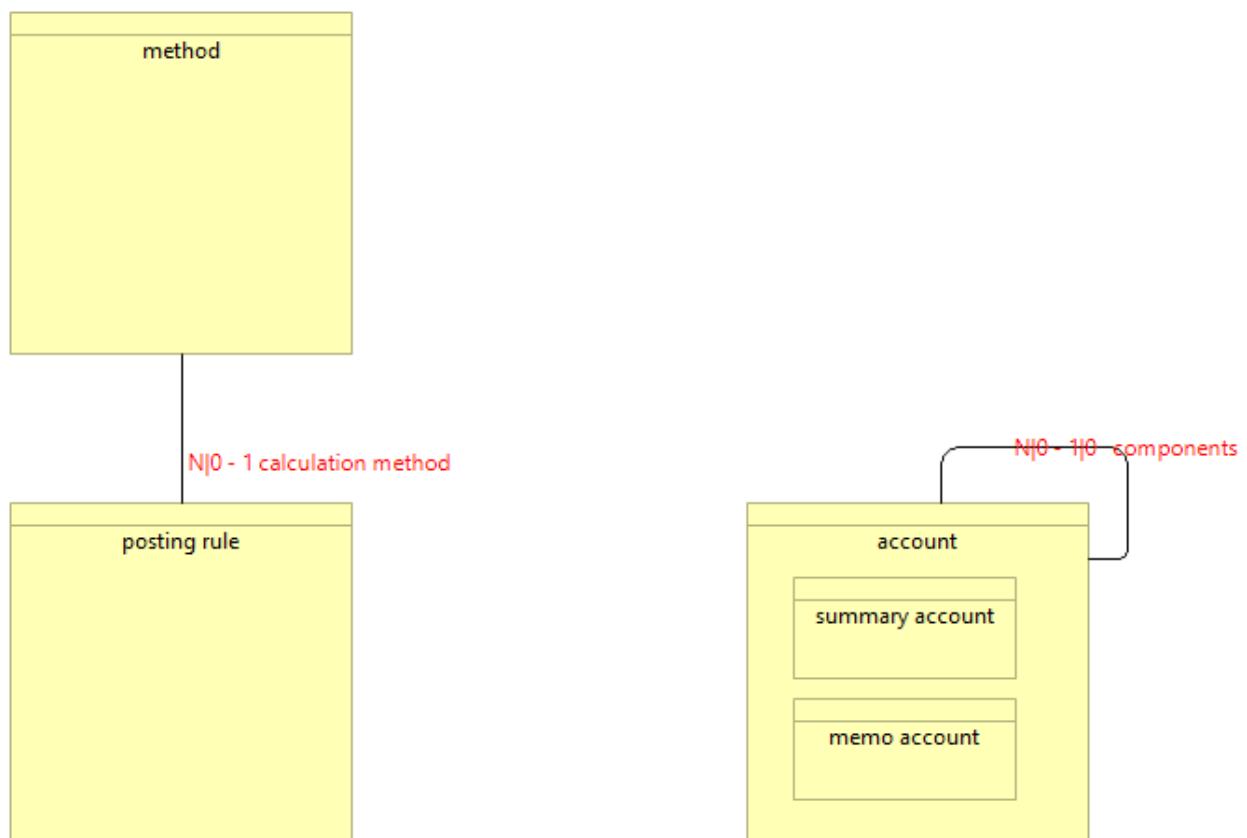
# SUMMARY ACCOUNT



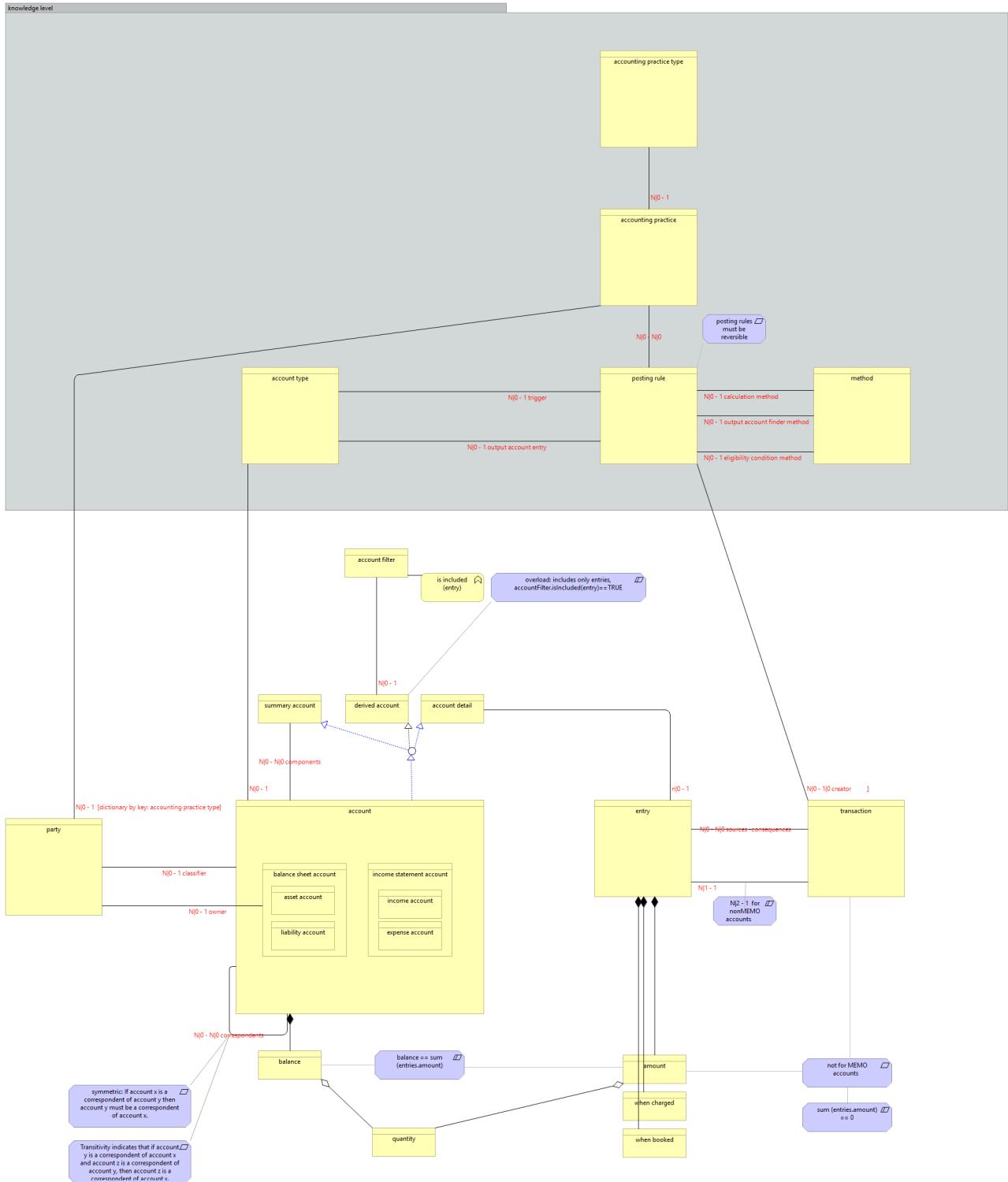
# MEMO ACCOUNT



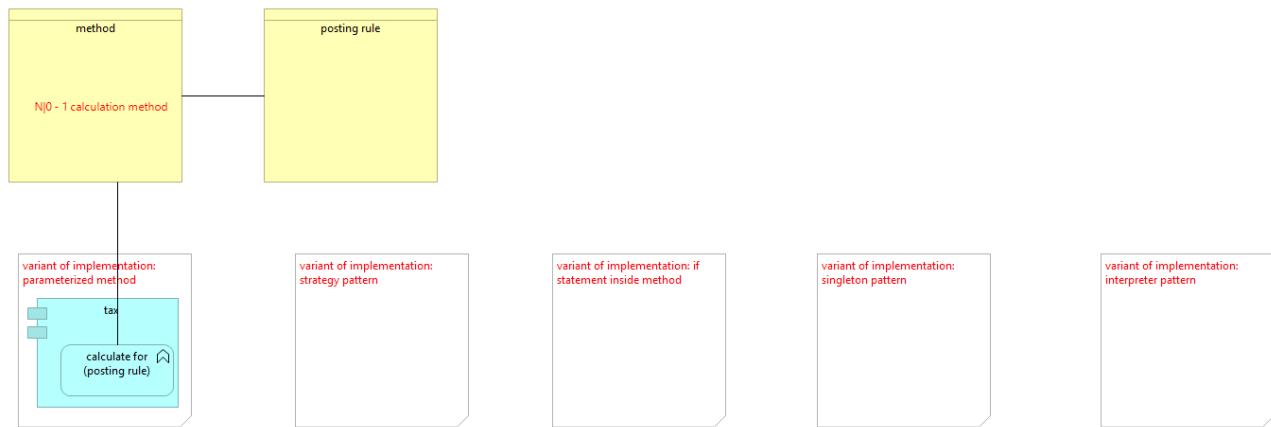
# POSTING RULES



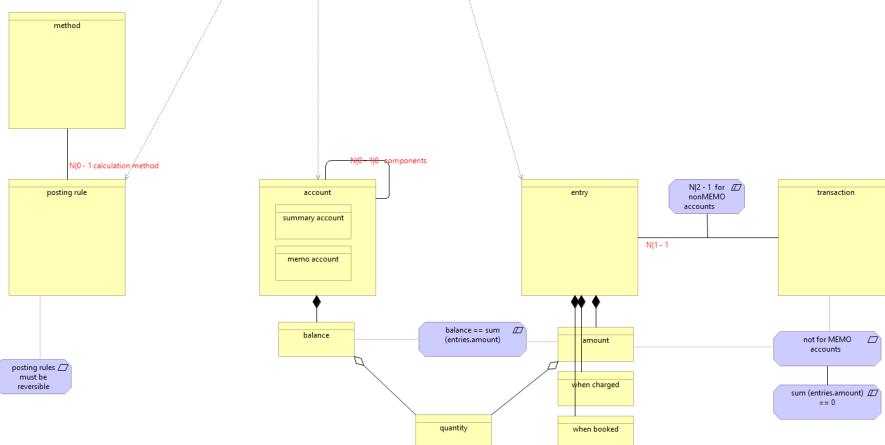
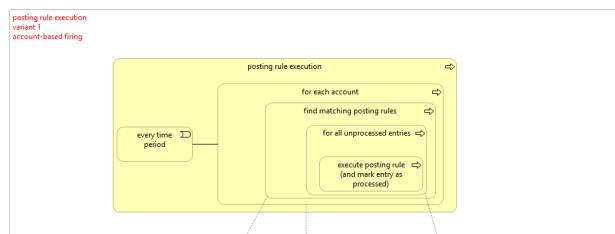
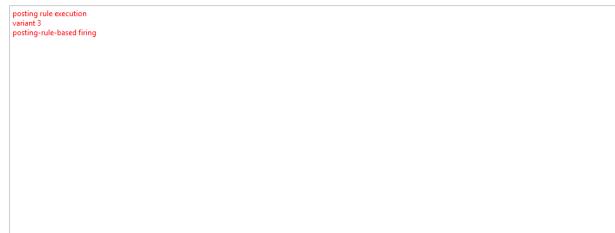
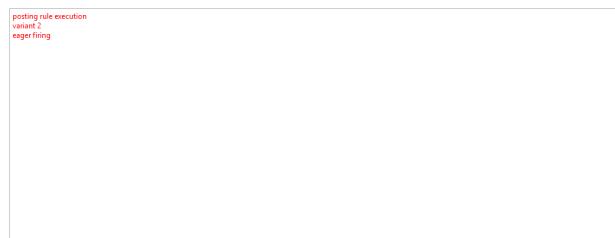
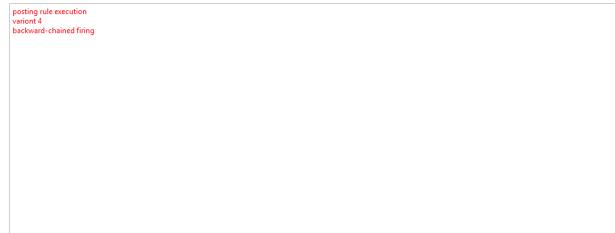
# INVENTORY AND ACCOUNTING



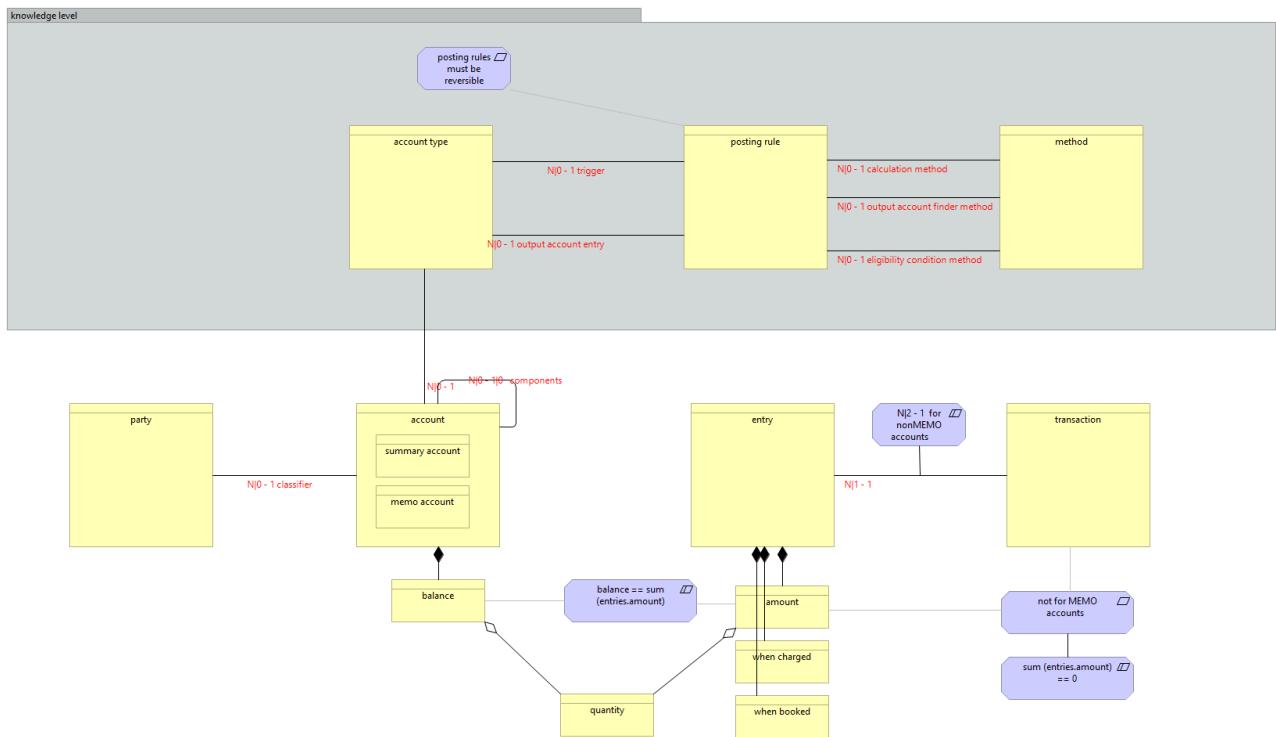
# INDIVIDUAL INSTANCE METHOD



# POSTING RULE EXECUTION



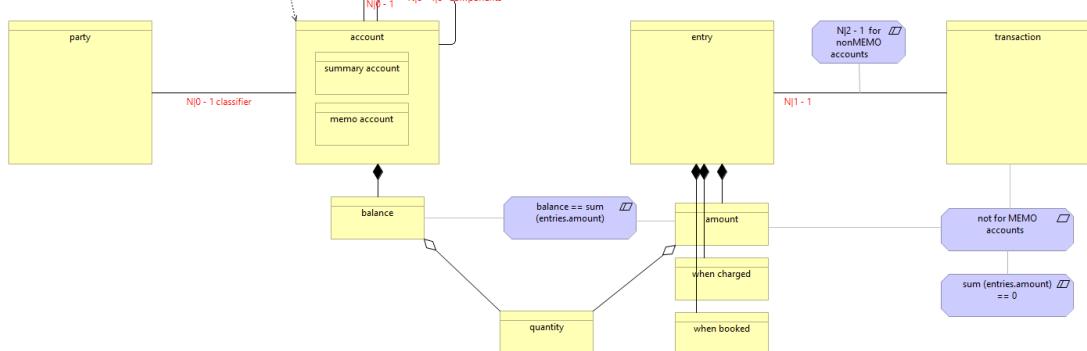
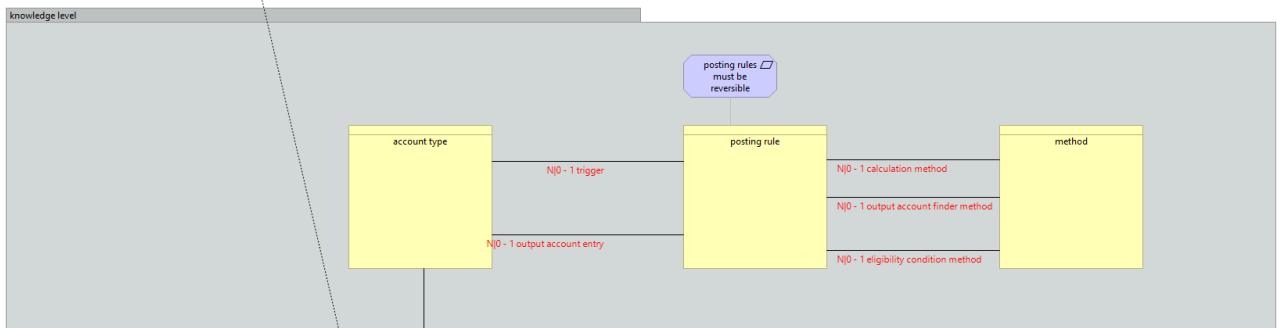
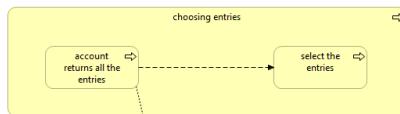
# **POSTING RULES FOR MANY ACCOUNTS**



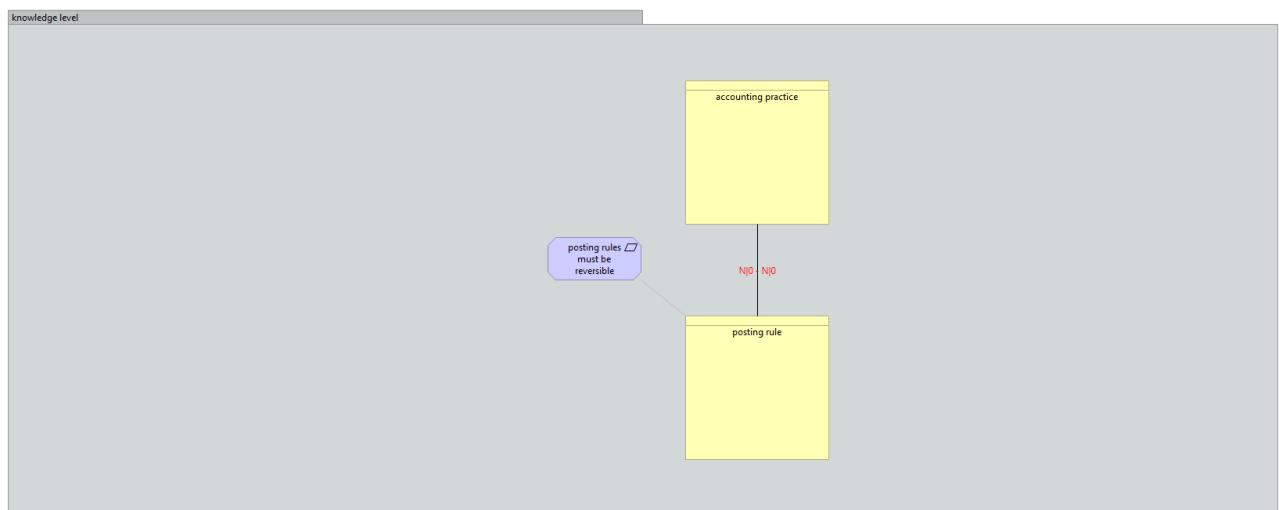
# CHOOSING ENTRIES

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

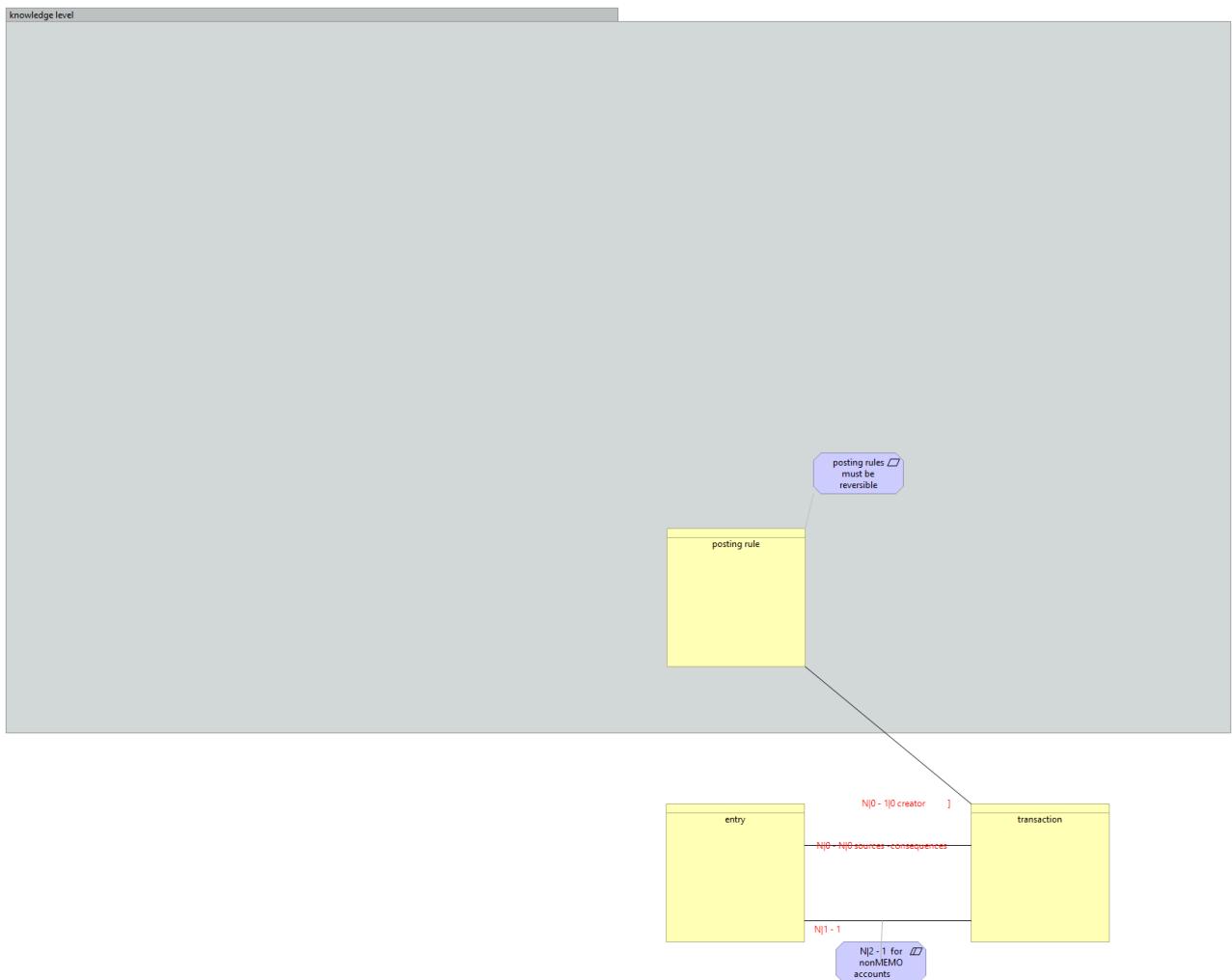
selection of entries for the application of the posing rules:  
variant 1  
The account returns all the entries, and the client selects the entries it needs.



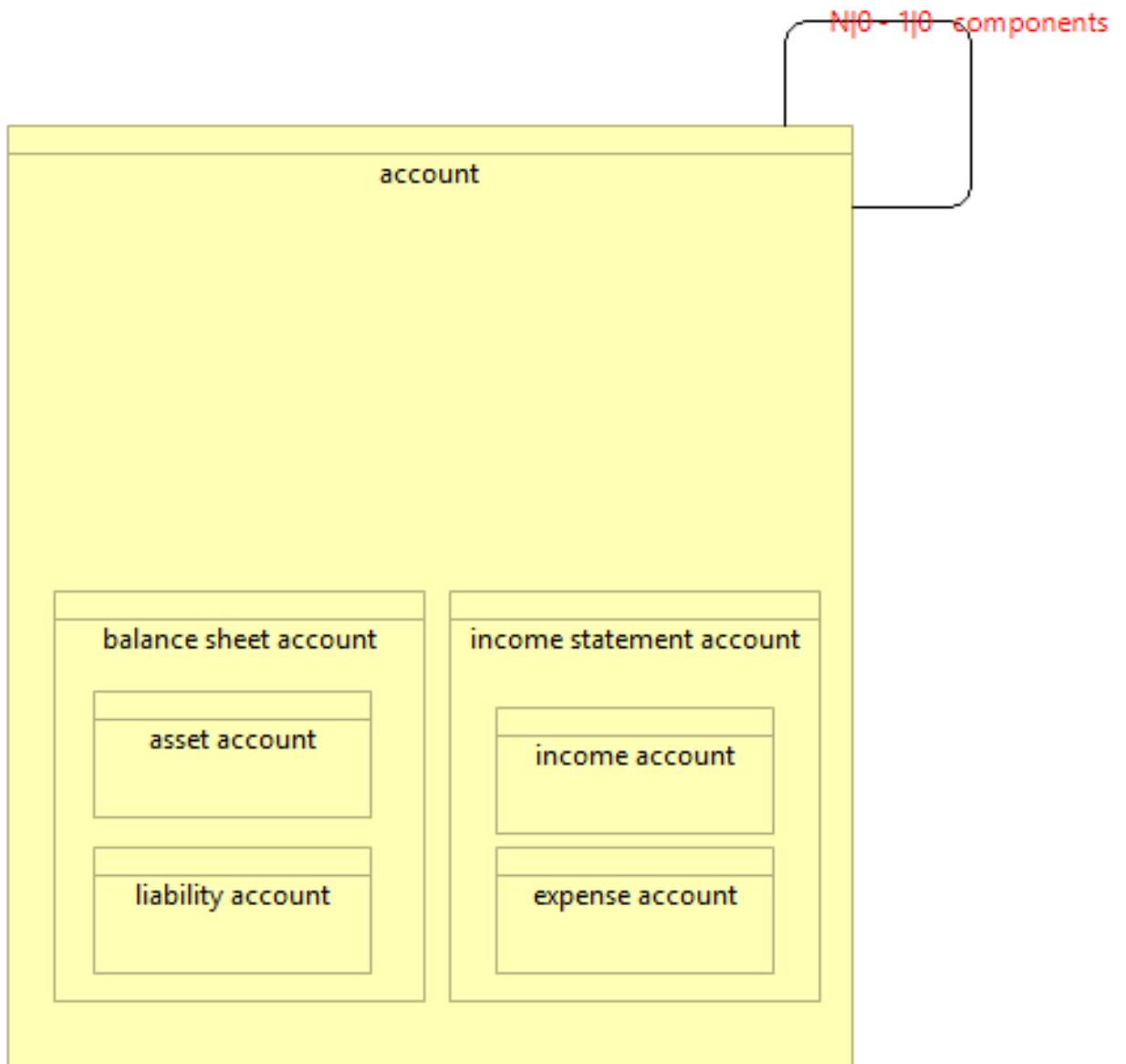
# ACCOUNTING PRACTICE



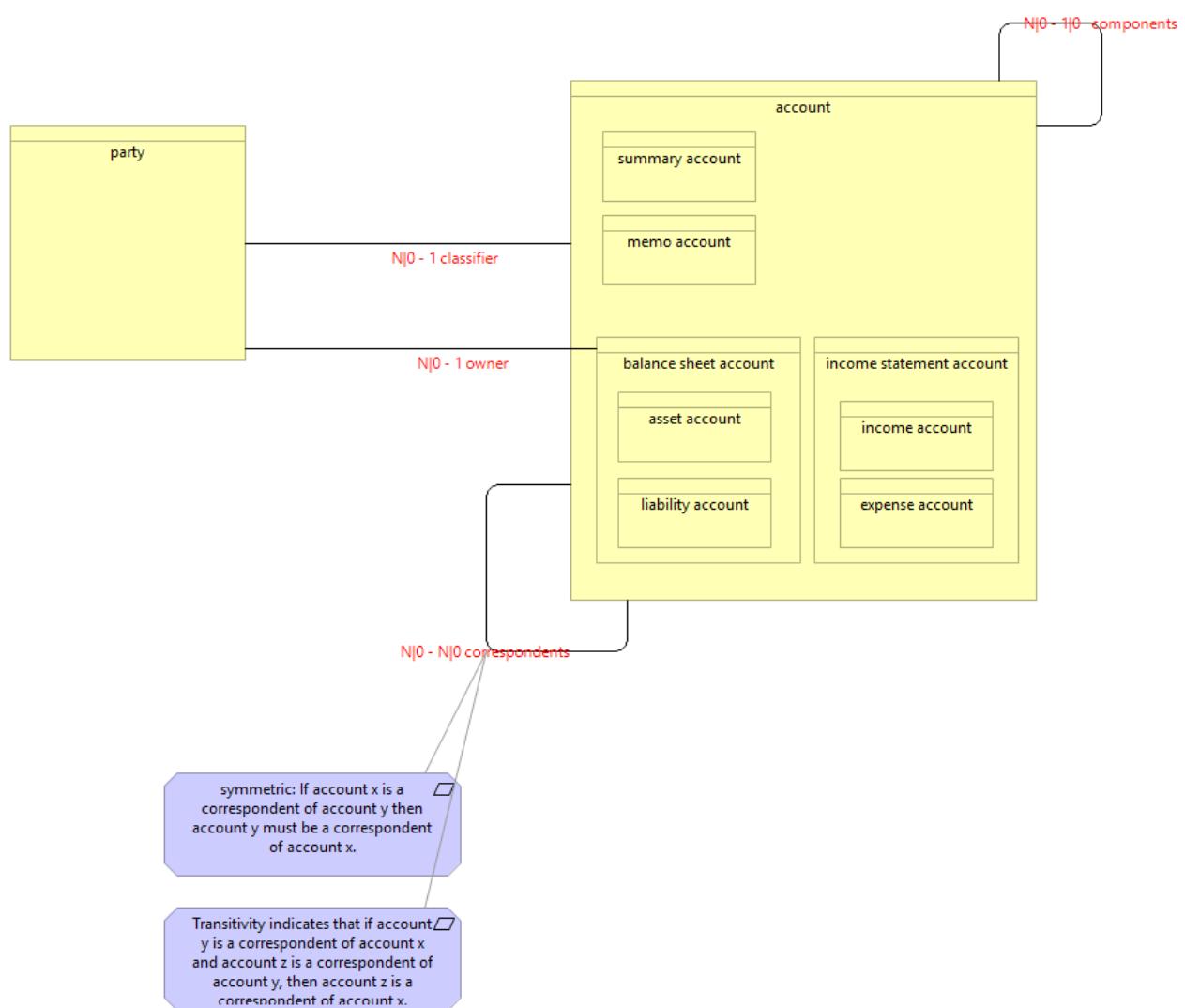
# SOURCES OF AN ENTRY



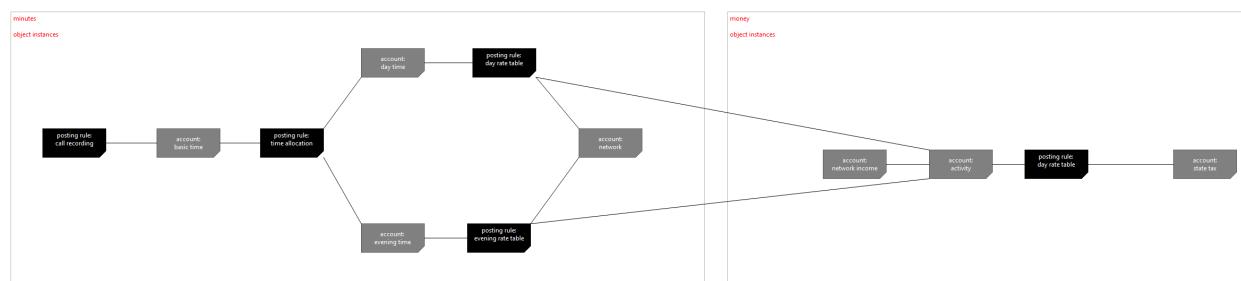
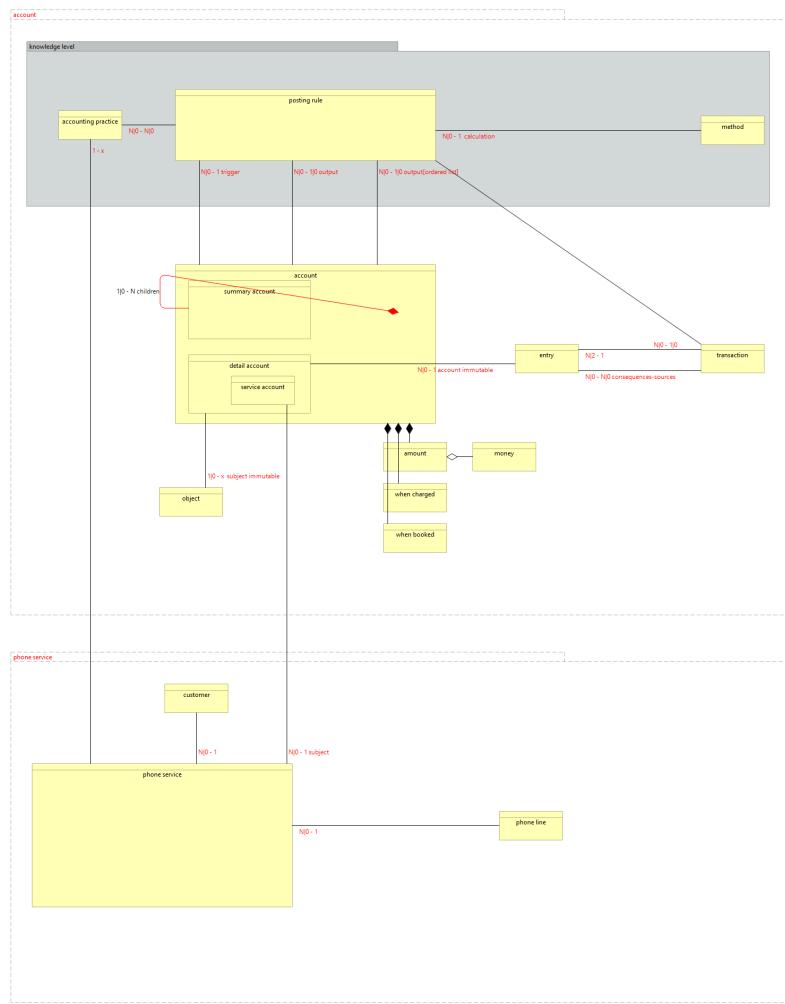
# BALANCE SHEET AND INCOME STATEMENT



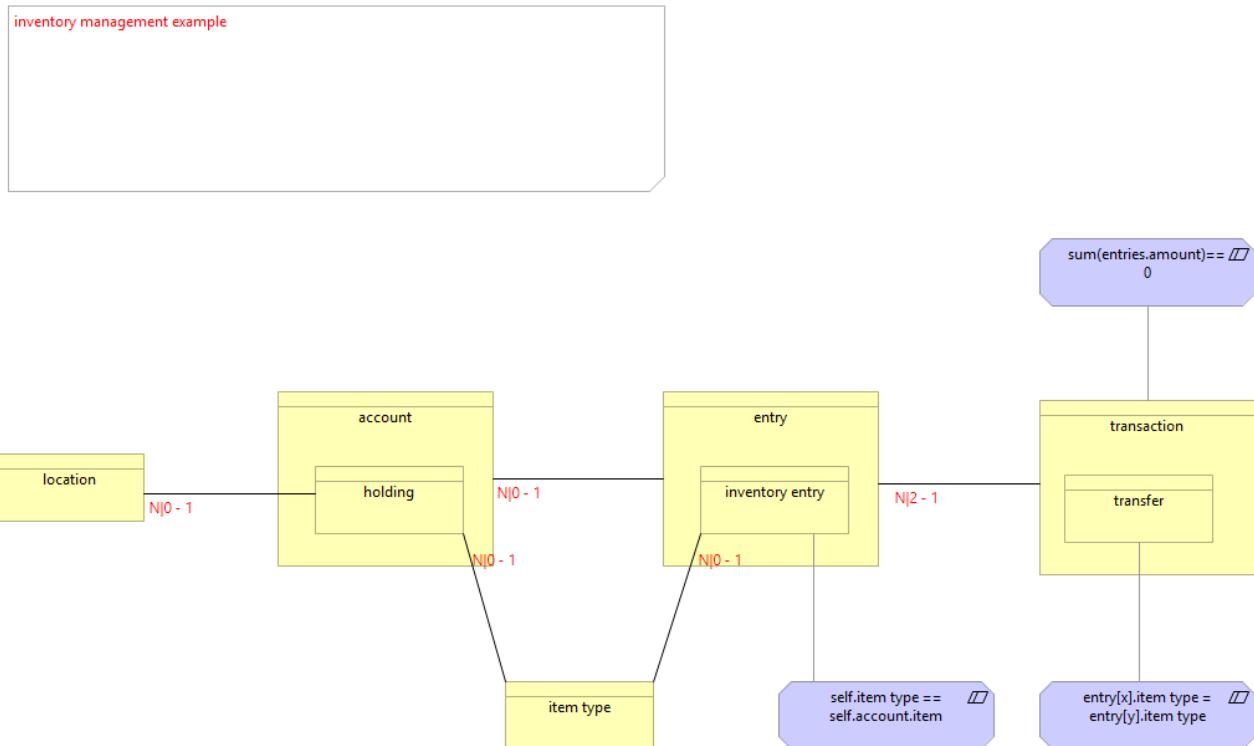
# CORRESPONDING ACCOUNT



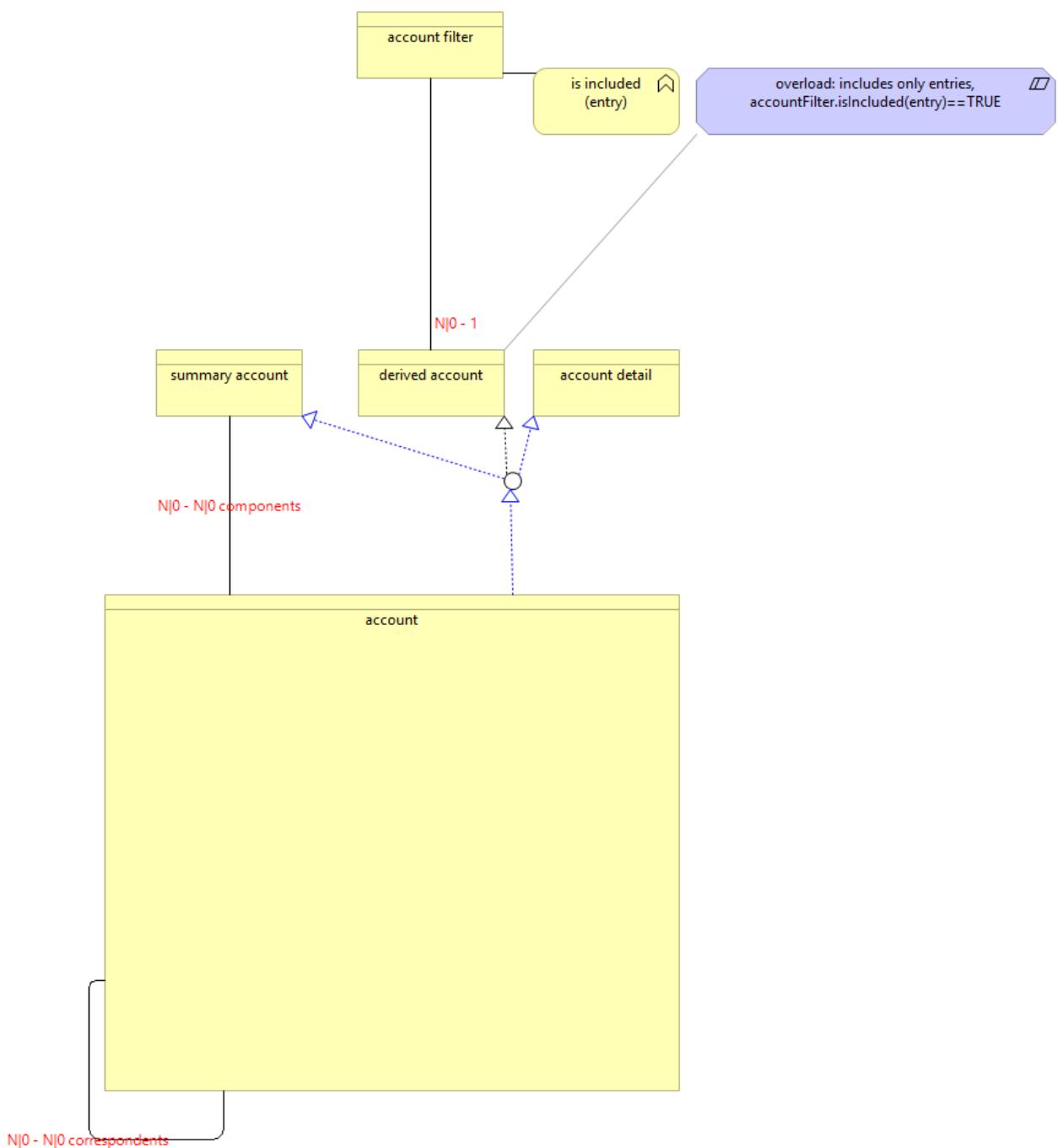
# SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)



# SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)



# BOOKING ENTRIES TO MULTIPLE ACCOUNTS

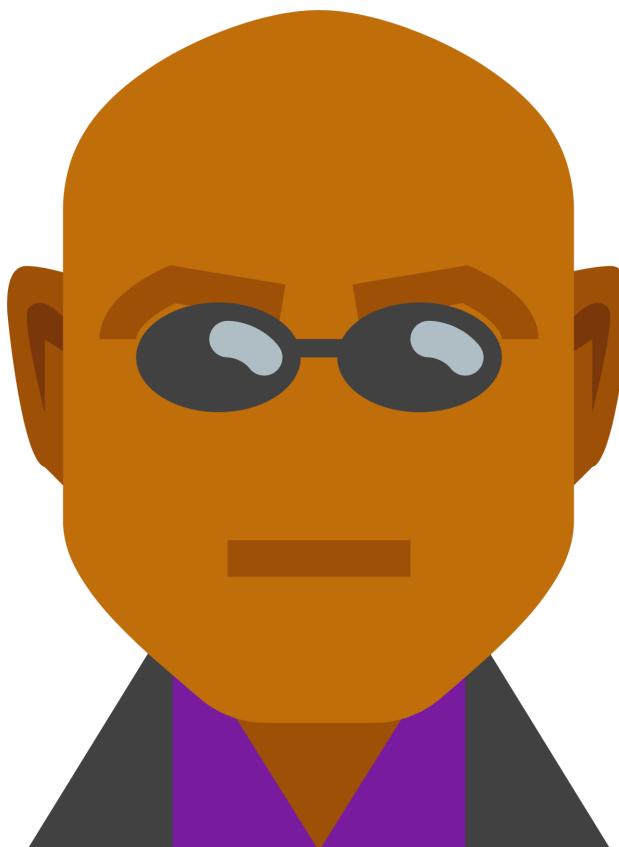


# PLANNING

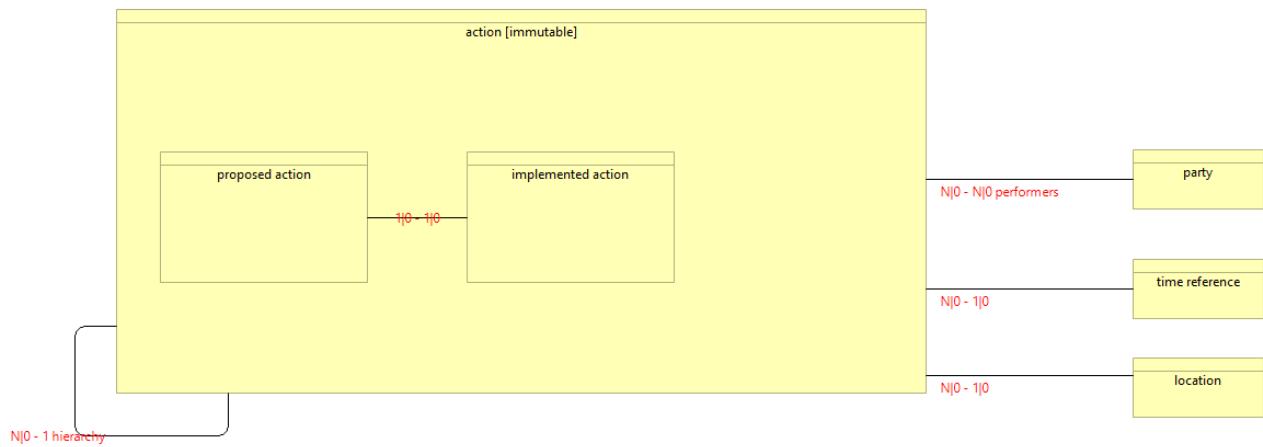
---

ANALYSIS PATTERNS

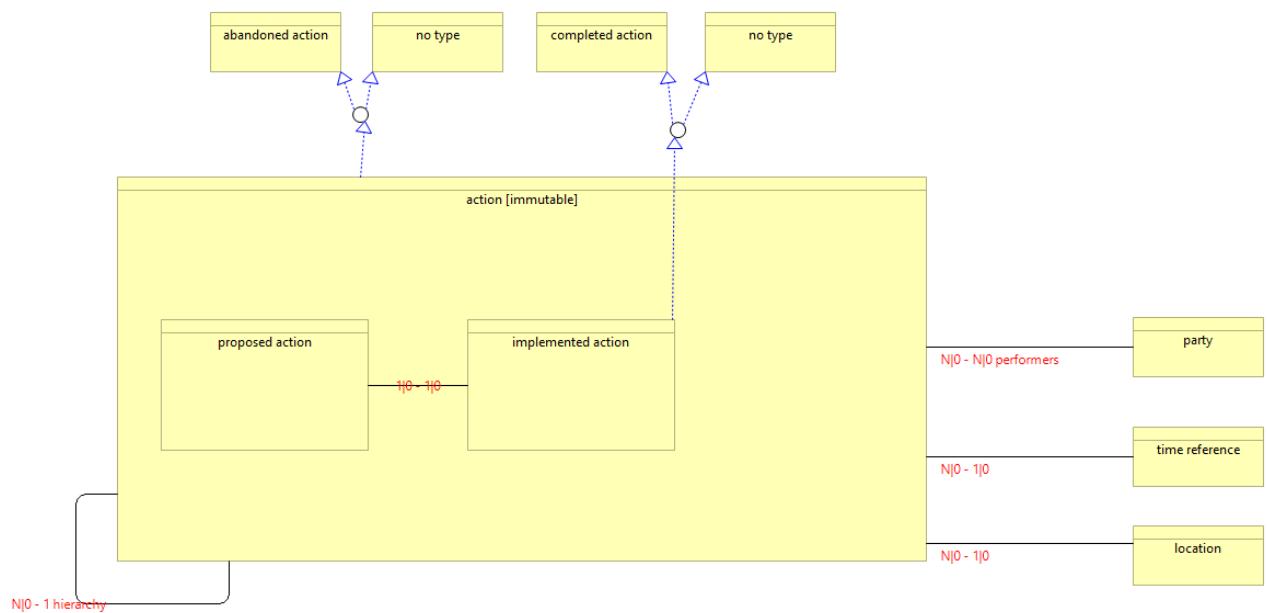
Martin Fowler



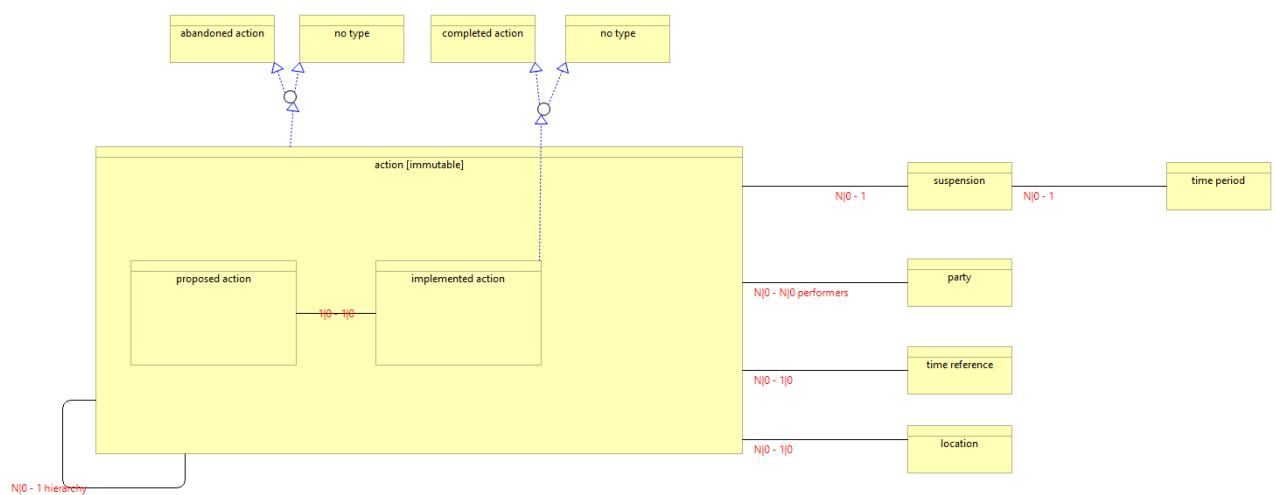
# PROPOSED AND IMPLEMENTED ACTION



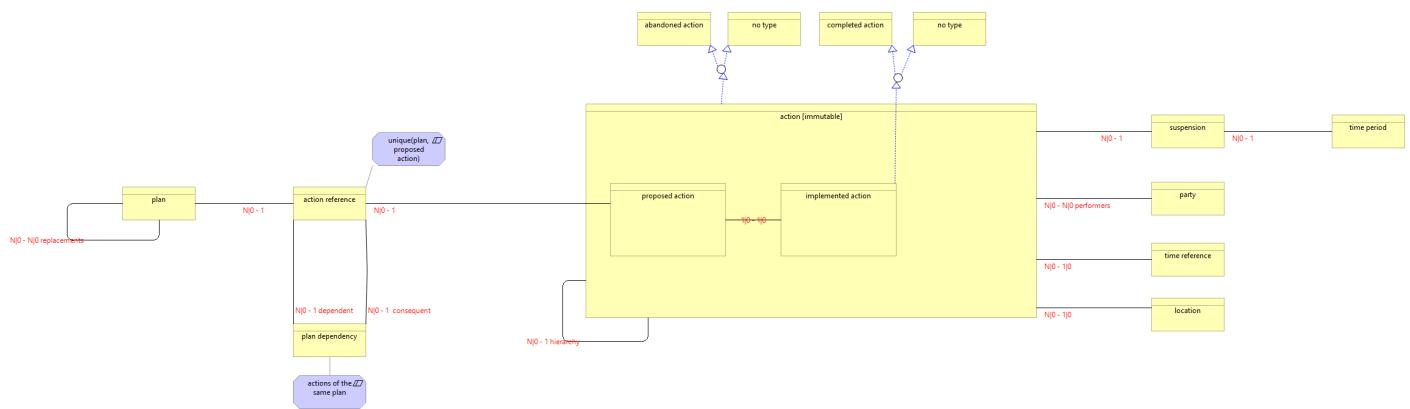
# COMPLETED AND ABANDONED ACTIONS



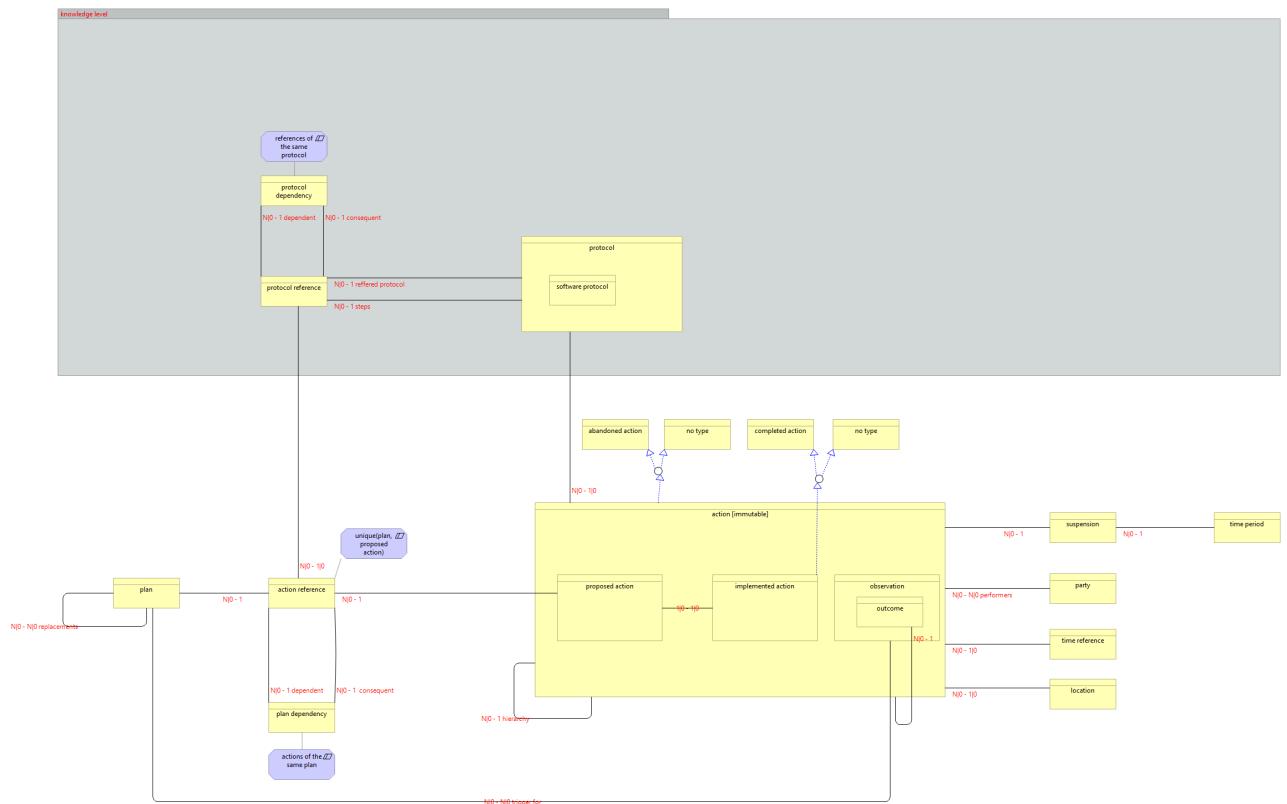
# SUSPENSION



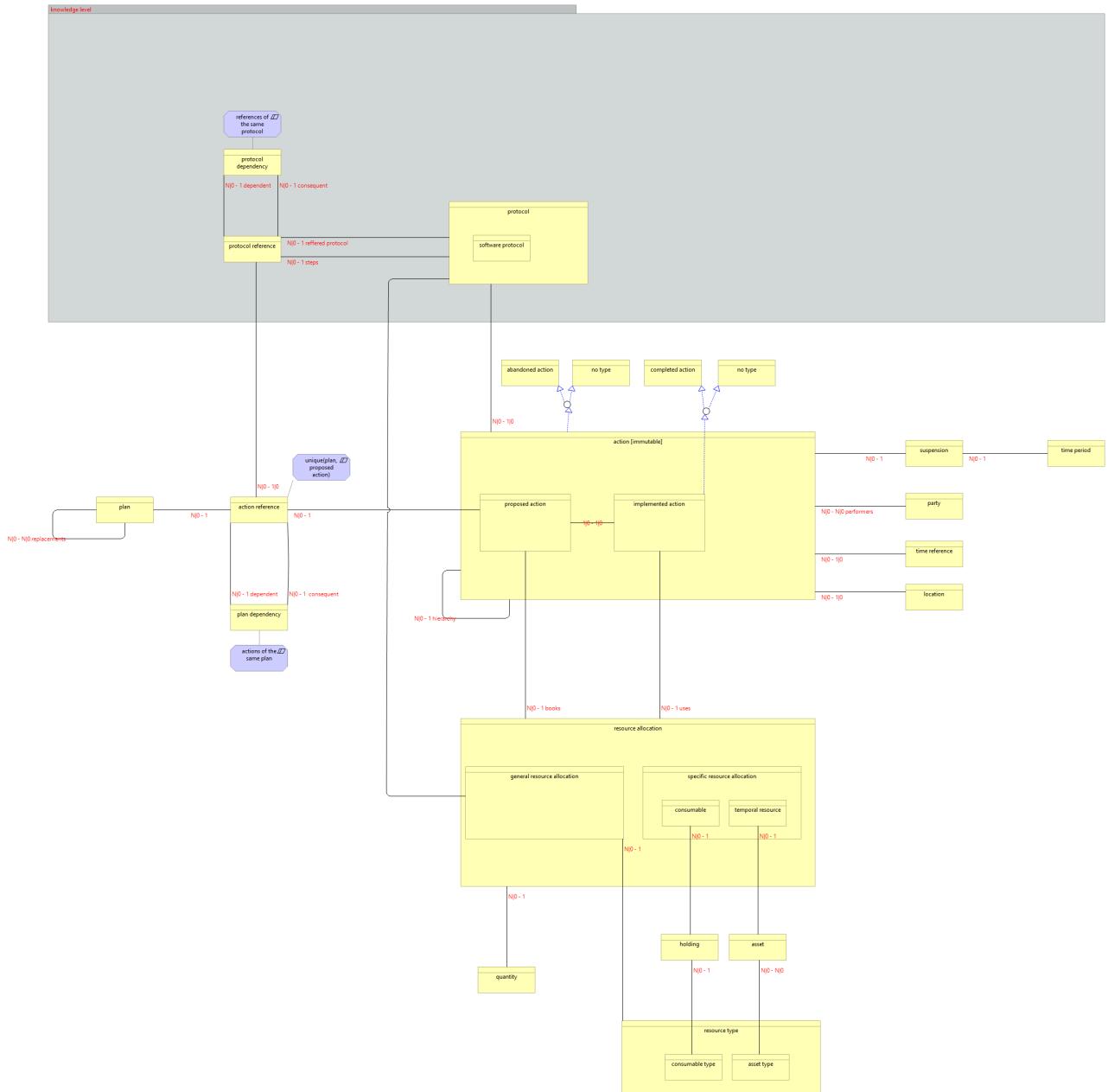
# PLAN



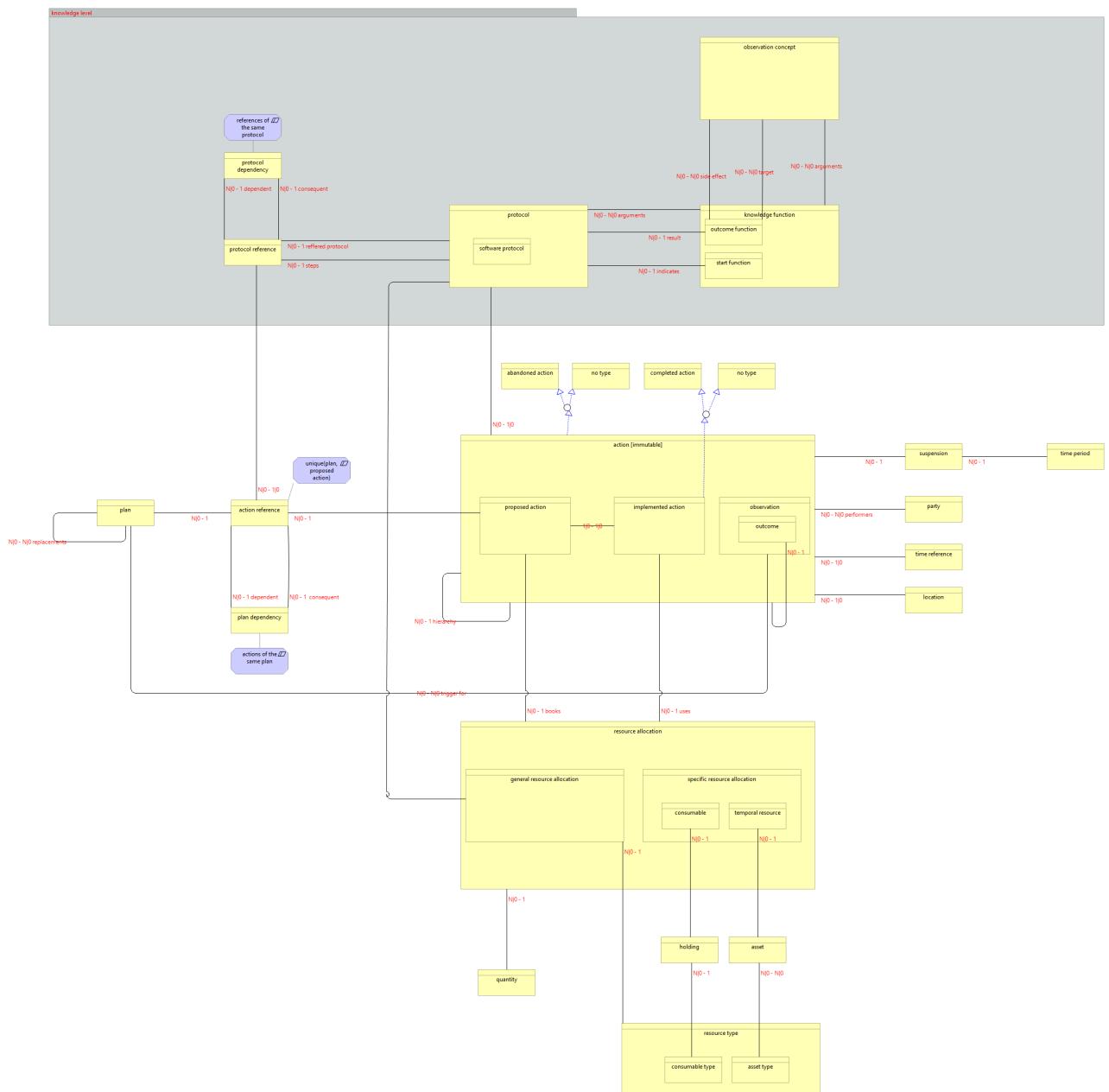
# PROTOCOL



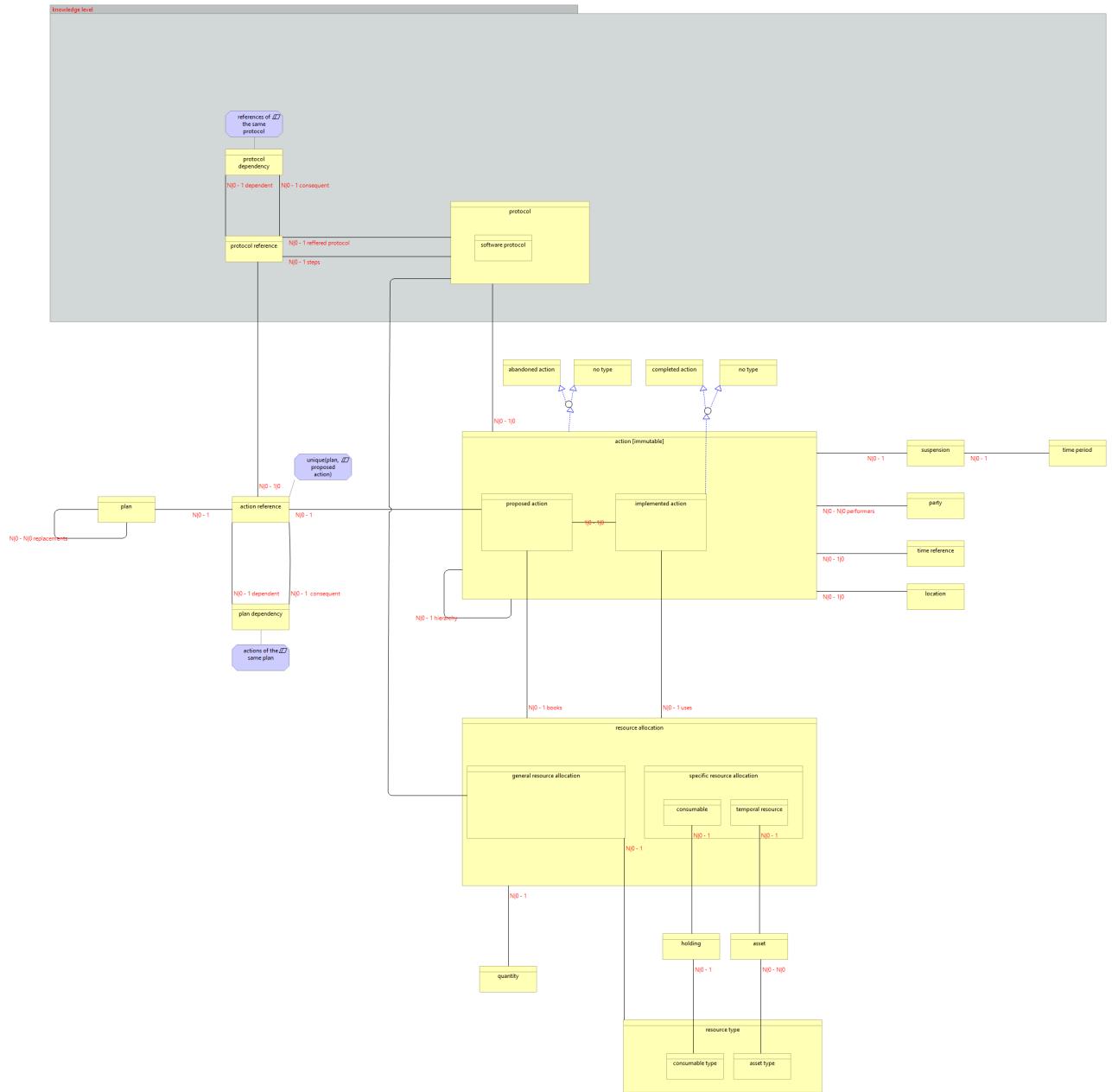
# RESOURCE ALLOCATION



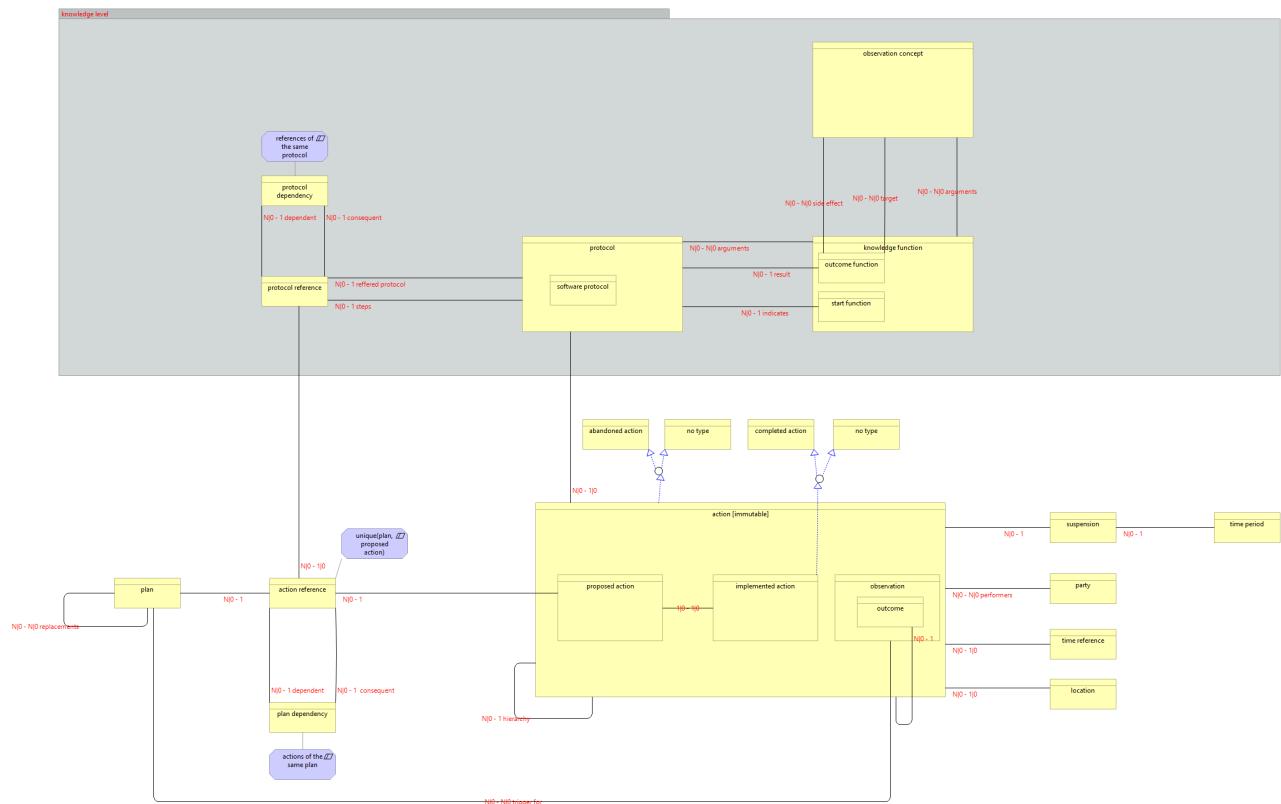
# PLANNING



# PLANNING (NO OUTCOME)



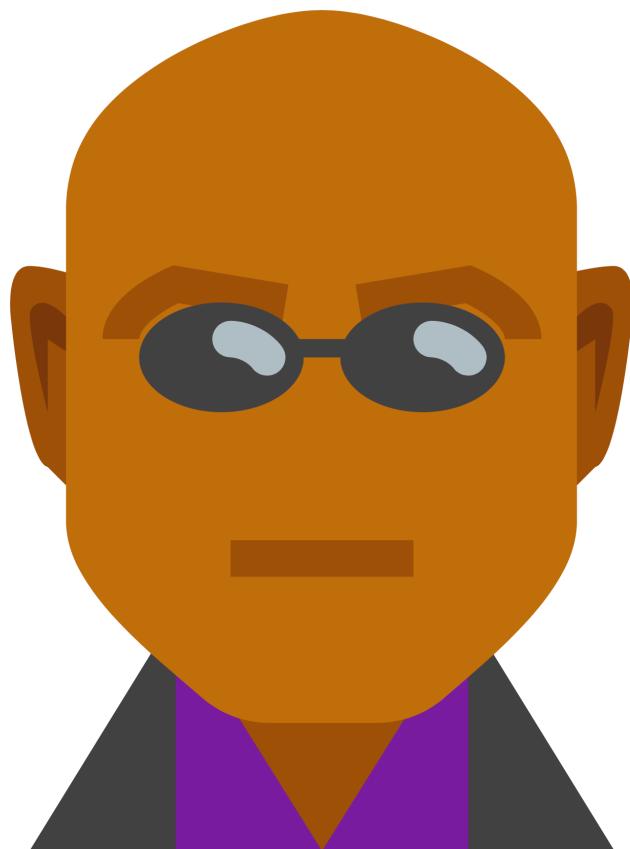
# OUTCOME AND START FUNCTIONS



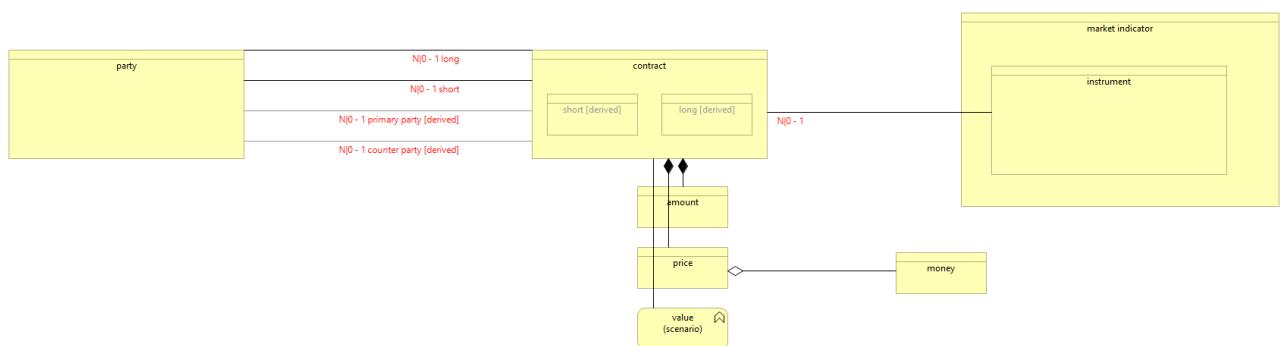
# TRADING

ANALYSIS PATTERNS

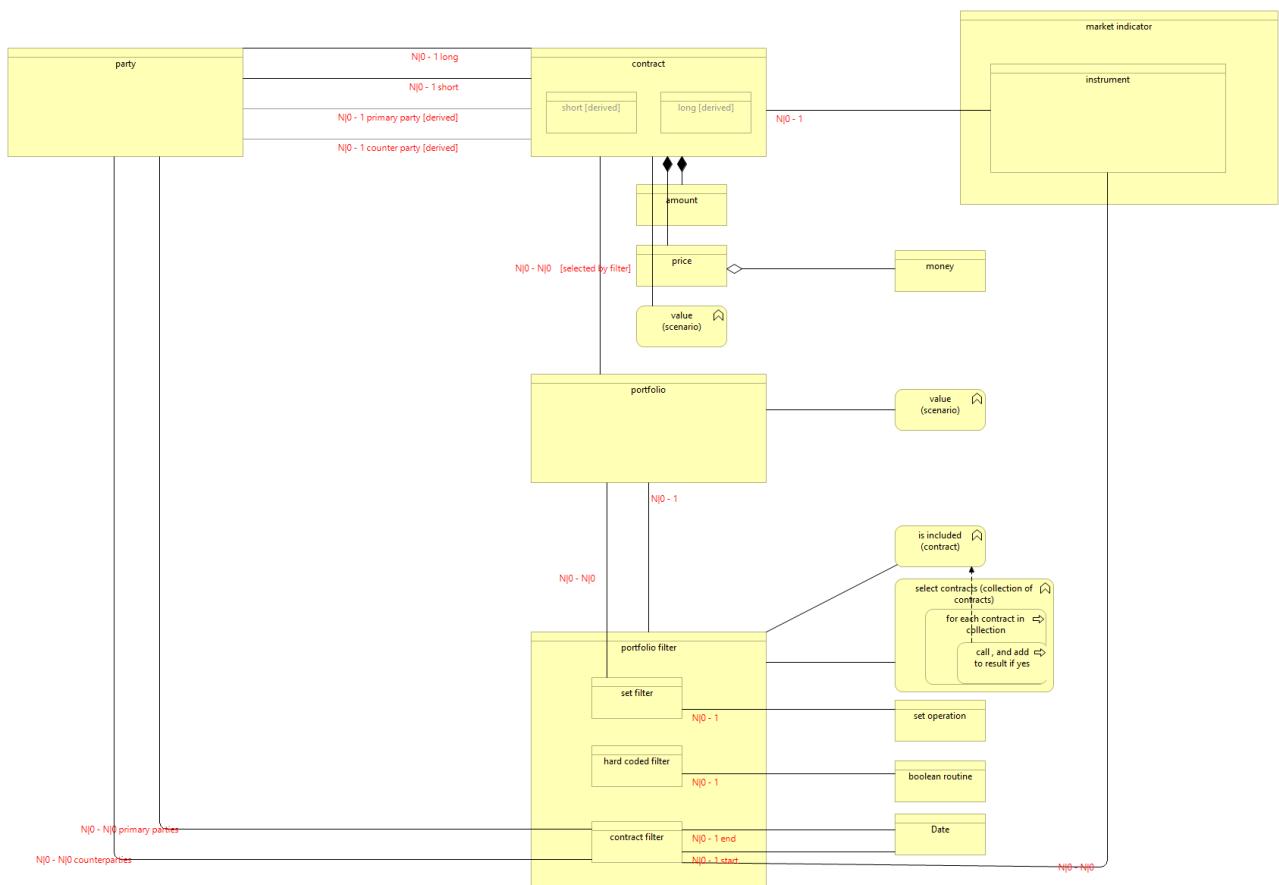
Martin Fowler



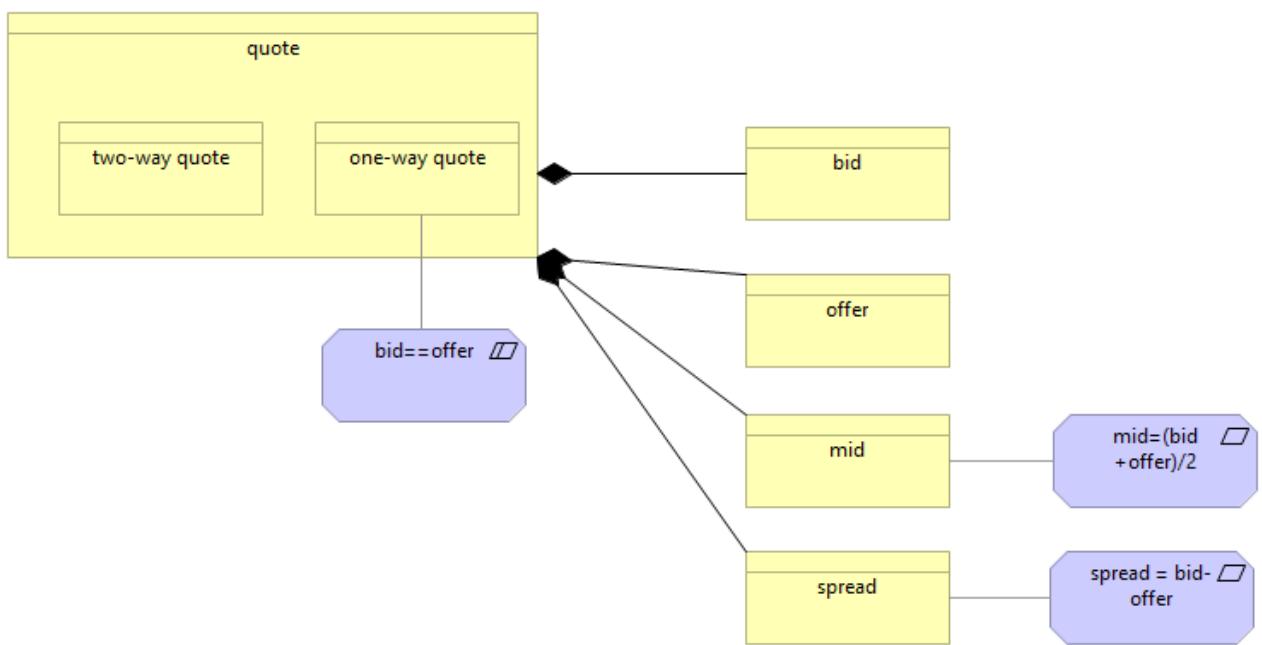
# CONTRACT



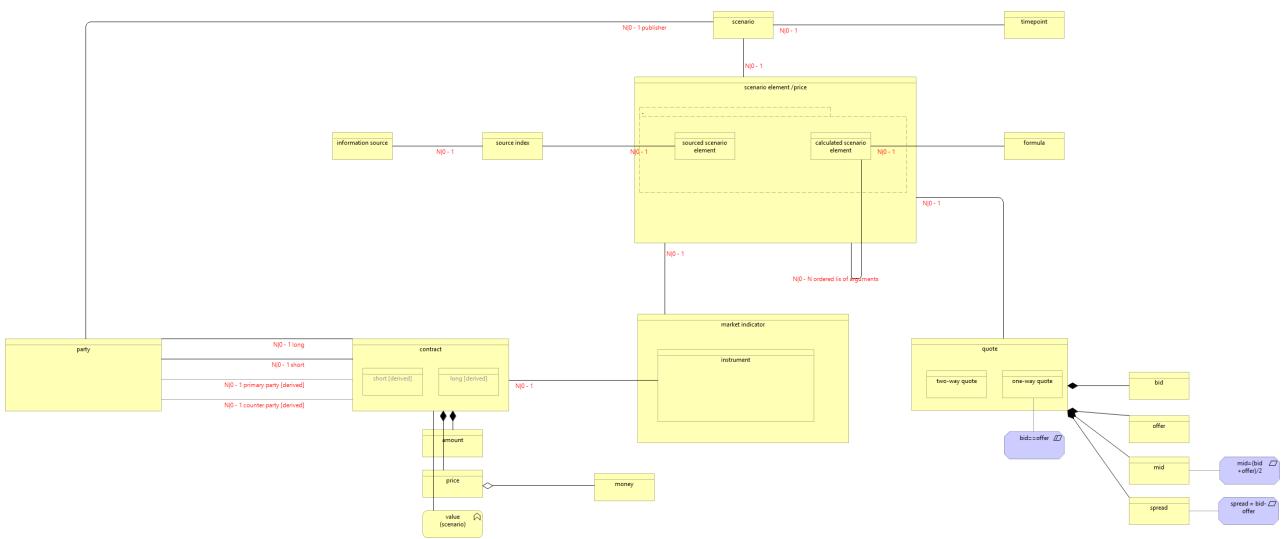
# PORTFOLIO



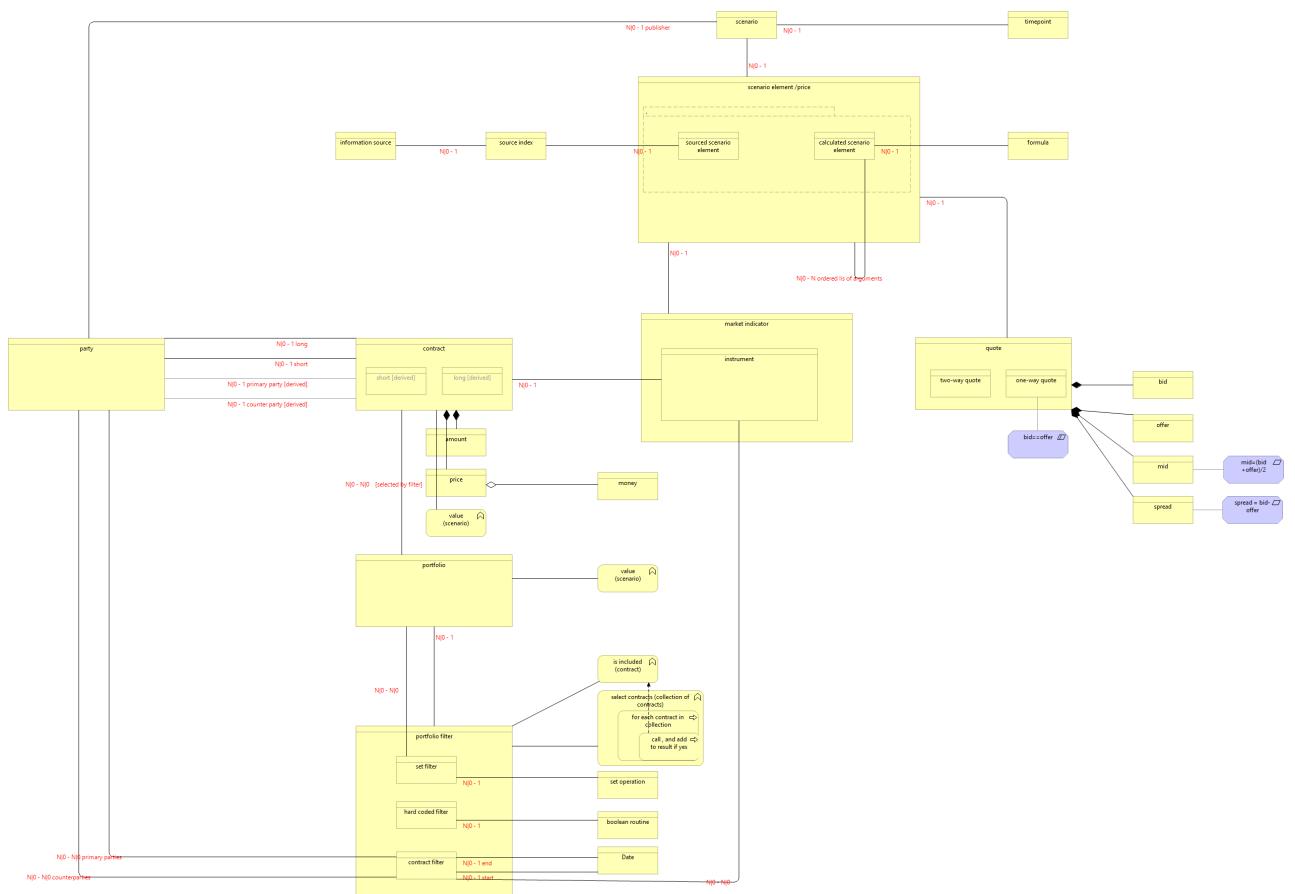
# QUOTE



# SCENARIO



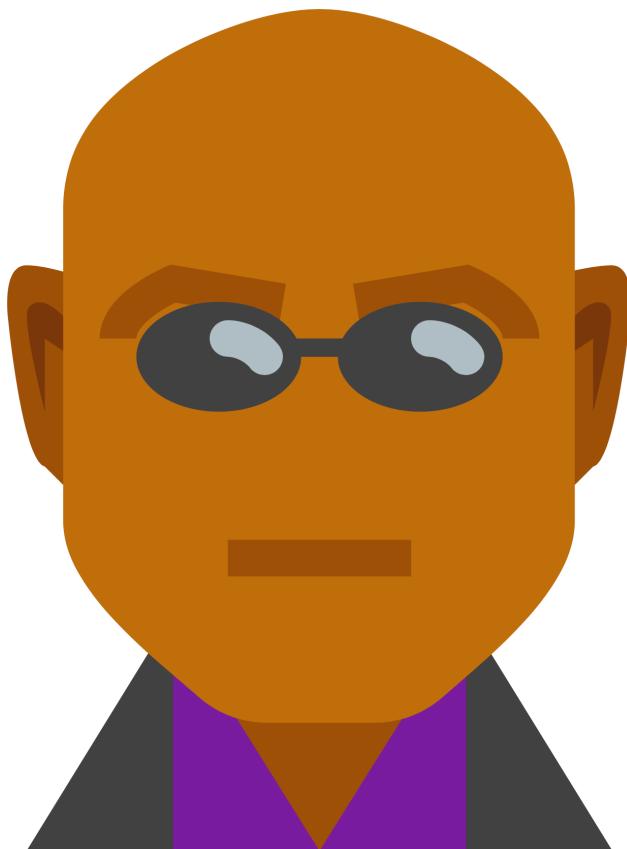
# TRADING



# DERIVATIVE CONTRACTS

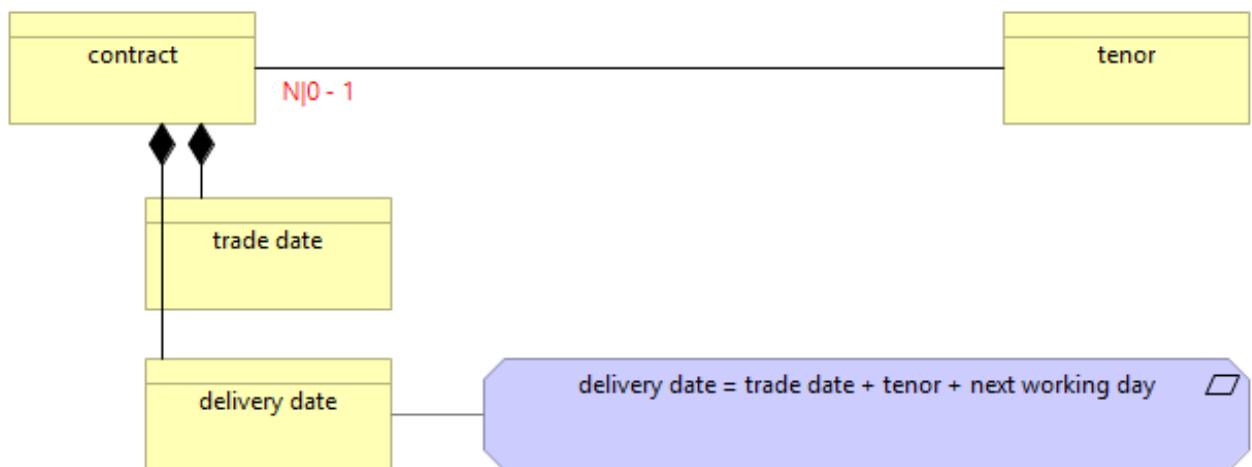
ANALYSIS PATTERNS

Martin Fowler



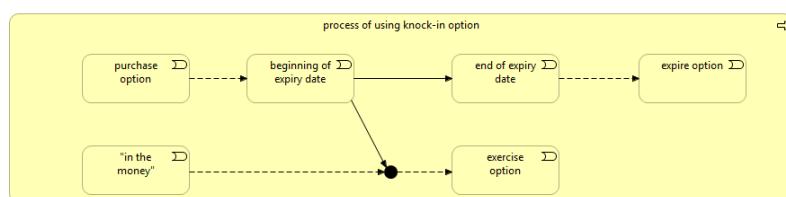
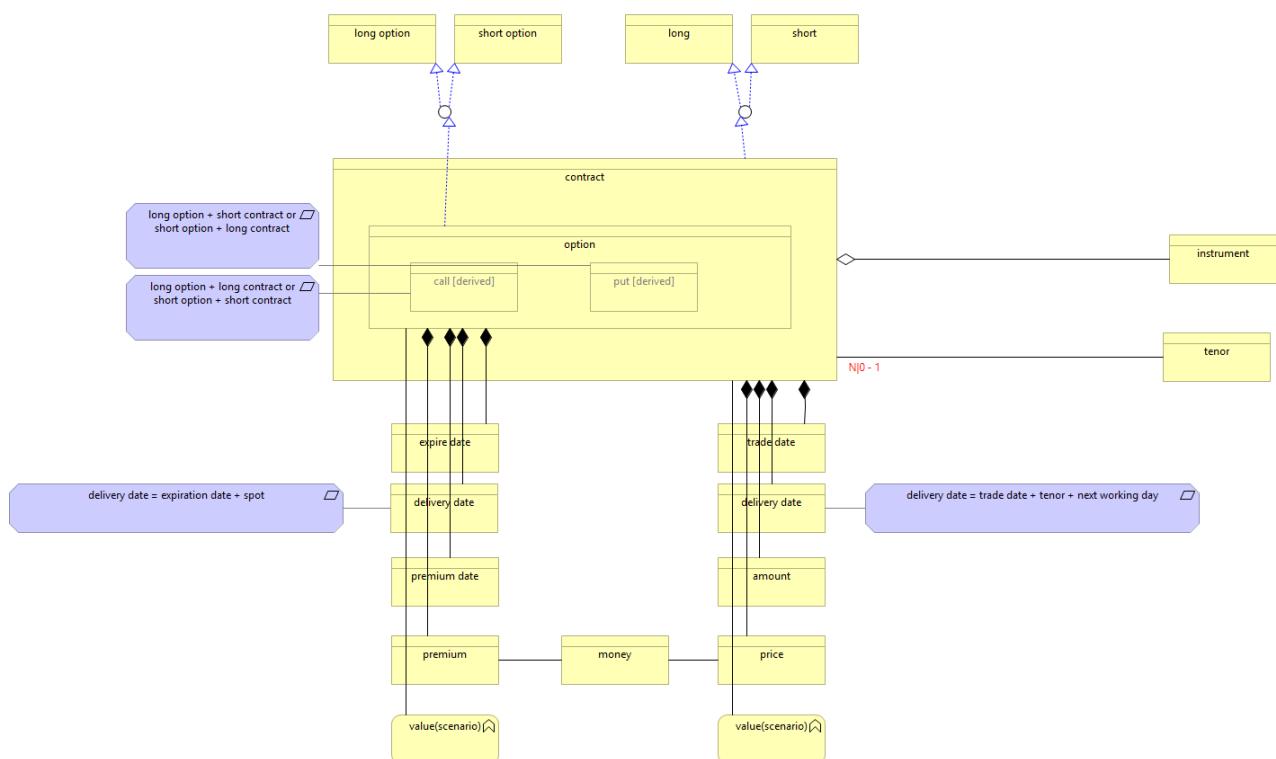
# FORWARD CONTRACTS

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



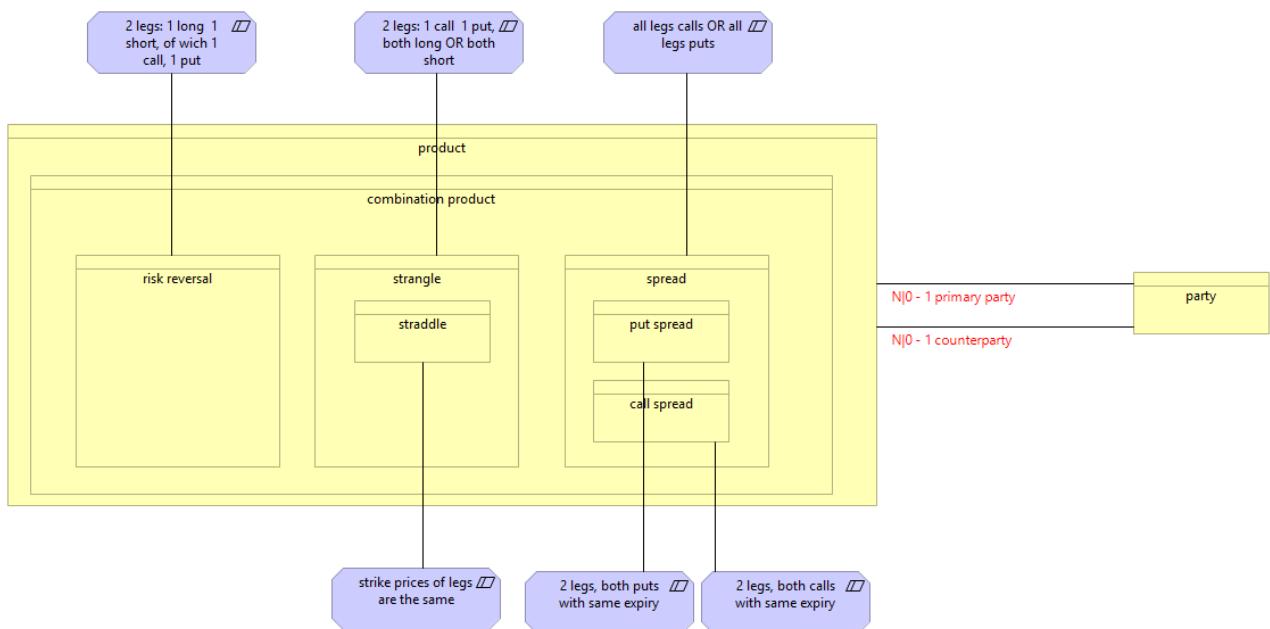
# OPTIONS

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



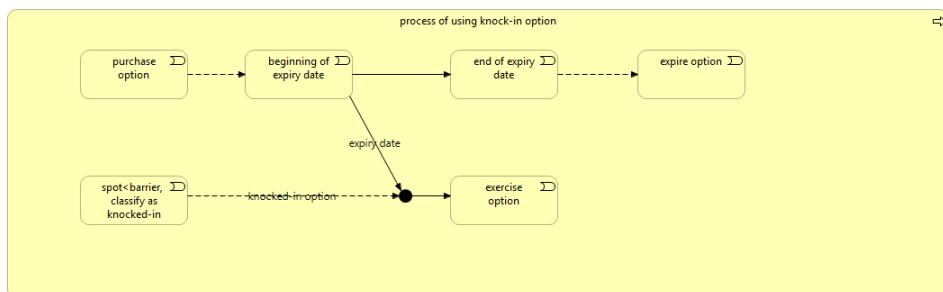
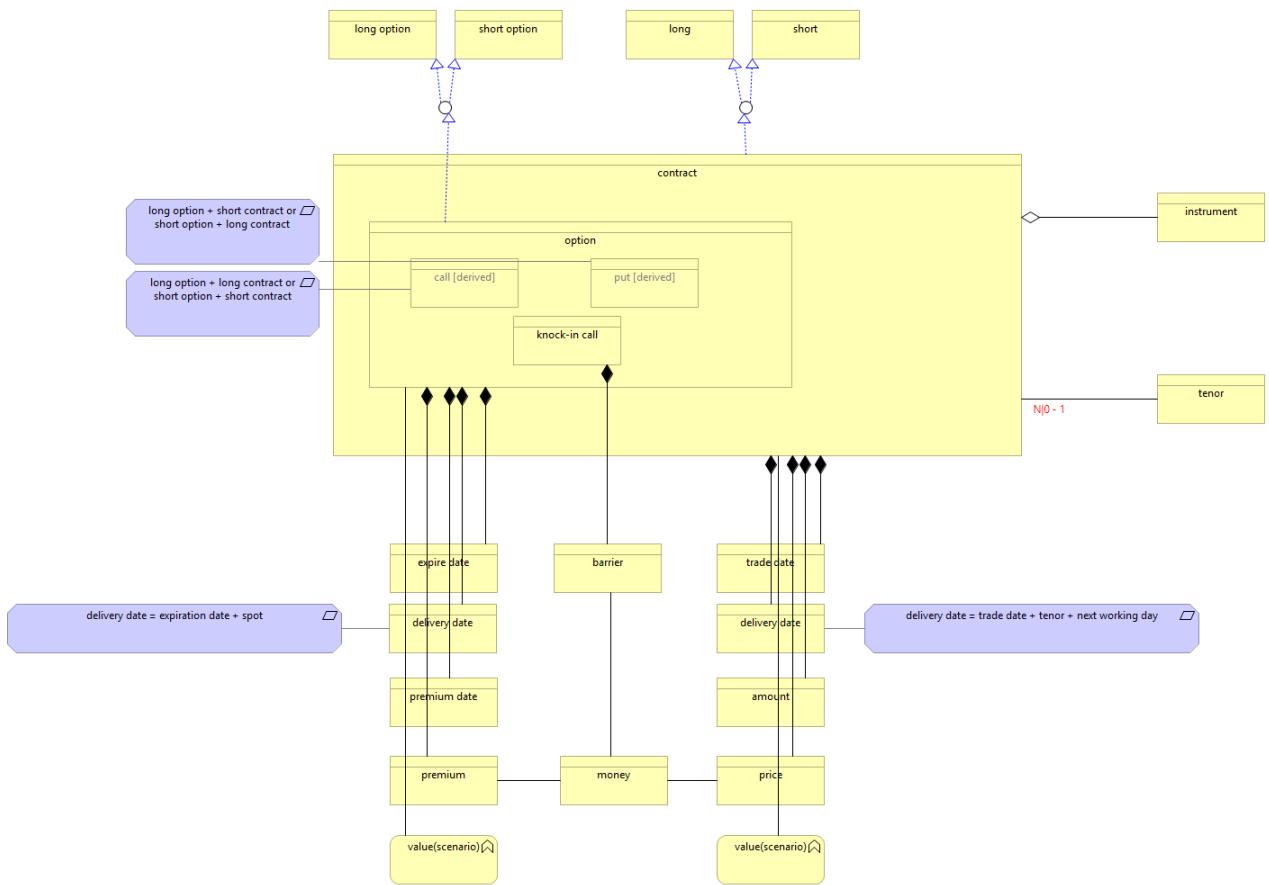
# PRODUCT

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



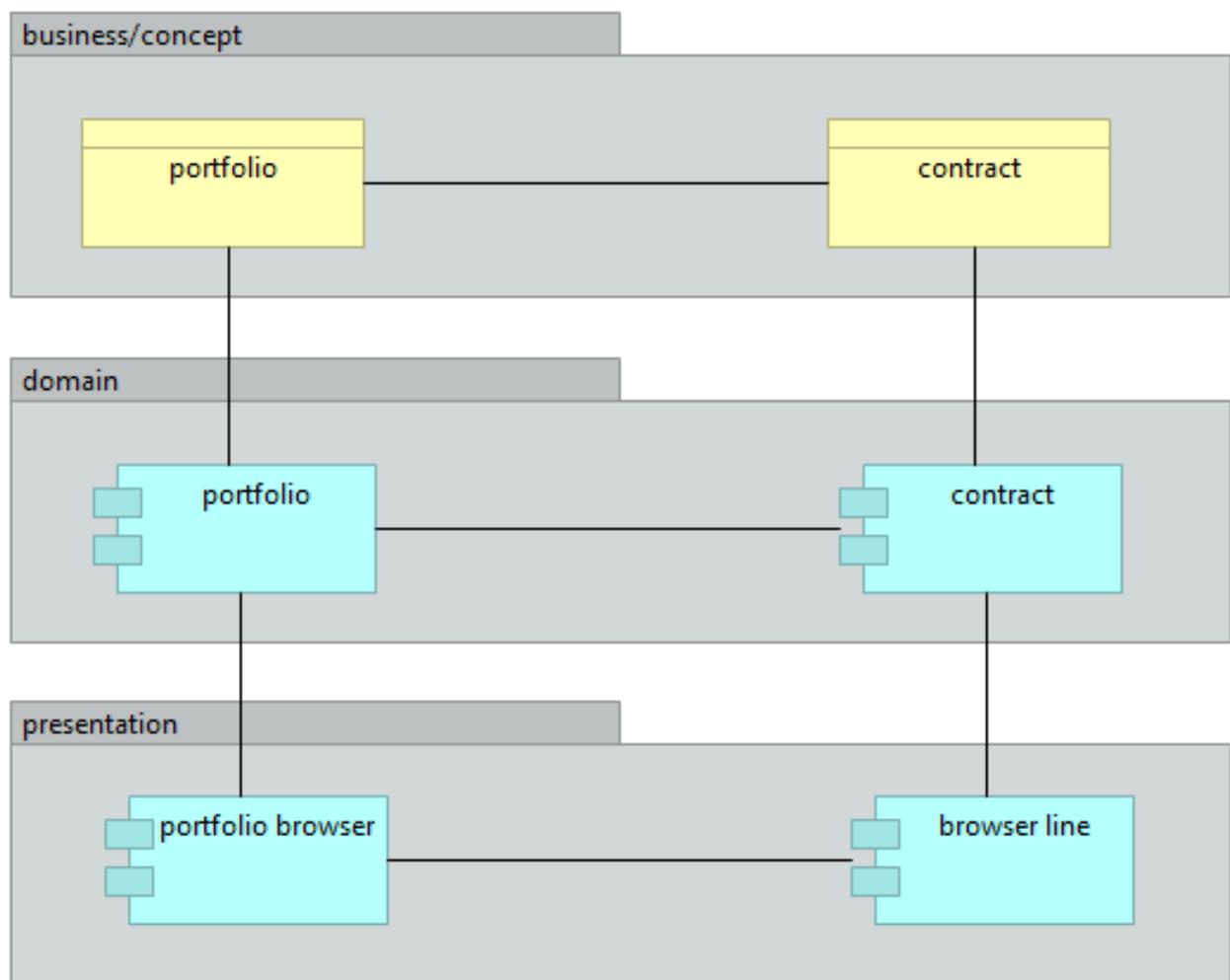
# SUBTYPE STATE MACHINES

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.

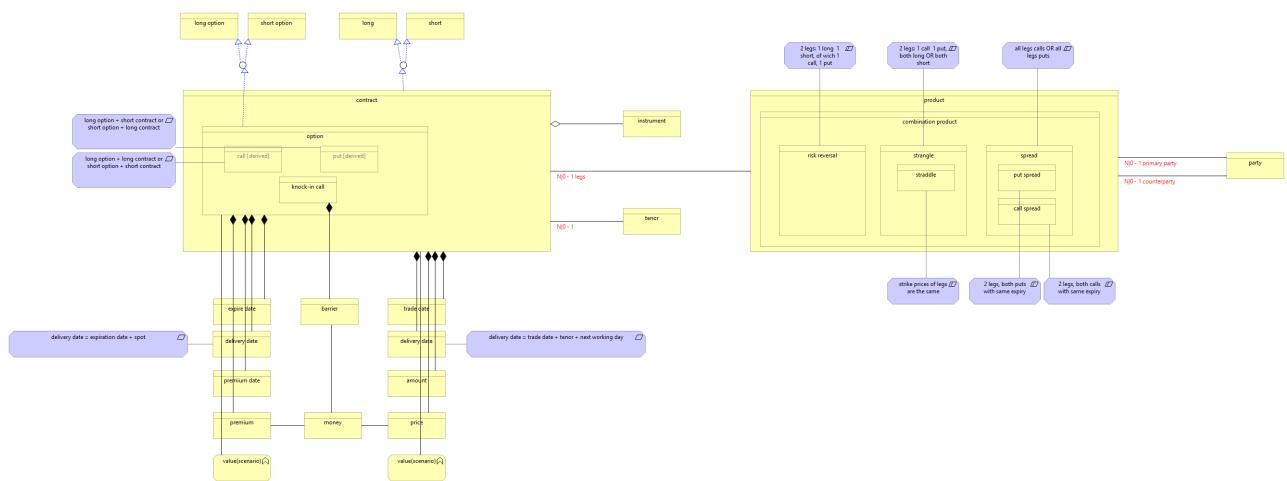


# PARALLEL APPLICATION AND DOMAIN HIERARCHIES

example, a report containing a list of contracts



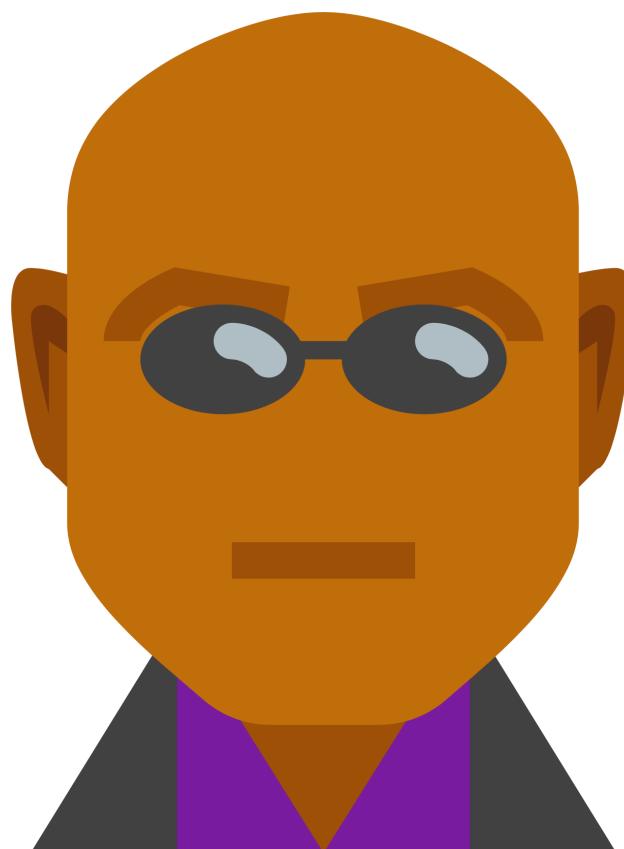
# DERIVATIVE CONTRACTS



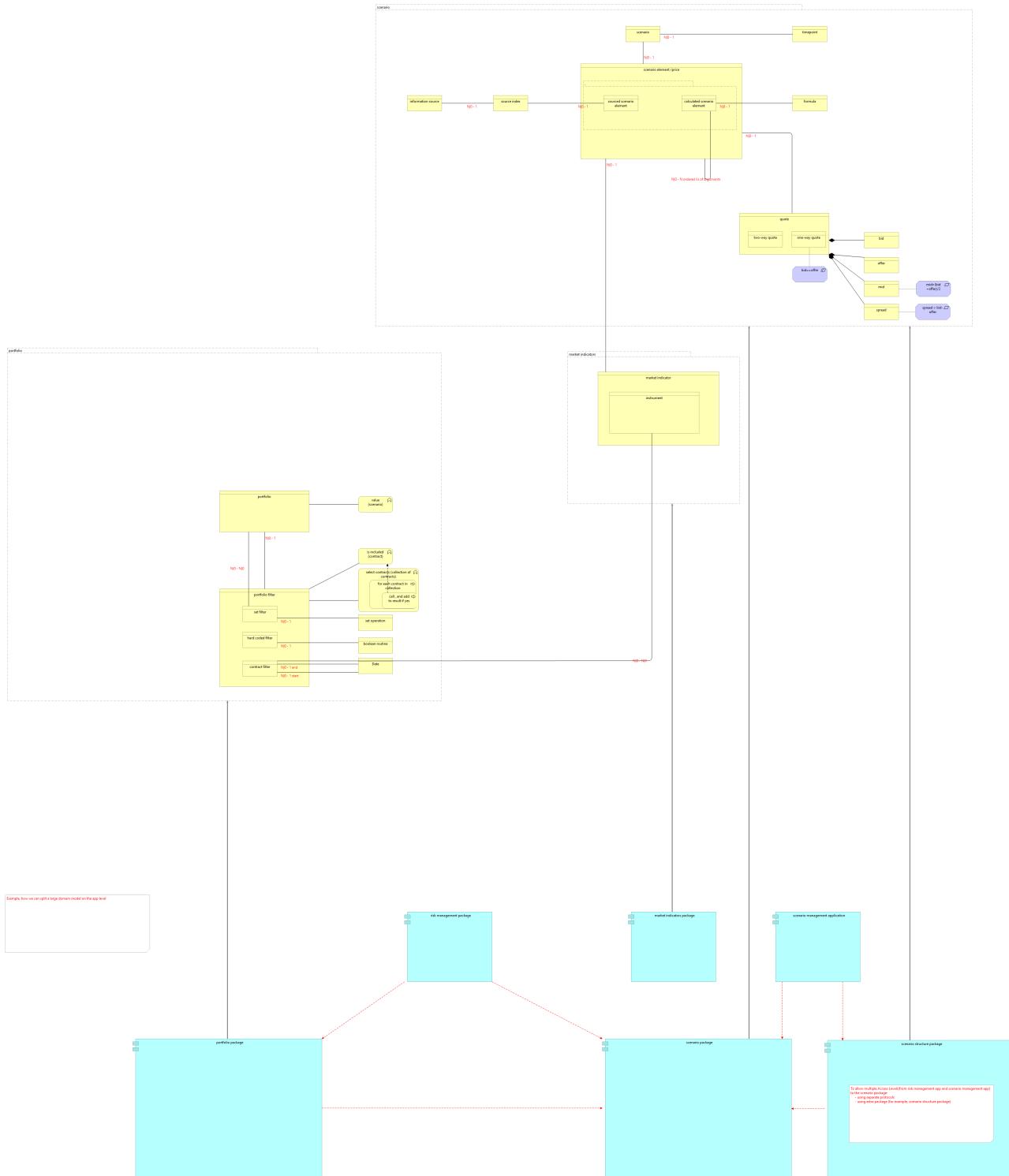
# TRADING PACKAGES

ANALYSIS PATTERNS

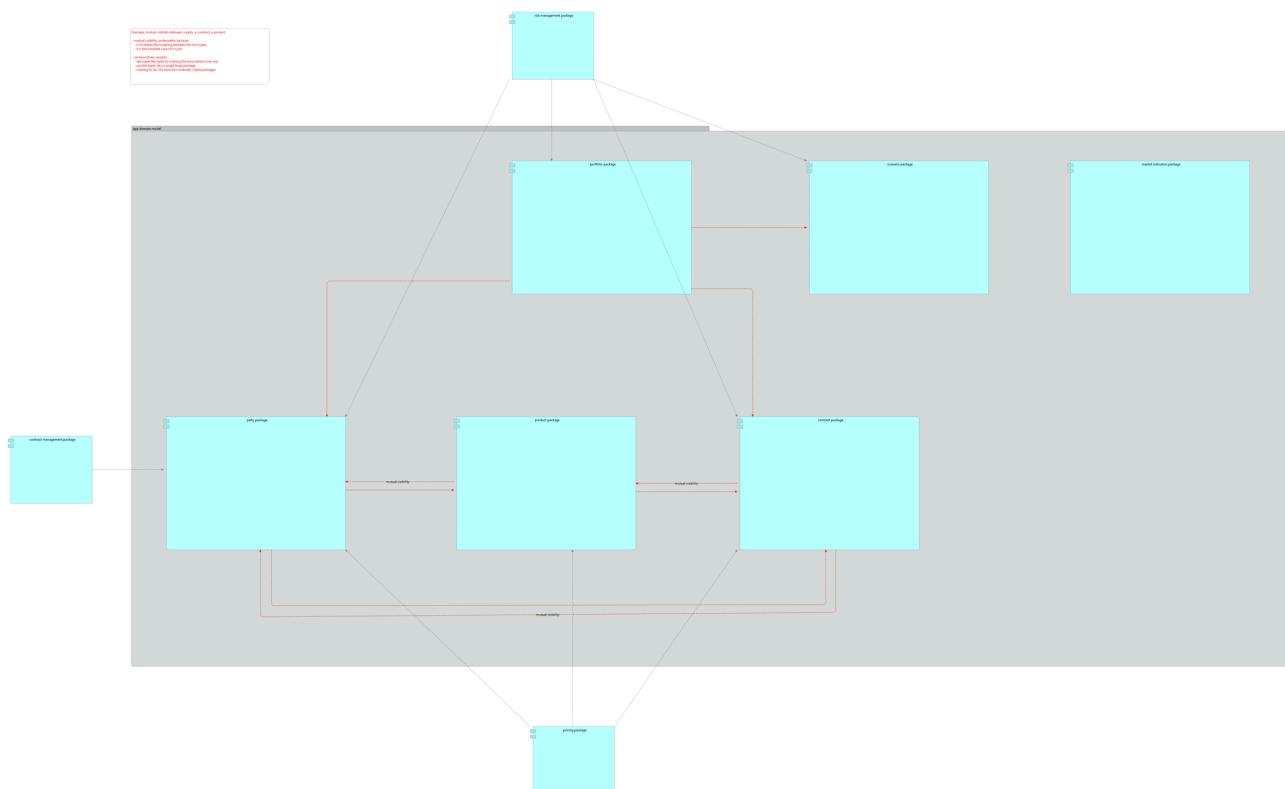
Martin Fowler



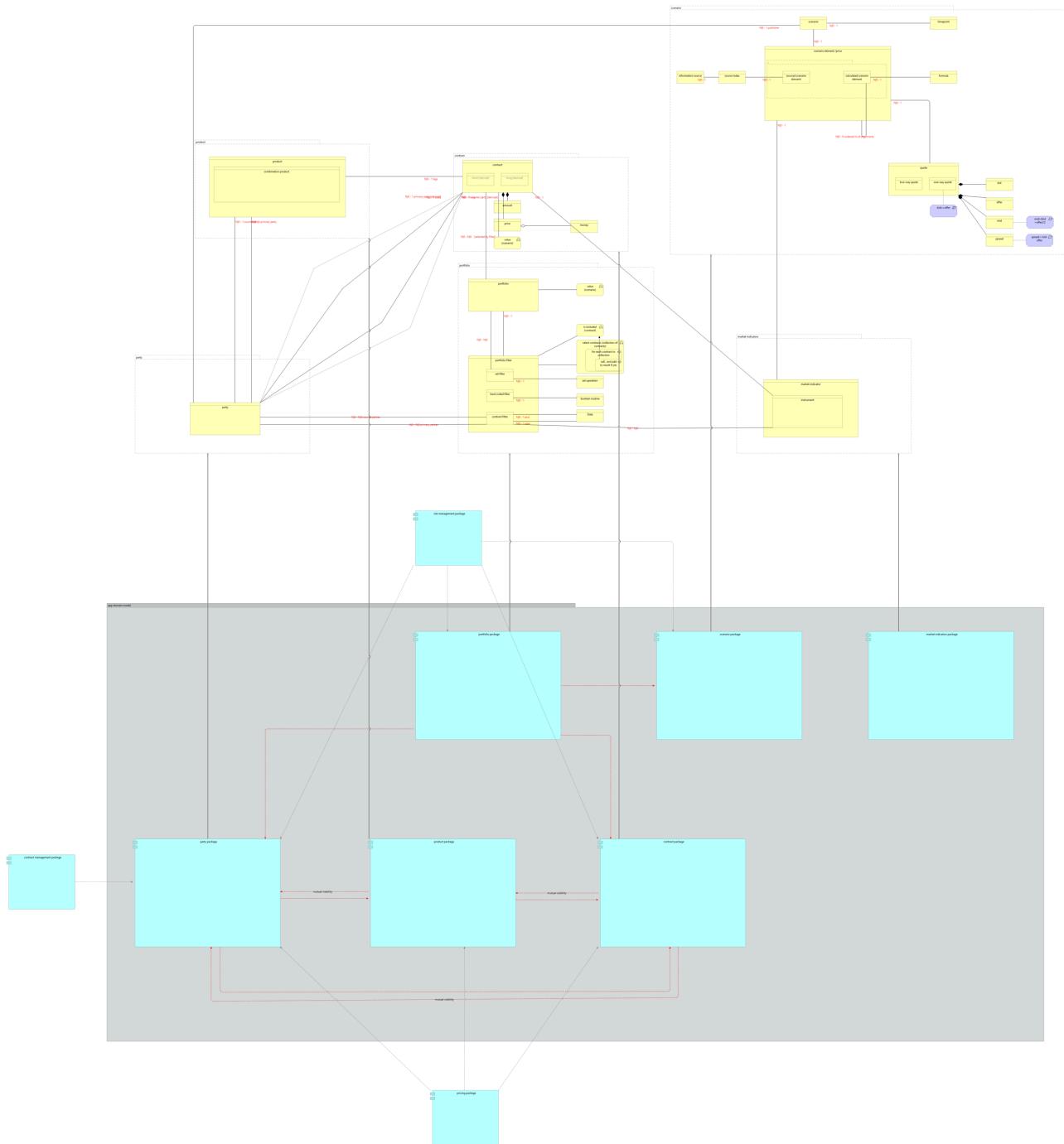
# MULTIPLE ACCESS LEVELS TO A PACKAGE



# MUTUAL VISIBILITY



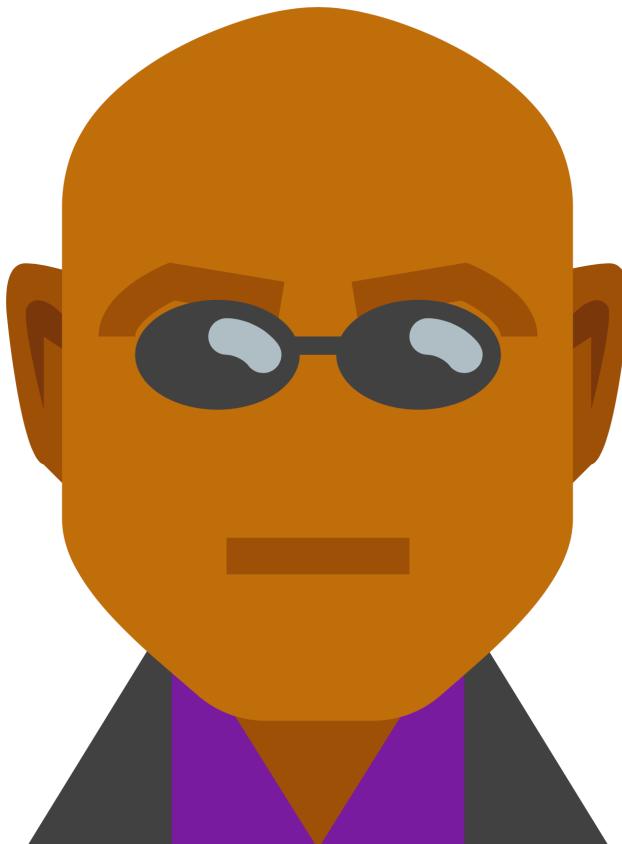
# TRADING PACKAGES



# LAYERED ARCHITECTURE

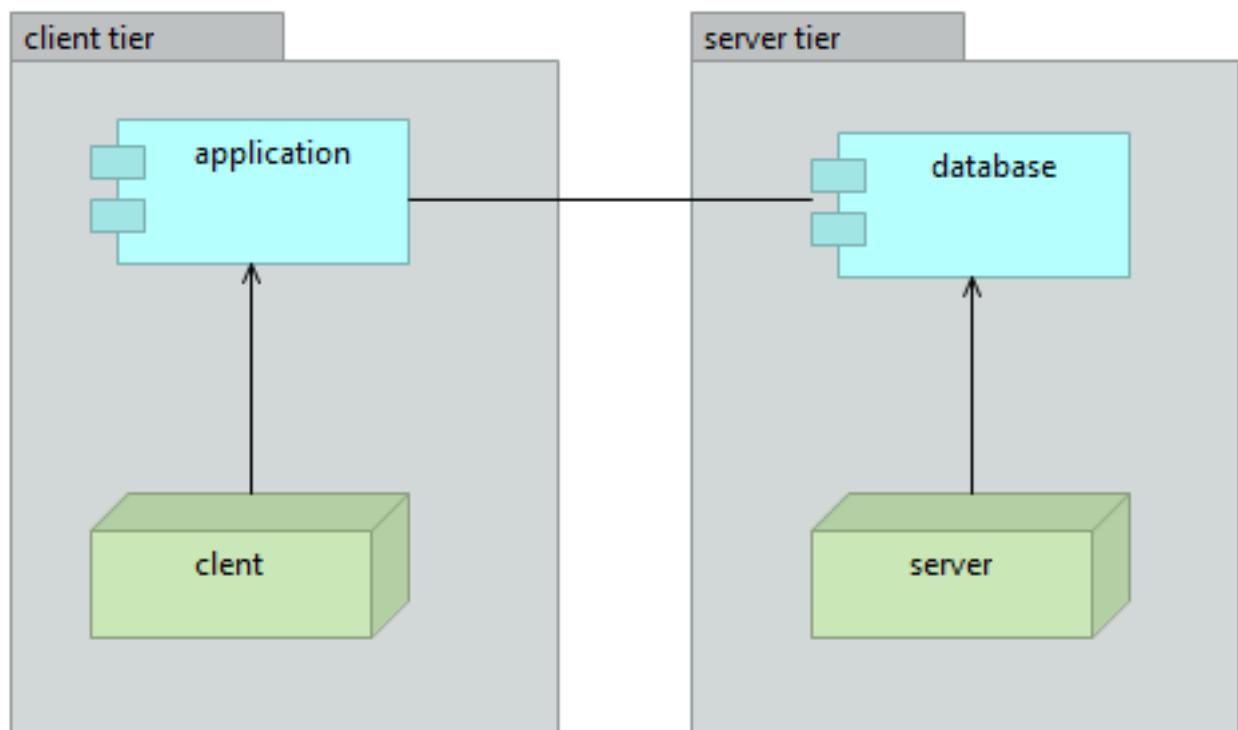
ANALYSIS PATTERNS

Martin Fowler

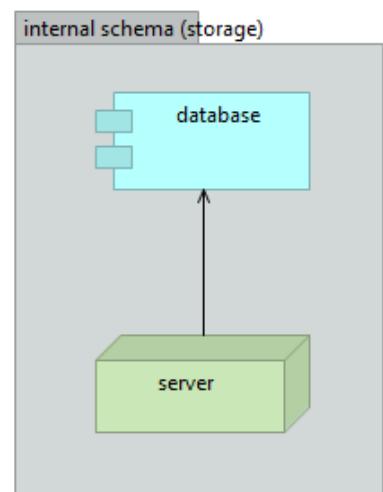
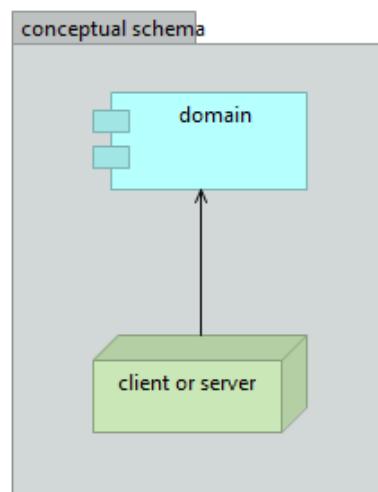
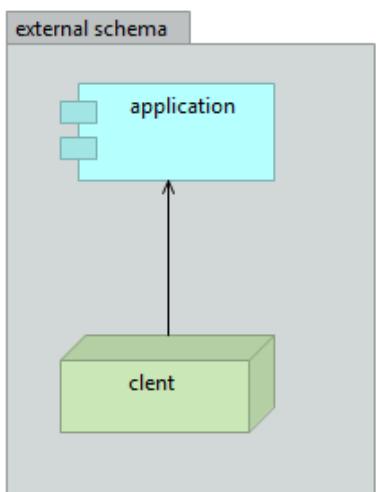


# TWO-TIER ARCHITECTURE

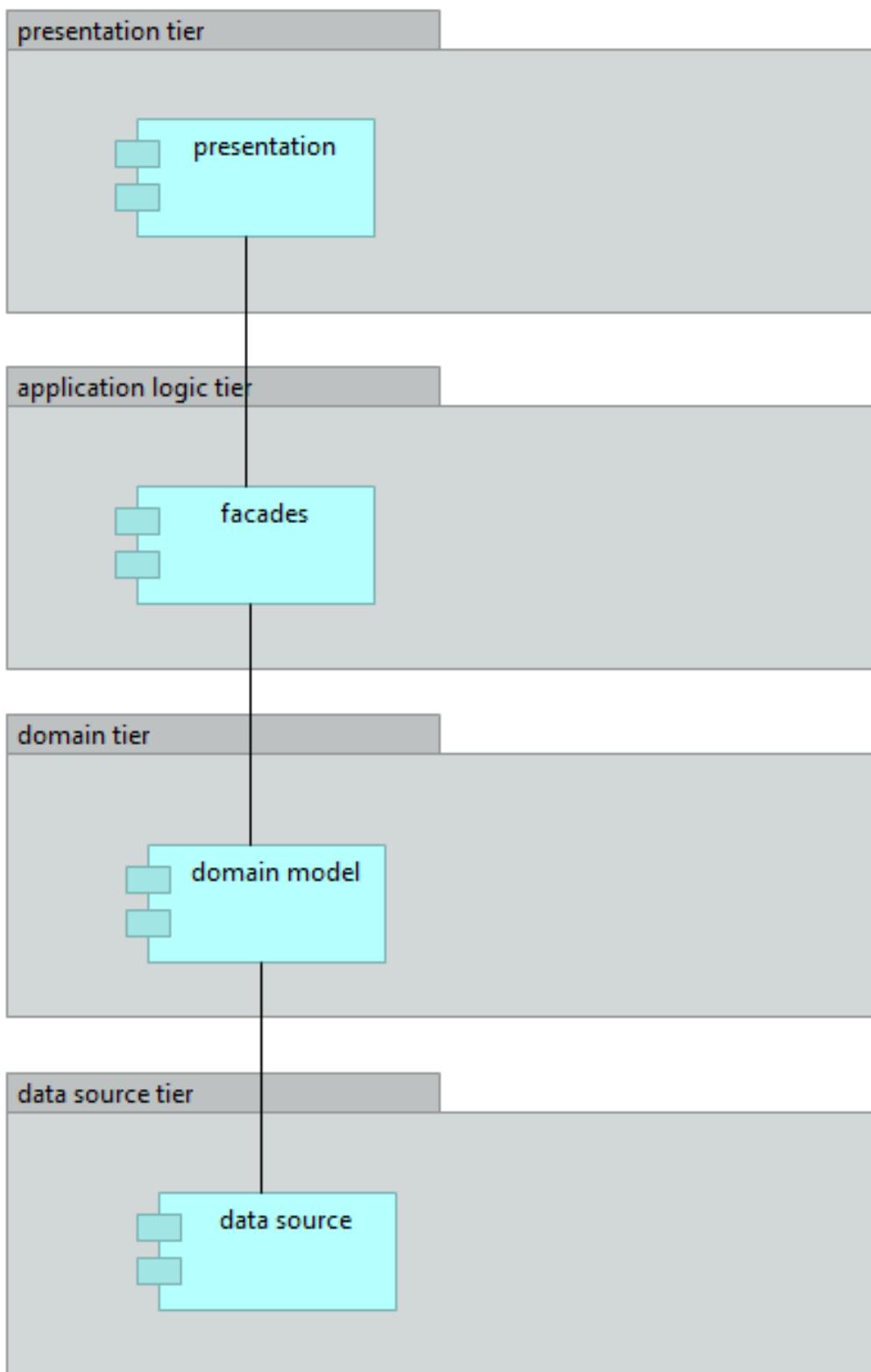
two-tier architecture



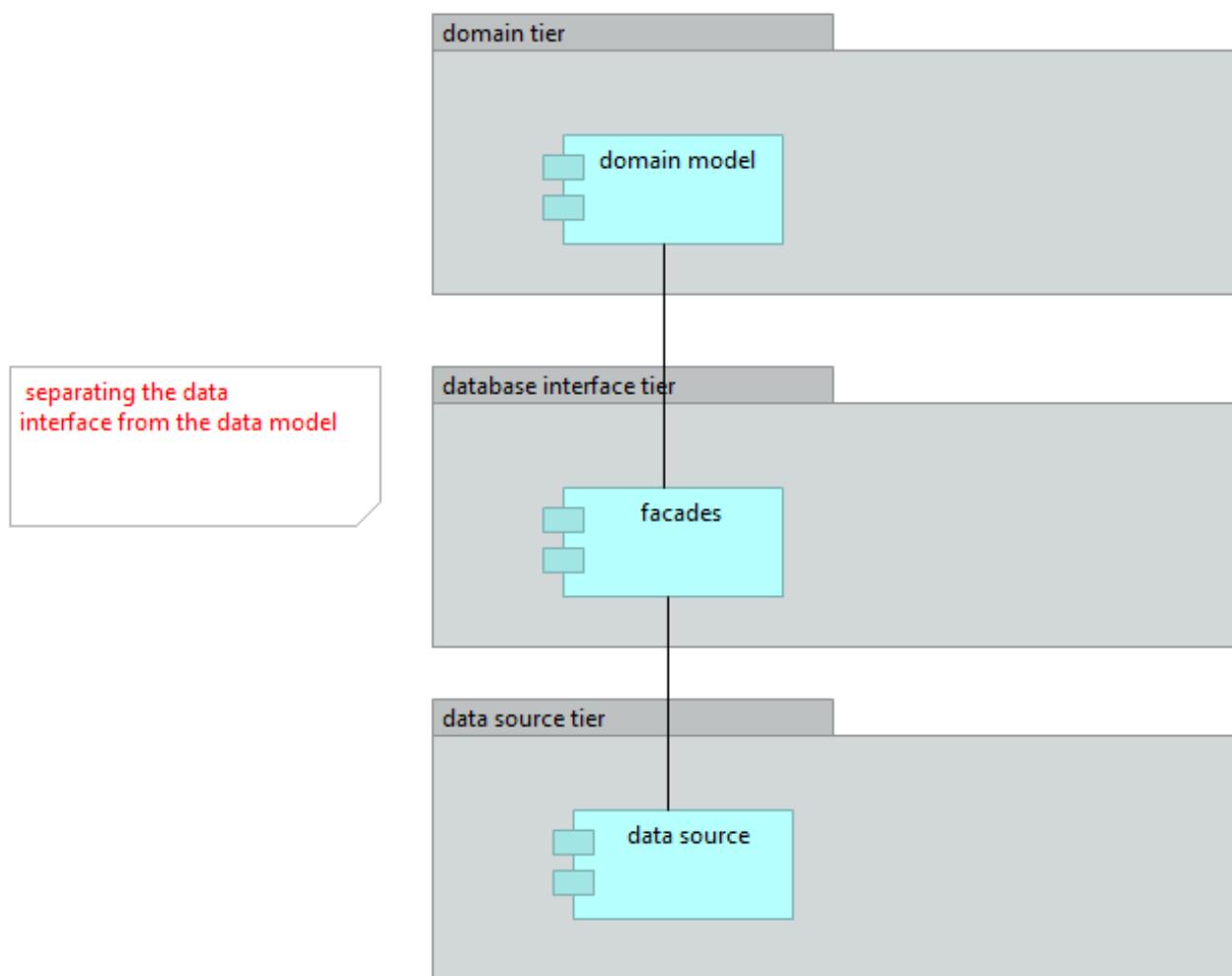
# THREE-TIER ARCHITECTURE



# PRESENTATION AND APPLICATION LOGIC



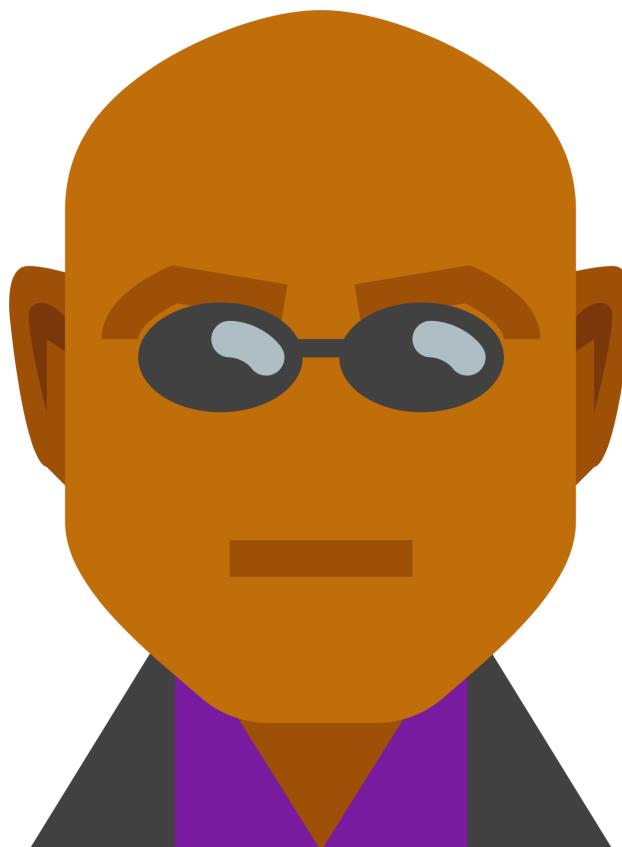
# DATABASE INTERACTION



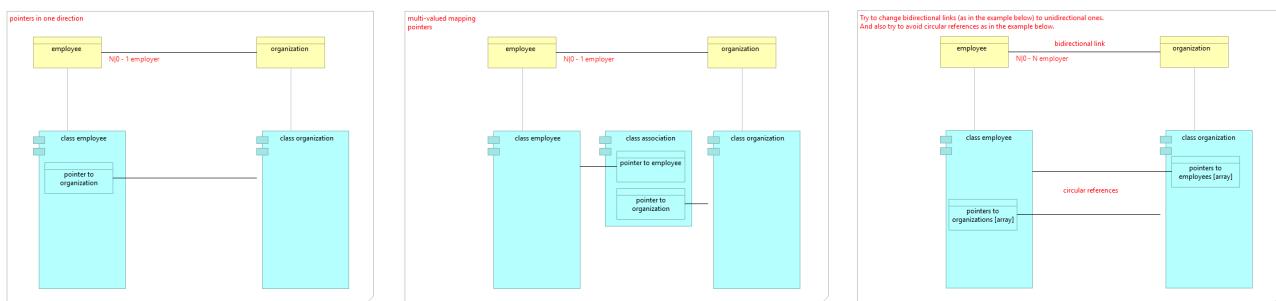
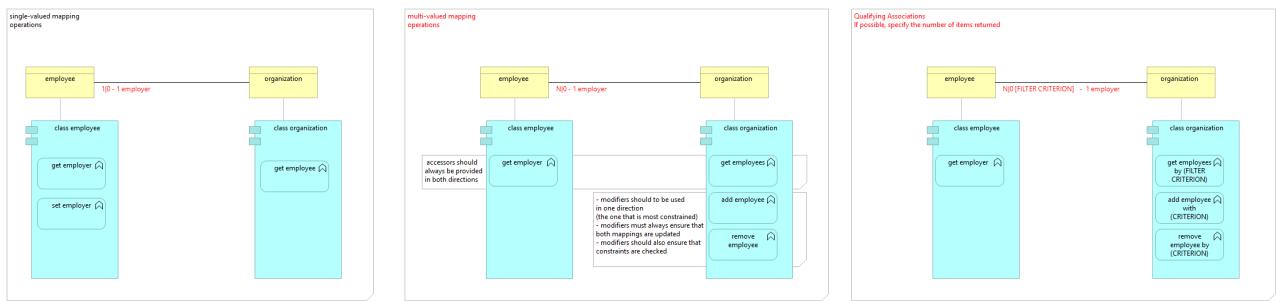
# TYPE MODEL DESIGN

ANALYSIS PATTERNS

Martin Fowler

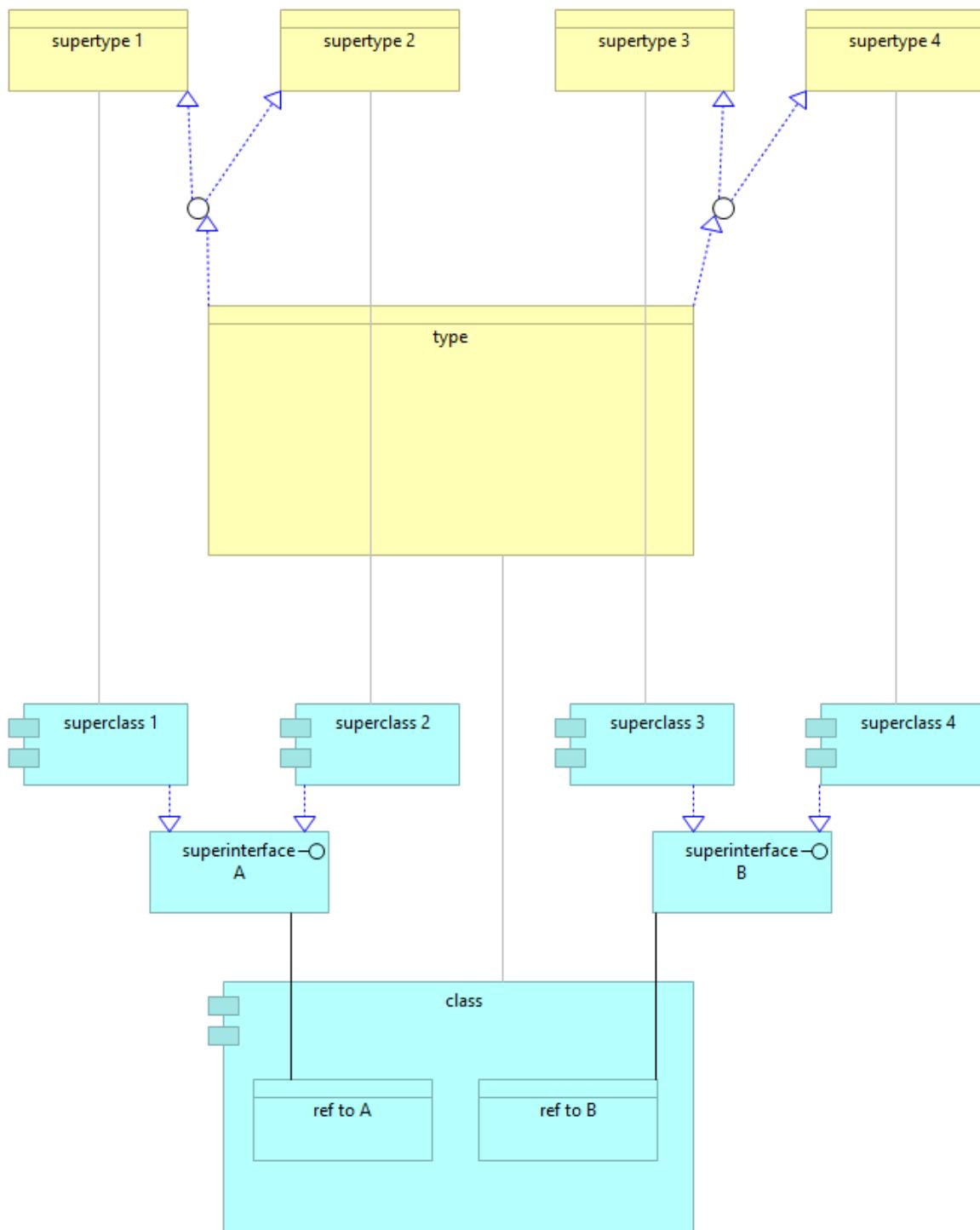


# IMPLEMENTING ASSOCIATIONS

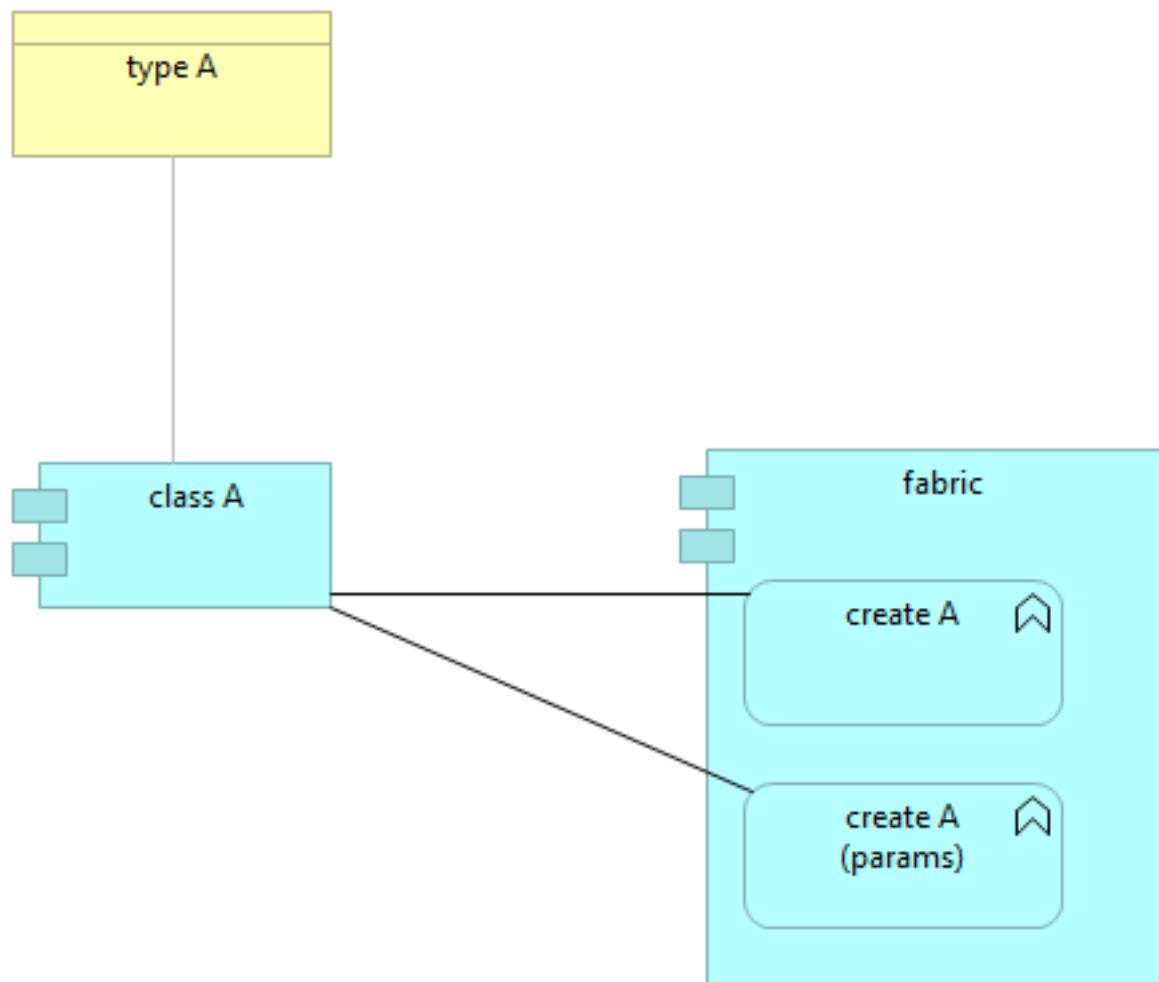


# IMPLEMENTING GENERALIZATION

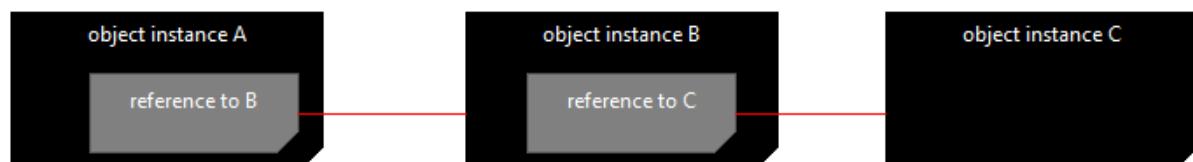
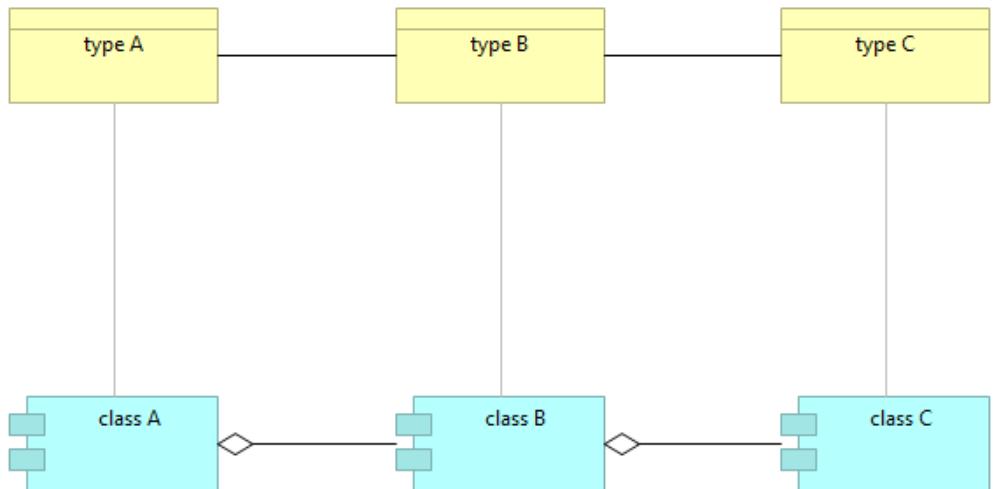
- example of implementation of multiple inheritance
- for example, the composition is applicable instead of inheritance



# OBJECT CREATION



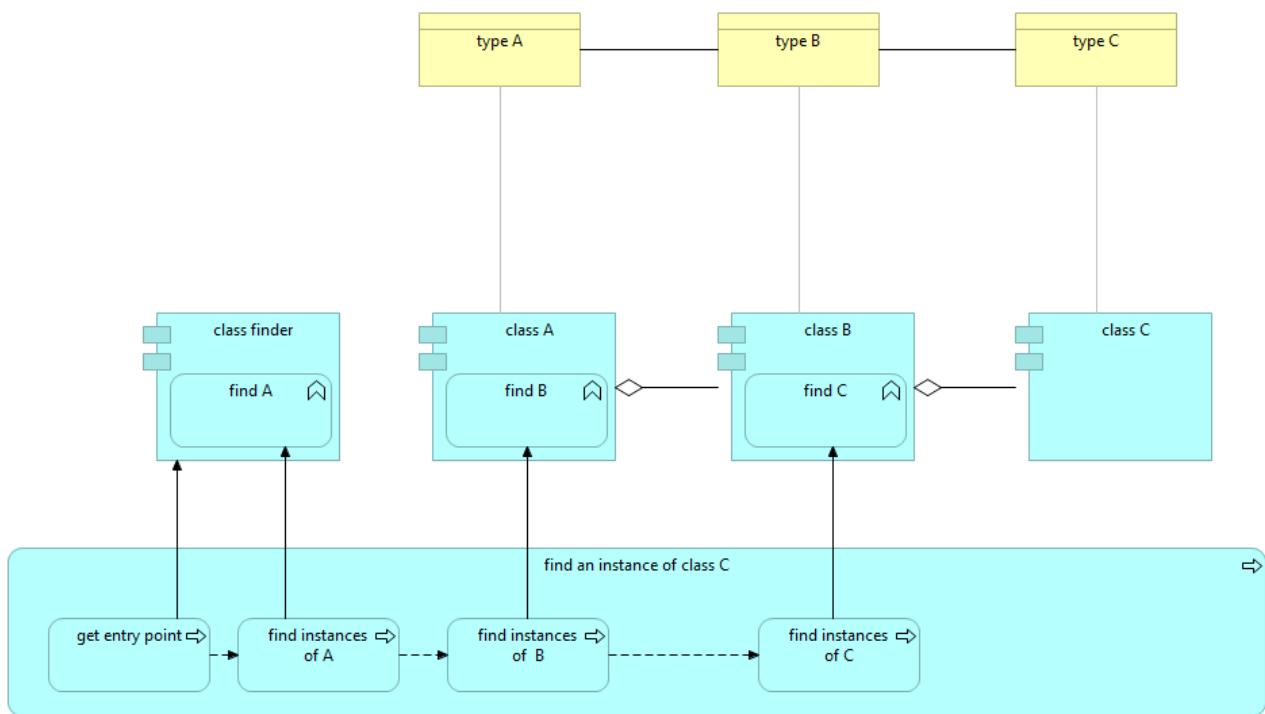
# OBJECT DESTRUCTION



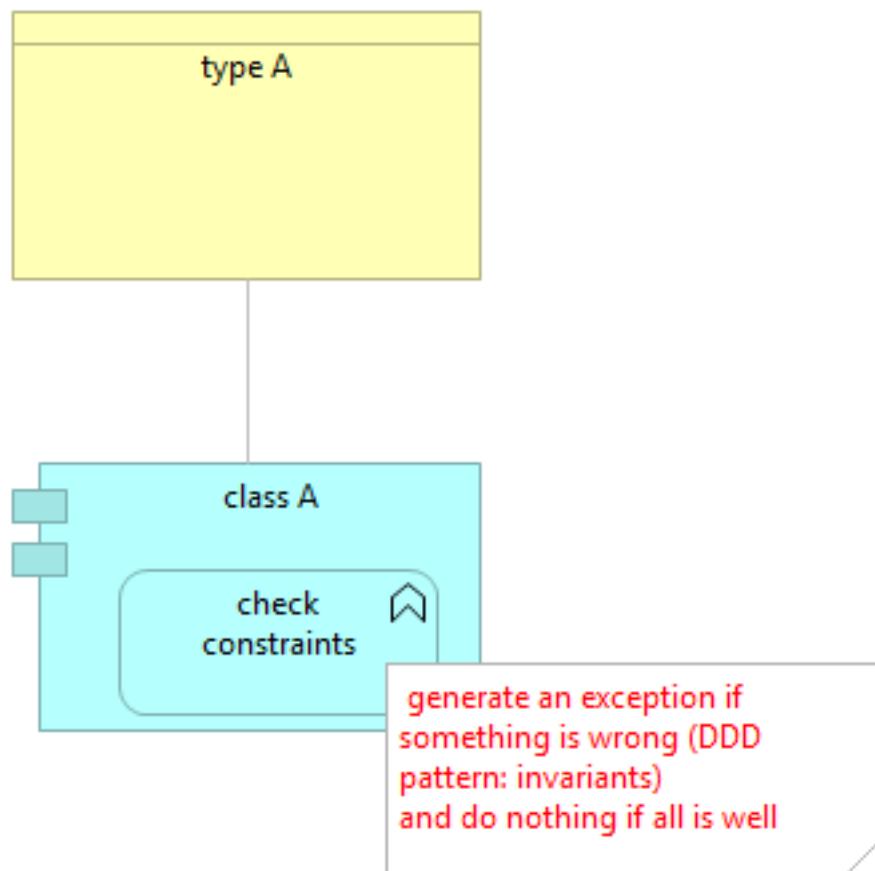
for example, delete object B  
- all constraints are met  
- no dangling references

object instance A

# ENTRY POINT.



# IMPLEMENTING CONSTRAINTS

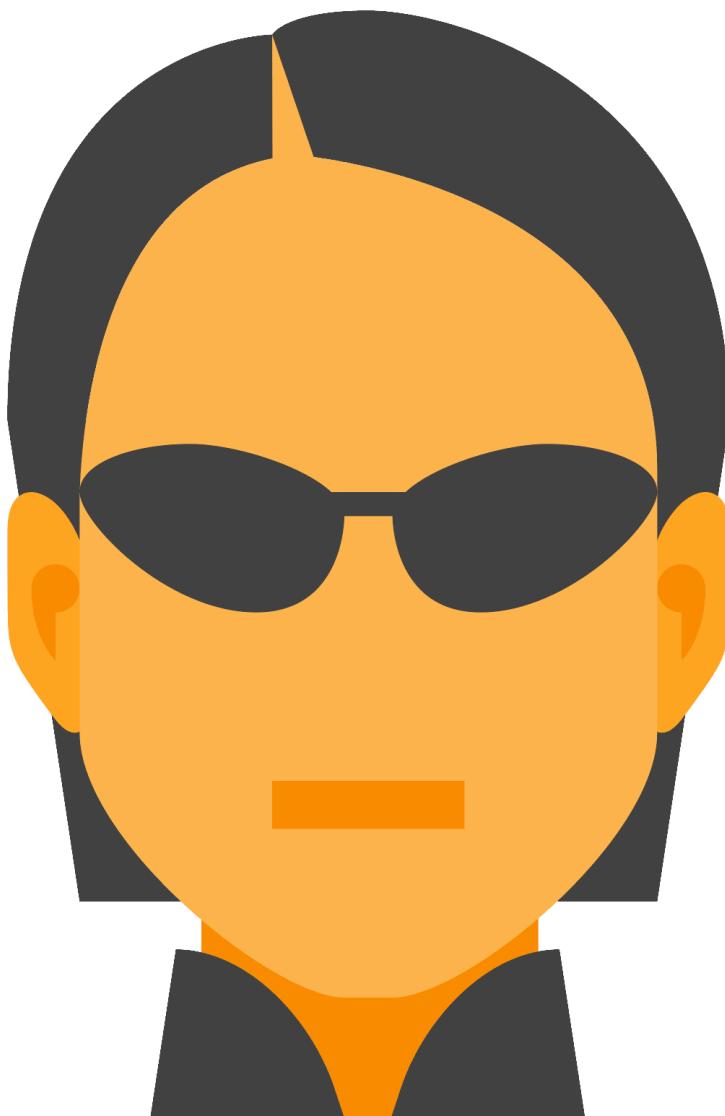


04

# Domain Driven Design

Tackling Complexity in the Heart  
of Software.

ERIC EVANS



# MODEL AND STRUCTURAL ELEMENTS

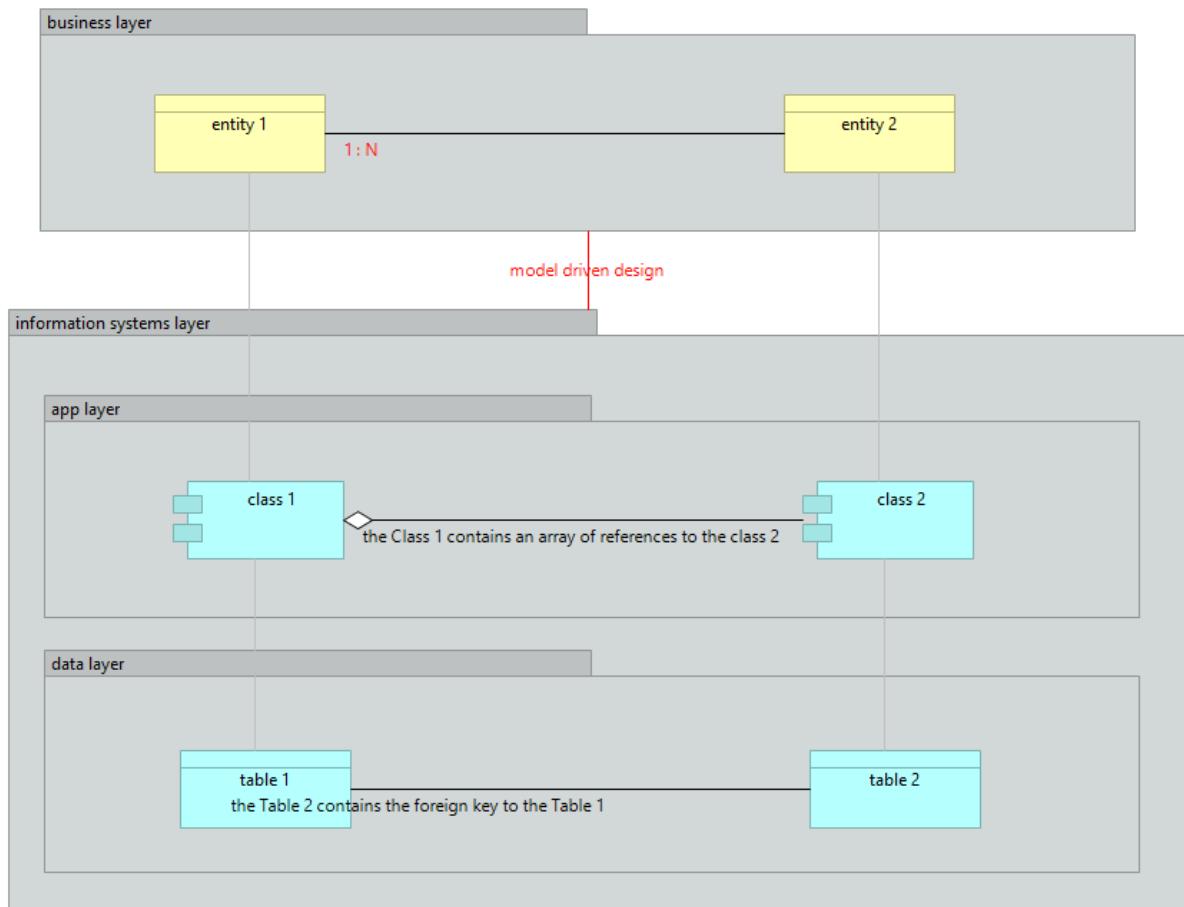
DOMAIN DRIVEN DESIGN

Eric Evans



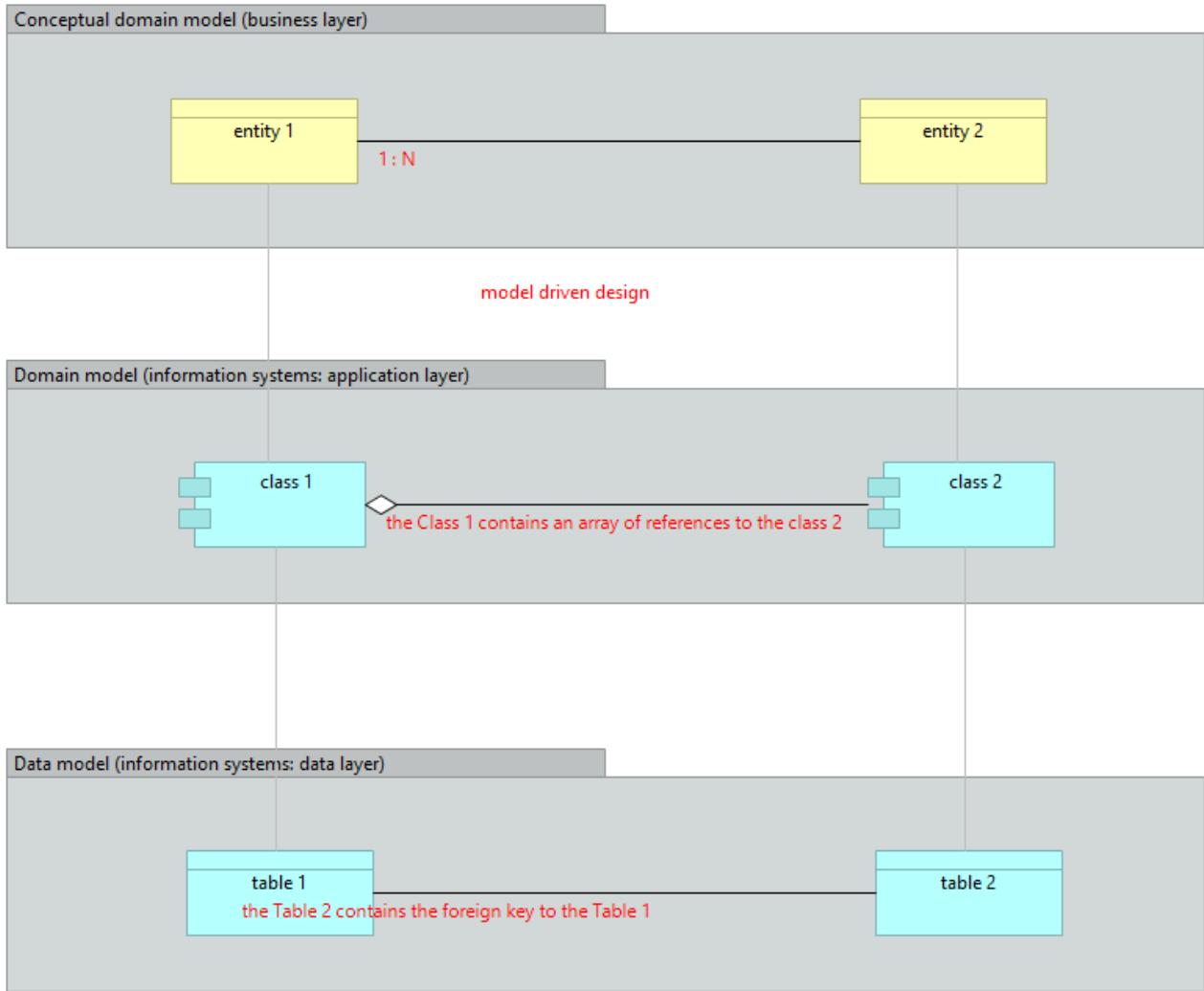
# MODEL-DRIVEN DESIGN

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.

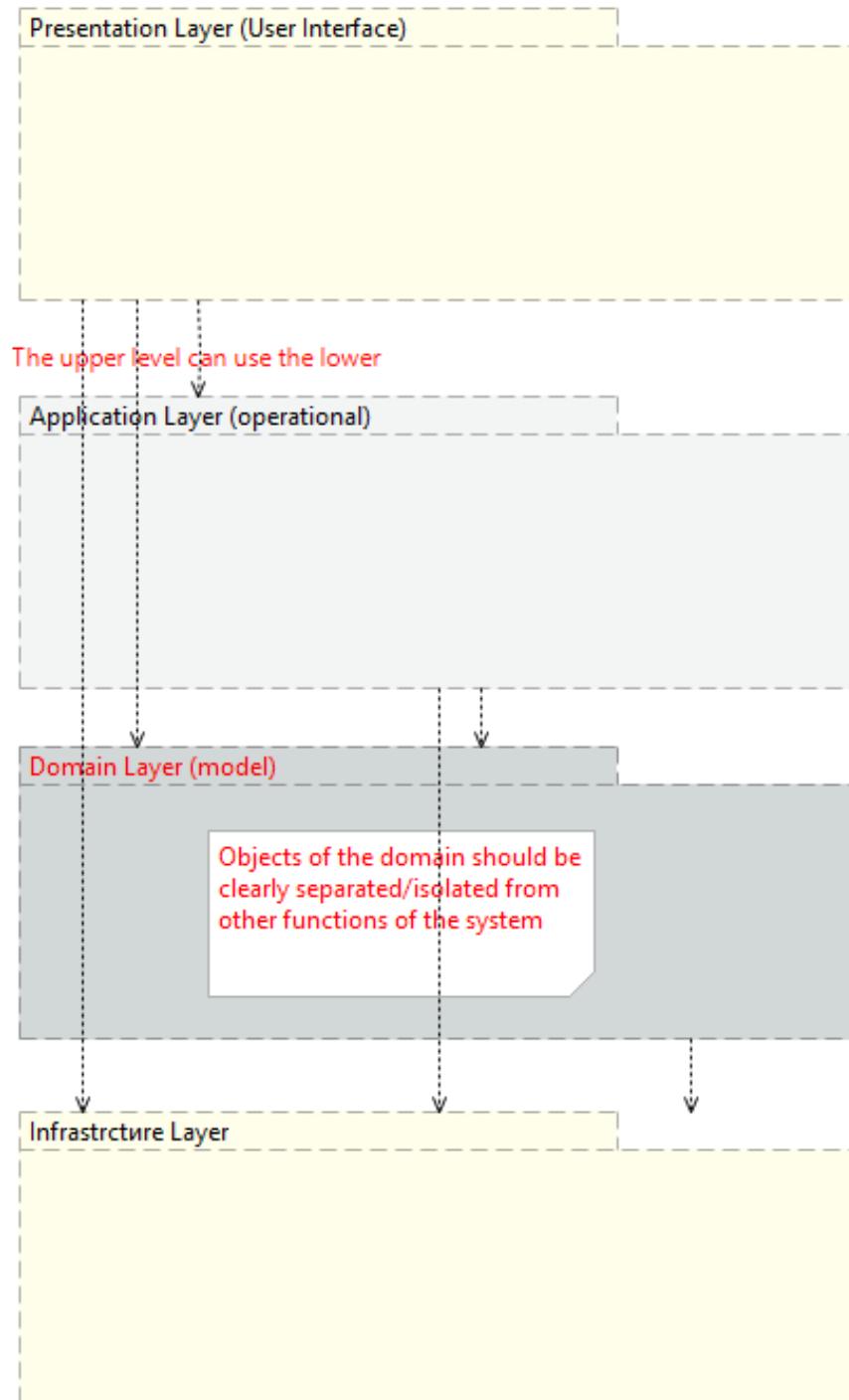
Domain model (information systems: application layer)

class 1

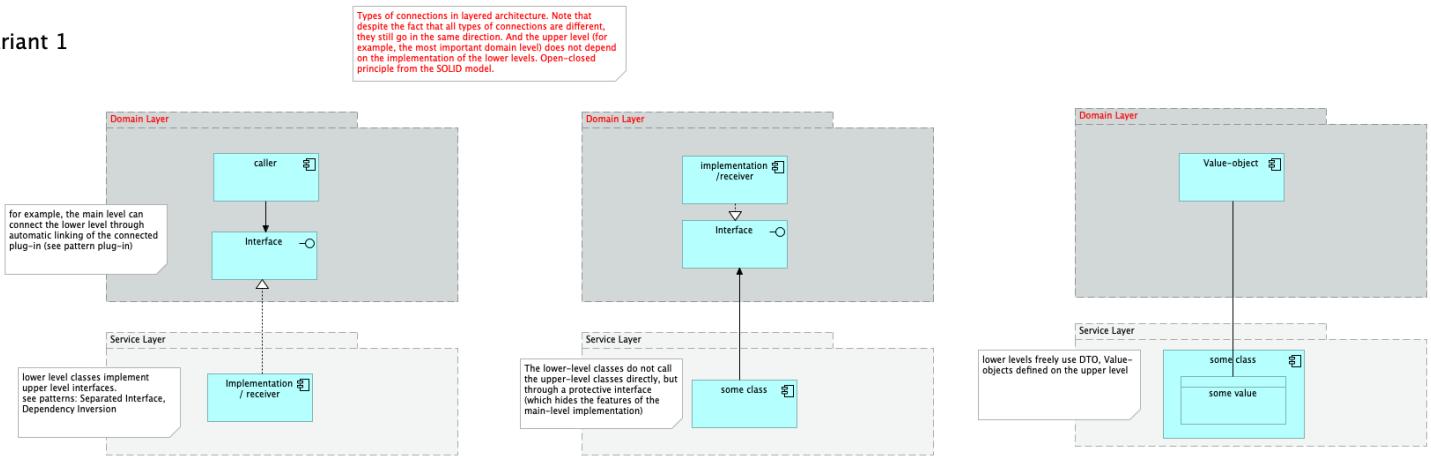
class 2

# LAYERED ARCHITECTURE (ASYMMETRIC )

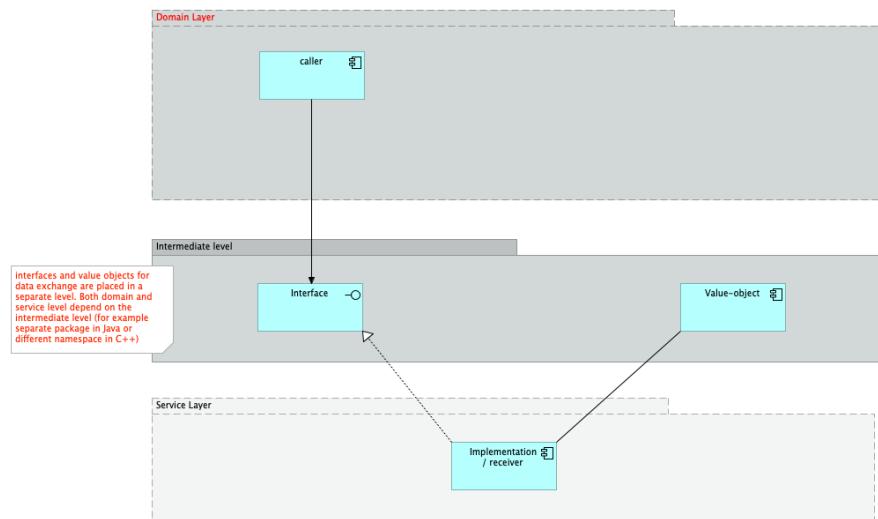
layered architecture  
Only domain layer is required



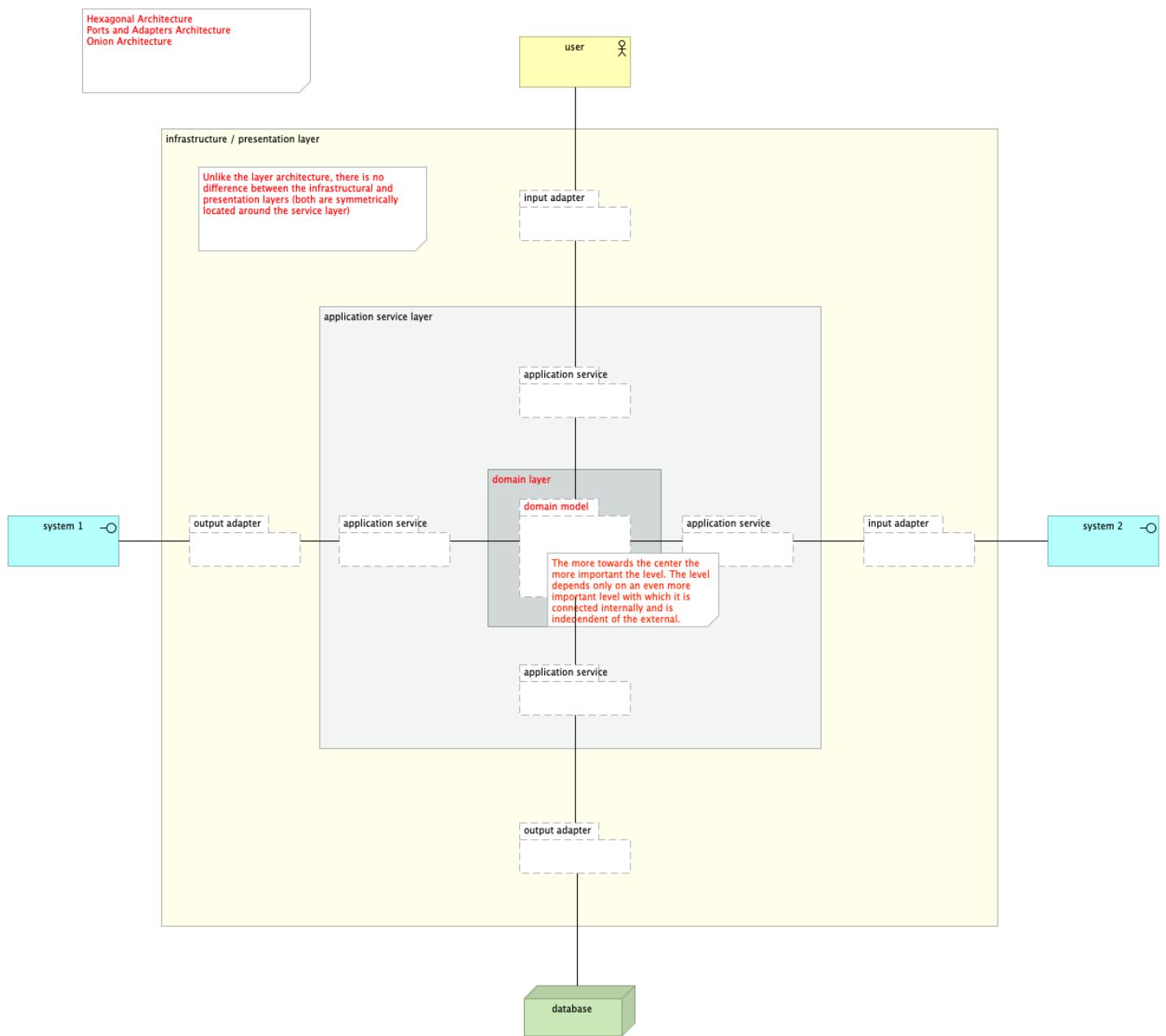
## variant 1

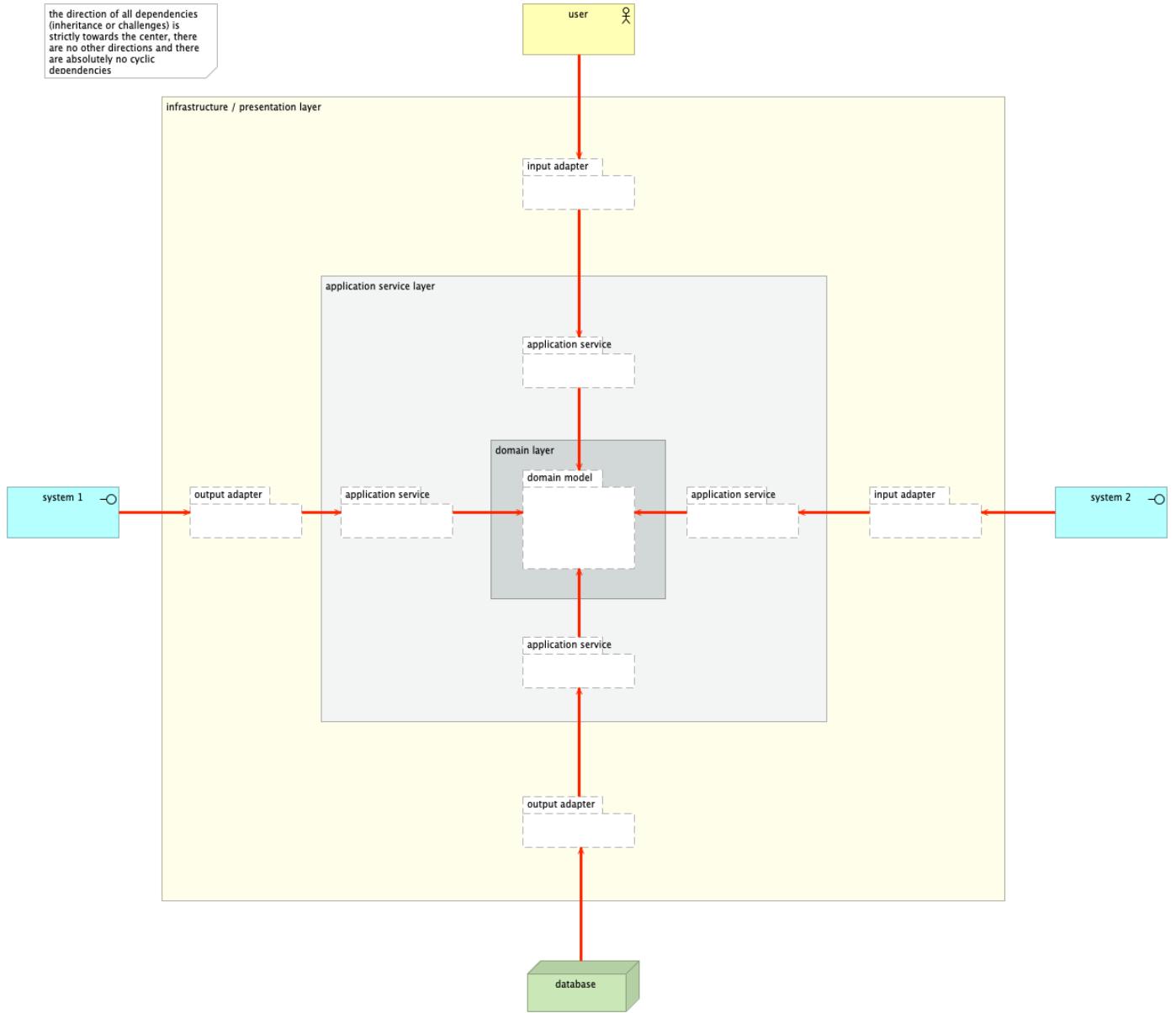


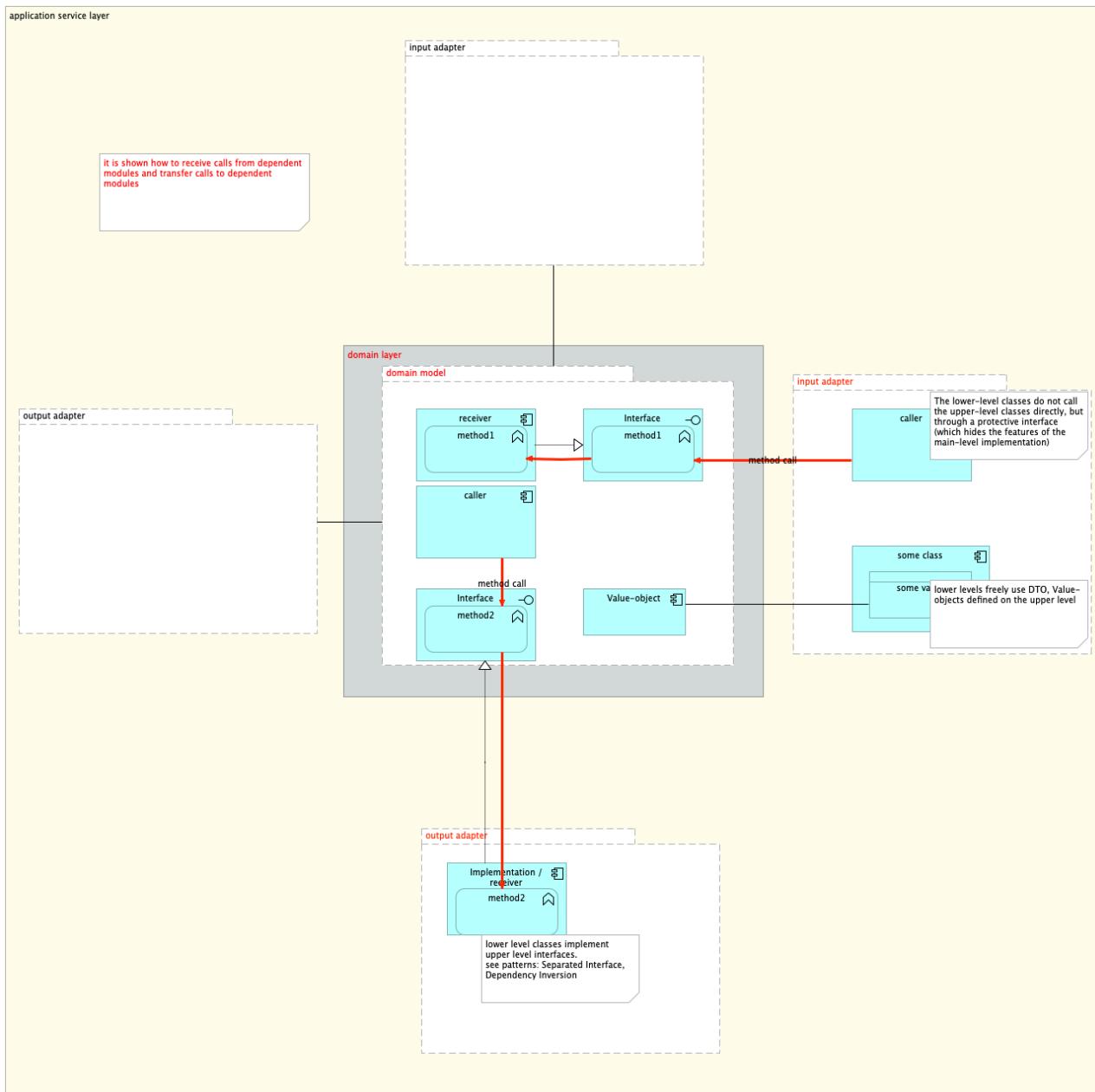
## variant 2



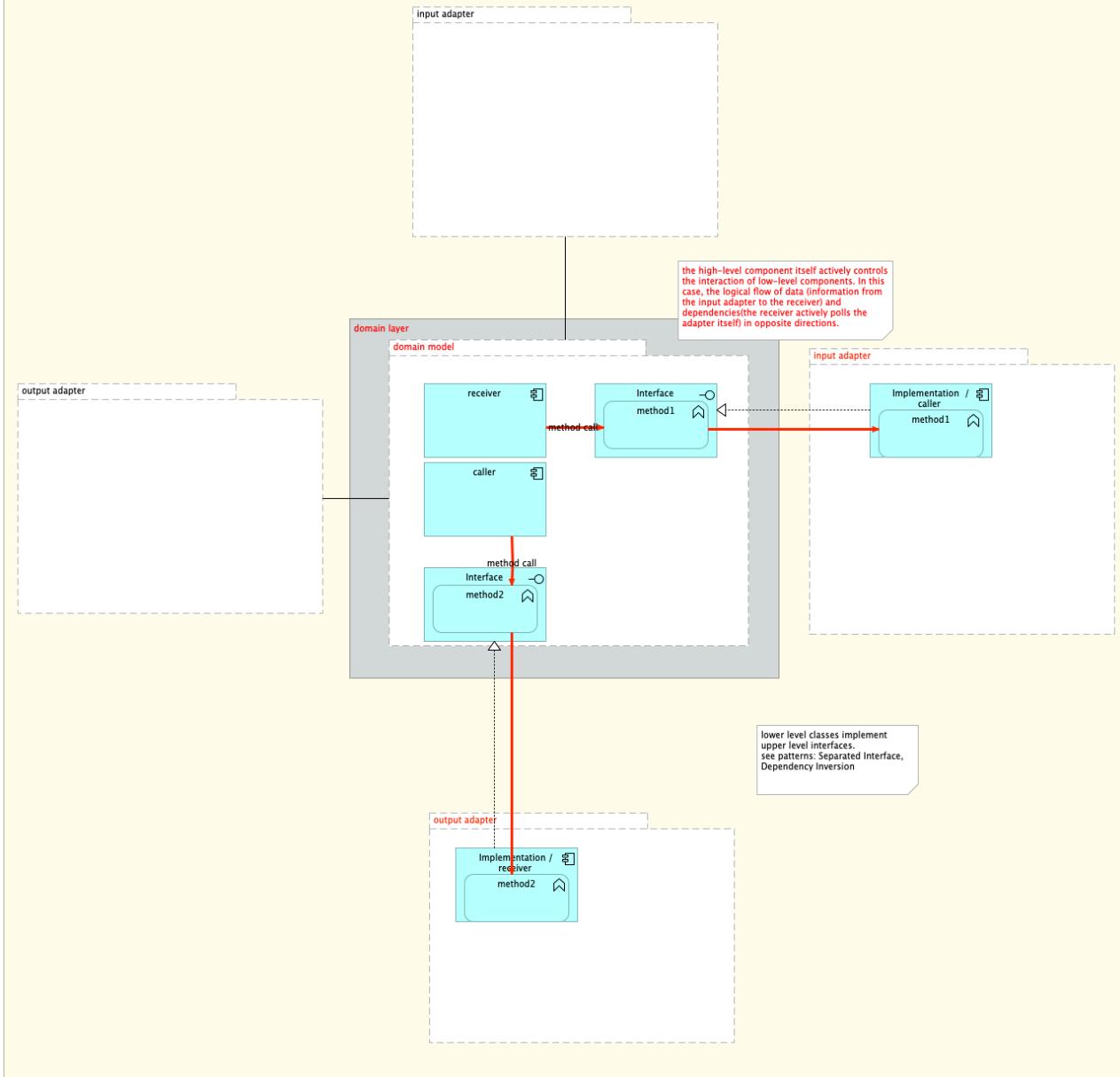
# HEXAGONAL ARCHITECTURE (SYMMETRIC)



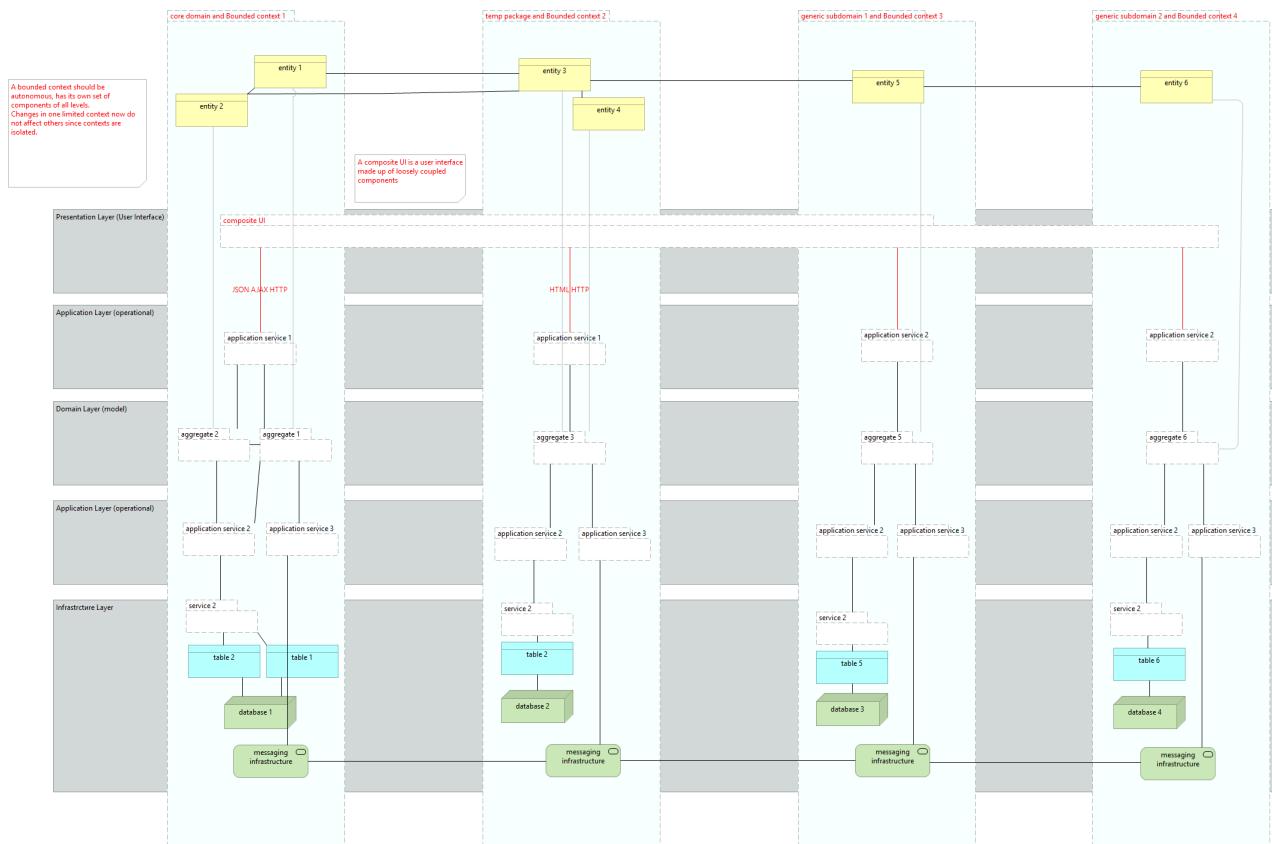




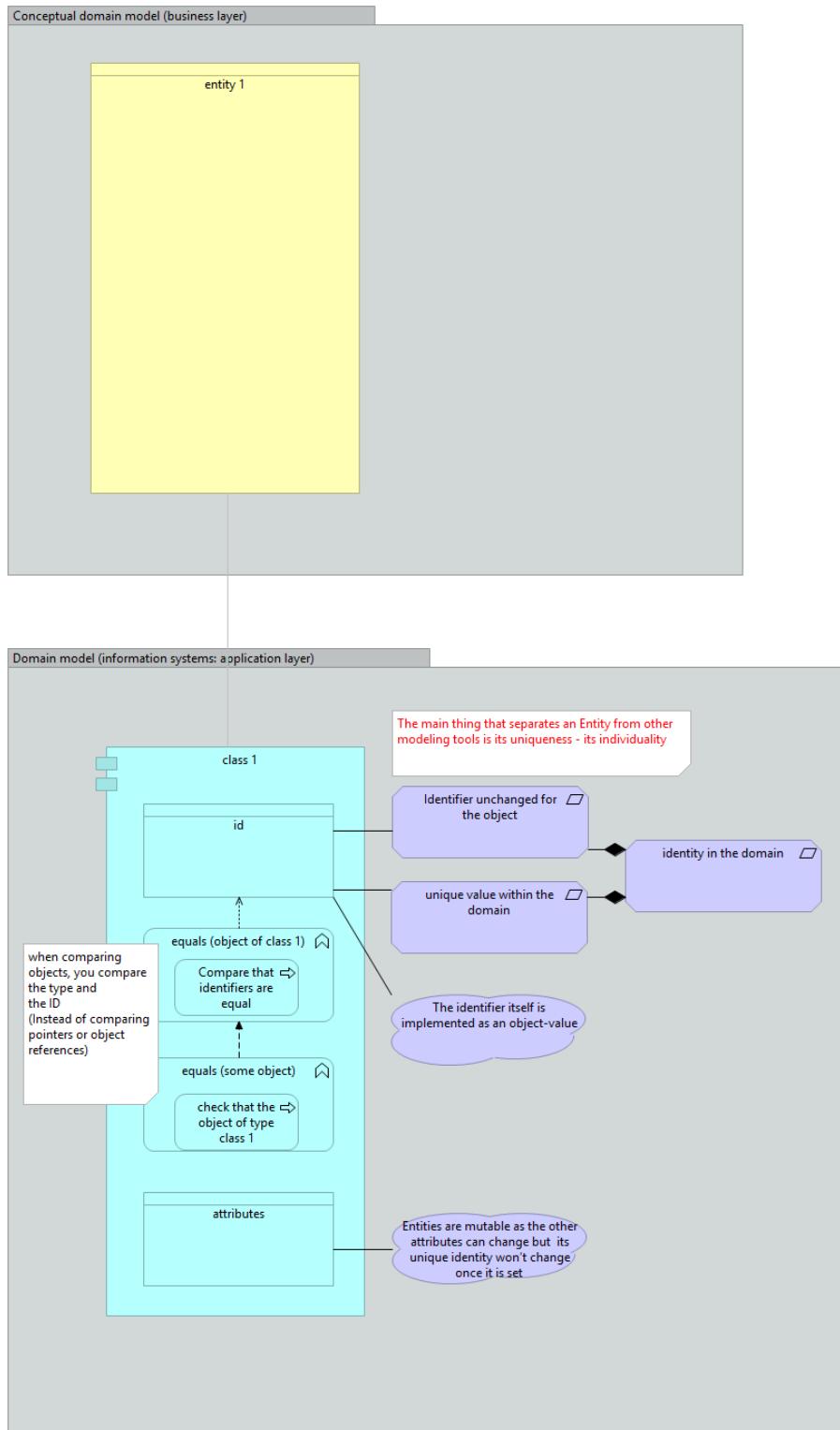
application service layer

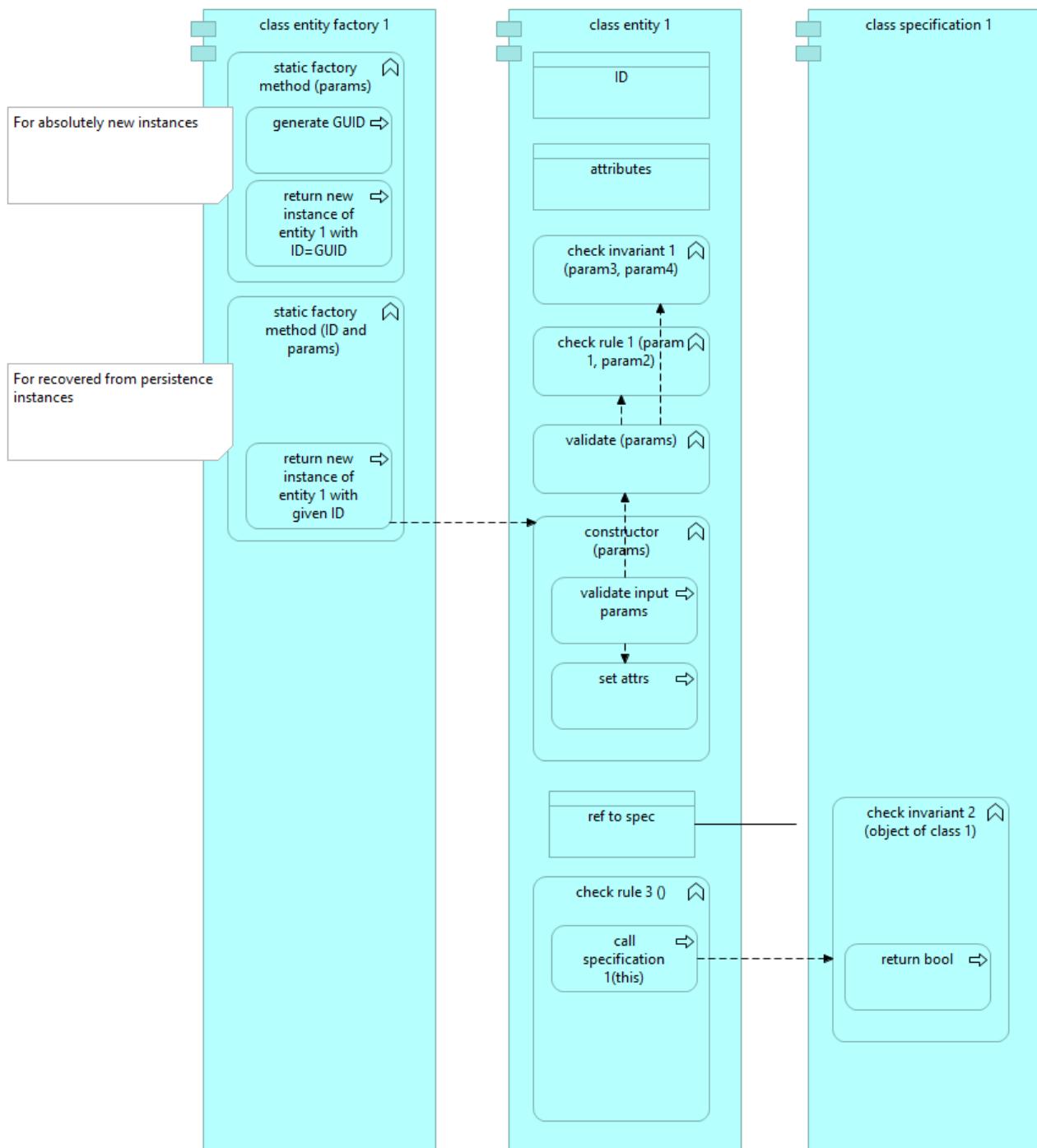


# COMPOSITE UI

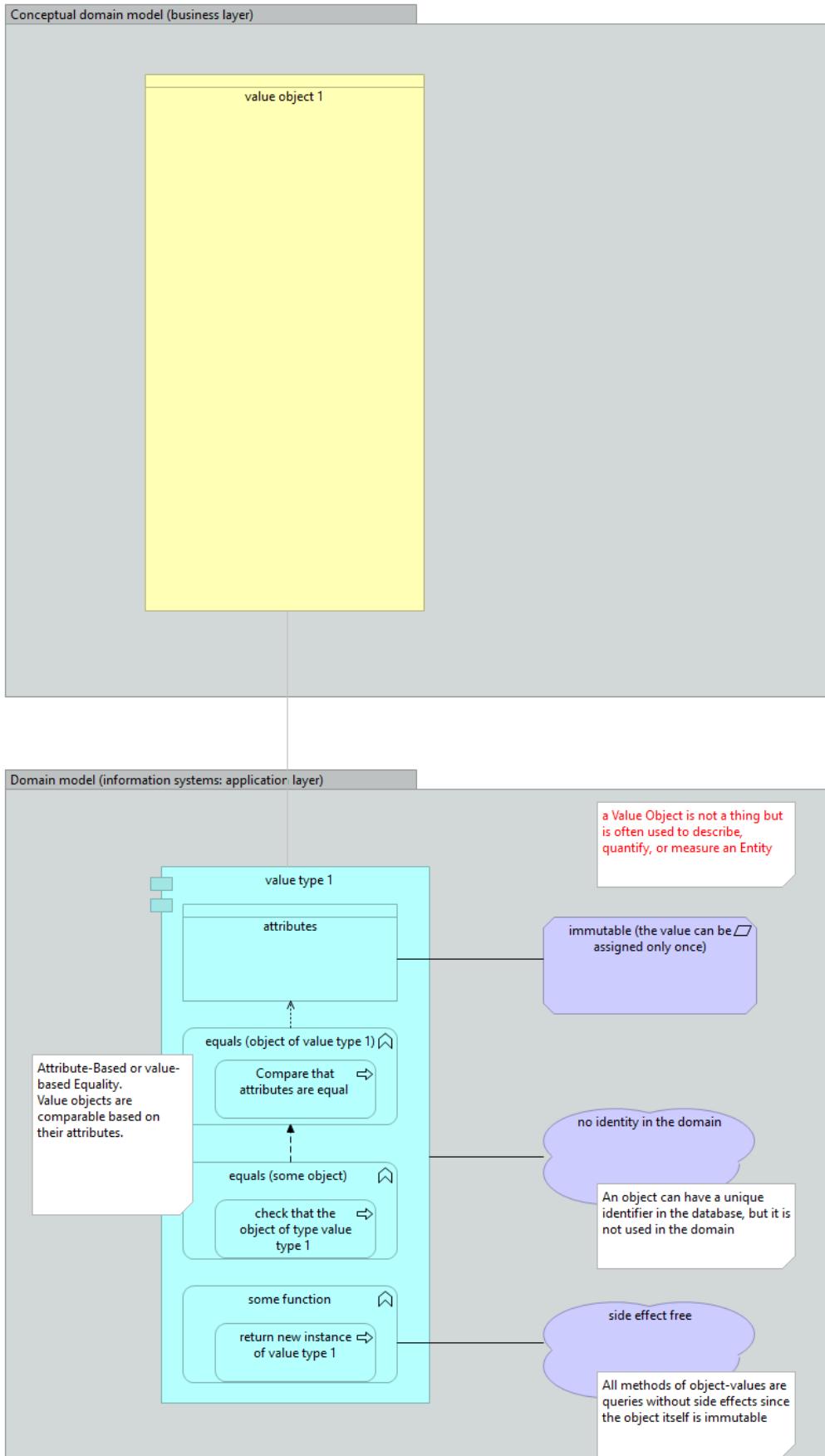


# ENTITIES

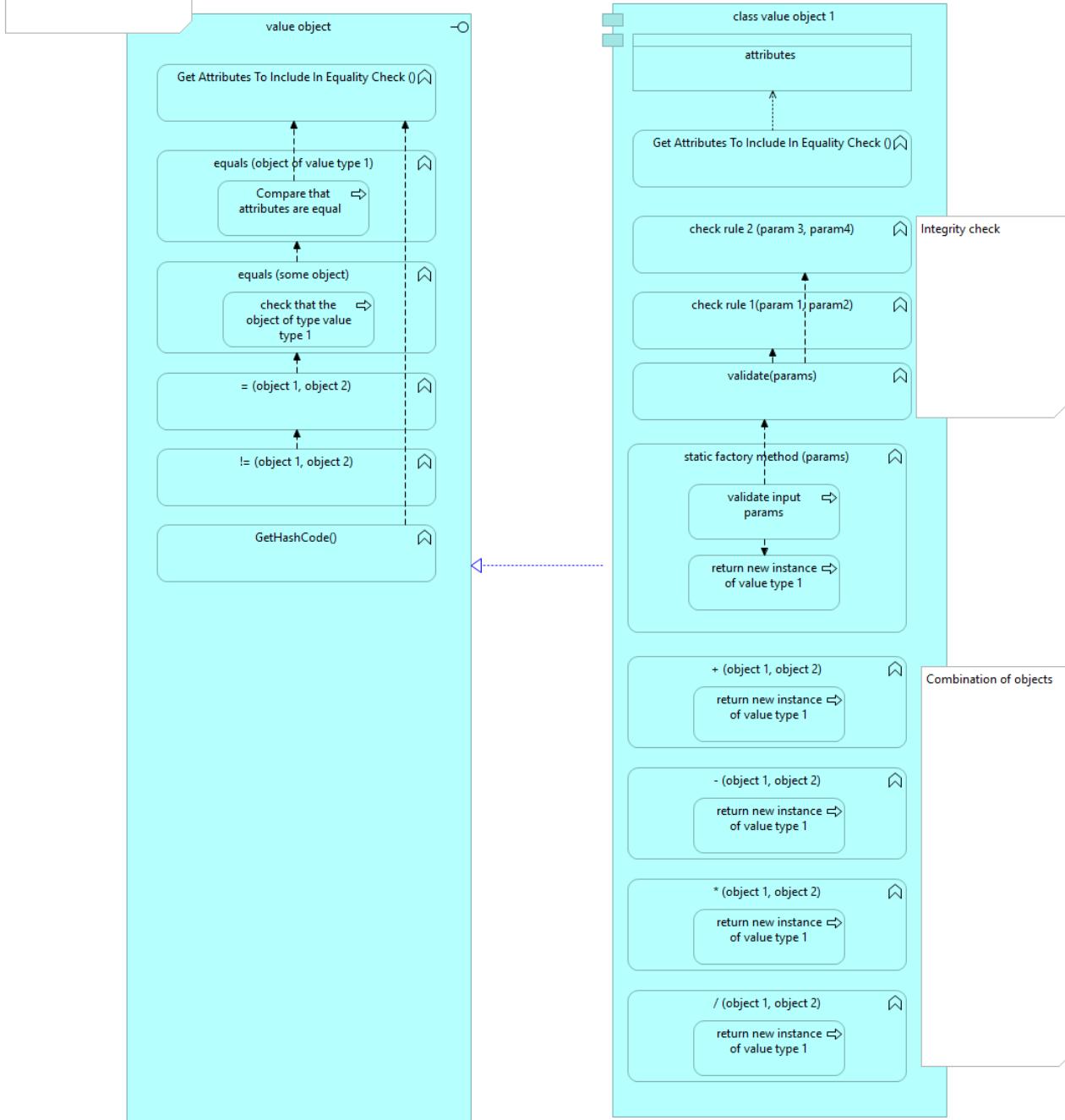




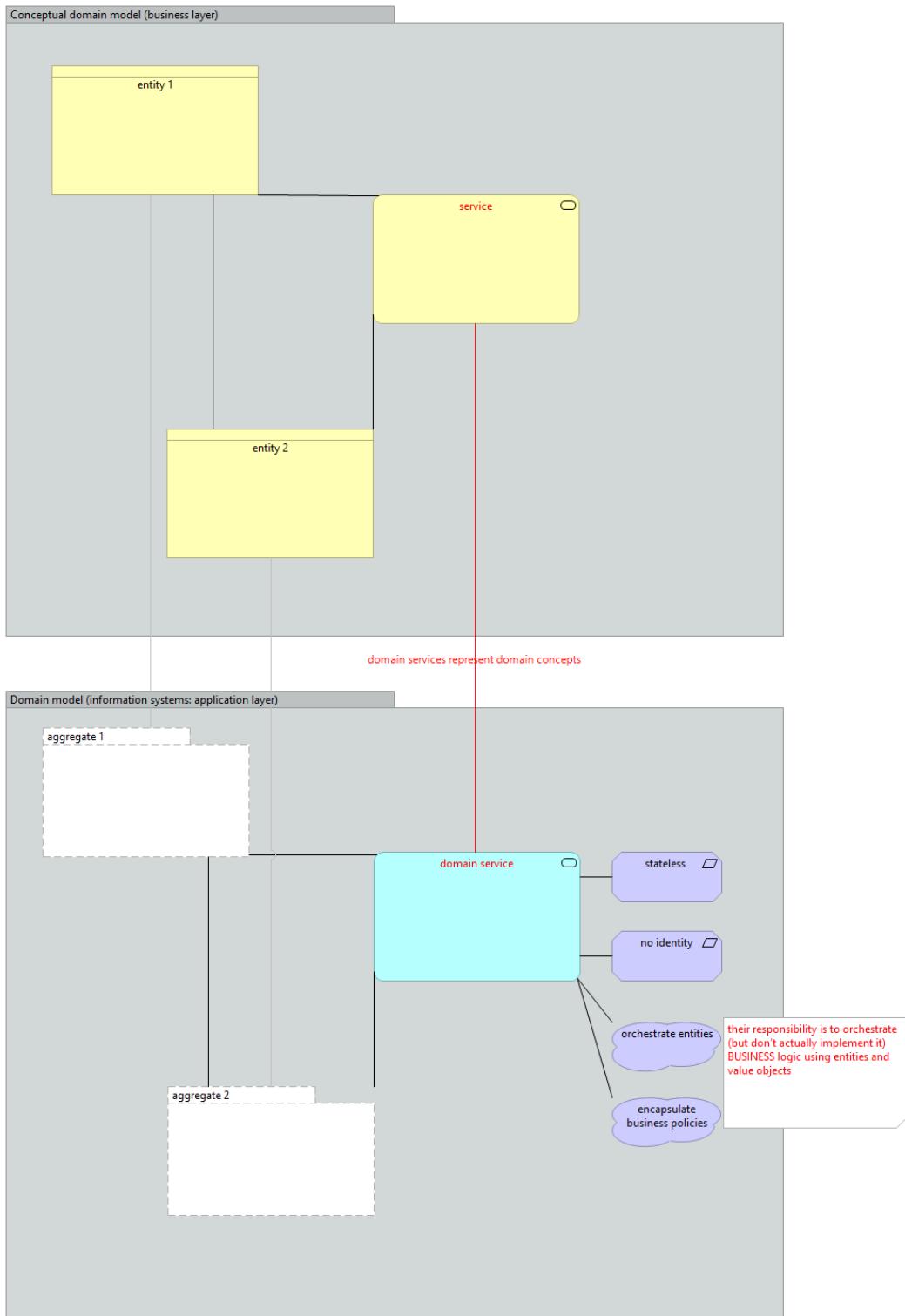
# VALUE-OBJECTS



See example: Fowler's Money

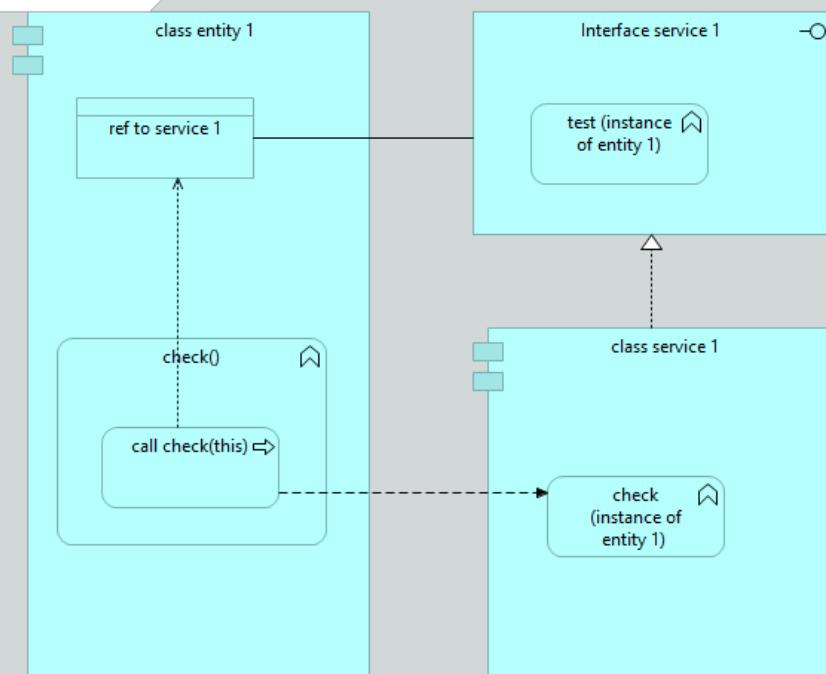


# DOMAIN SERVICES

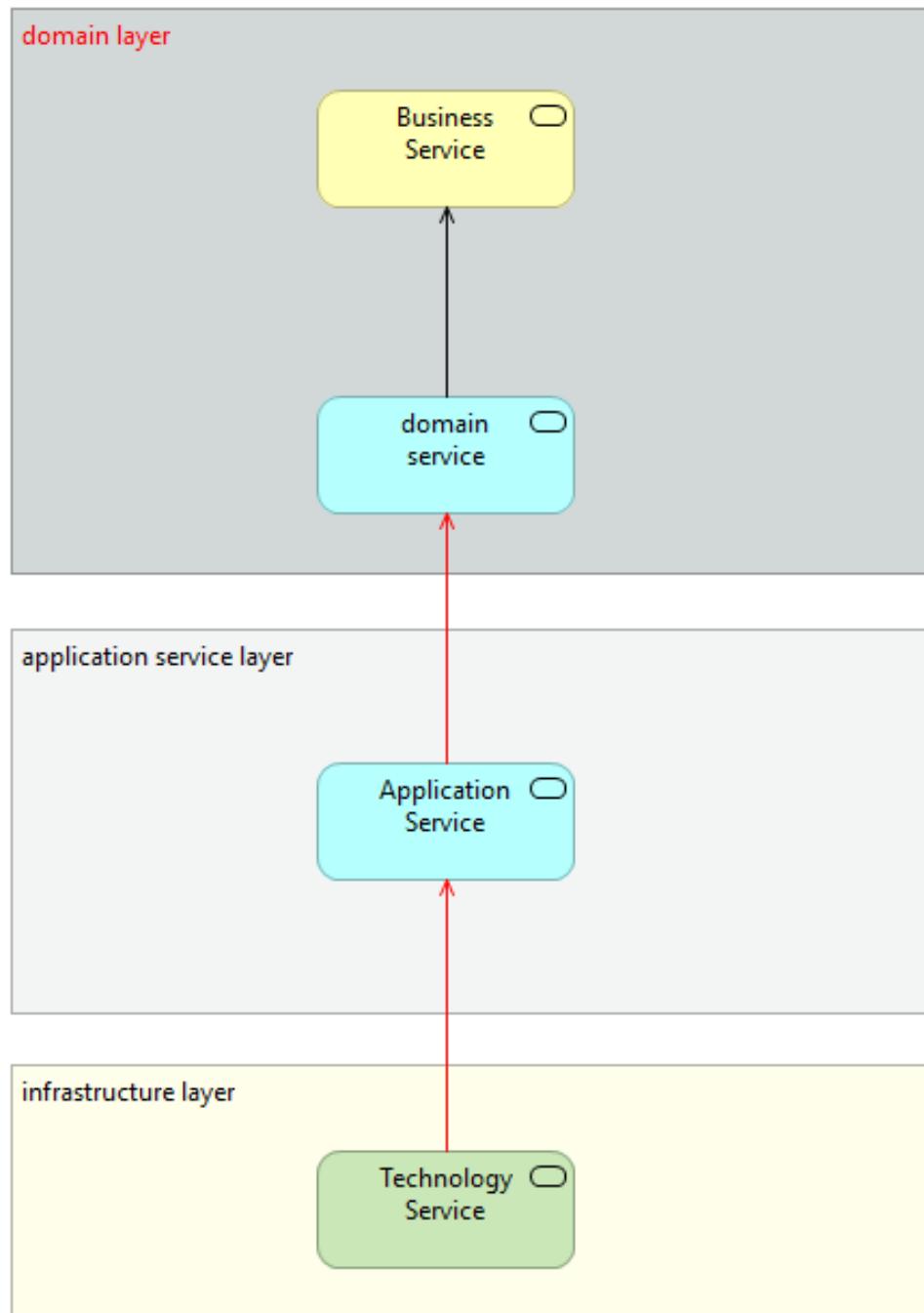


Domain model (information systems: application layer)

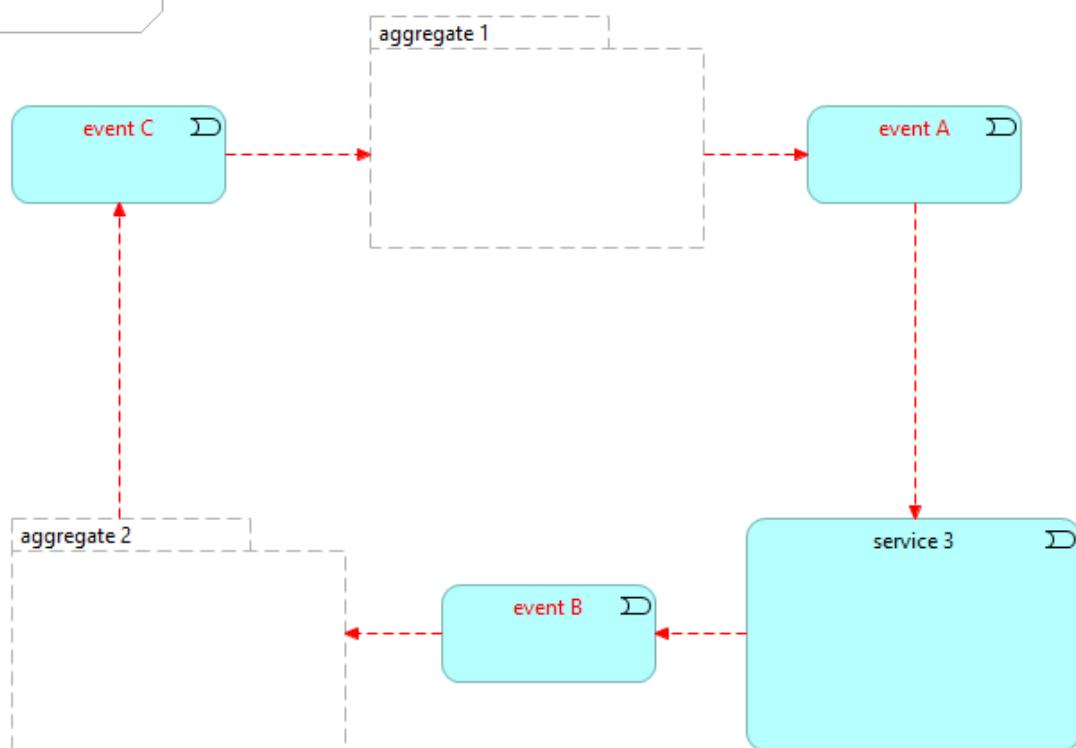
Example of communicating services with aggregates via direct service call

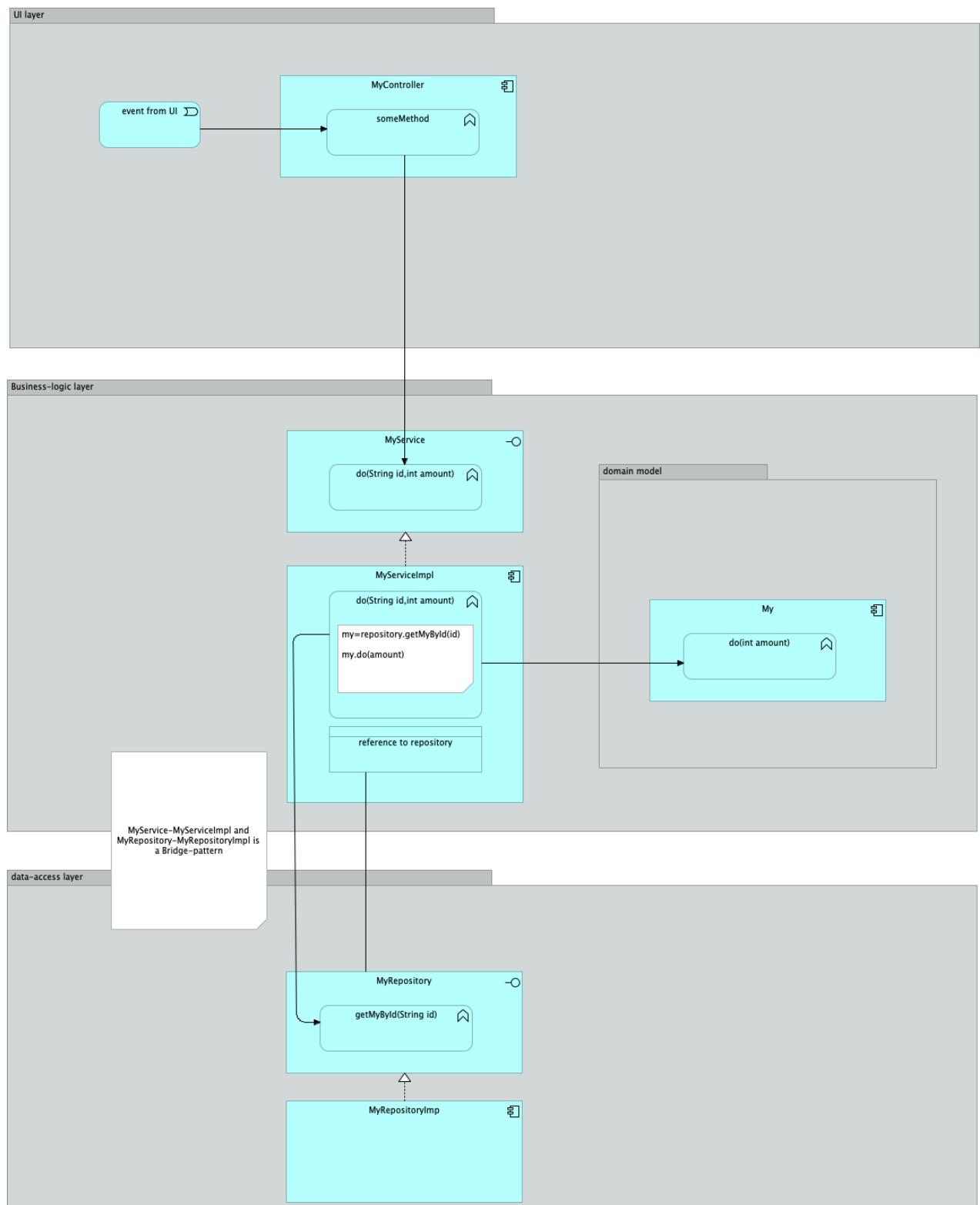


Services support each other

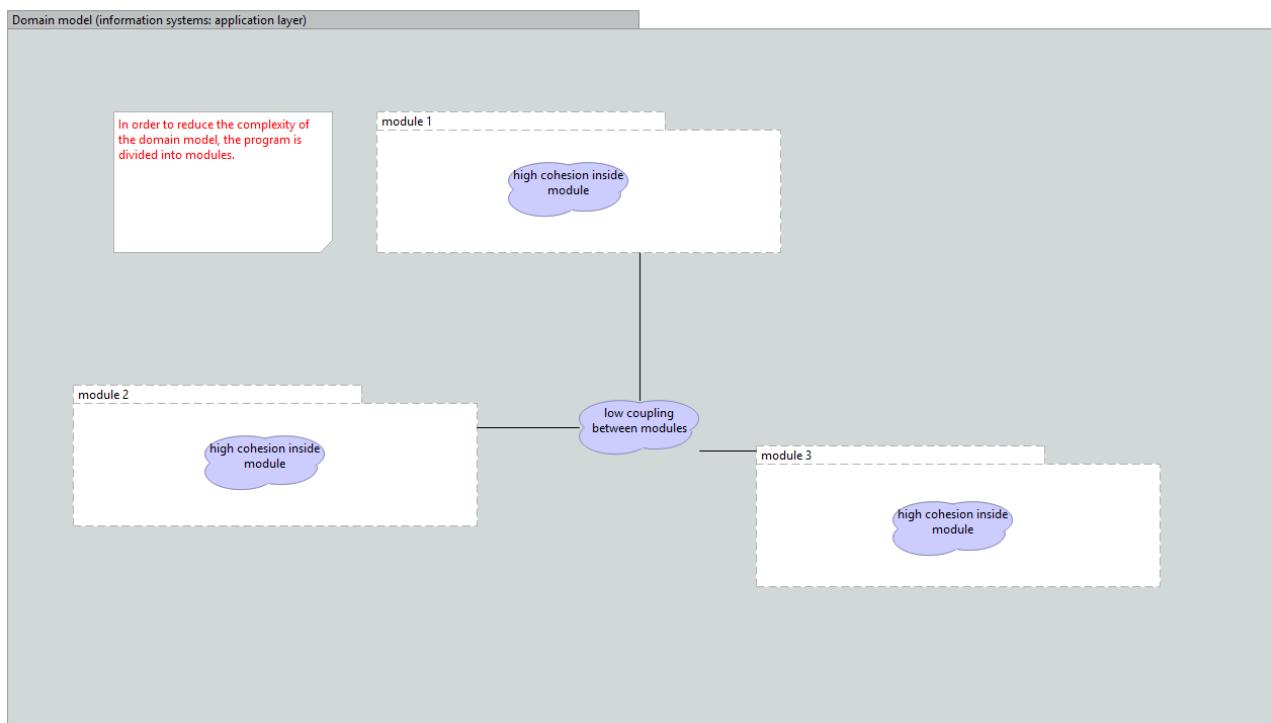


Example of communicating services with aggregates through events

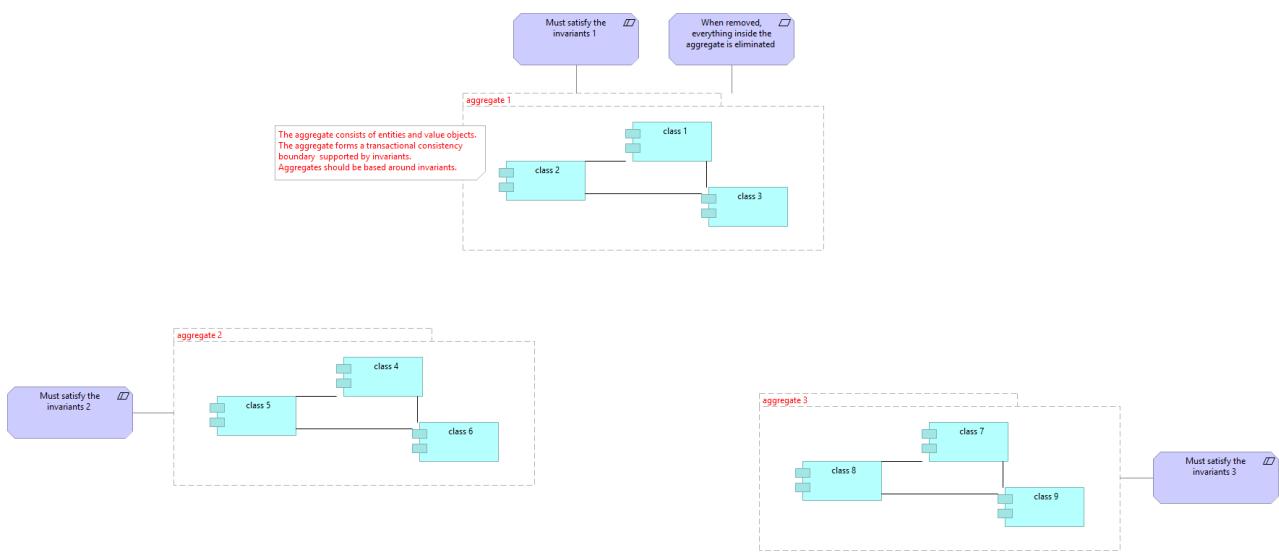




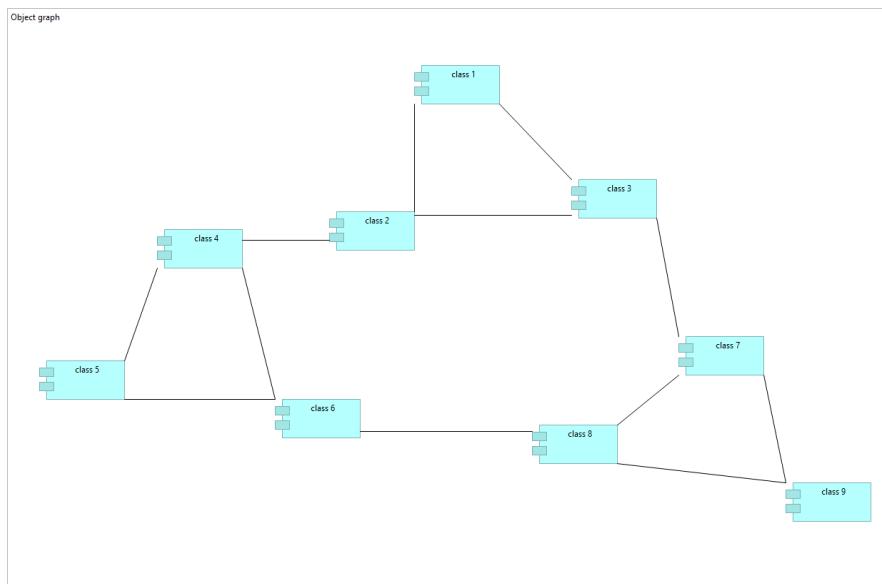
# MODULES



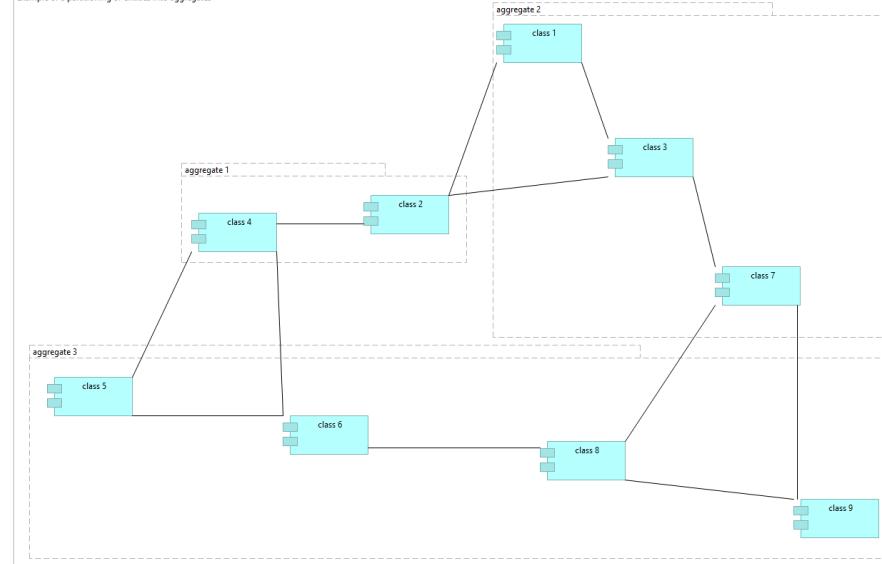
# AGGREGATES



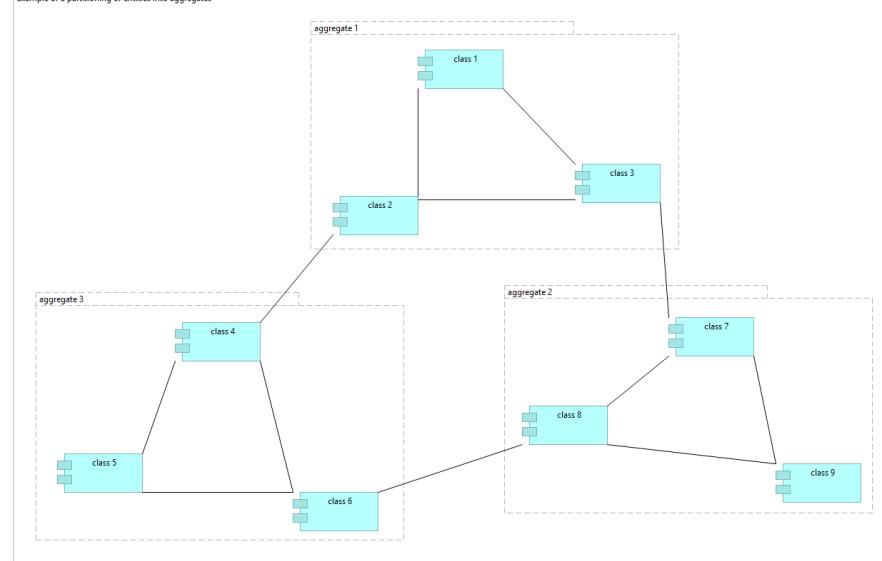
In the course of modeling, think about different ways of splitting the model into aggregates



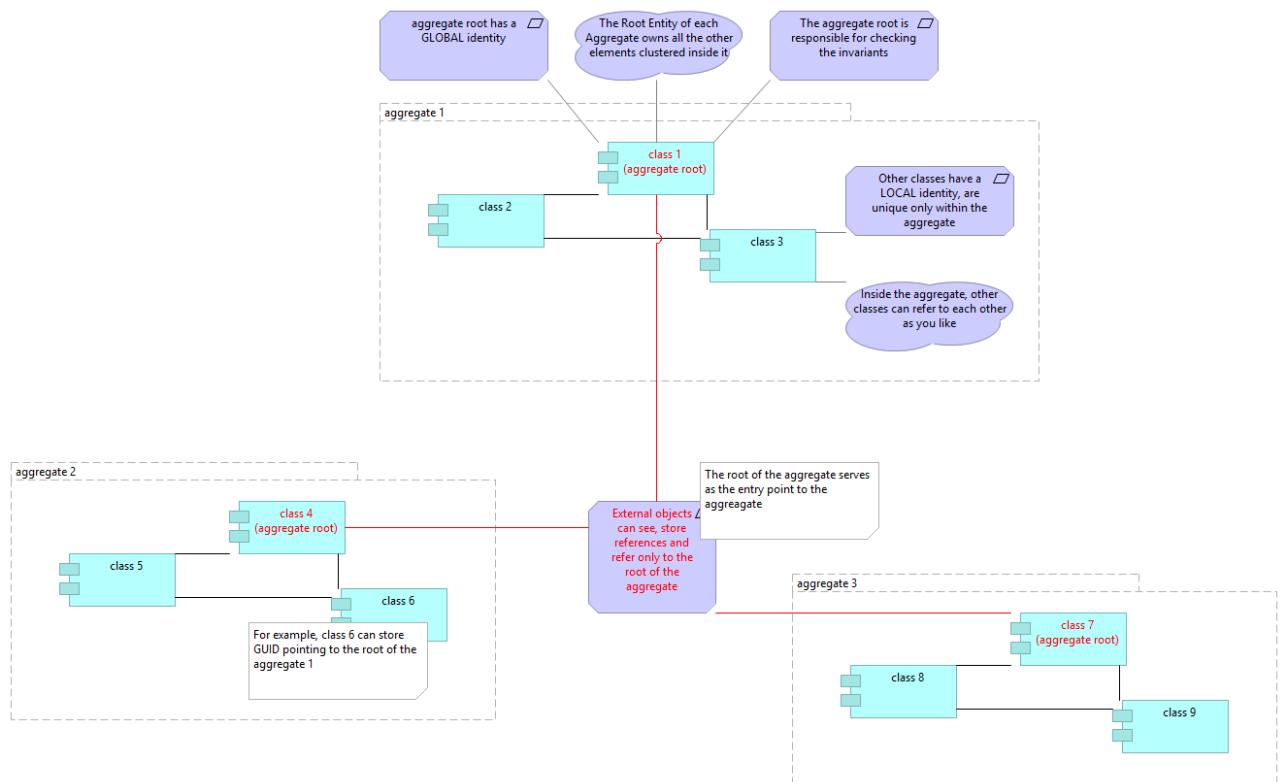
Example of a partitioning of entities into aggregates



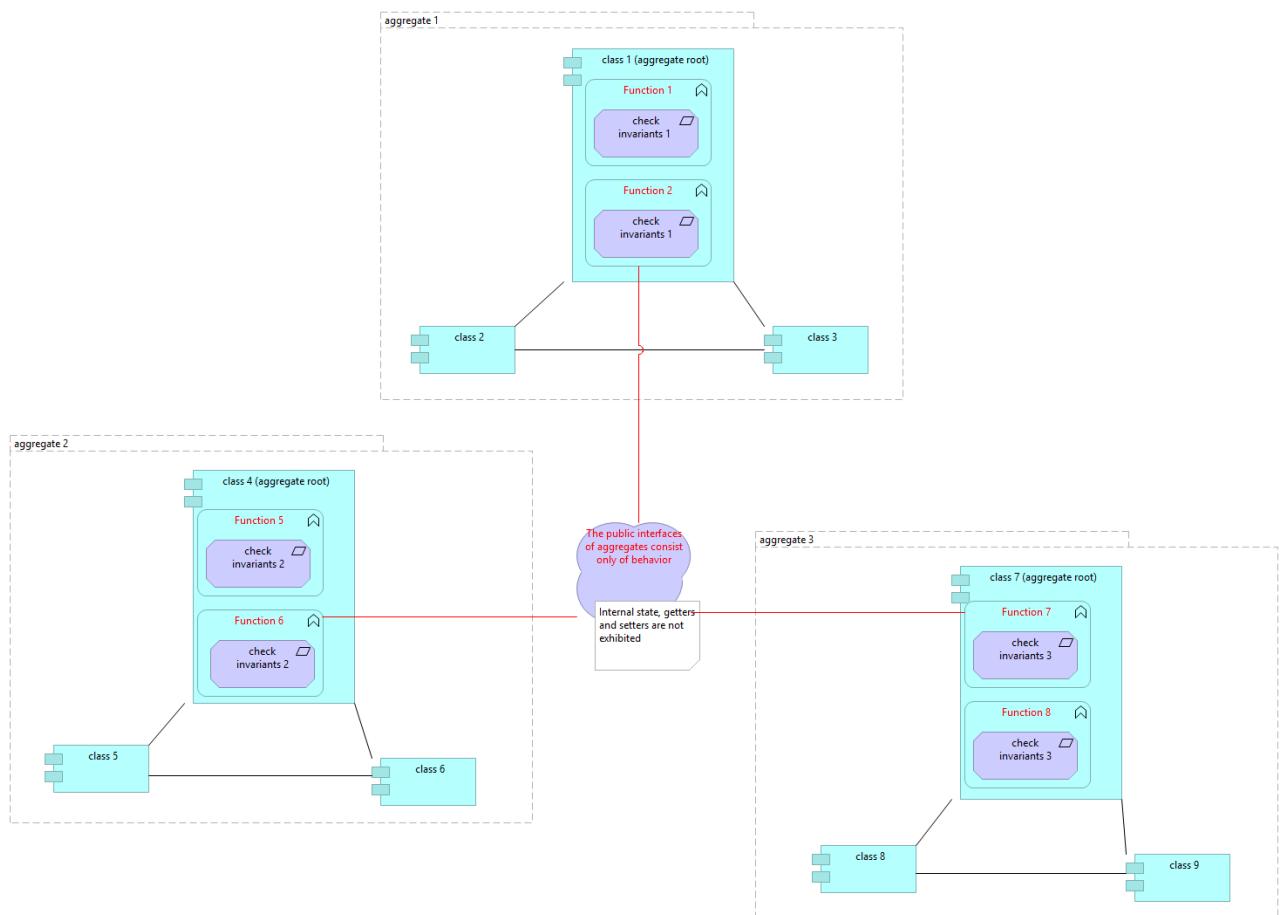
Example of a partitioning of entities into aggregates



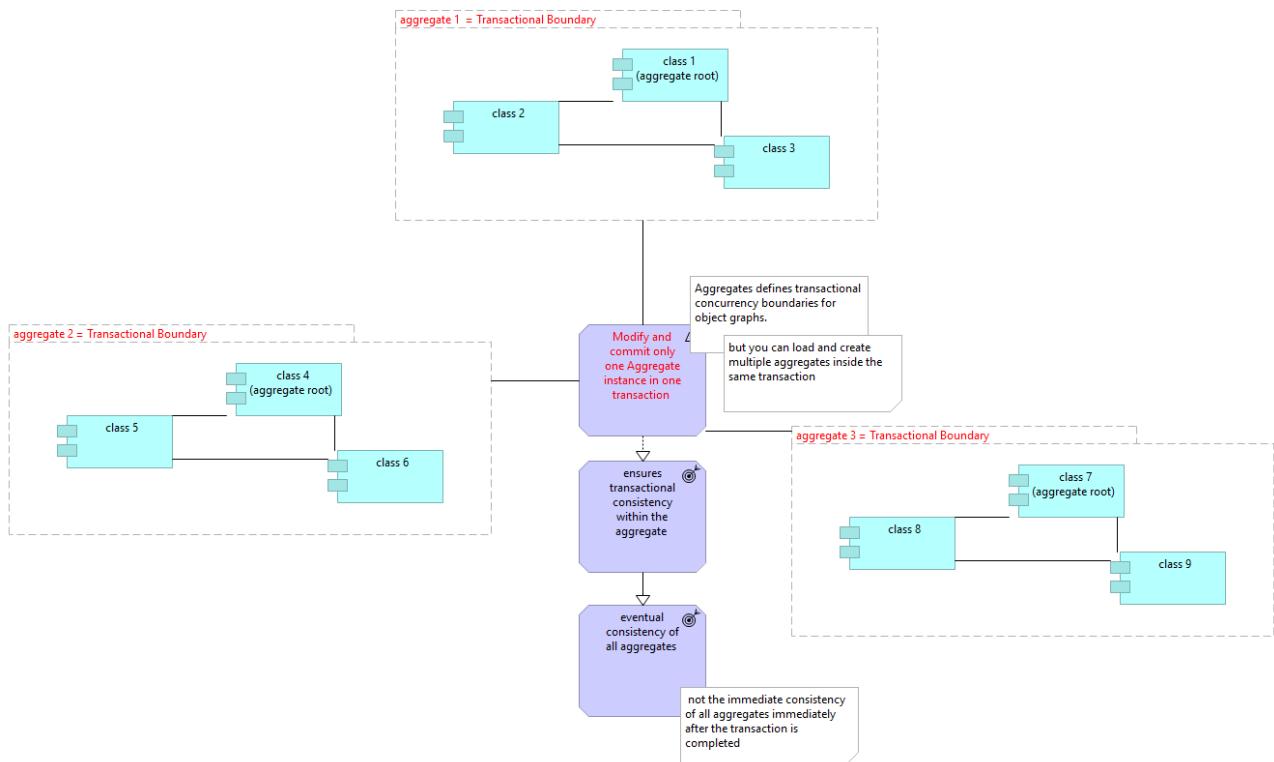
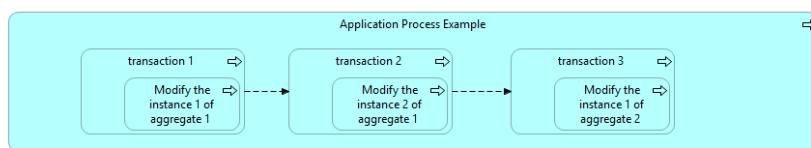
# AGGREGATE ROOT



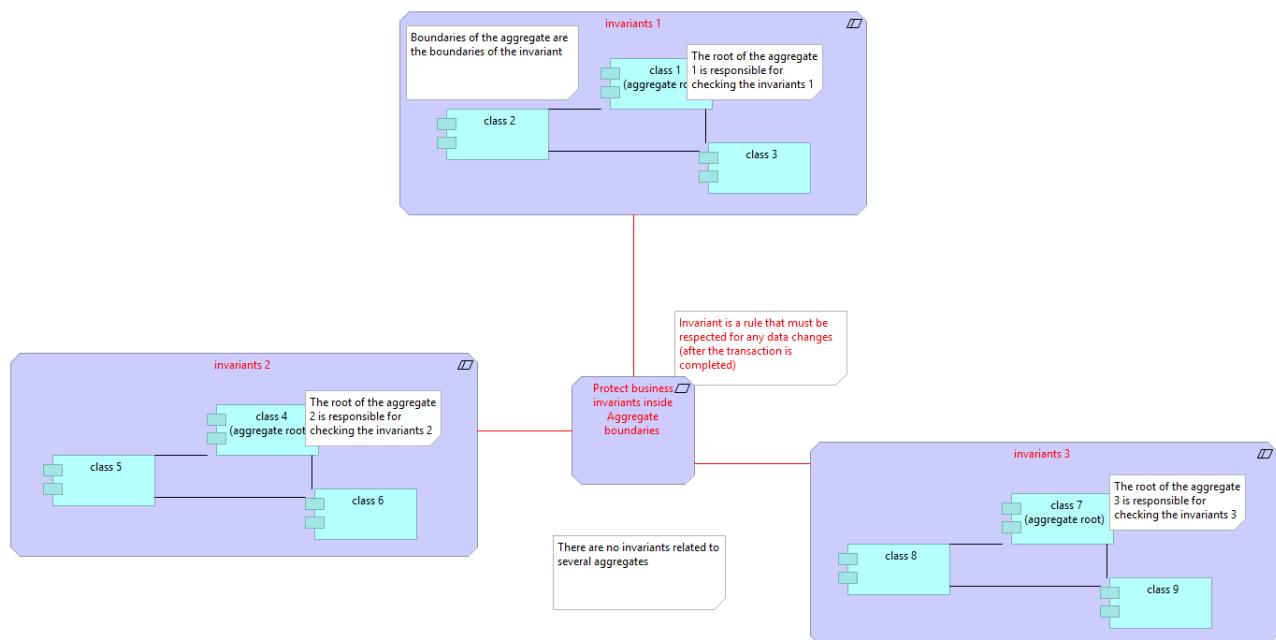
# BEHAVIOR-FOCUSED AGGREGATE ROOT



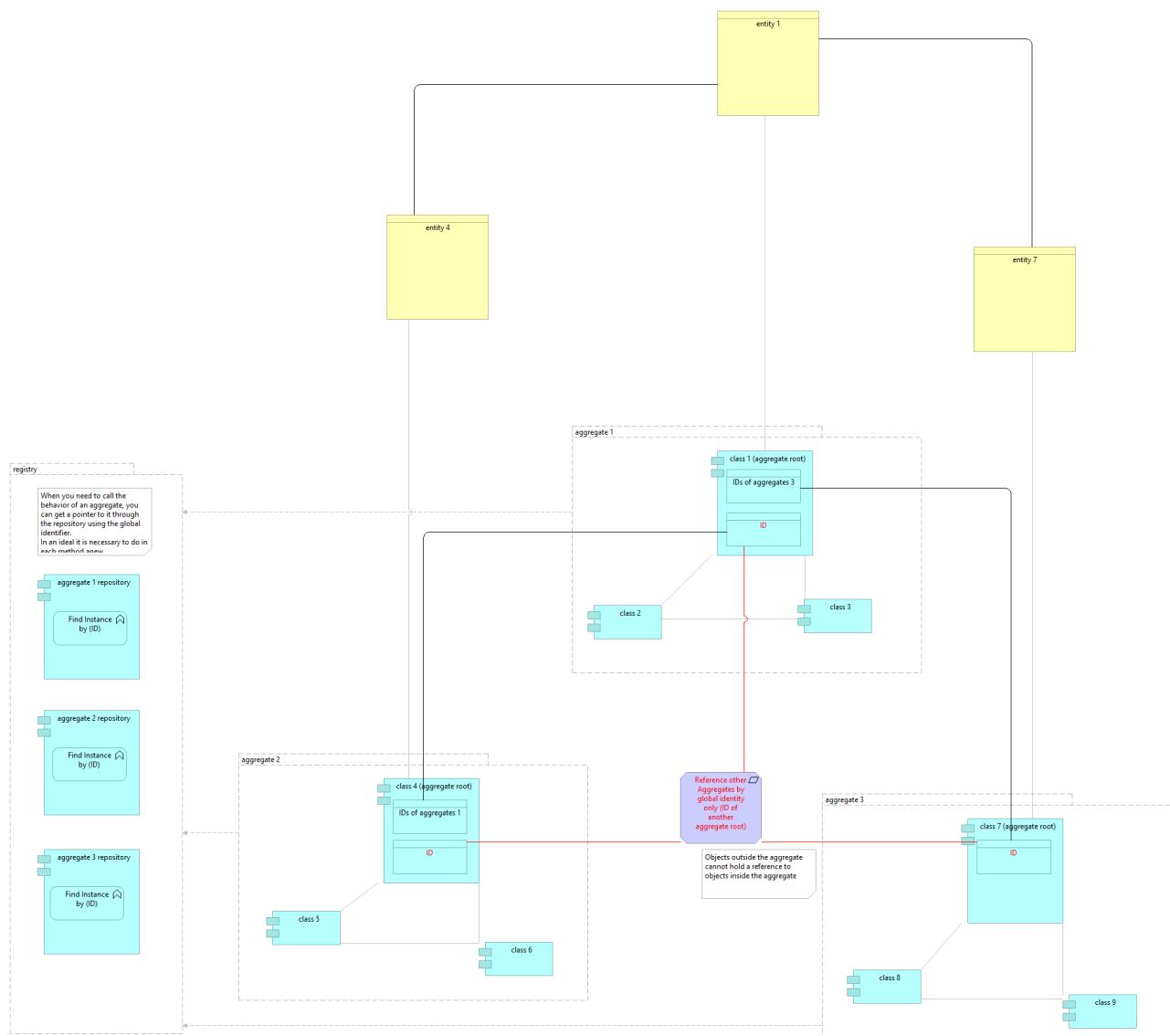
# MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION



# PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES



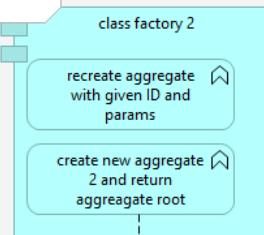
# REFERENCE OTHER AGGREGATES BY IDENTITY ONLY



# FACTORIES

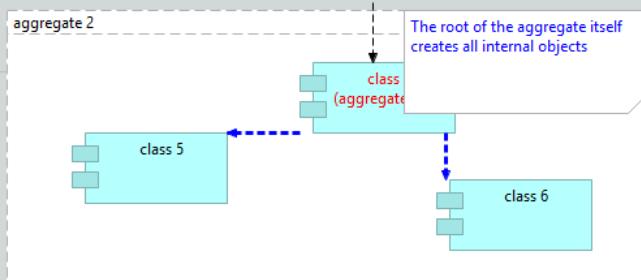
application service layer

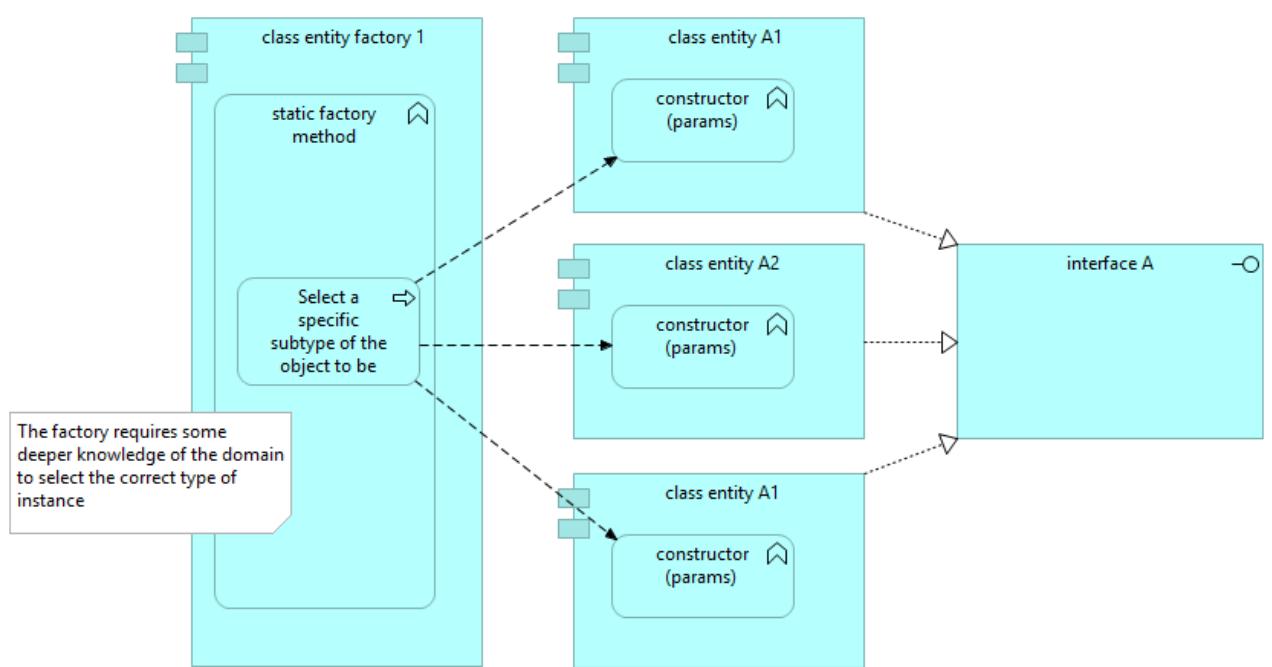
Use factory to create aggregates, complex entities and value objects.  
Use factory to re-create domain objects from persistent storage.  
The factory creates an aggregate entirely, with the satisfaction of all invariants  
GoF pattern: factory



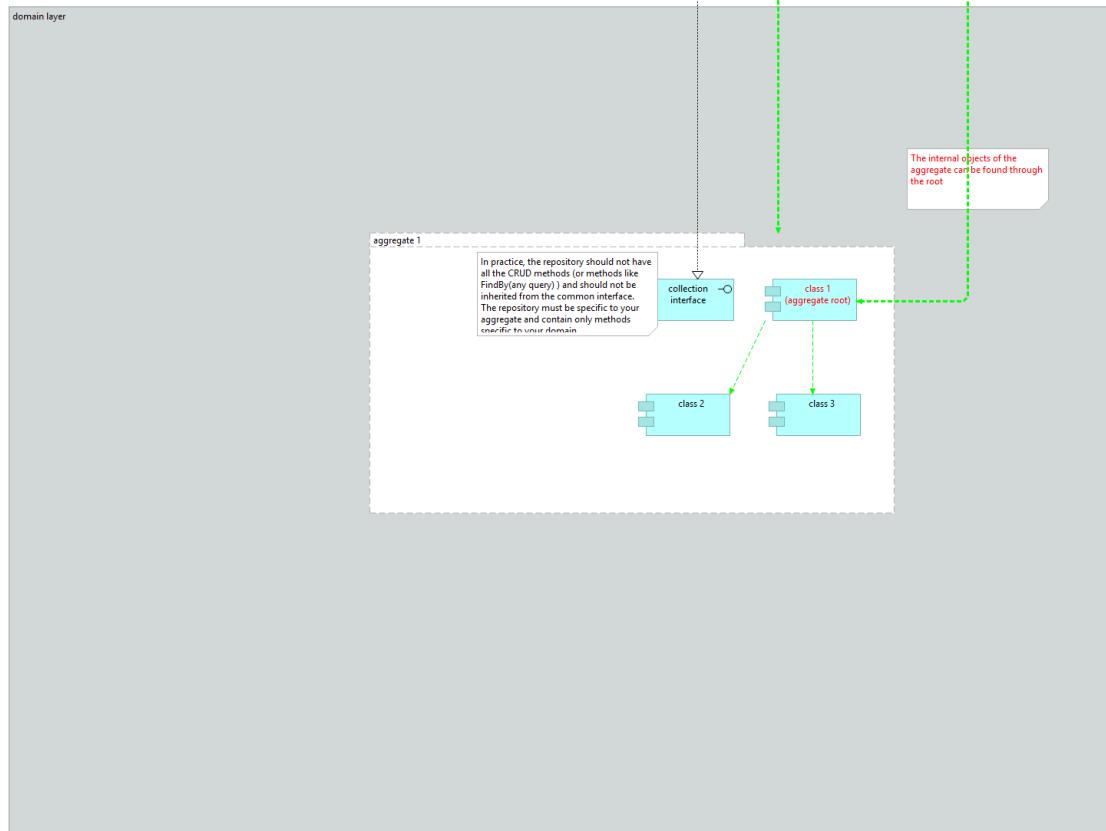
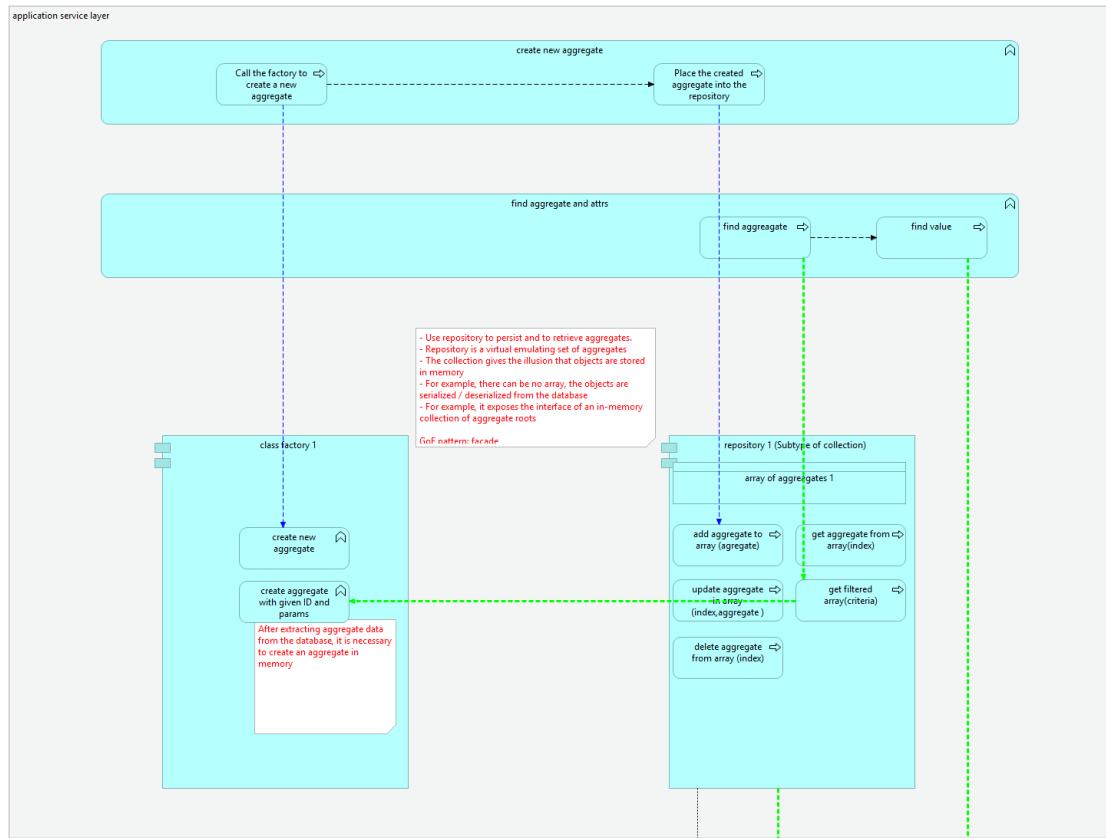
domain layer

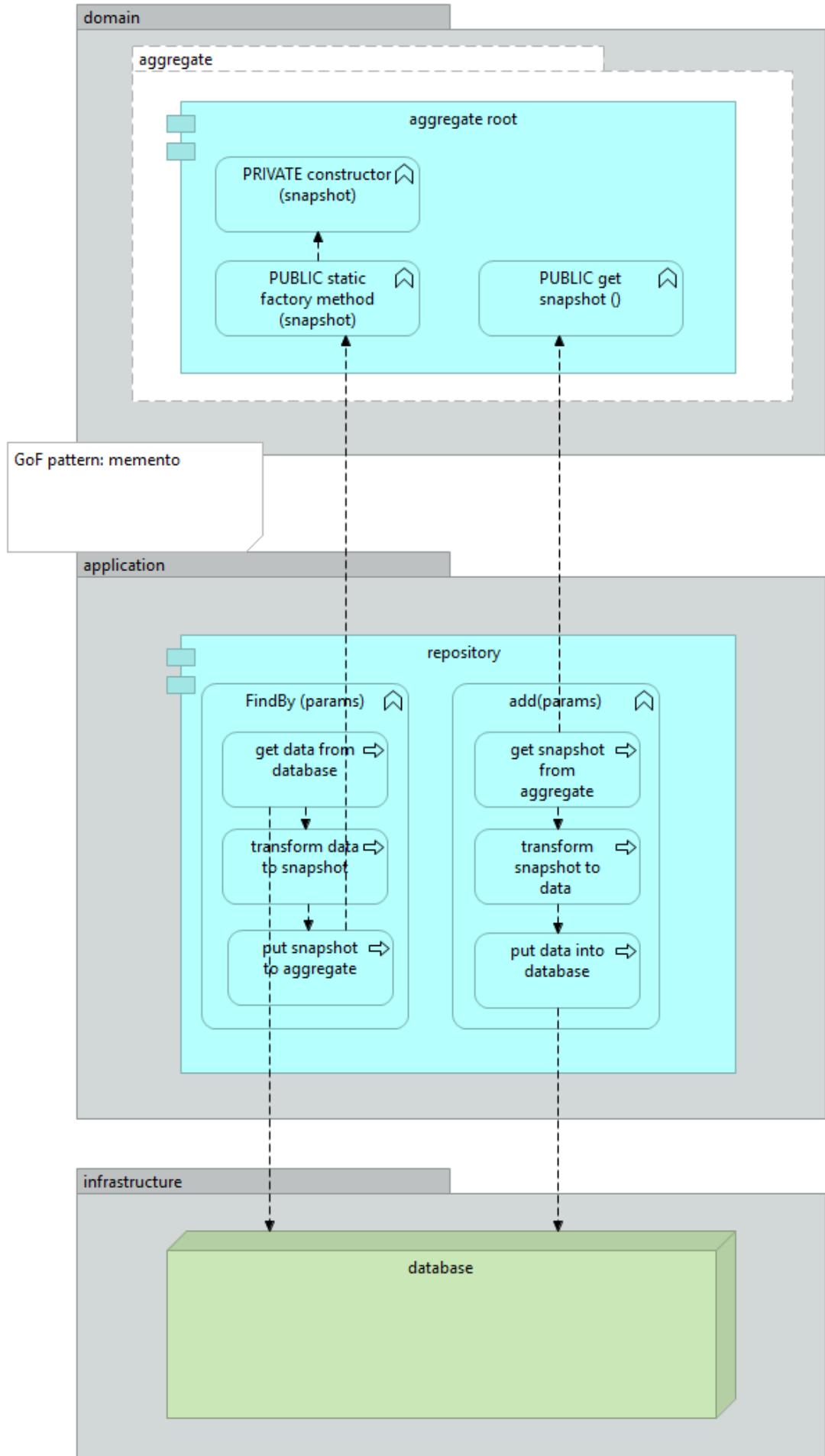
Must satisfy the invariants 2 after creation

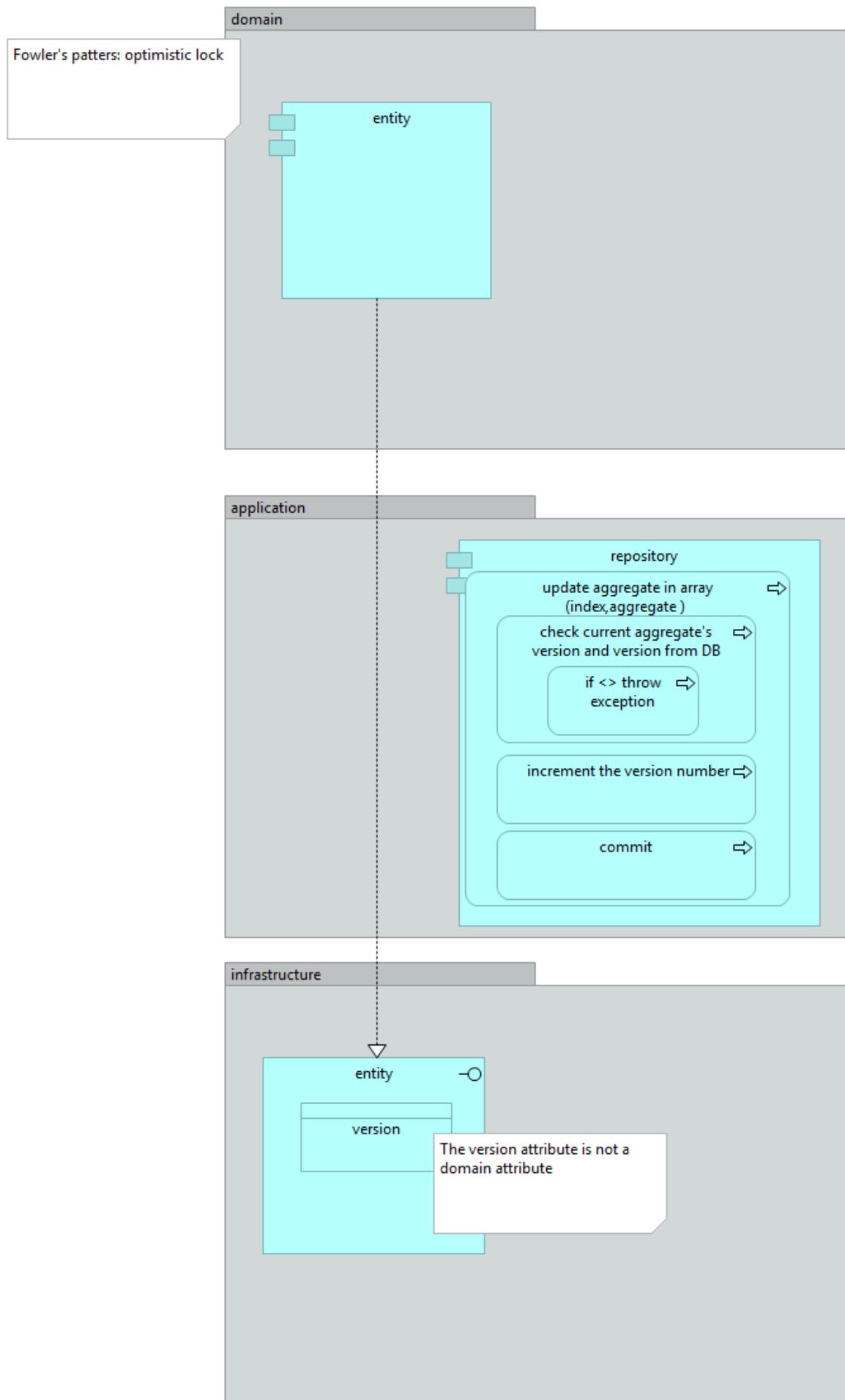


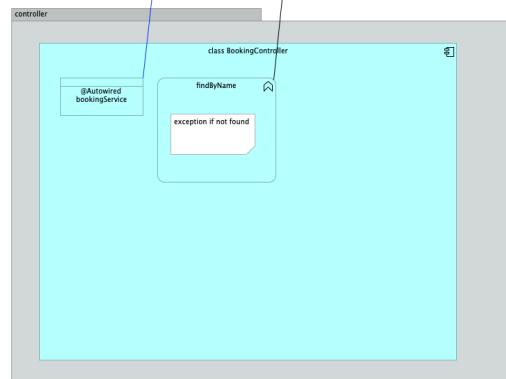
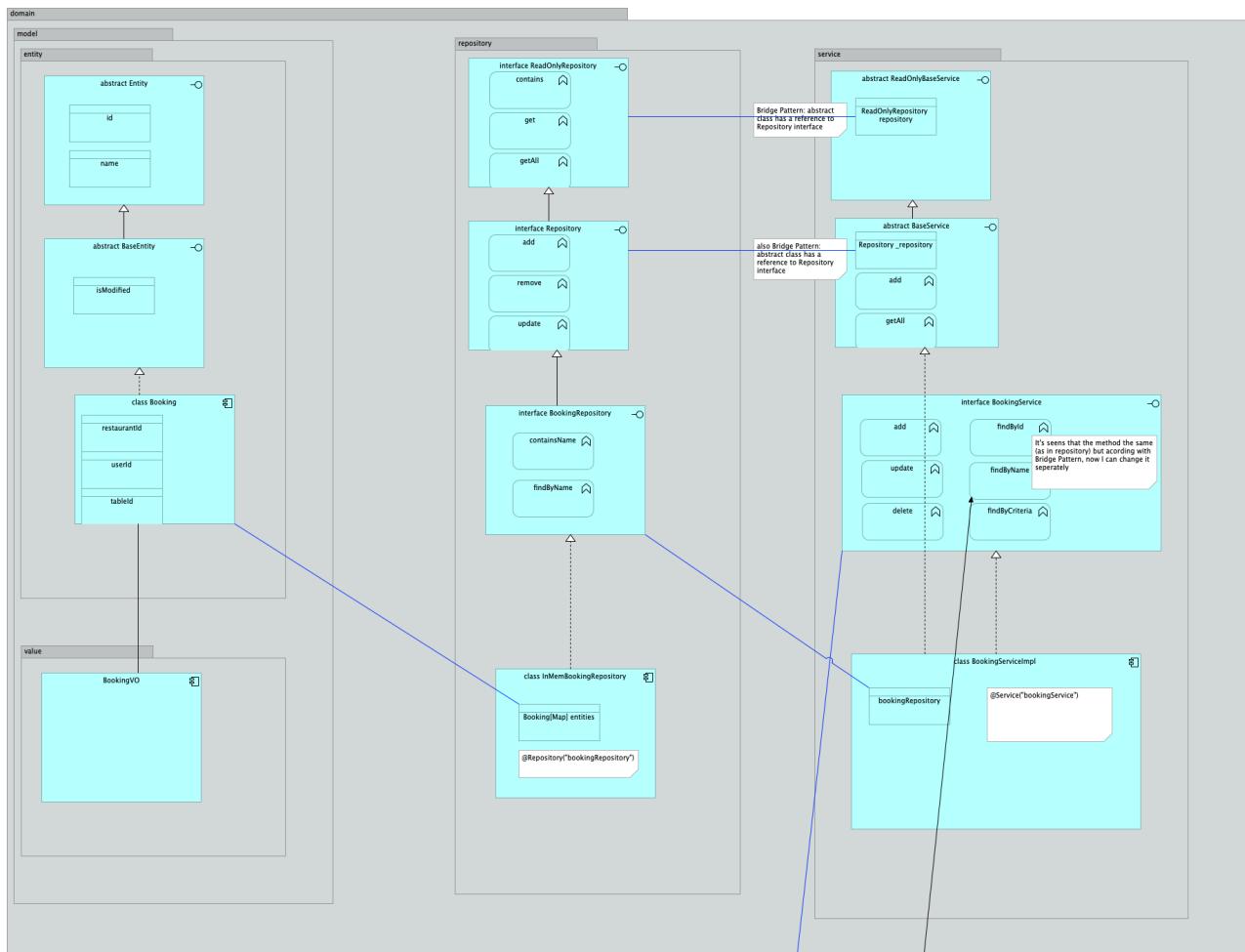


# REPOSITORIES









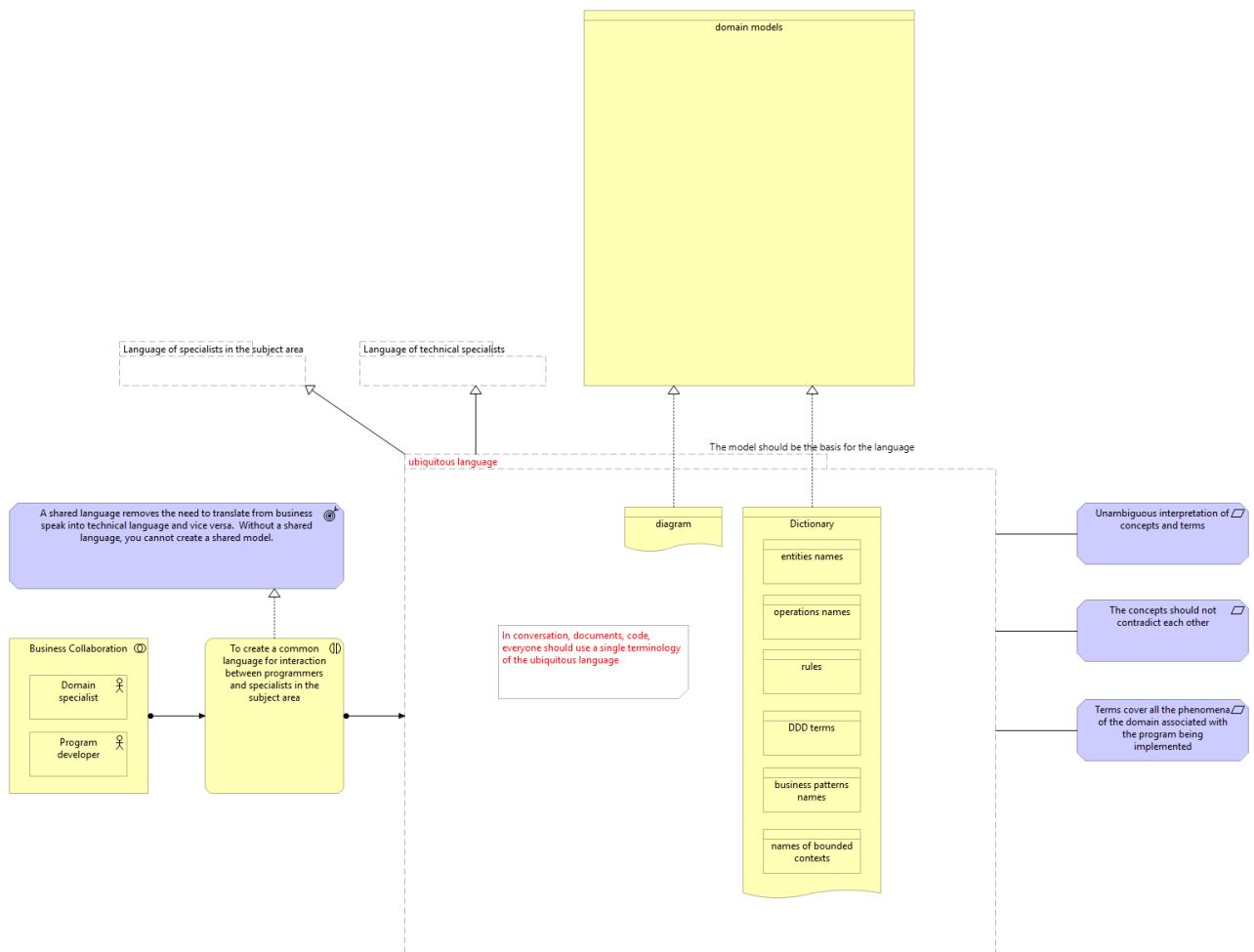
# SUPPLE DESIGN

DOMAIN DRIVEN DESIGN

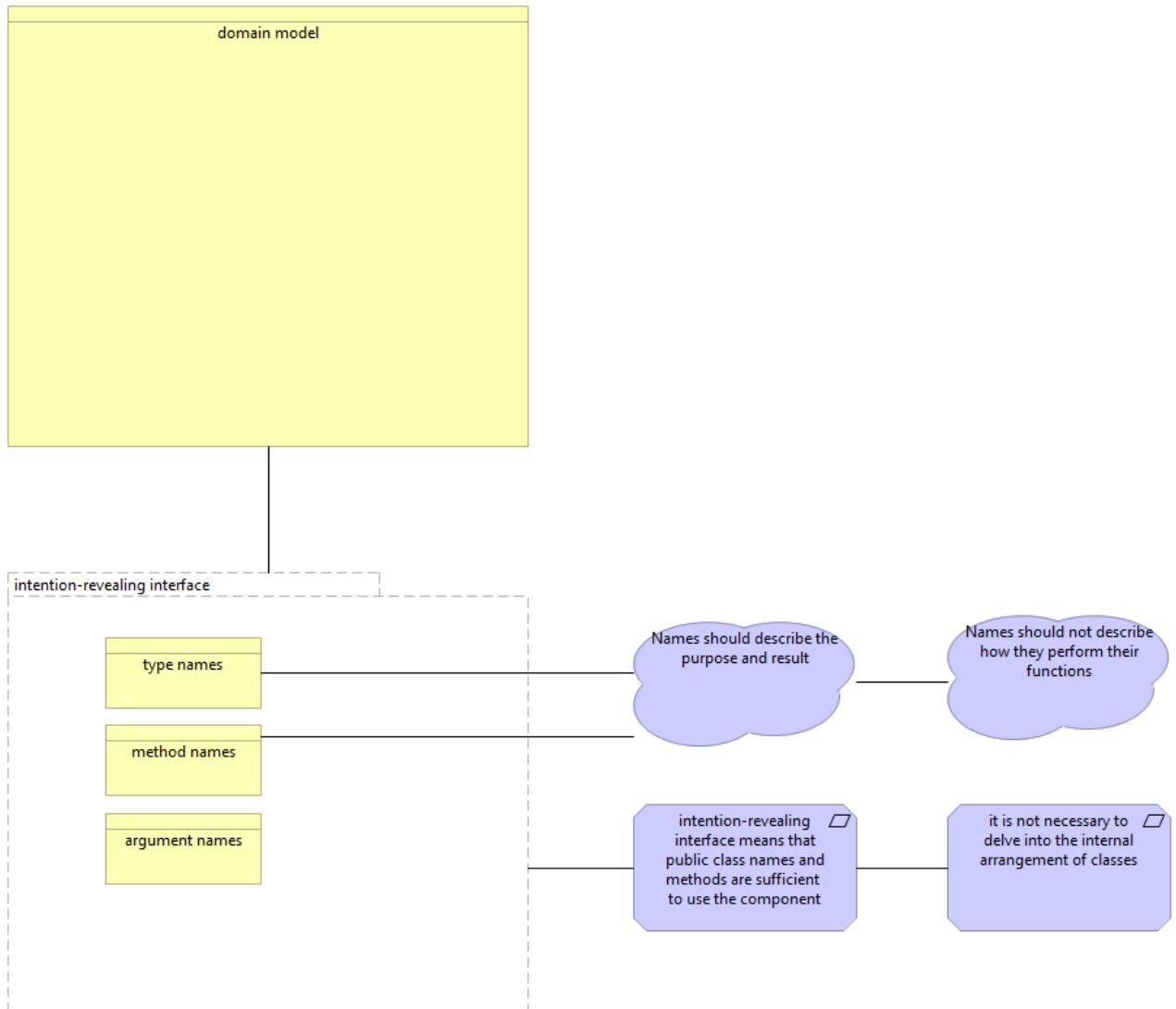
Eric Evans



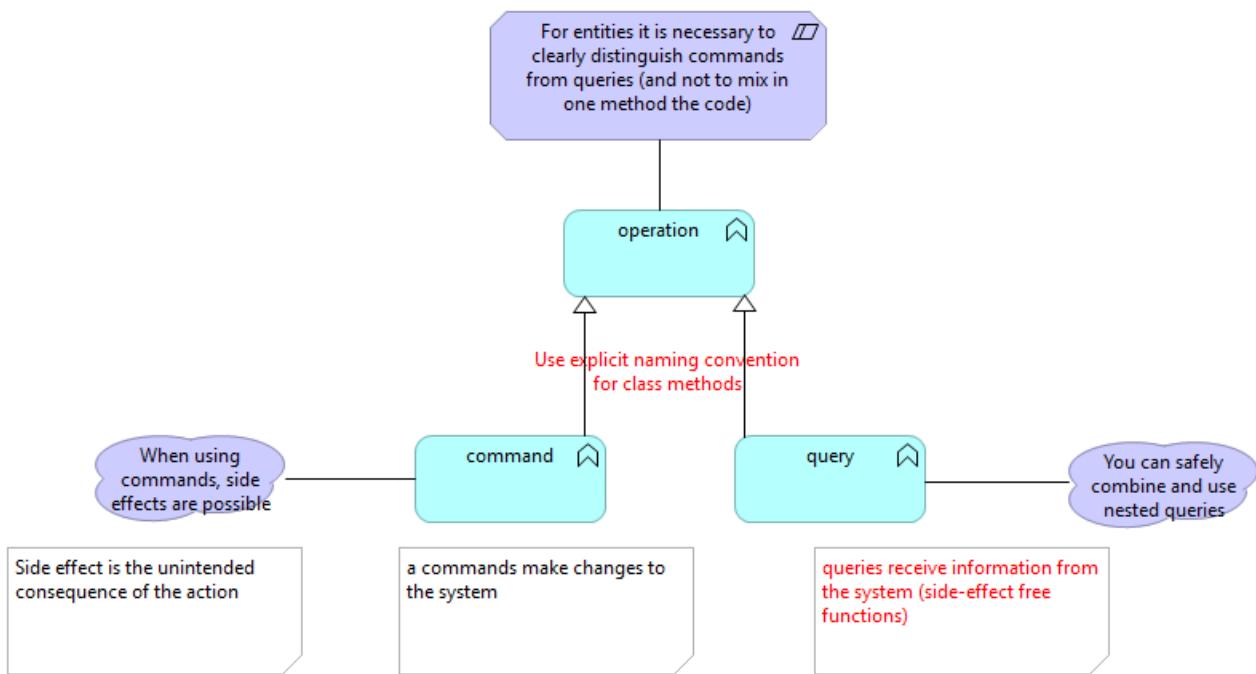
# UBIQUITOUS LANGUAGE



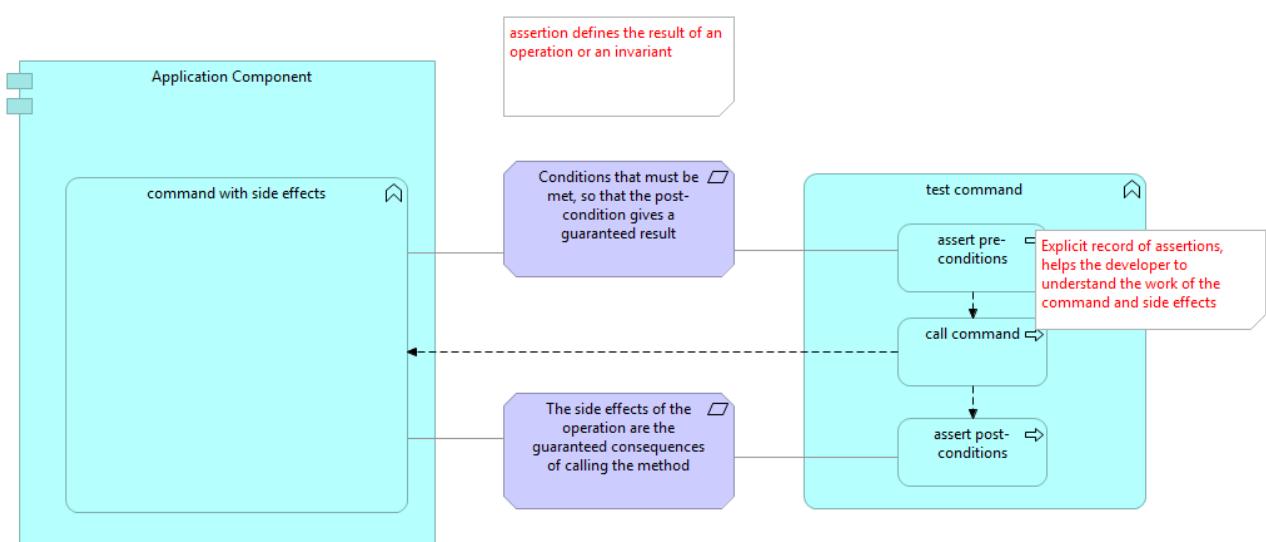
# INTENTION-REVEALING INTERFACES



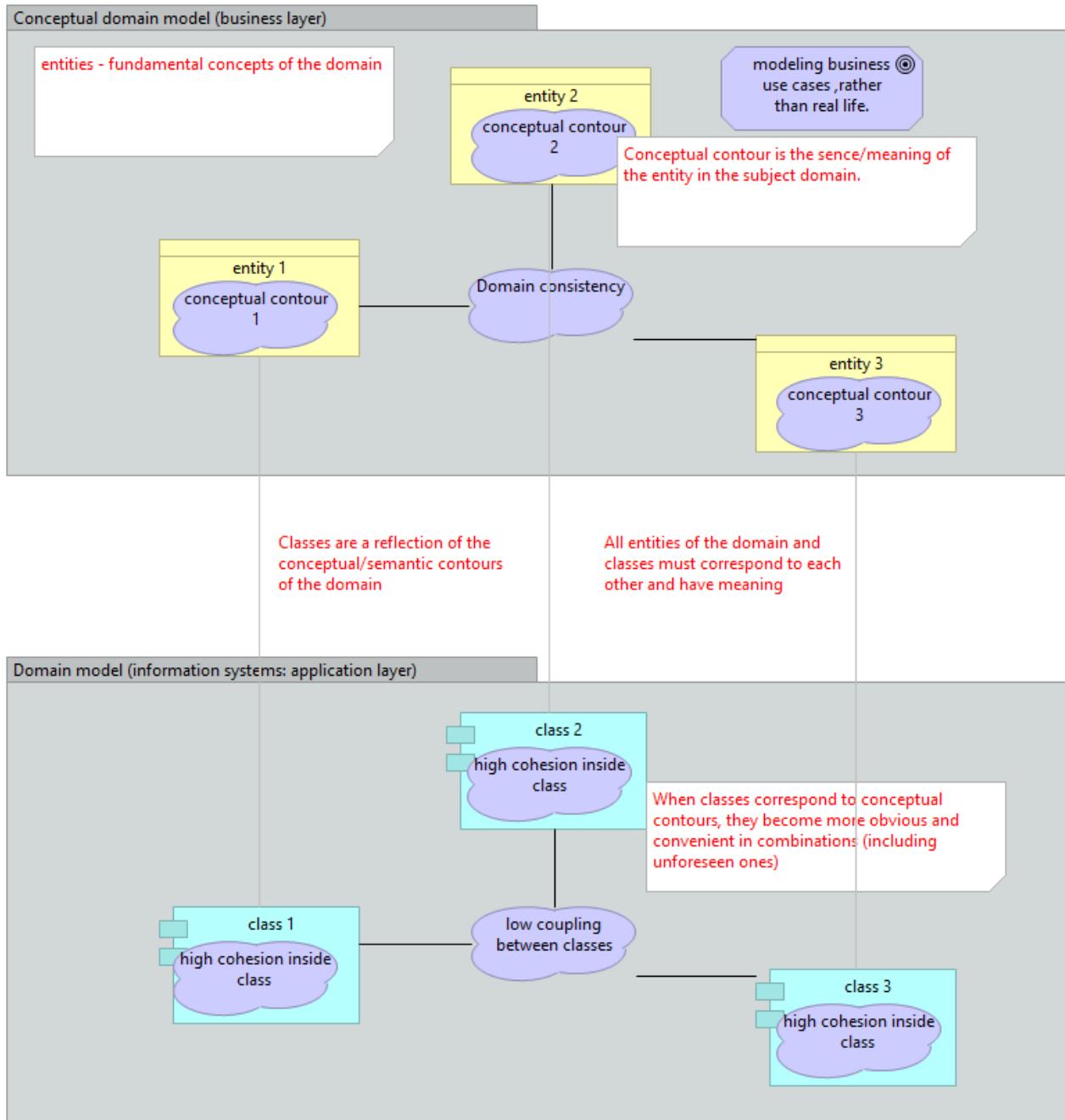
# SIDE-EFFECT FREE FUNCTIONS



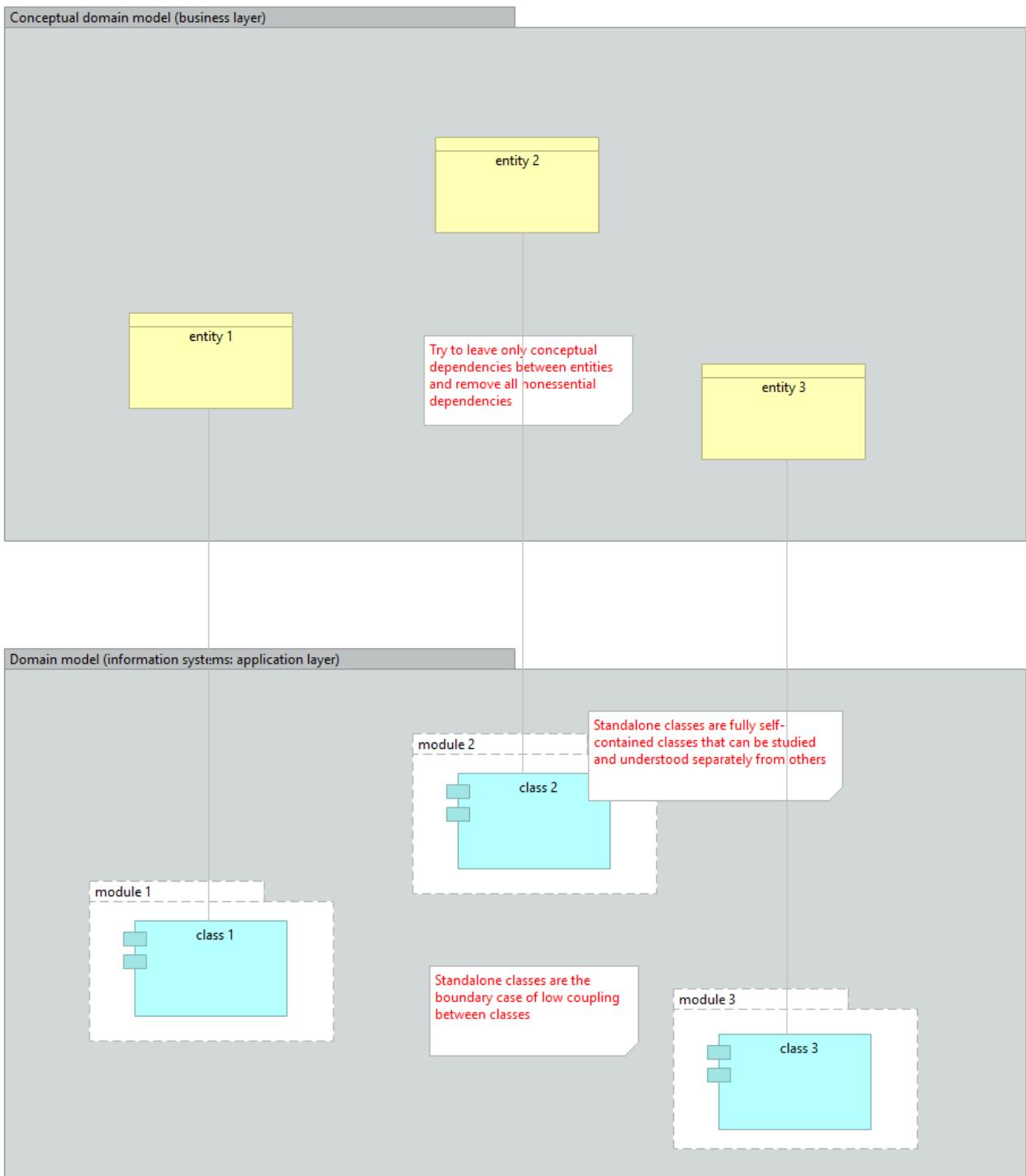
# ASSERTIONS



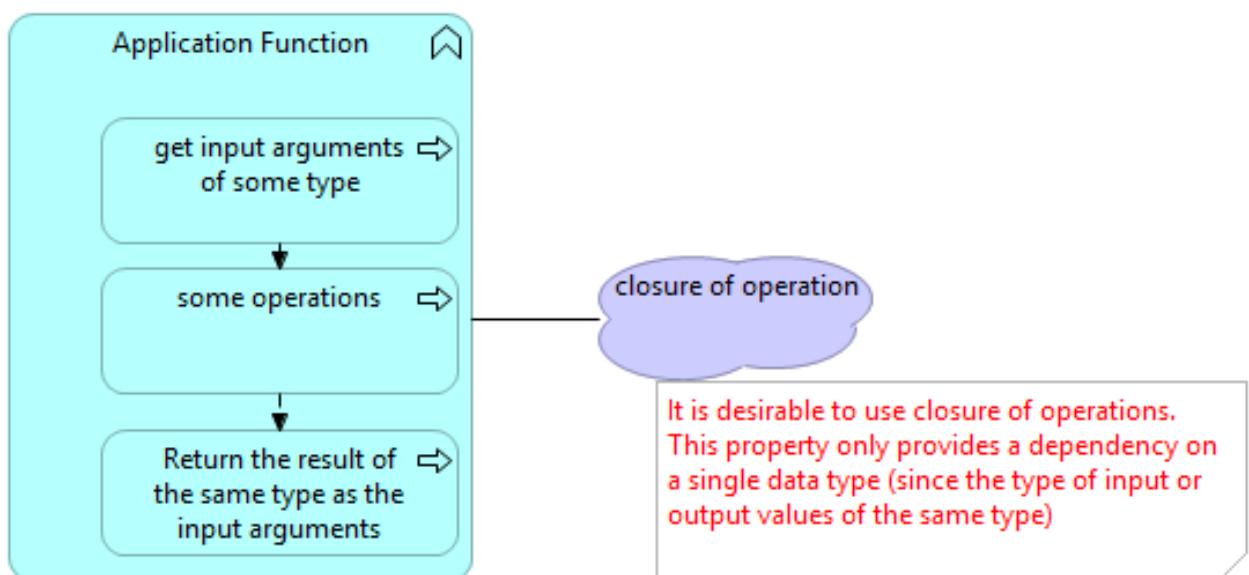
# CONCEPTUAL CONTOURS



# STANDALONE CLASSES



# CLOSURE OF OPERATIONS



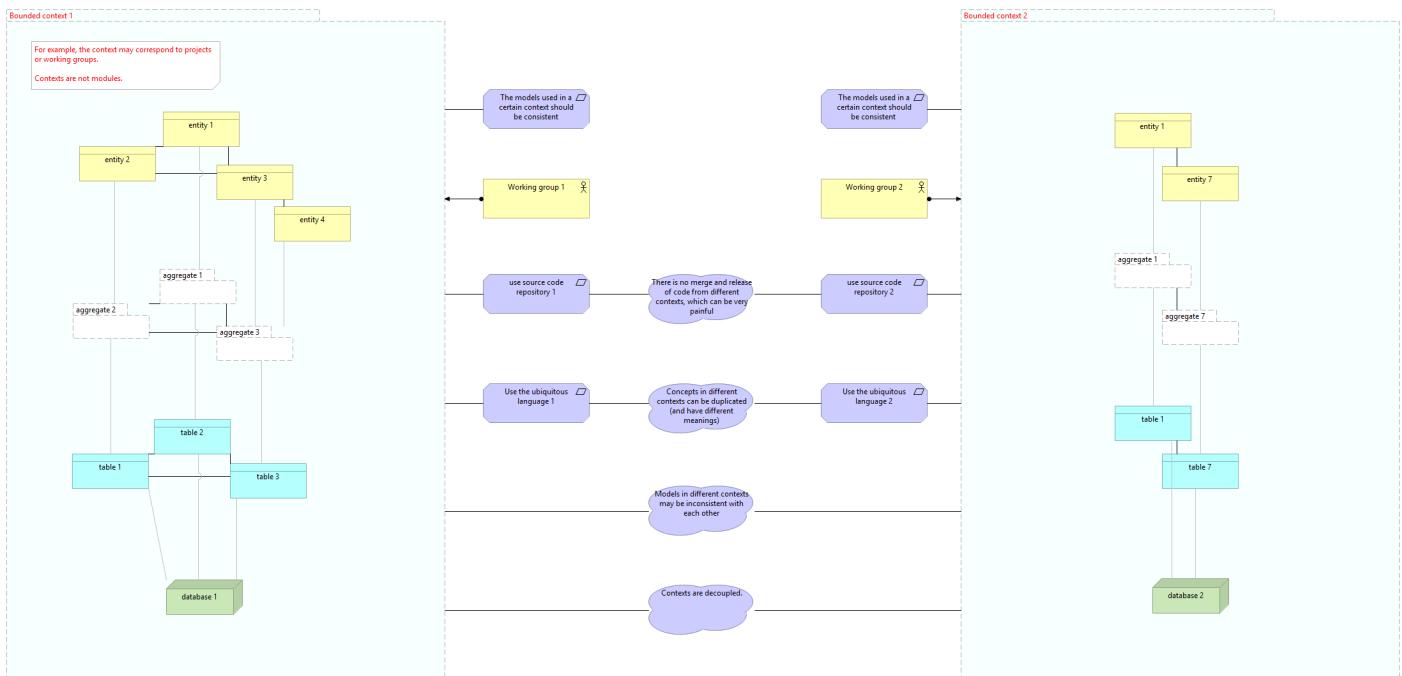
# MODEL INTEGRITY AND CONTEXT

DOMAIN DRIVEN DESIGN

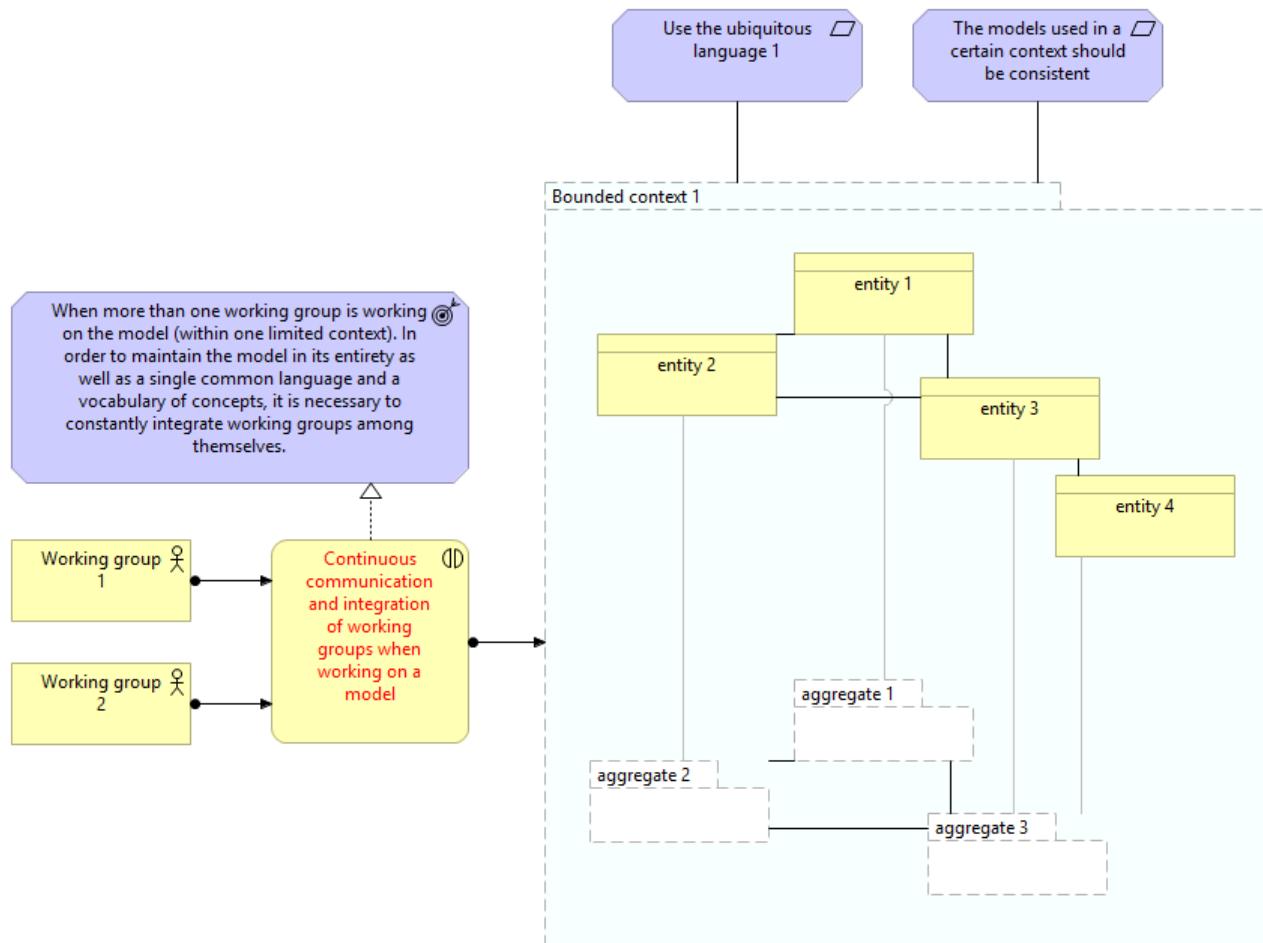
Eric Evans



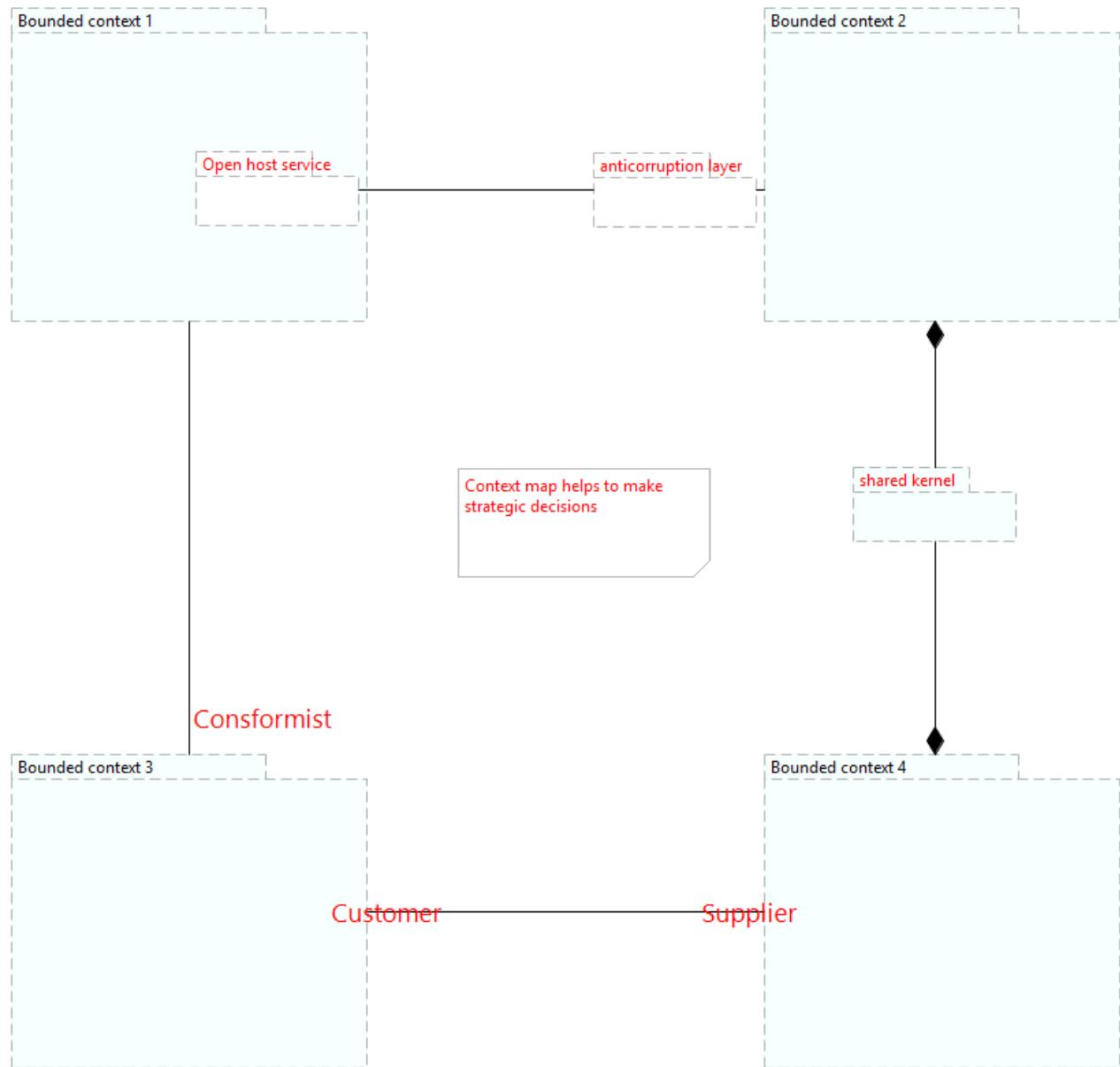
# BOUNDED CONTEXT



# CONTINUOUS INTEGRATION

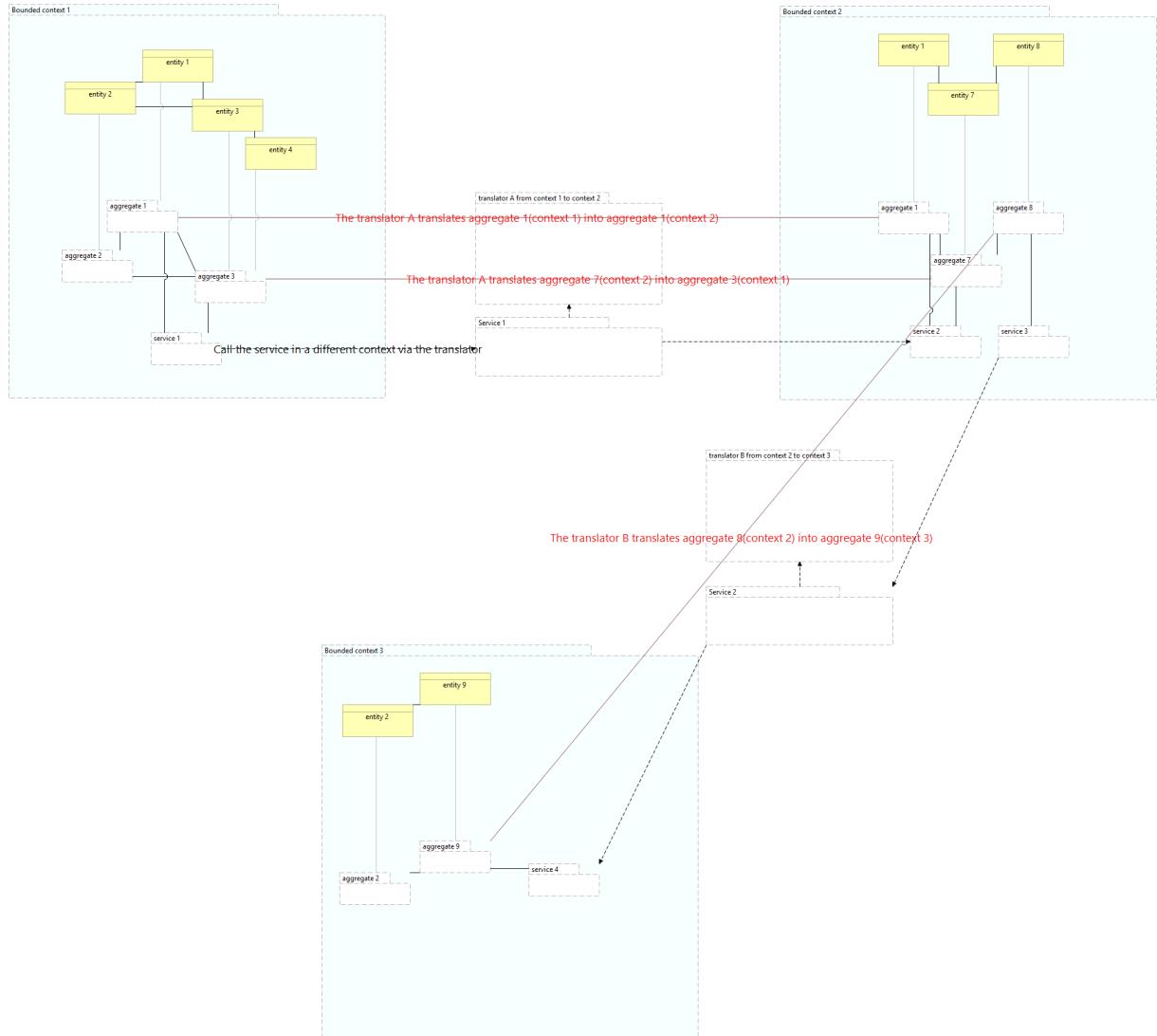


# STRATEGIC CONTEXT MAP

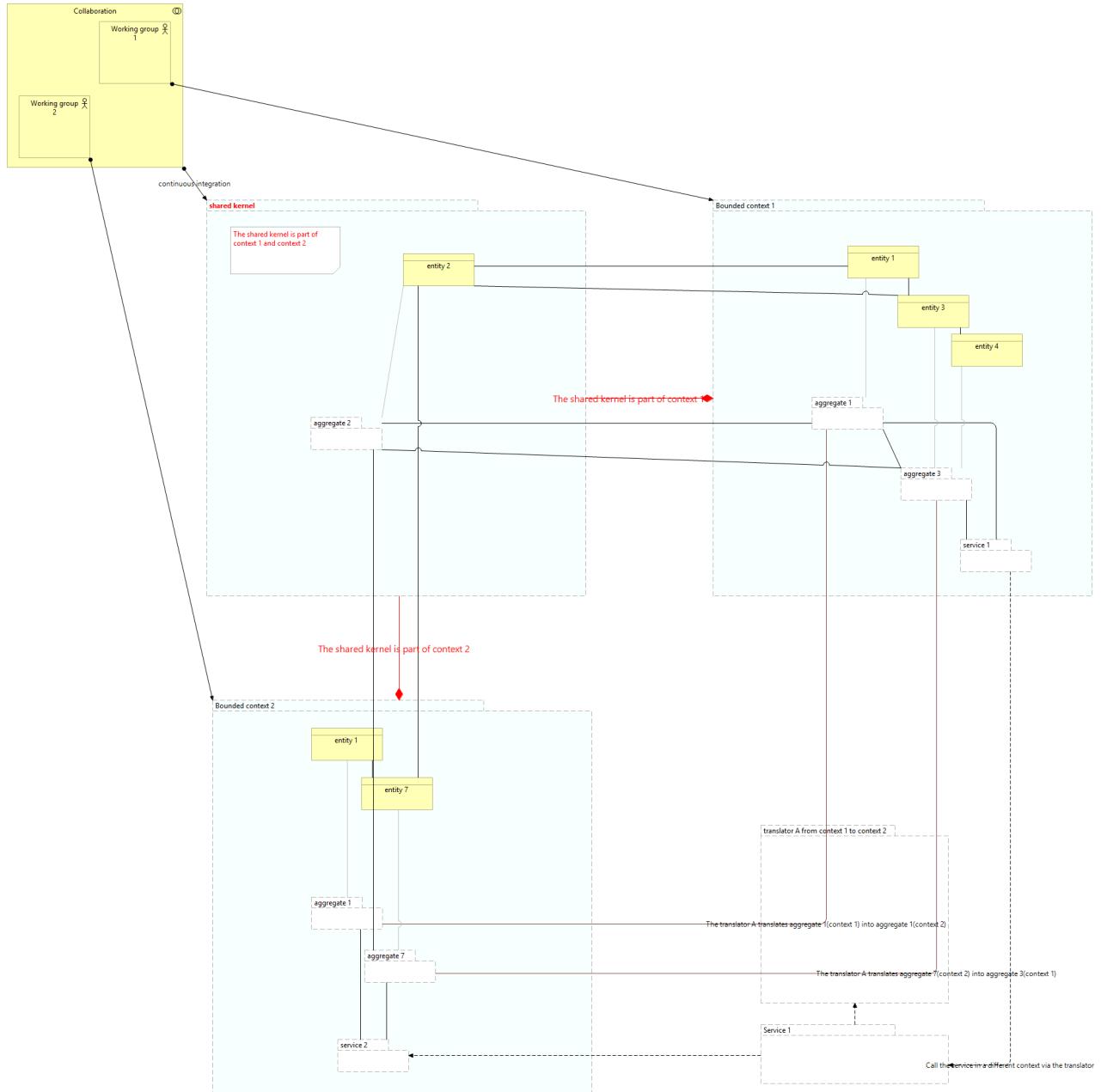


# CONTEXTUAL MAP

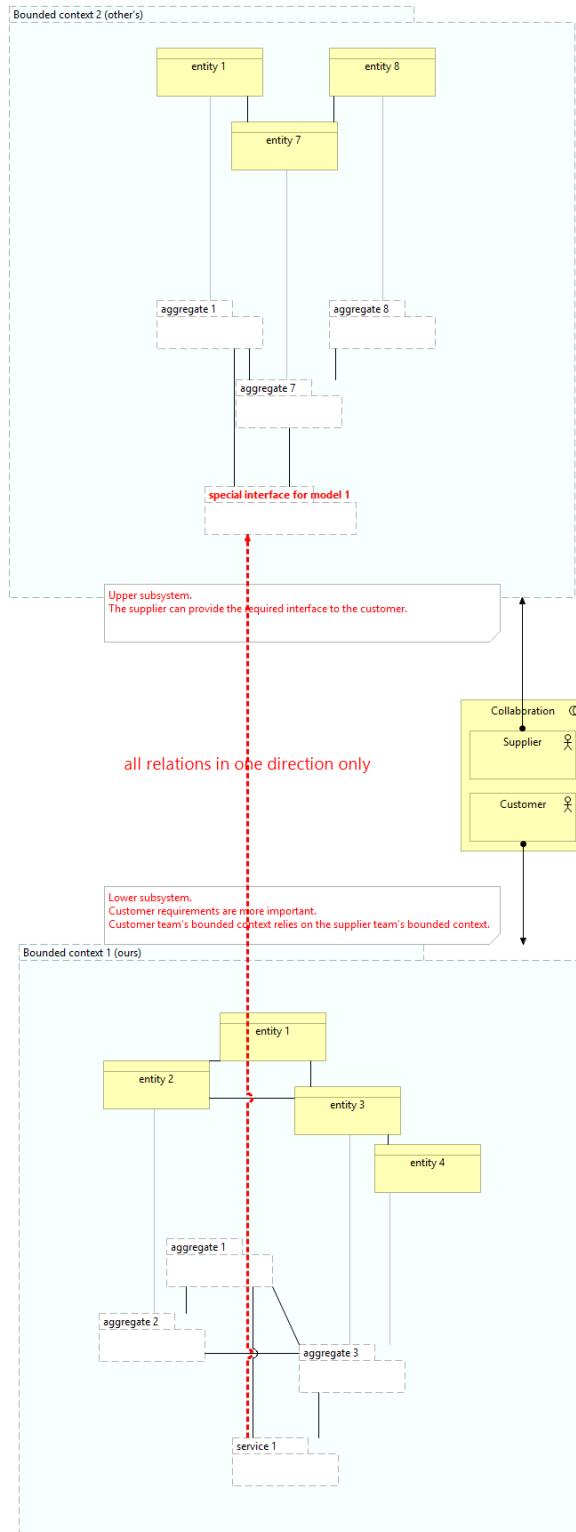
A single map of all contexts.  
Integration of all bounded contexts.



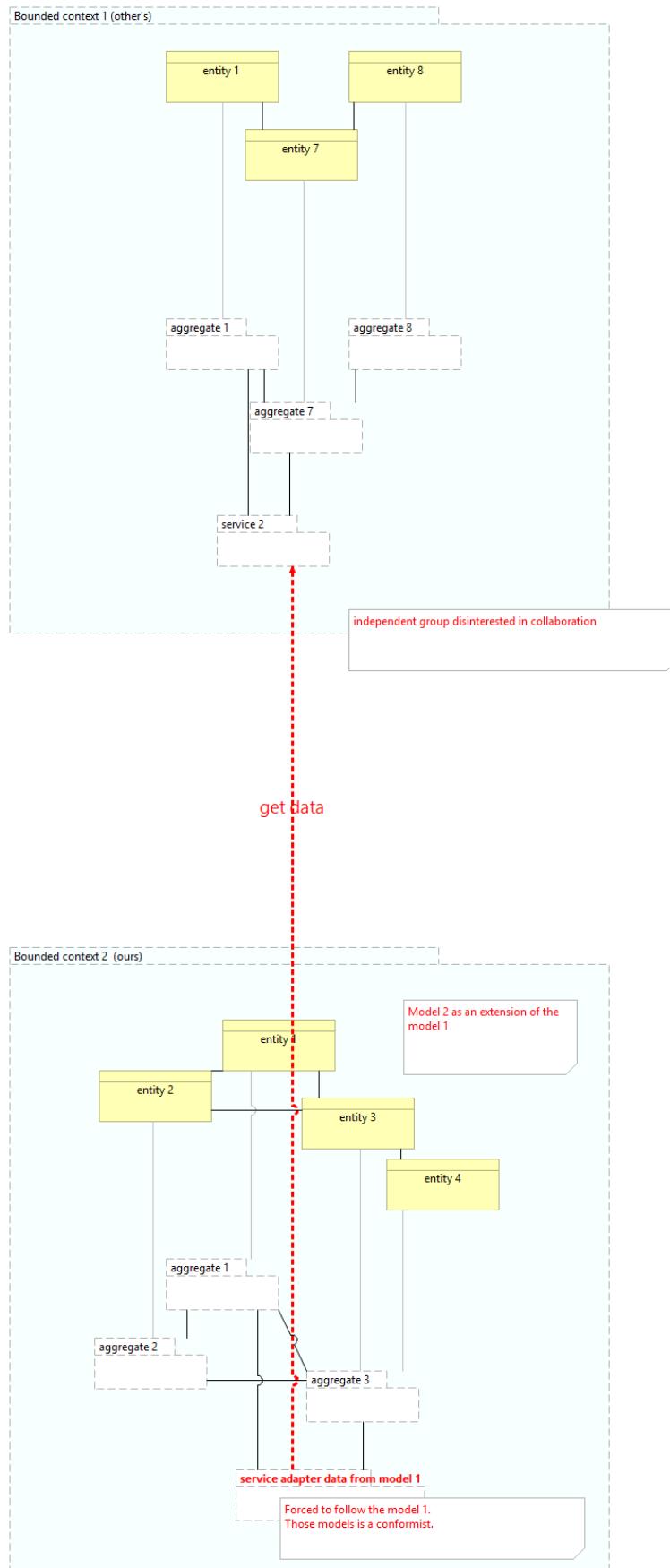
# SHARED KERNEL



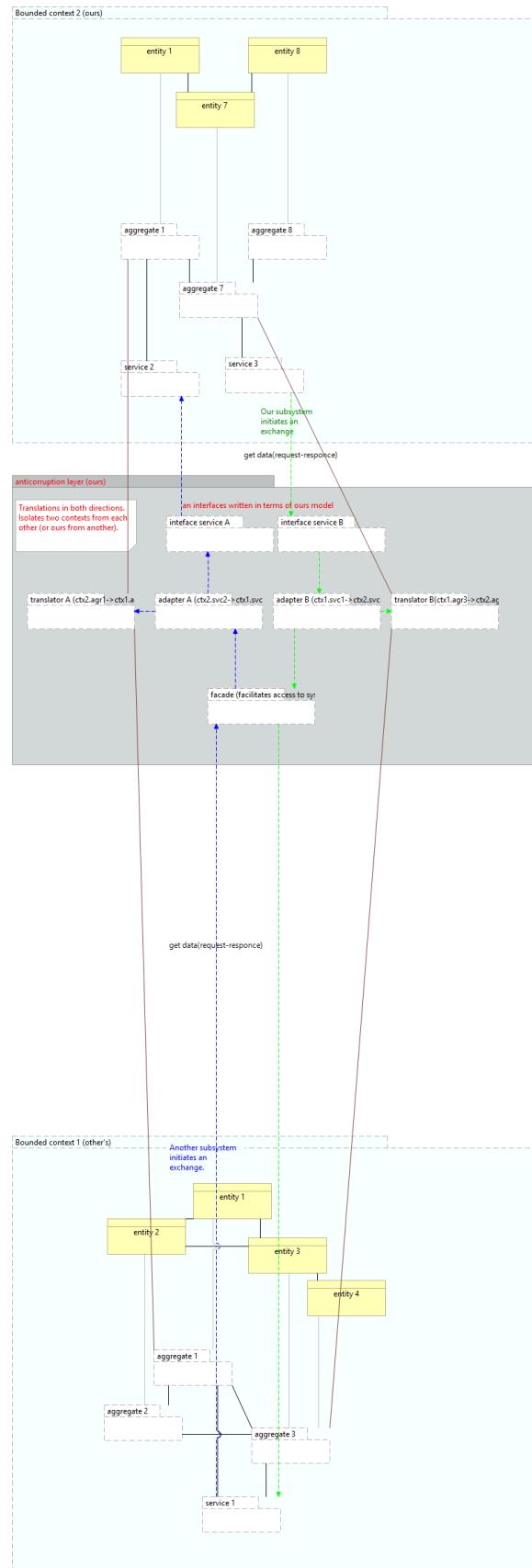
# CUSTOMER-SUPPLIER TEAMS



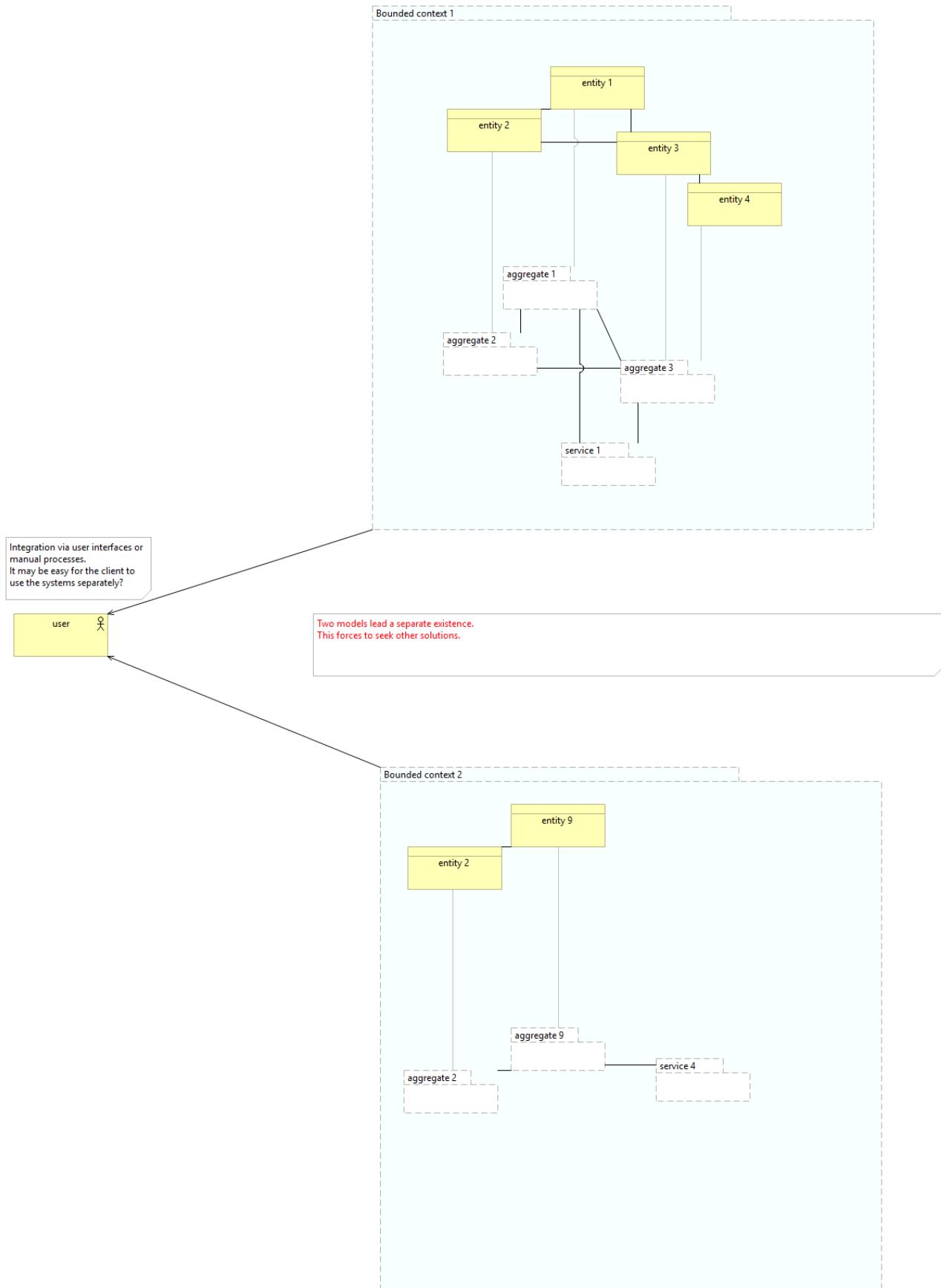
# CONFORMIST



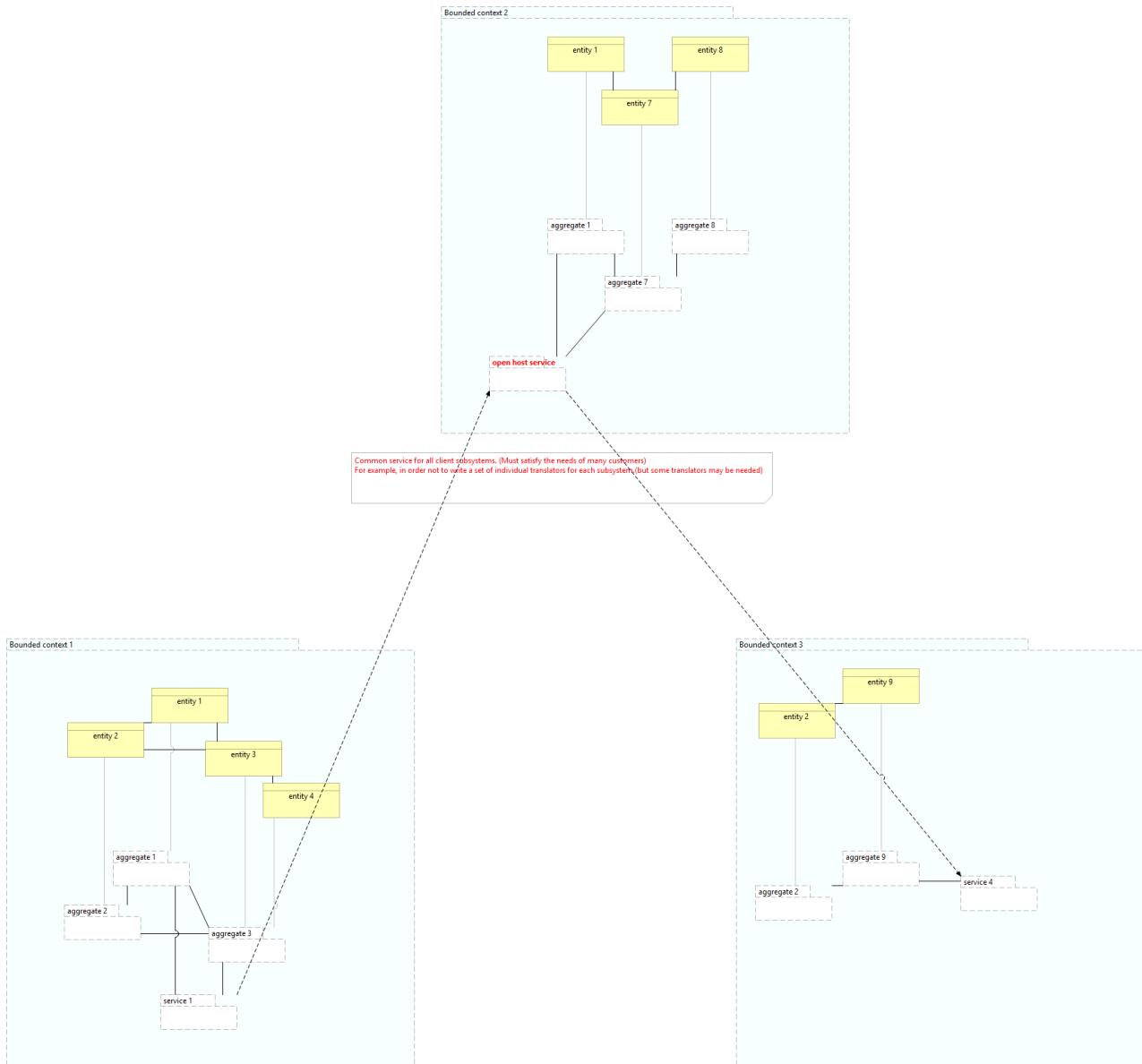
# ANTICORRUPTION LAYER



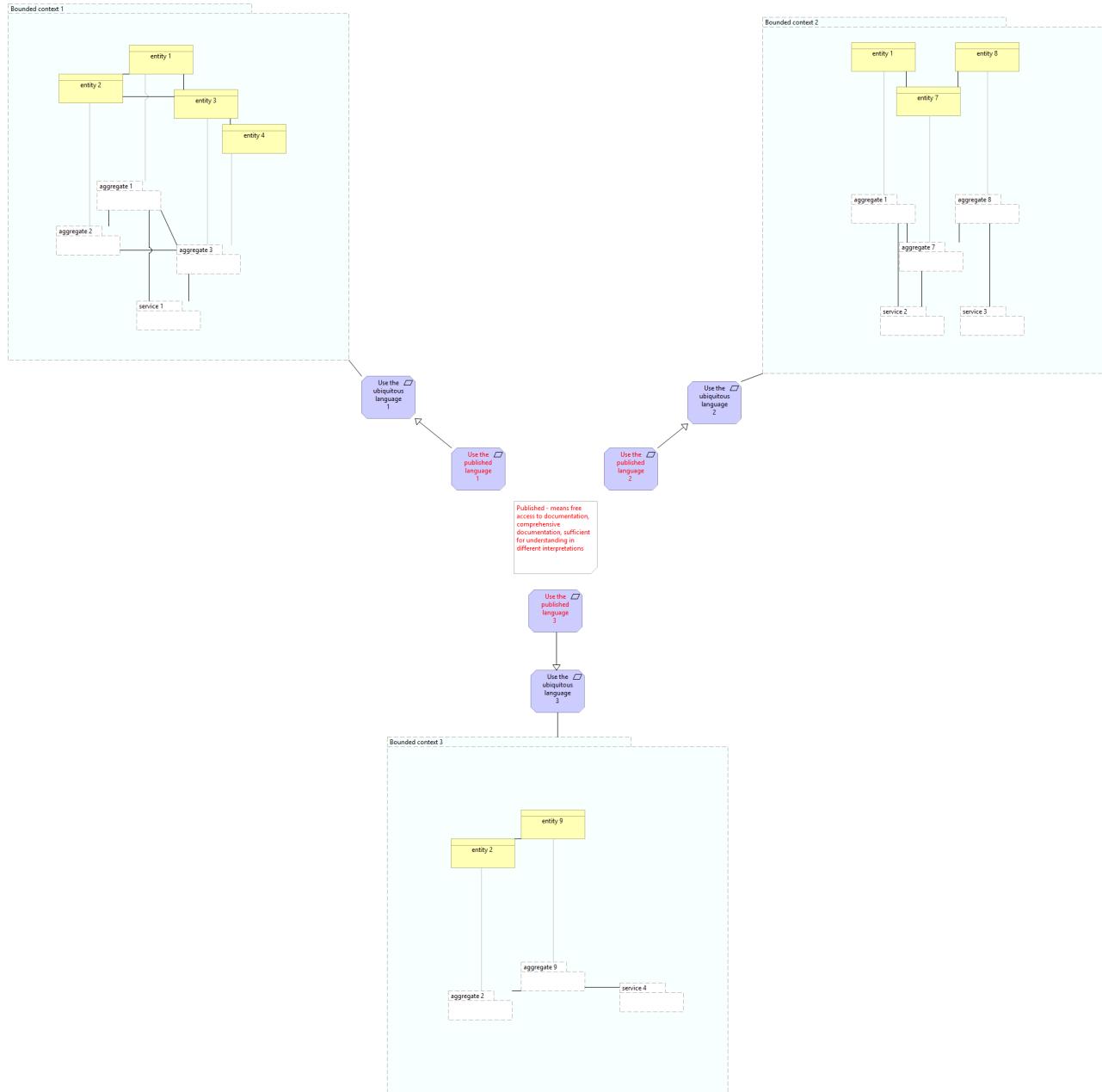
# SEPARATE WAYS



# OPEN HOST SERVICE



# PUBLISHED LANGUAGE



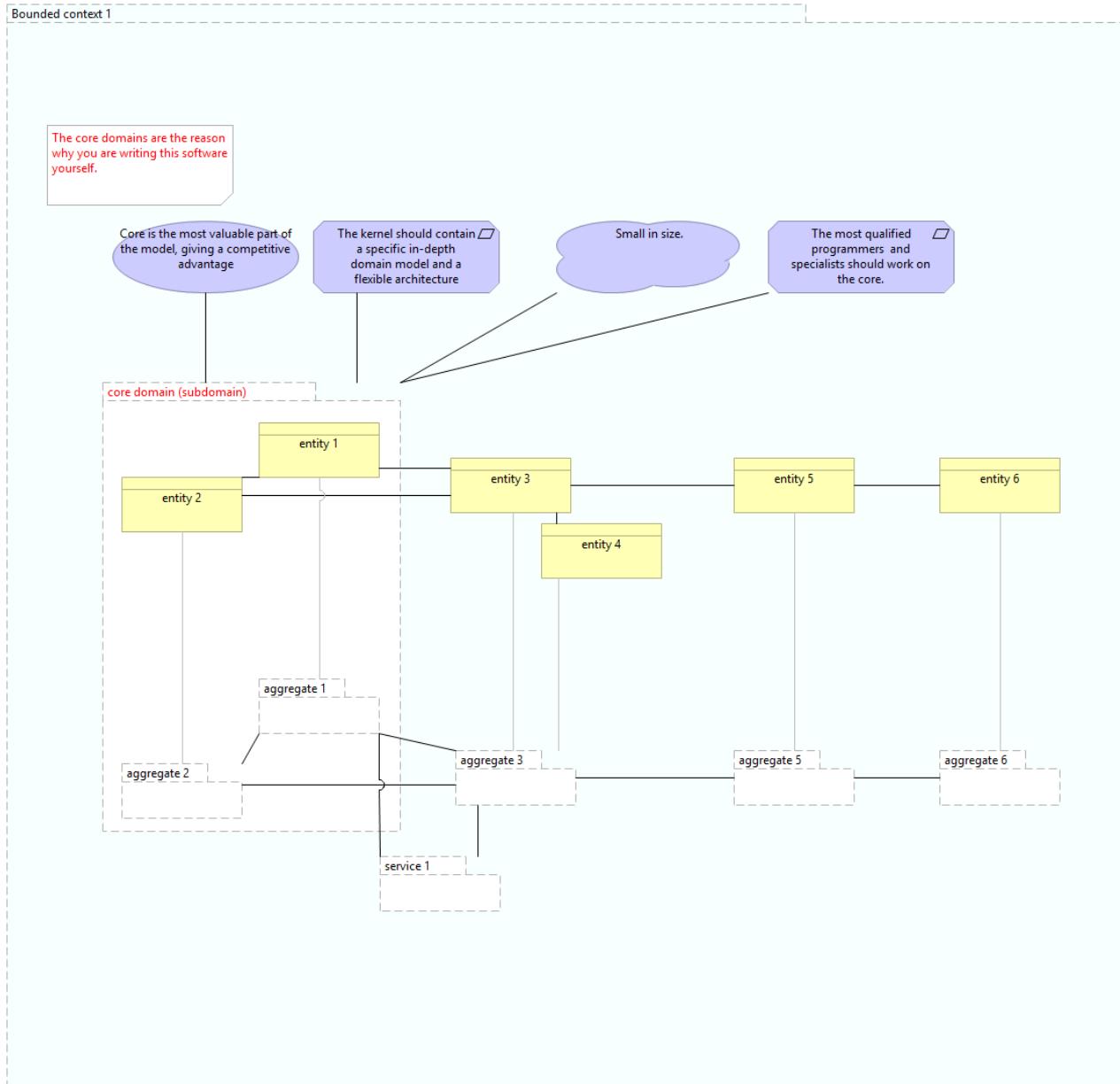
# DISTILLATION

DOMAIN DRIVEN DESIGN

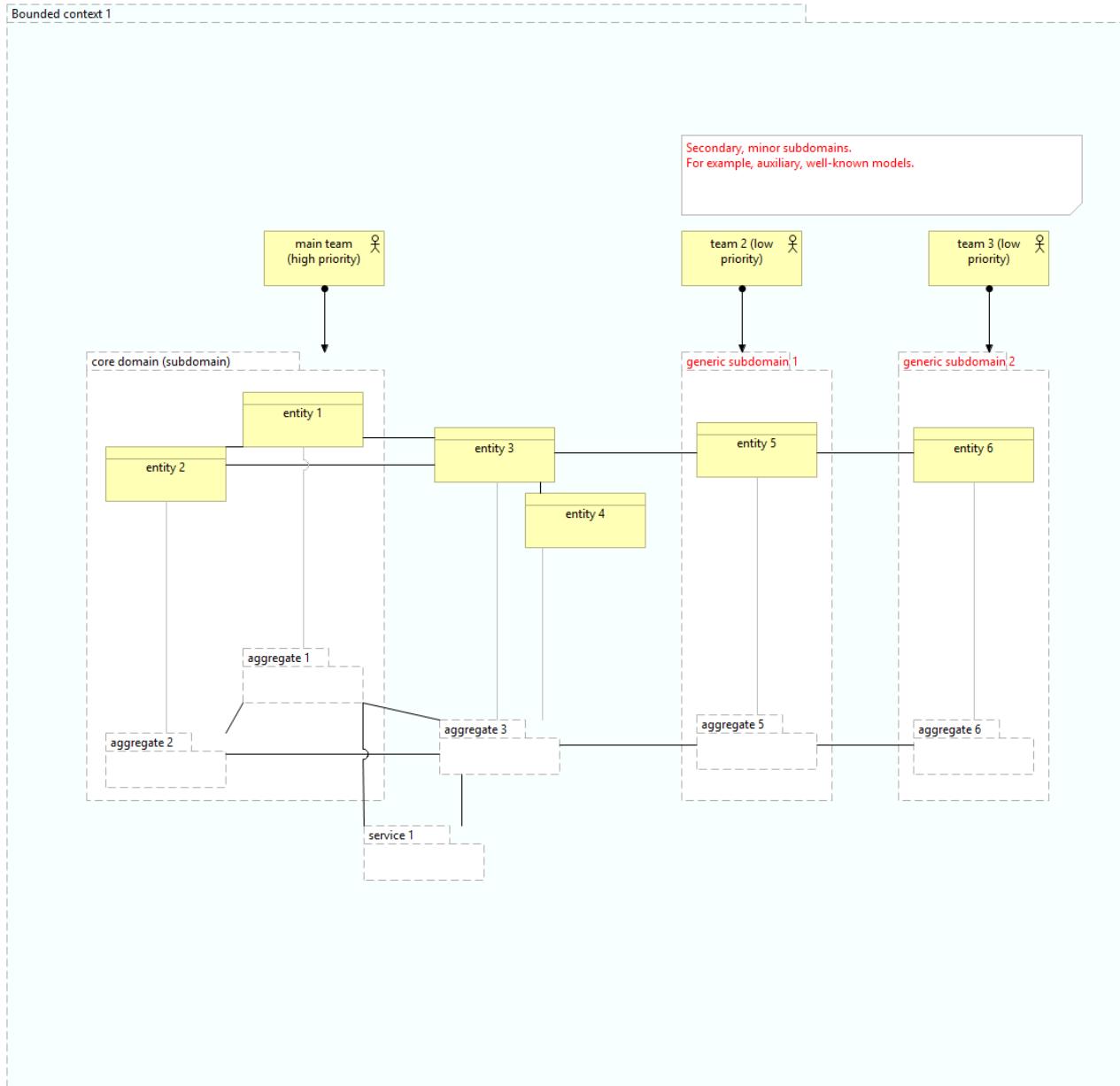
Eric Evans



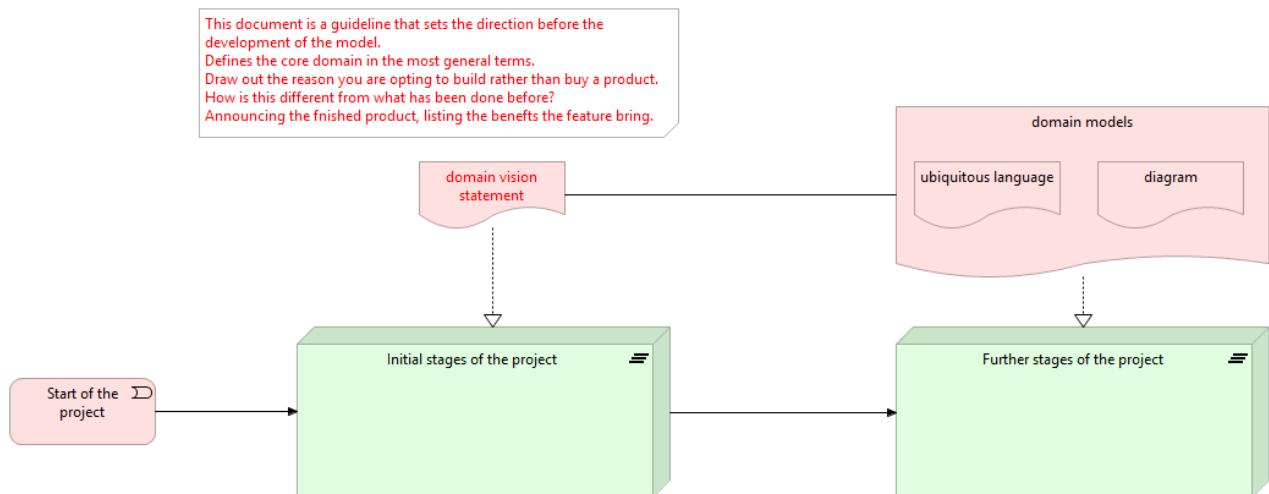
# CORE DOMAIN



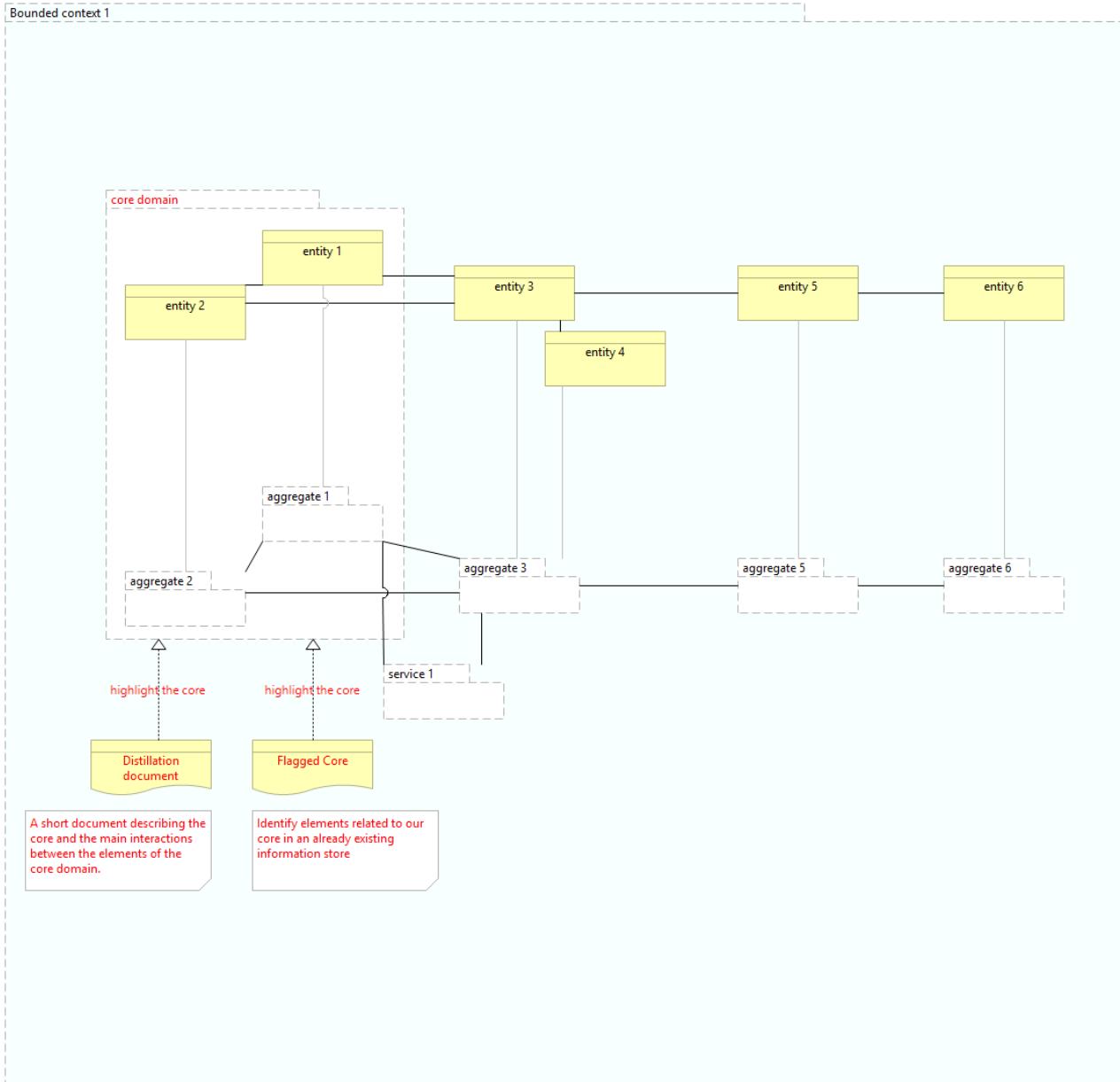
# GENERIC SUBDOMAINS



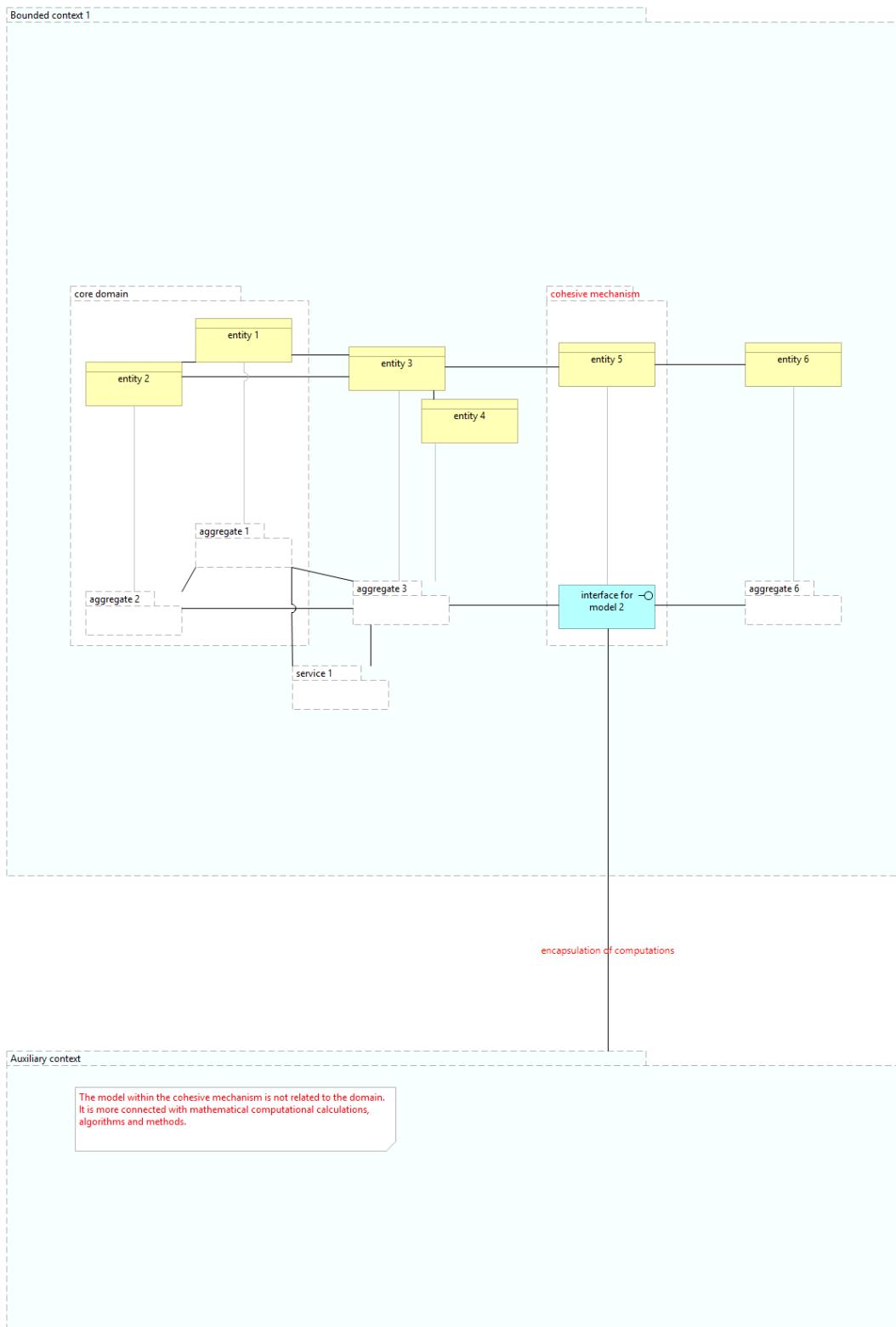
# DOMAIN VISION STATEMENT



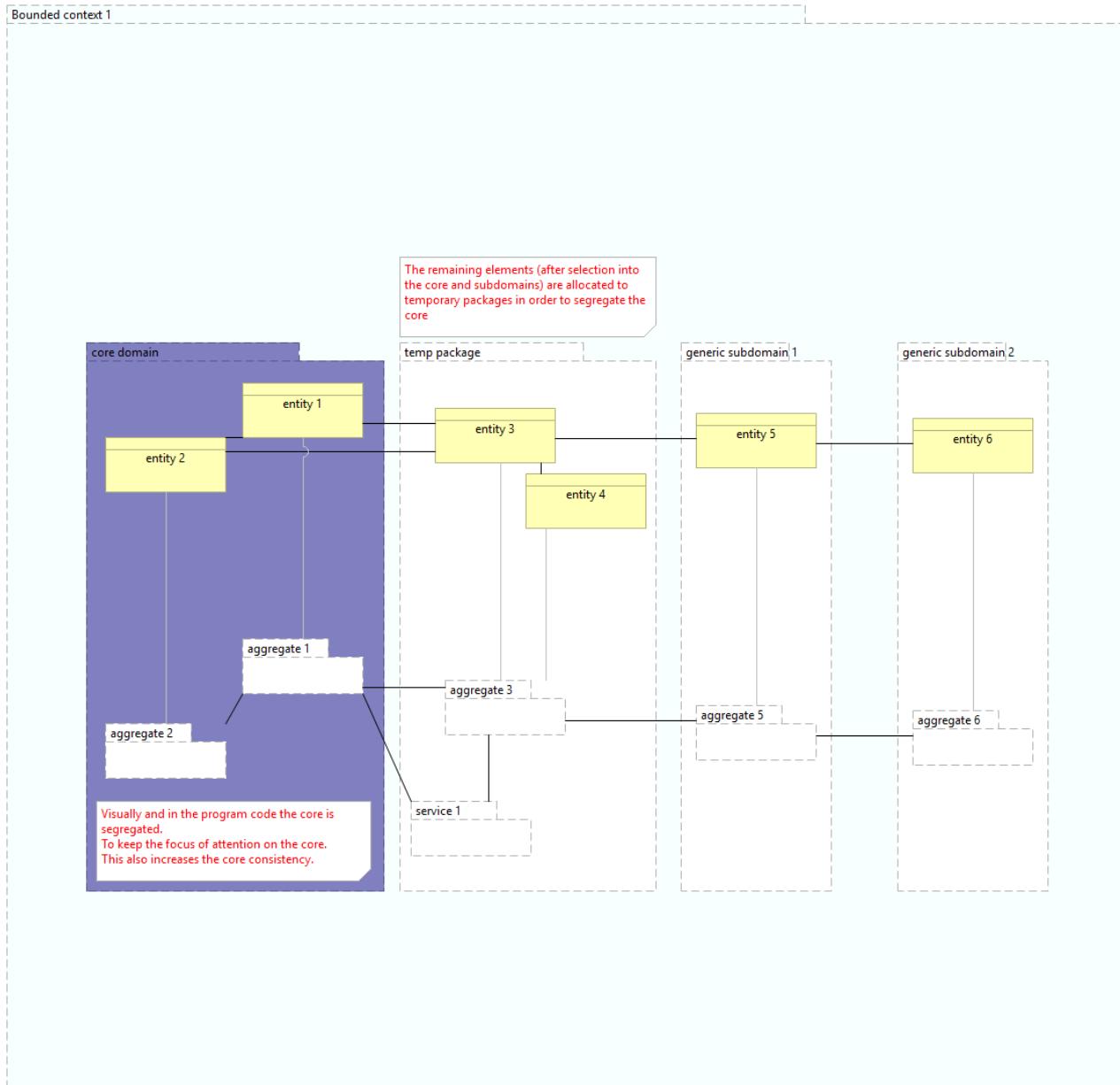
# HIGHLIGHTED CORE



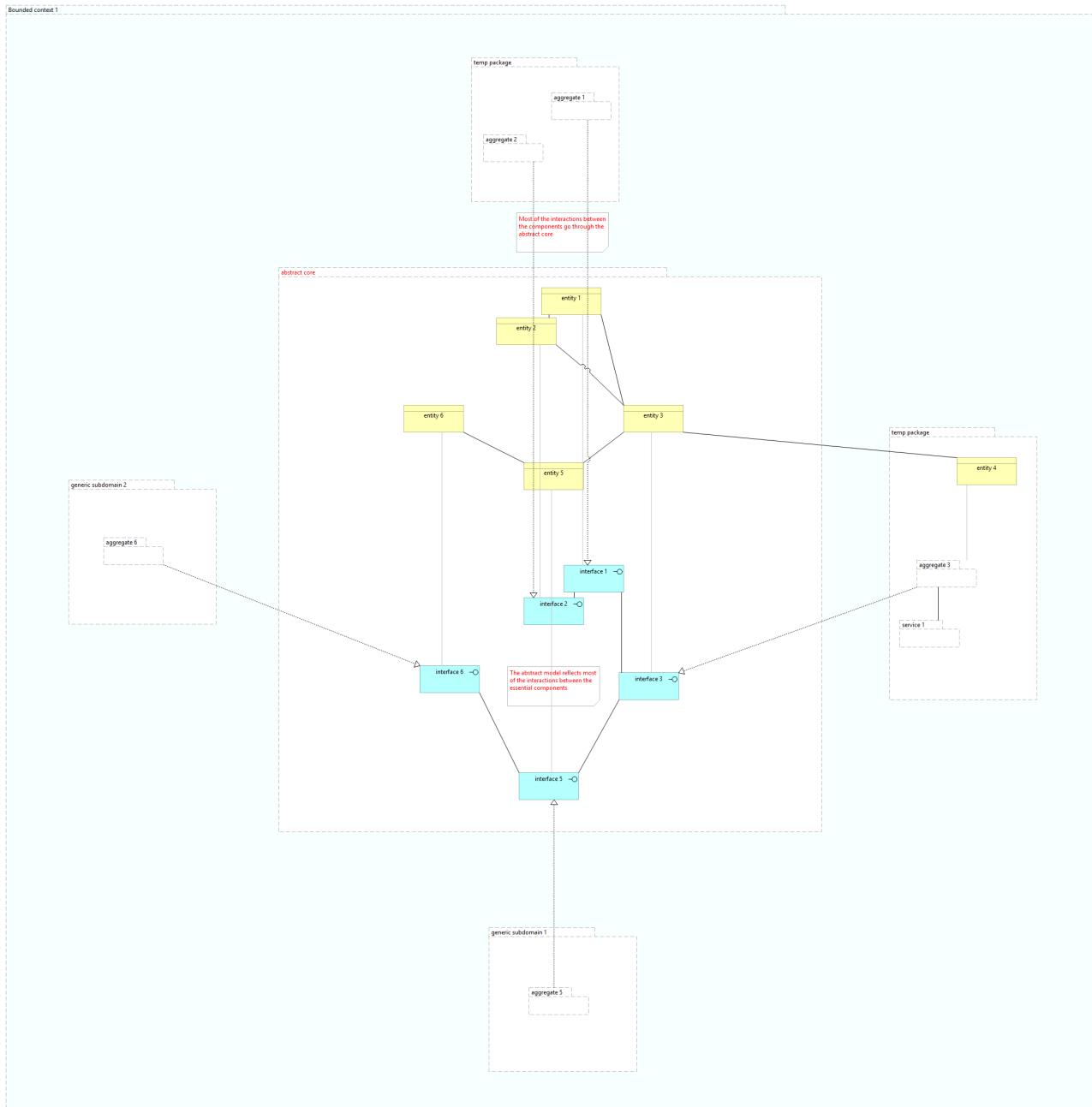
# COHESIVE MECHANISMS



# SEGREGATED CORE



# ABSTRACT CORE



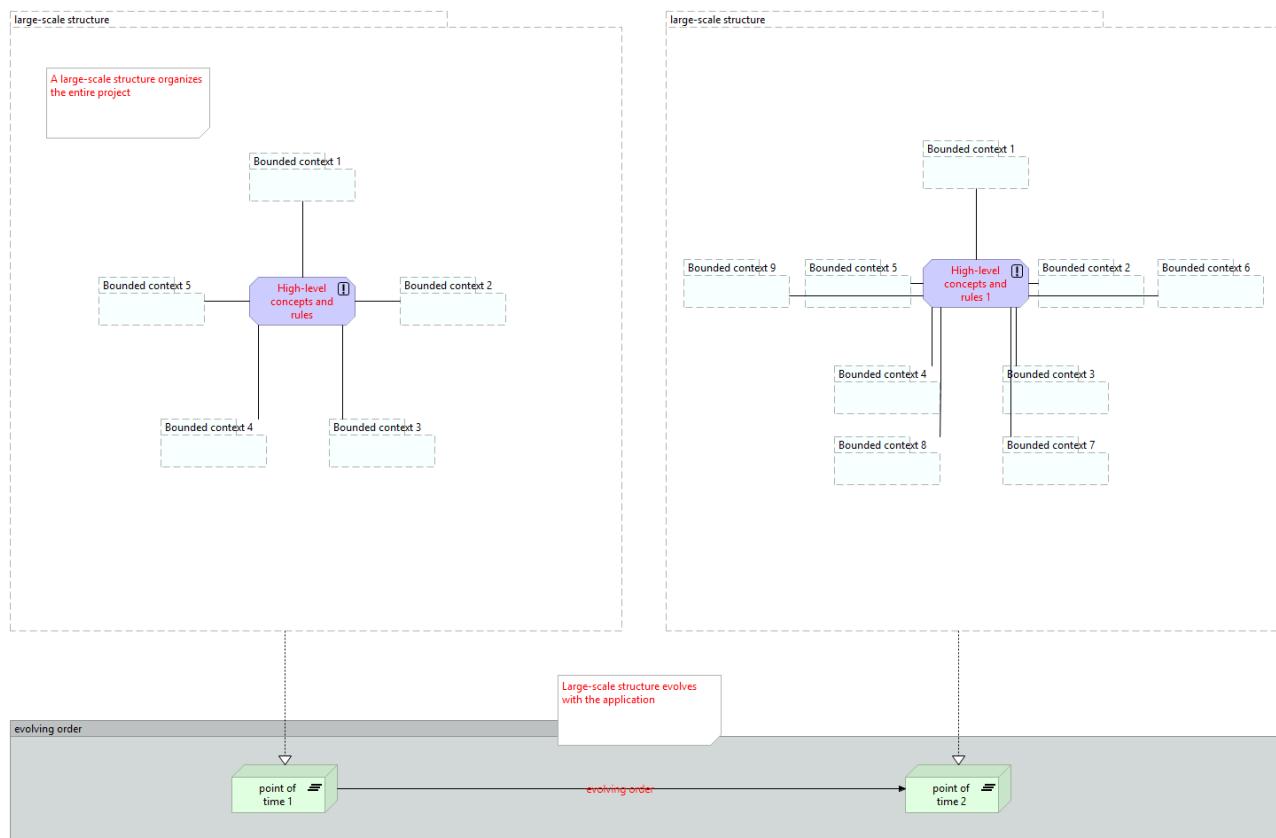
# LARGE-SCALE STRUCTURE

DOMAIN DRIVEN DESIGN

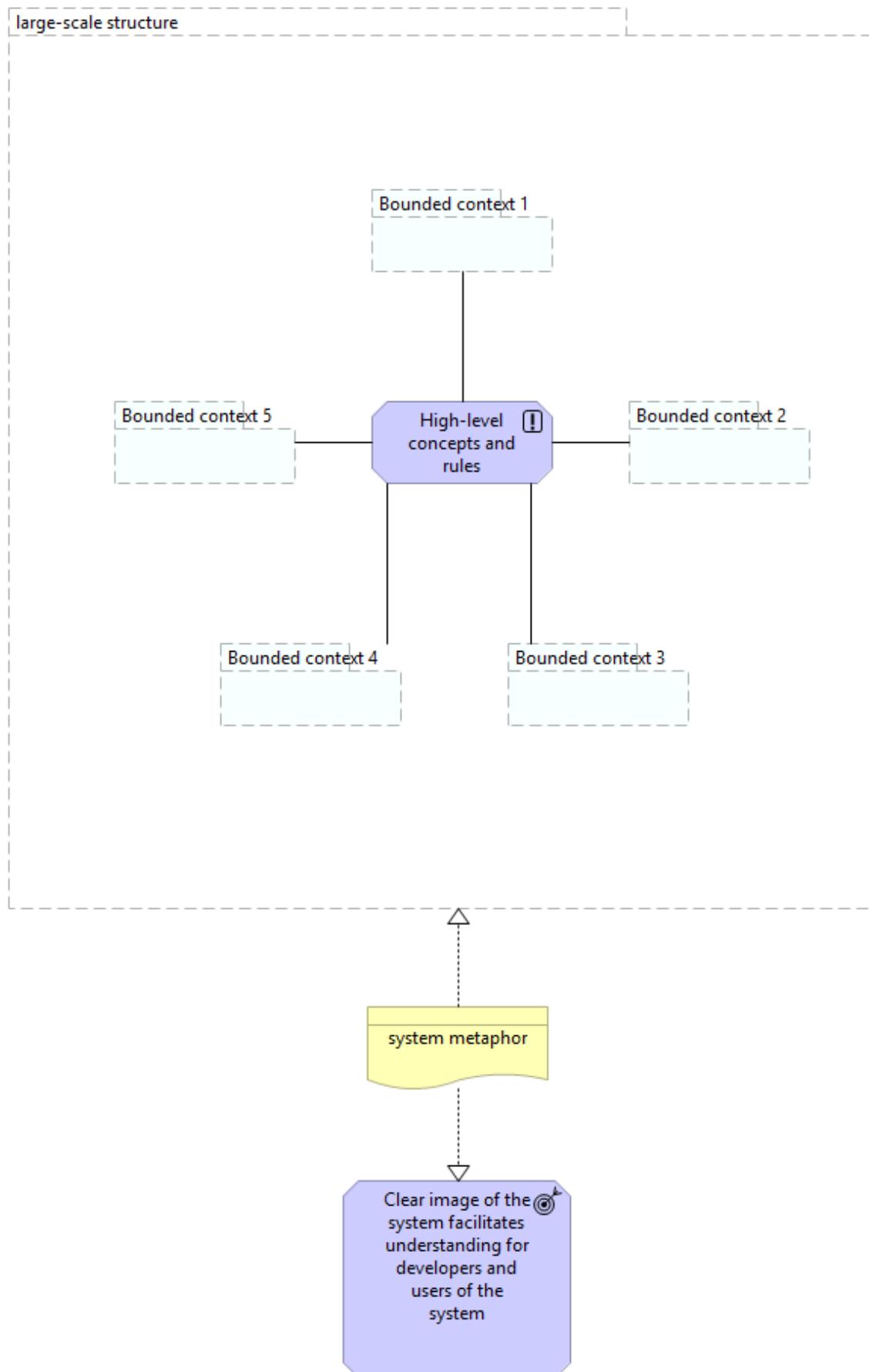
Eric Evans



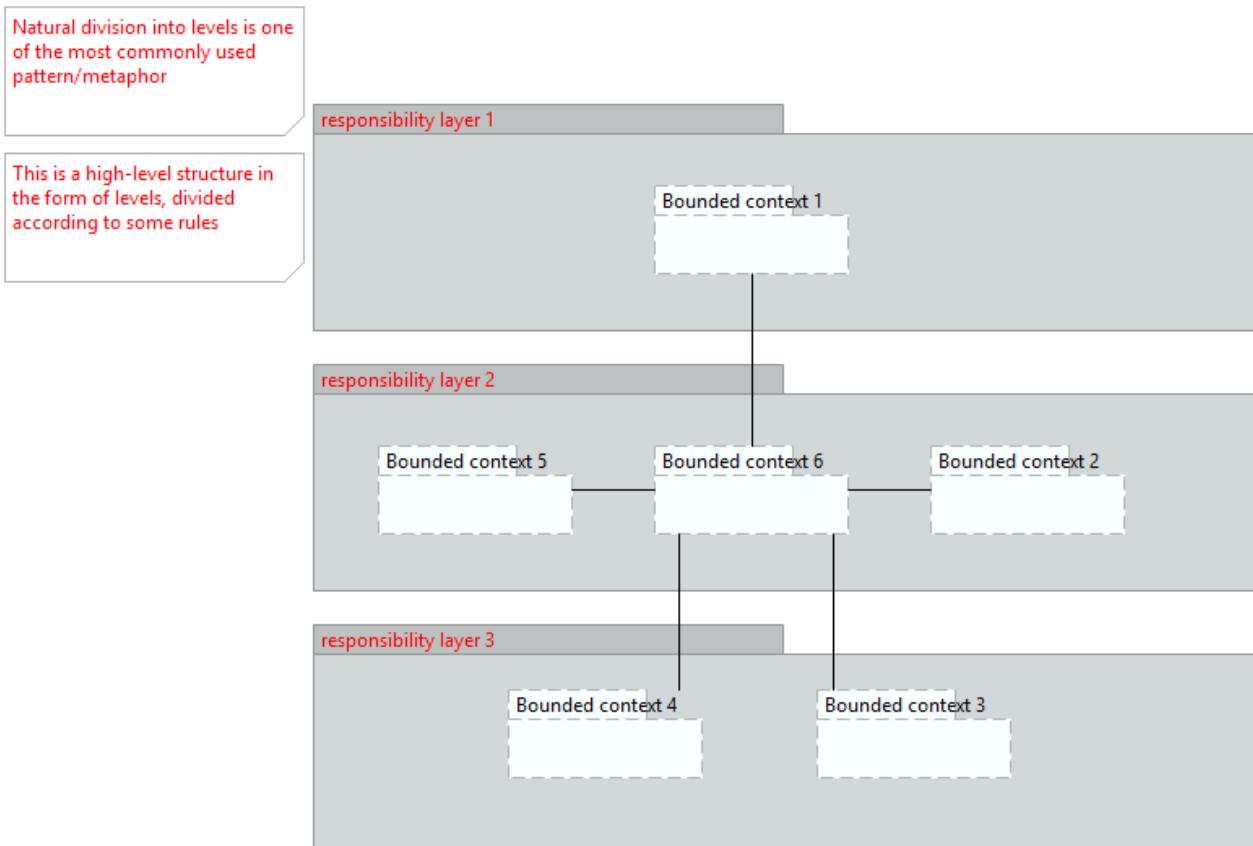
# EVOLVING ORDER



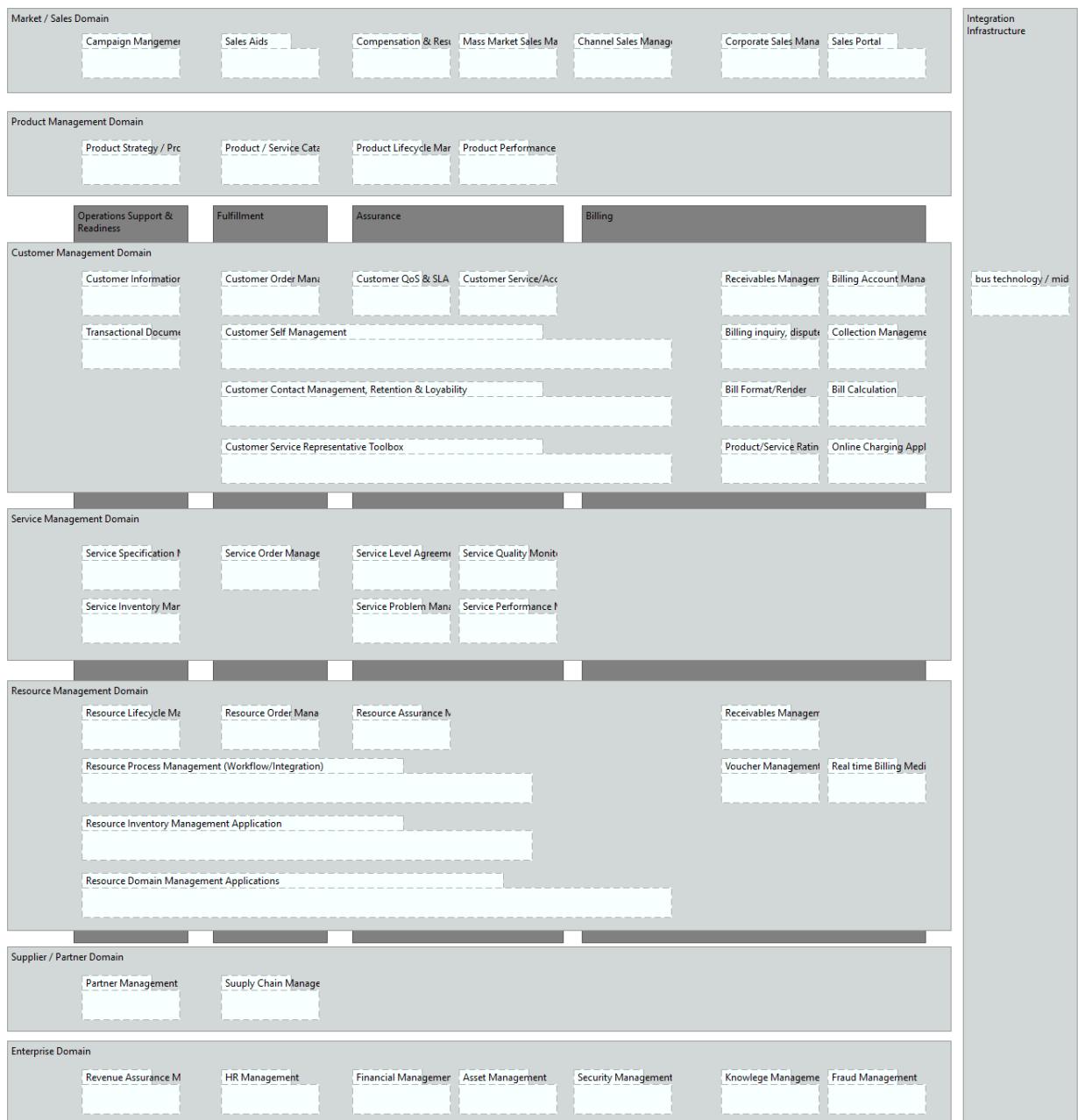
# SYSTEM METAPHOR



# 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

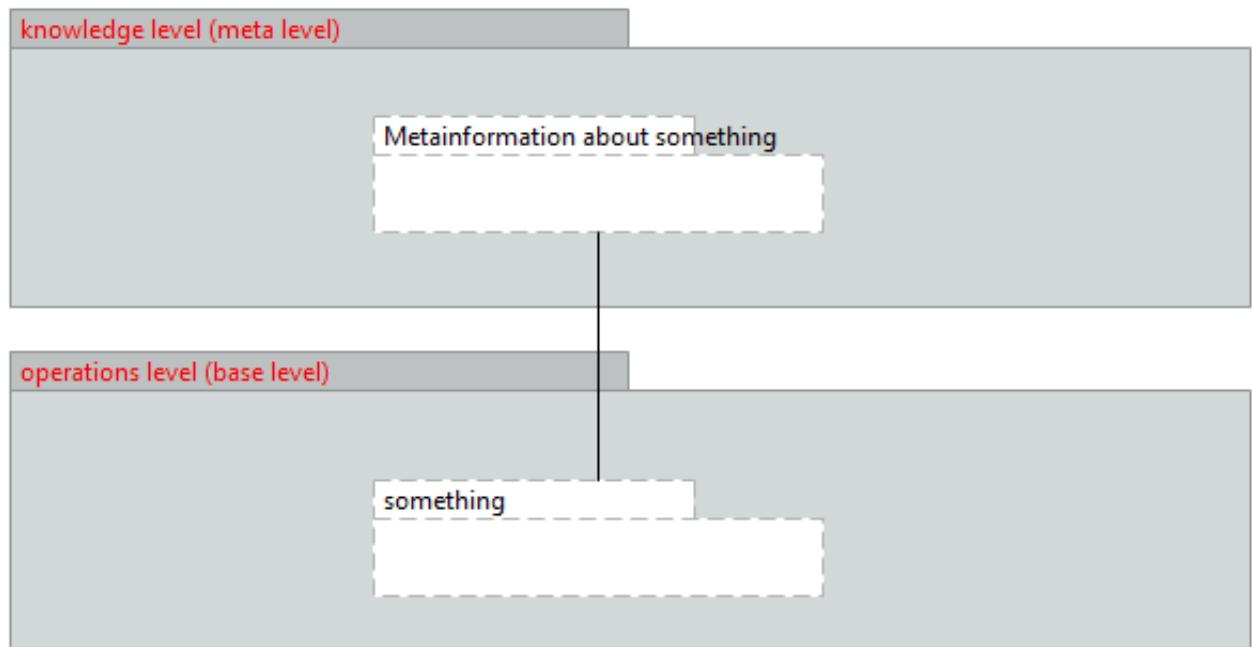
Bounded context 10

Service

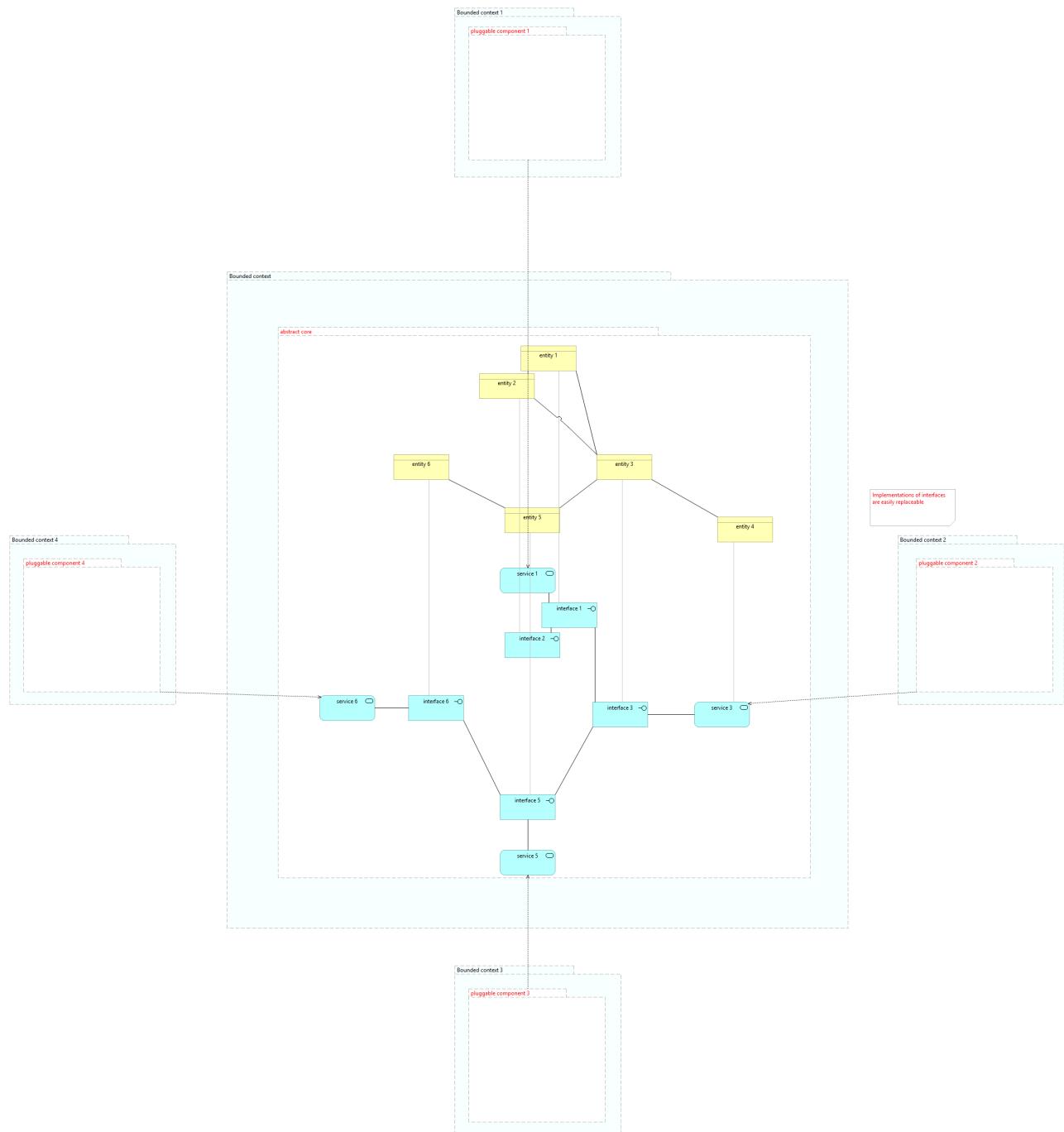
Bounded context 11



# KNOWLEDGE LEVEL



# PLUGGABLE COMPONENT FRAMEWORK



# ADDITIONAL PATTERNS

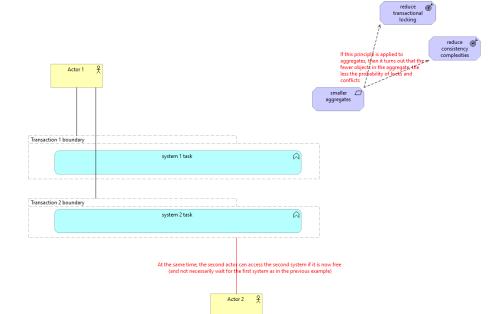
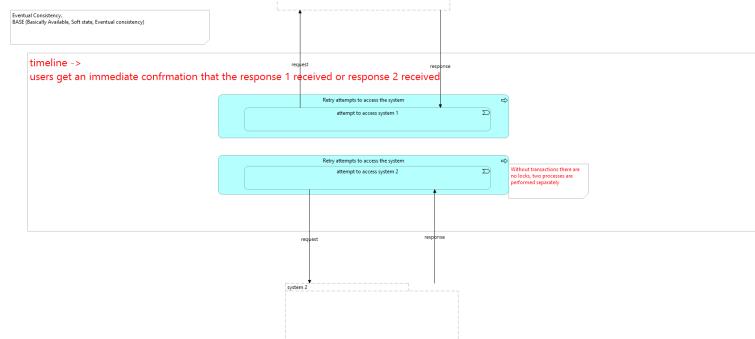
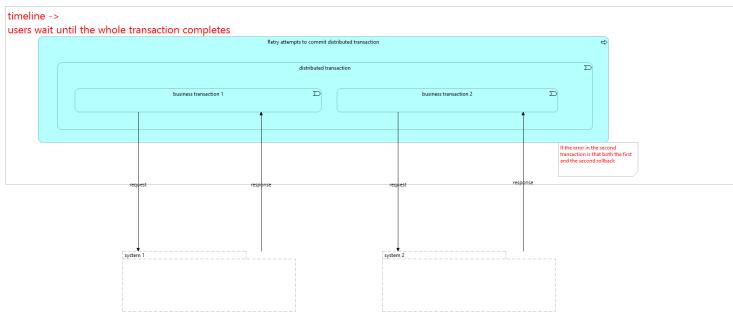
DOMAIN DRIVEN DESIGN

Eric Evans

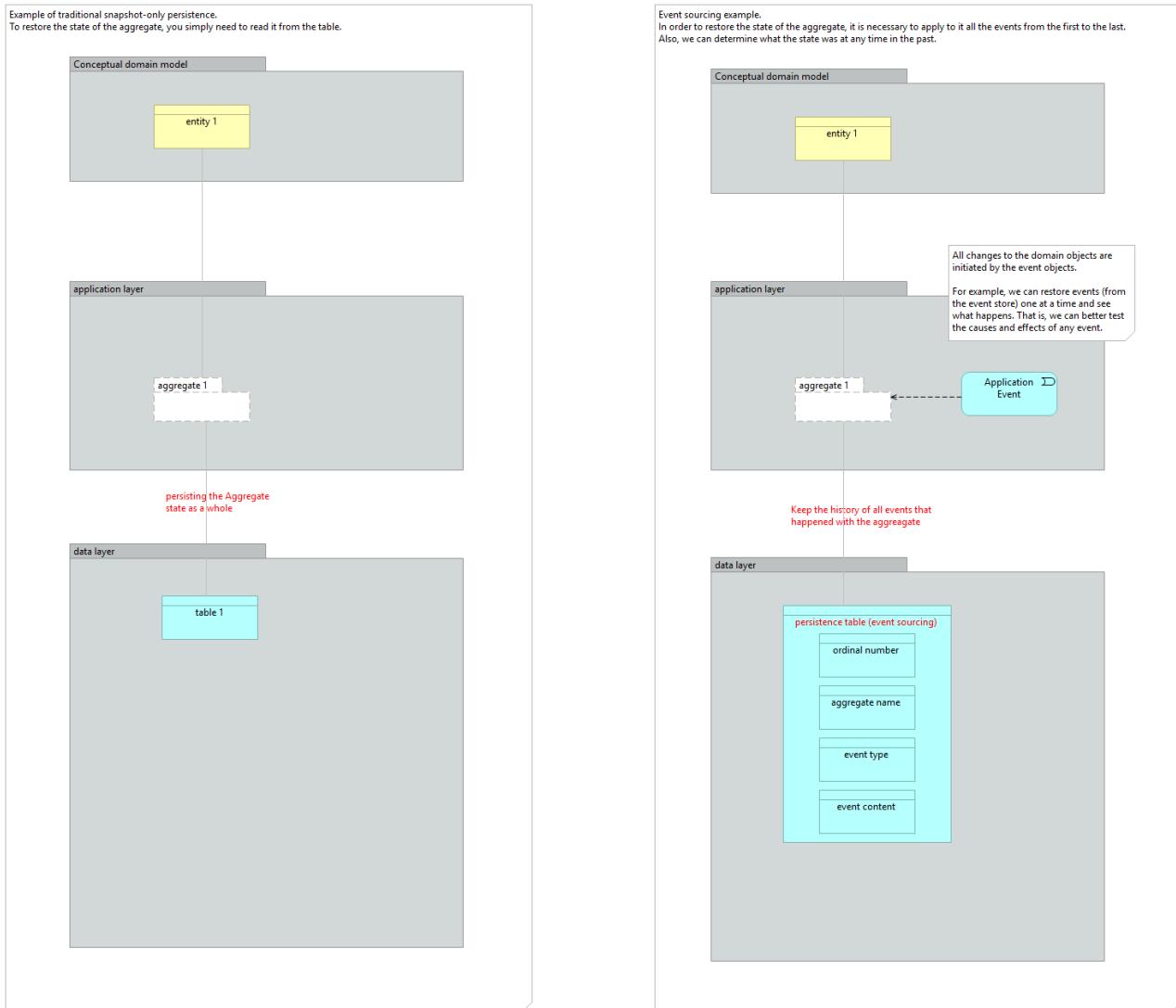


# TYPES OF CONSISTENCY

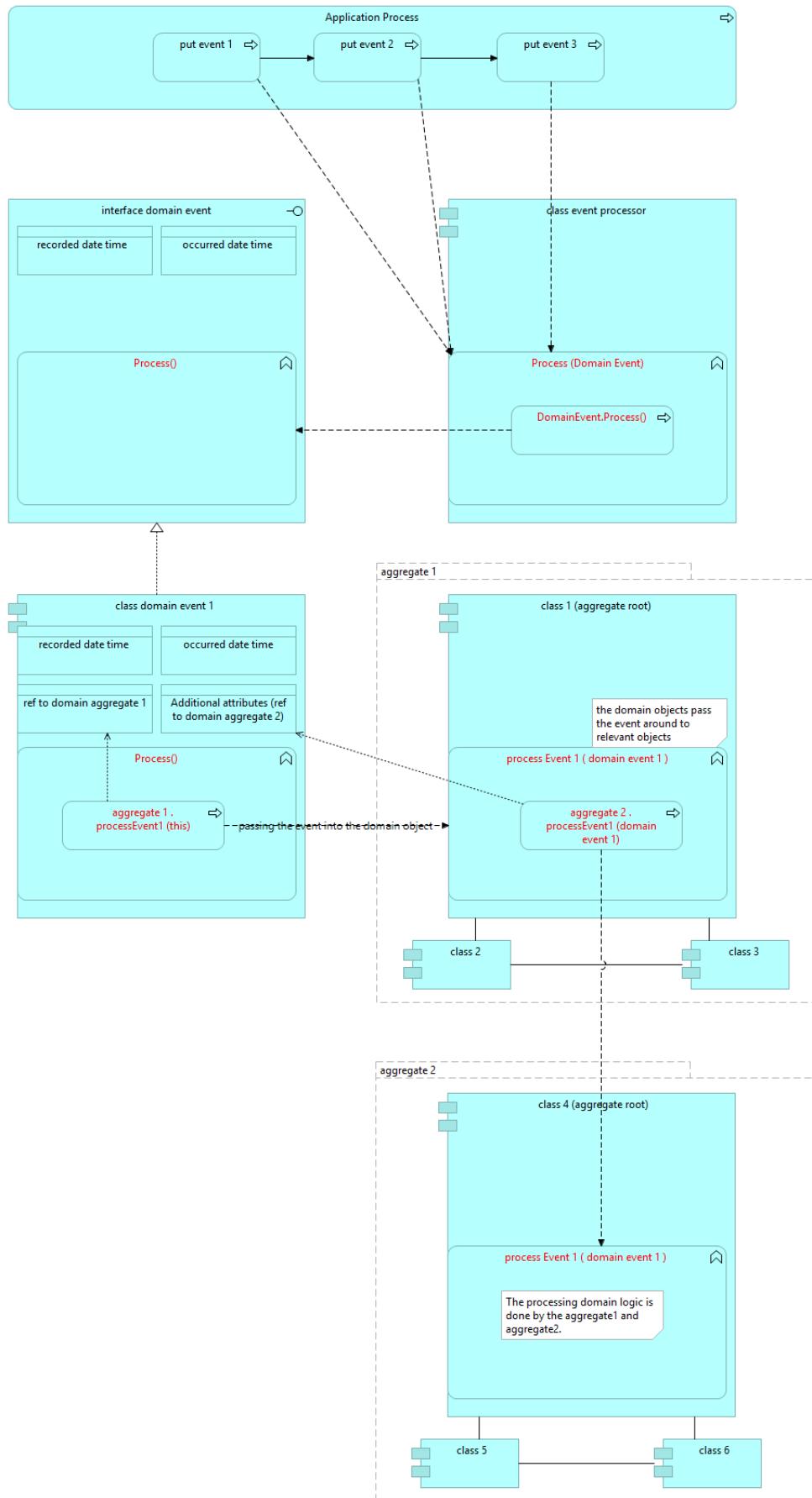
**Transactional consistency:**  
When a transaction is finished the system is in a consistent state.  
C� A distributed transaction is a workflow.  
The more objects involved in a distributed transaction, the greater the risk of potential loss and conflicts.



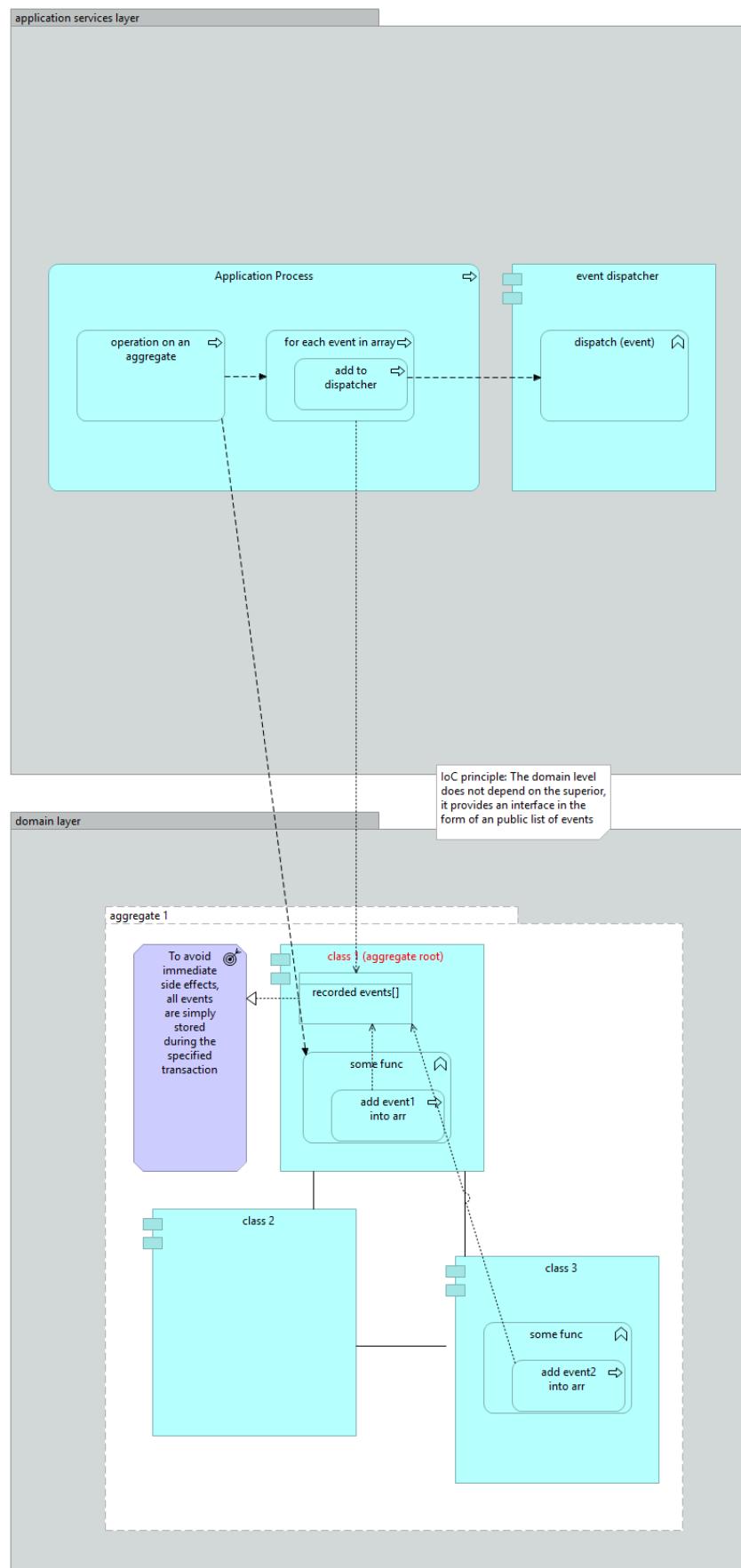
# EVENT SOURCING



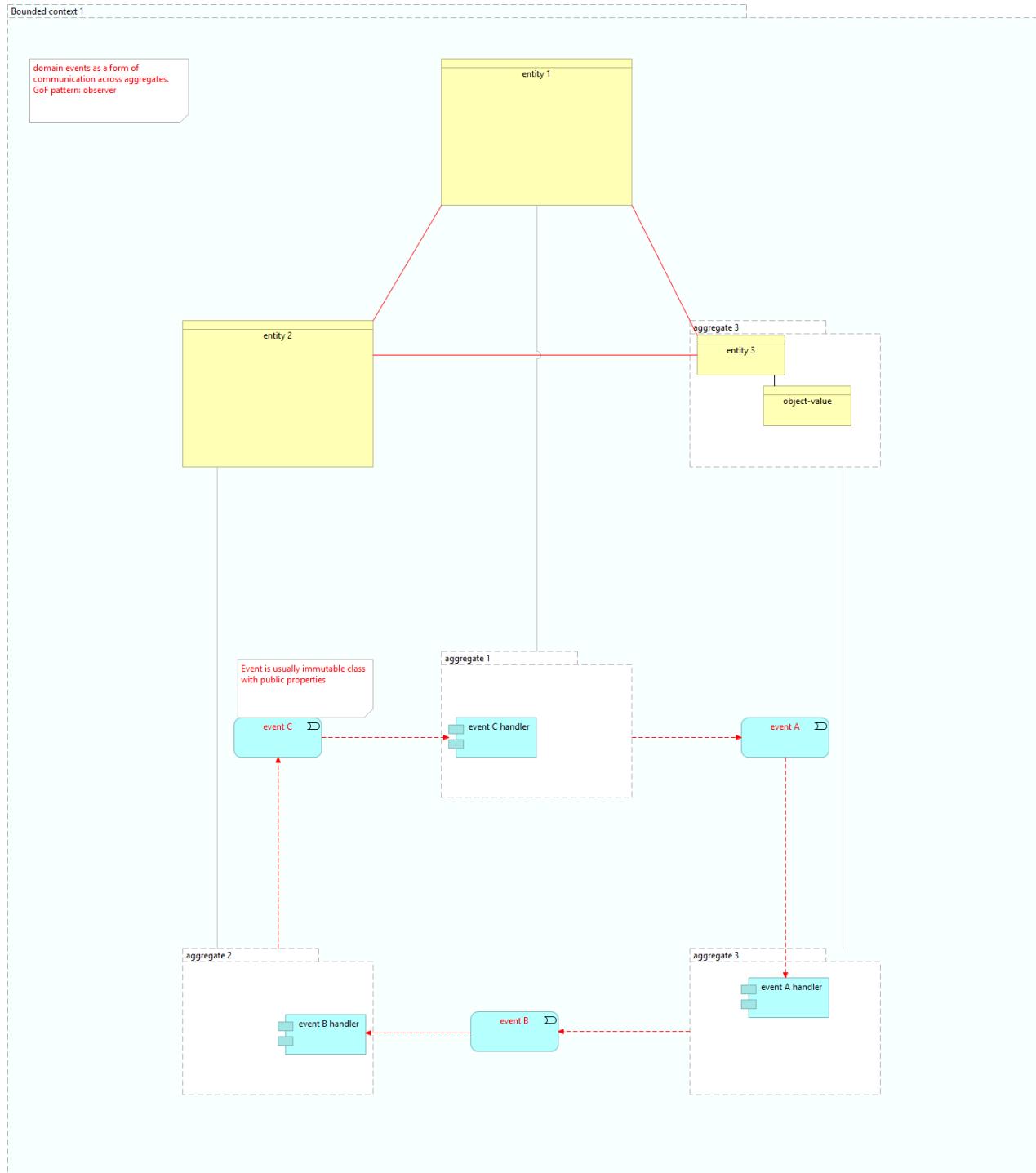
# EVENT PROCESSOR



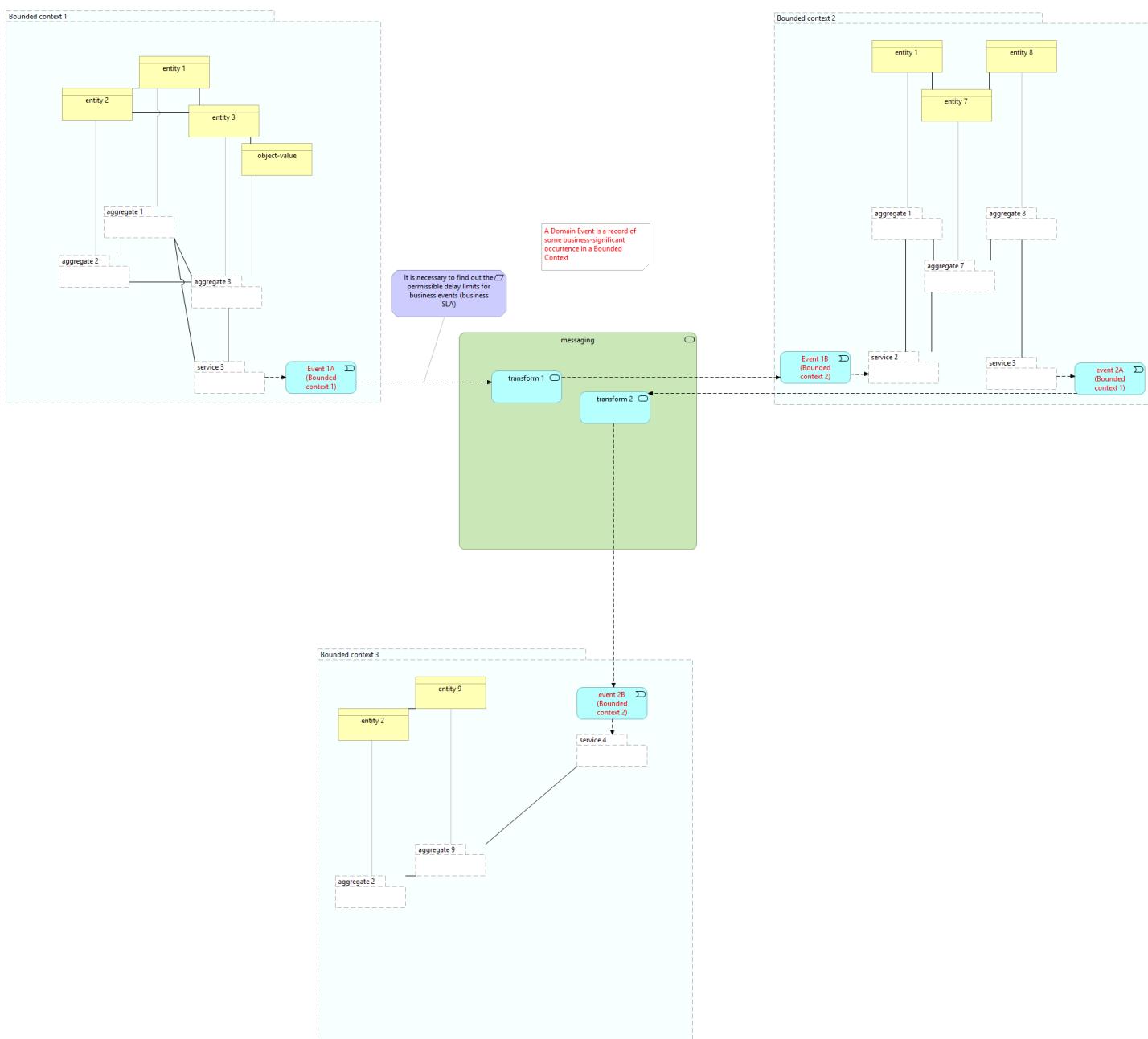
# EVENT DISPATCHER



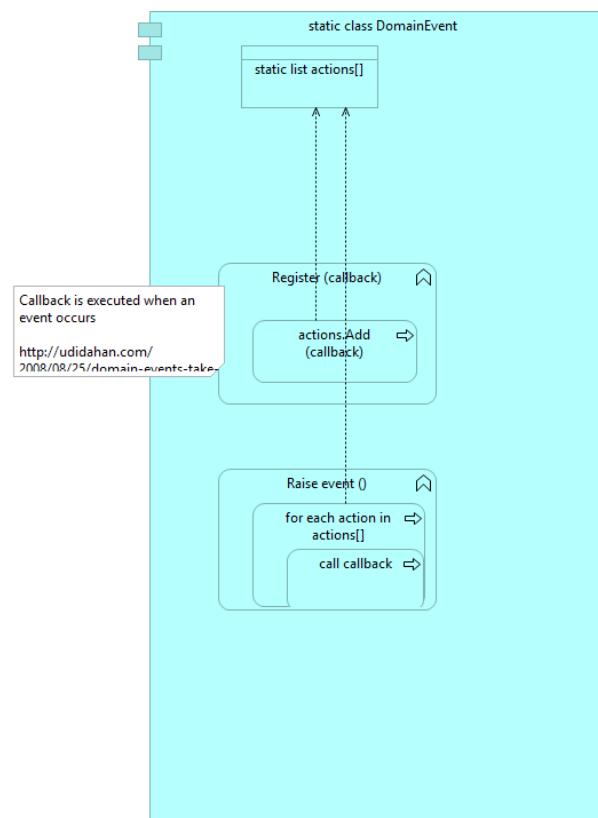
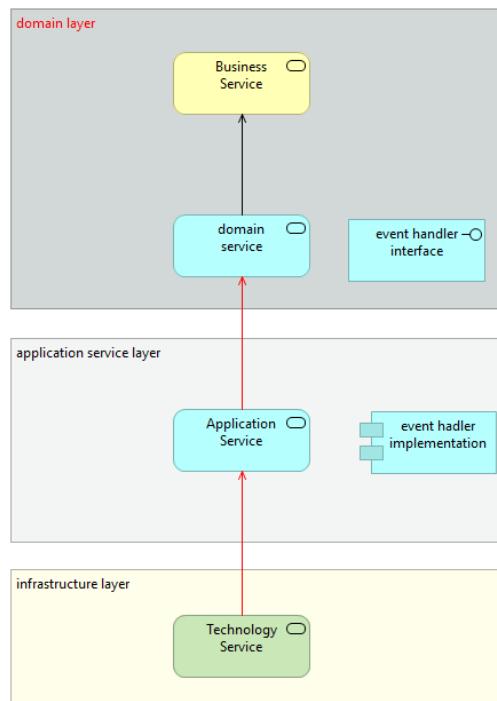
# INTERNAL DOMAIN EVENTS



# EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS

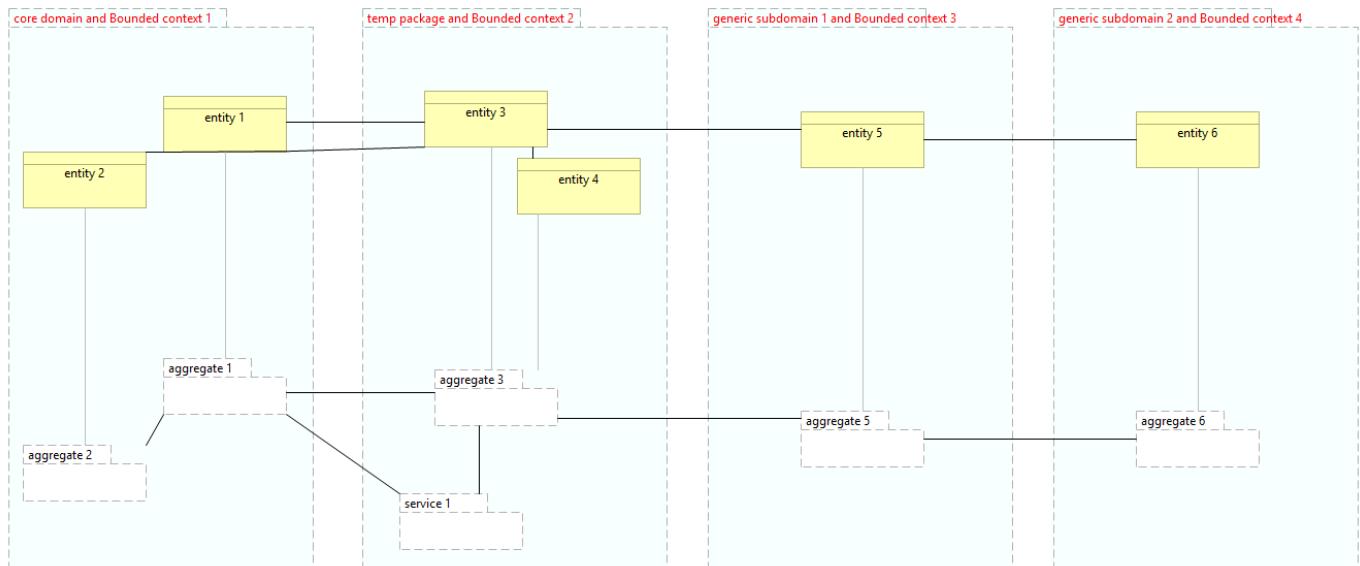


# STATIC DOMAIN EVENTS CLASS

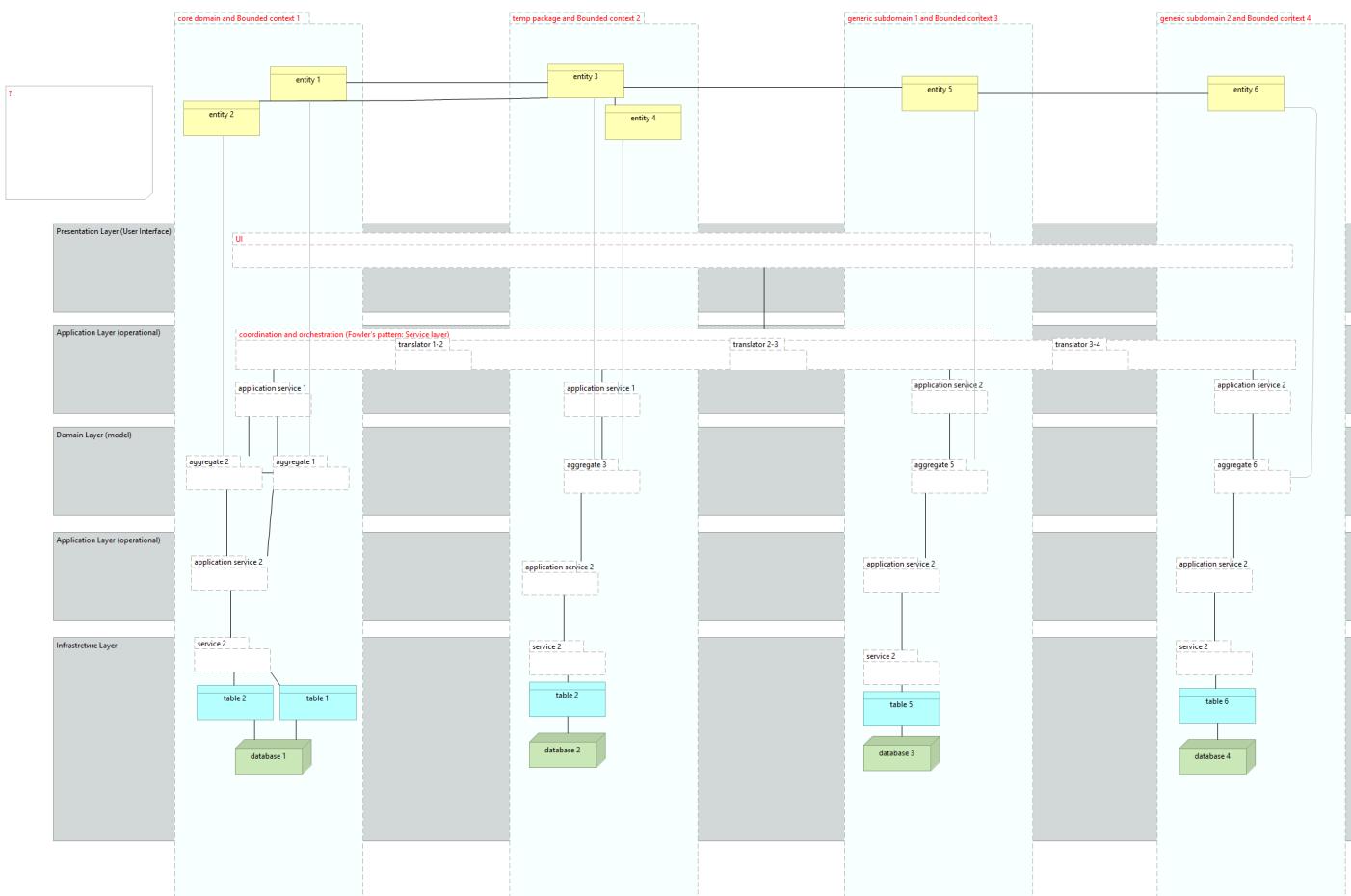


# ONE SUBDOMAIN PER BOUNDED CONTEXT

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

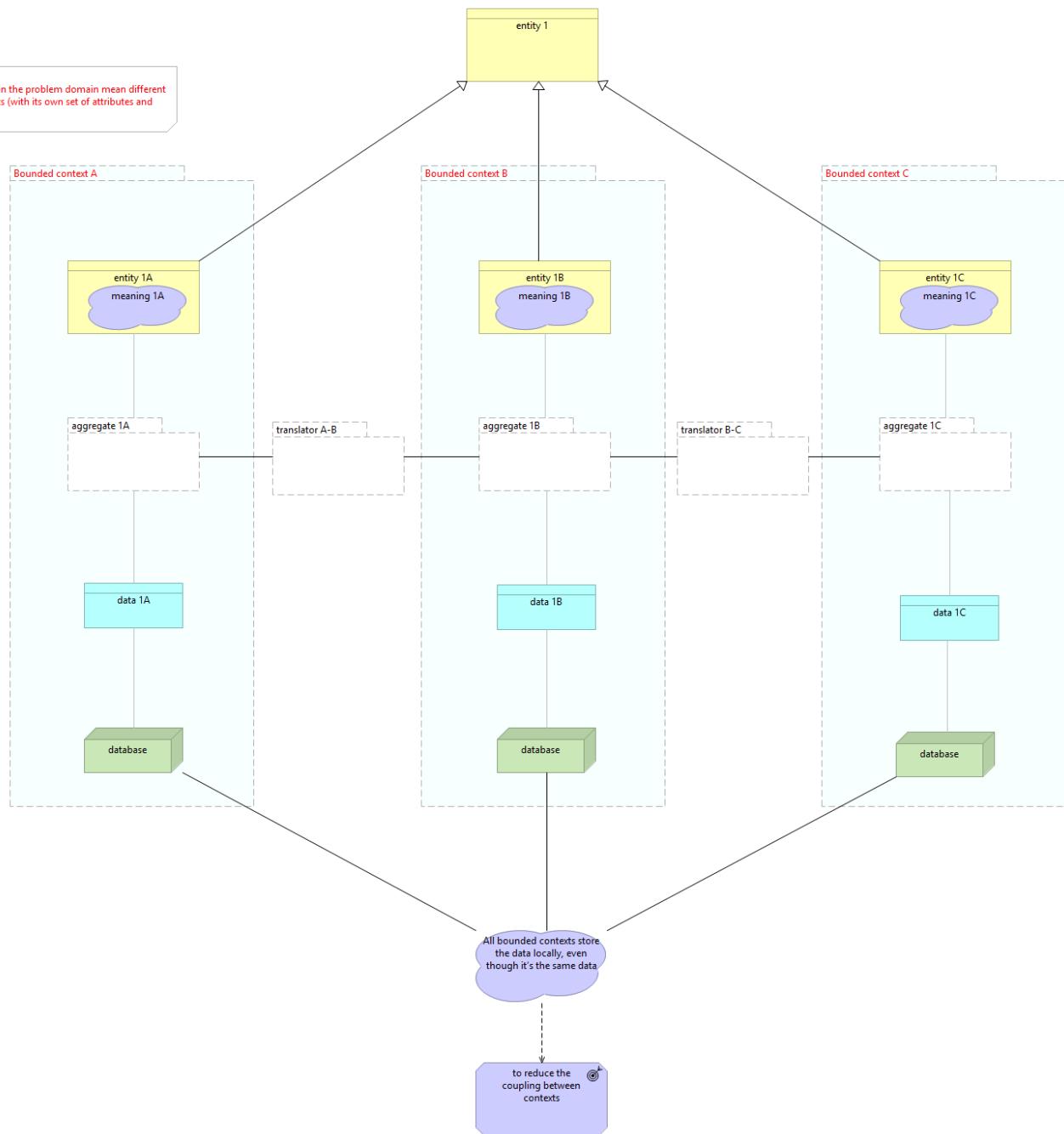


# THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS

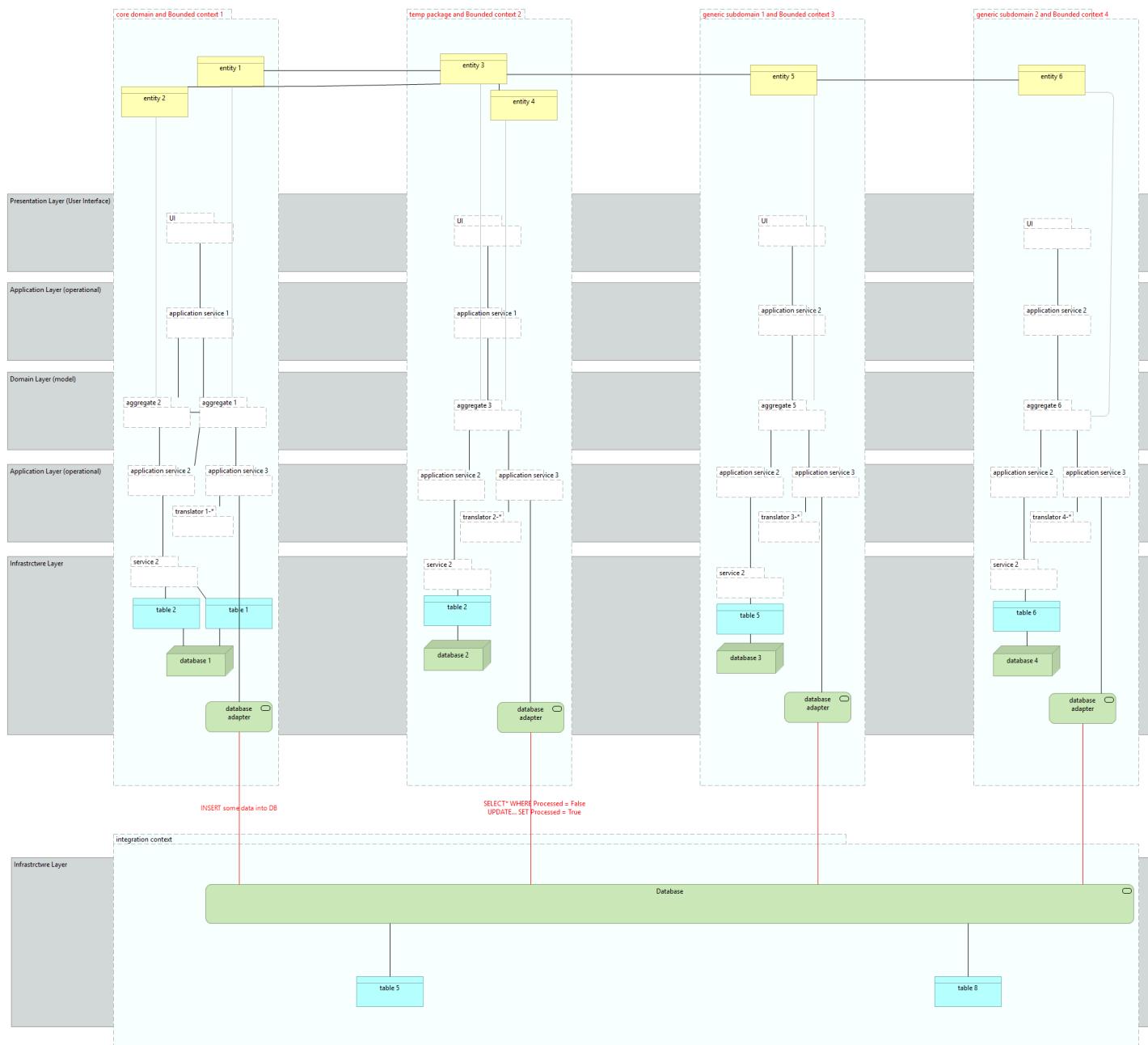


# THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS

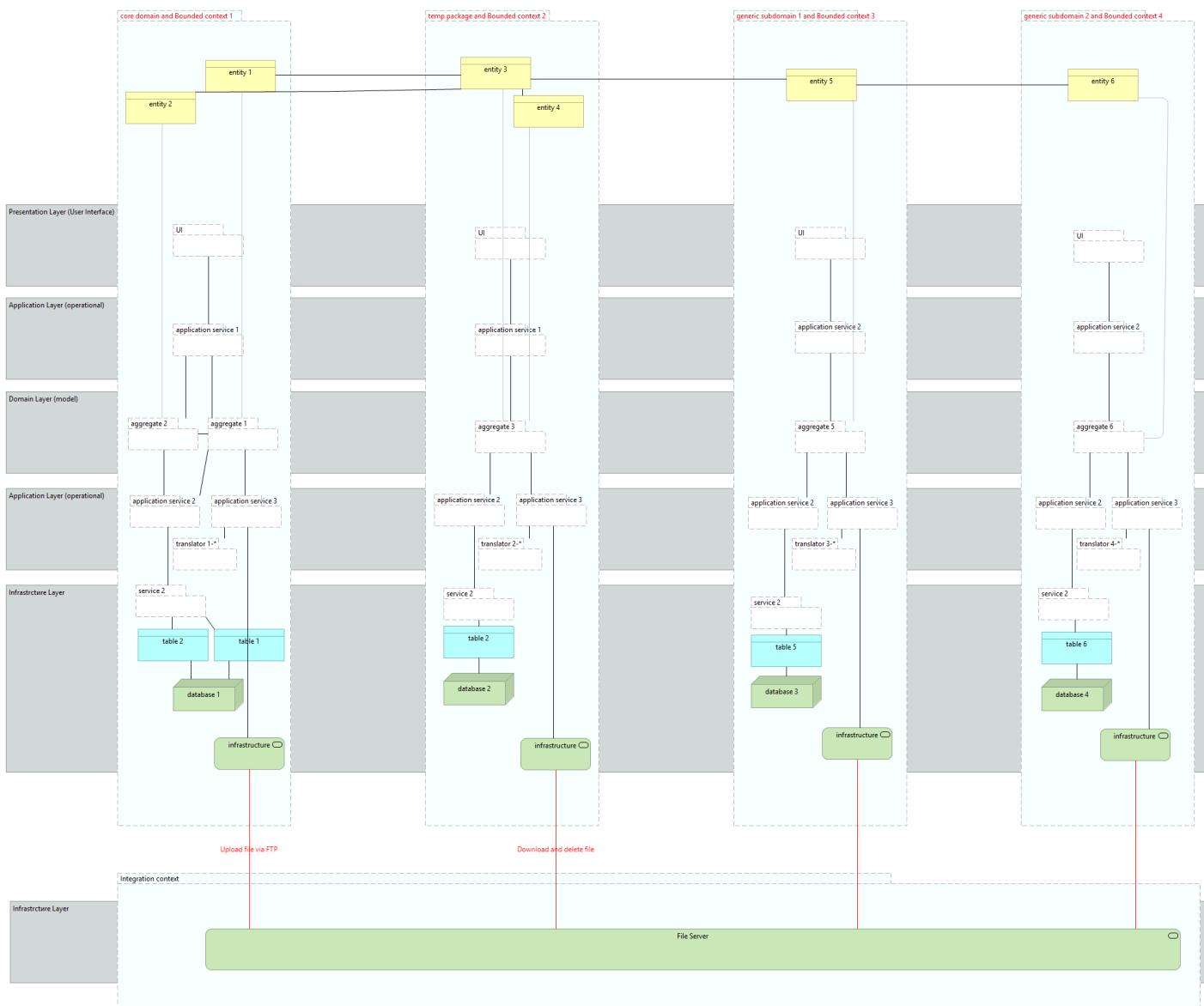
**Example**  
The same physical entity in the problem domain mean different things in different contexts (with its own set of attributes and aspects of behavior).



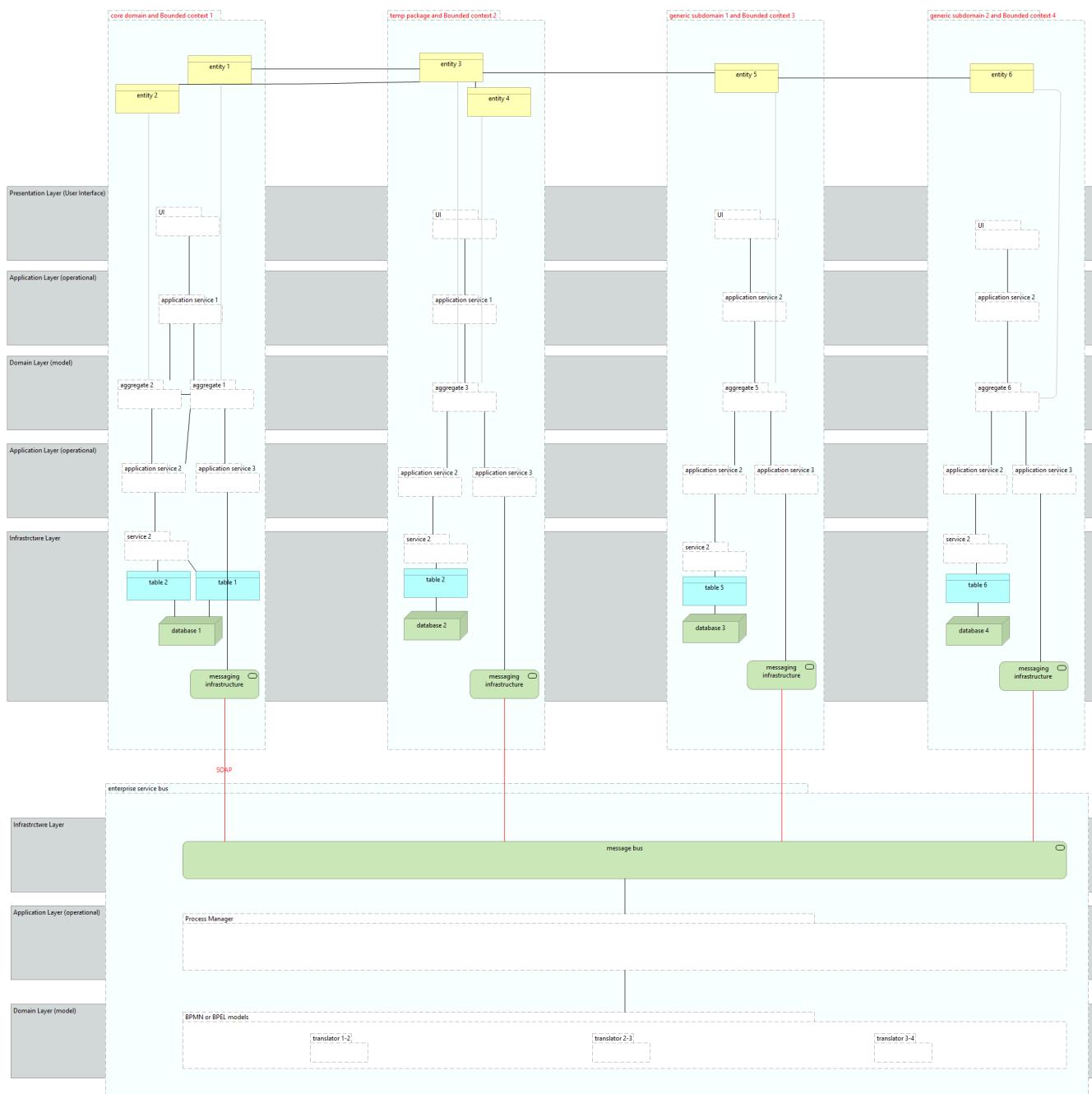
# INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE



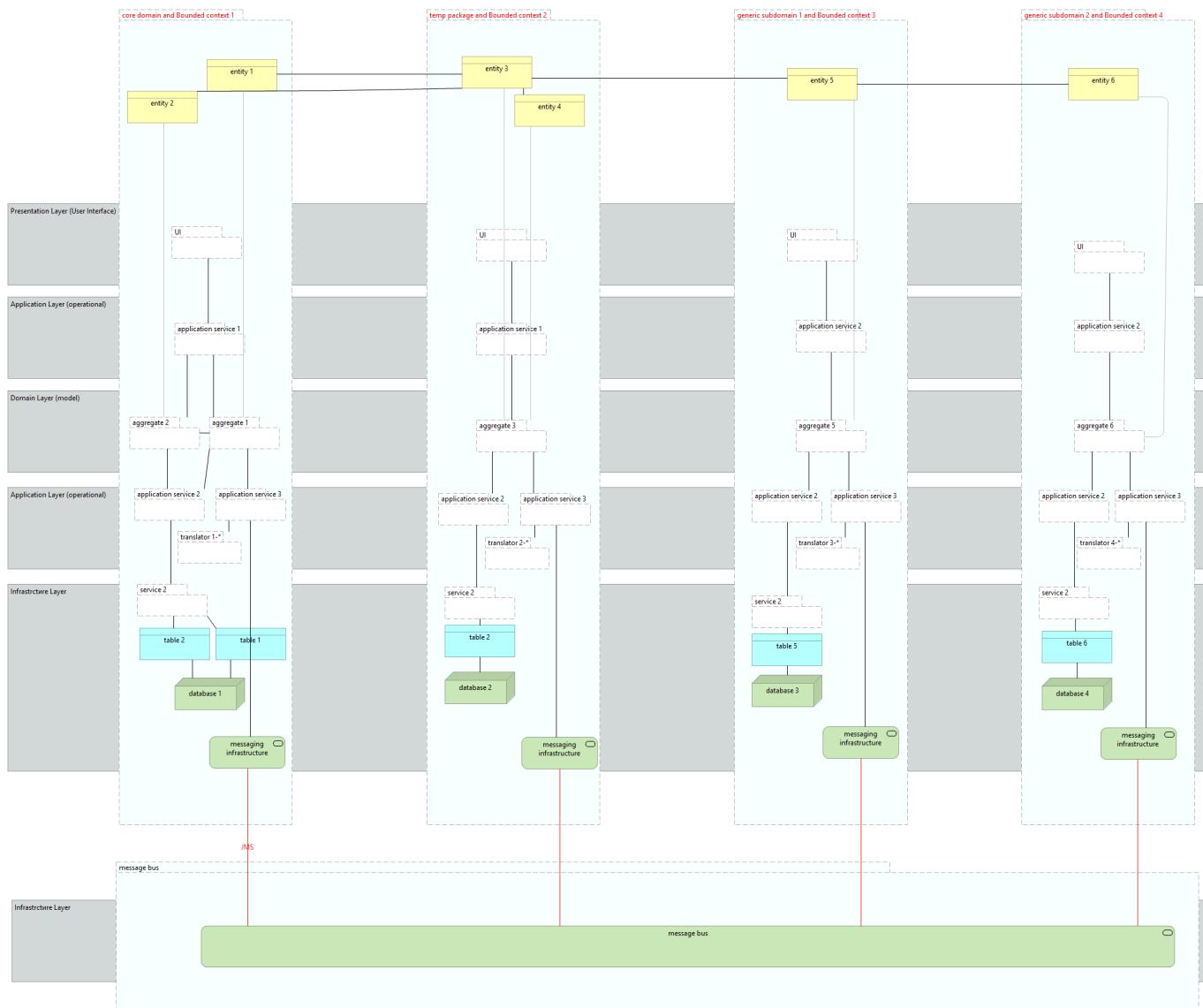
# INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES



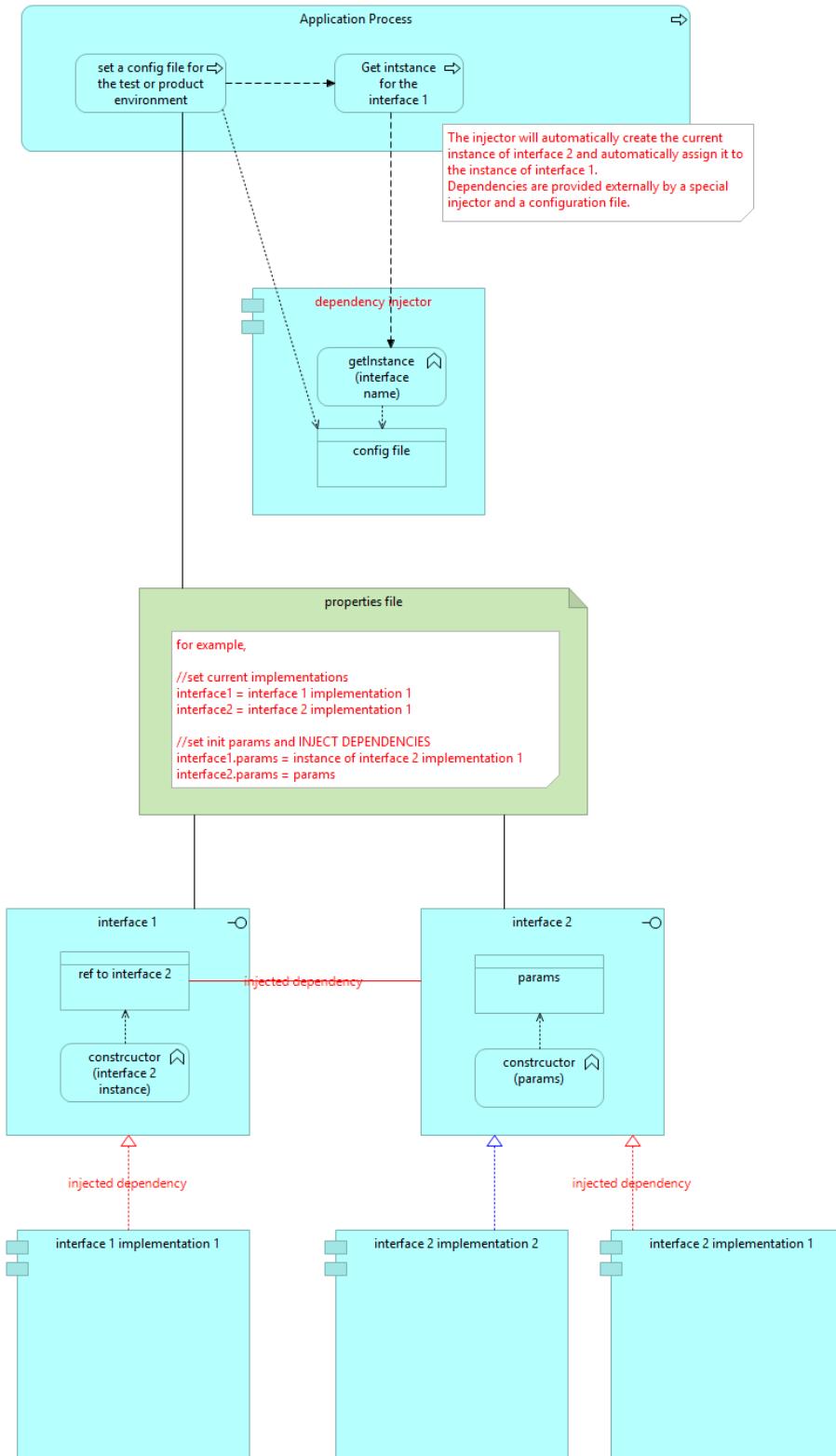
# INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS



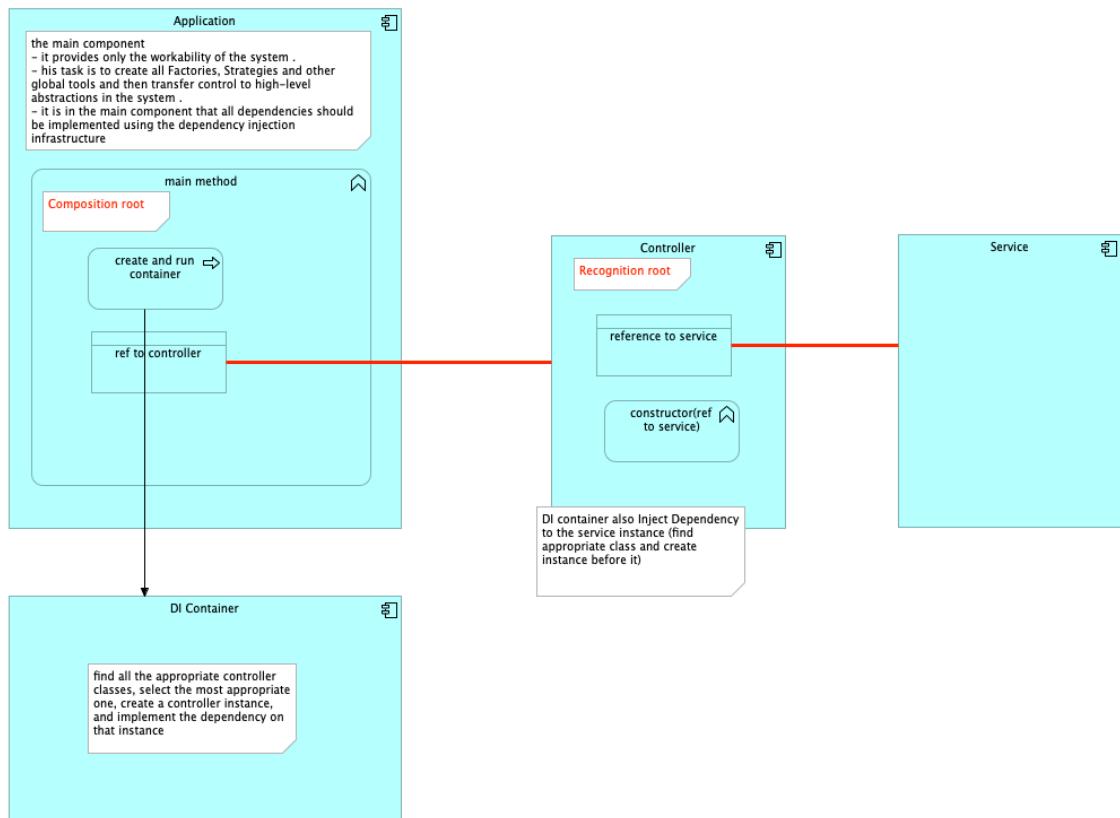
# INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE



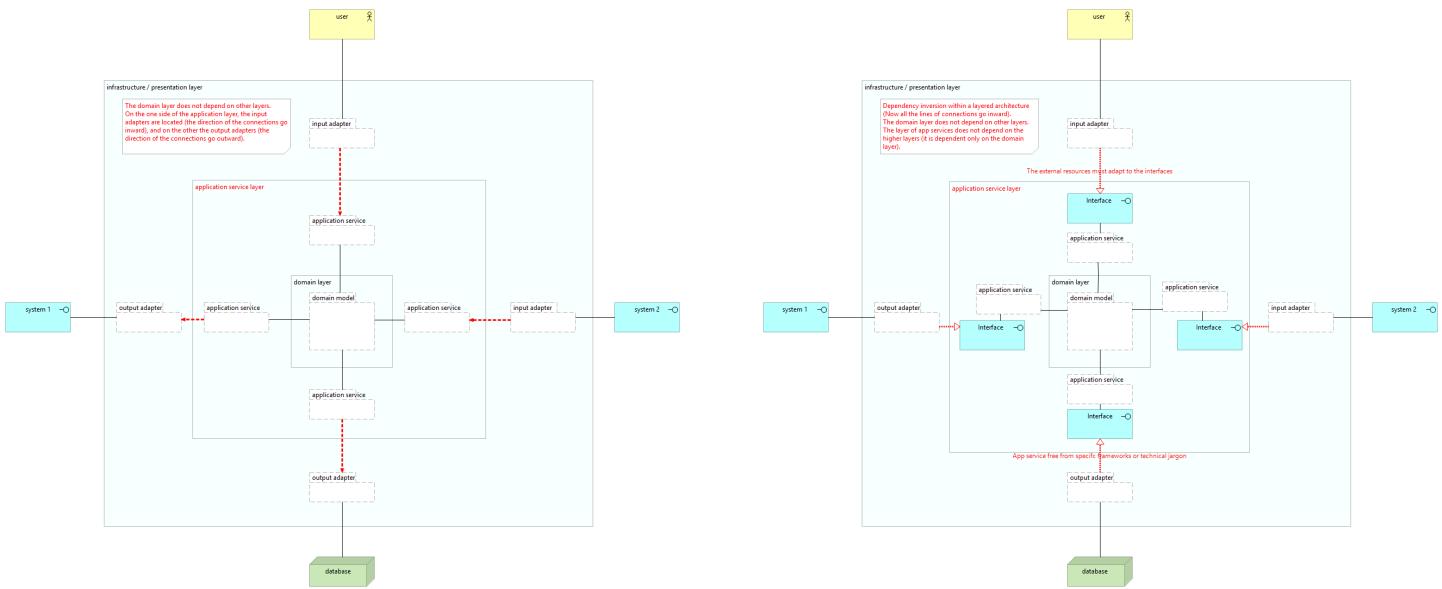
# DEPENDENCY INJECTION



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

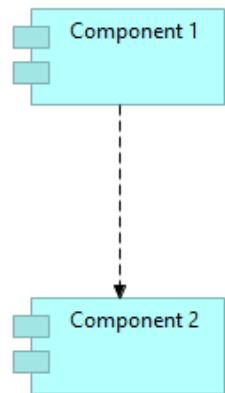


# INVERSION OF CONTROL

IoC

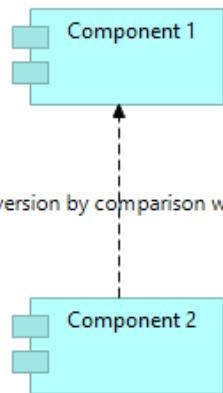
Inversion of control is about who initiates messages.

Variant 1



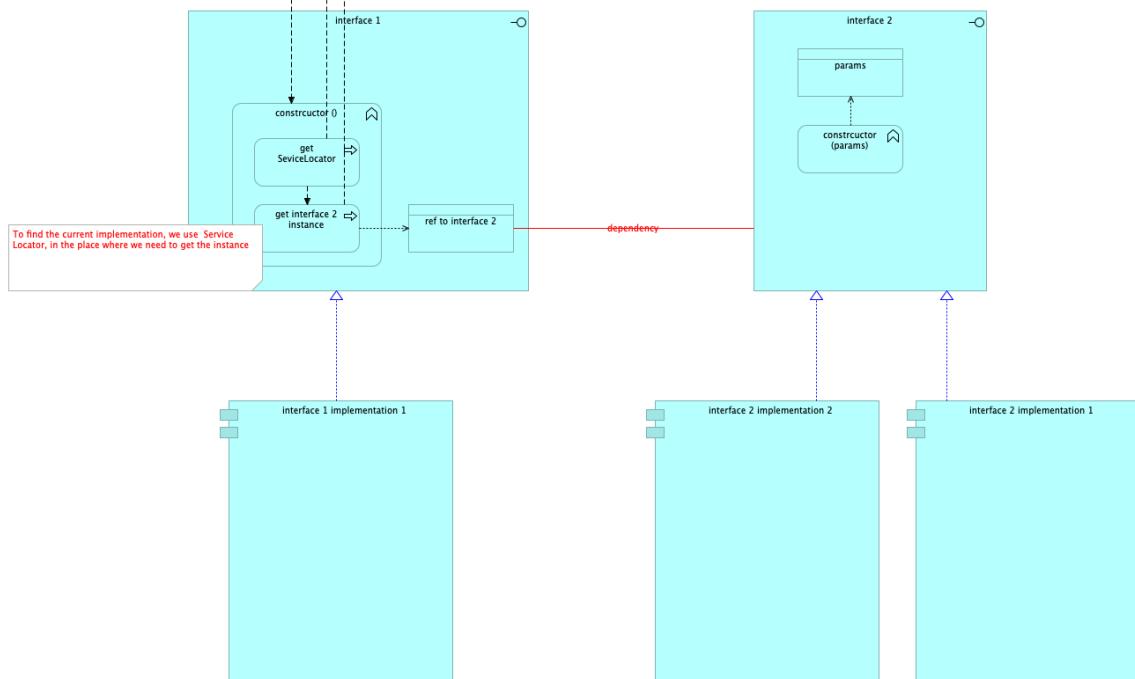
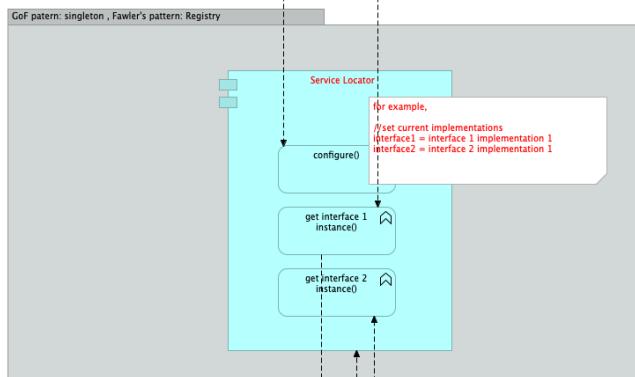
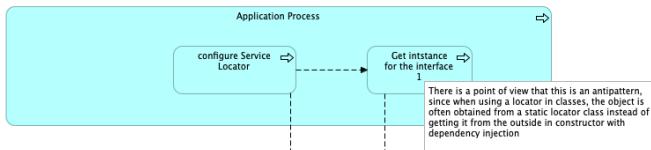
Variant 2

Here the control inversion by comparison with the previous case



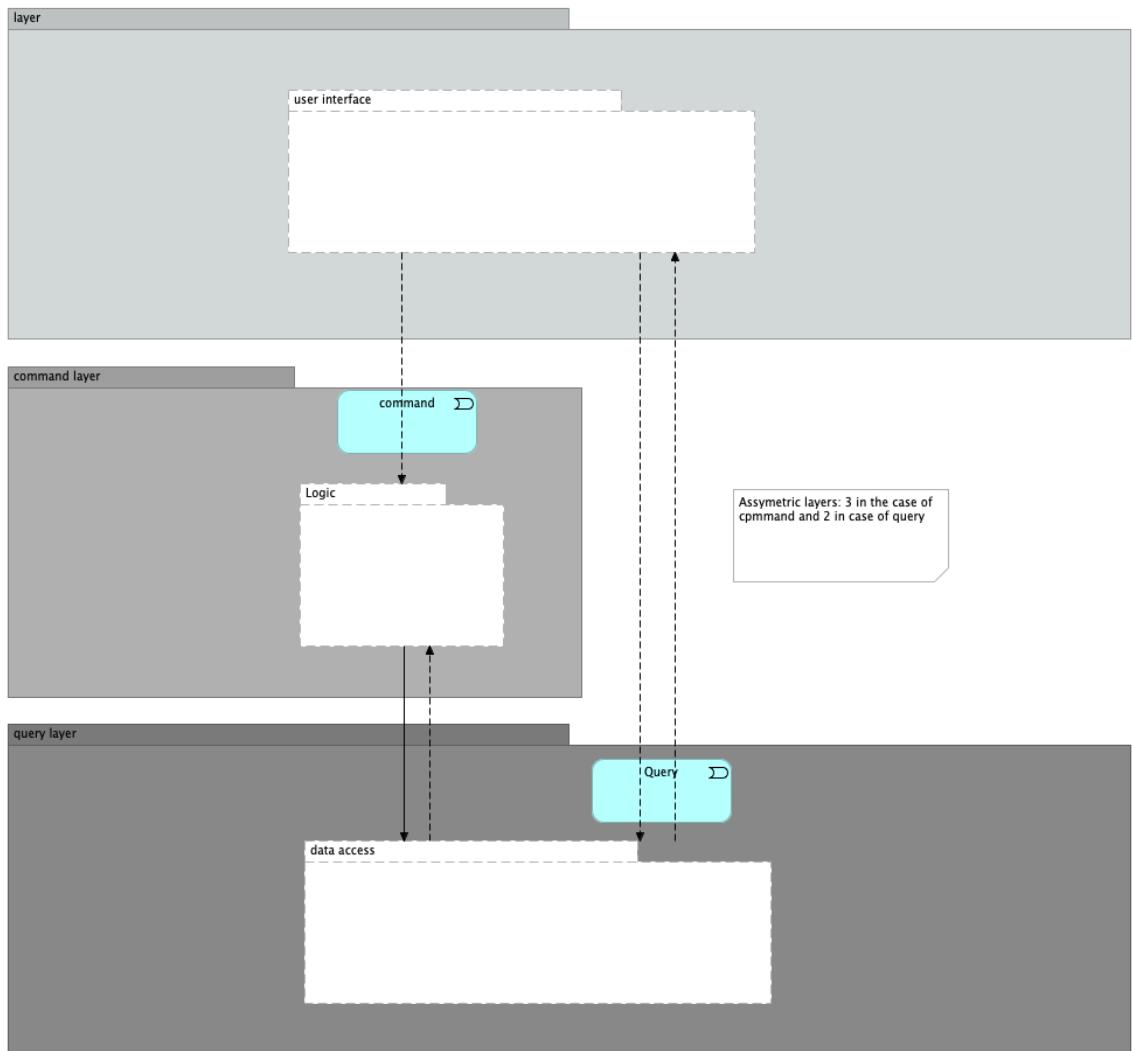
# SERVICE LOCATOR

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



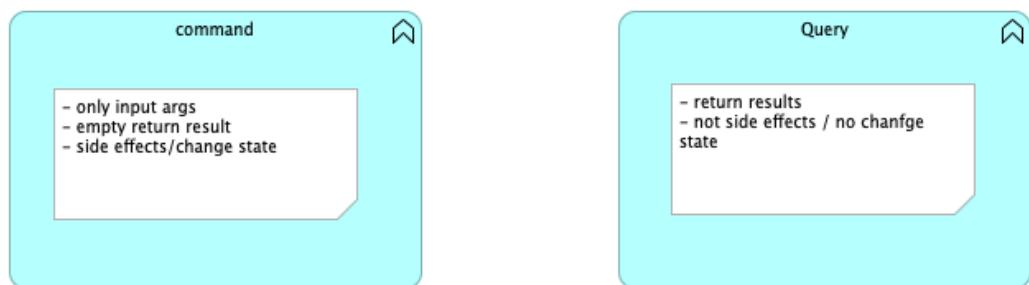
# CQRS

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

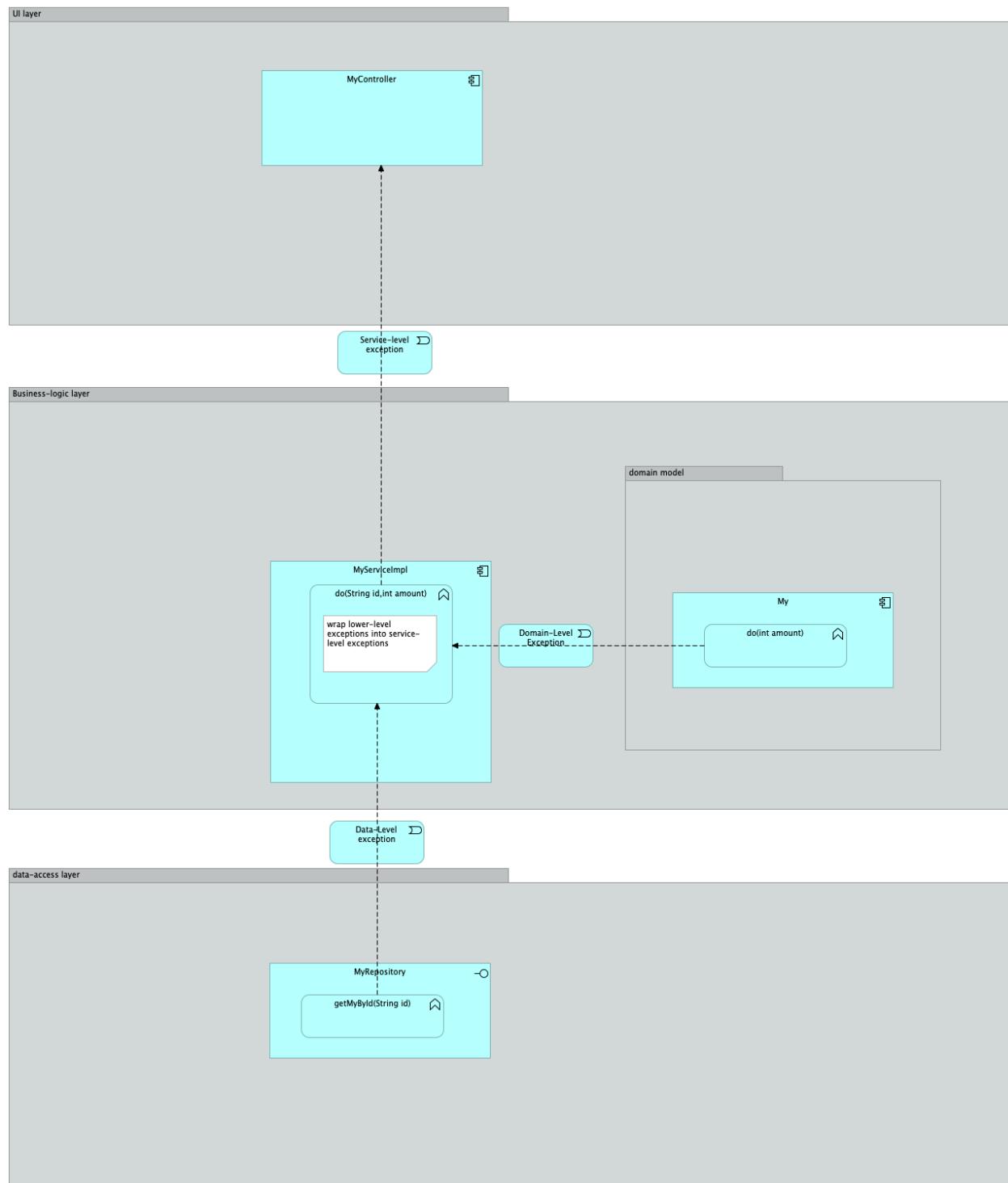


# CQS

CQS Command/Query Separation

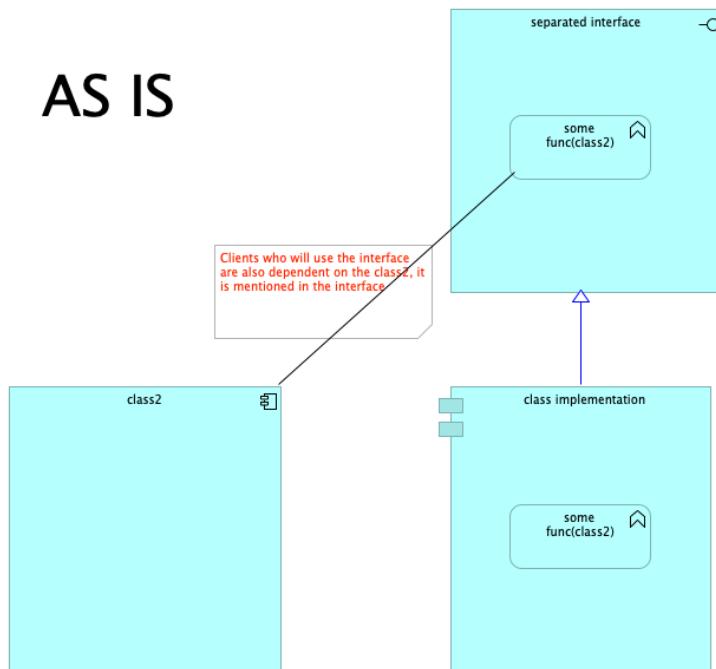


# WRAP LOW-LEVEL EXCEPTIONS

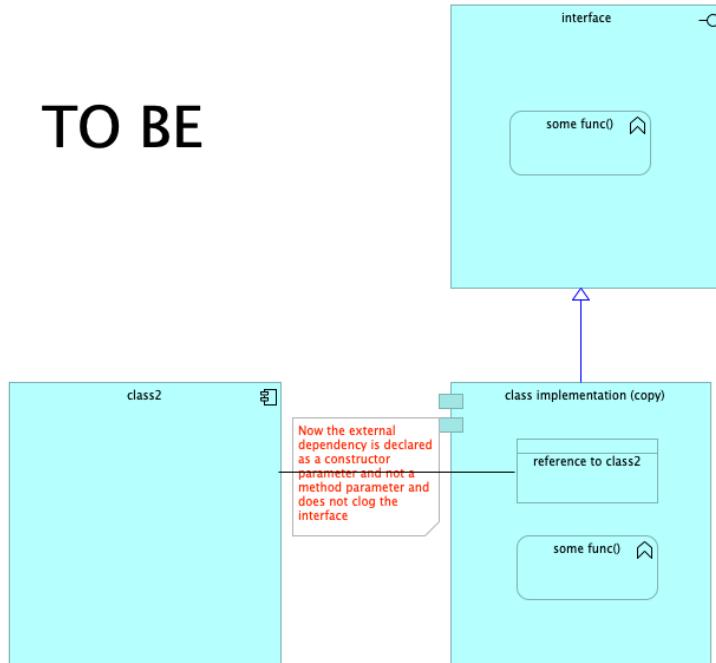


# EXTRACT DEPENDENCY FROM INTERFACE TO CONSTRUCTOR

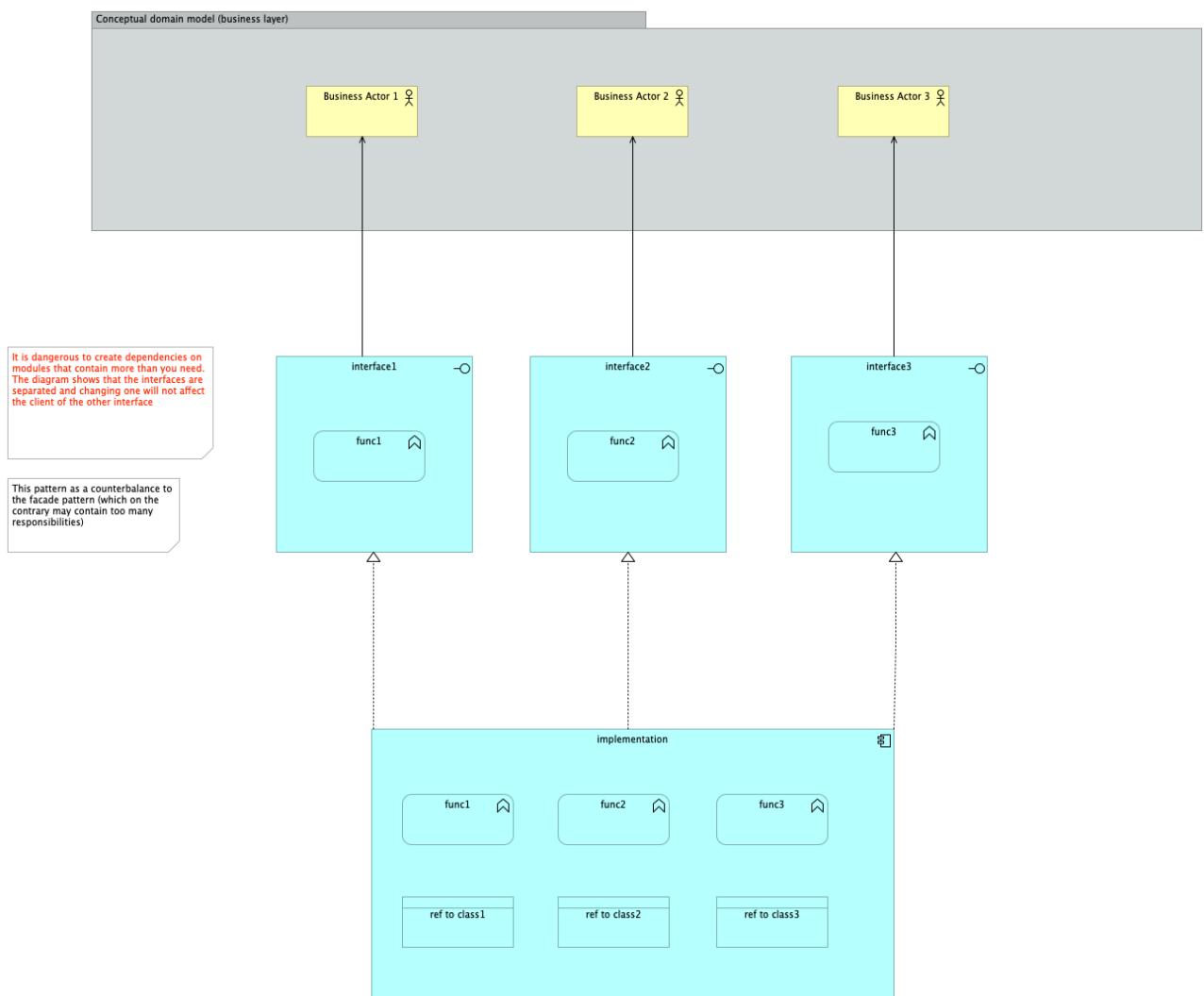
AS IS



TO BE

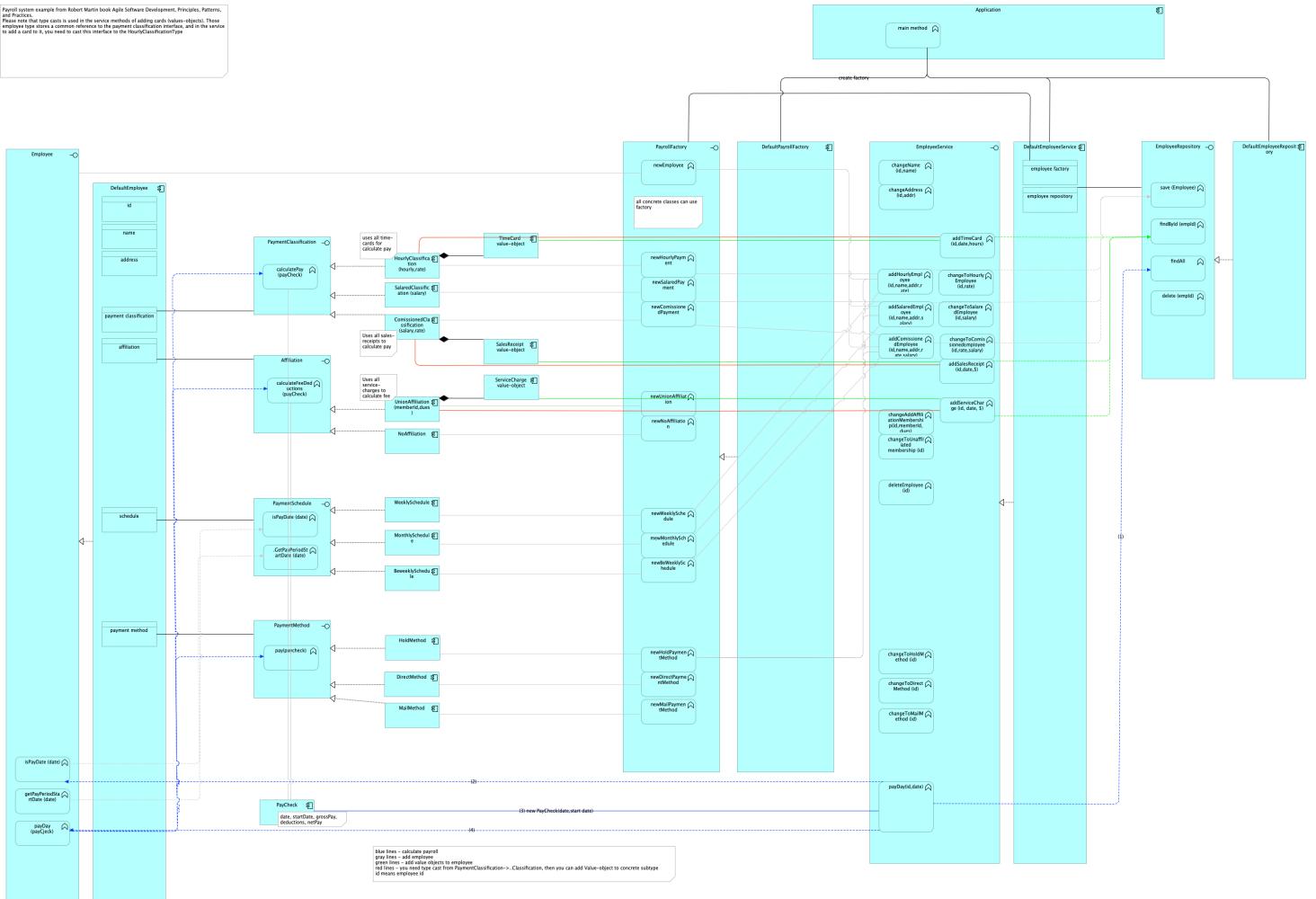


# INTERFACE SEGREGATION

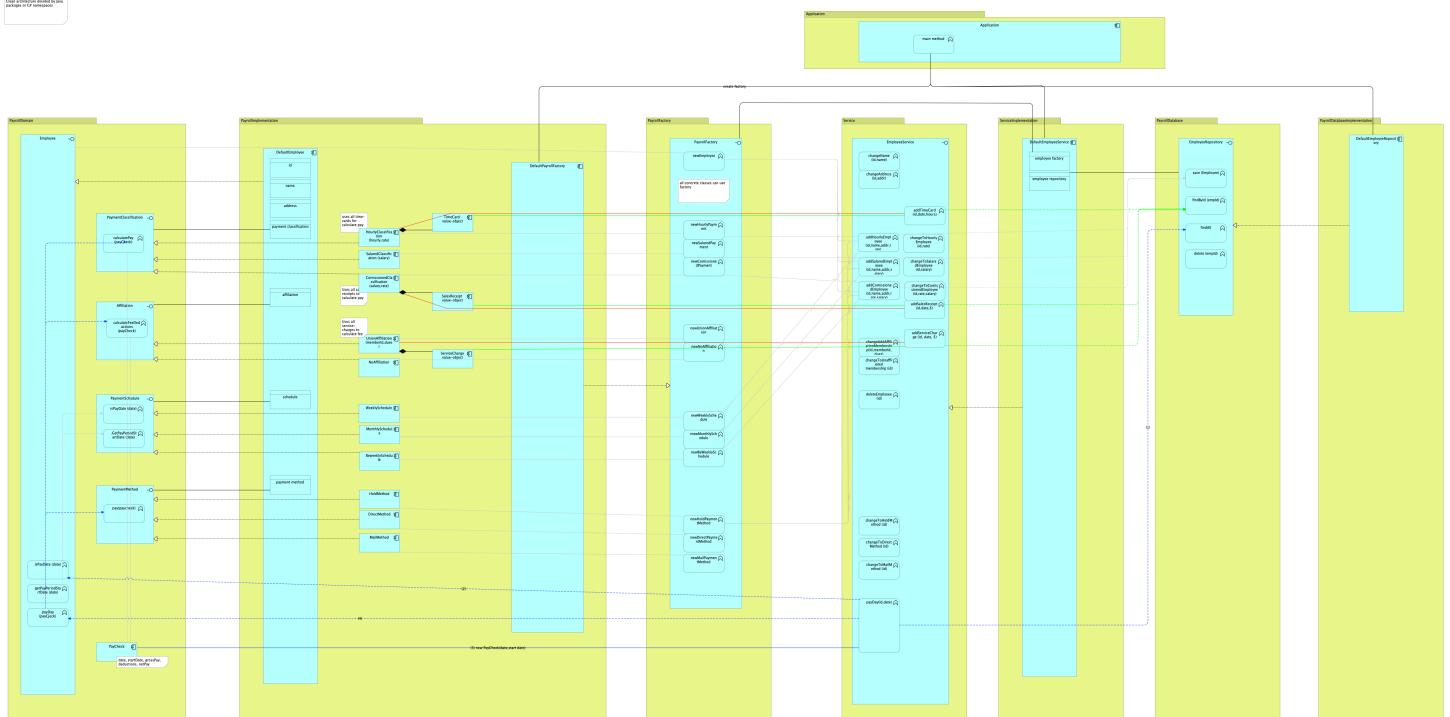


# CLEAN ARCHITECTURE

Payroll system example from Robert Martin book *Agile Software Development, Principles, Patterns, and Practices*.  
 Payroll system type casts is used in the service methods of adding card (value object). These employee type stores a concrete reference to the payment classification interface, and in the service to add a card to it, you need to cast this interface to the `HourlyClassification` type.



Open a structure model to view  
details of UML structure





SOME ARCHITECTURAL DIAGRAMS  
ARE CONSTRUCTED USING  
JARCHITECT, COURTESY OF  
CODEGEAR / 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/BIAN](https://github.com/wilmerkrisp/BIAN))



KrispWilmer