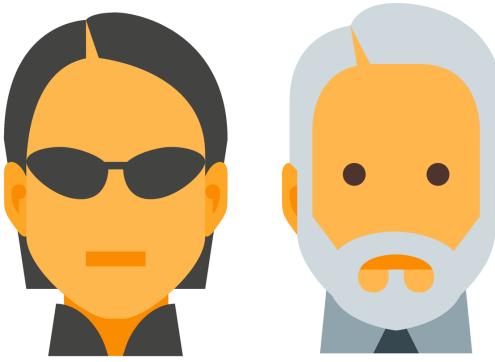


# 155 PATTERNS



## 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

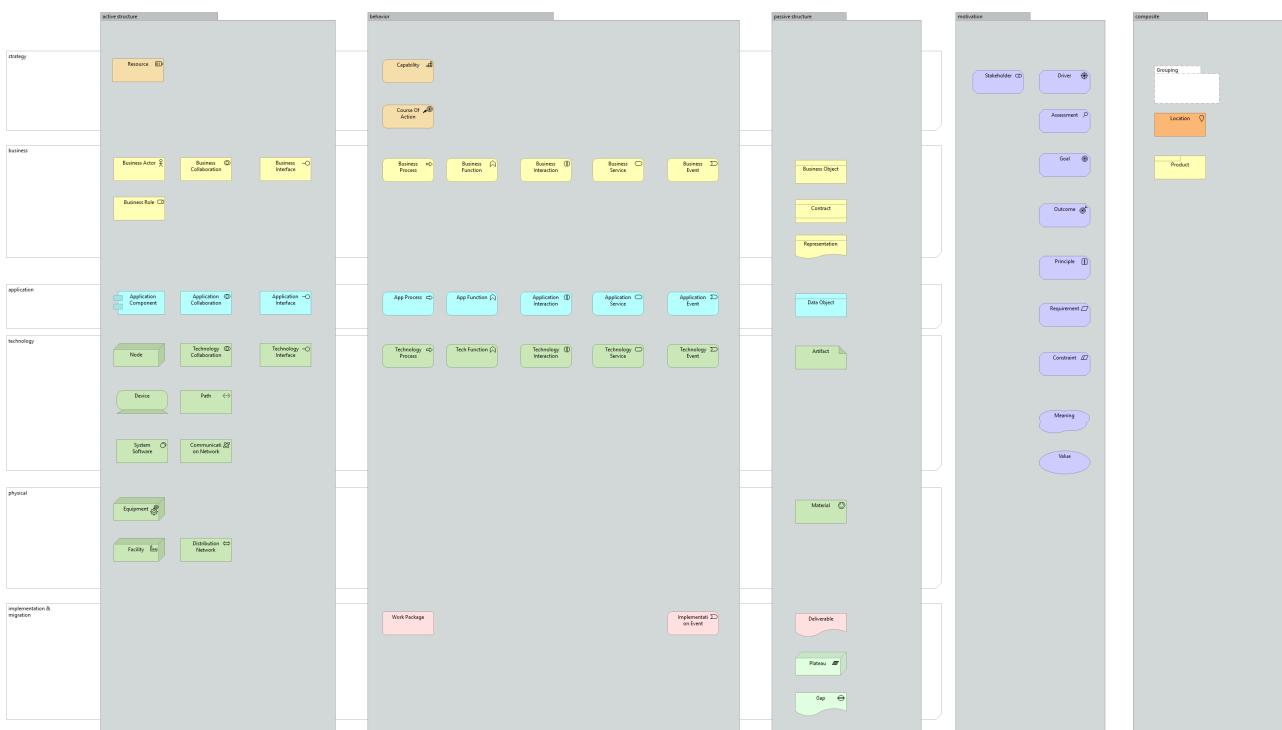
Tackling Complexity in the Heart of Software.

ERIC EVANS

# USED NOTATION

ARCHIMATE METAMODEL

The Open Group



<b>DESIGN PATTERNS</b>	11
CREATIONAL PATTERNS	12
ABSTRACT FACTORY	13
BUILDER	15
FACTORY METHOD	17
PROTOTYPE	19
SINGLETON	21
STRUCTURAL PATTERNS	23
ADAPTER OF CLASS	24
ADAPTER OF OBJECT	26
BRIDGE	29
COMPOSITE	31
DECORATOR	33
FAÇADE	35
FLYWEIGHT	37
FLYWEIGHT + COMPOSITE	39
PROXY	40
BEHAVIORAL PATTERNS	42
CHAIN OF RESPONSIBILITY	43
COMMAND	45
INTERPRETER	47
ITERATOR	49
MEDIATOR	51
MEMENTO	53
OBSERVER	55
STATE	57
STRATEGY	59
TEMPLATE METHOD	61
VISITOR	63
<b>ENTERPRISE PATTERNS</b>	65
BUSINESS LOGIC	66
DOMAIN MODEL	67
SERVICE LAYER.	68
TRANSACTION SCRIPT	69
TABLE MODULE	70
DATA SOURCES	71
ACTIVE RECORD	72
DATA MAPPER	73
ROW DATA GATEWAY	74
TABLE DATA GATEWAY	75
MODELING BEHAVIOR	76

IDENTITY MAP	77
LAZY LOAD	78
UNIT OF WORK.	79
MODELING STRUCTURE HIERARCHY	80
CLASS TABLE INHERITANCE	81
CONCRETE TABLE INHERITANCE	82
INHERITANCE MAPPERS	83
SINGLE TABLE INHERITANCE	84
MODELING STRUCTURE RELATIONS	85
ASSOCIATION TABLE MAPPING	86
DEPENDENT MAPPING	87
EMBEDDED VALUE	88
FOREIGN KEY MAPPING	89
IDENTITY FIELD	90
SERIALIZED LOB	91
METADATA	92
METADATA MAPPING	93
QUERY OBJECT	94
REPOSITORY	95
WEB REPRESENTATION CONTROLLER	96
MODEL VIEW CONTROLLER	97
APPLICATION CONTROLLER	98
FRONT CONTROLLER	99
PAGE CONTROLLER	100
WEB REPRESENTATION VIEW	101
TEMPLATE VIEW	102
TRANSFORM VIEW	103
TWO STEP VIEW	104
DISTRIBUTED PROCESSING	105
DATA TRANSFER OBJECT	106
REMOTE FAÇADE	107
PARALLEL PROCESSING	108
COARSE-GRAINED LOCK	109
IMPLICIT LOCK	110
OPTIMISTIC OFFLINE LOCK	111
PESSIMISTIC OFFLINE LOCK	112
SESSION STATE	113
CLIENT SESSION STATE	114
DATABASE SESSION STATE	115
SERVER SESSION STATE	116
COMMON PATTERNS	117
GATEWAY	118
LAYER SUPERTYPE	119

MAPPER	120
MONEY	121
PLUGIN	122
RECORD SET	123
REGISTRY	124
SEPARATED INTERFACE	125
SERVICE STUB	126
SPECIAL CASE	127
VALUE OBJECT	128
<b>ANALYSIS PATTERNS</b>	<b>129</b>
ACCOUNTABILITY	131
PARTY	132
ACCOUNTABILITY	133
ORGANIZATION HIERARCHIES	135
ORGANIZATION STRUCTURE	136
ACCOUNTABILITY KNOWLEDGE LEVEL	137
PARTY TYPE GENERALIZATIONS	138
HIERARCHIC ACCOUNTABILITY	139
OPERATING SCOPES	140
POST	141
OBSERVATIONS AND MEASUREMENTS	142
QUANTITY	143
CONVERSION RATIO	144
OBSERVATIONS AND MEASUREMENTS	145
COMPOUND UNITS	146
MEASUREMENT	147
OBSERVATION	148
SUBTYPING OBSERVATION CONCEPTS	149
PROTOCOL	150
DUAL TIME RECORD	151
REJECTED OBSERVATION	152
ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION	153
ASSOCIATED OBSERVATION	154
PROCESS OF OBSERVATION	155
OBSERVATIONS FOR CORPORATE FINANCE	156
ENTERPRISE SEGMENT	157
MEASUREMENT PROTOCOL	158
RANGE	159
OBSERVATIONS FOR CORPORATE FINANCE	160
PHENOMENON WITH RANGE	163
REFERRING TO OBJECTS	164
NAME	165
IDENTIFICATION SCHEME	166

OBJECT MERGE	167
OBJECT EQUIVALENCE	168
REFERRING TO OBJECTS	169
INVENTORY AND ACCOUNTING	170
ACCOUNT	171
TRANSACTIONS	172
SUMMARY ACCOUNT	173
MEMO ACCOUNT	174
POSTING RULES	175
INVENTORY AND ACCOUNTING	176
INDIVIDUAL INSTANCE METHOD	177
POSTING RULE EXECUTION	178
POSTING RULES FOR MANY ACCOUNTS	179
CHOOSING ENTRIES	180
ACCOUNTING PRACTICE	181
SOURCES OF AN ENTRY	182
BALANCE SHEET AND INCOME STATEMENT	183
CORRESPONDING ACCOUNT	184
SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)	185
SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)	186
BOOKING ENTRIES TO MULTIPLE ACCOUNTS	187
PLANNING	188
PROPOSED AND IMPLEMENTED ACTION	189
COMPLETED AND ABANDONED ACTIONS	190
SUSPENSION	191
PLAN	192
PROTOCOL	193
RESOURCE ALLOCATION	194
PLANNING	195
PLANNING (NO OUTCOME)	196
OUTCOME AND START FUNCTIONS	197
TRADING	198
CONTRACT	199
PORTFOLIO	200
QUOTE	201
SCENARIO	202
TRADING	203
DERIVATIVE CONTRACTS	204
FORWARD CONTRACTS	205
OPTIONS	206
PRODUCT	207
SUBTYPE STATE MACHINES	208
PARALLEL APPLICATION AND DOMAIN HIERARCHIES	209

DERIVATIVE CONTRACTS	210
TRADING PACKAGES	211
MULTIPLE ACCESS LEVELS TO A PACKAGE	212
MUTUAL VISIBILITY	213
TRADING PACKAGES	214
LAYERED ARCHITECTURE FOR INFORMATION SYSTEMS	215
TWO-TIER ARCHITECTURE	216
THREE-TIER ARCHITECTURE	217
PRESENTATION AND APPLICATION LOGIC	218
DATABASE INTERACTION	219
TYPE MODEL DESIGN	220
IMPLEMENTING ASSOCIATIONS	221
IMPLEMENTING GENERALIZATION	222
OBJECT CREATION	223
OBJECT DESTRUCTION	224
ENTRY POINT.	225
IMPLEMENTING CONSTRAINTS	226
<b>DOMAIN DRIVEN DESIGN</b>	227
MODEL AND STRUCTURAL ELEMENTS	228
MODEL-DRIVEN DESIGN	229
LAYERED ARCHITECTURE (ASYMMETRIC )	231
HEXAGONAL ARCHITECTURE (SYMMETRIC)	232
COMPOSITE UI	233
ENTITIES	234
VALUE-OBJECTS	236
DOMAIN SERVICES	238
MODULES	242
AGGREGATES	243
AGGREGATE ROOT	245
BEHAVIOR-FOCUSED AGGREGATE ROOT	246
MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION	247
PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES	248
REFERENCE OTHER AGGREGATES BY IDENTITY ONLY	249
FACTORIES	250
REPOSITORIES	252
SUPPLE DESIGN	255
UBIQUITOUS LANGUAGE	256
INTENTION-REVEALING INTERFACES	257
SIDE-EFFECT FREE FUNCTIONS	258
ASSERTIONS	259
CONCEPTUAL CONTOURS	260
STANDALONE CLASSES	261
CLOSURE OF OPERATIONS	262

MODEL INTEGRITY AND CONTEXT	263
BOUNDED CONTEXT	264
CONTINUOUS INTEGRATION	265
STRATEGIC CONTEXT MAP	266
CONTEXTUAL MAP	267
SHARED KERNEL	268
CUSTOMER-SUPPLIER TEAMS	269
CONFORMIST	270
ANTICORRUPTION LAYER	271
SEPARATE WAYS	272
OPEN HOST SERVICE	273
PUBLISHED LANGUAGE	274
DISTILLATION	275
CORE DOMAIN	276
GENERIC SUBDOMAINS	277
DOMAIN VISION STATEMENT	278
HIGHLIGHTED CORE	279
COHESIVE MECHANISMS	280
SEGREGATED CORE	281
ABSTRACT CORE	282
LARGE-SCALE STRUCTURE	283
EVOLVING ORDER	284
SYSTEM METAPHOR	285
RESPONSIBILITY LAYERS	286
KNOWLEDGE LEVEL	289
PLUGGABLE COMPONENT FRAMEWORK	290
ADDITIONAL PATTERNS	291
TYPES OF CONSISTENCY	292
EVENT SOURCING	293
EVENT PROCESSOR	294
EVENT DISPATCHER	295
INTERNAL DOMAIN EVENTS	296
EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS	297
STATIC DOMAIN EVENTS CLASS	298
ONE SUBDOMAIN PER BOUNDED CONTEXT	299
THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS	300
THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS	301
INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE	302
INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES	303
INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS	304
INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE	305
DEPENDENCY INJECTION	306
DEPENDENCY INVERSION	307

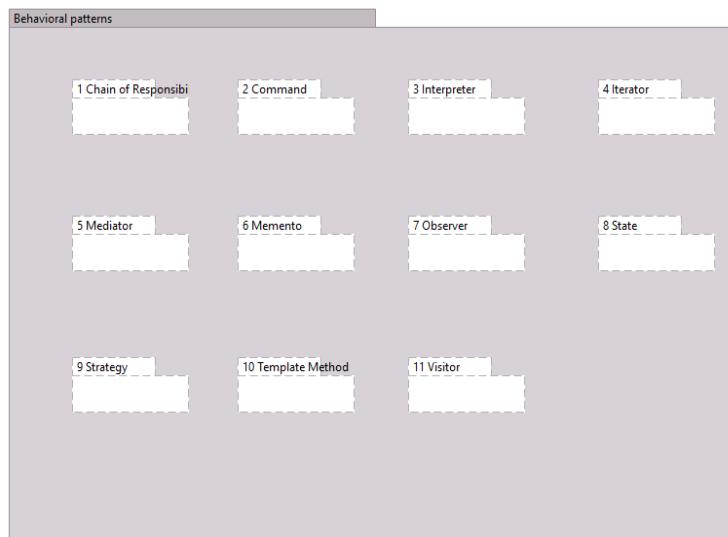
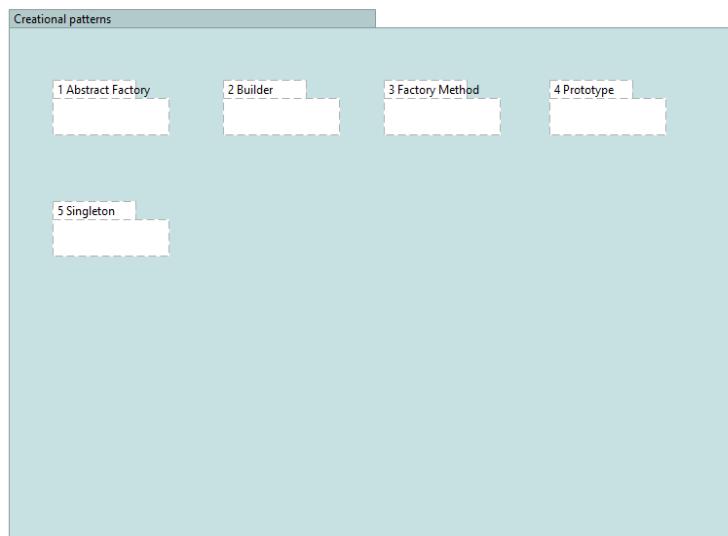
INVERSION OF CONTROL	308
SERVICE LOCATOR	309

01

# Design Patterns

Elements of Reusable Object-Oriented Software

GANG OF FOUR



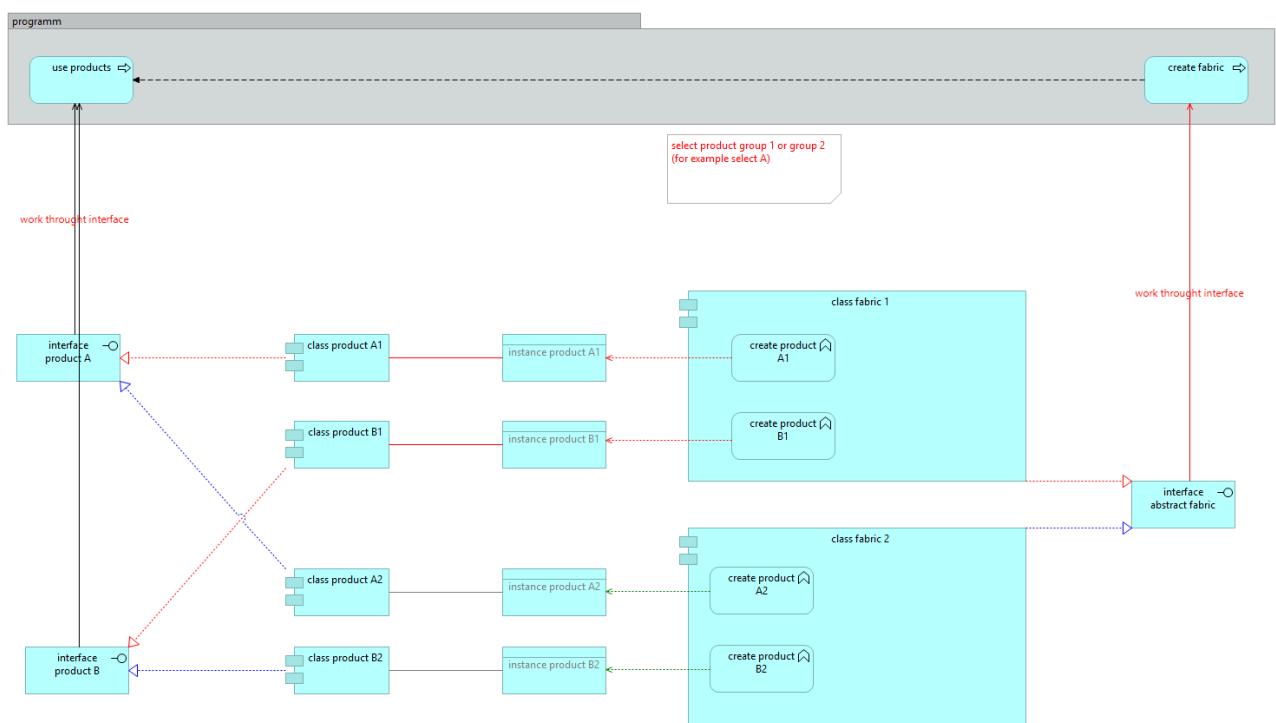
# CREATIONAL PATTERNS

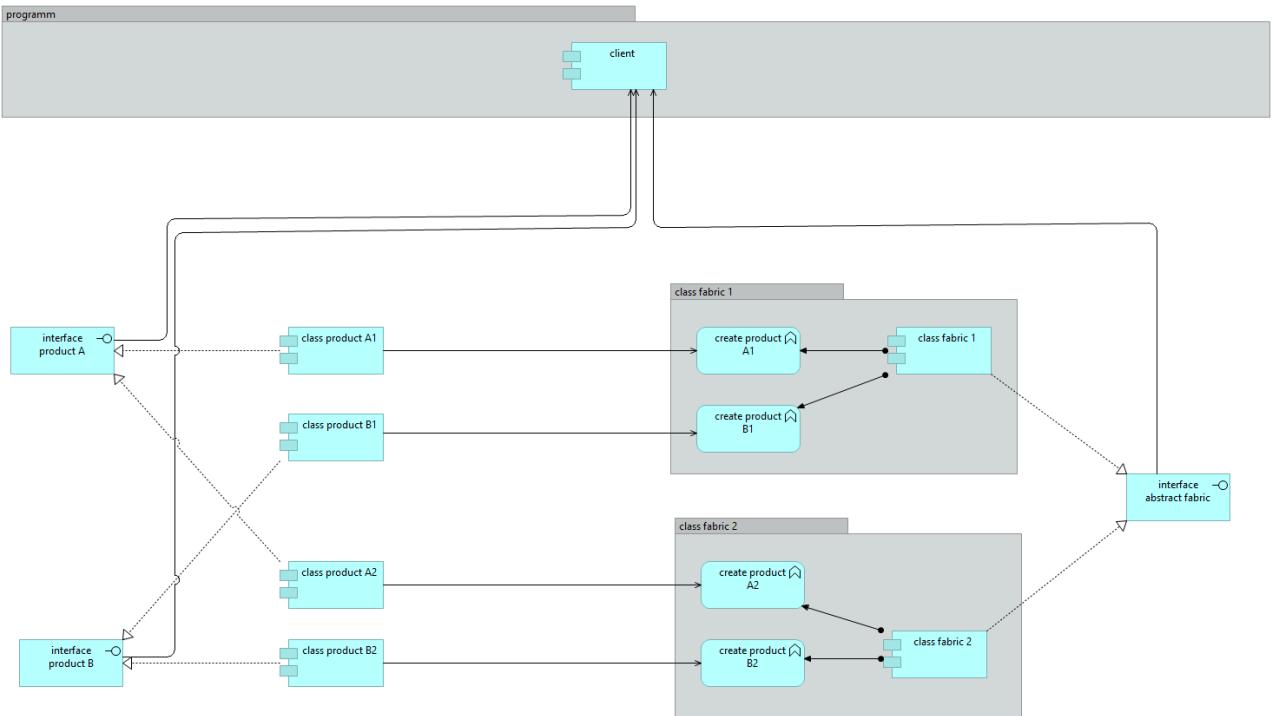
DESIGN PATTERNS

GoF

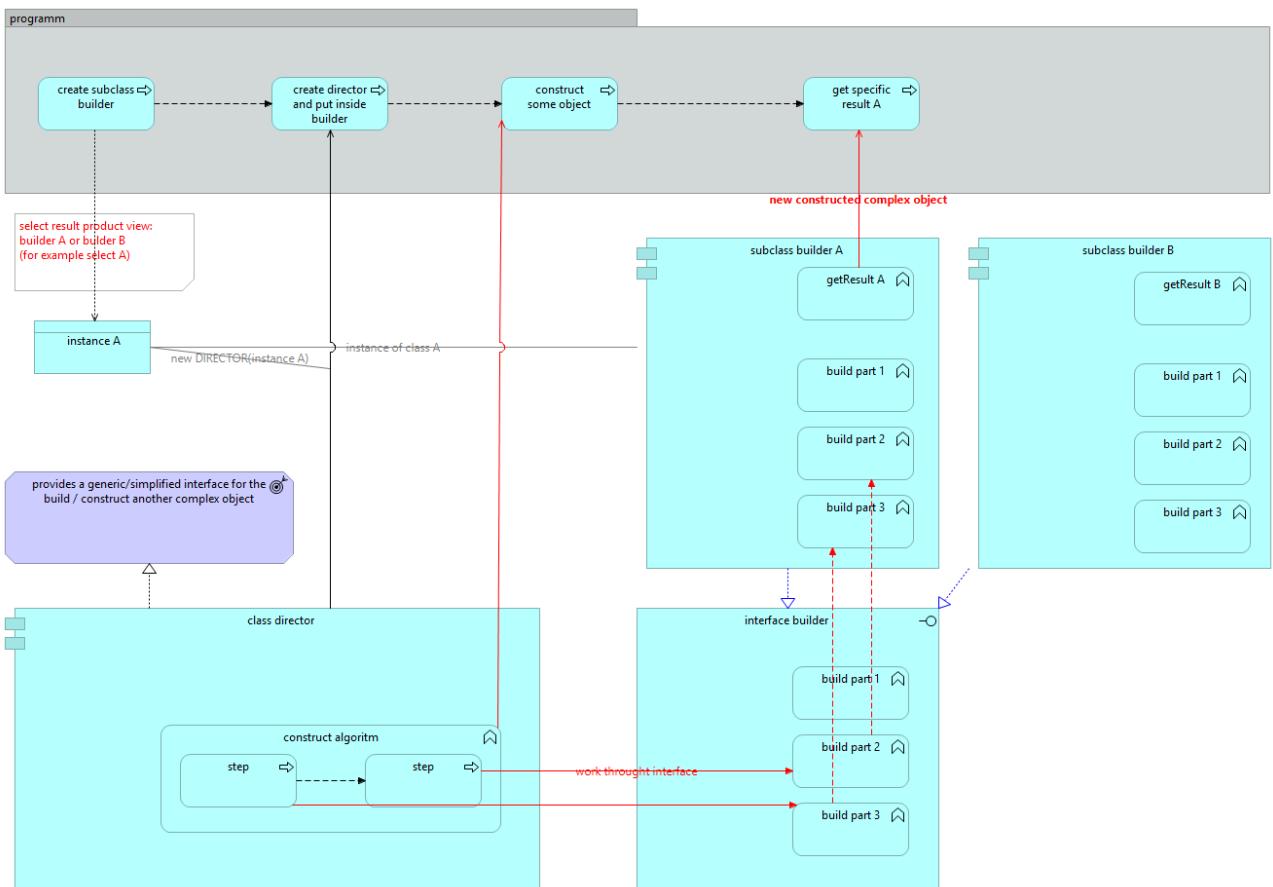


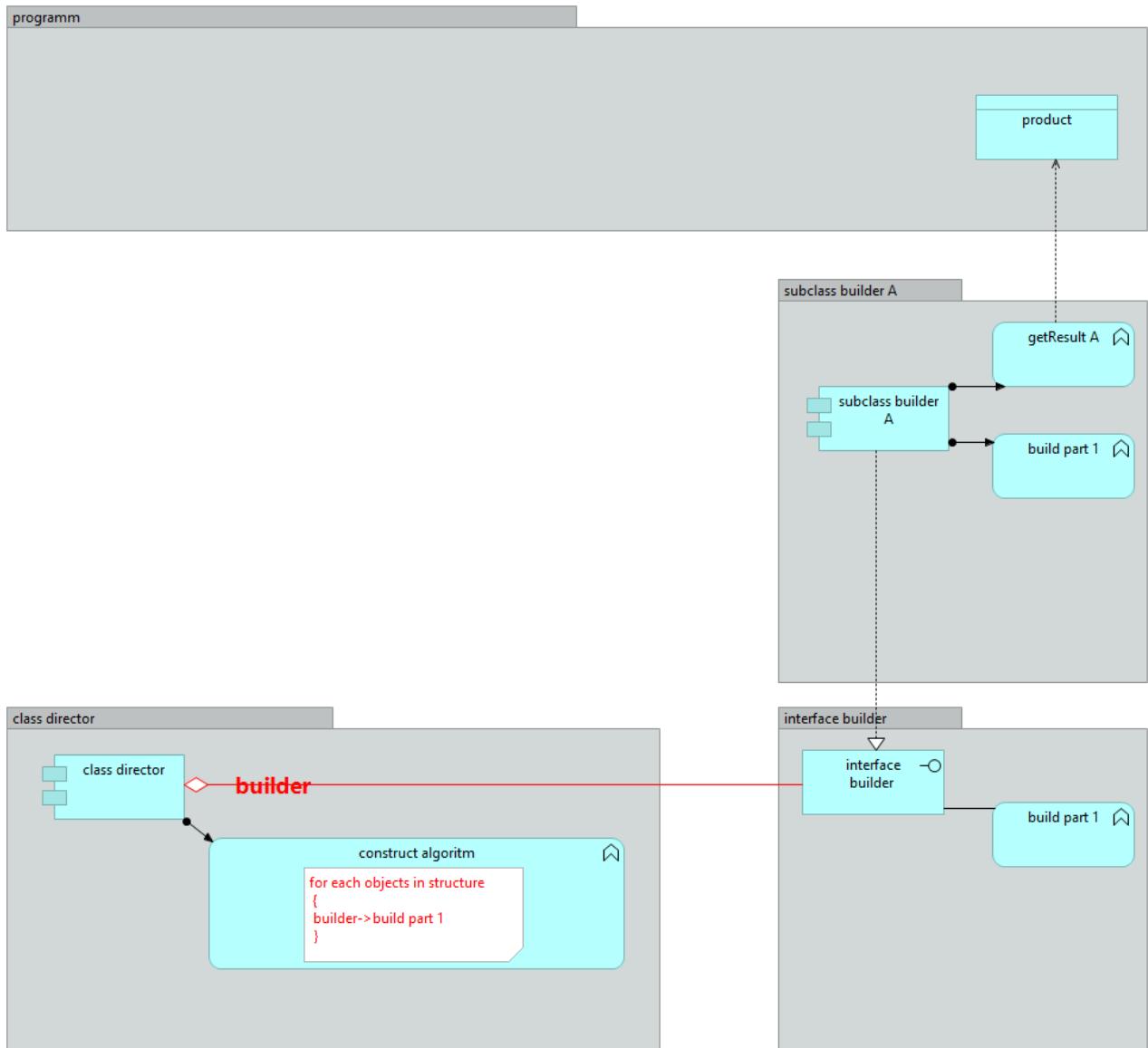
# ABSTRACT FACTORY



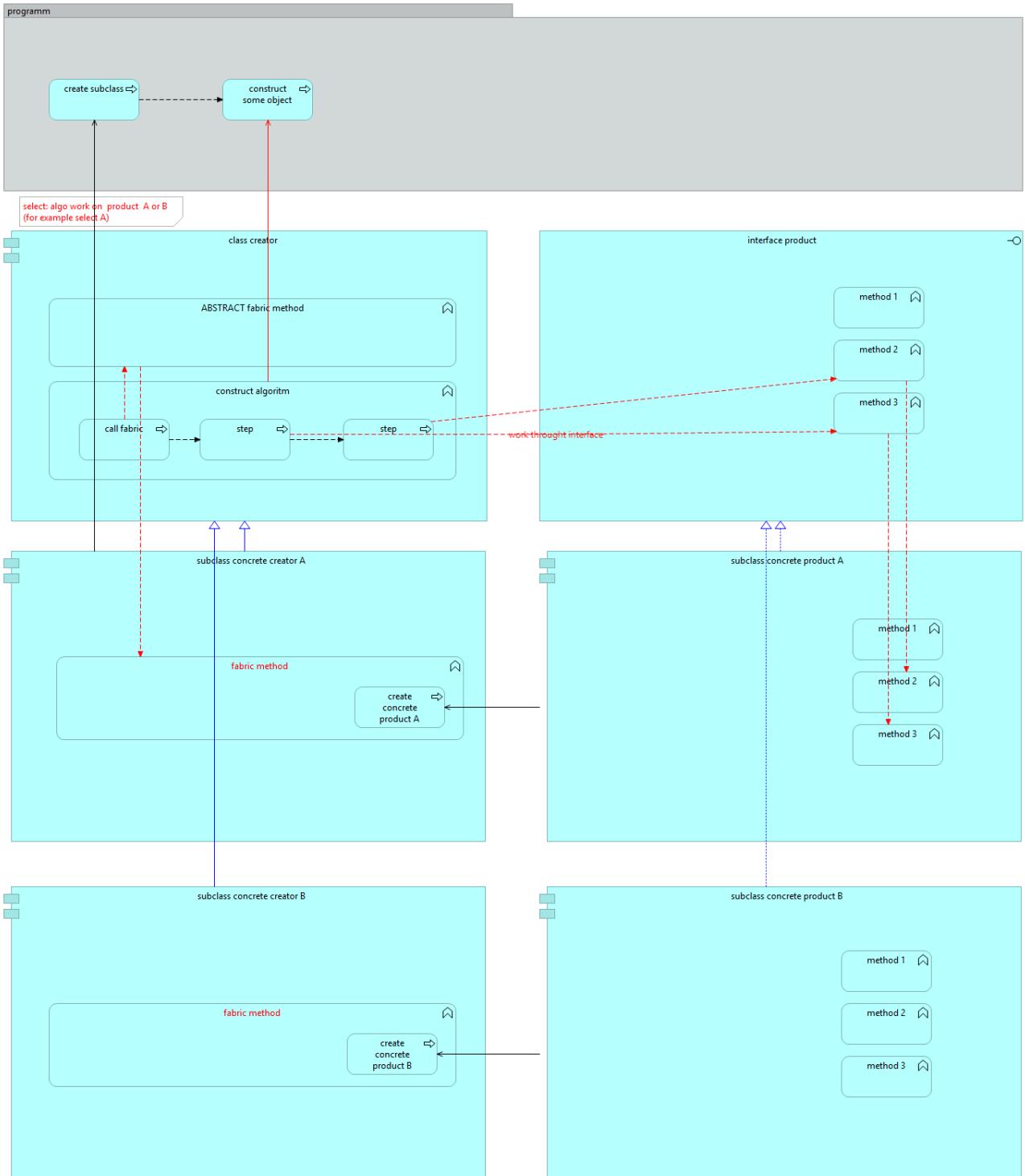


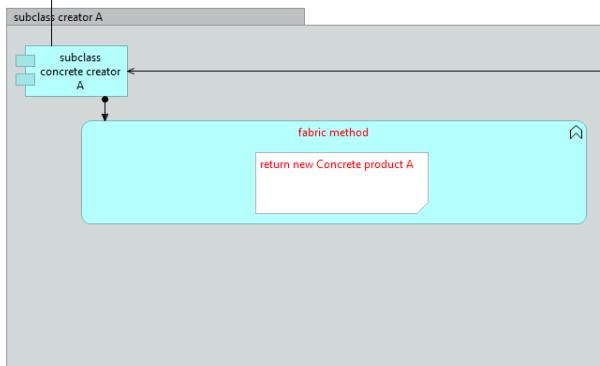
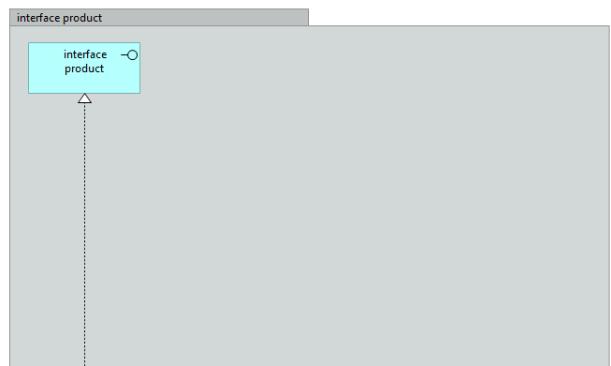
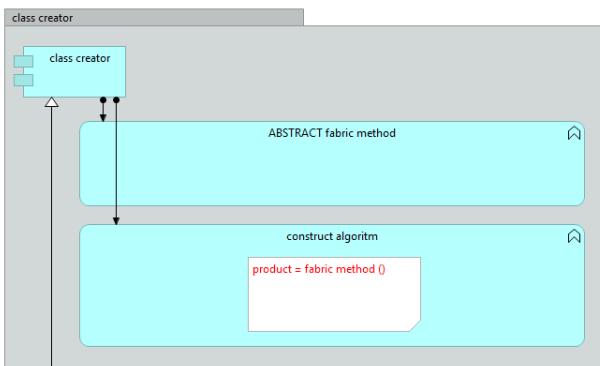
# BUILDER



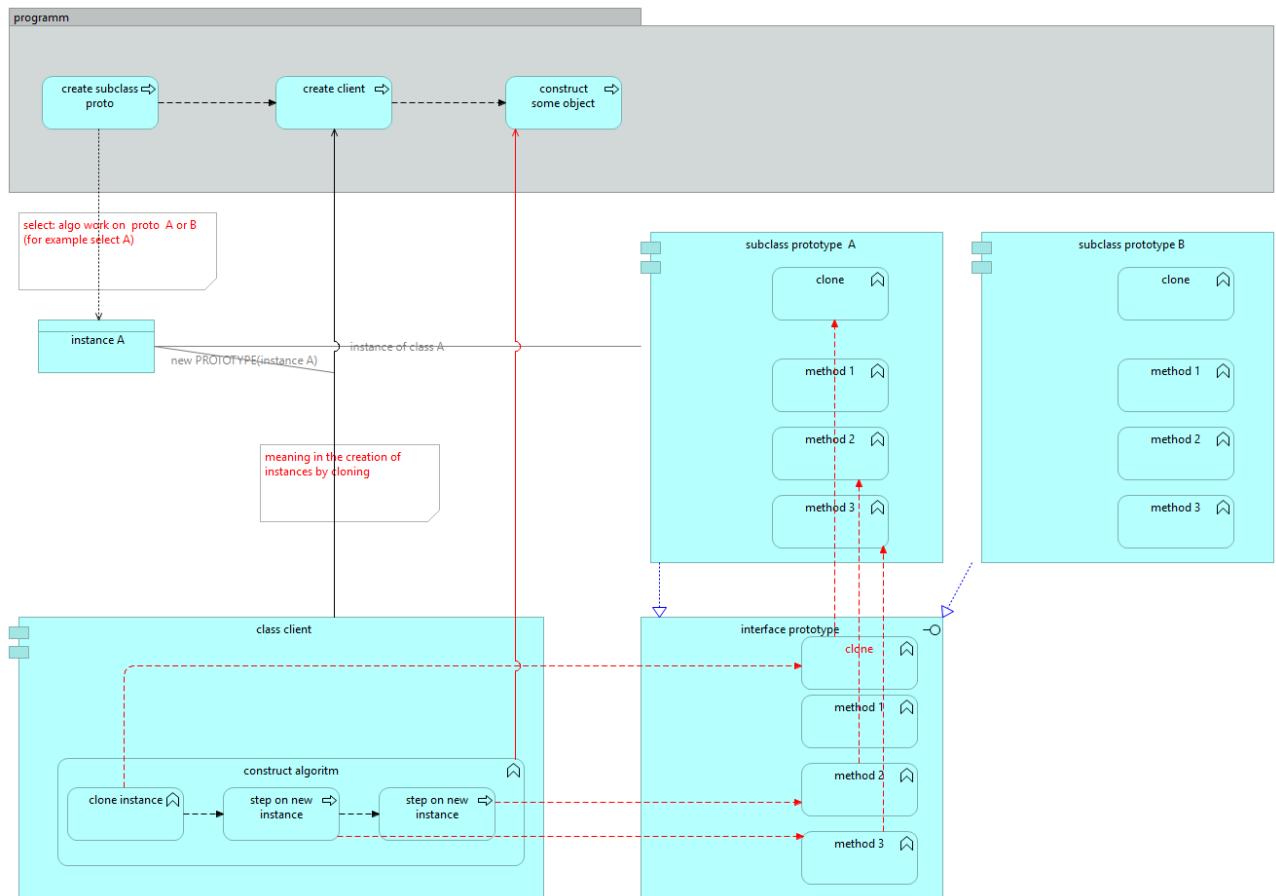


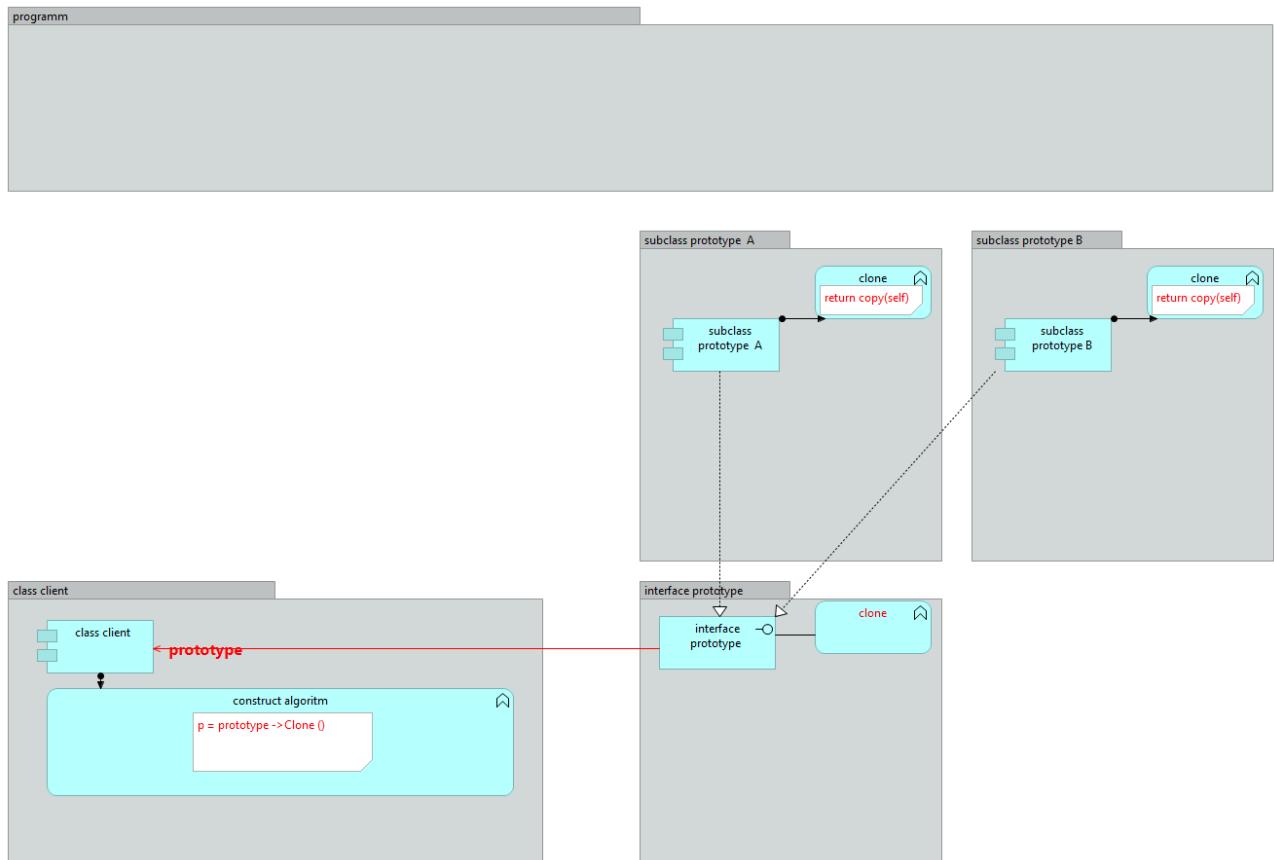
# FACTORY METHOD



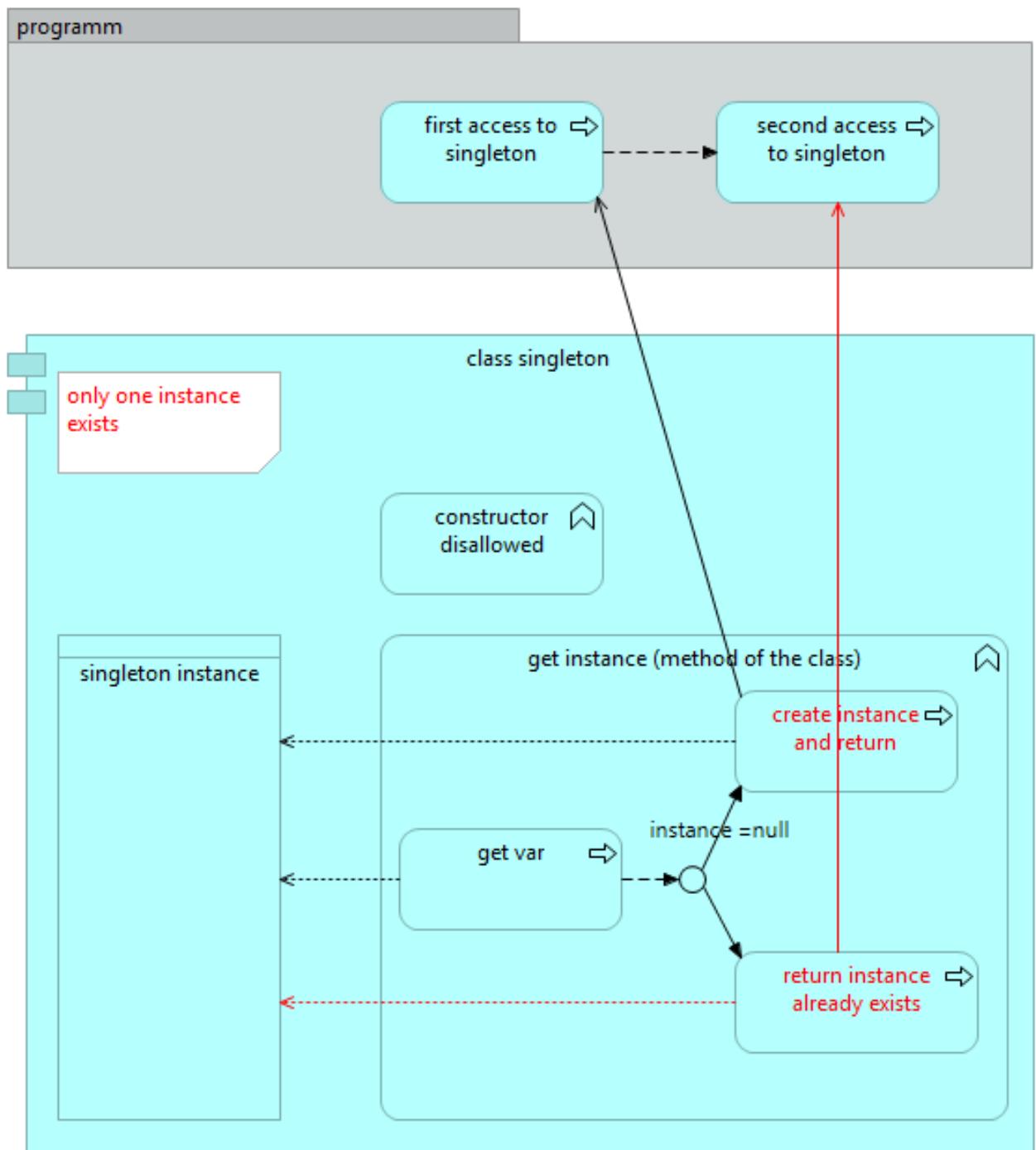


# PROTOTYPE





# SINGLETON



programm

class singleton

  class singleton

  singleton instance

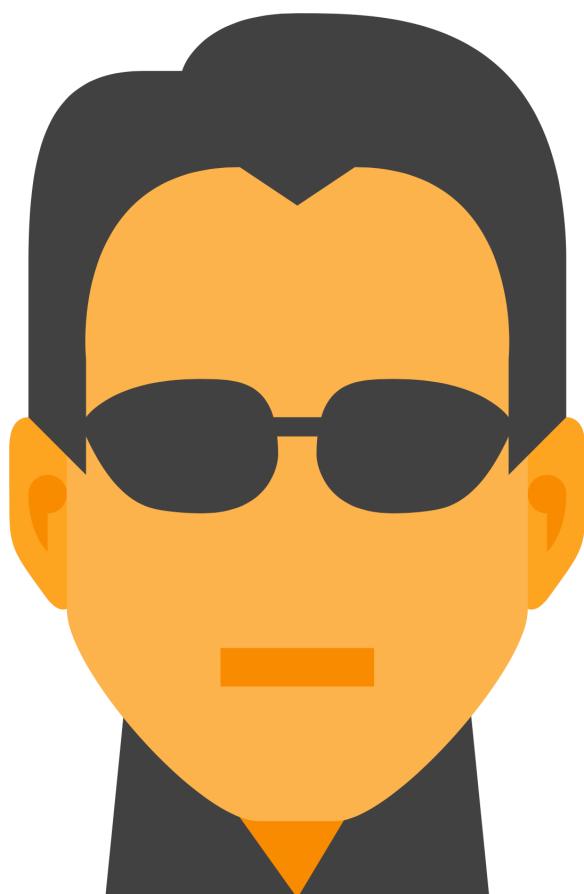
  get instance (method of the class)

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

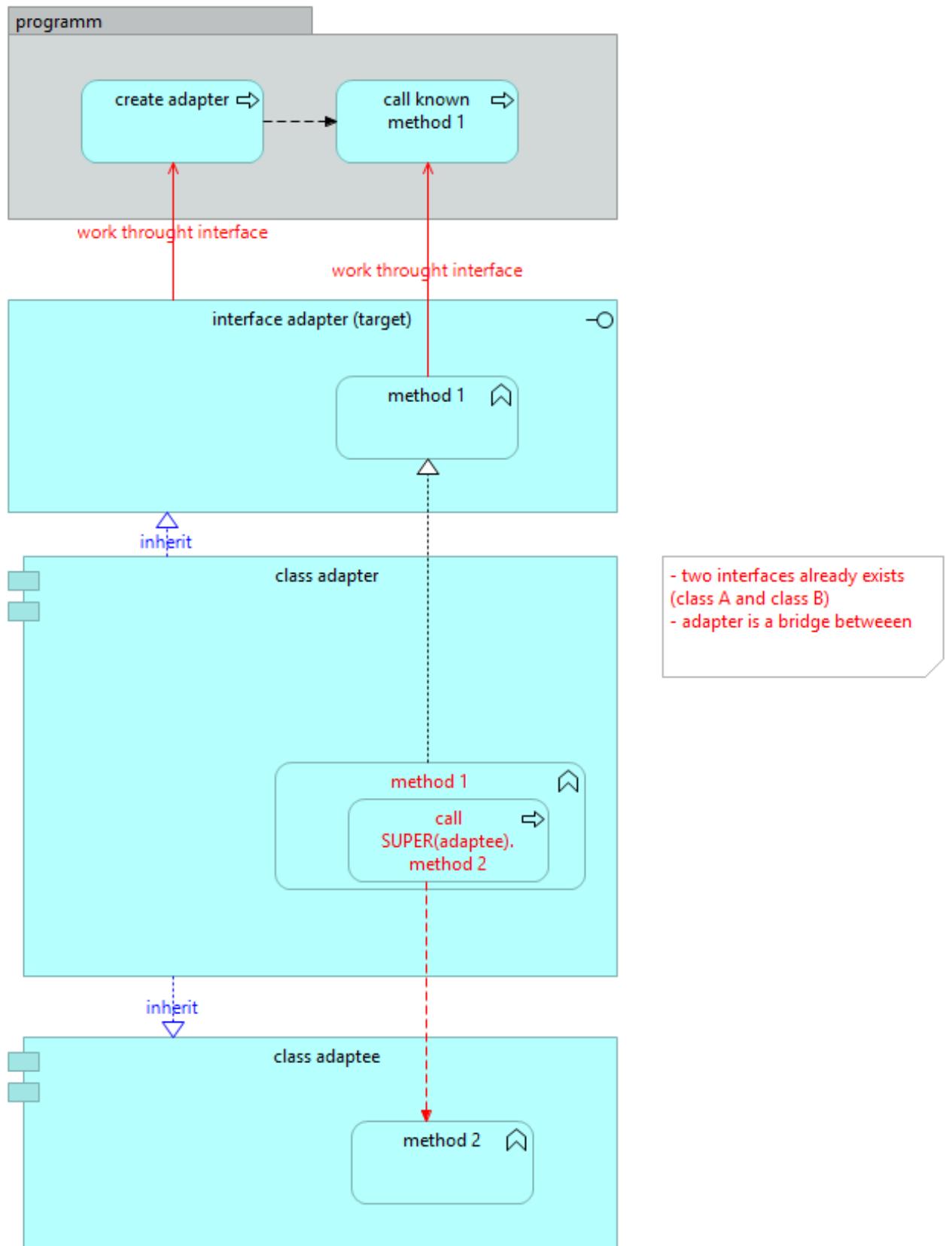
# STRUCTURAL PATTERNS

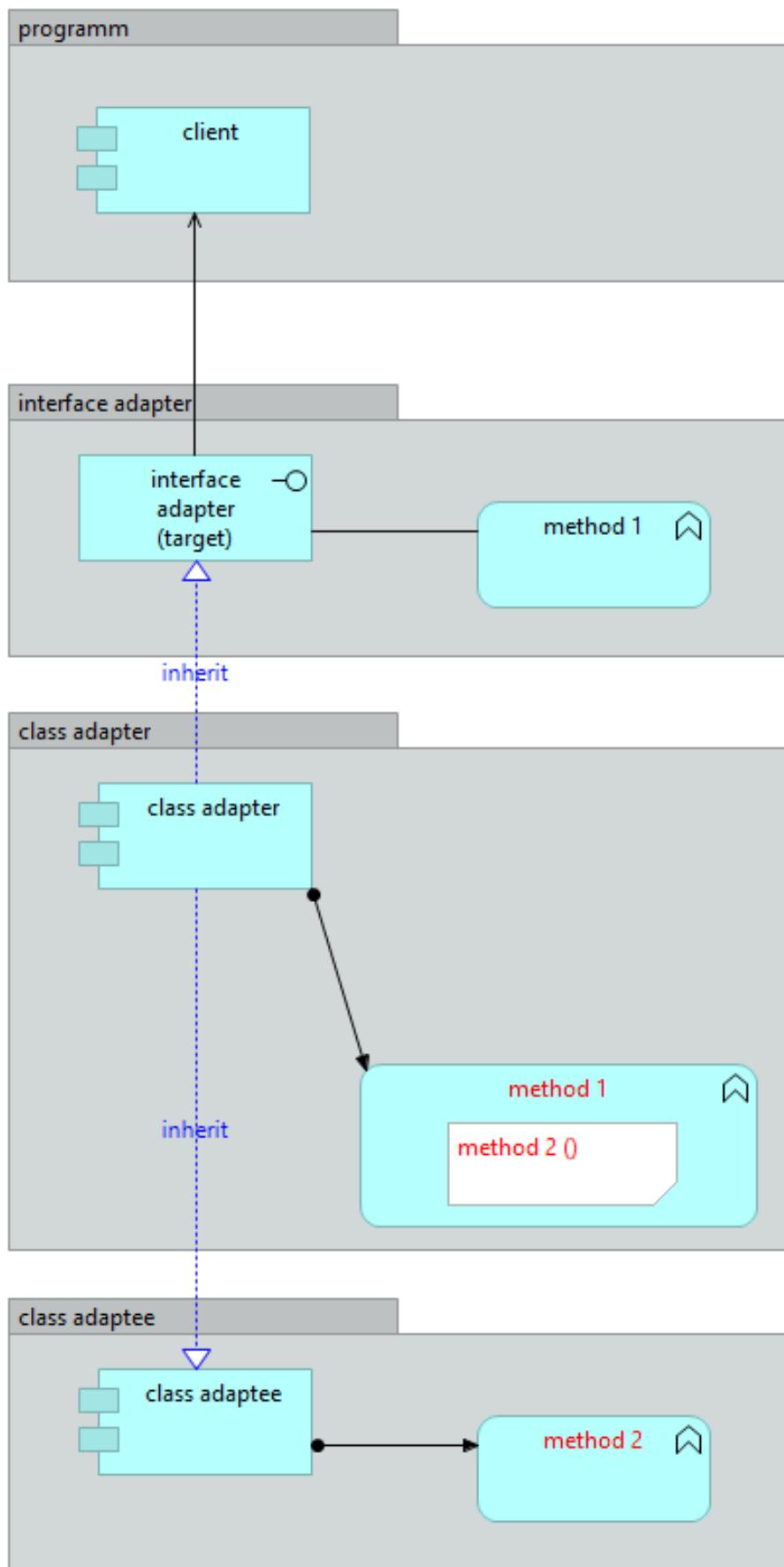
DESIGN PATTERNS

GoF

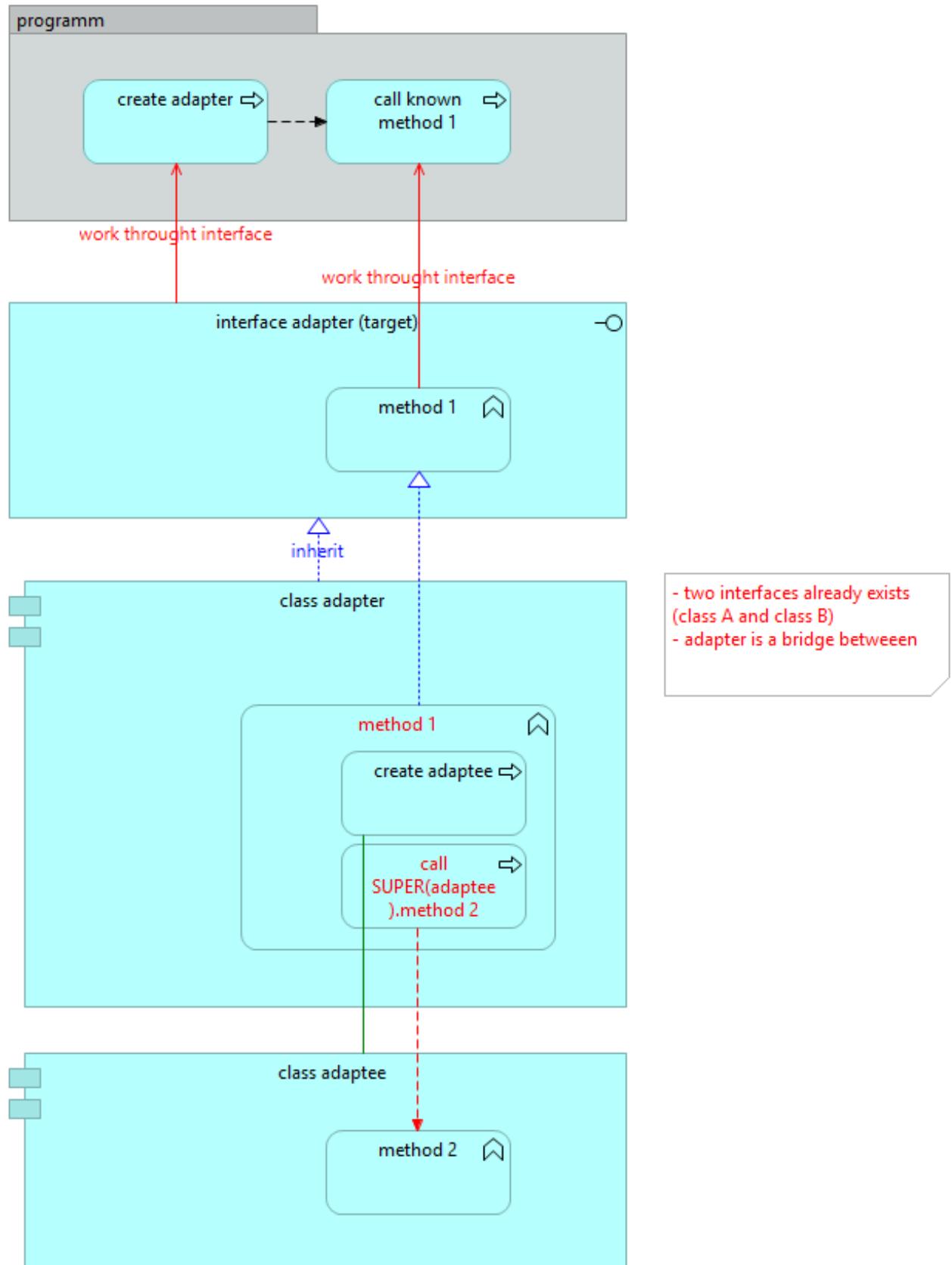


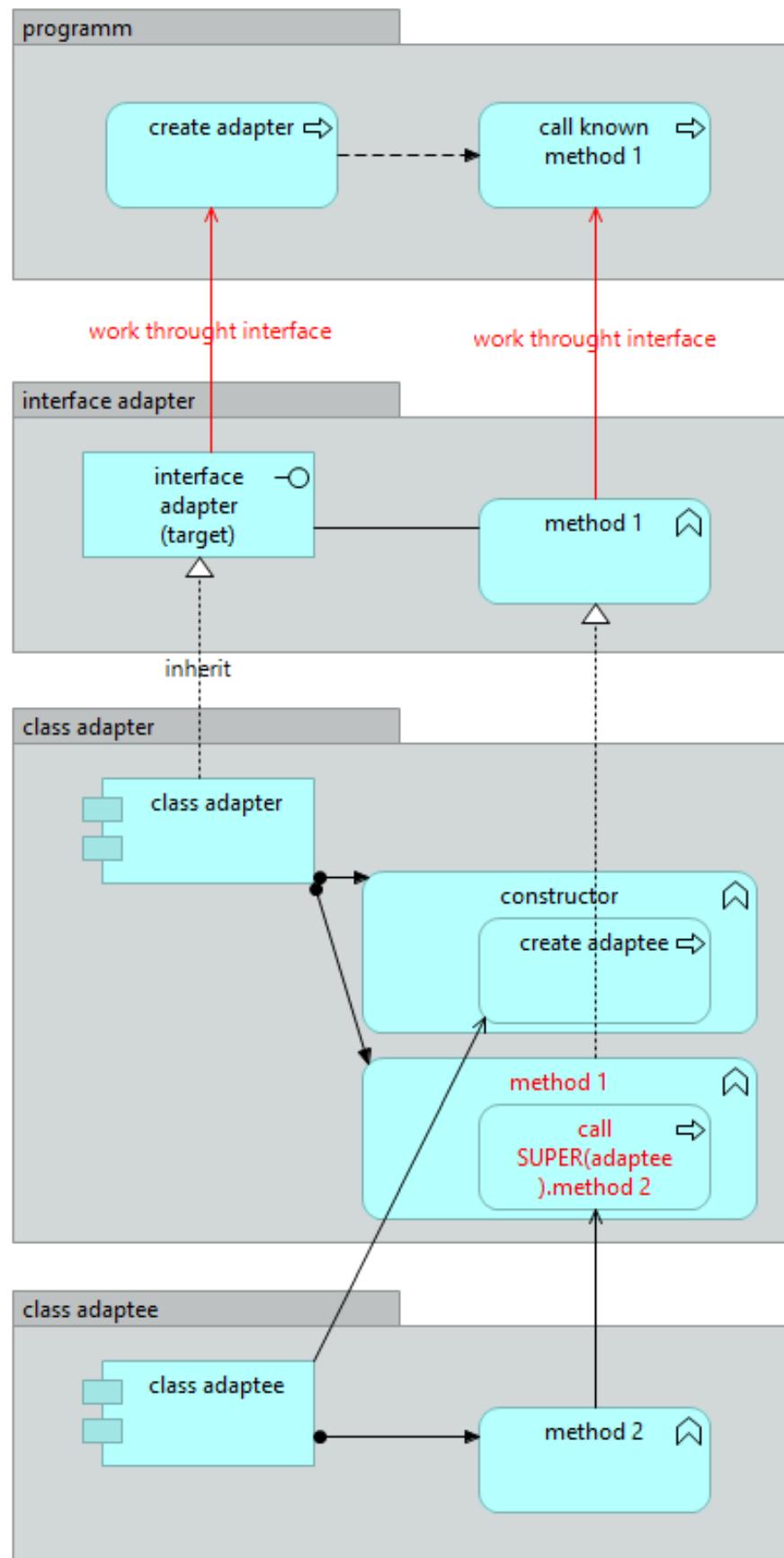
# ADAPTER OF CLASS

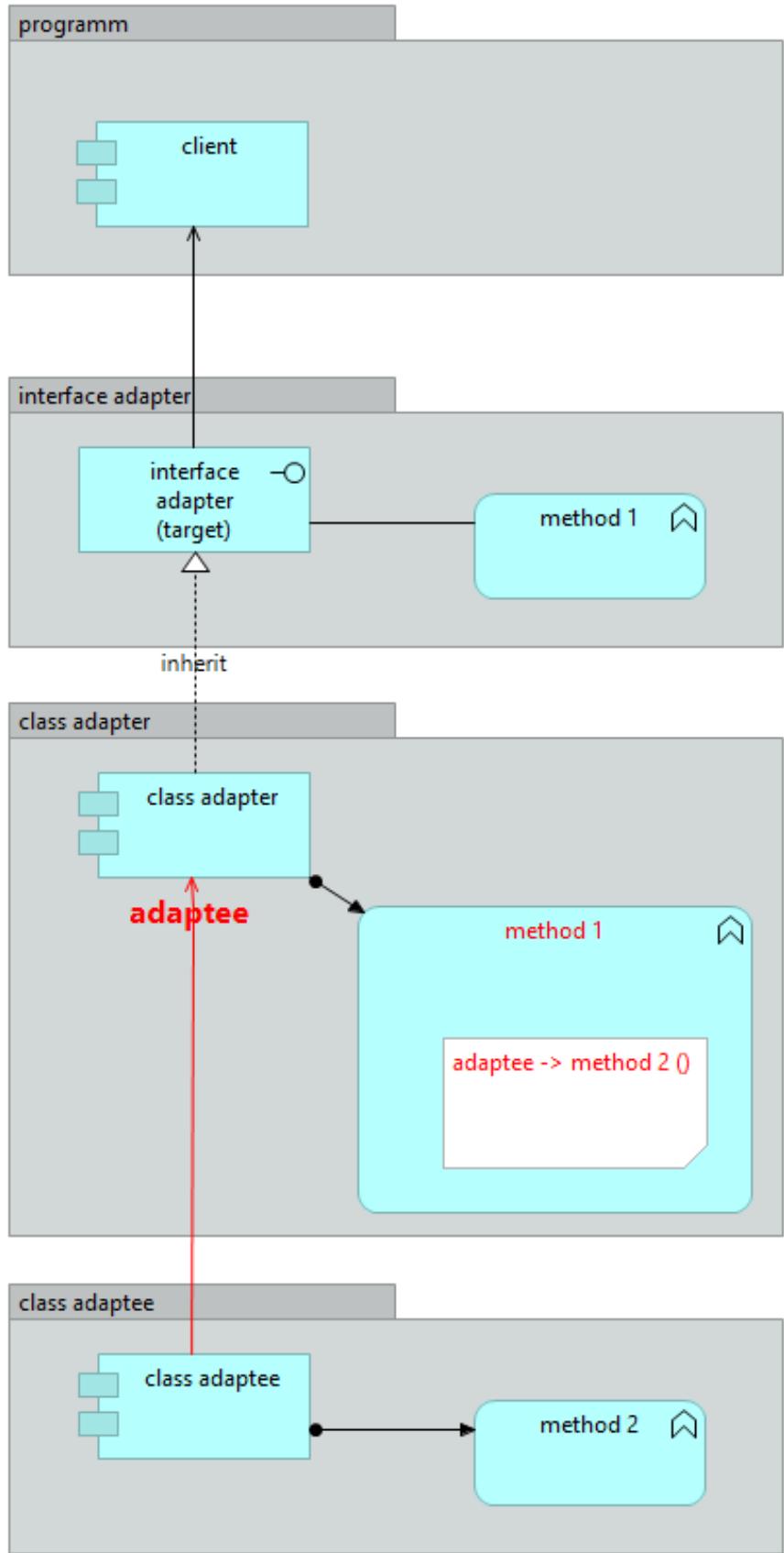




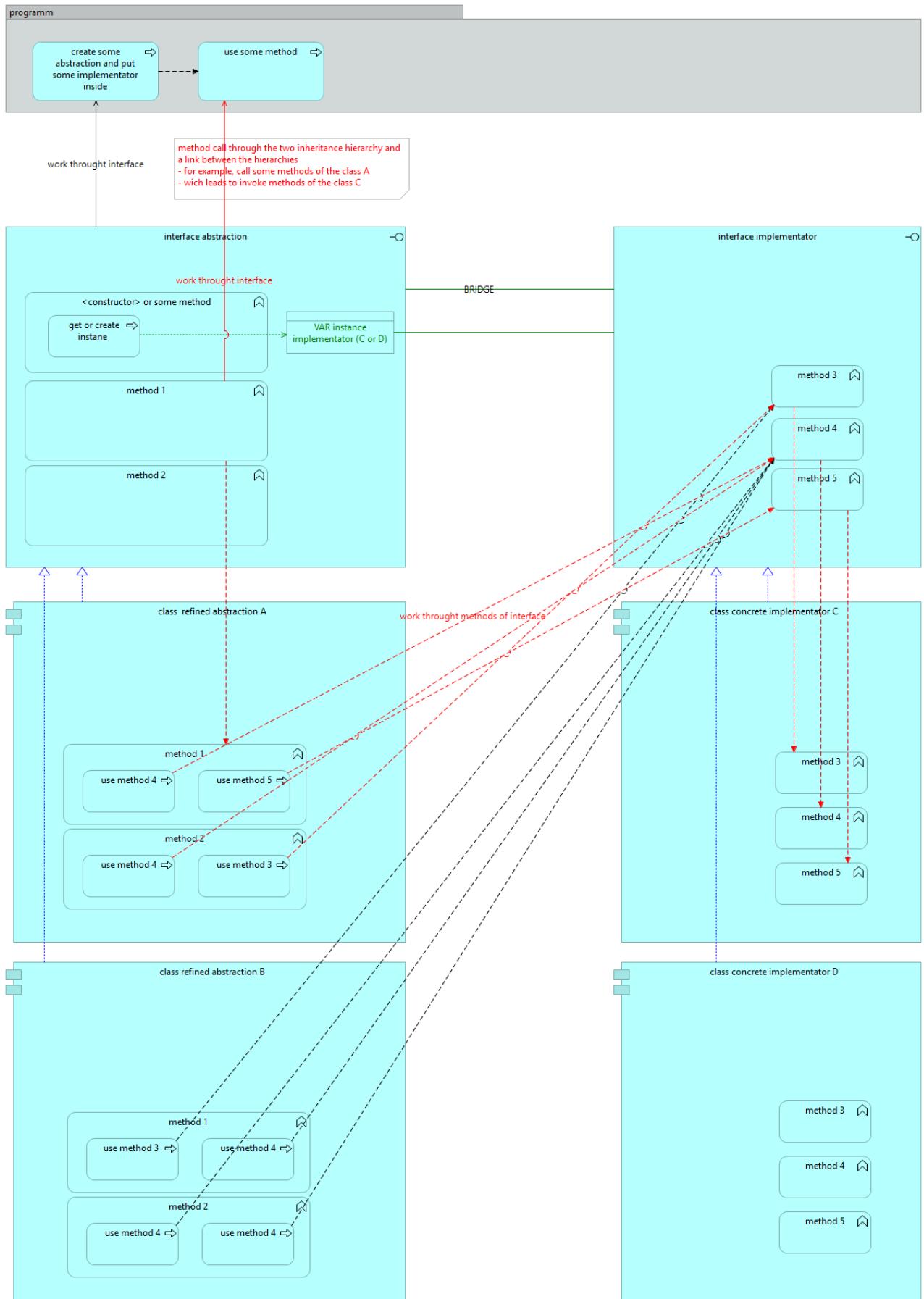
# ADAPTER OF OBJECT

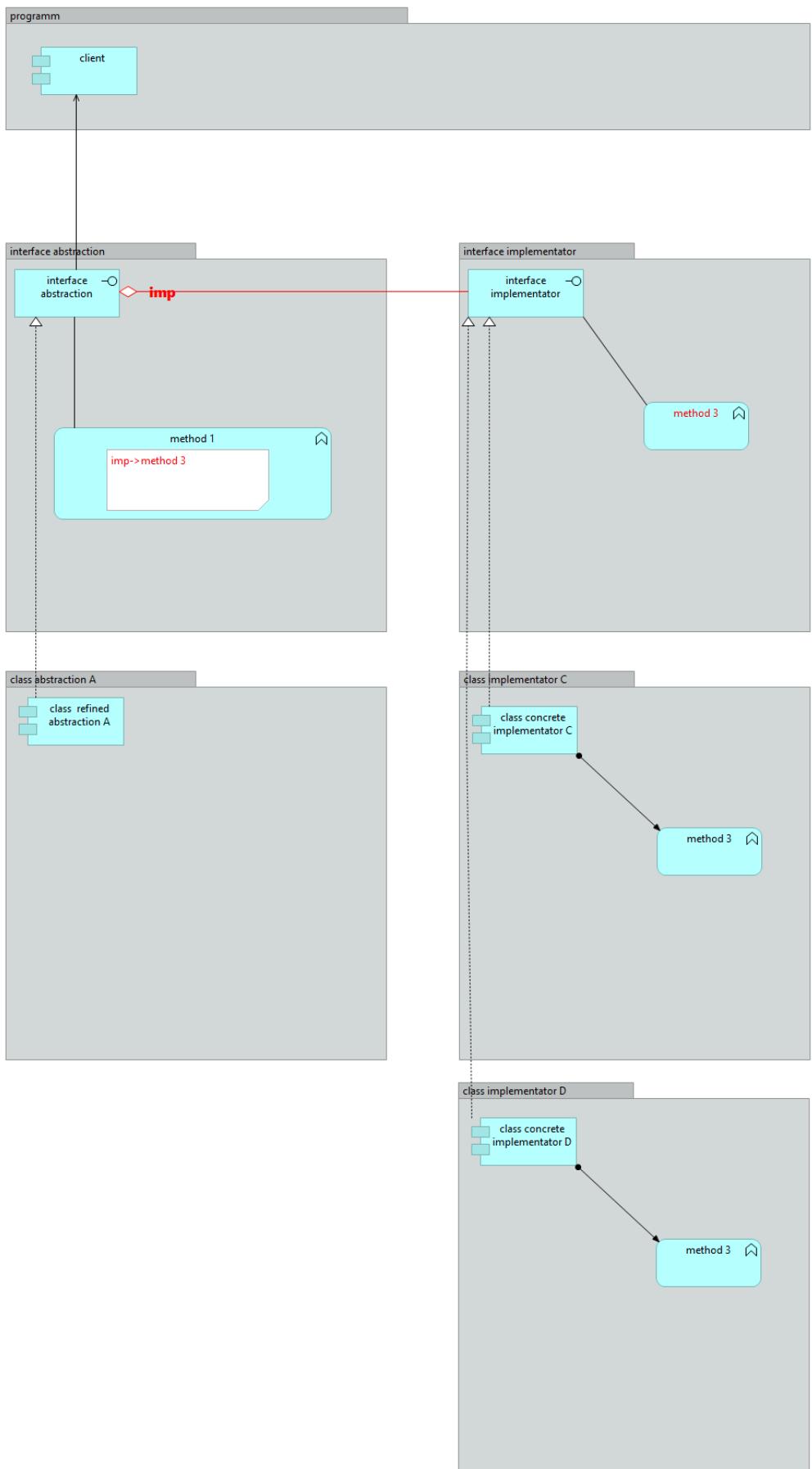




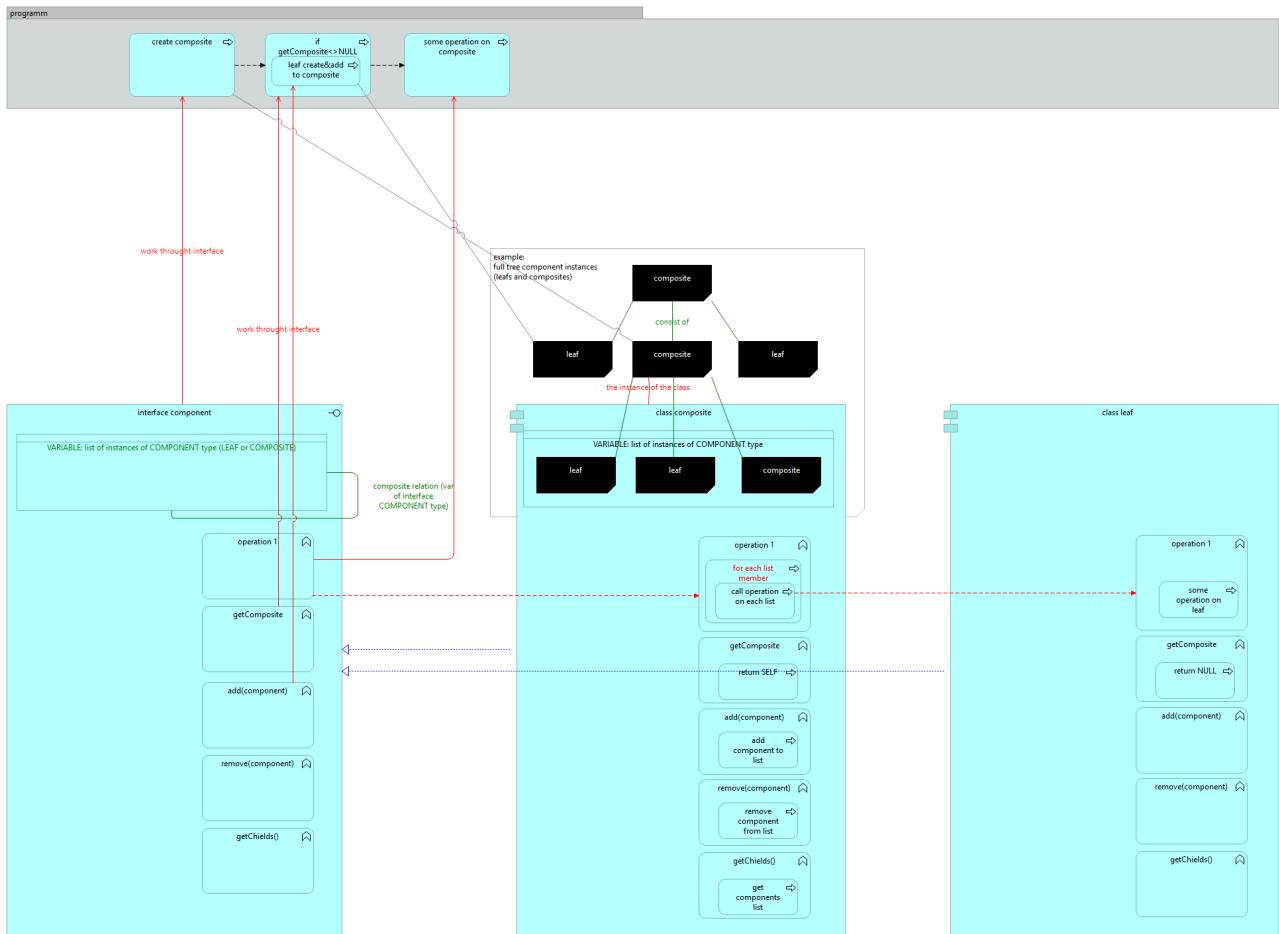


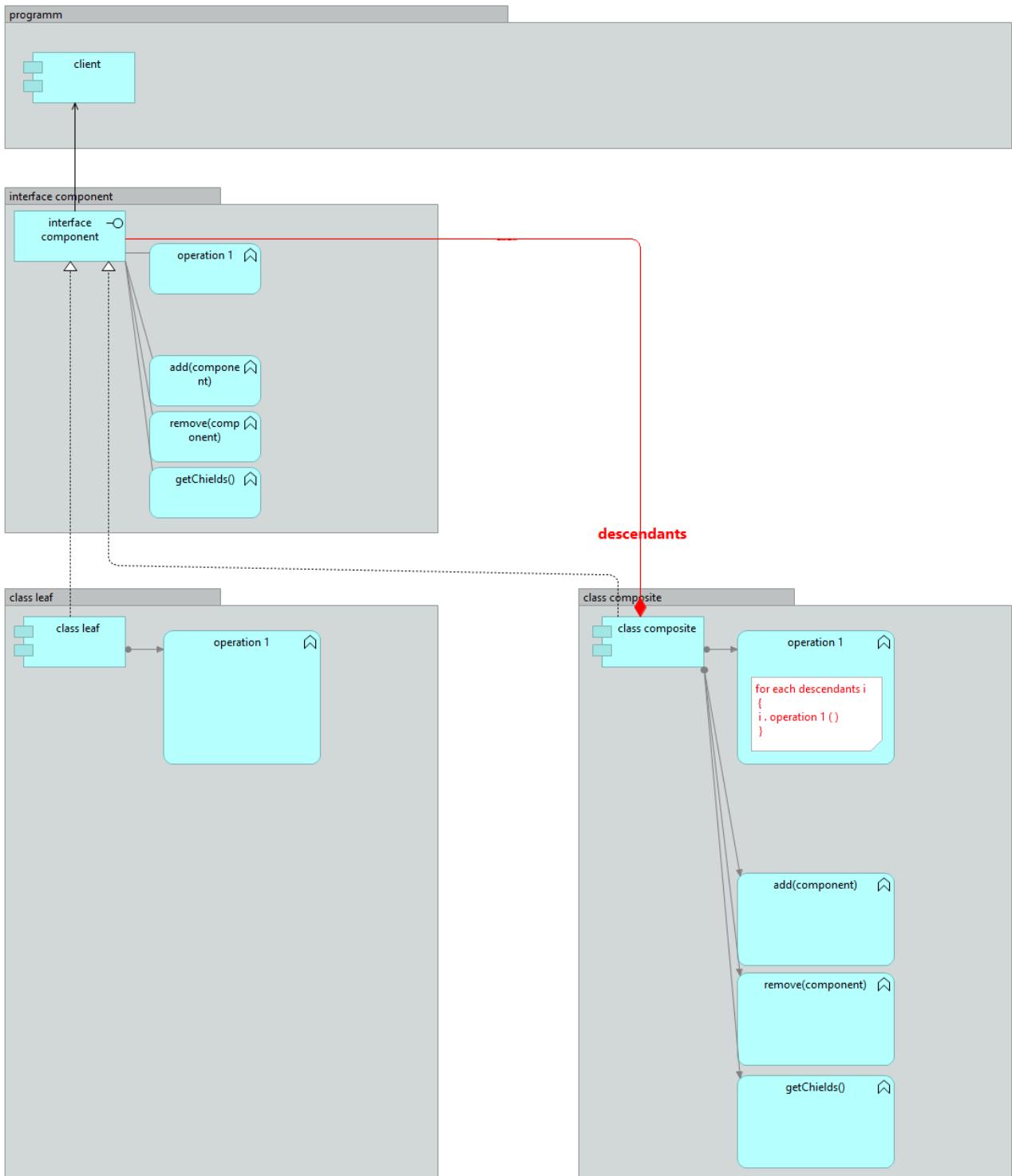
# BRIDGE



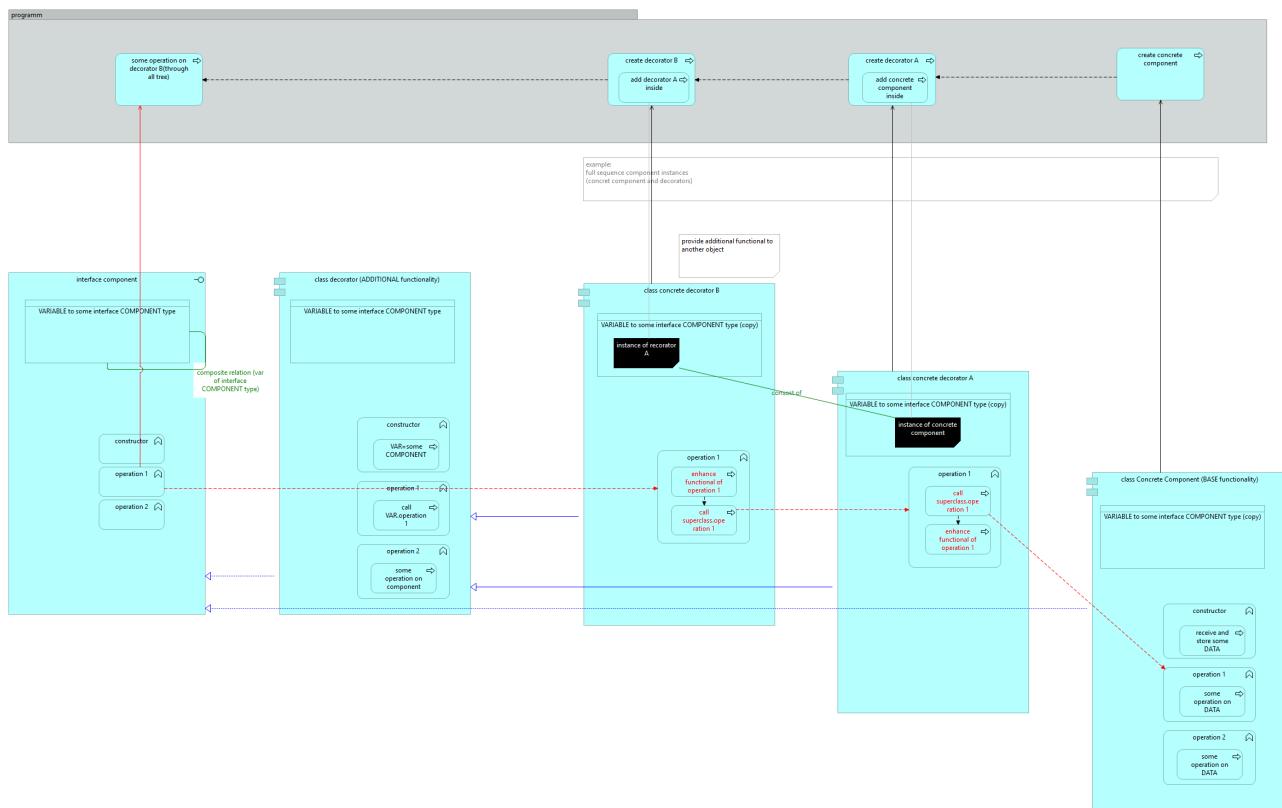


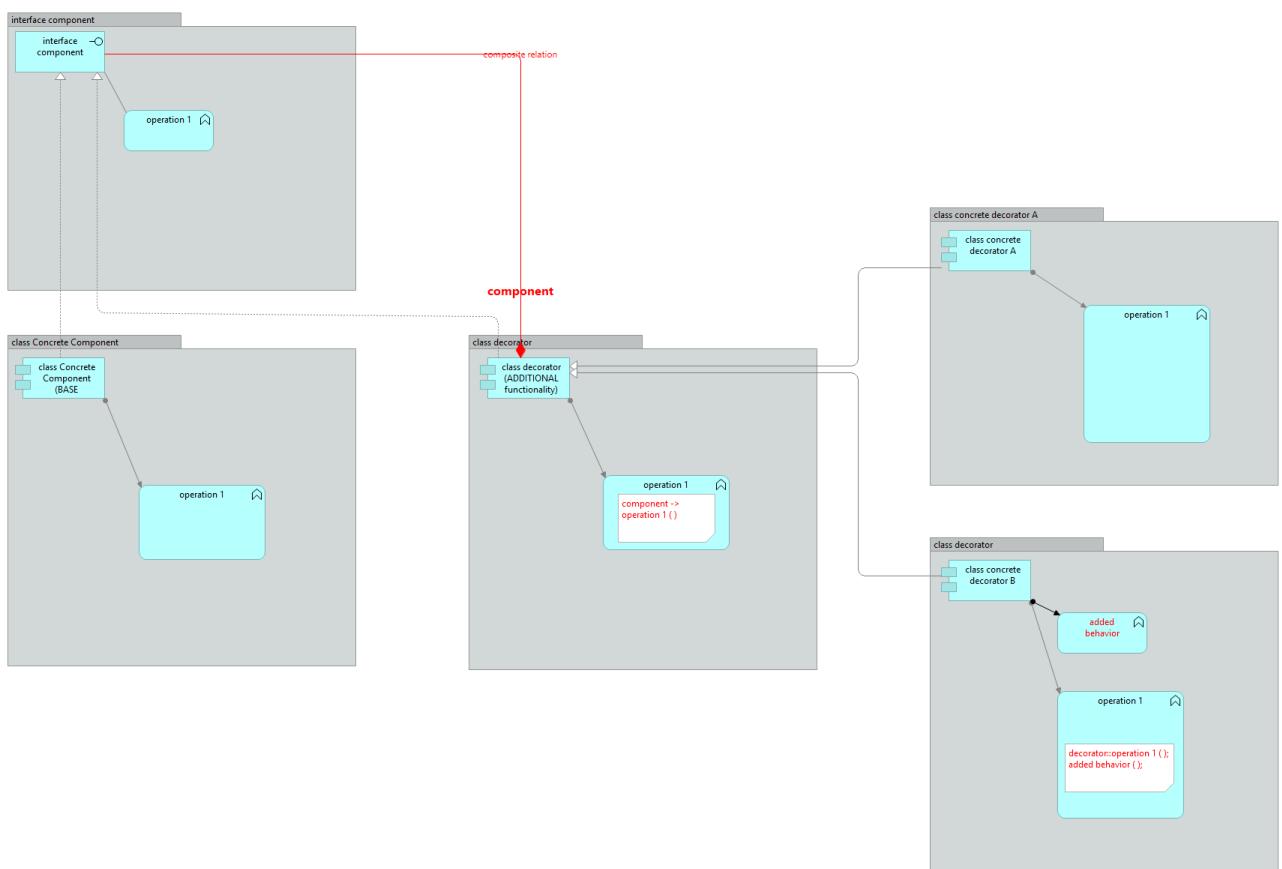
# COMPOSITE



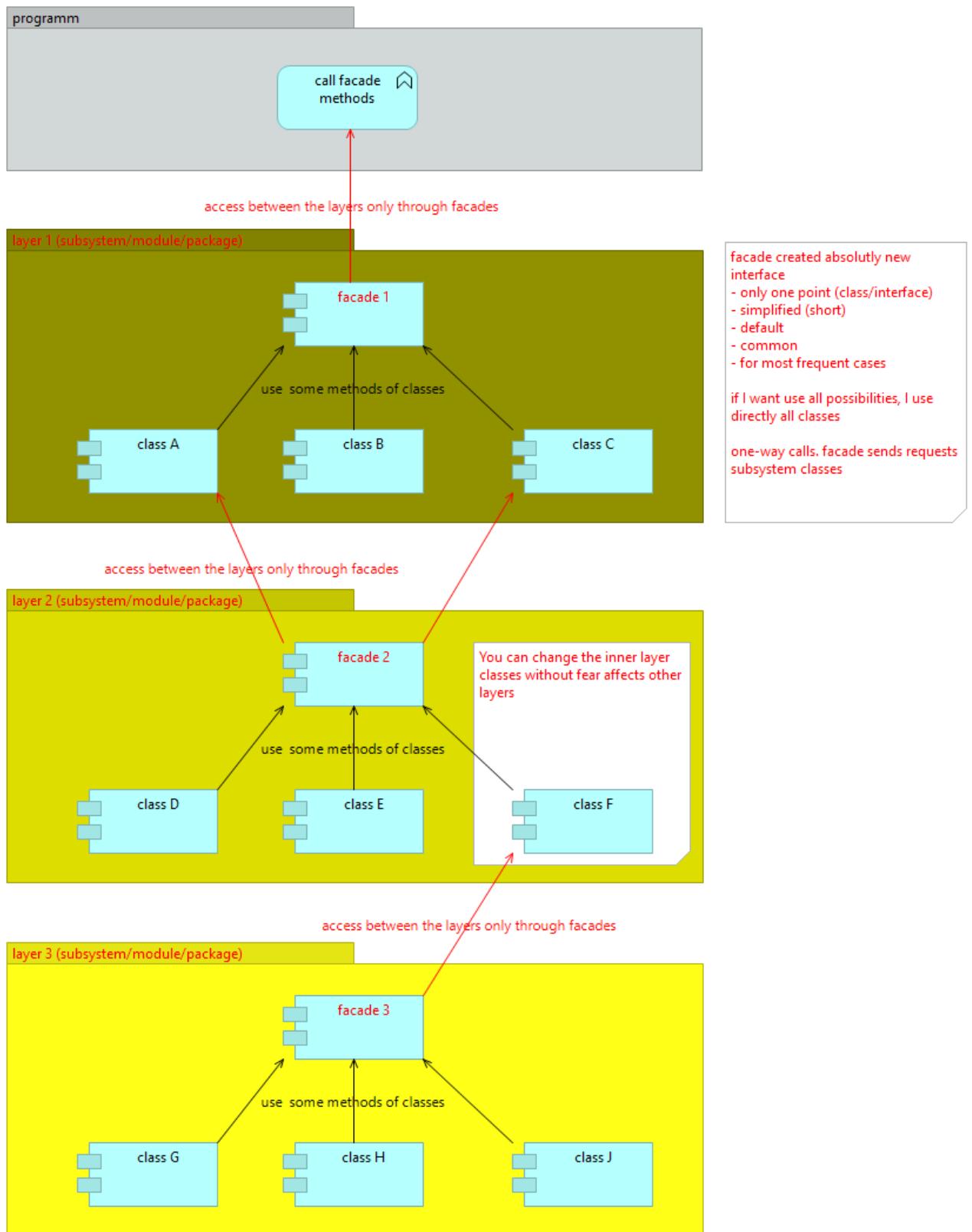


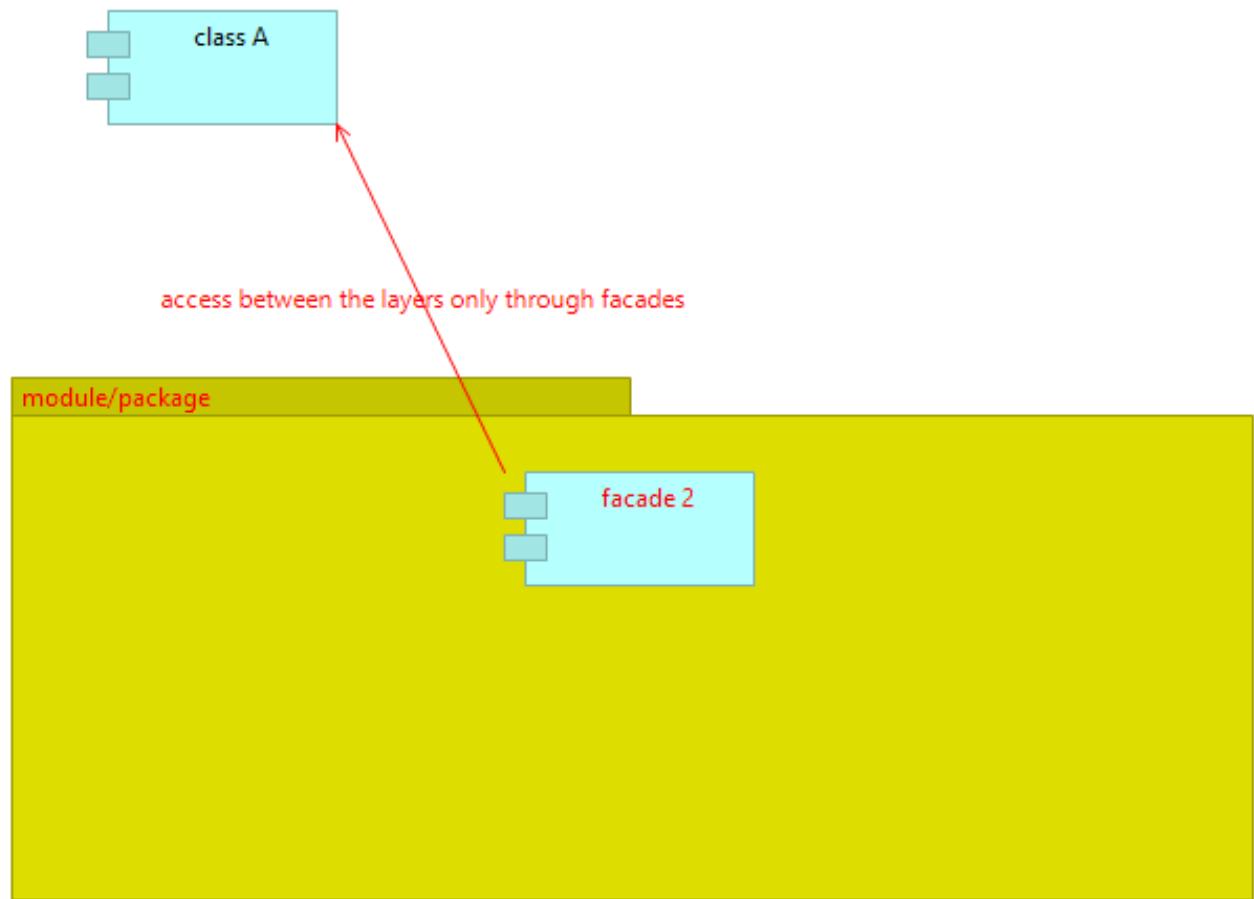
# DECORATOR



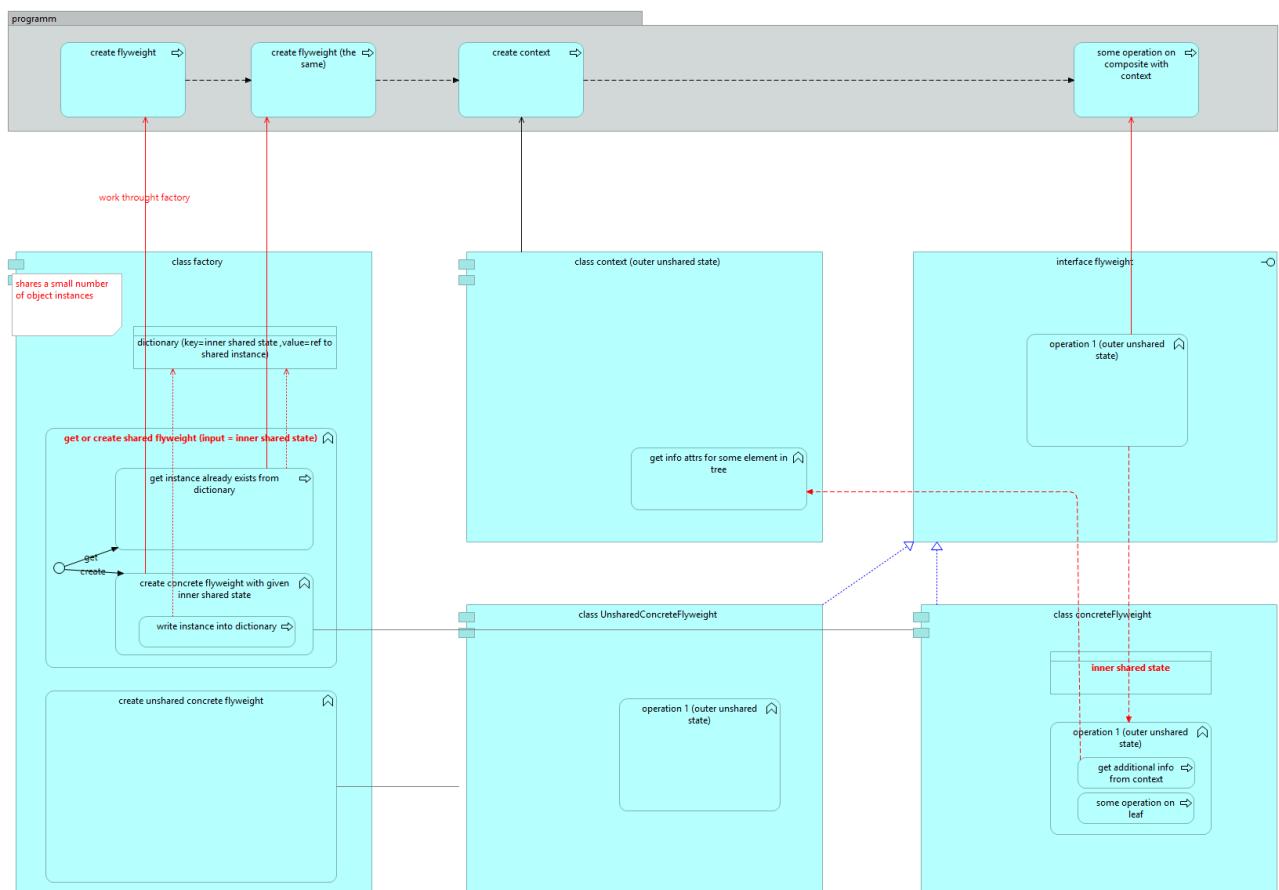


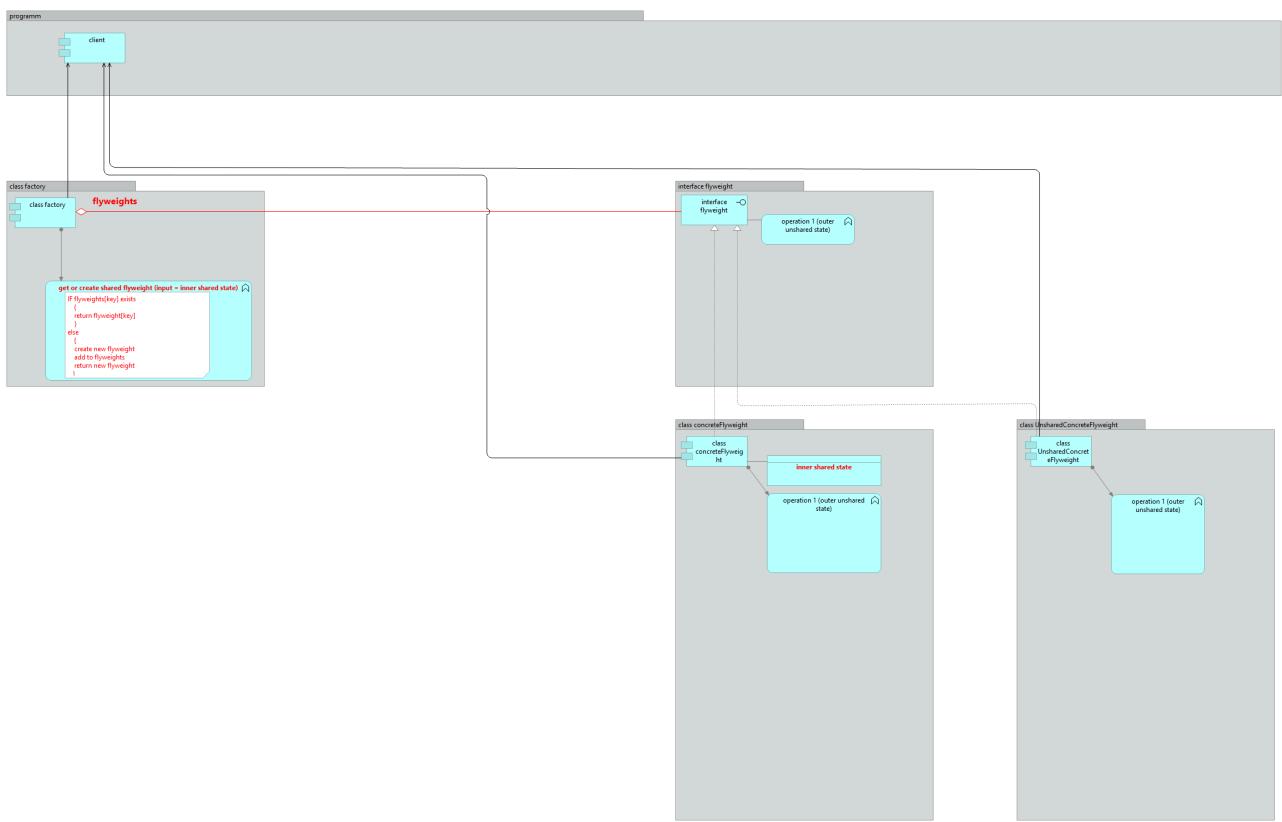
# FACADE



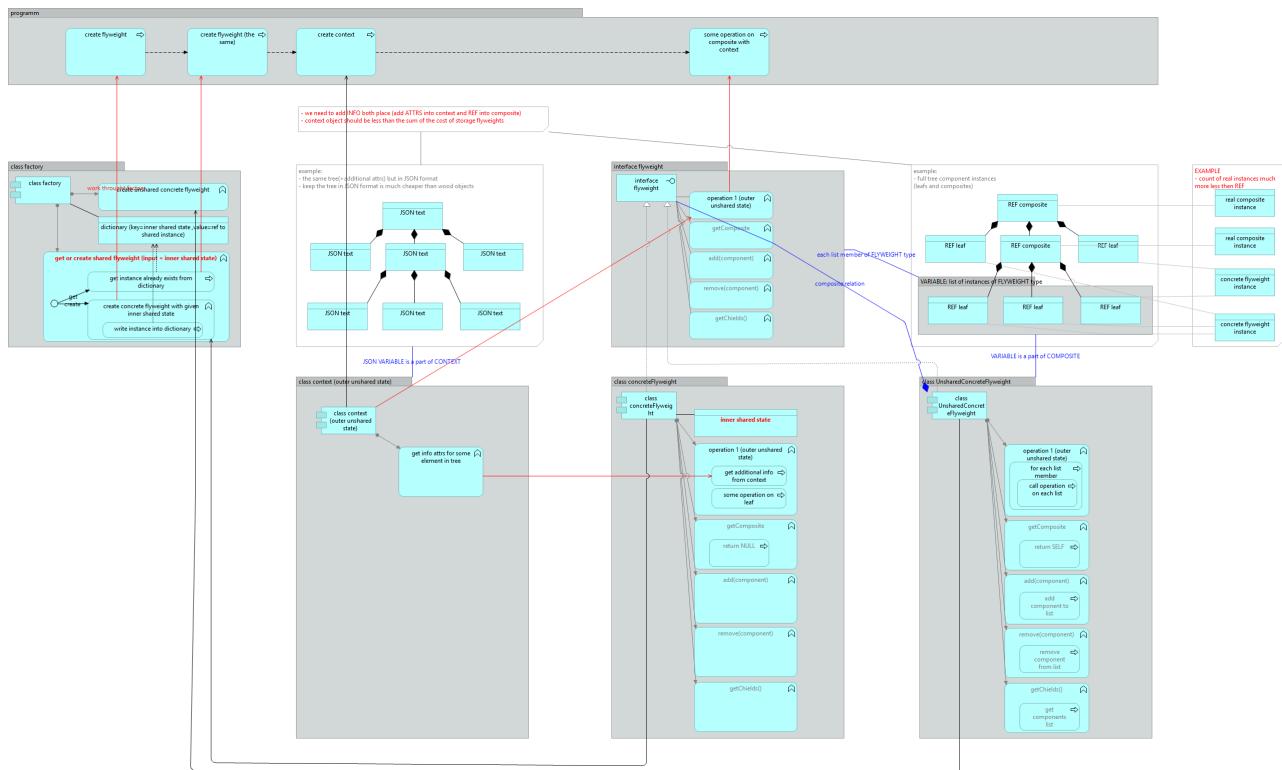


# FLYWEIGHT

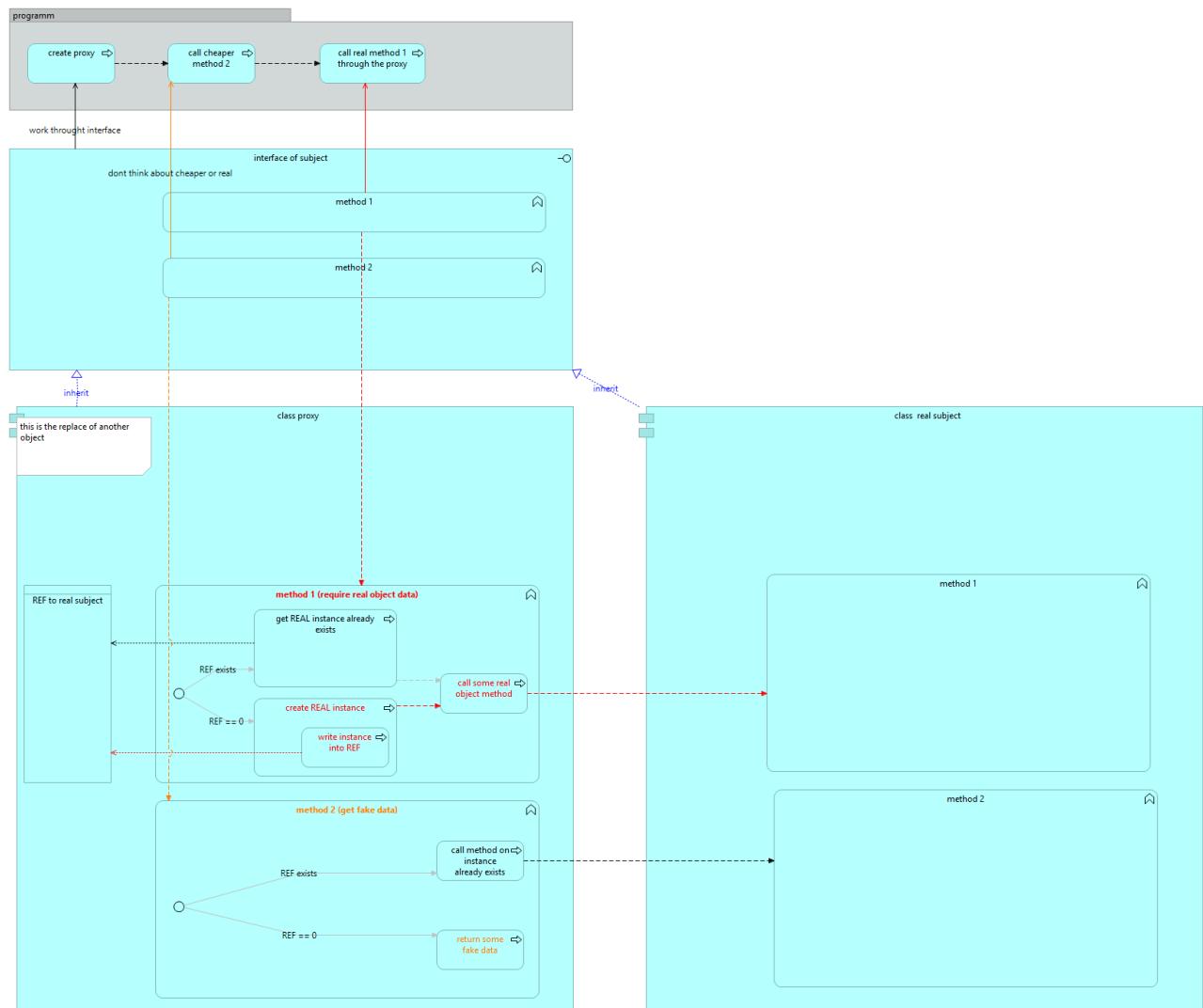


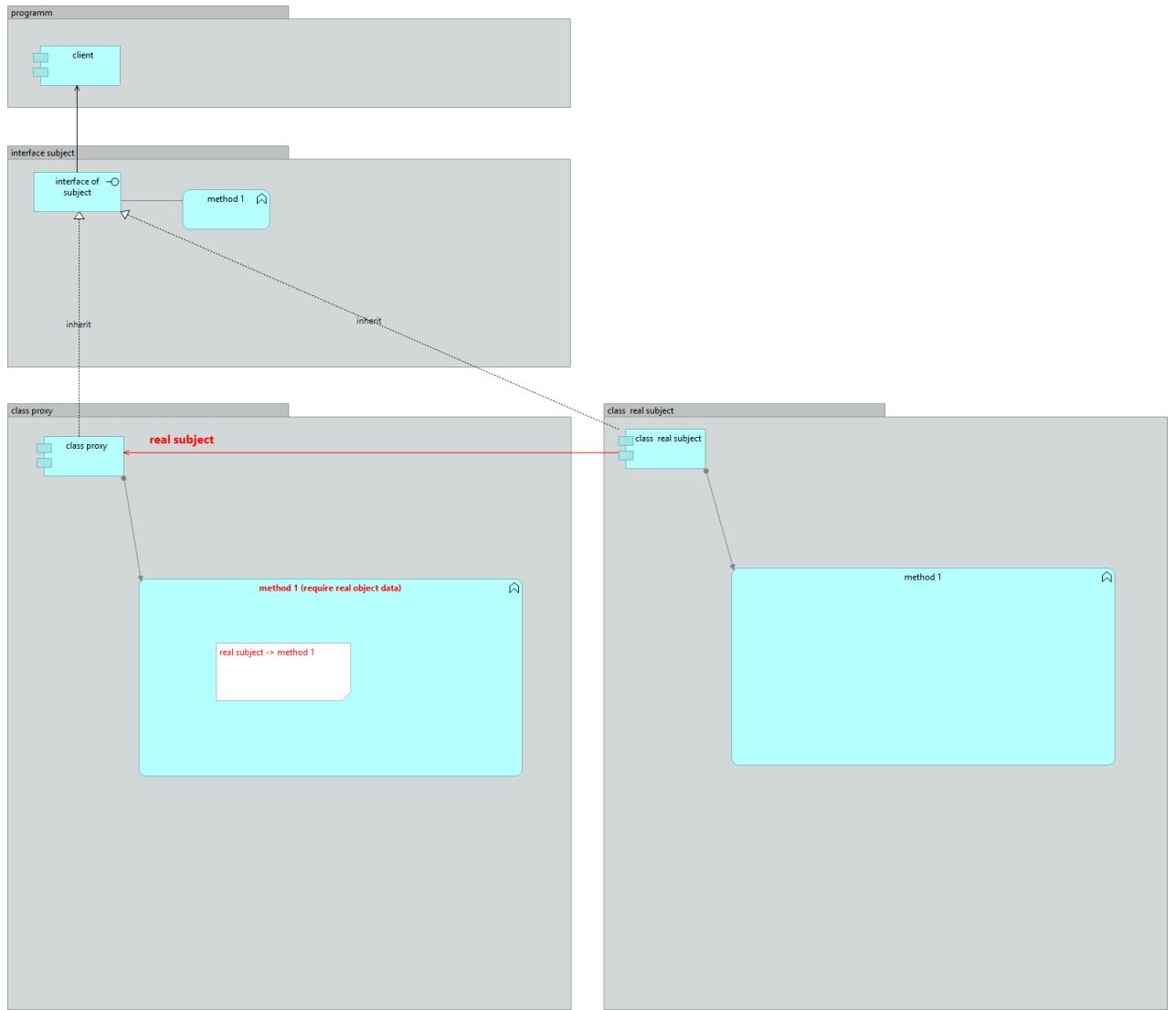


# FLYWEIGHT + COMPOSITE



# PROXY





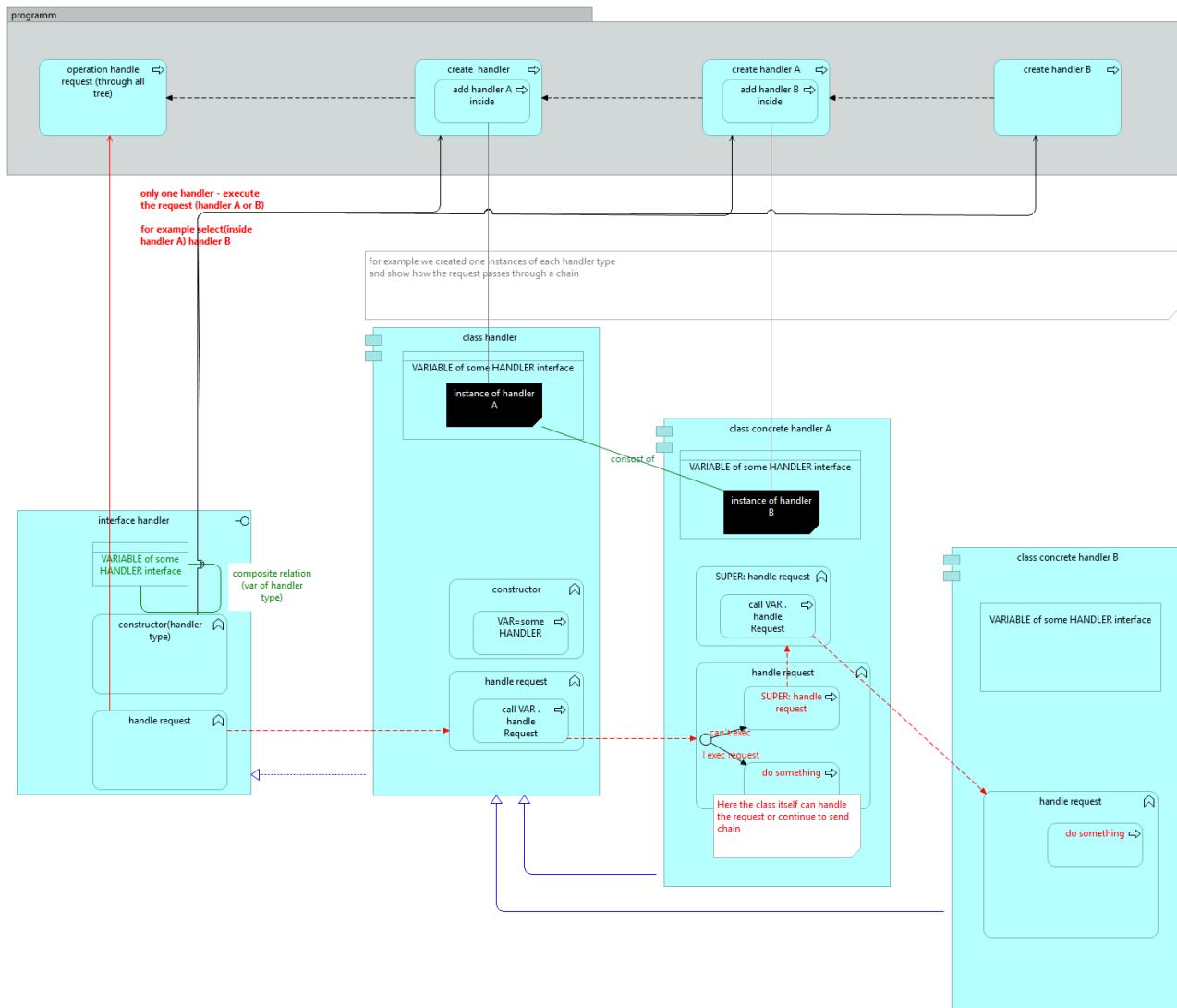
# BEHAVIORAL PATTERNS

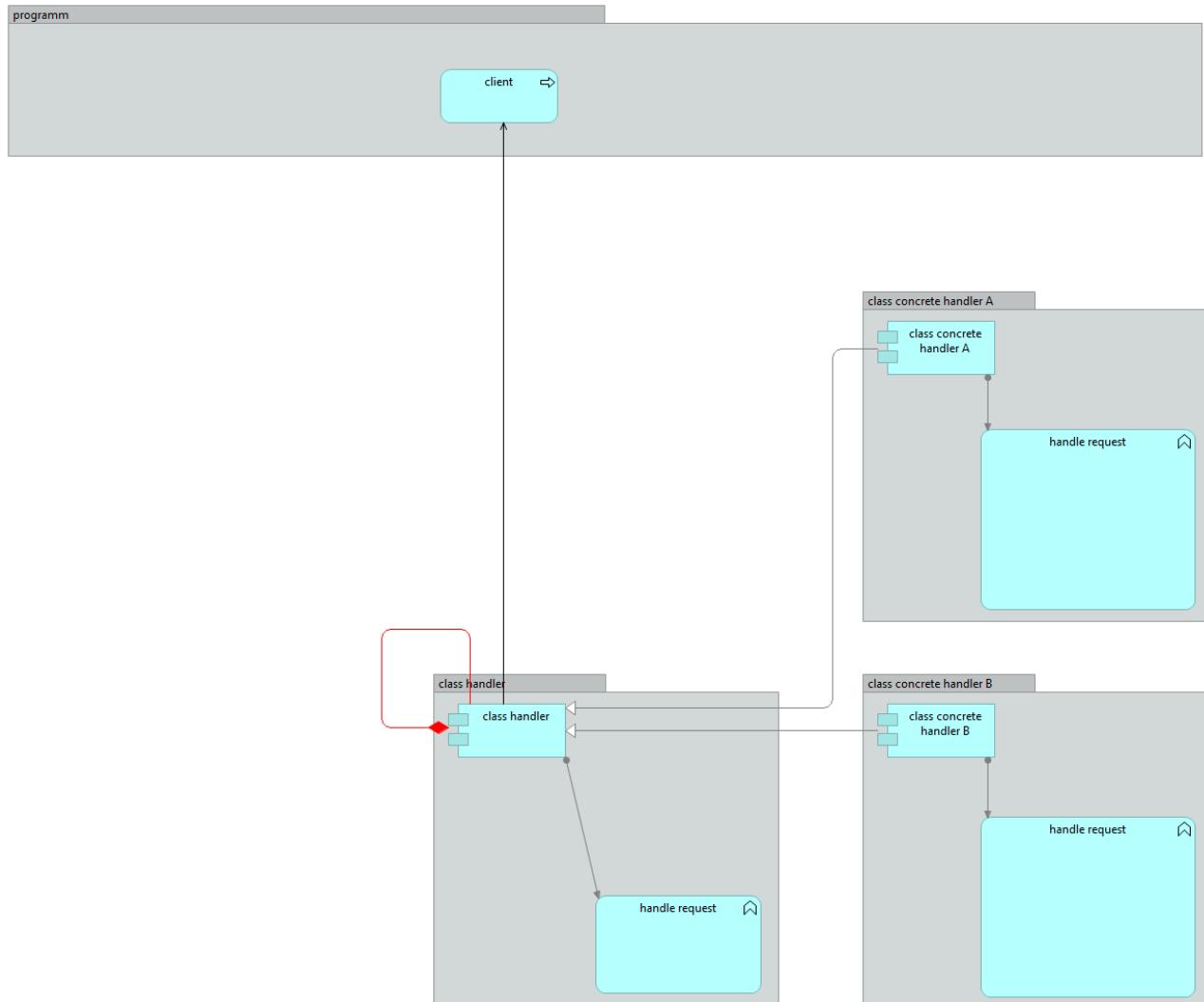
DESIGN PATTERNS

GoF

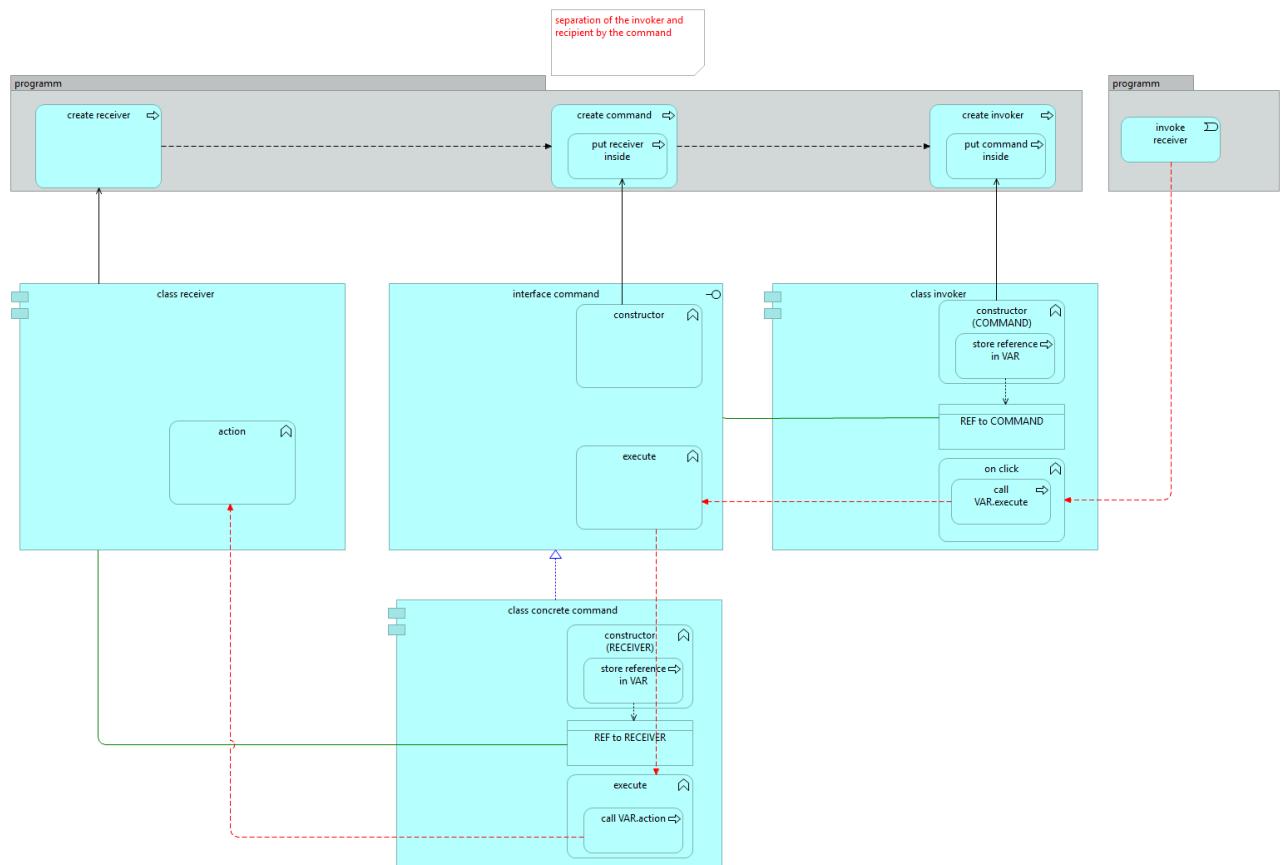


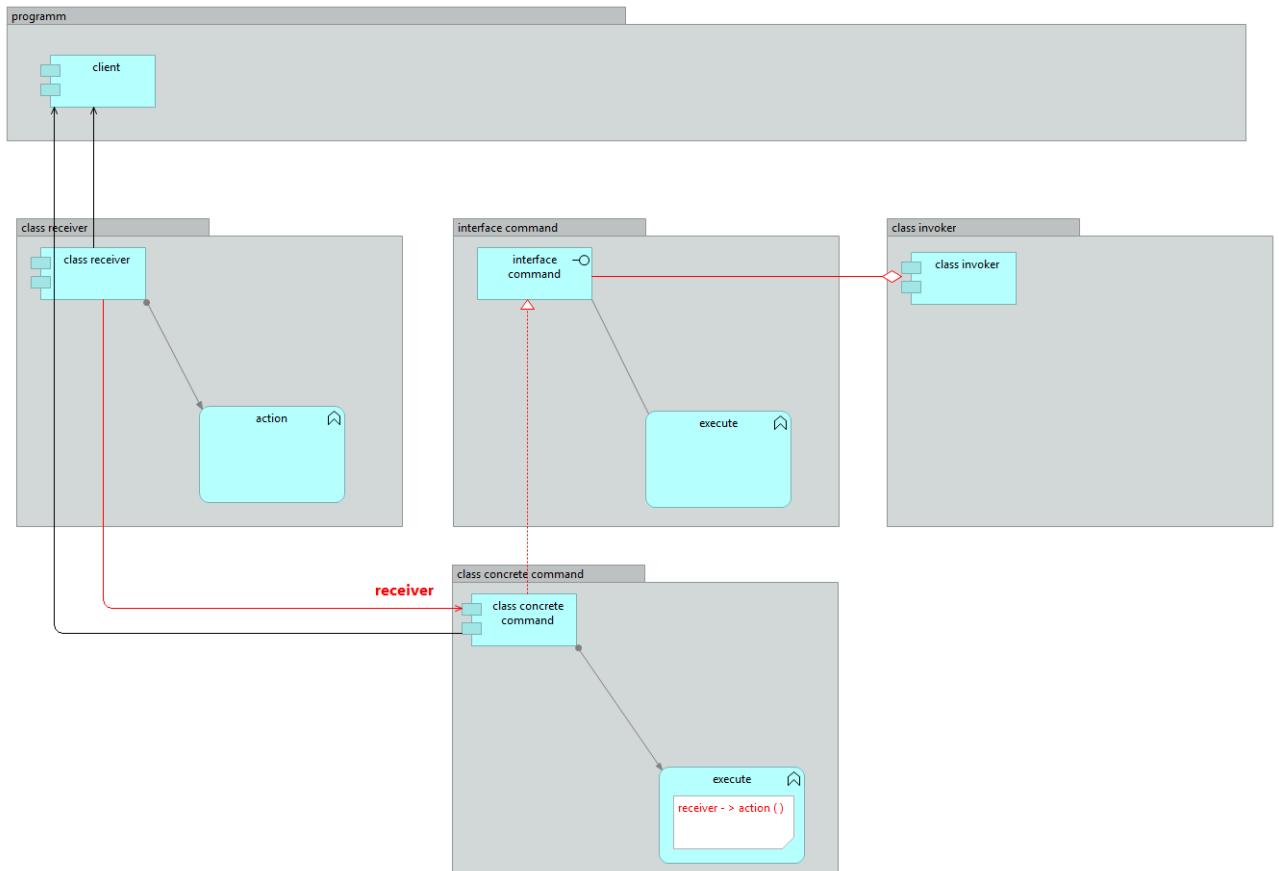
# CHAIN OF RESPONSIBILITY



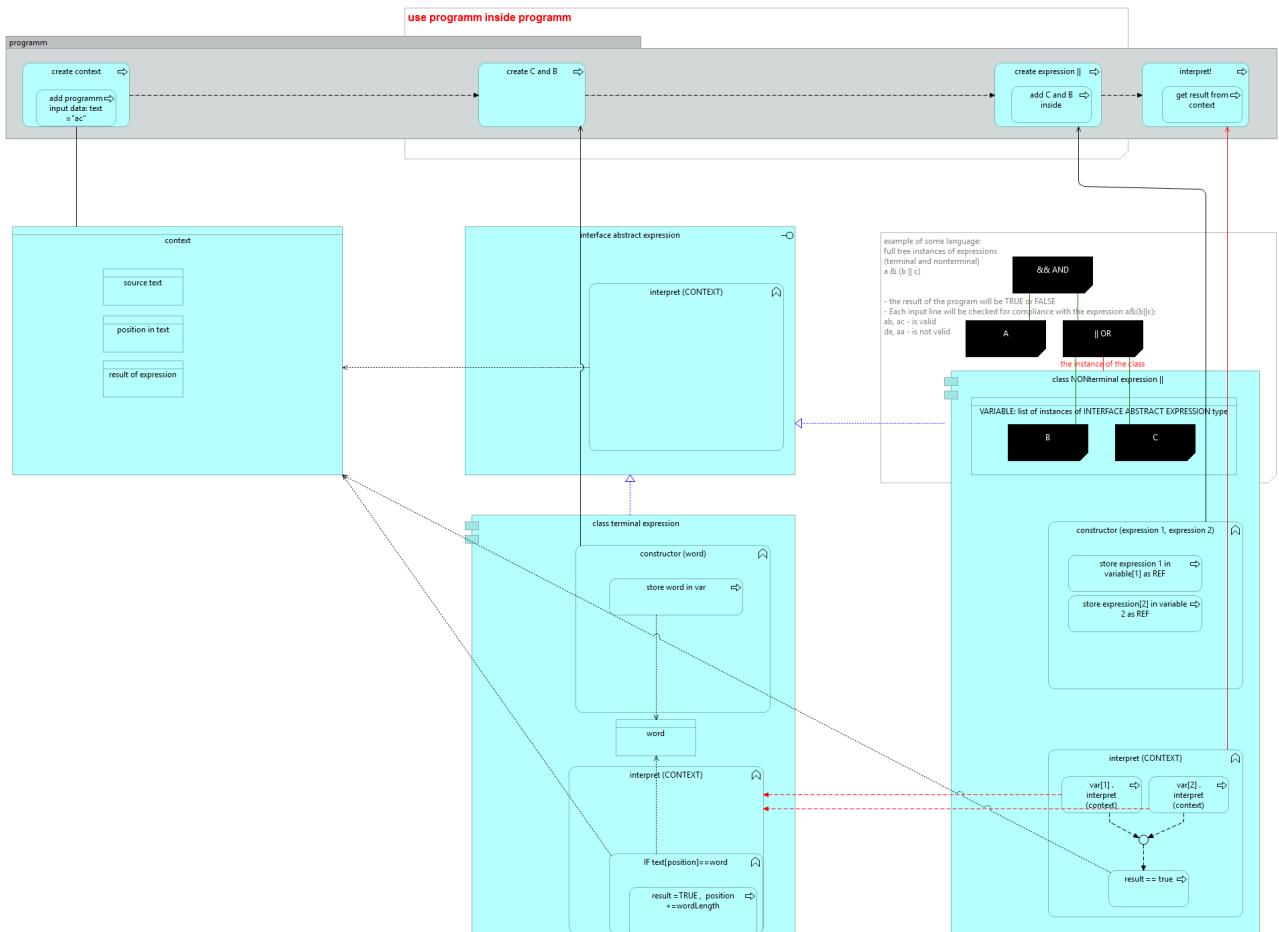


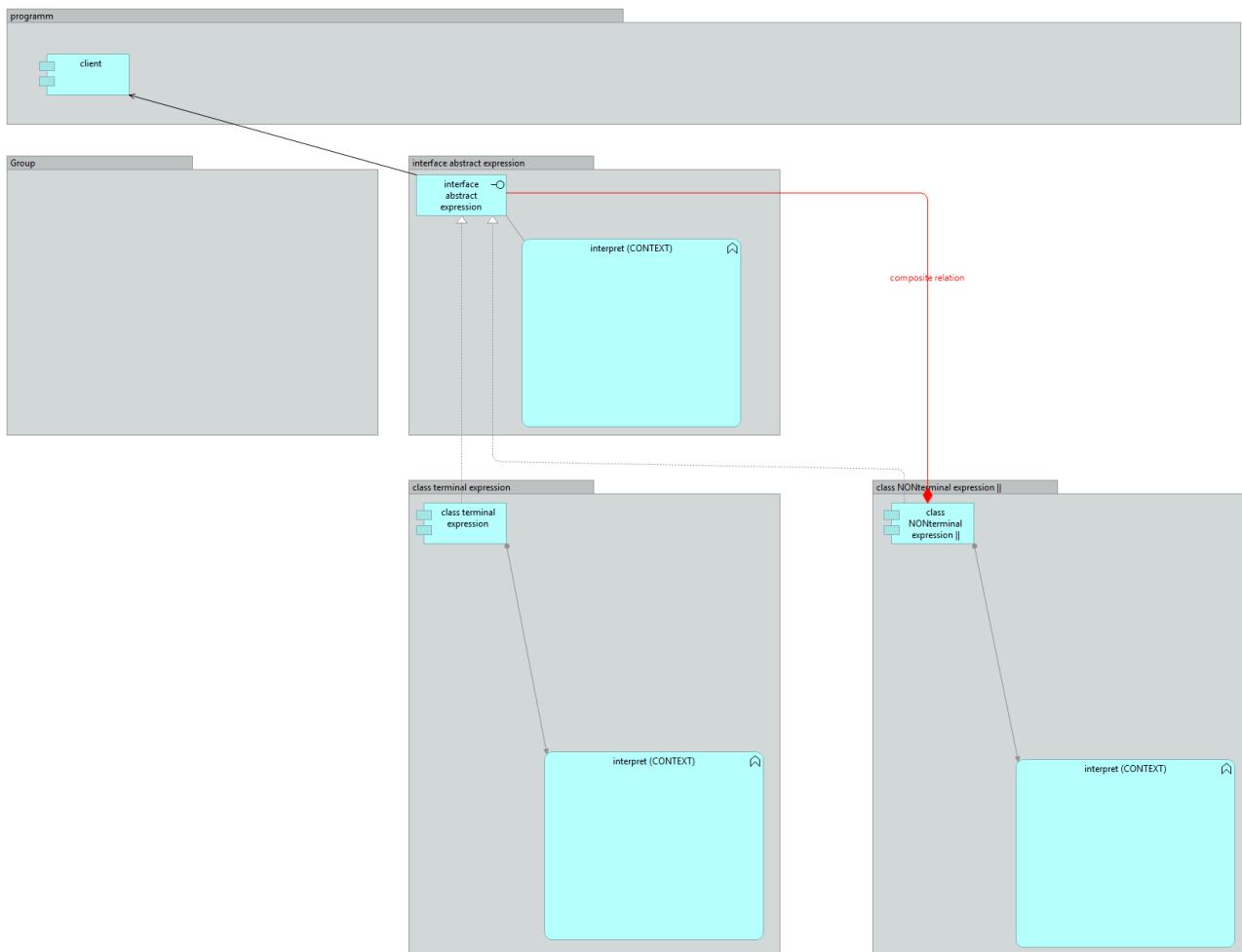
# COMMAND



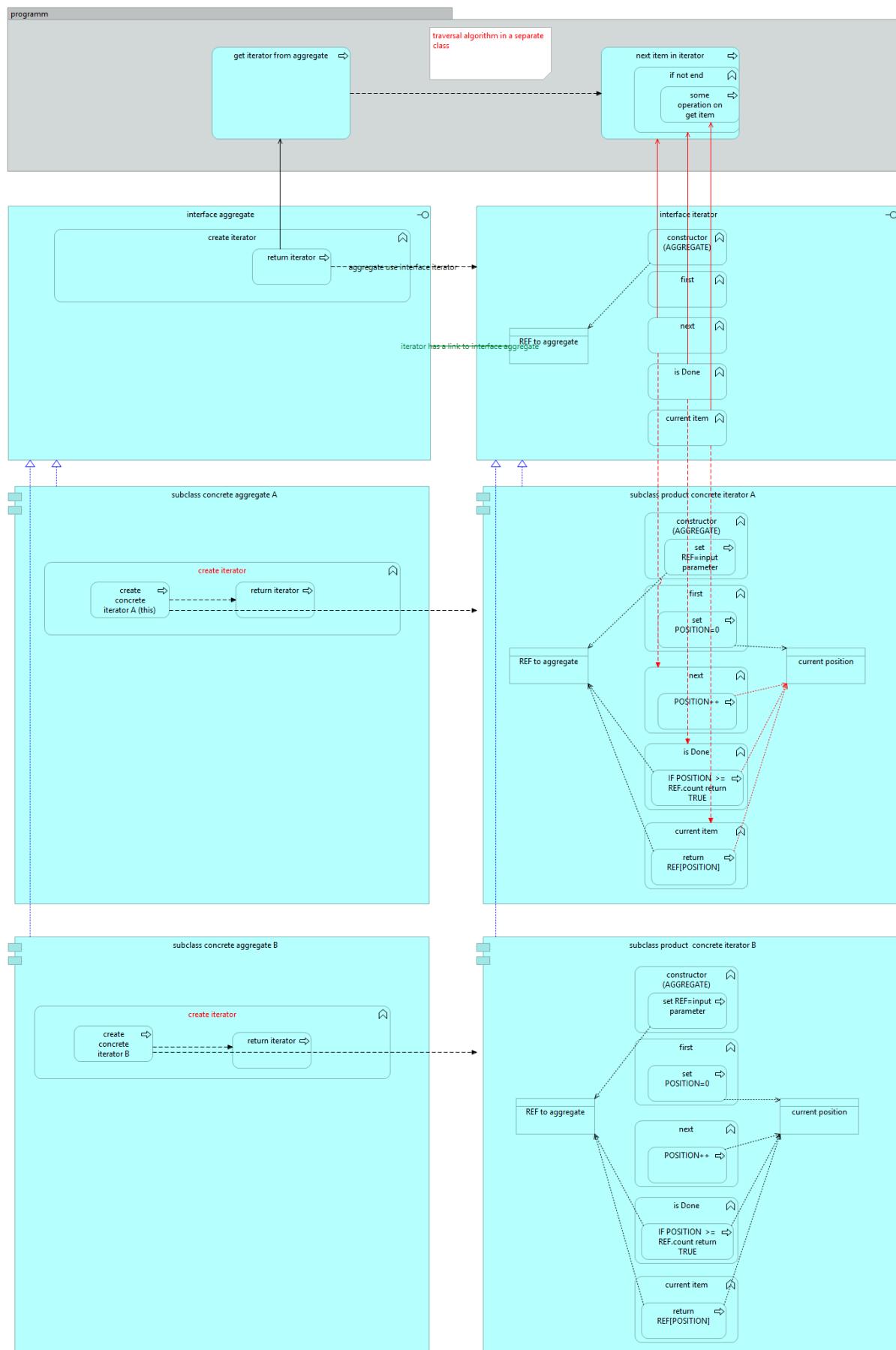


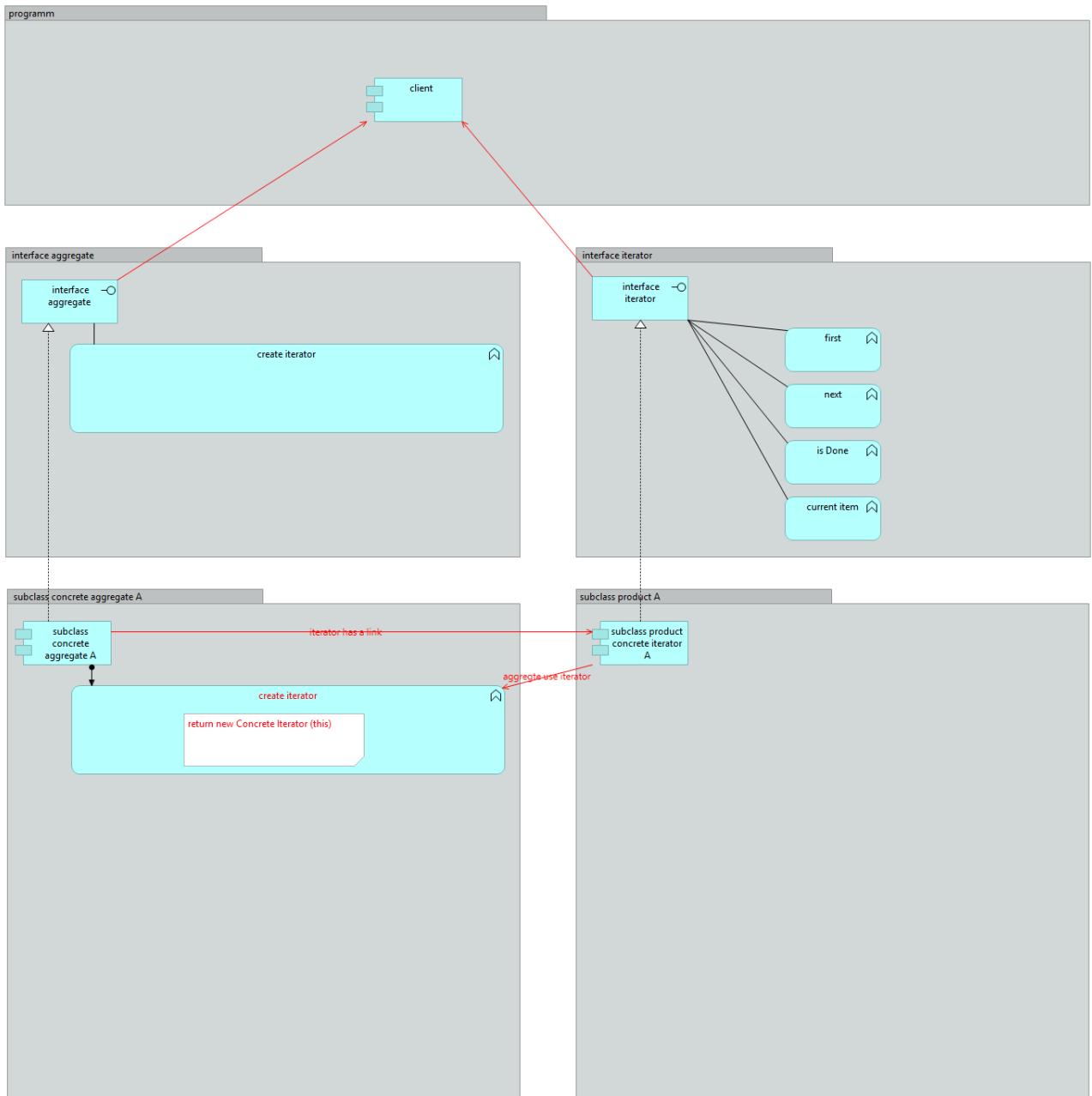
# INTERPRETER



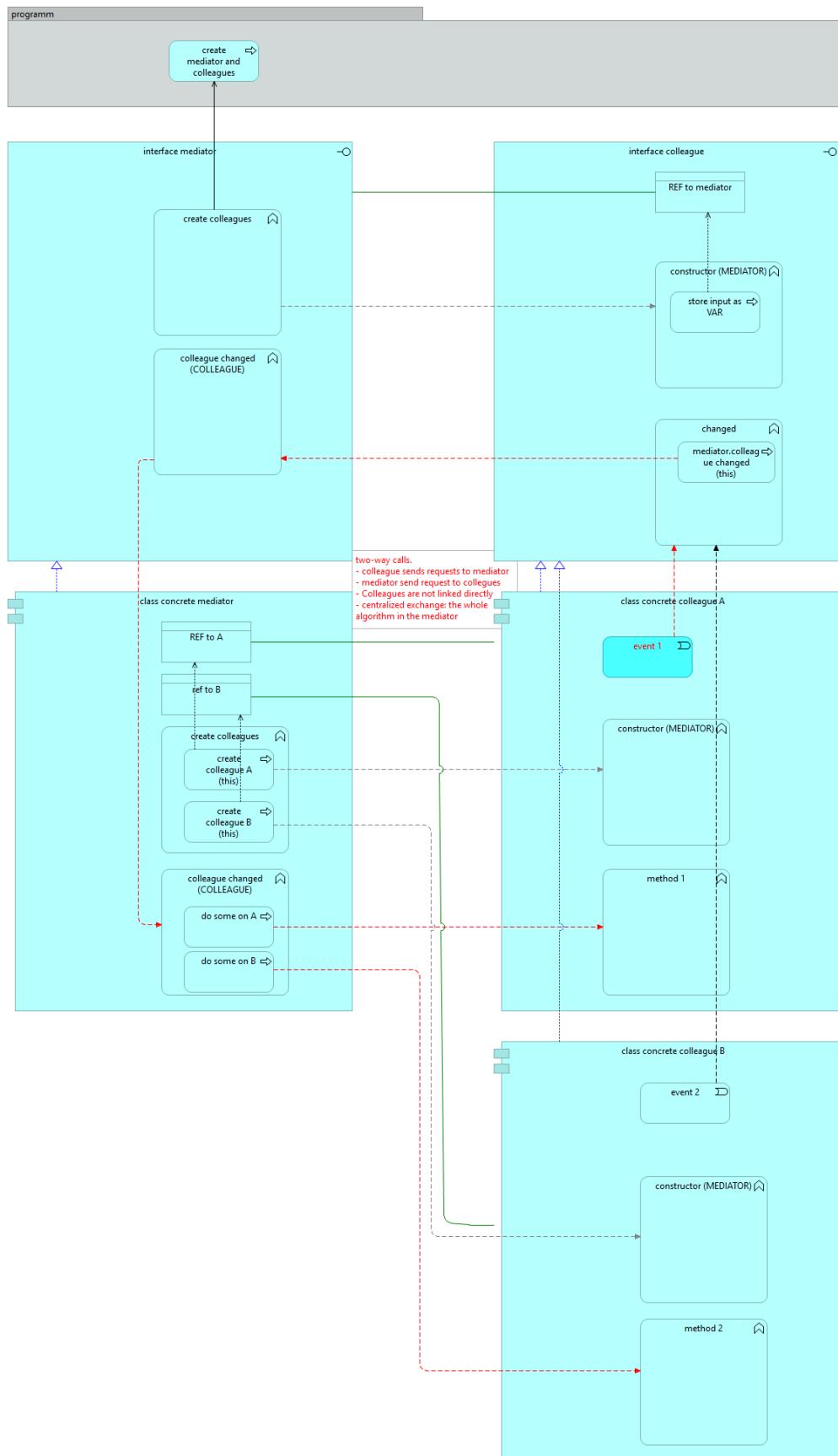


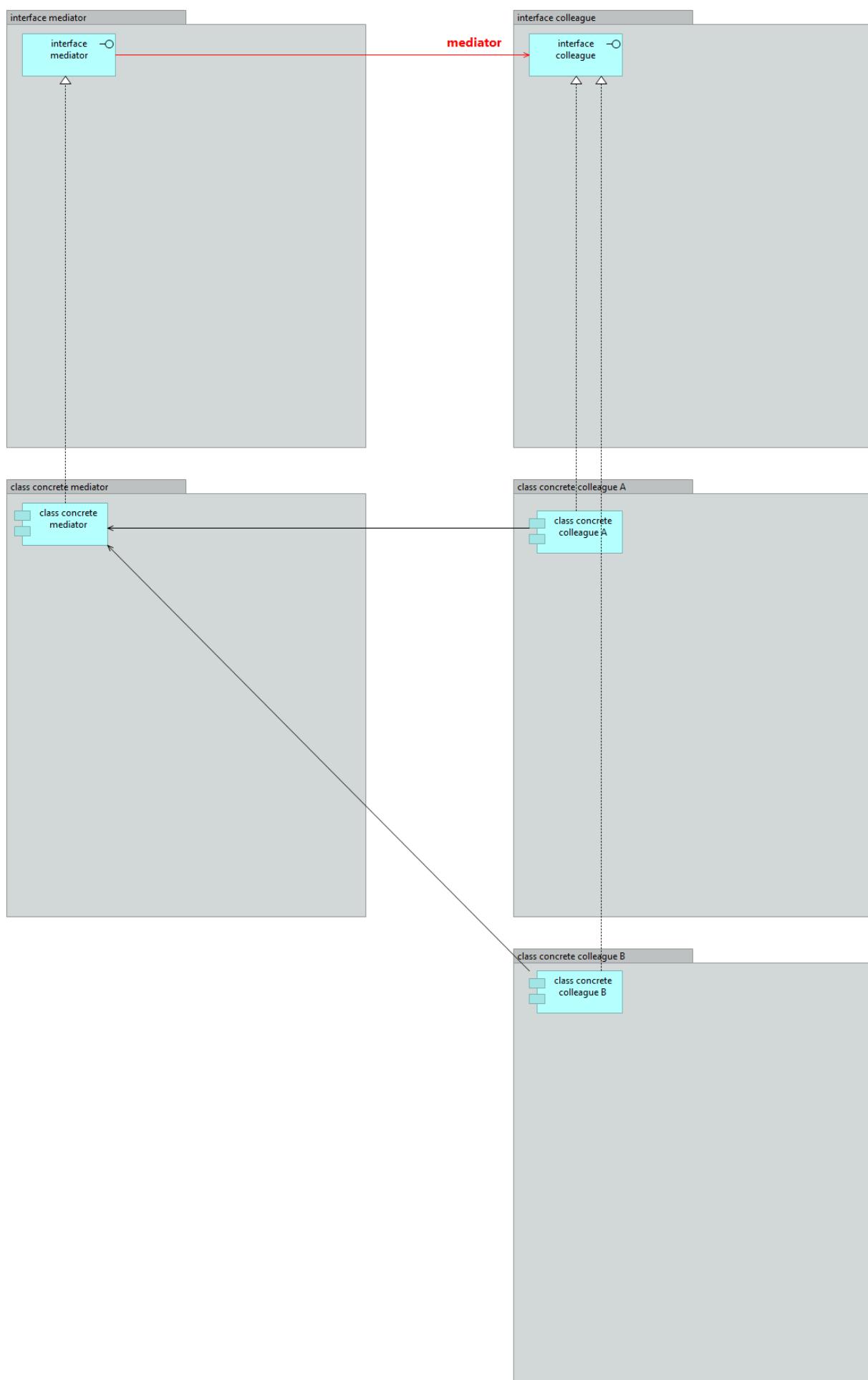
# ITERATOR



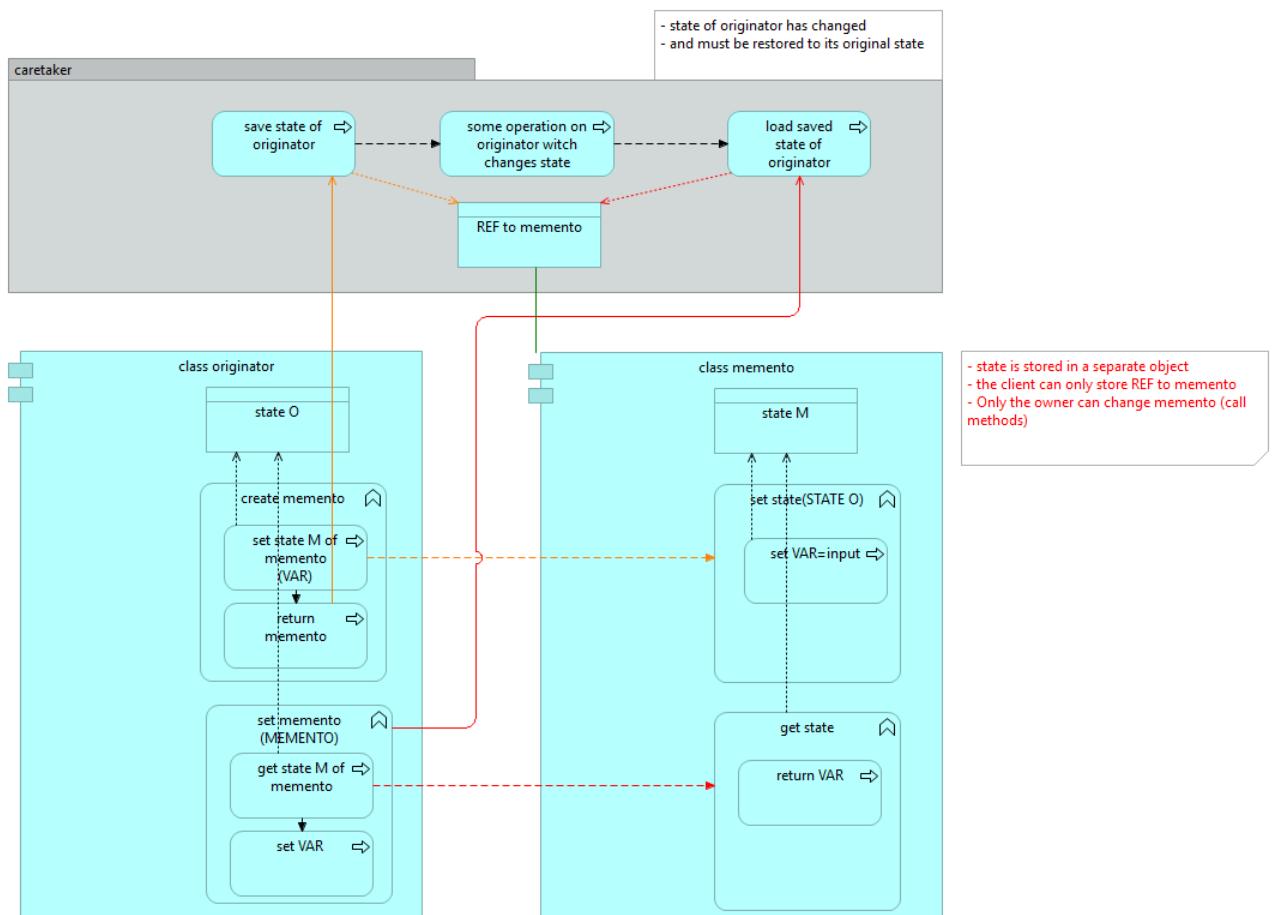


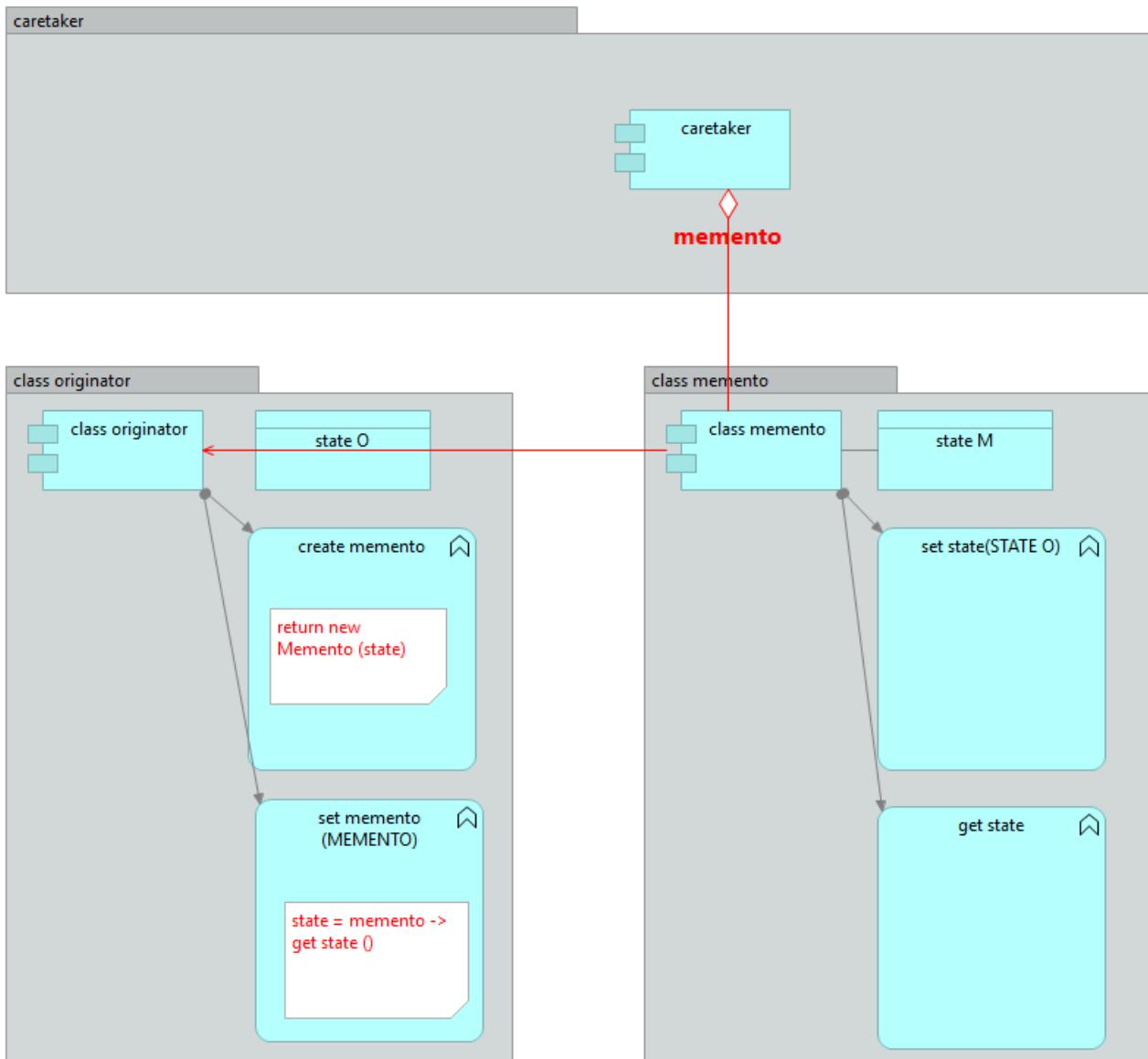
# MEDIATOR



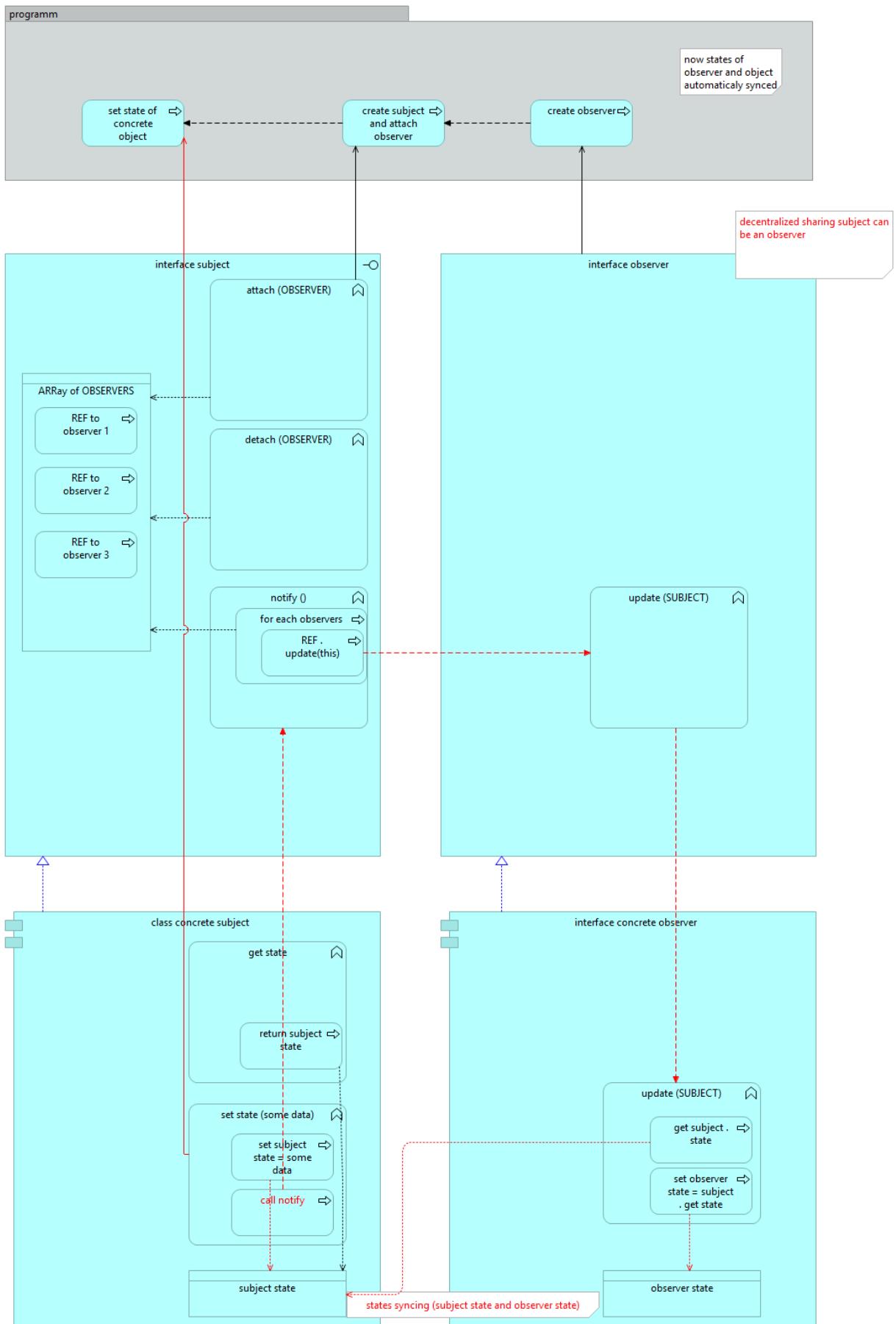


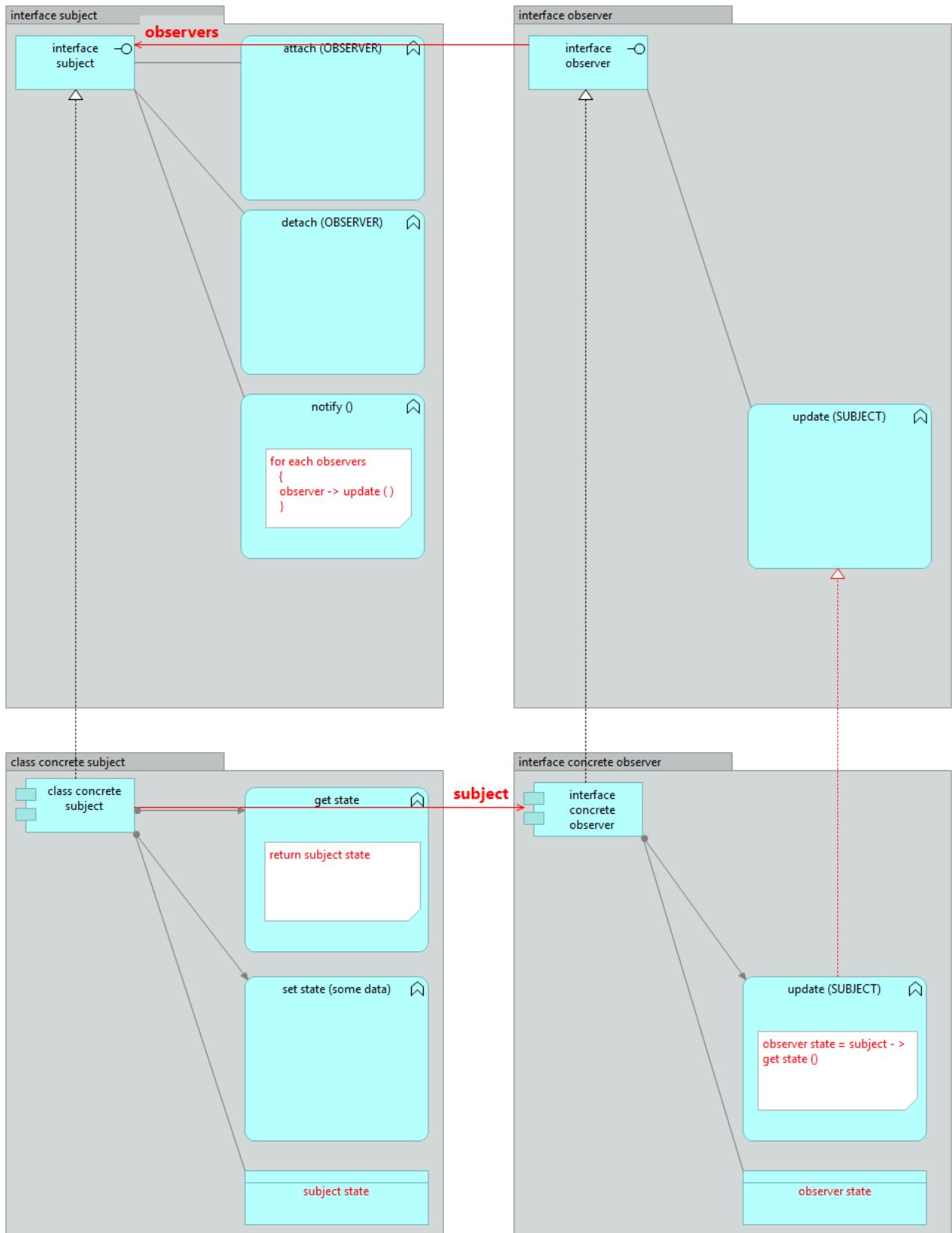
# MEMENTO



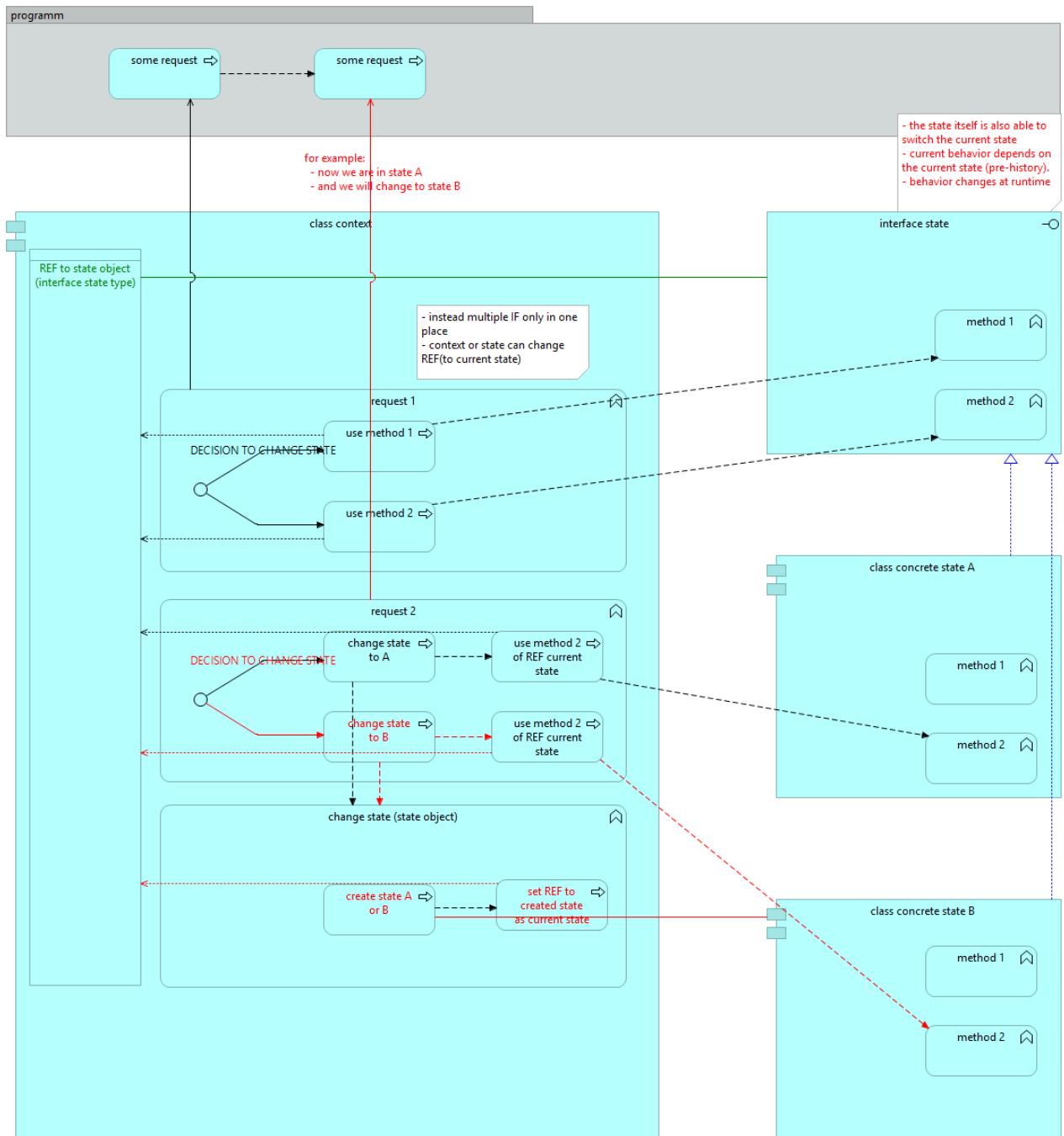


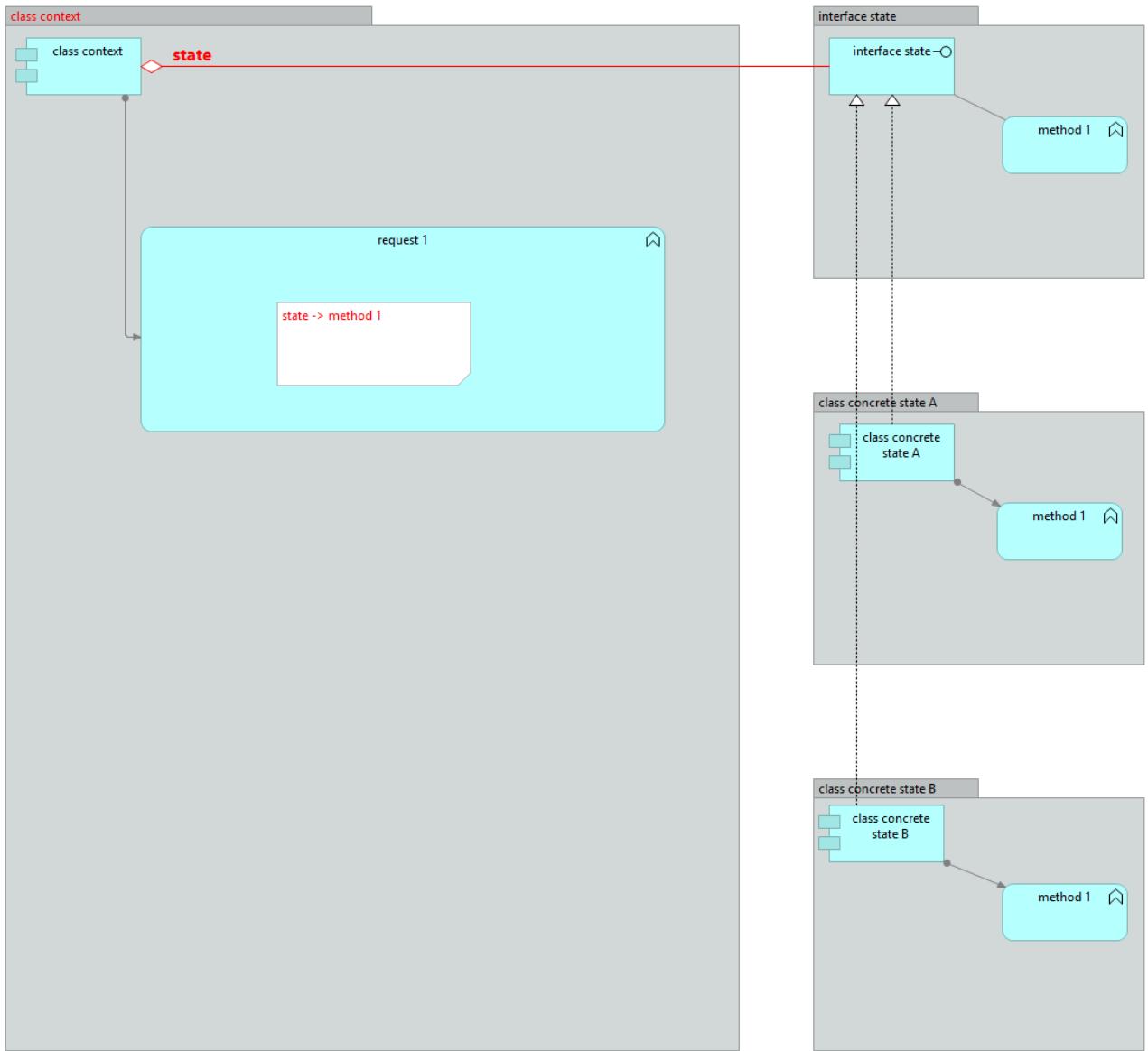
# OBSERVER



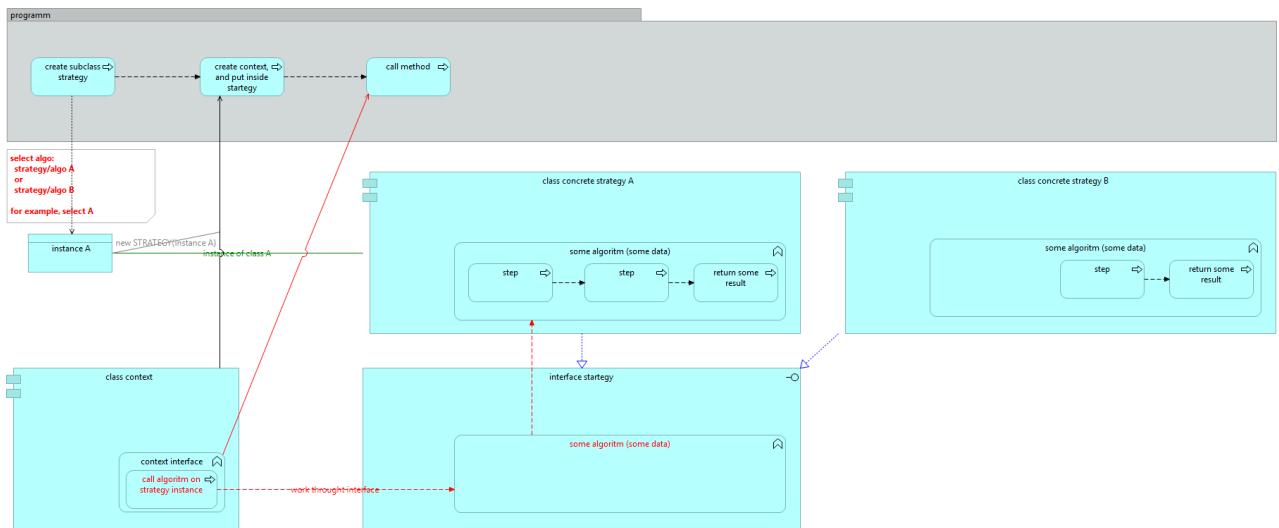


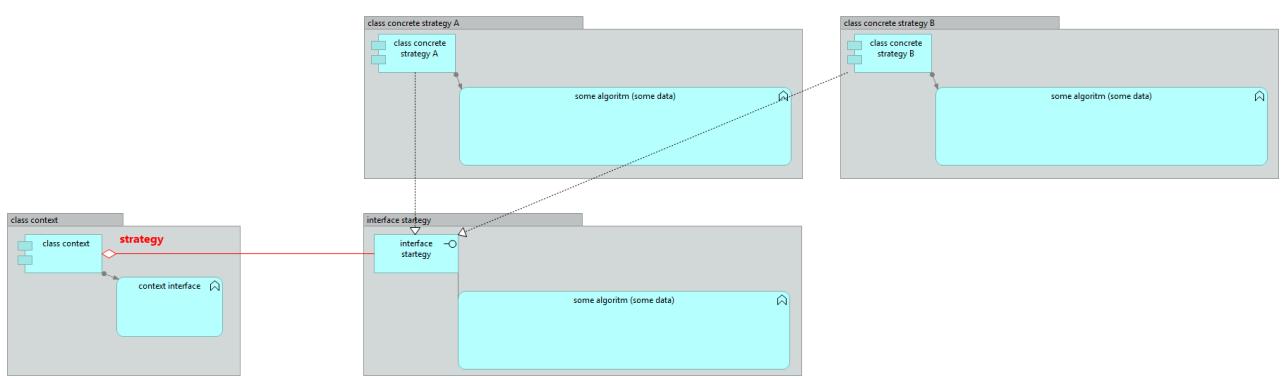
# STATE



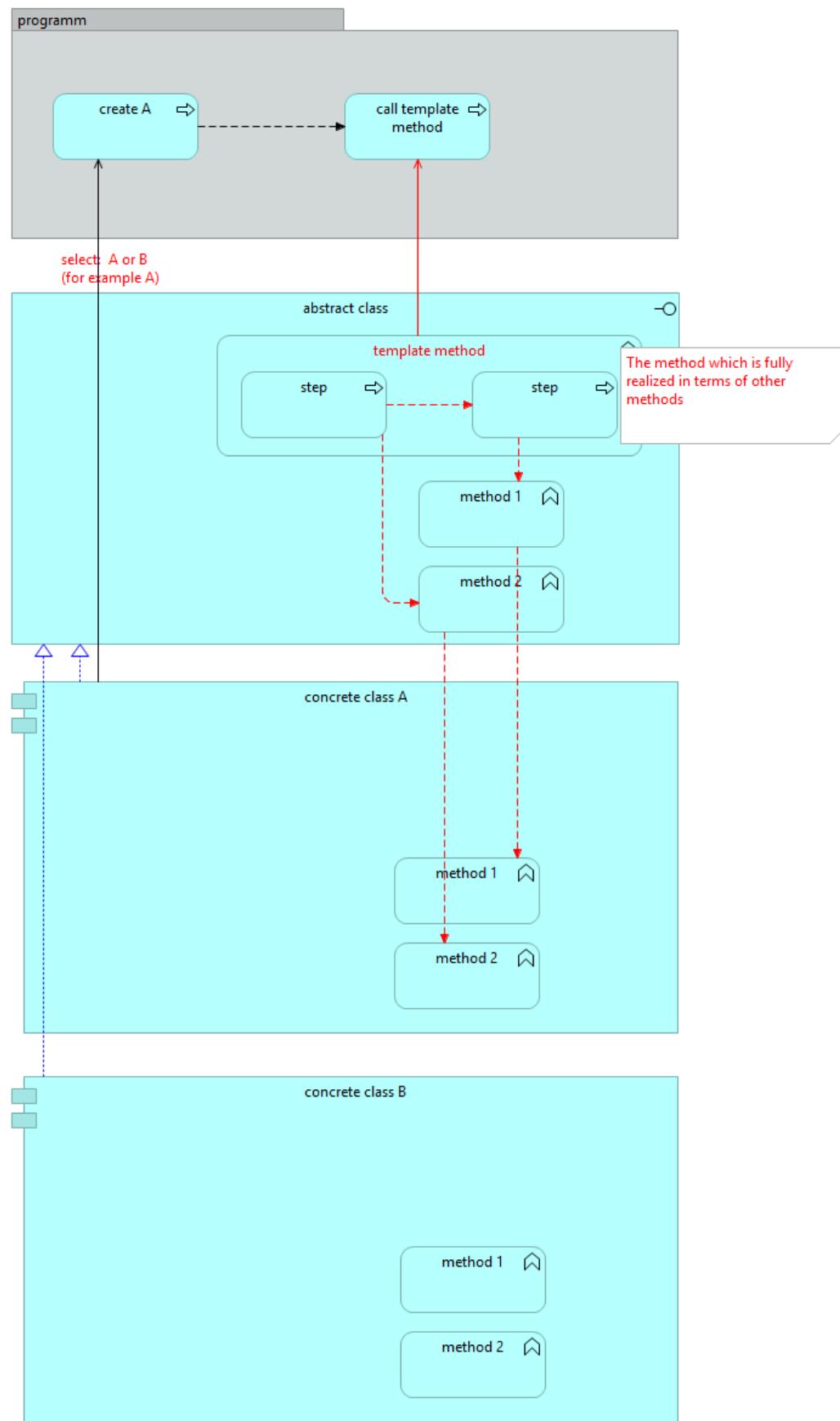


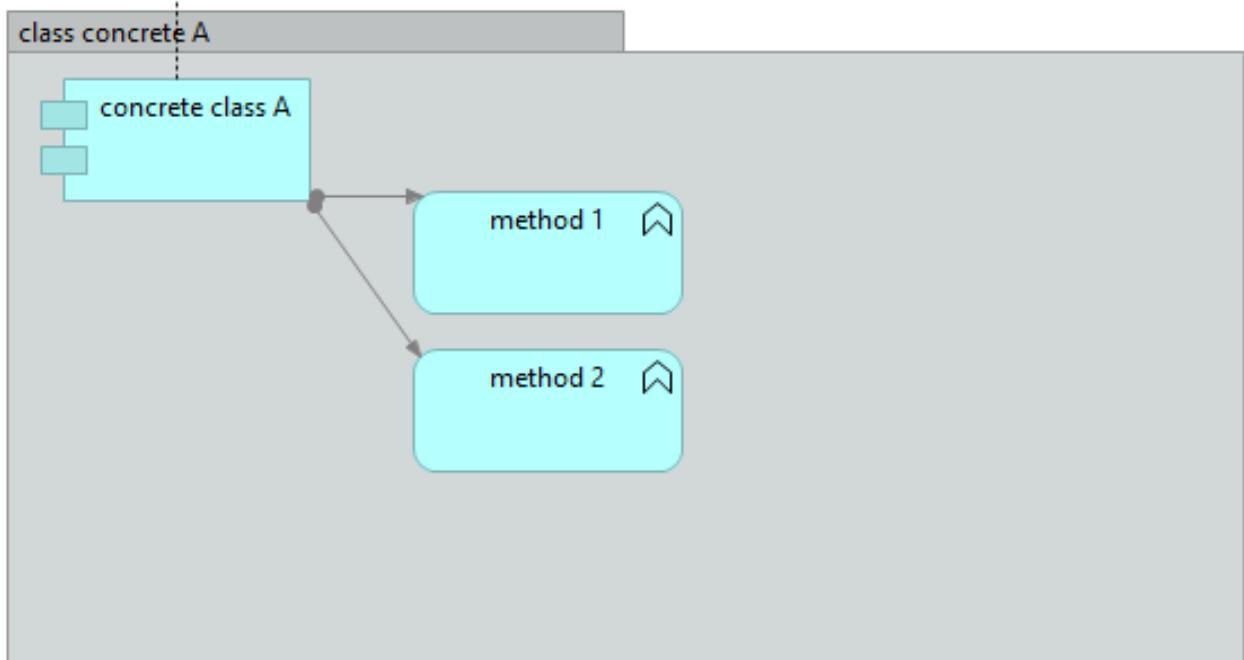
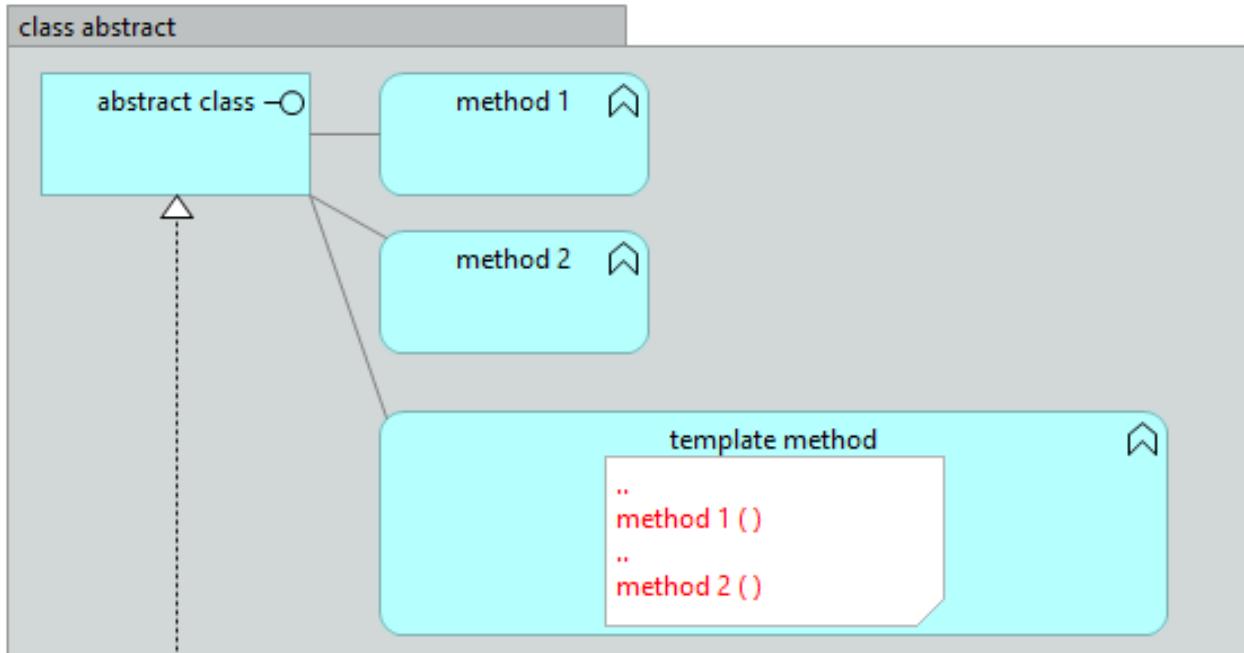
# STRATEGY



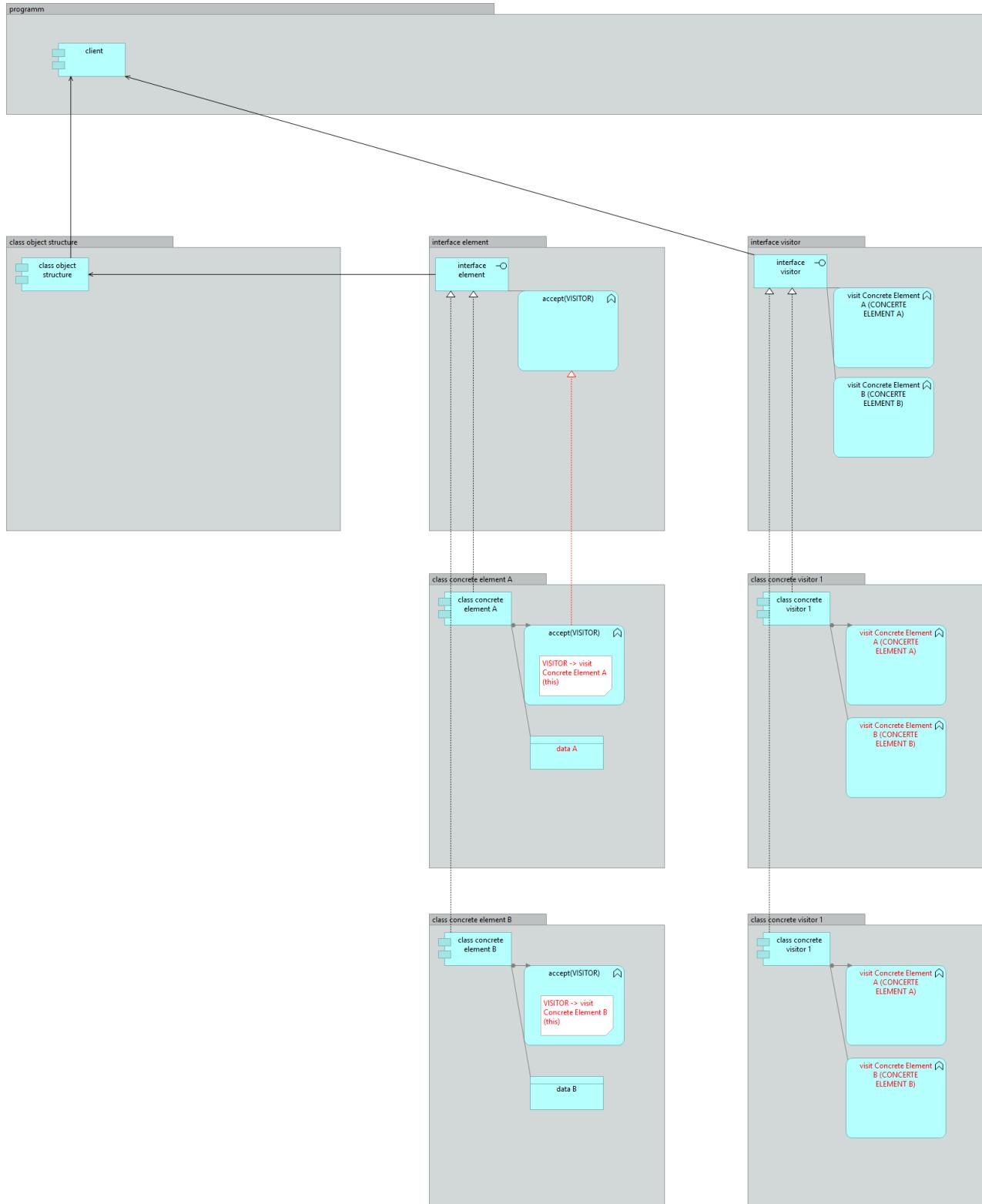


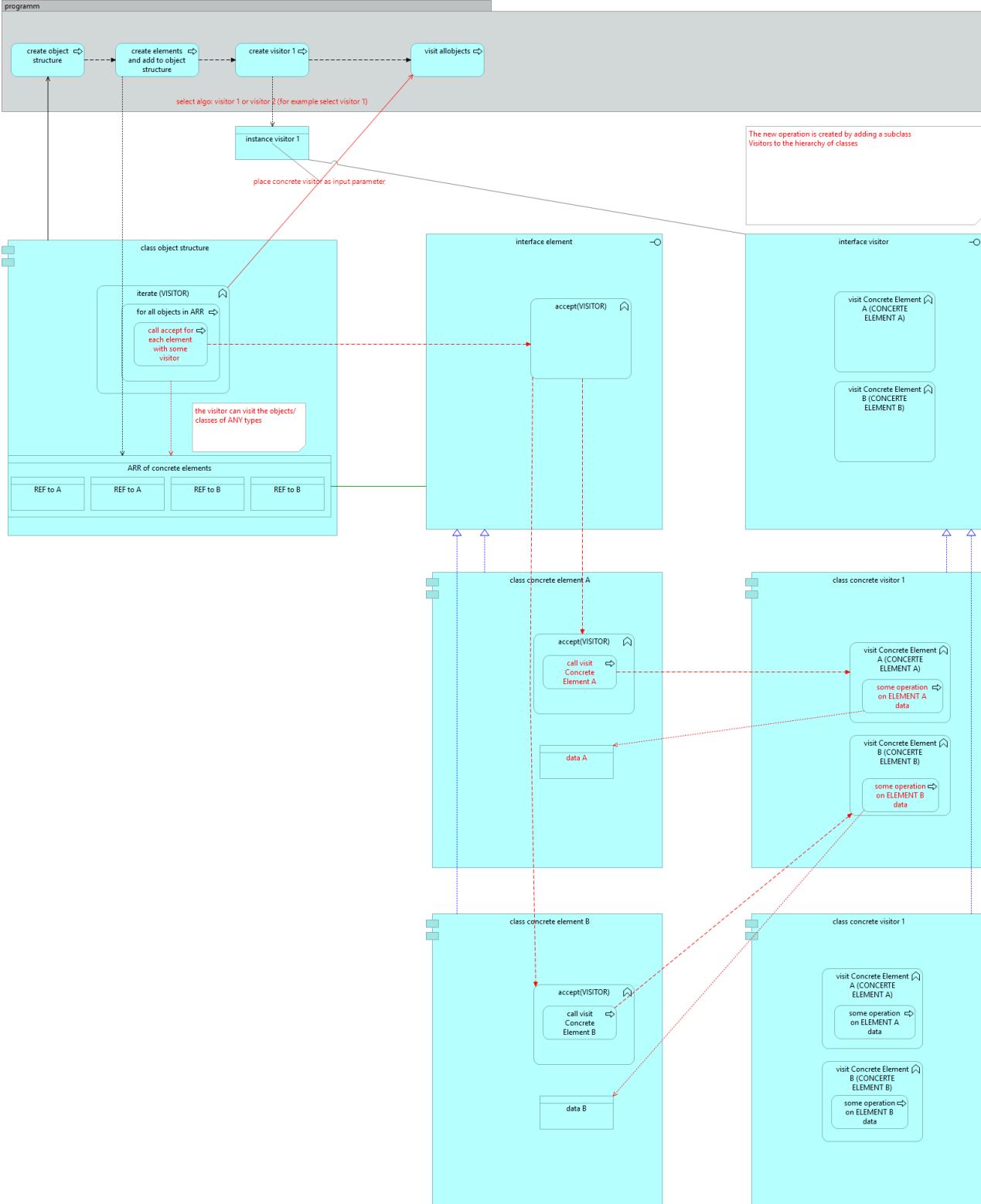
# TEMPLATE METHOD





# VISITOR





02

# Enterprise Patterns

Catalog of Patterns of Enterprise  
Application Architecture

MARTIN FOWLER



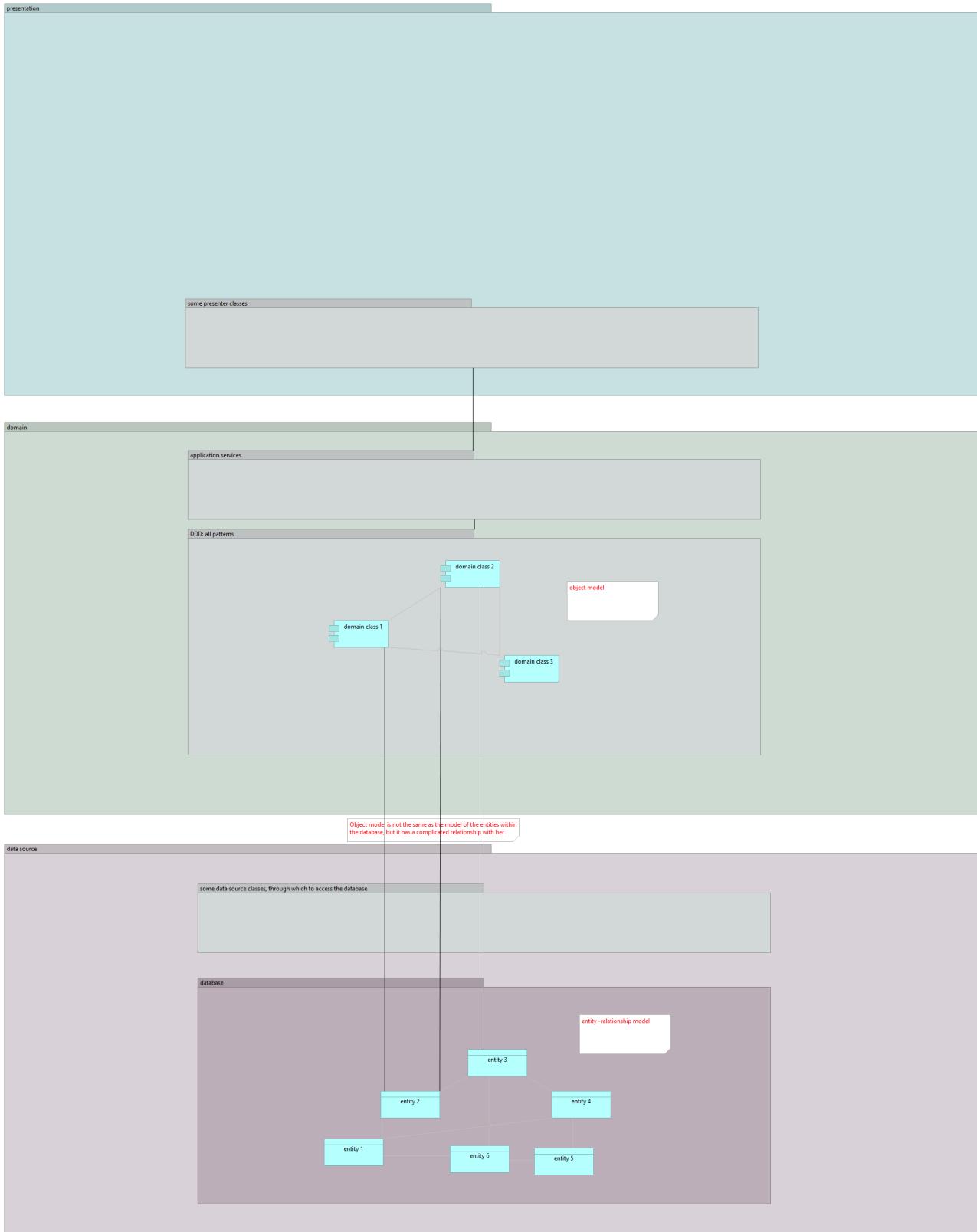
# BUSINESS LOGIC

ENTERPRISE PATTERNS

Martin Fowler



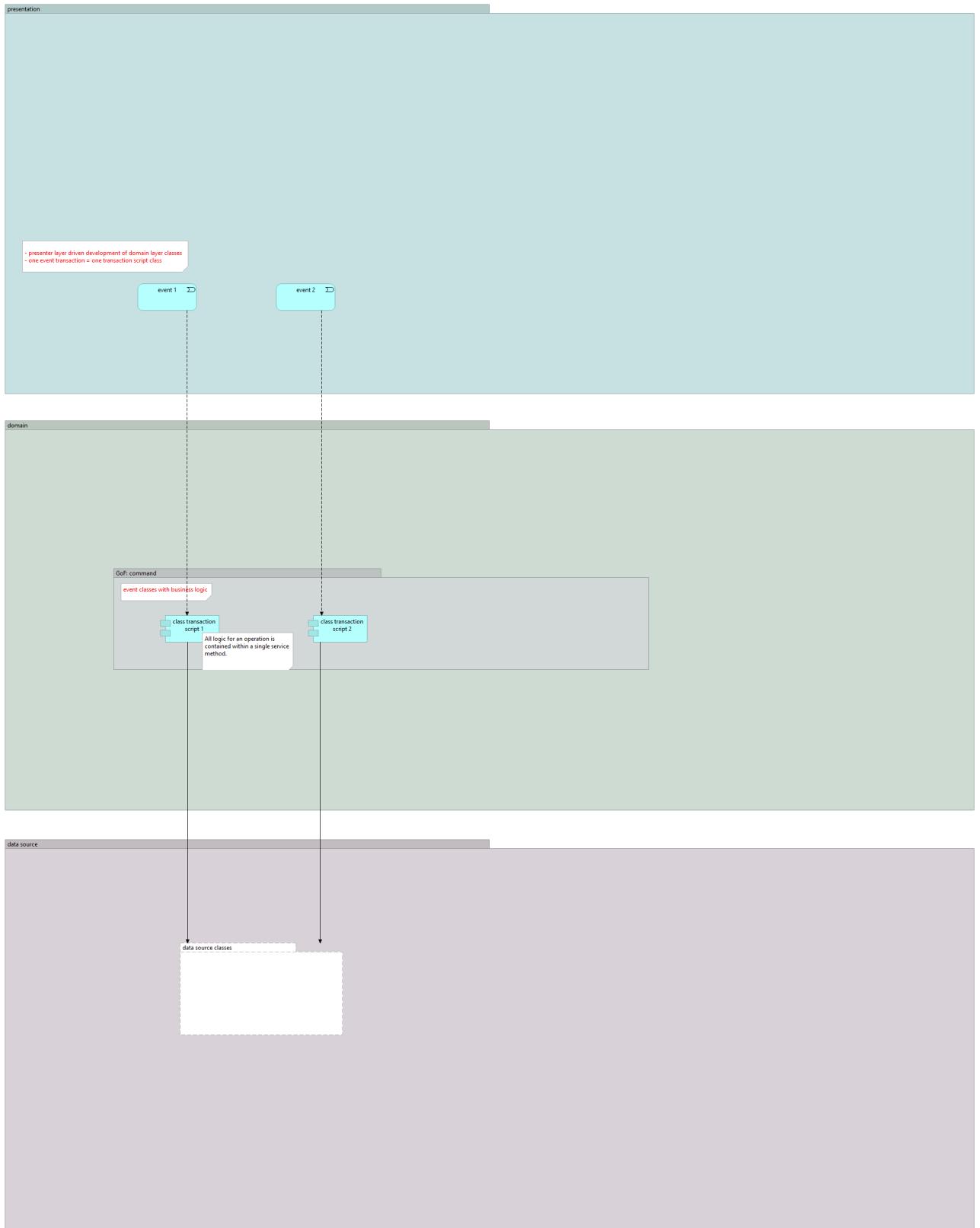
# DOMAIN MODEL



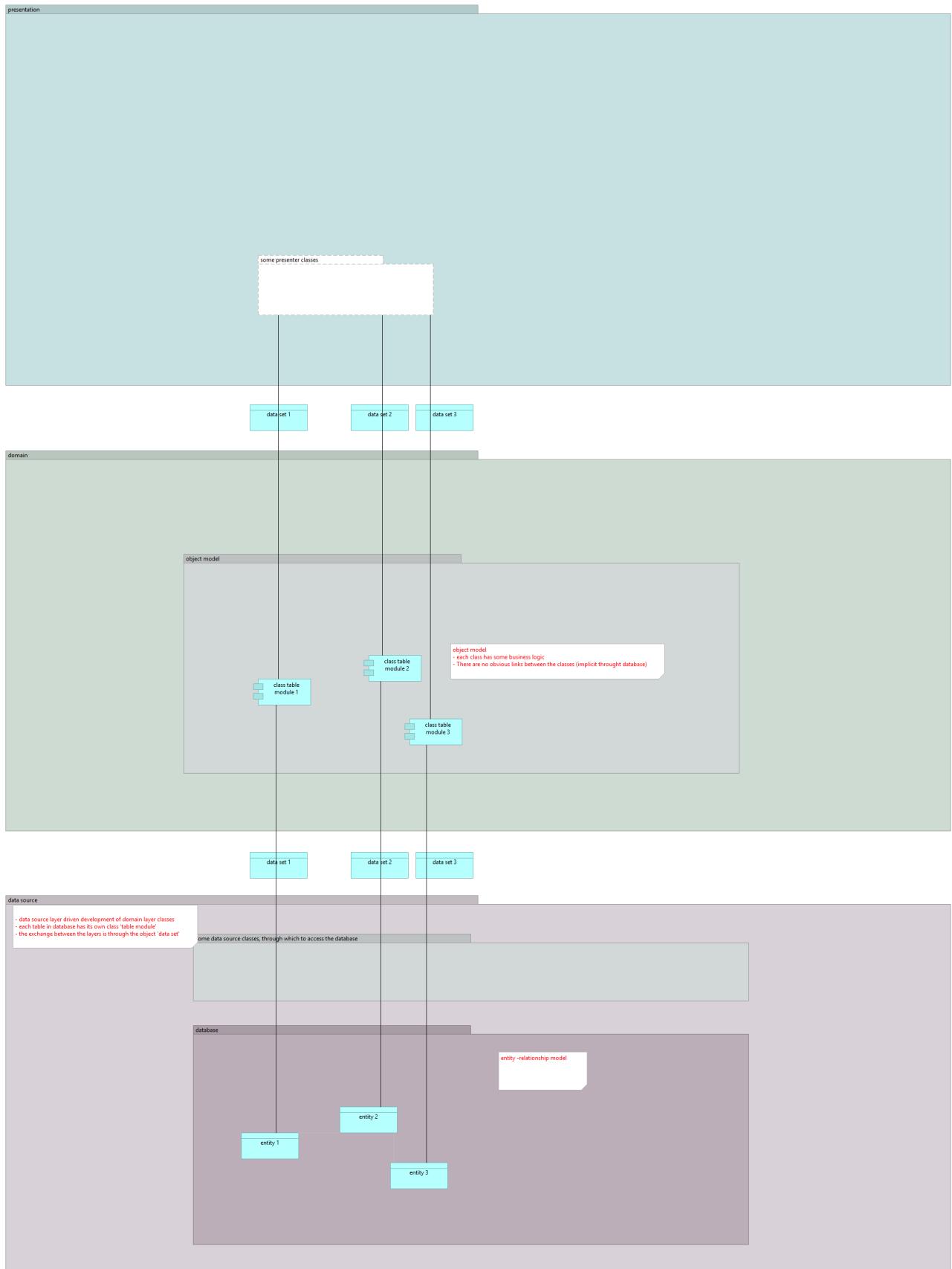
# SERVICE LAYER.



# TRANSACTION SCRIPT



# TABLE MODULE



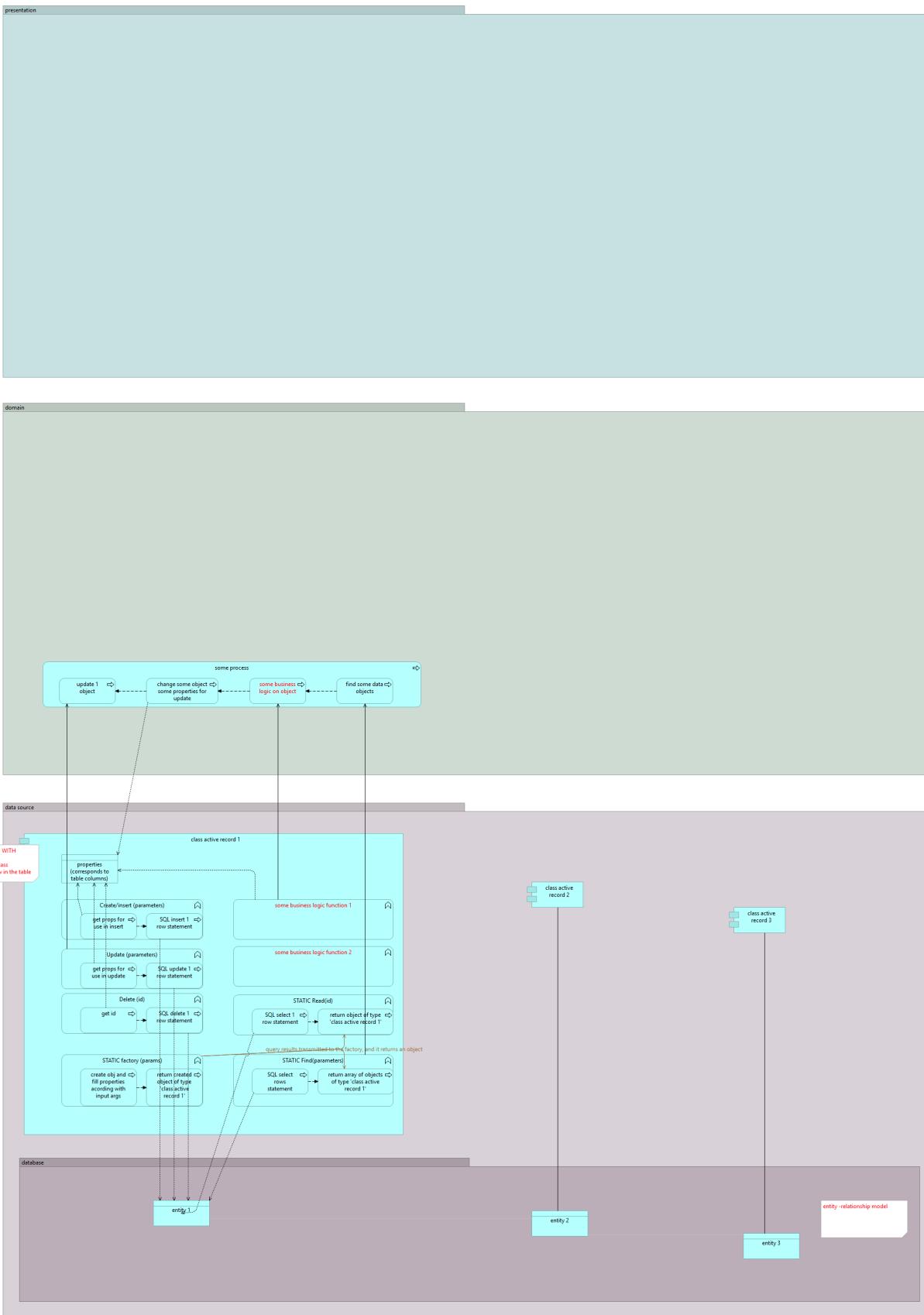
# DATA SOURCES

ENTERPRISE PATTERNS

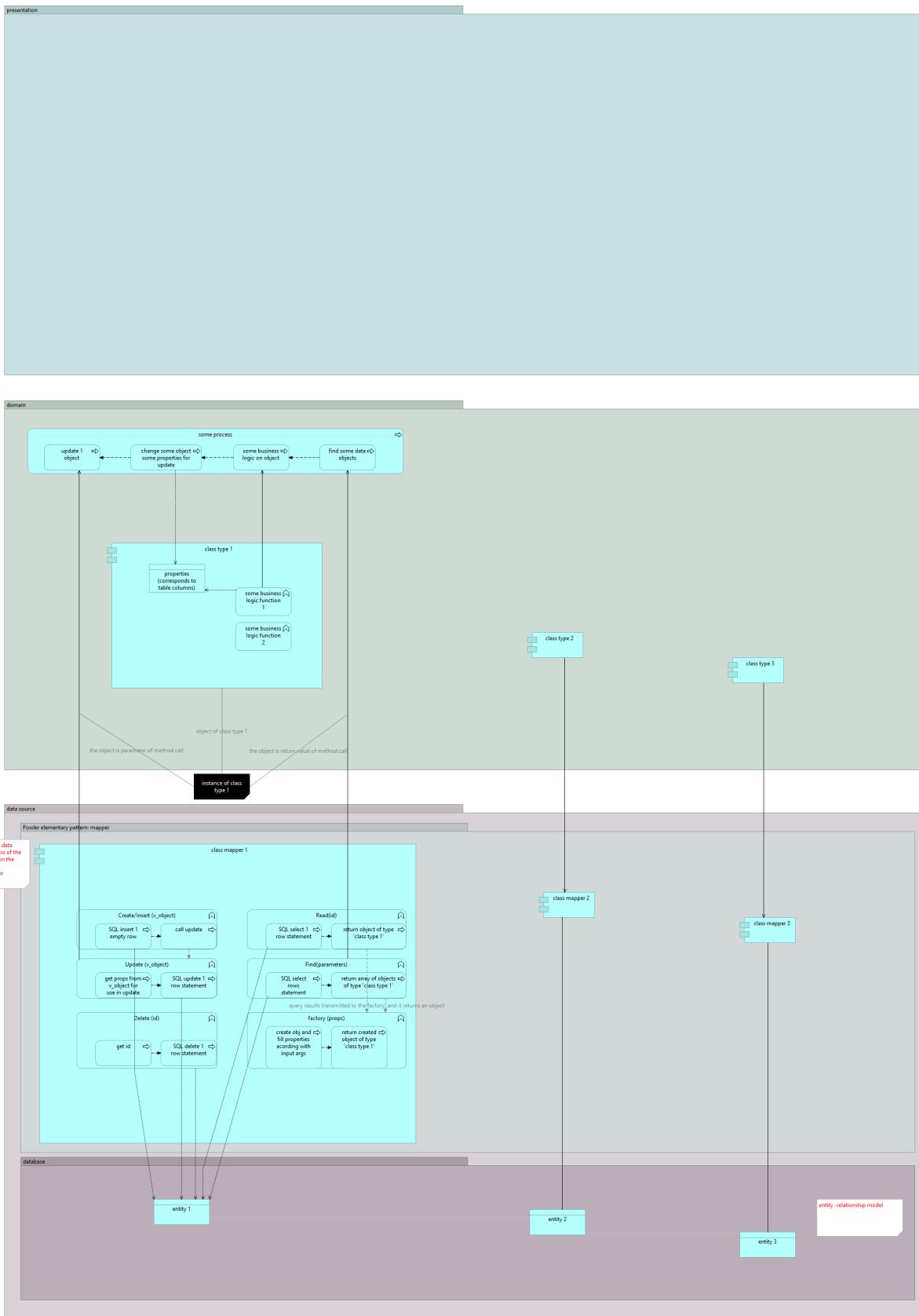
Martin Fowler



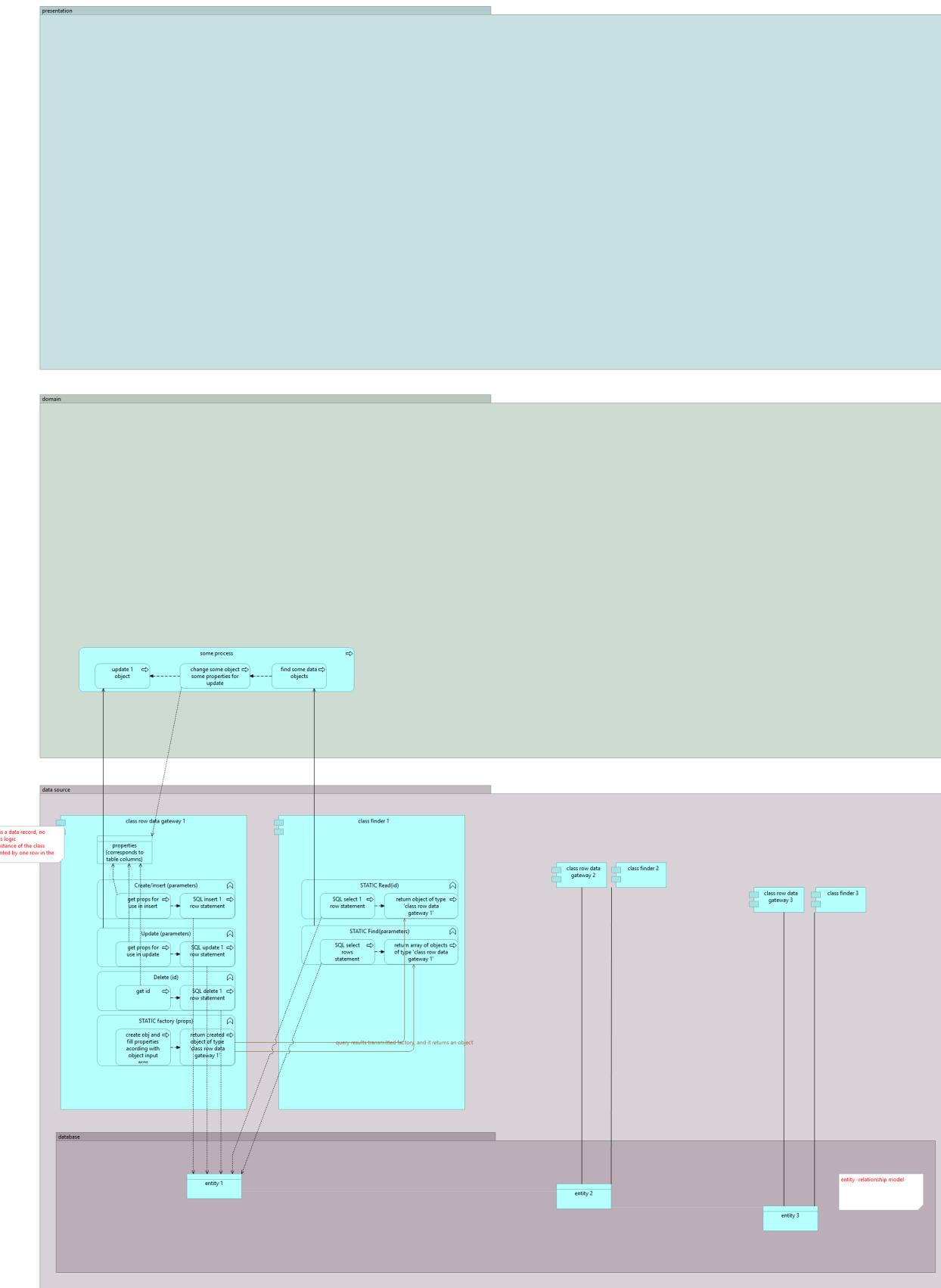
# ACTIVE RECORD



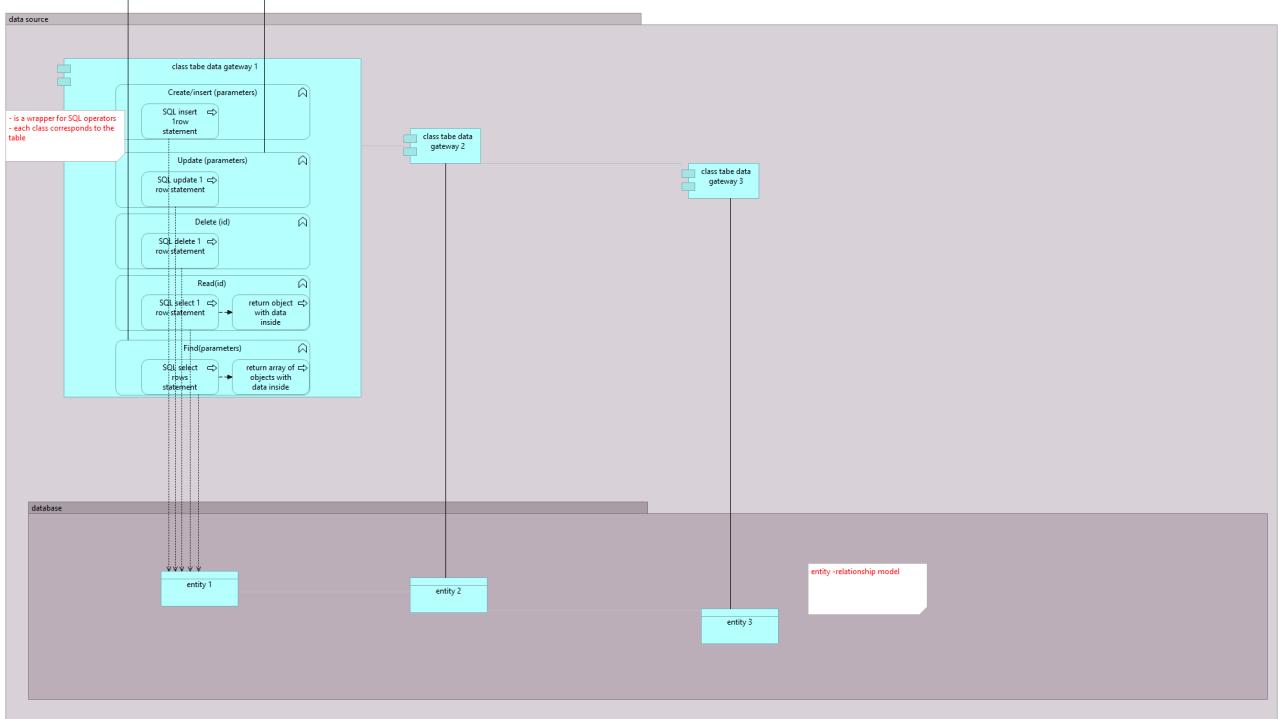
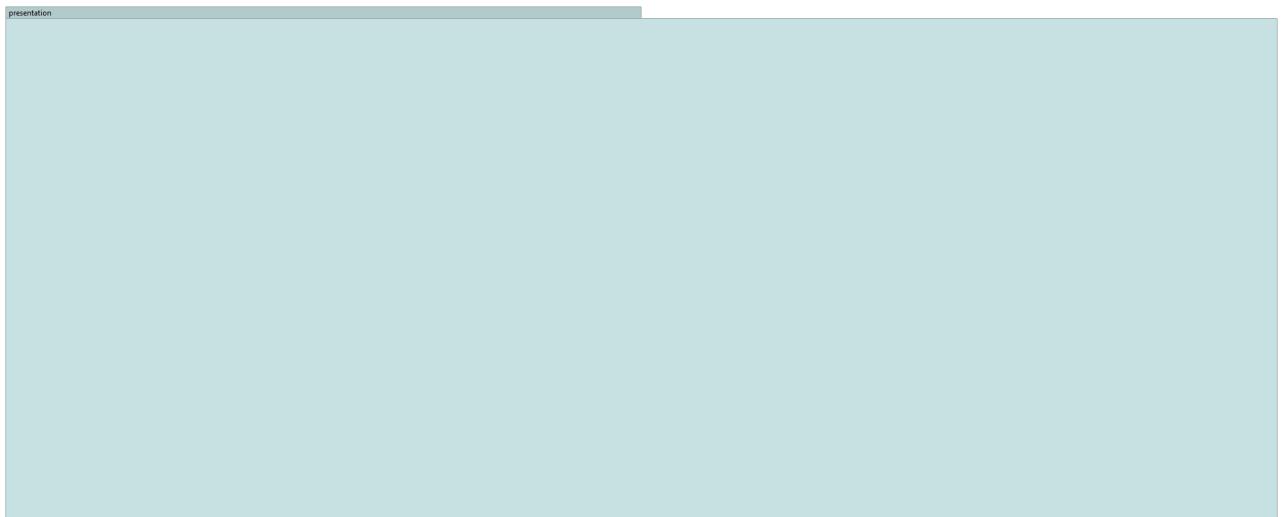
# DATA MAPPER



# ROW DATA GATEWAY



# TABLE DATA GATEWAY



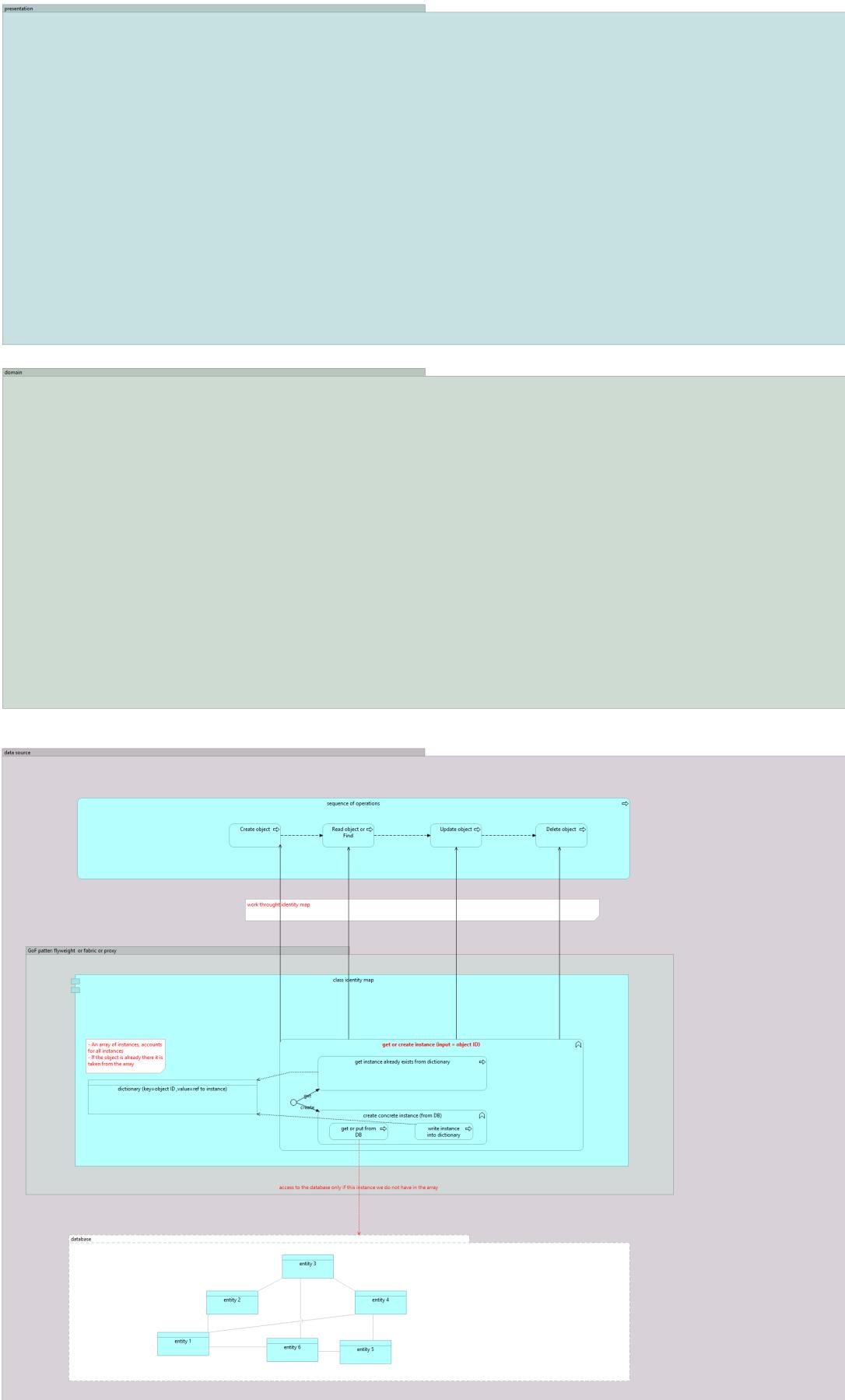
# MODELING BEHAVIOR

ENTERPRISE PATTERNS

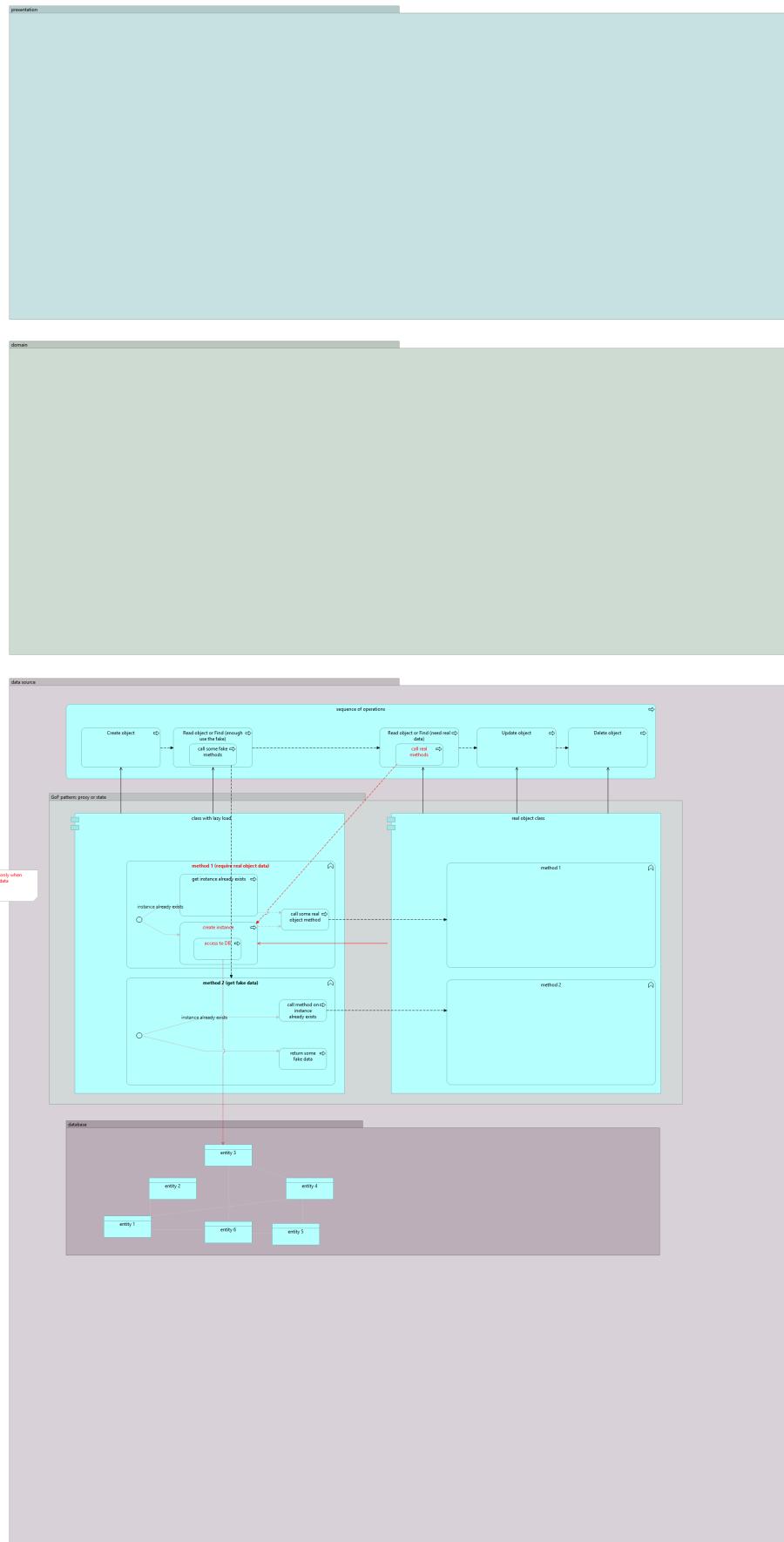
Martin Fowler



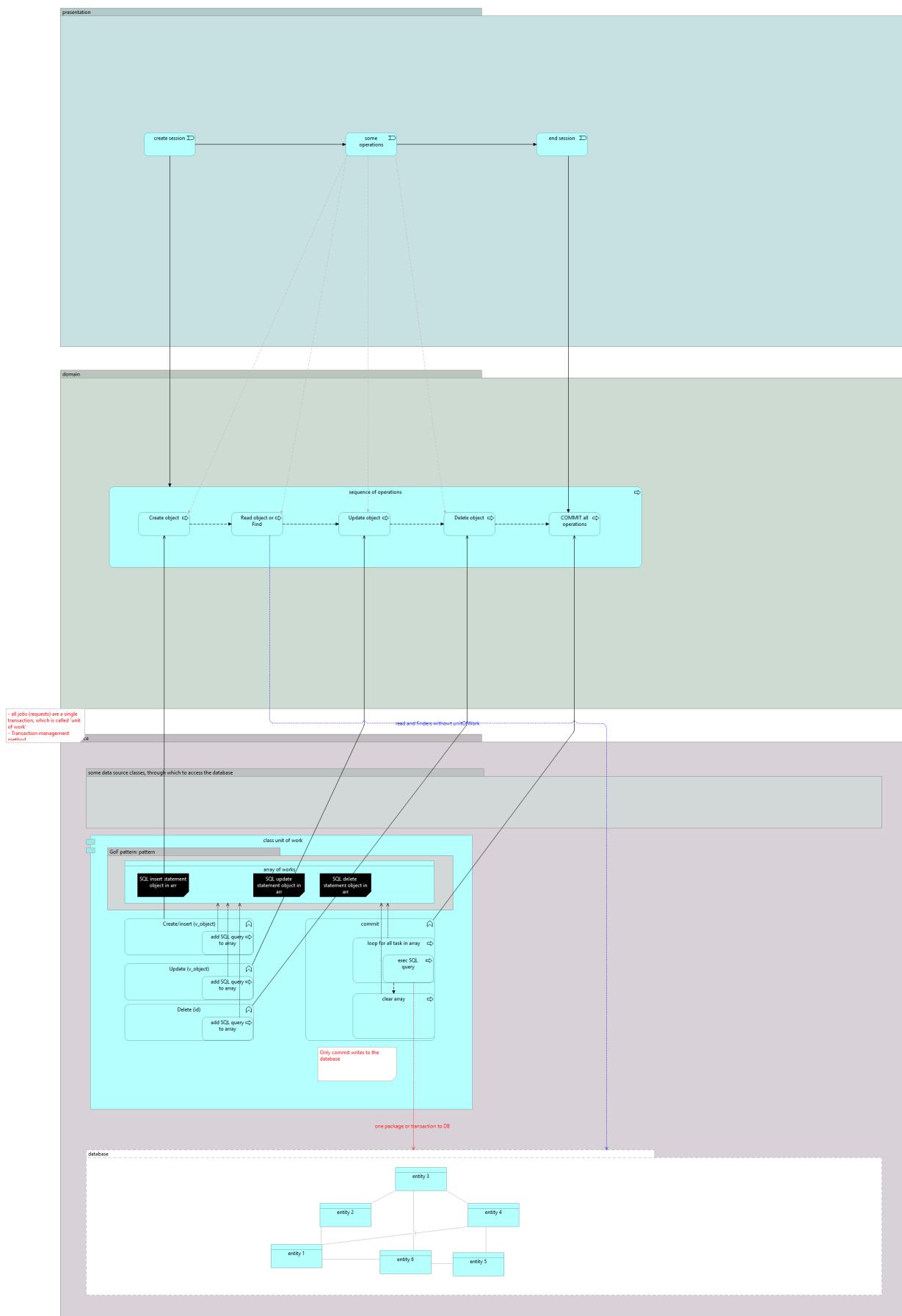
# IDENTITY MAP



# LAZY LOAD



# UNIT OF WORK.



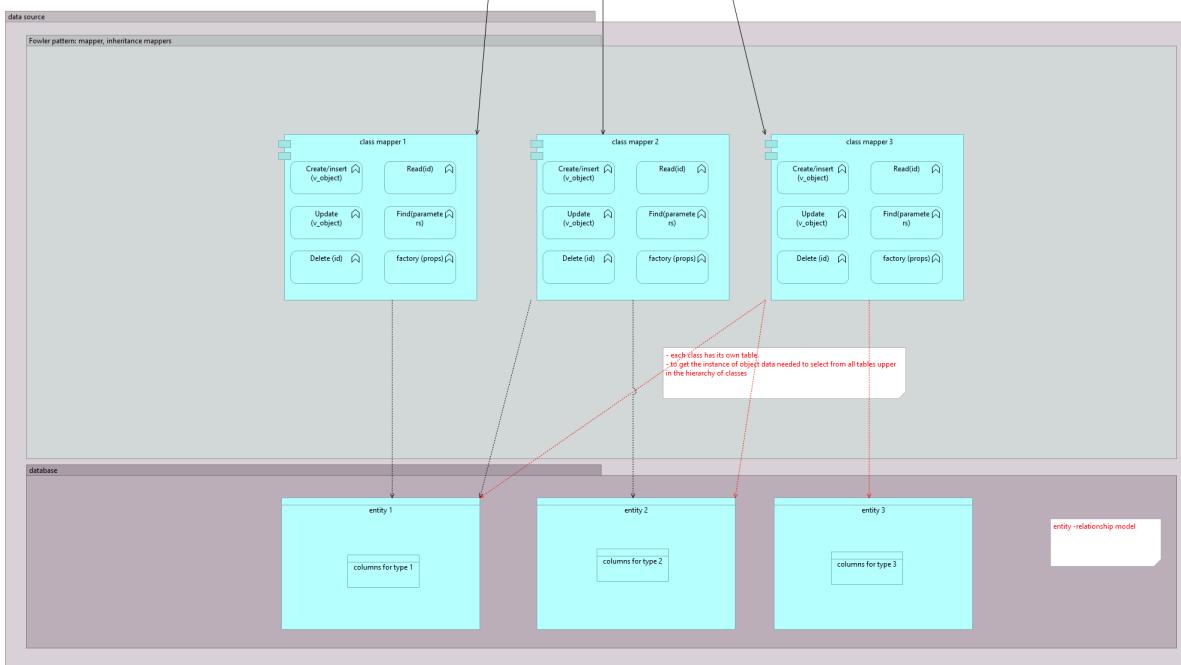
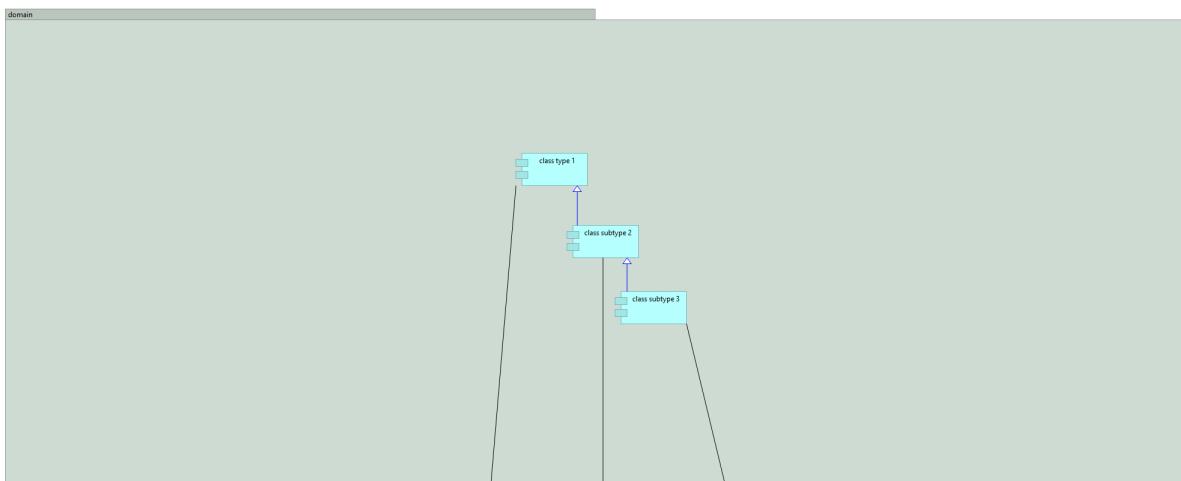
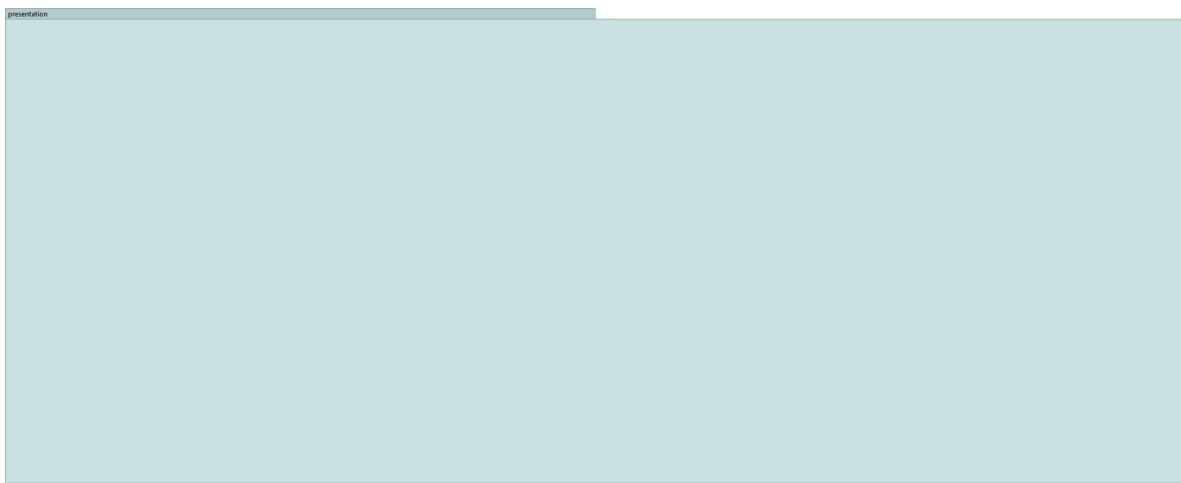
# MODELING STRUCTURE HIERARCHY

ENTERPRISE PATTERNS

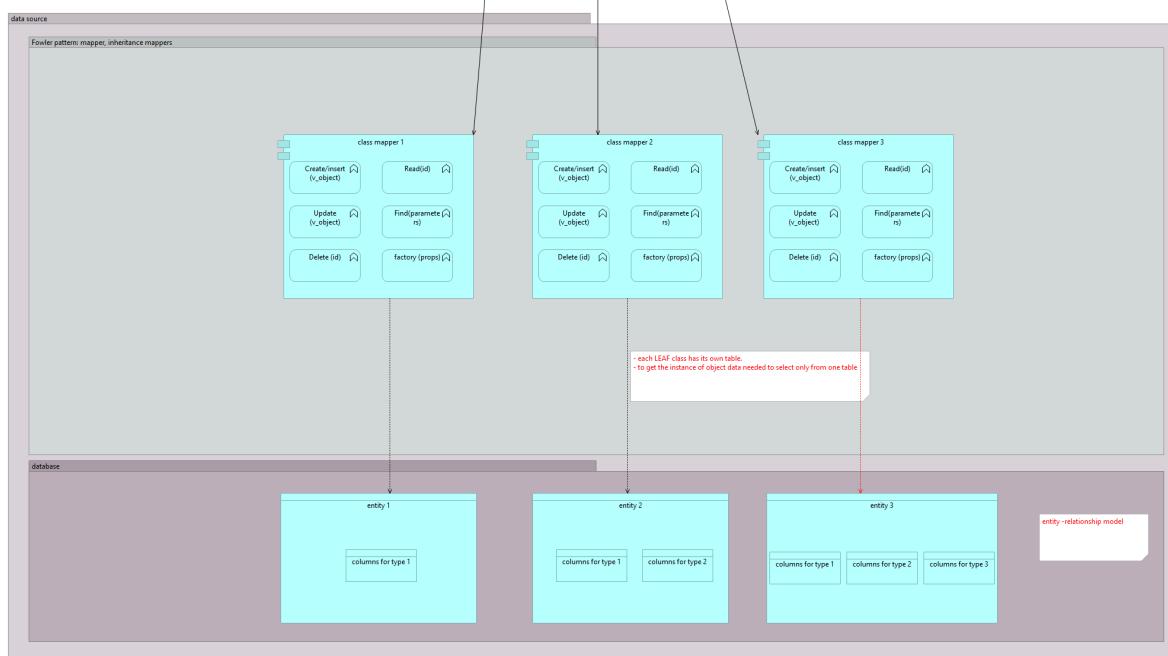
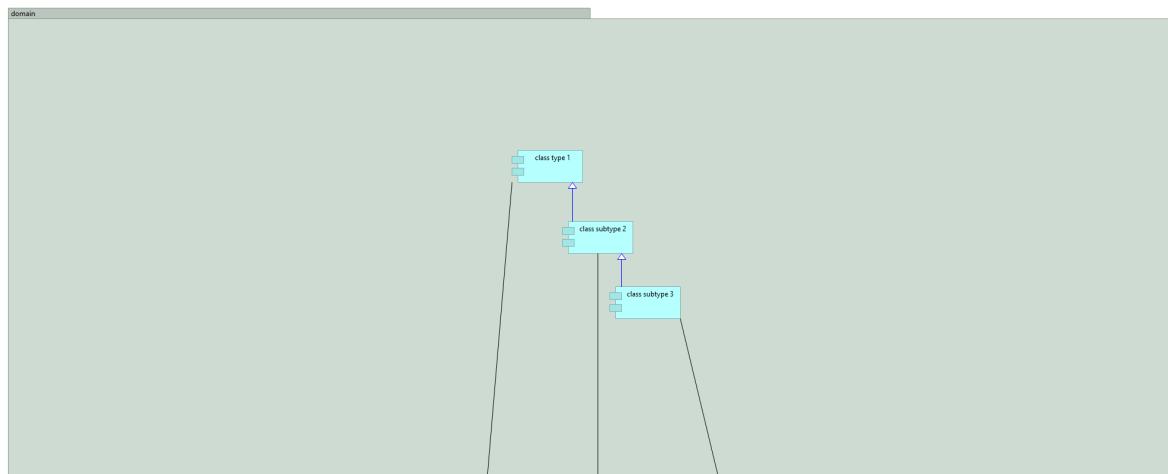
Martin Fowler



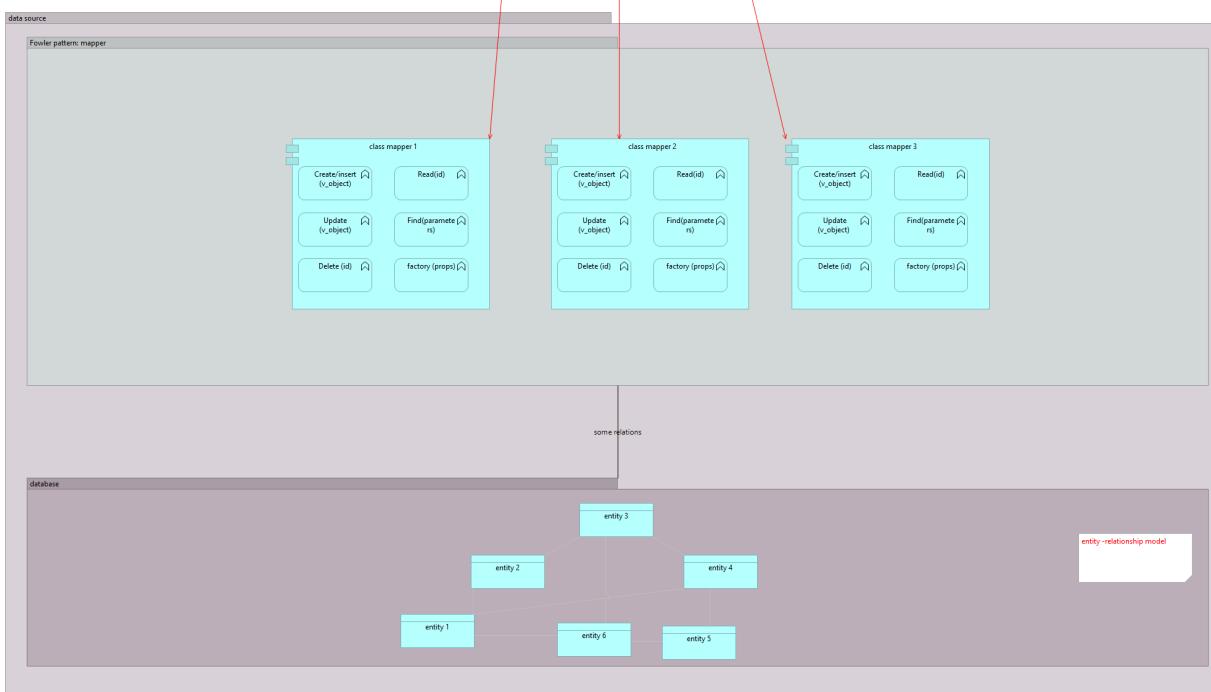
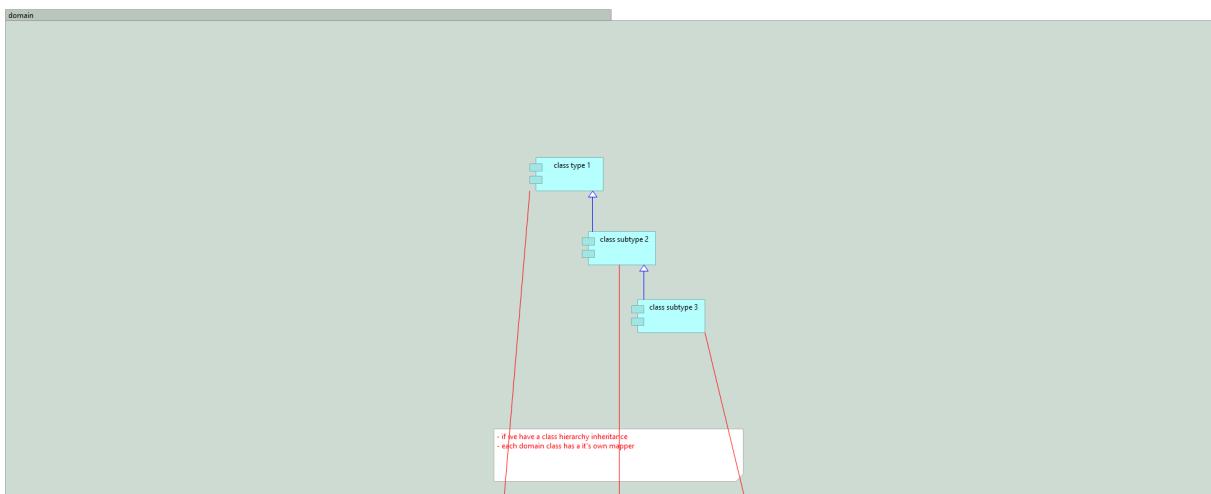
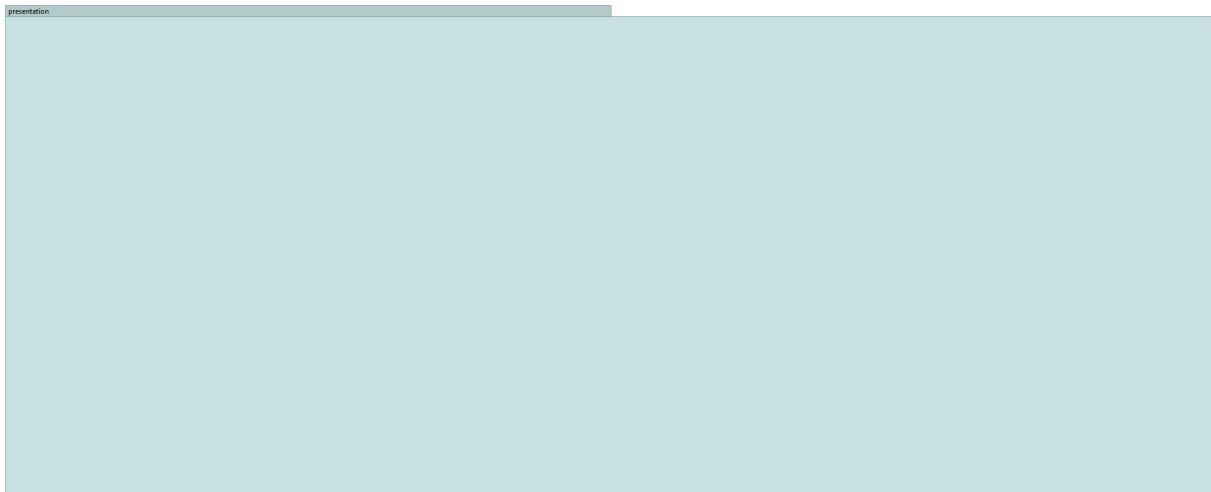
# CLASS TABLE INHERITANCE



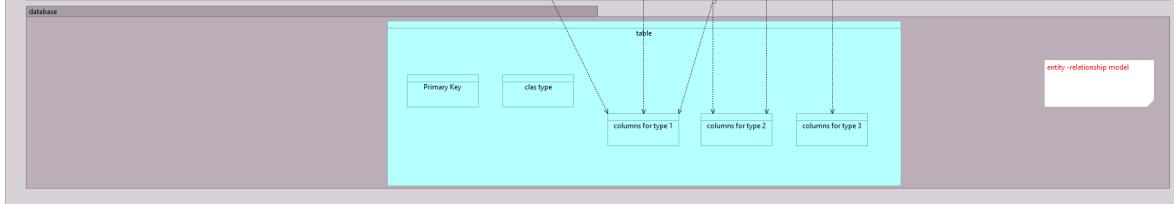
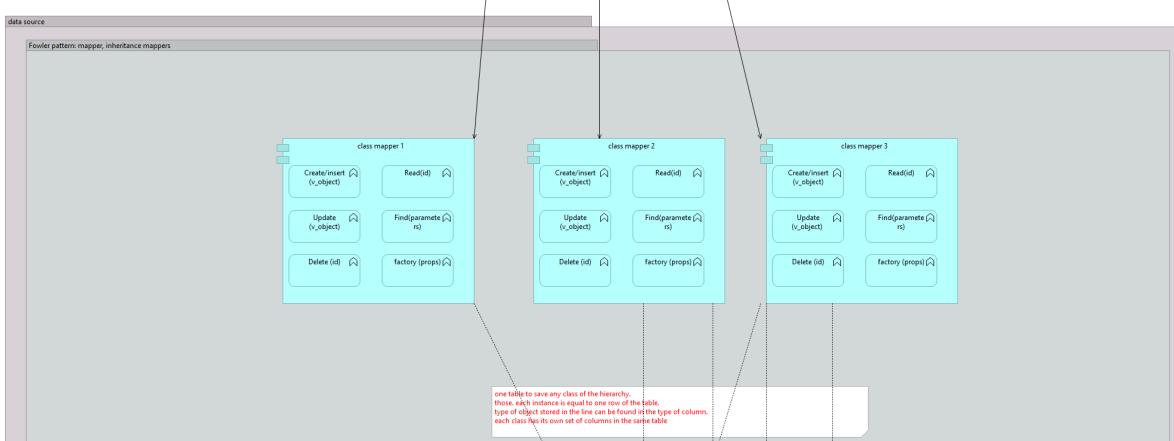
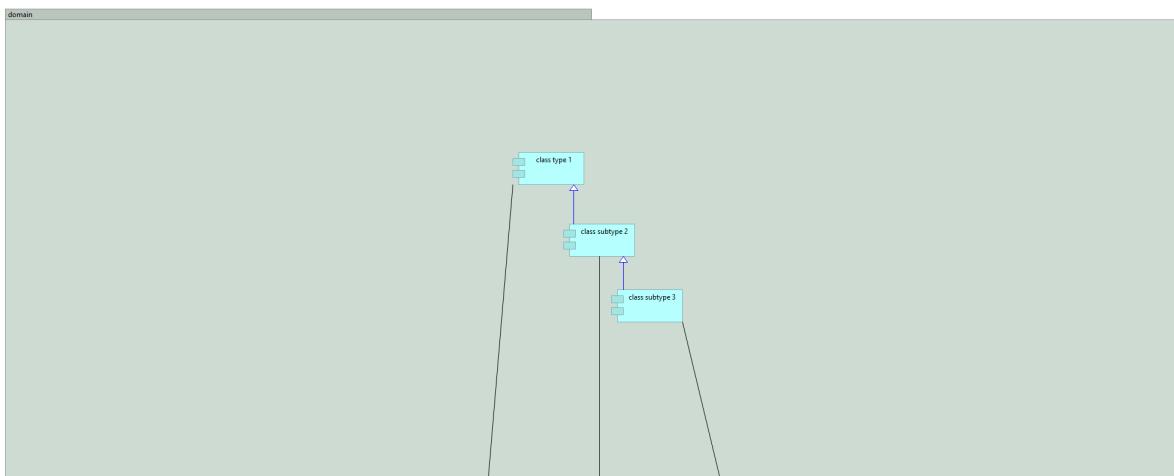
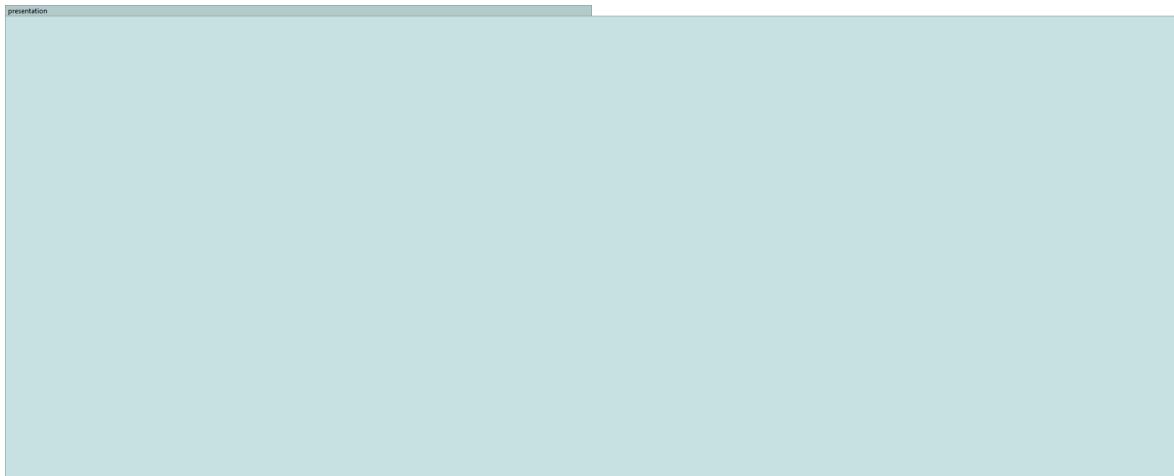
# CONCRETE TABLE INHERITANCE



# INHERITANCE MAPPERS



# SINGLE TABLE INHERITANCE



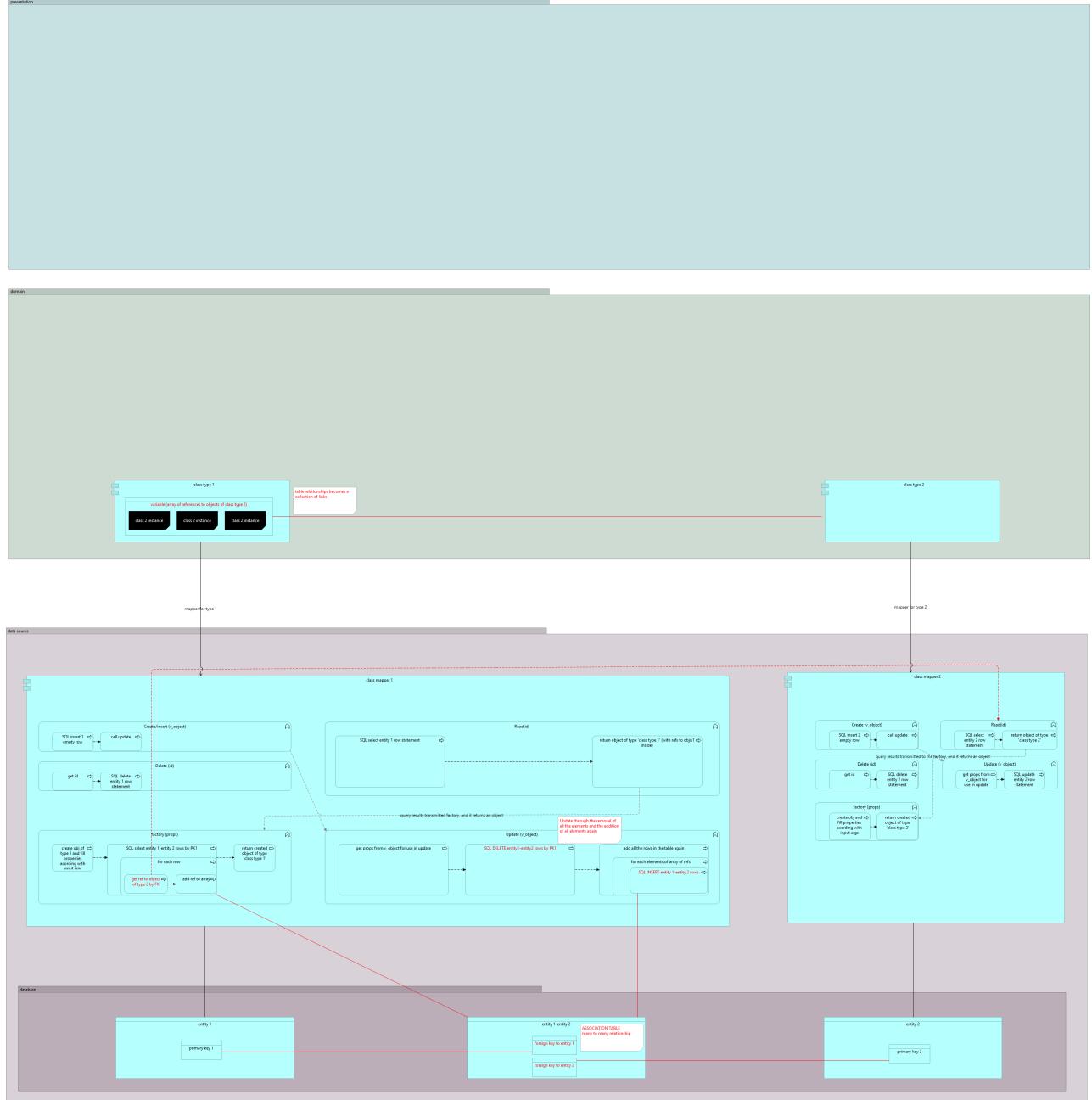
# MODELING STRUCTURE RELATIONS

ENTERPRISE PATTERNS

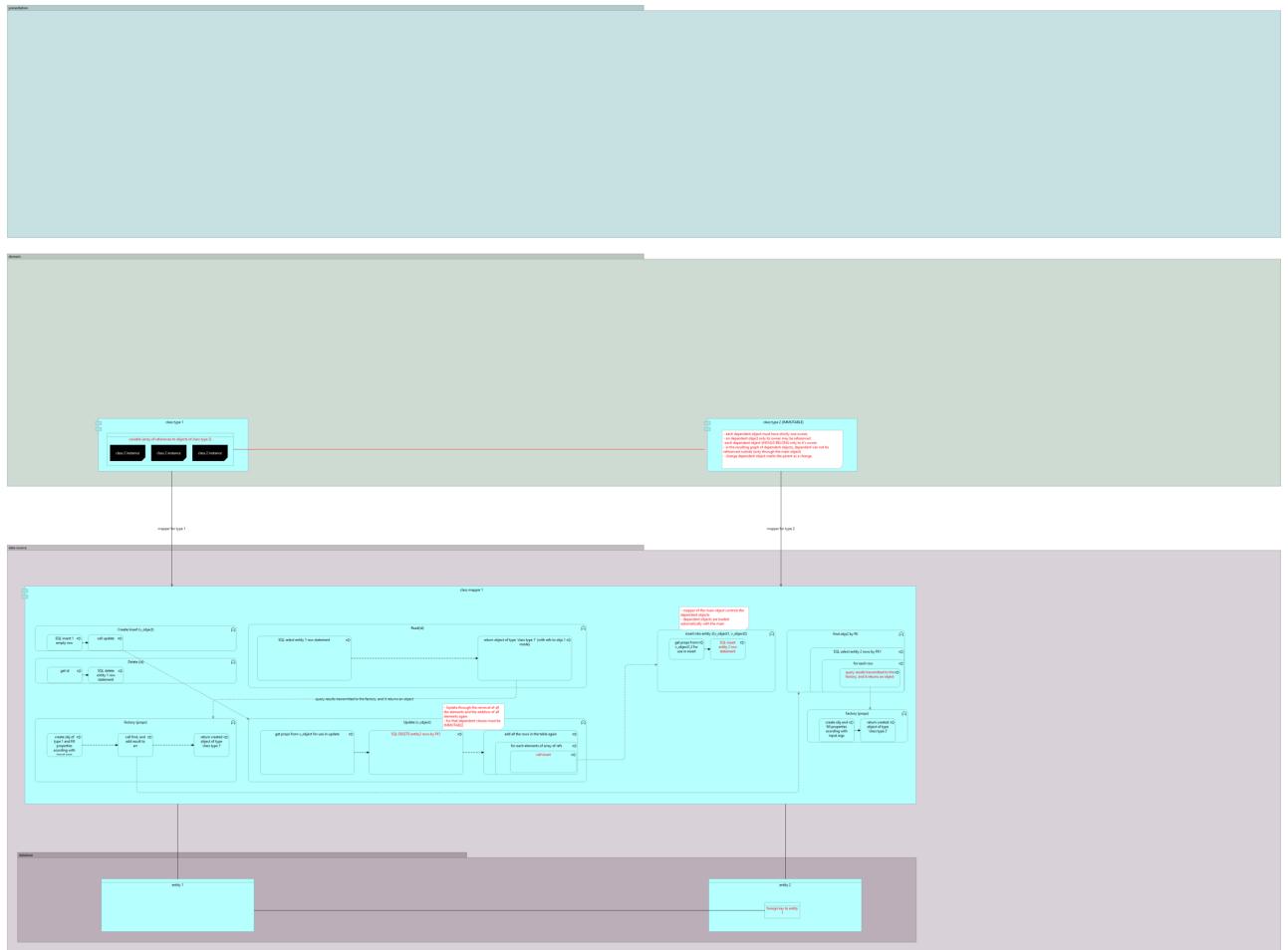
Martin Fowler



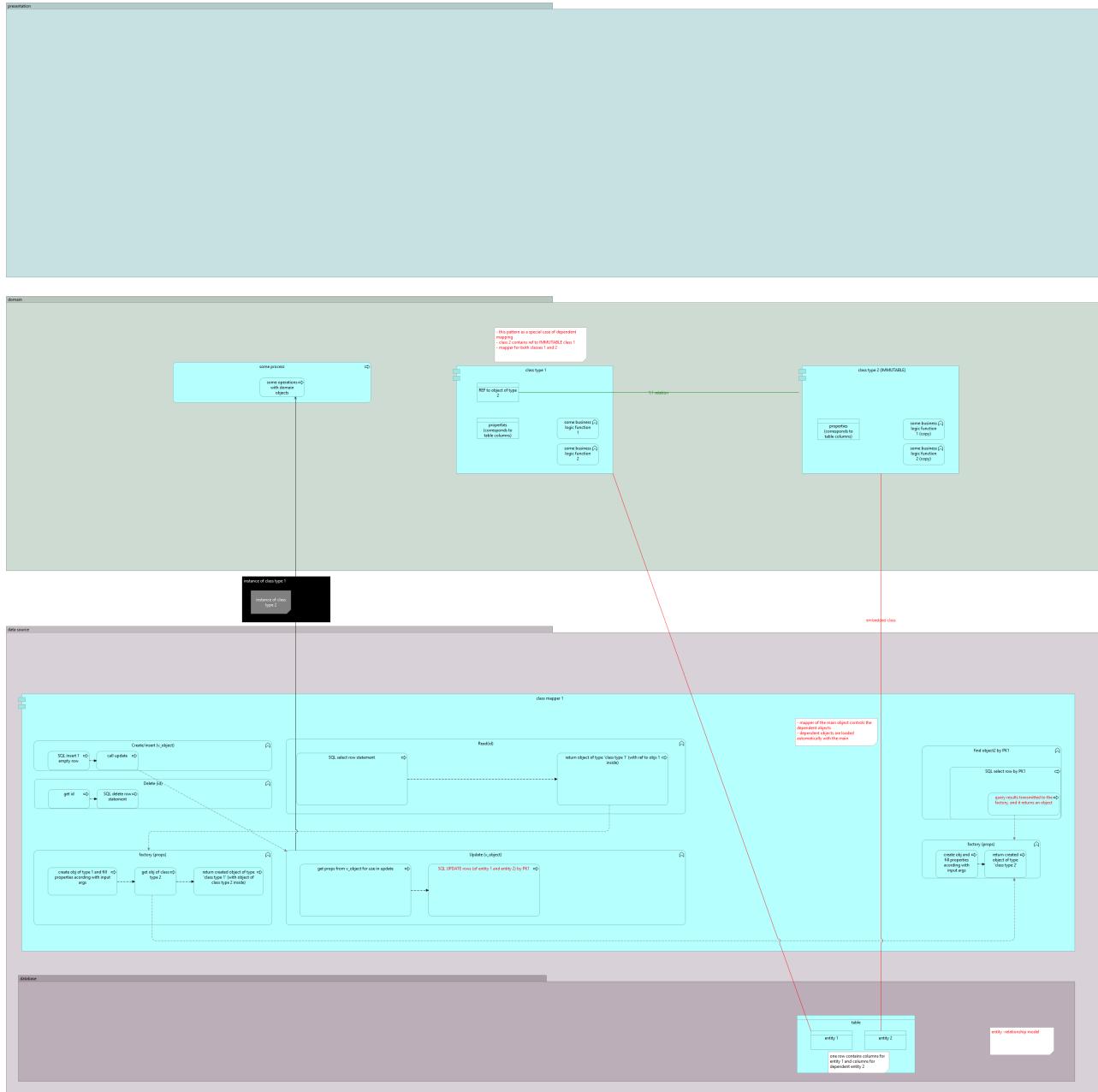
# ASSOCIATION TABLE MAPPING



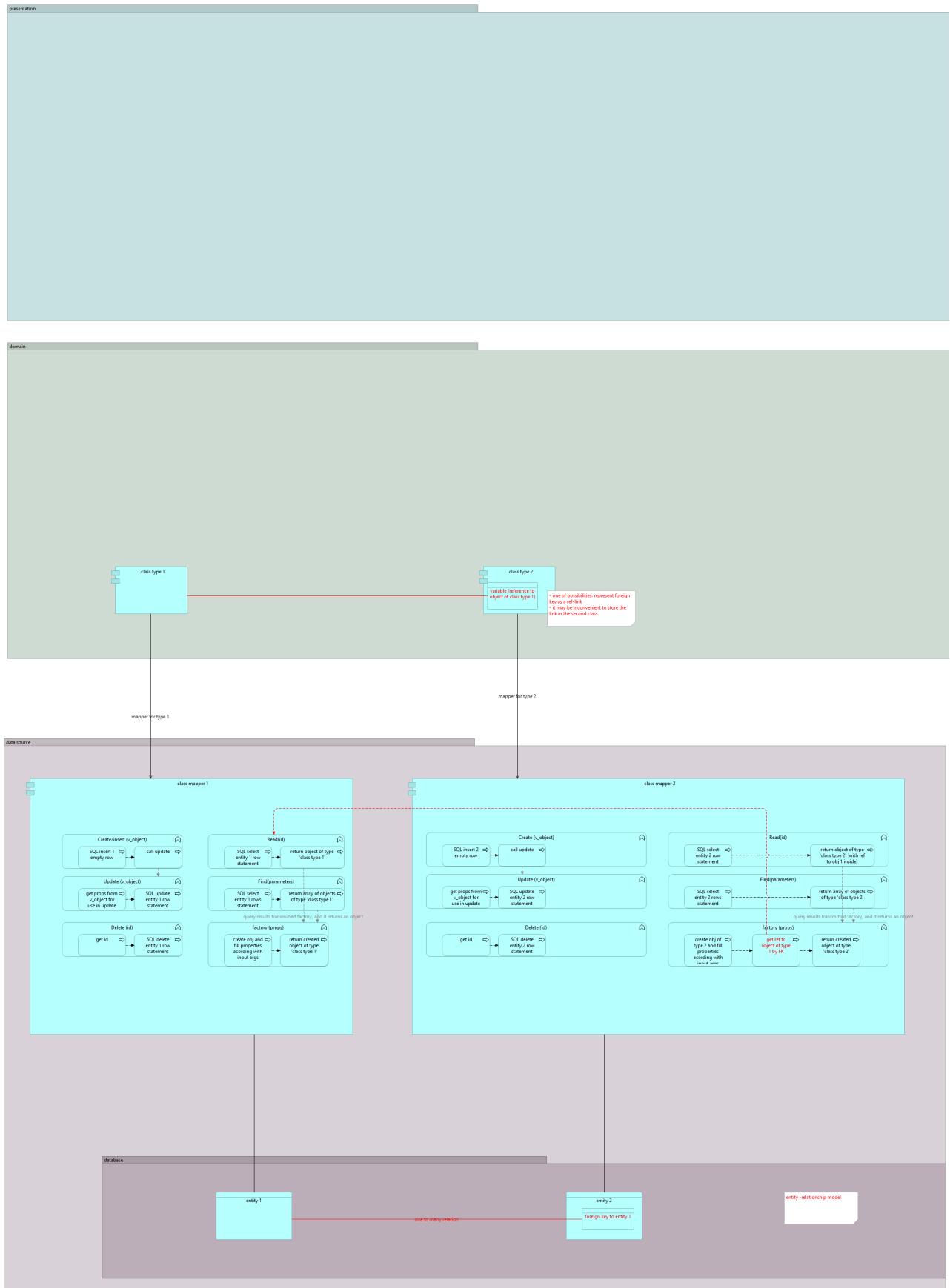
# DEPENDENT MAPPING



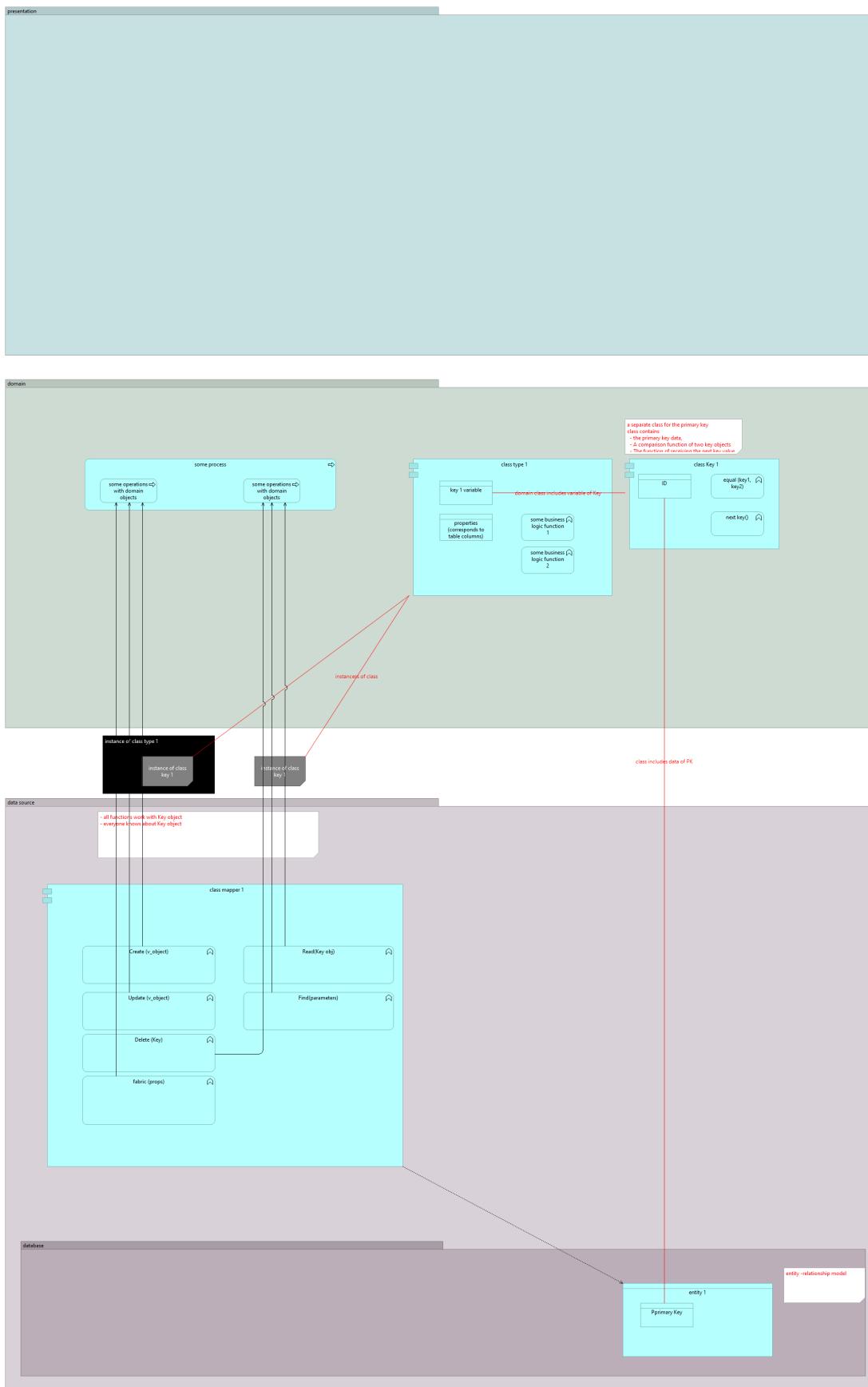
# EMBEDDED VALUE



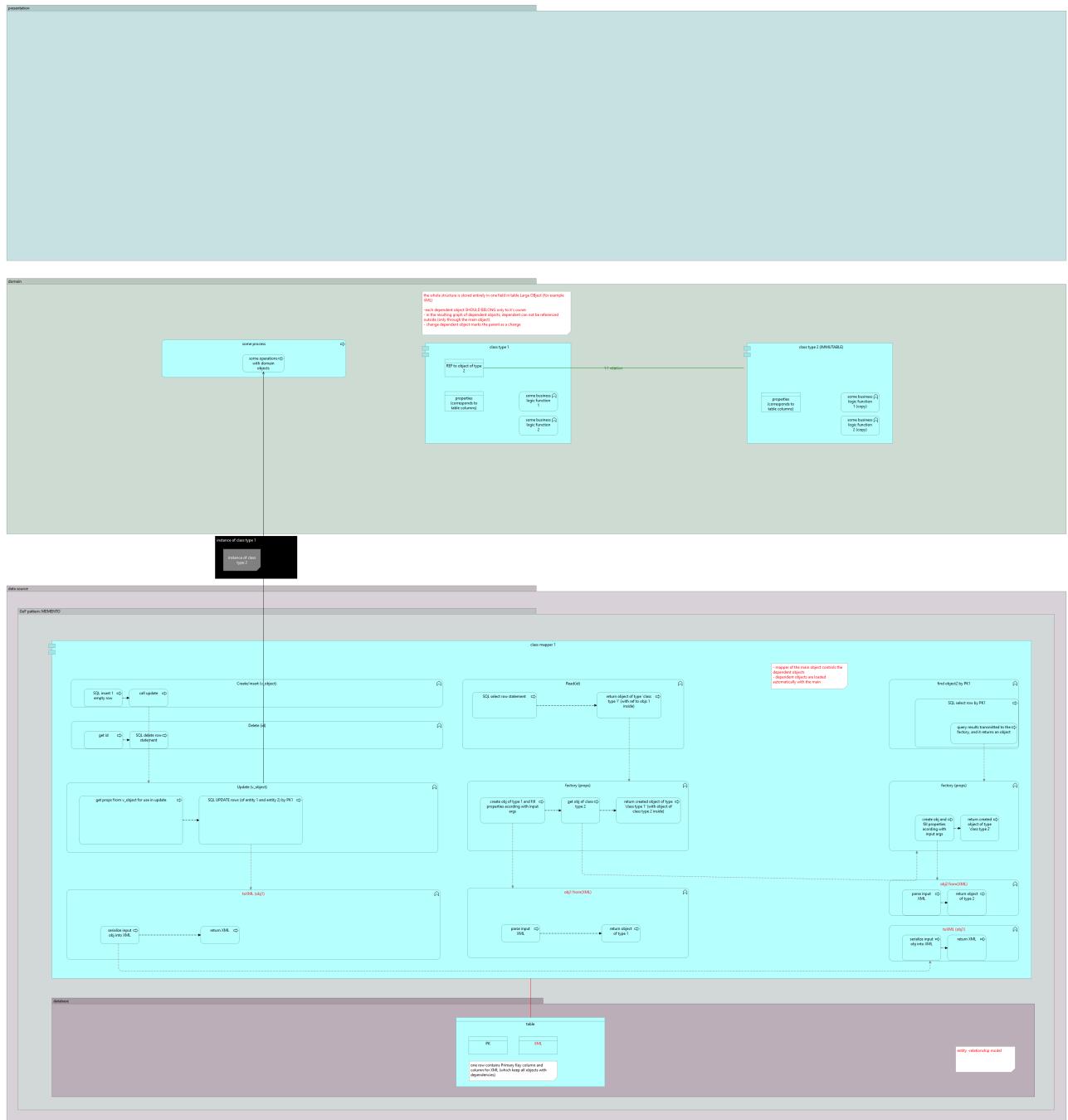
# FOREIGN KEY MAPPING



# IDENTITY FIELD



# SERIALIZED LOB



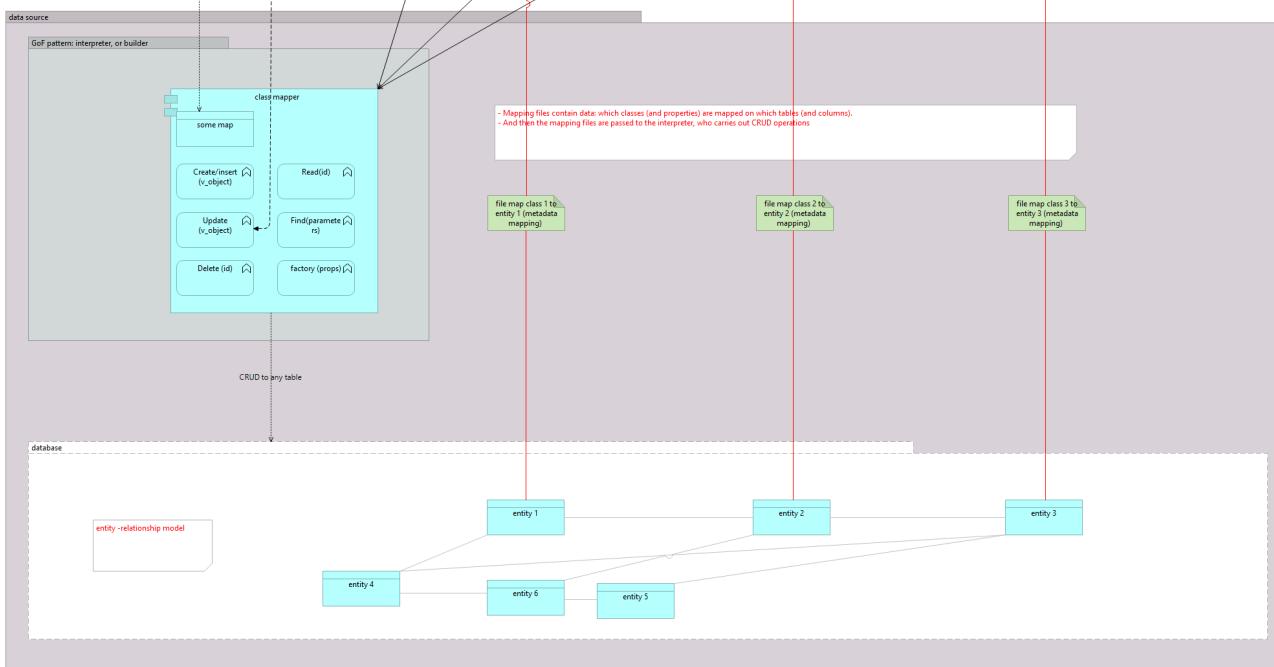
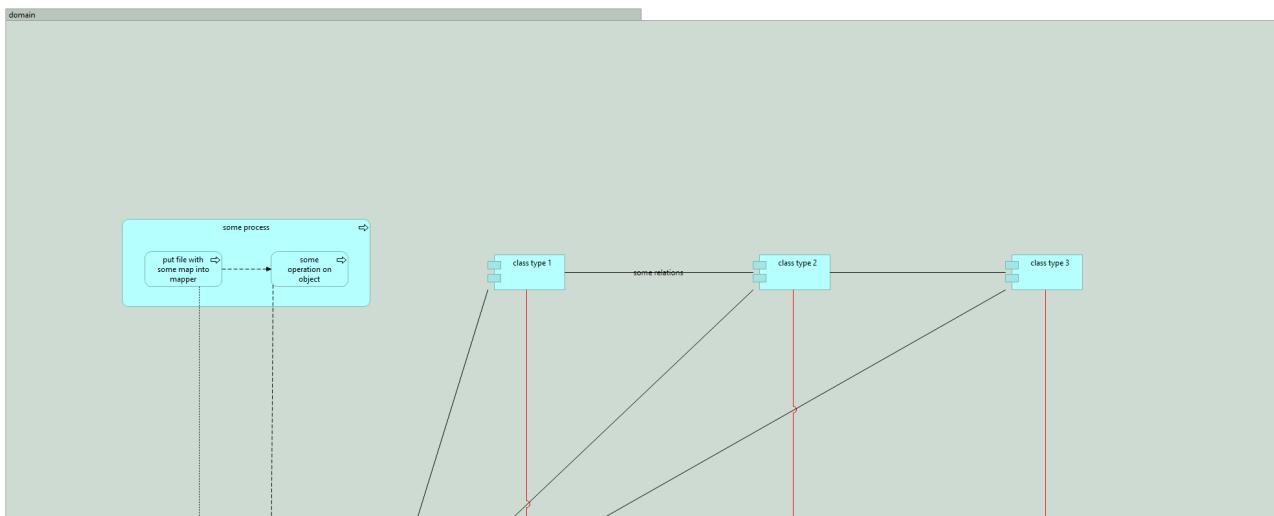
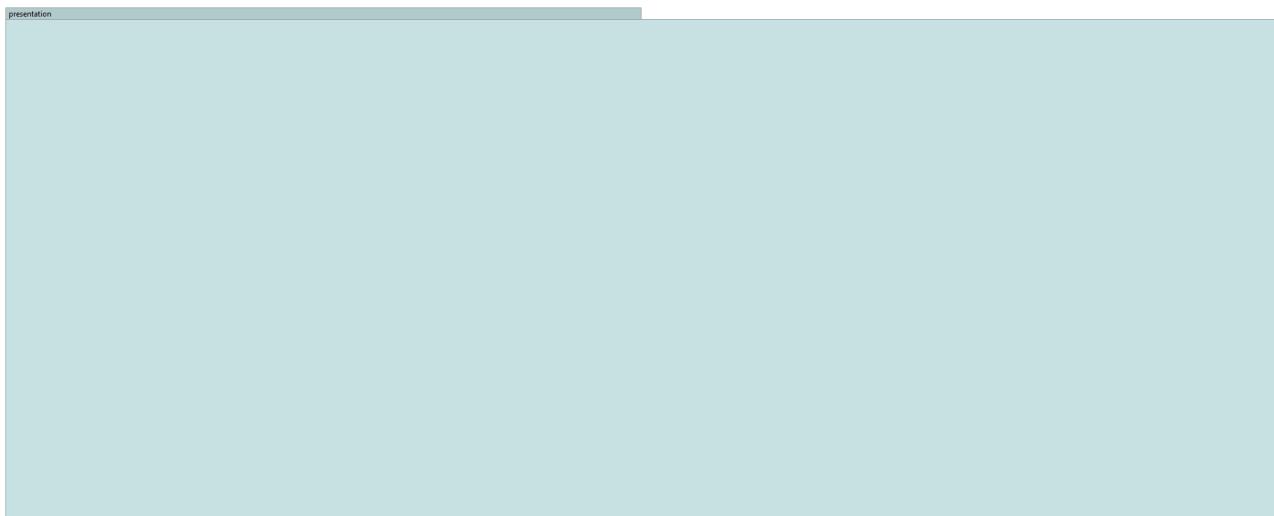
# METADATA

ENTERPRISE PATTERNS

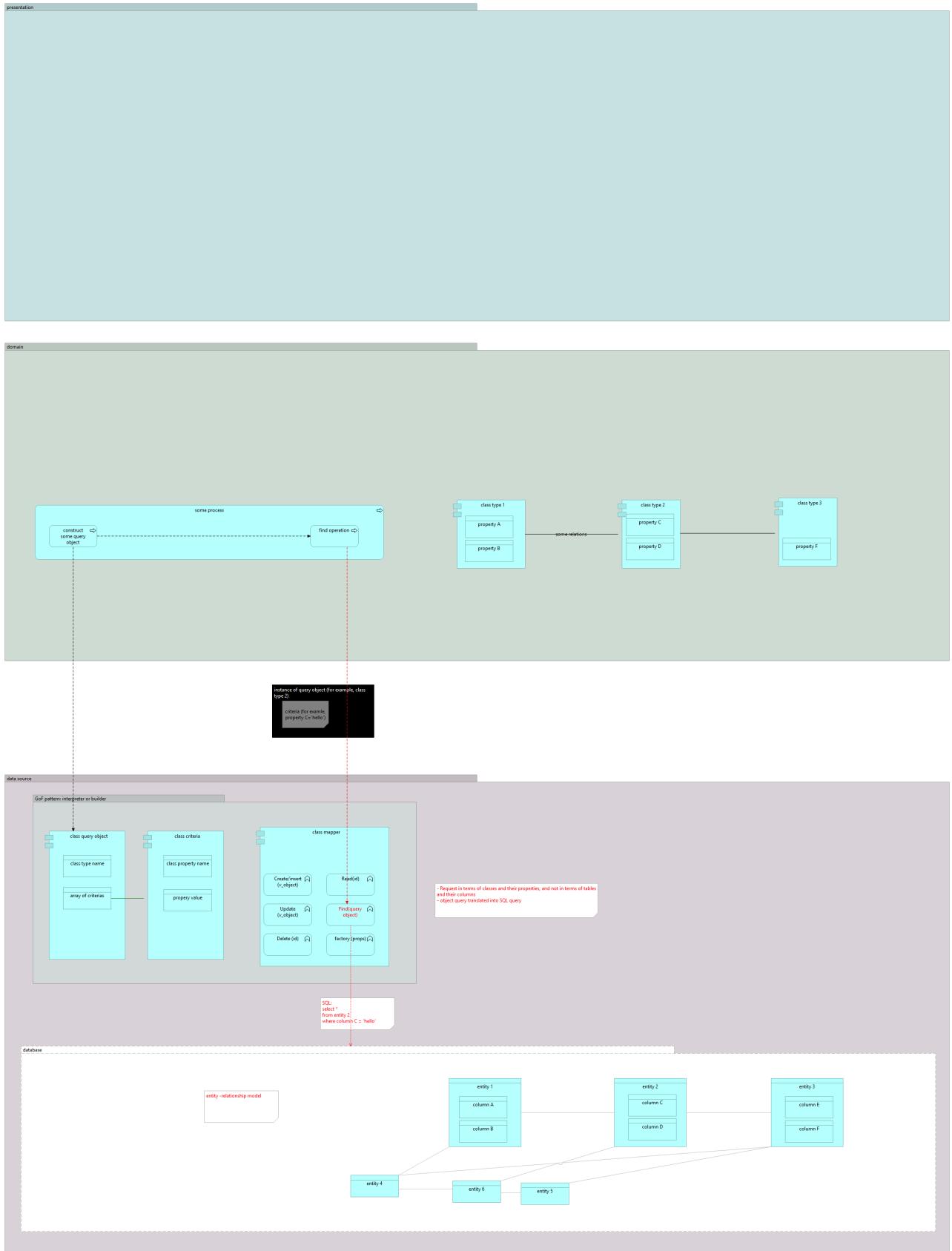
Martin Fowler



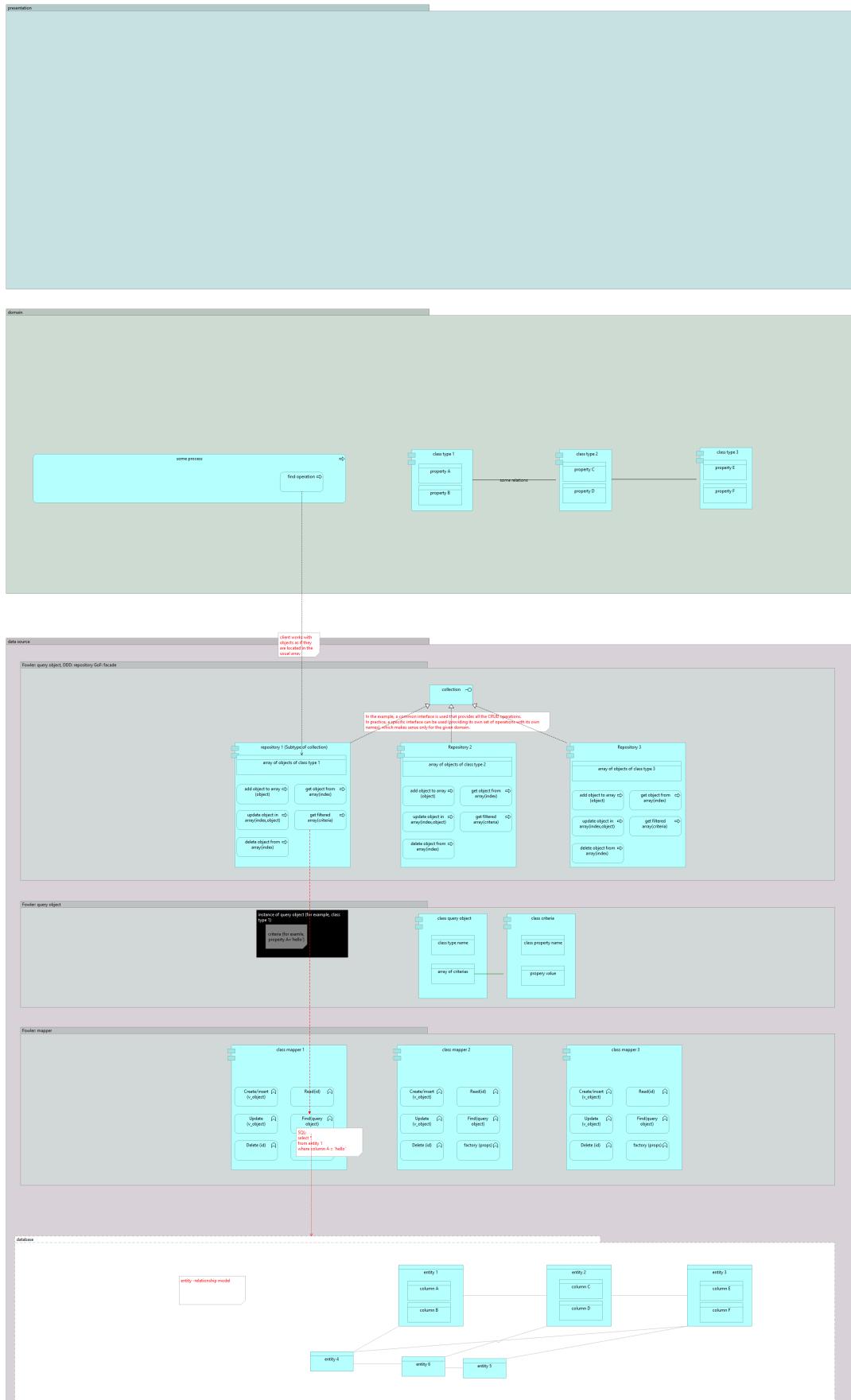
# METADATA MAPPING



# QUERY OBJECT



# REPOSITORY



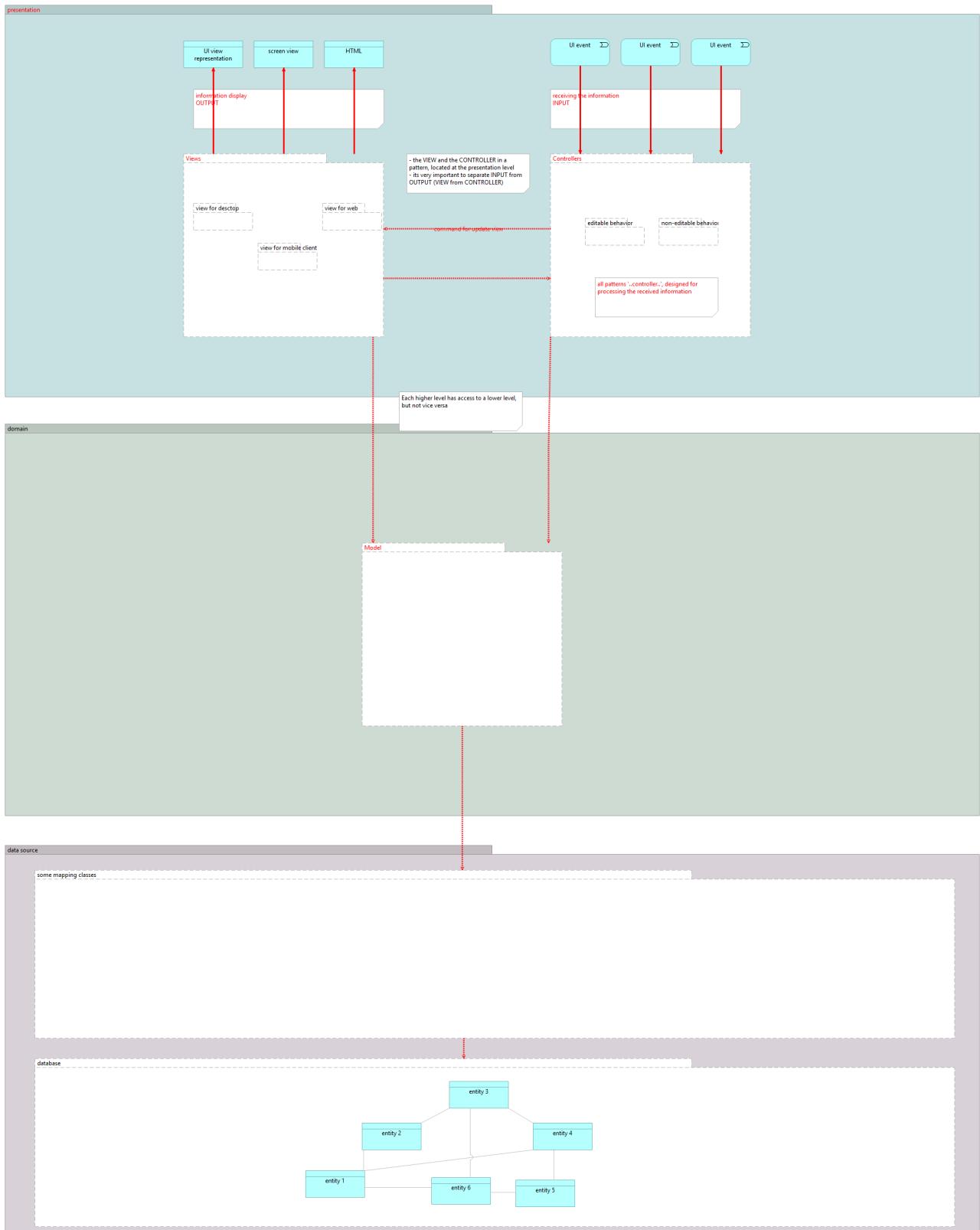
# WEB REPRESENTATION CONTROLLER

ENTERPRISE PATTERNS

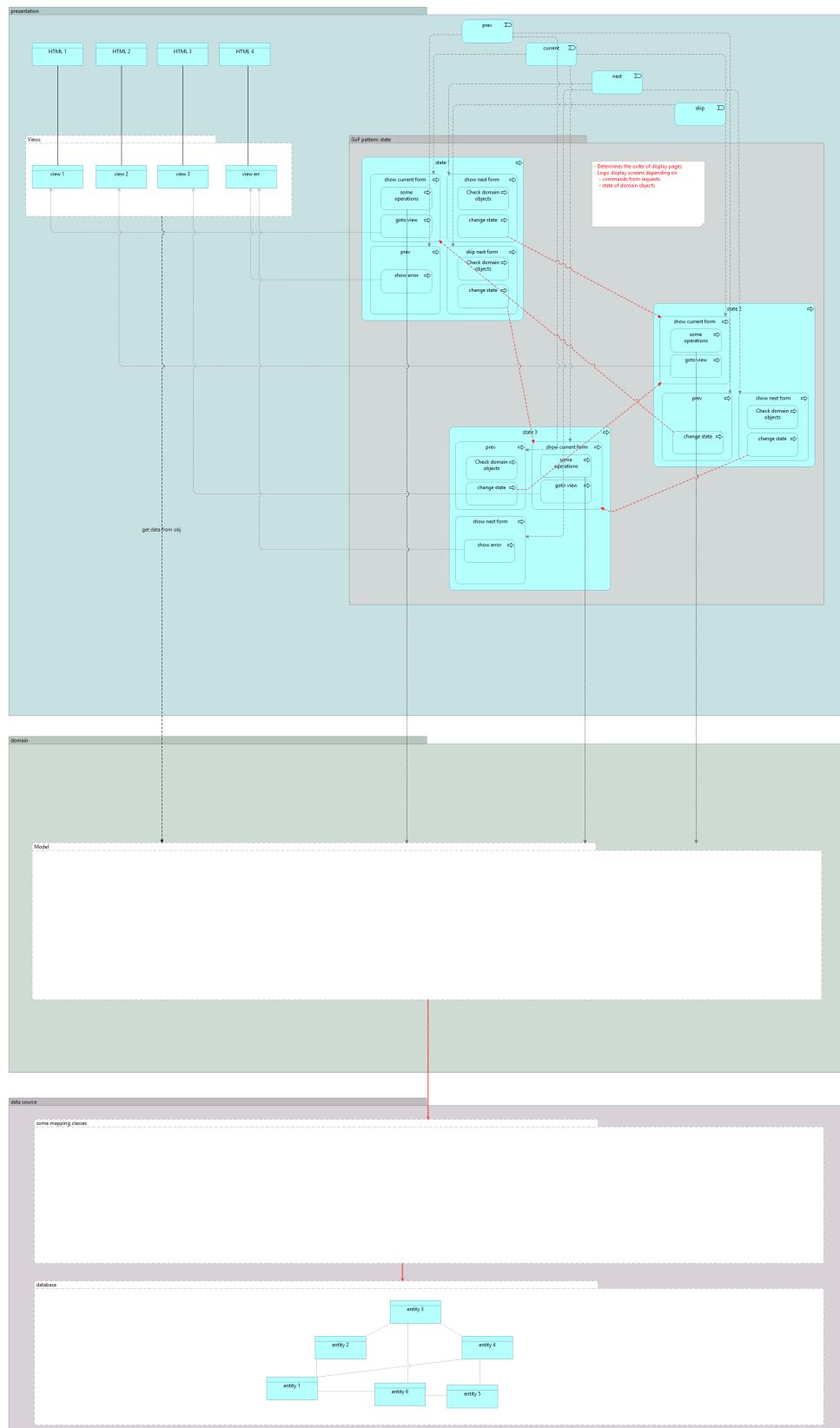
Martin Fowler



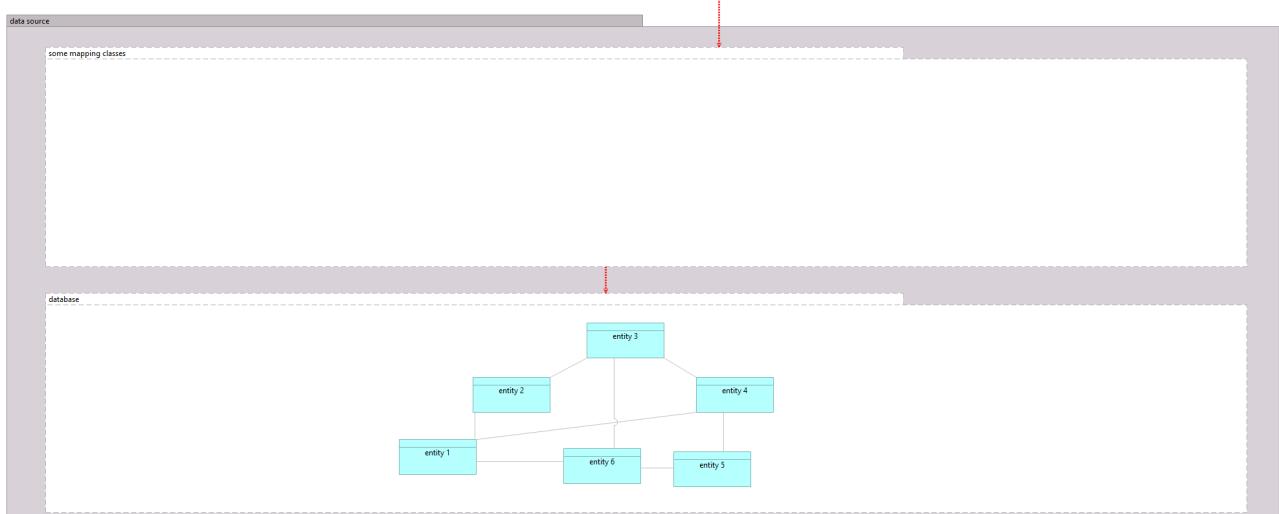
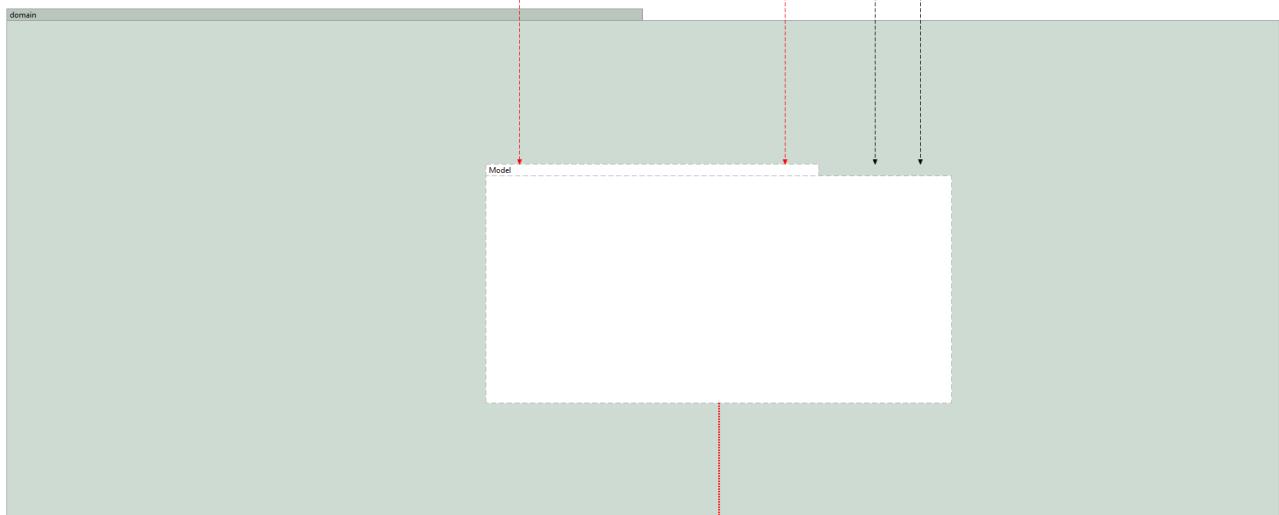
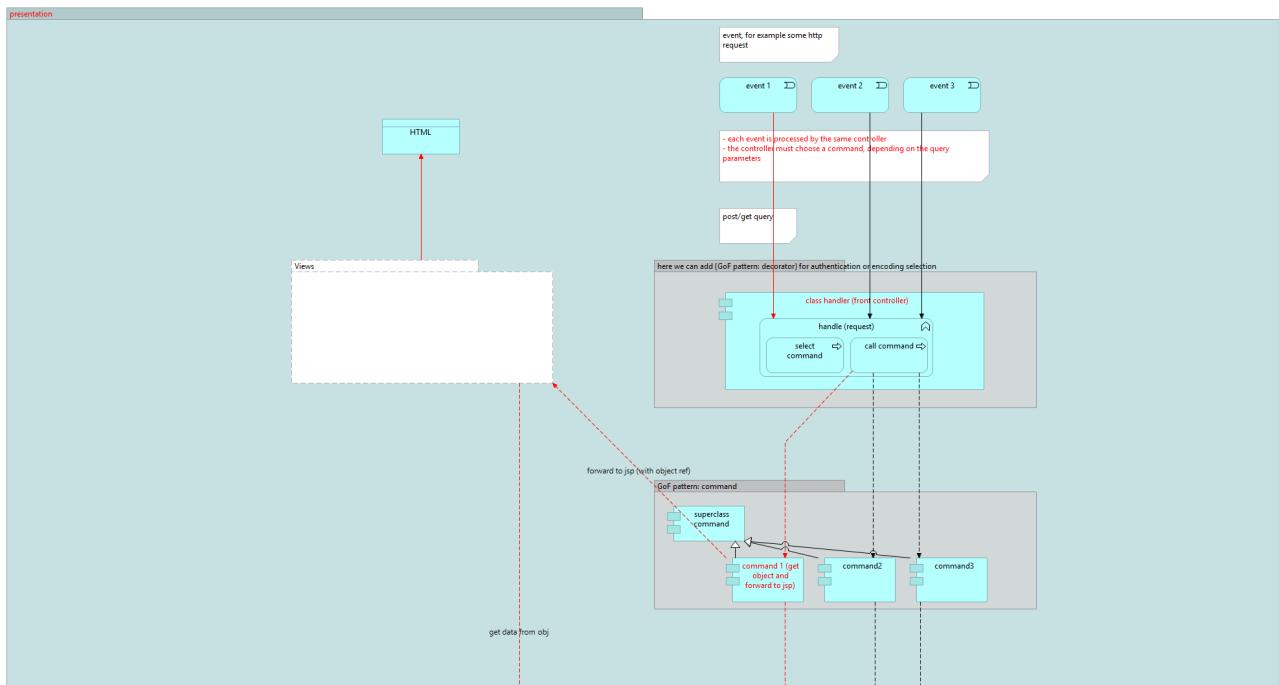
# MODEL VIEW CONTROLLER



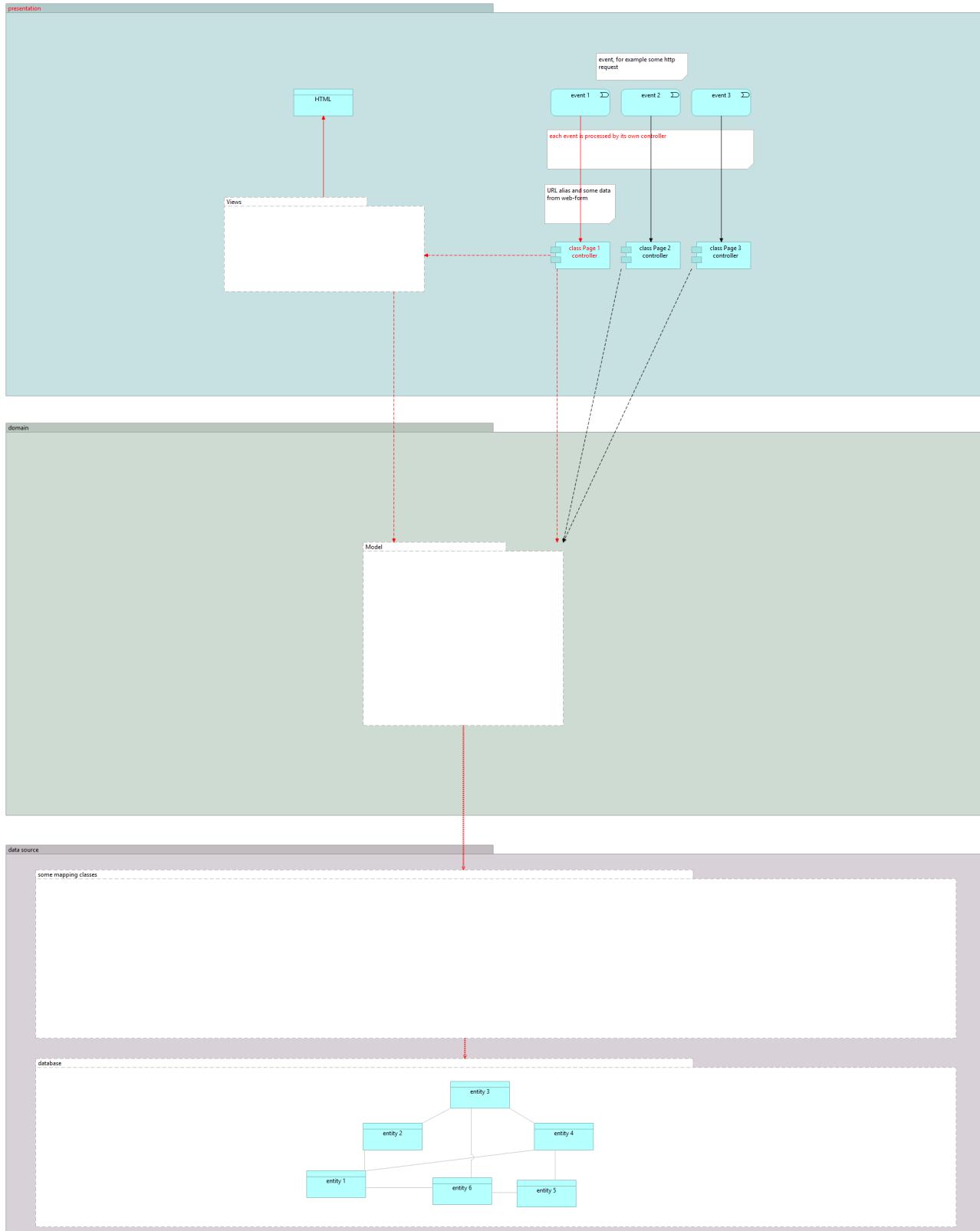
# APPLICATION CONTROLLER



# FRONT CONTROLLER



# PAGE CONTROLLER



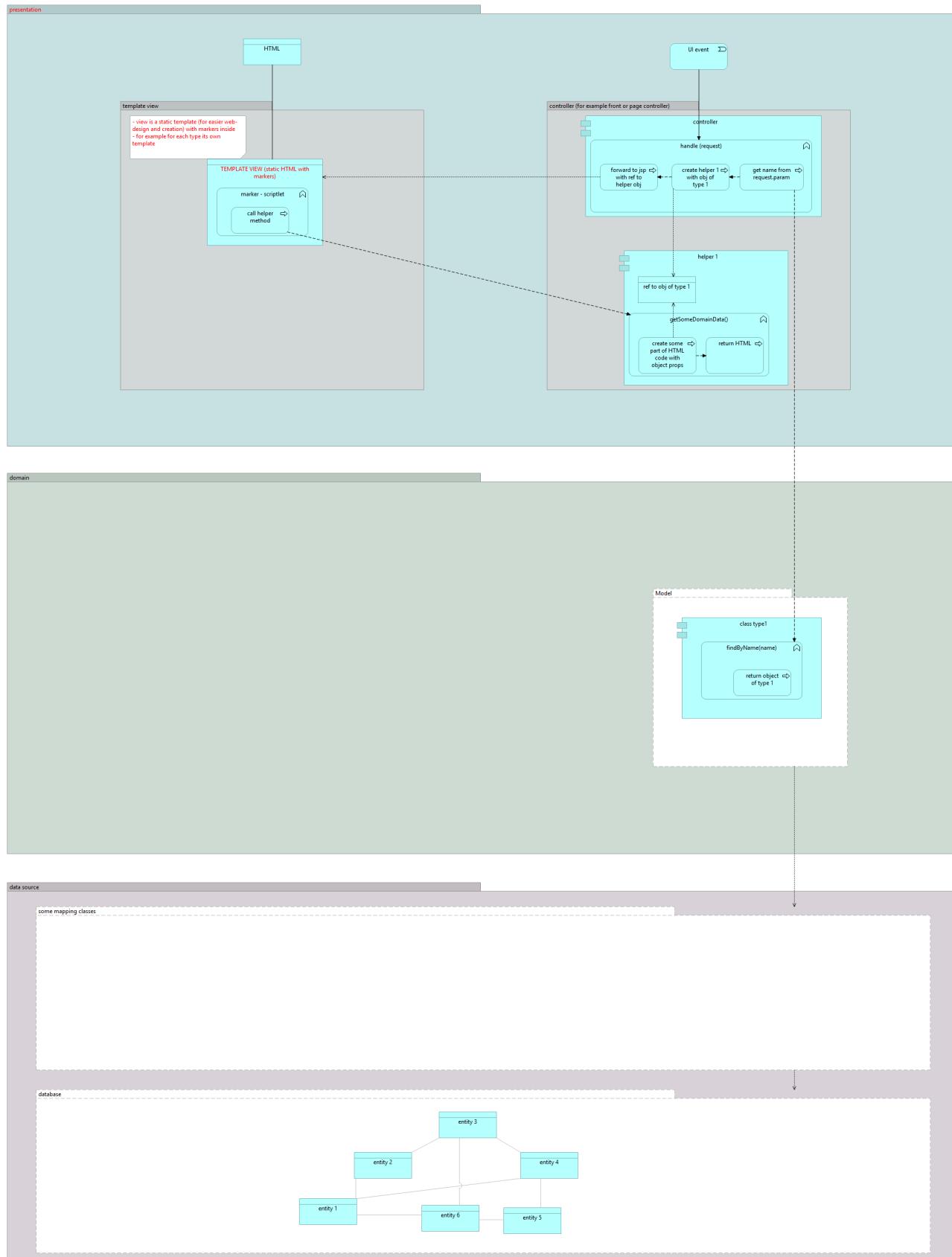
# WEB REPRESENTATION VIEW

ENTERPRISE PATTERNS

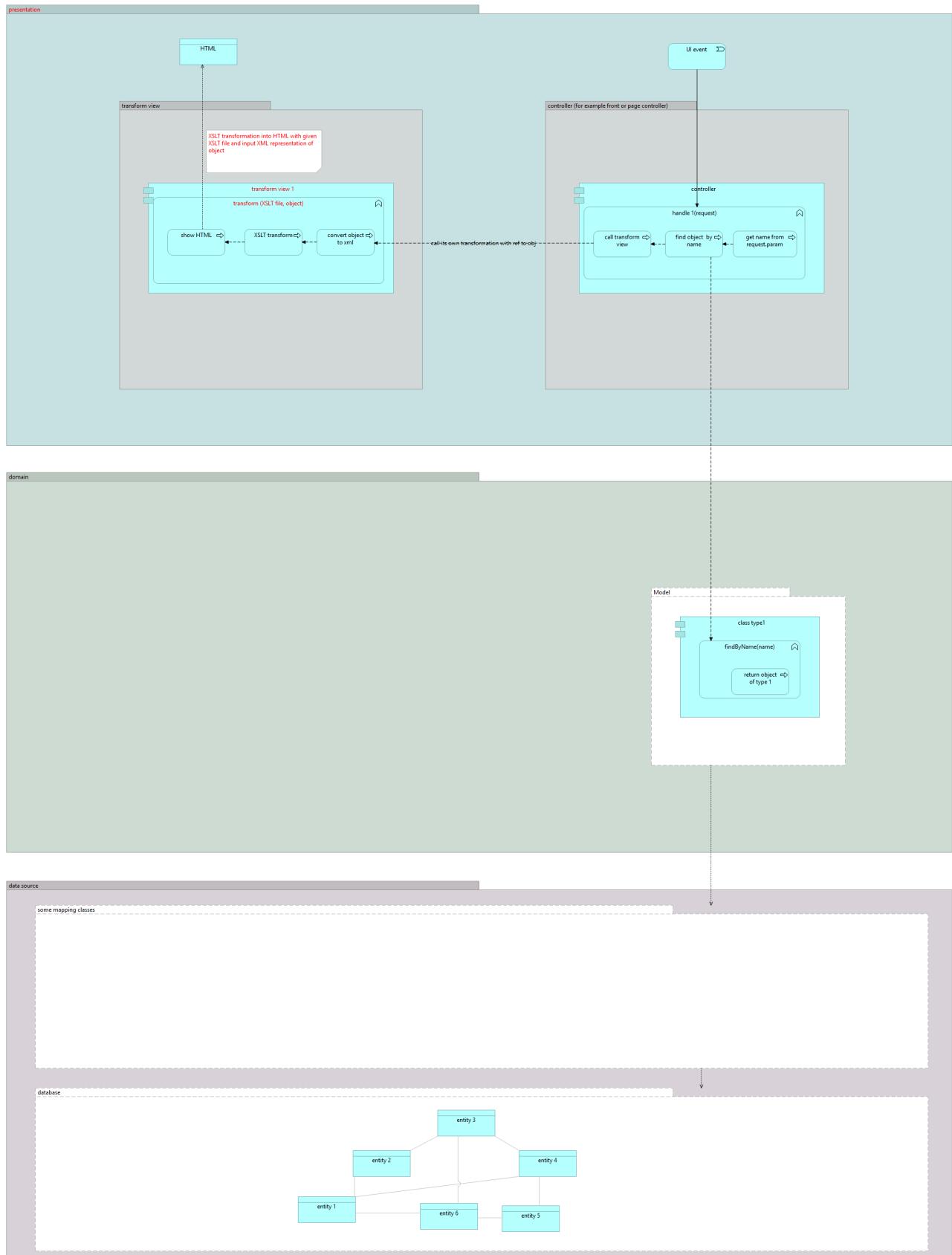
Martin Fowler



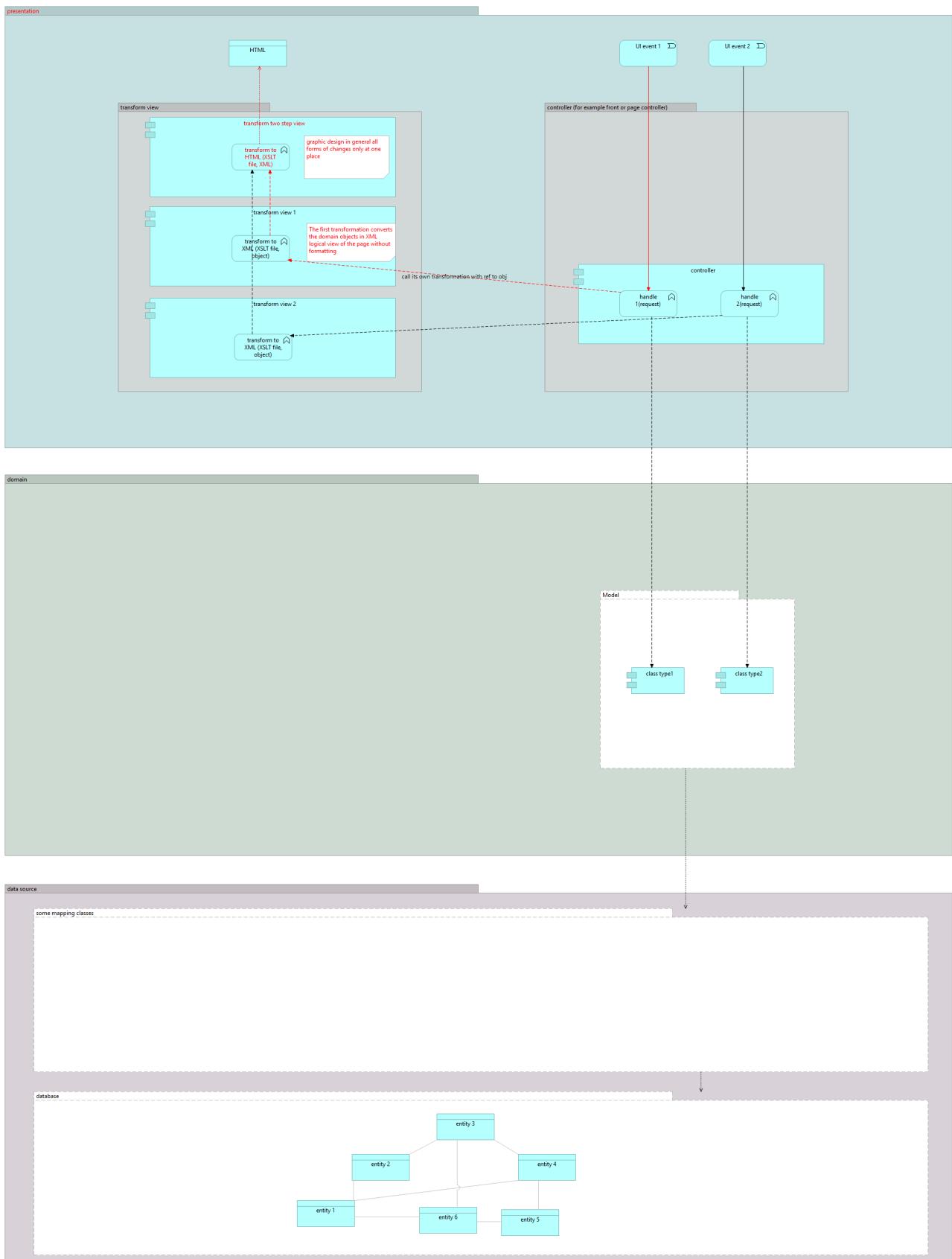
# TEMPLATE VIEW



# TRANSFORM VIEW



# TWO STEP VIEW



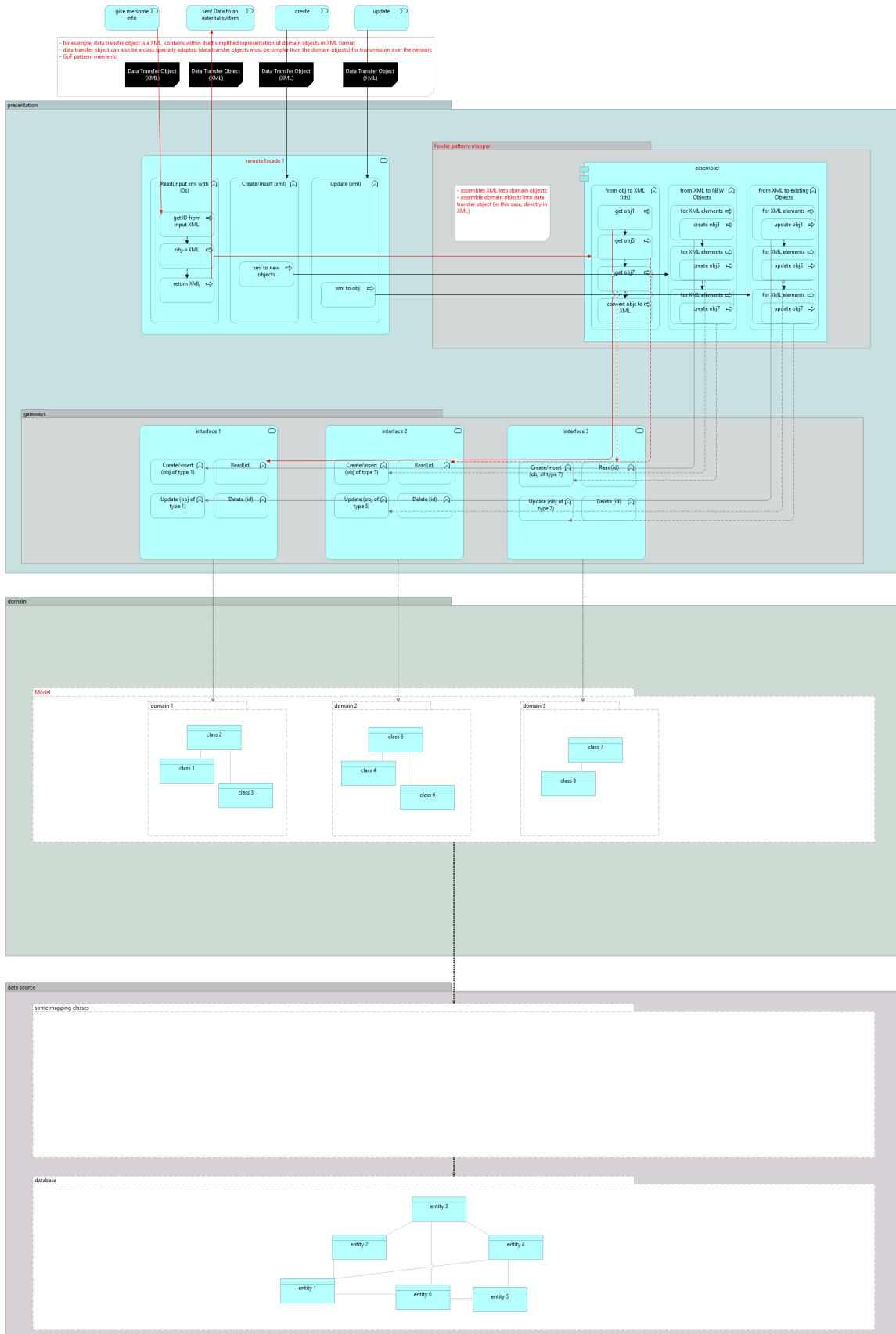
# DISTRIBUTED PROCESSING

ENTERPRISE PATTERNS

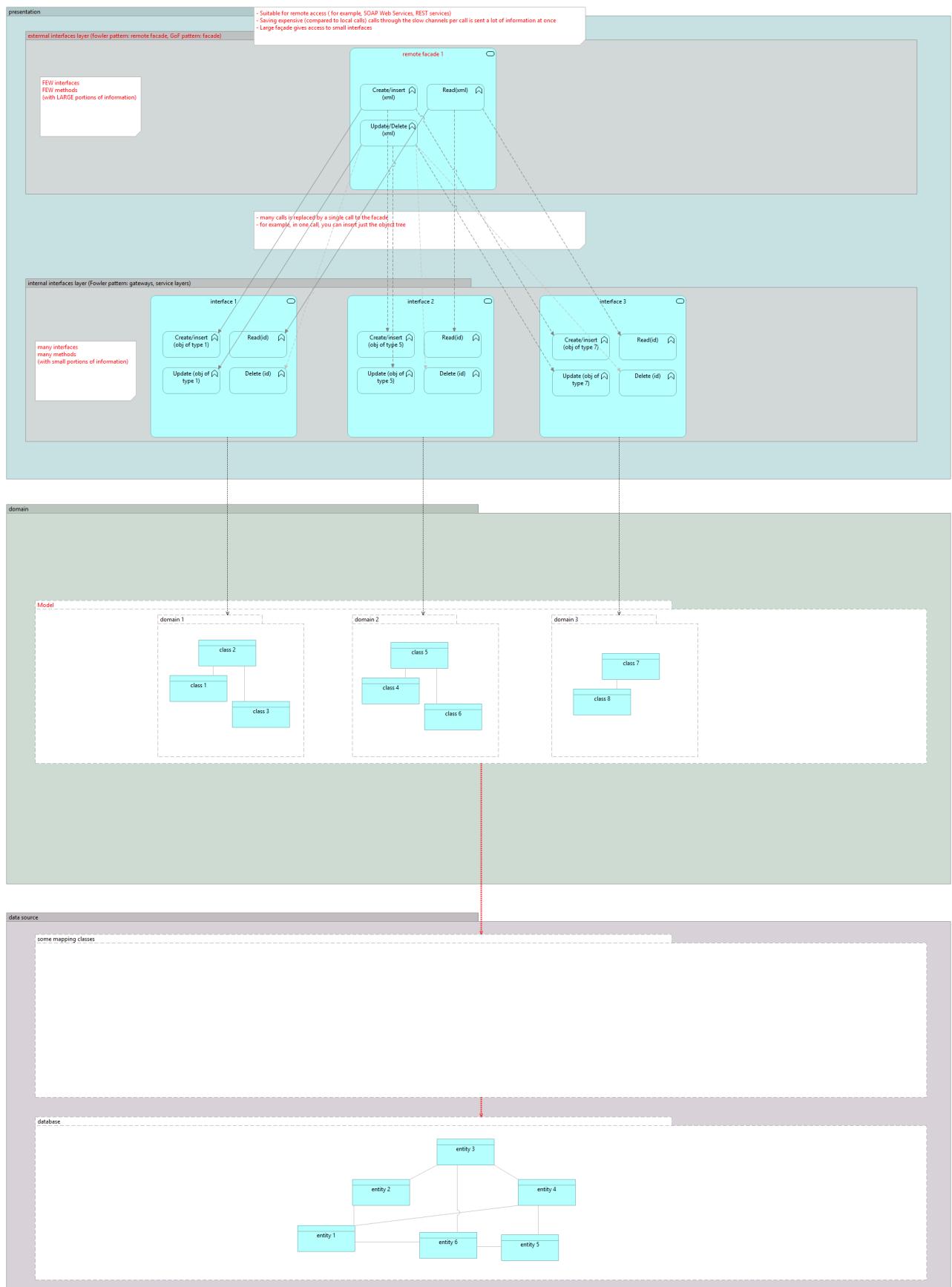
Martin Fowler



# DATA TRANSFER OBJECT



# REMOTE FAÇADE



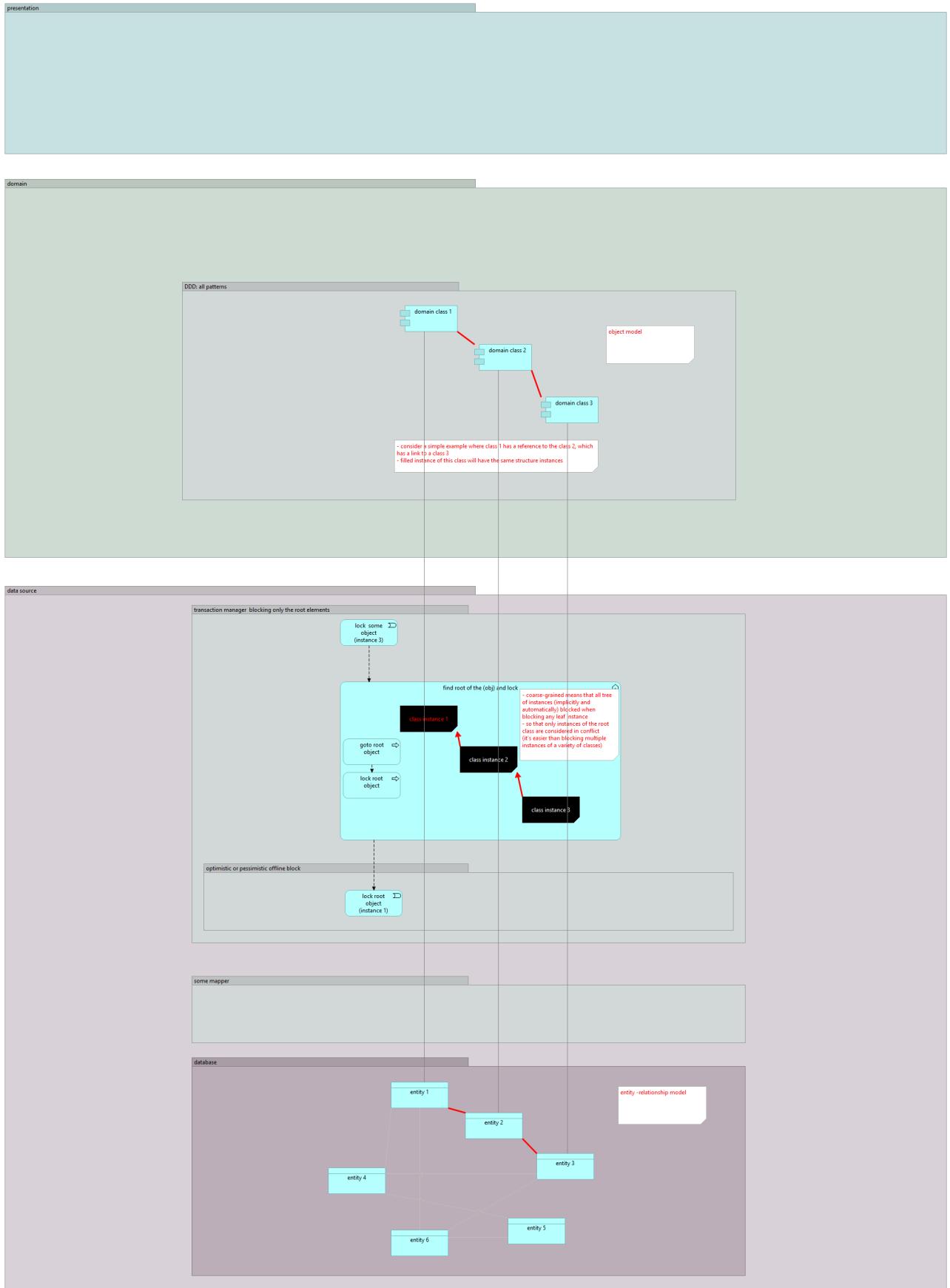
# PARALLEL PROCESSING

ENTERPRISE PATTERNS

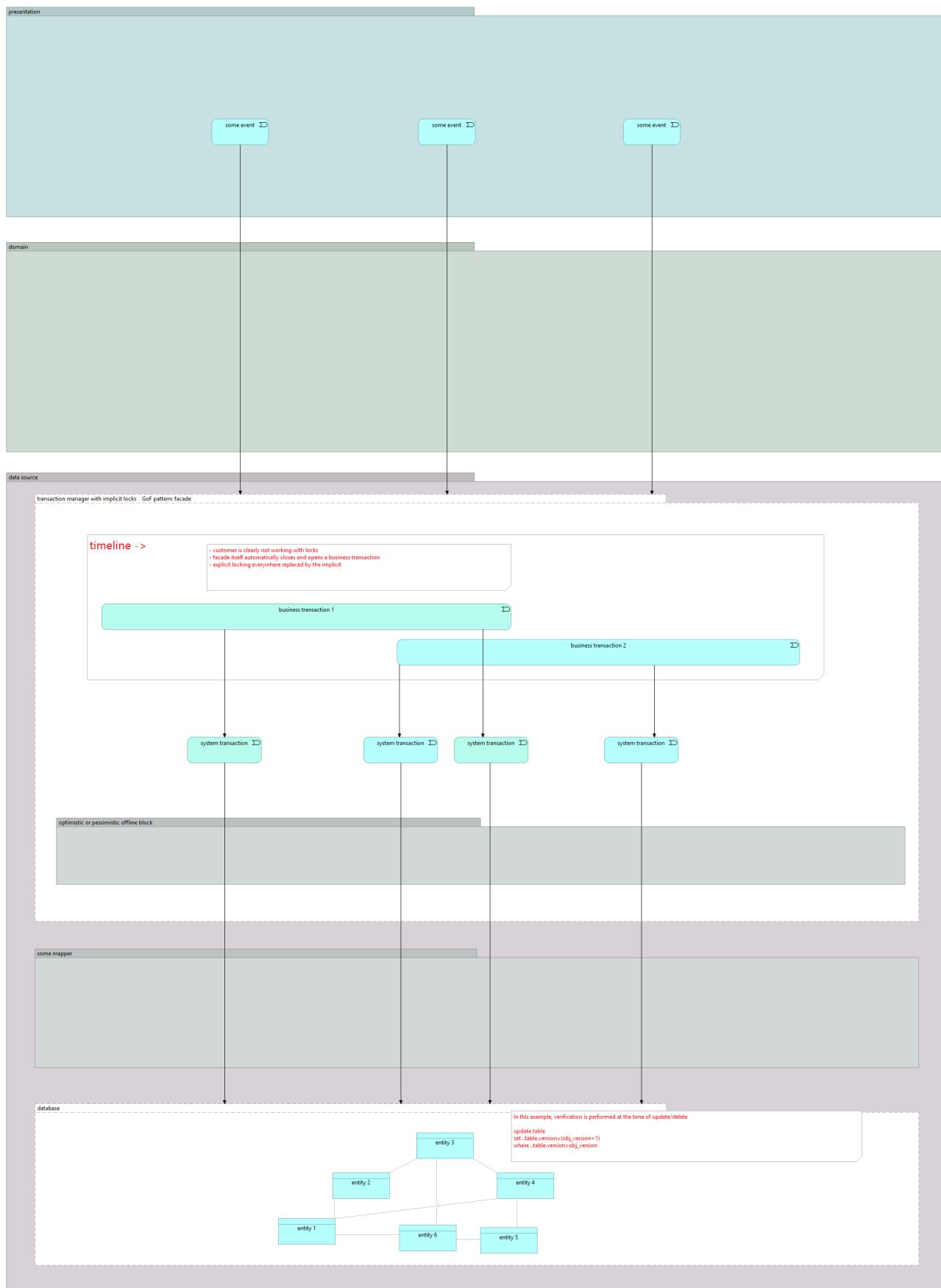
Martin Fowler



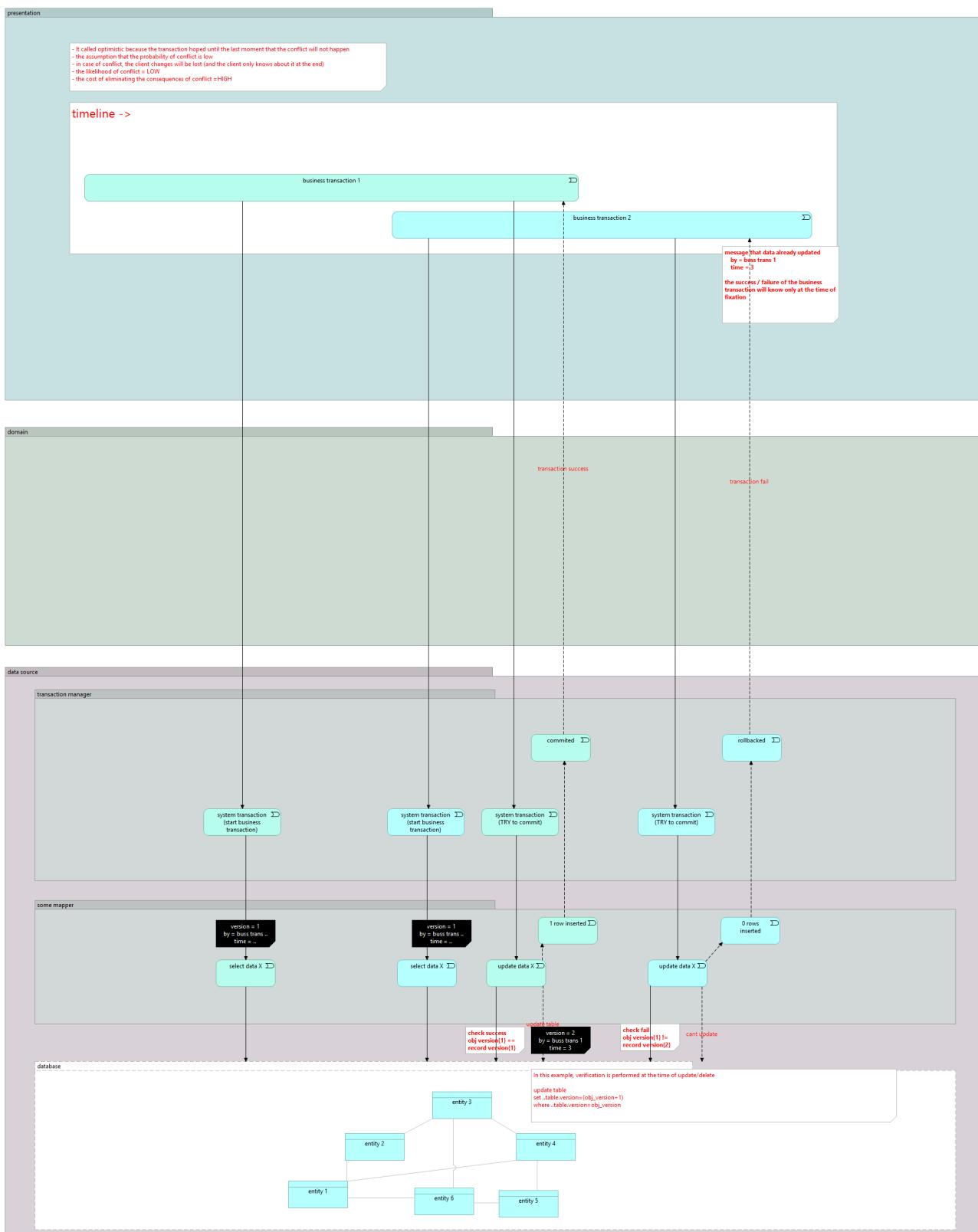
# COARSE-GRAINED LOCK



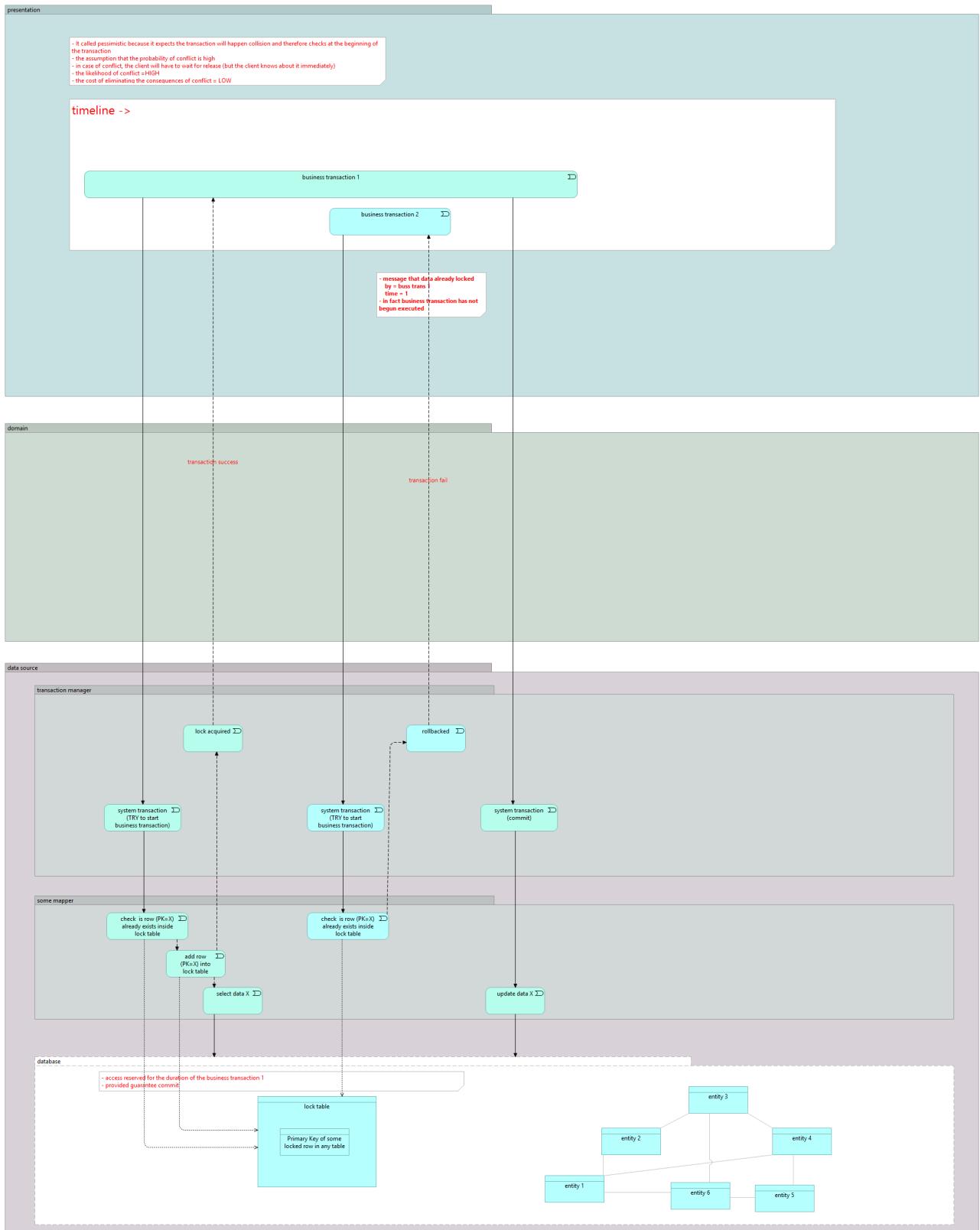
# IMPLICIT LOCK



# OPTIMISTIC OFFLINE LOCK



# PESSIMISTIC OFFLINE LOCK



# SESSION STATE

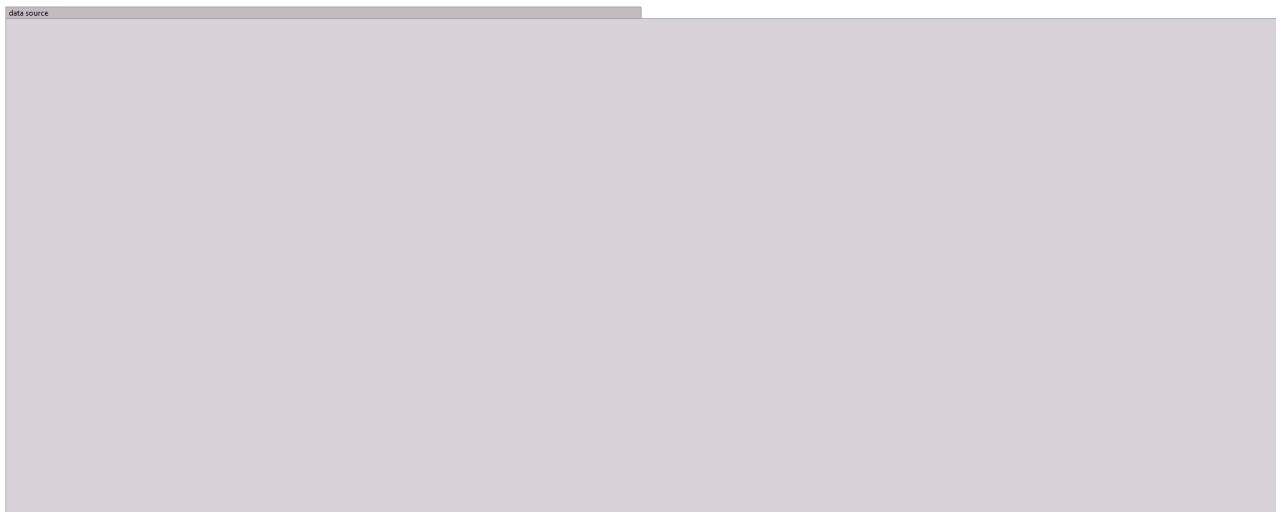
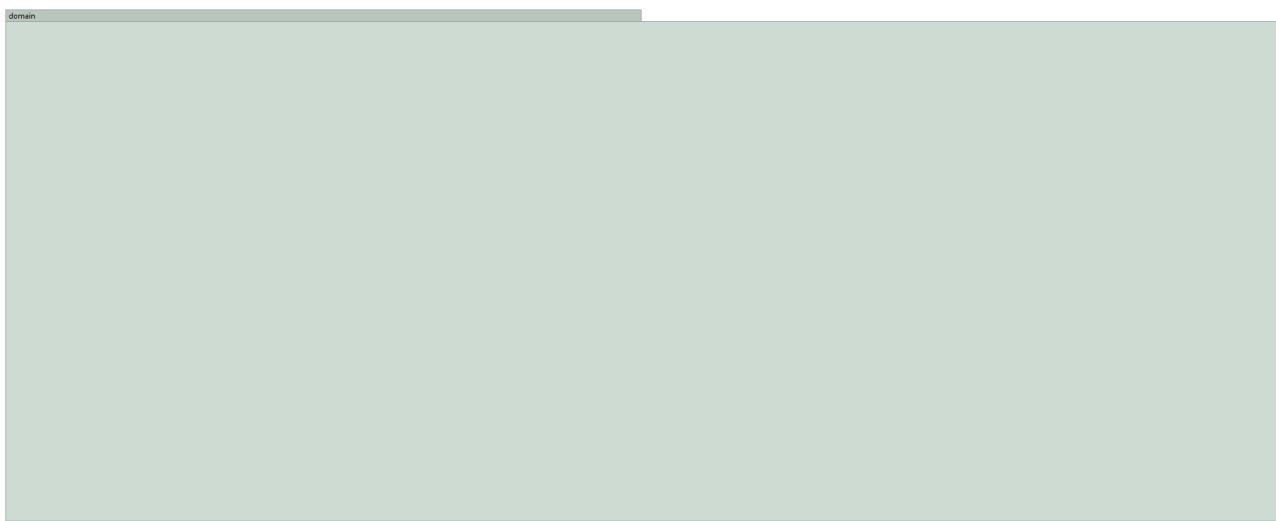
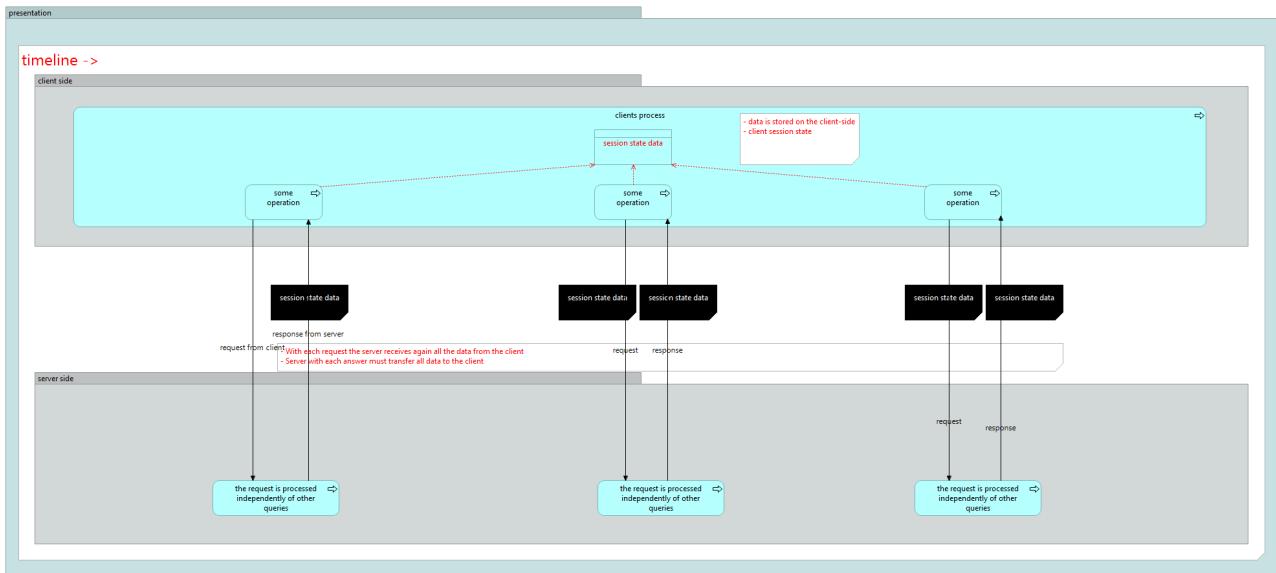
---

ENTERPRISE PATTERNS

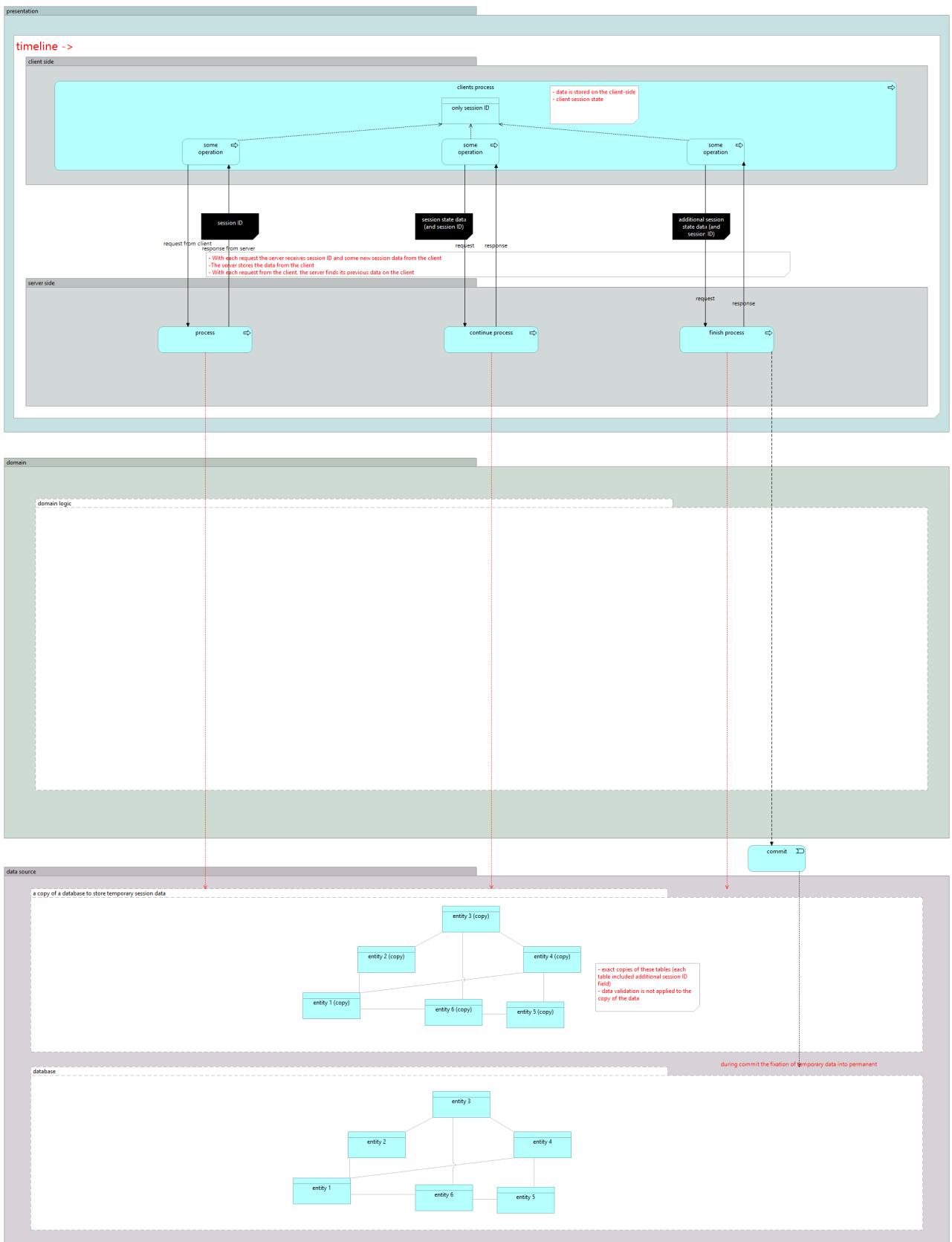
Martin Fowler



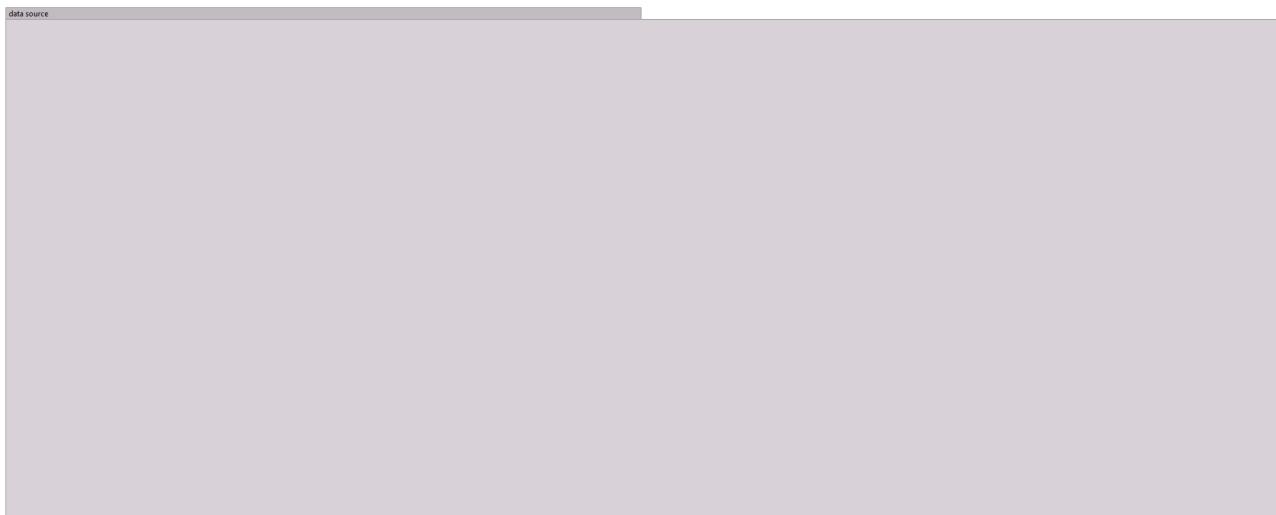
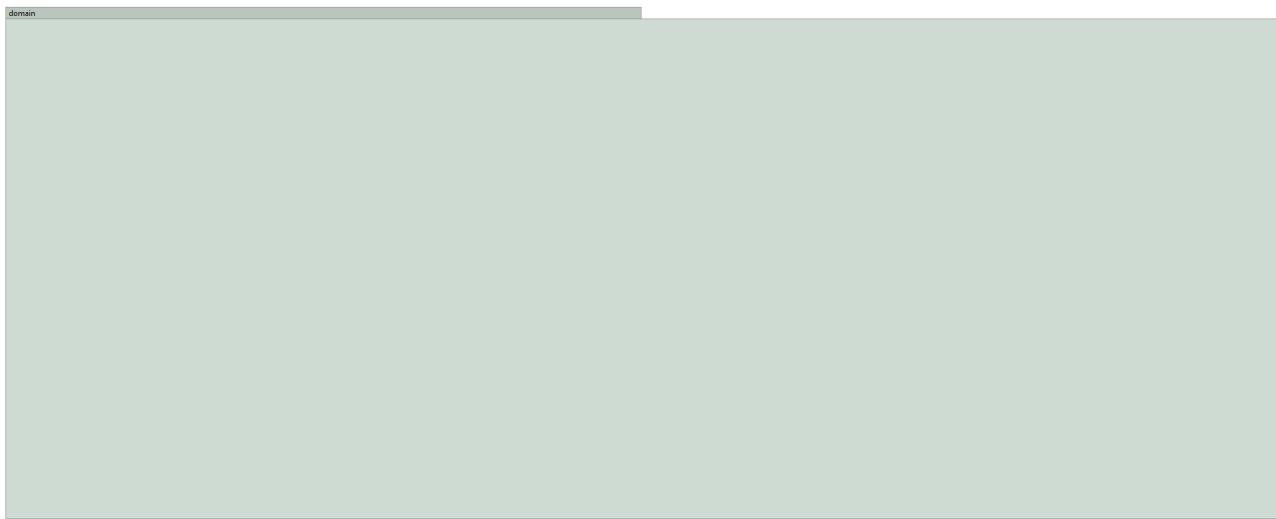
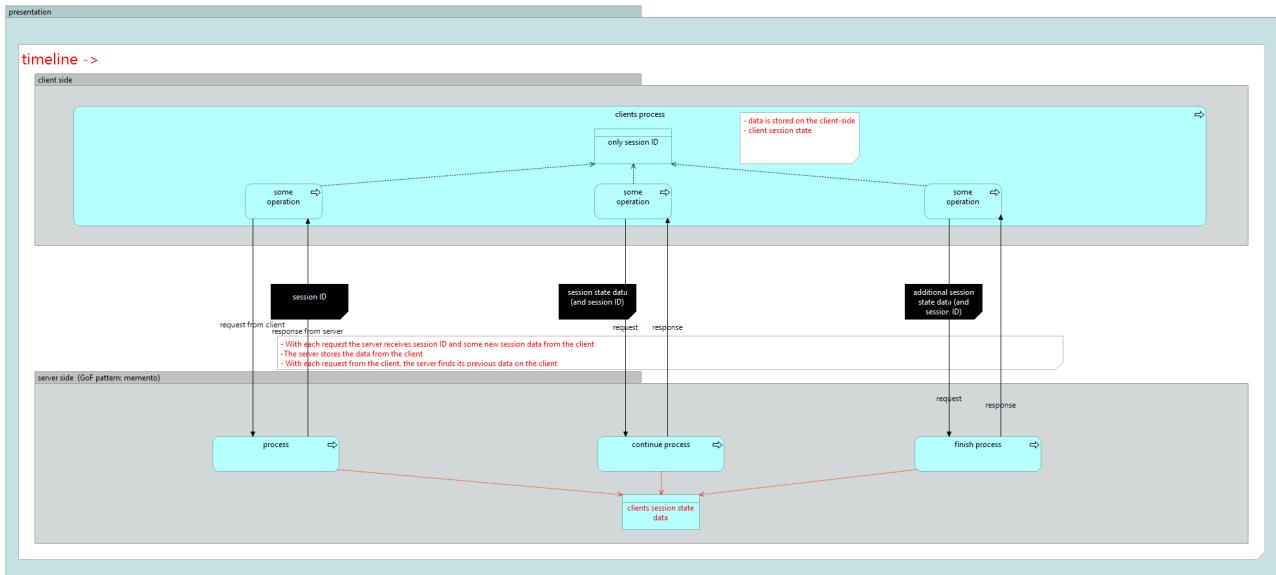
# CLIENT SESSION STATE



# DATABASE SESSION STATE



# SERVER SESSION STATE



# COMMON PATTERNS

---

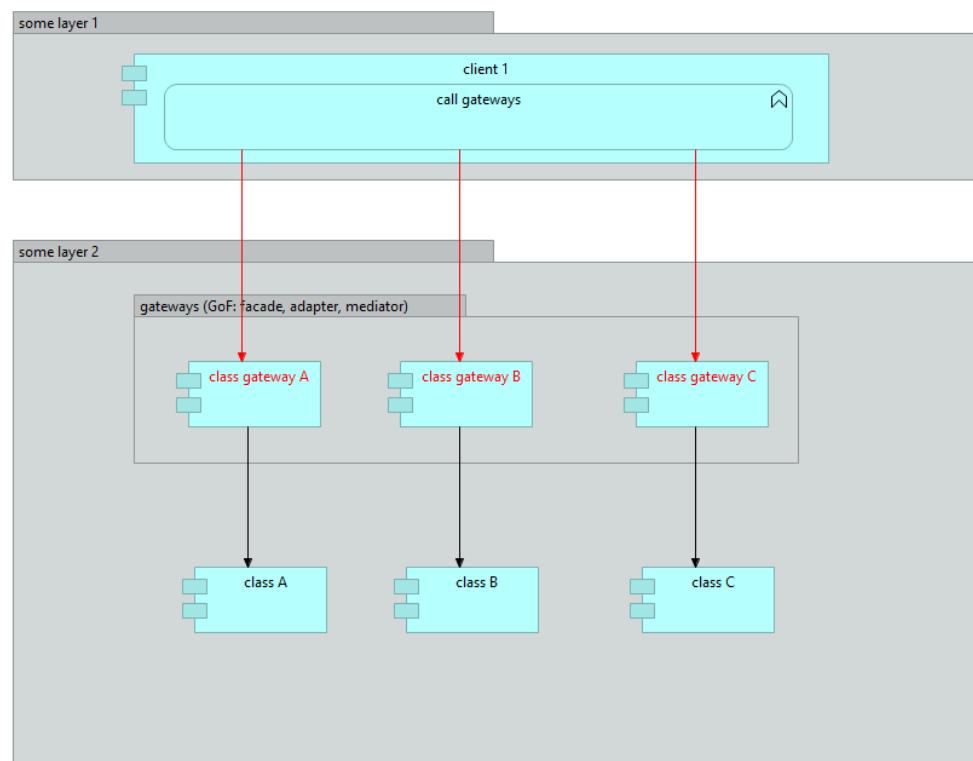
ENTERPRISE PATTERNS

Martin Fowler

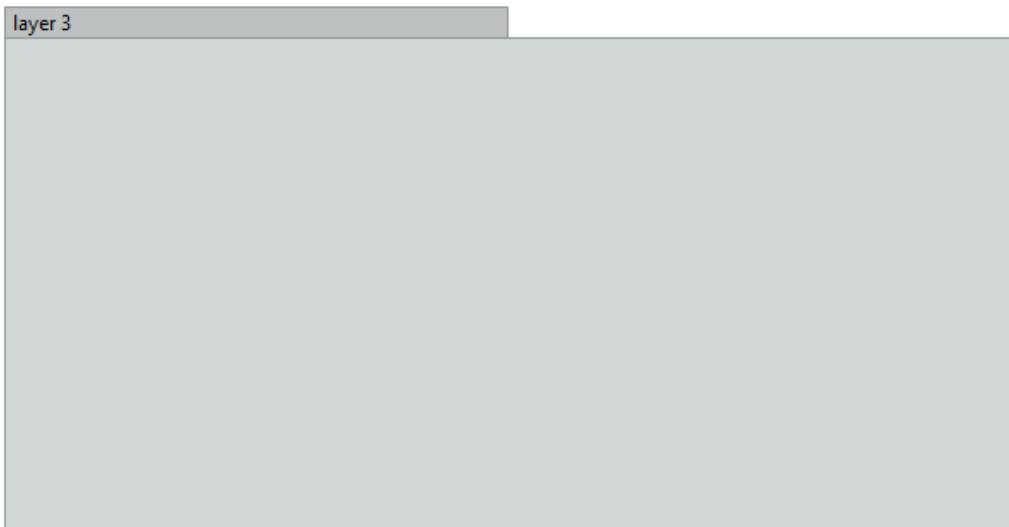
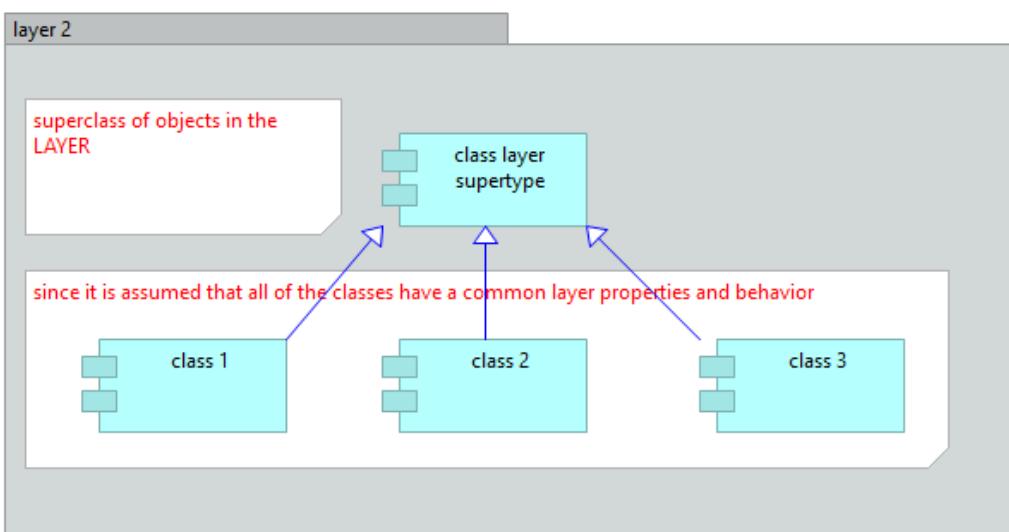


# GATEWAY

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



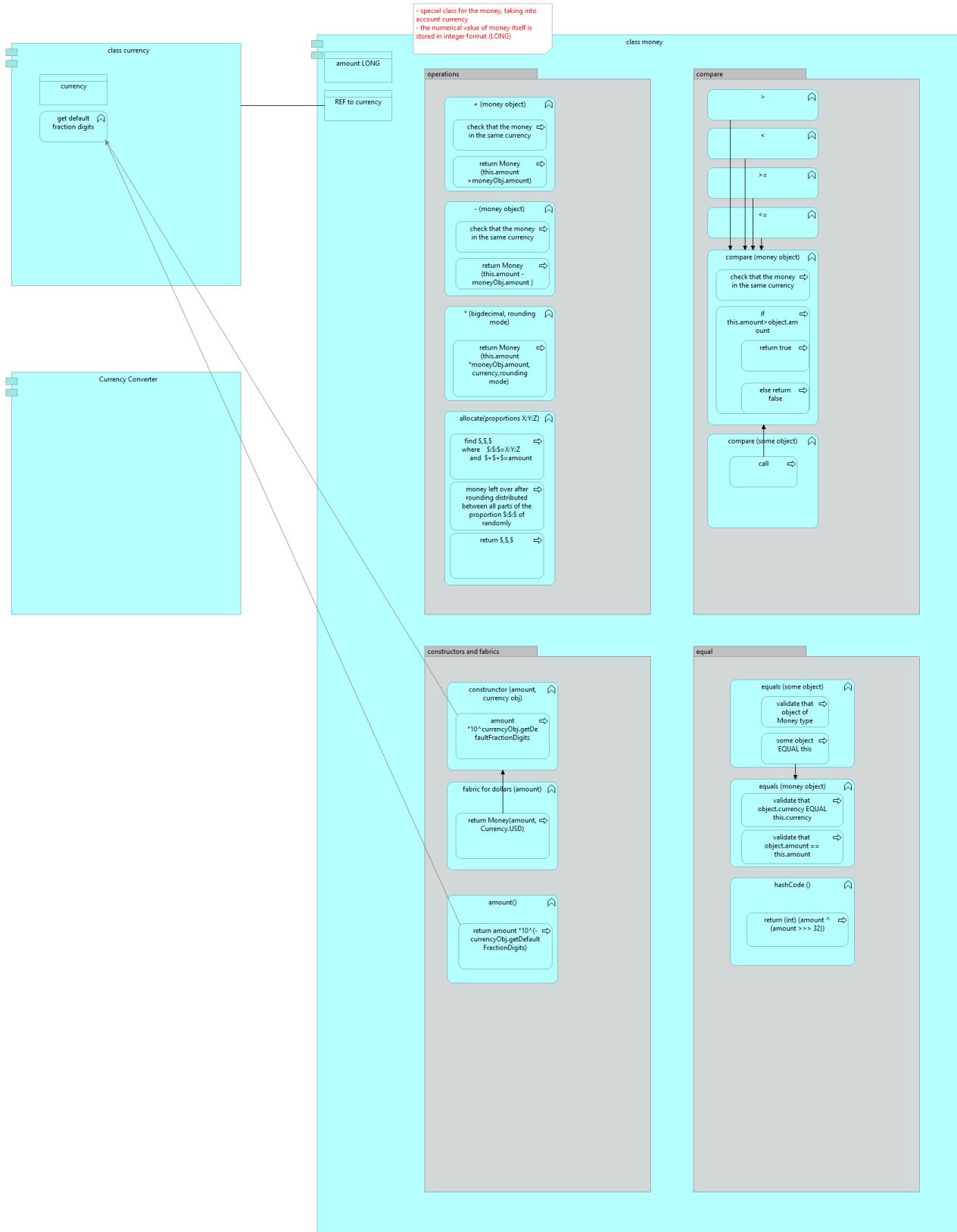
# LAYER SUPERTYPE



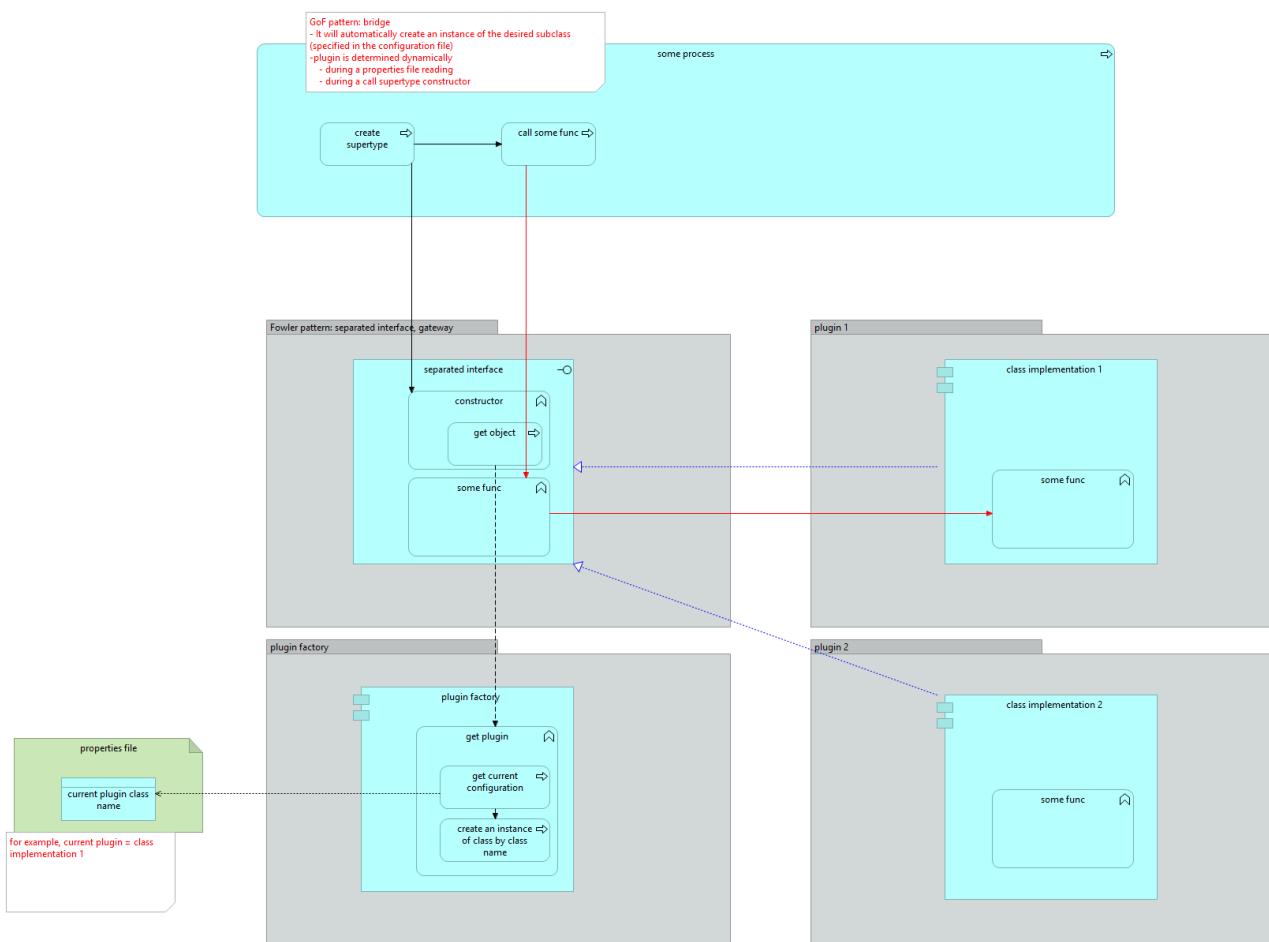
# MAPPER



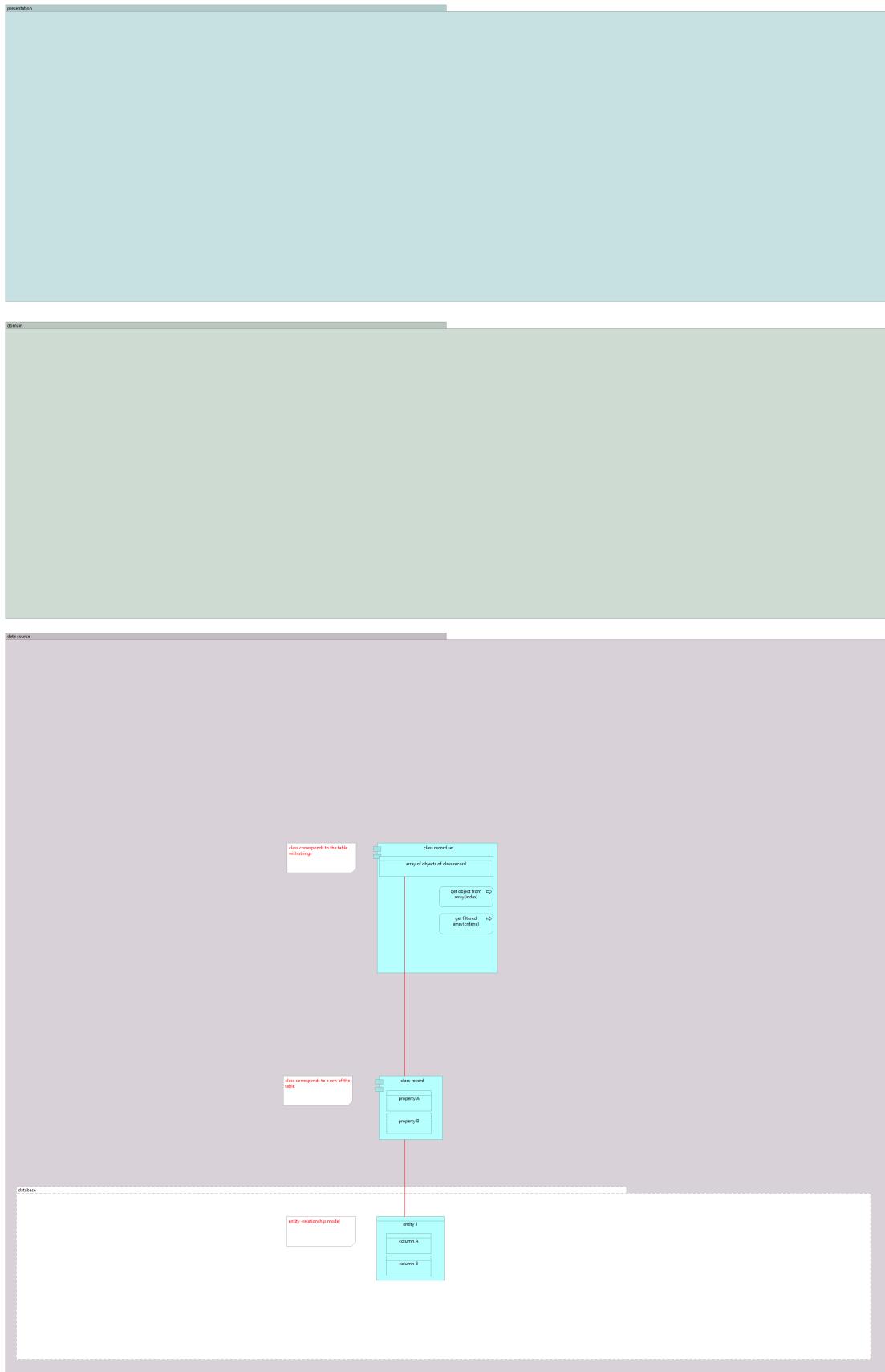
# MONEY



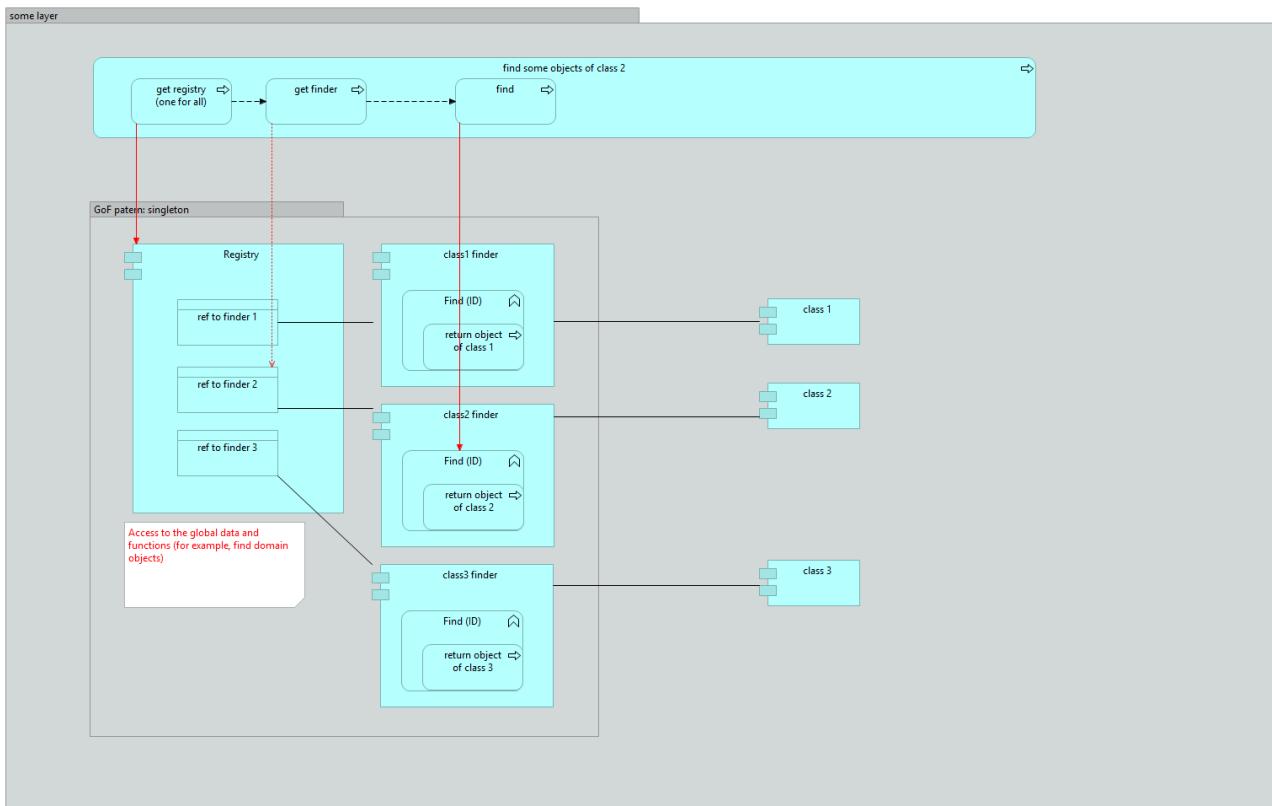
# PLUGIN



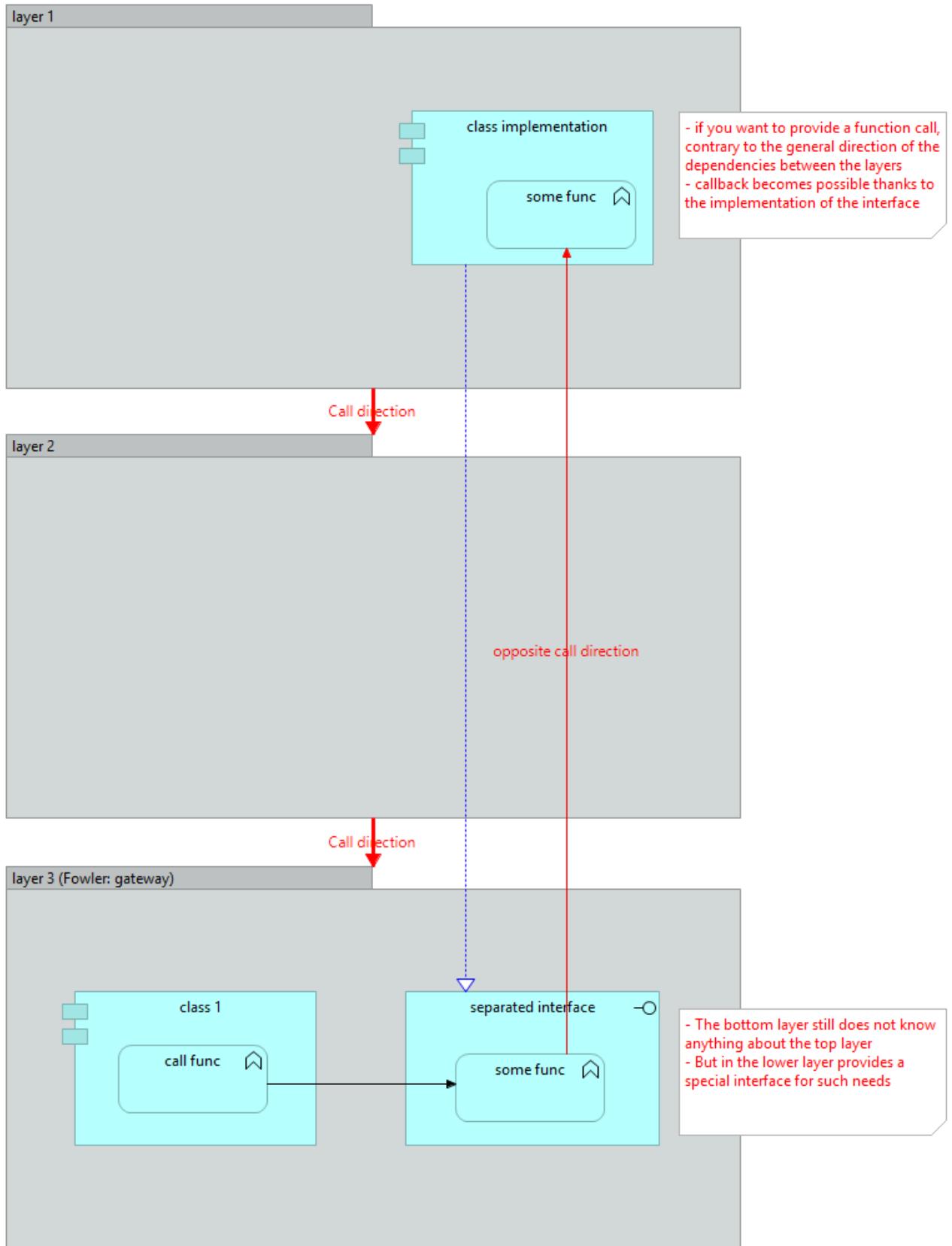
# RECORD SET



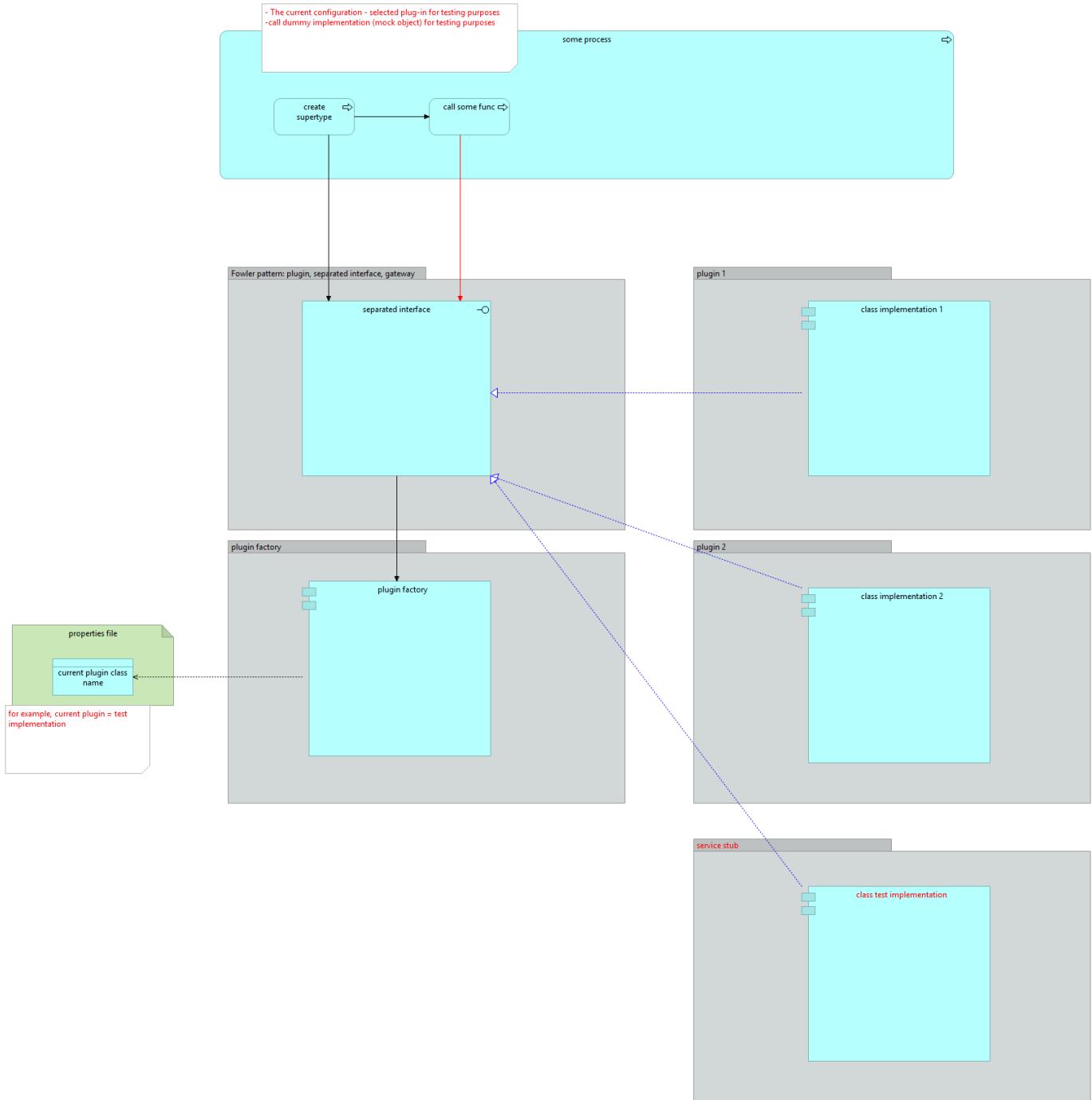
# REGISTRY



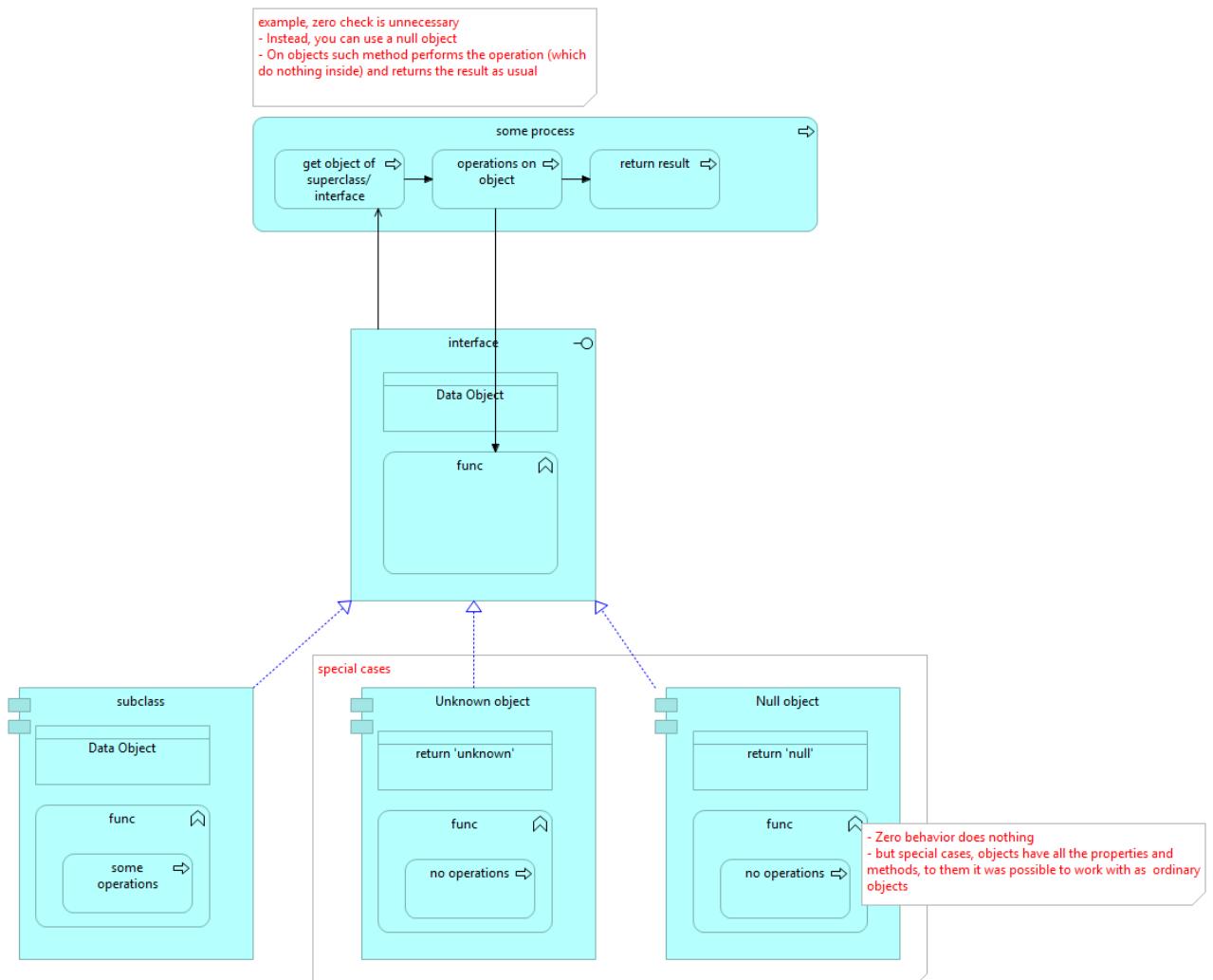
# SEPARATED INTERFACE



# SERVICE STUB

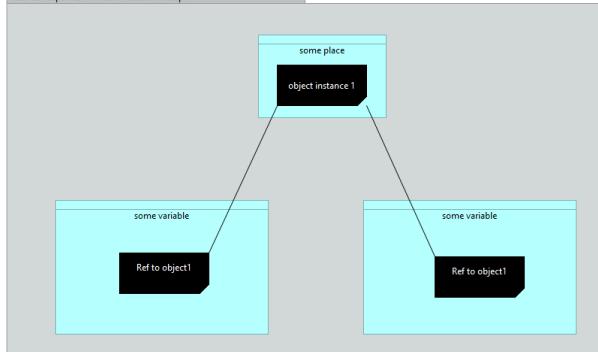


# SPECIAL CASE

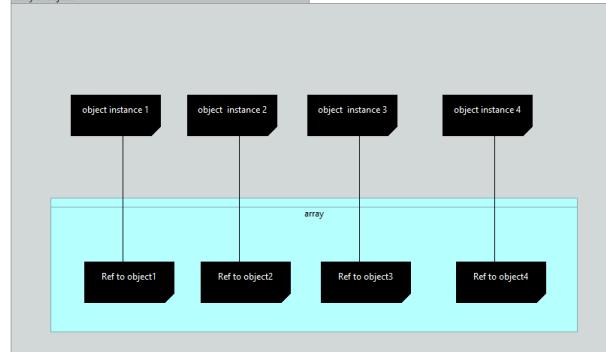


# VALUE OBJECT

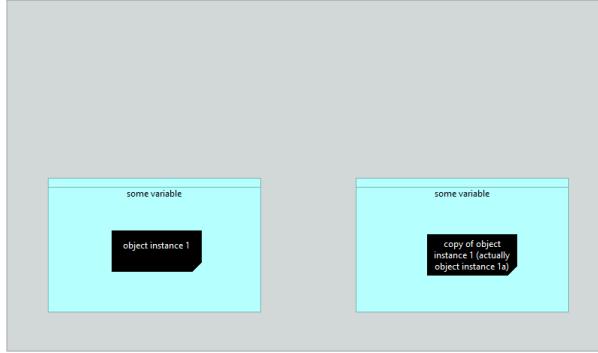
class concept and link transmission concept



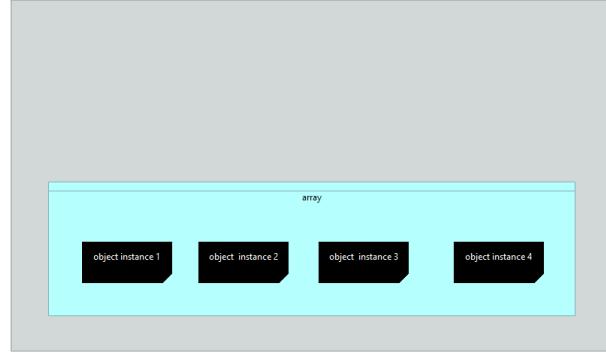
array of objects



value object concept and by value transfer concept



array of value objects

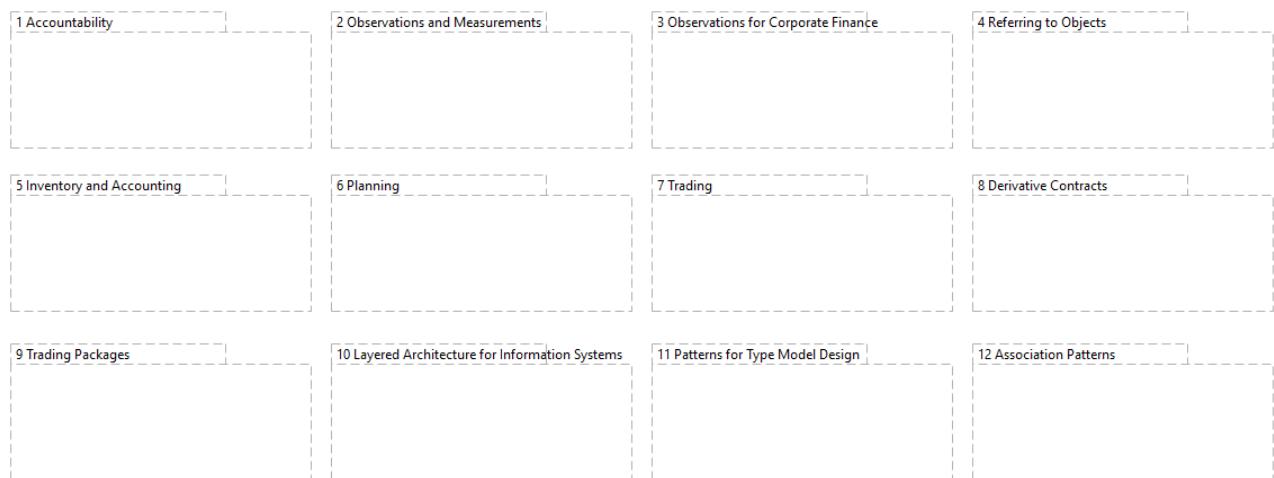


03

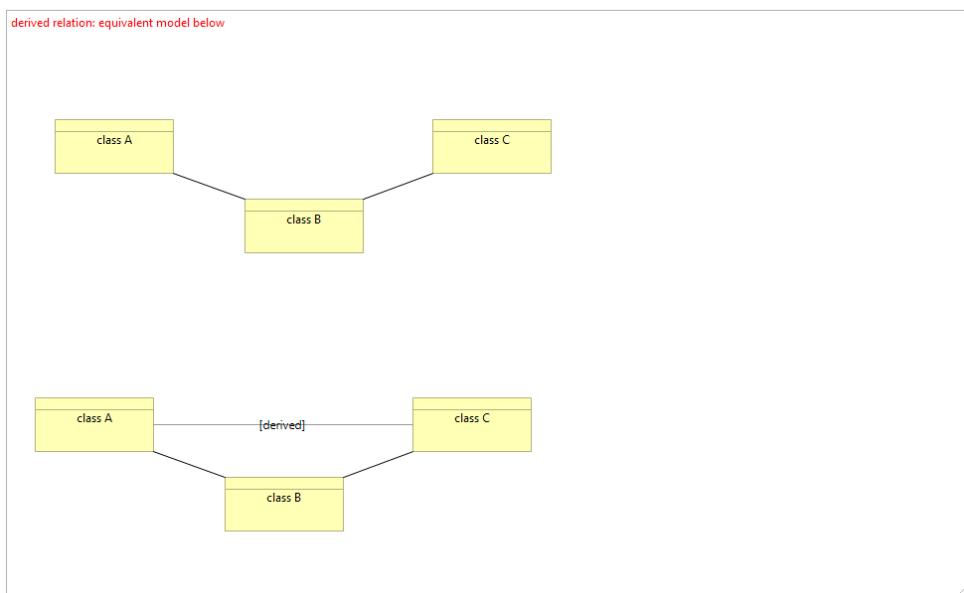
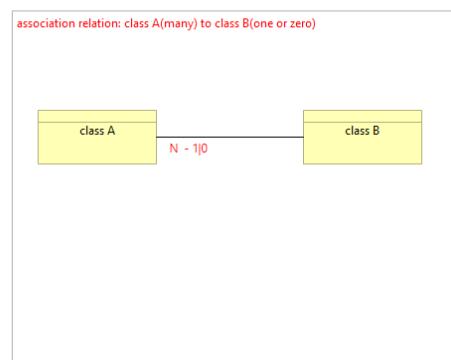
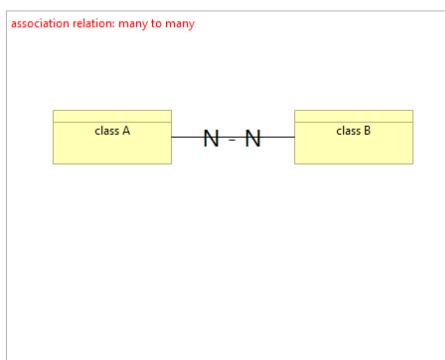
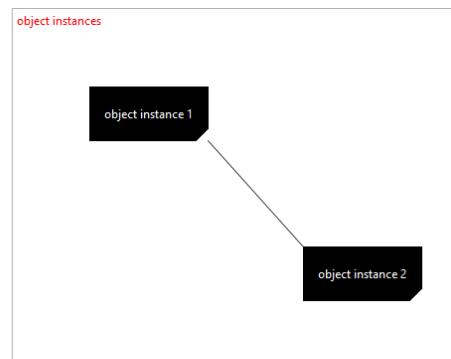
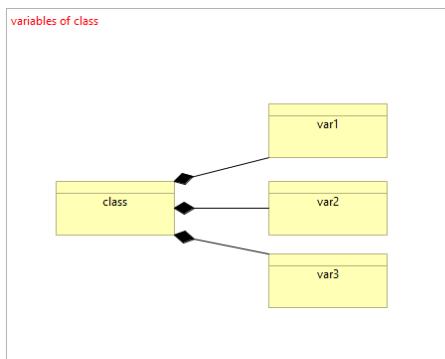
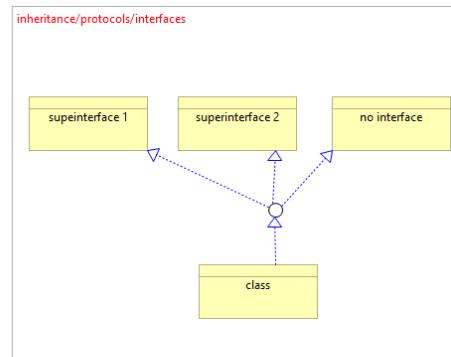
# Analysis Patterns

Reusable Object Models

MARTIN FOWLER



# USED NOTATION

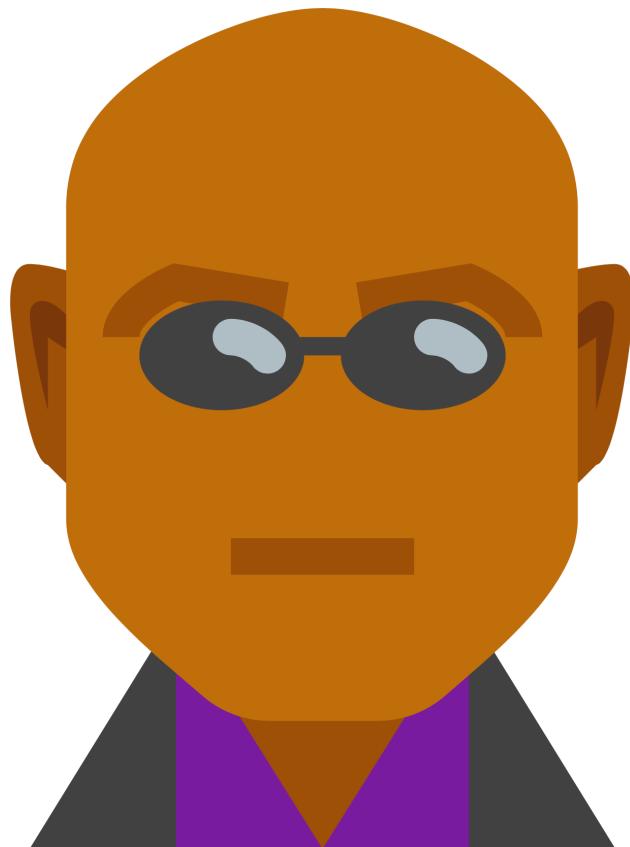


# ACCOUNTABILITY

---

ANALYSIS PATTERNS

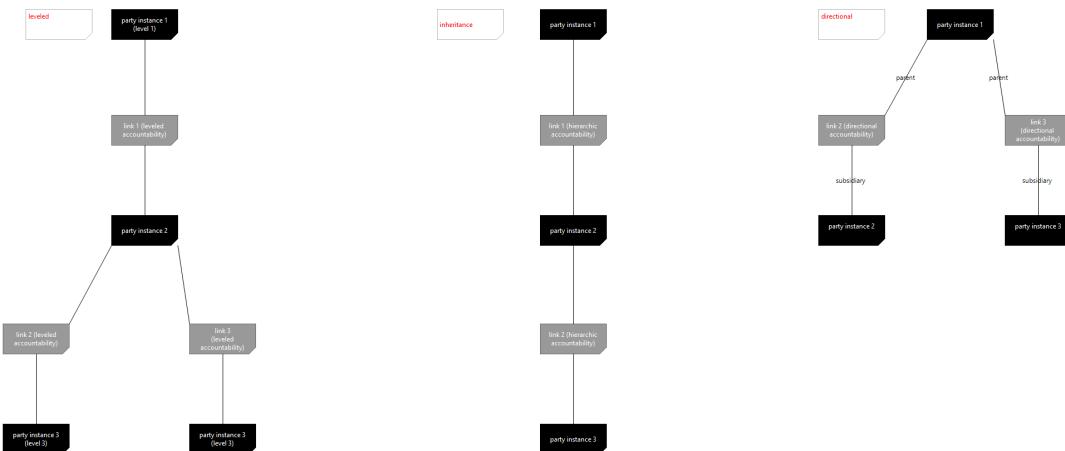
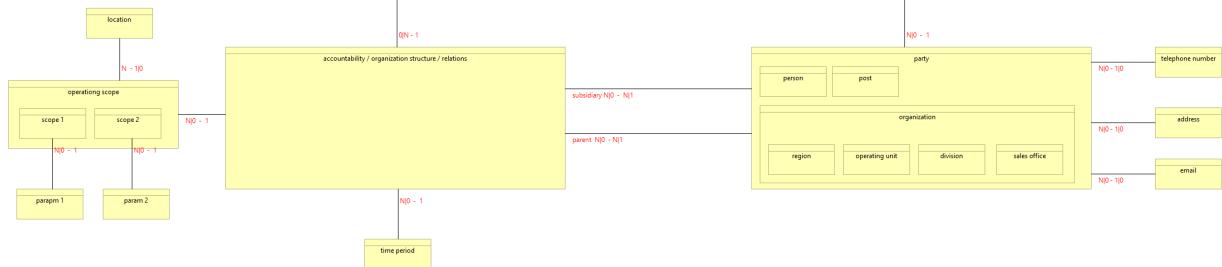
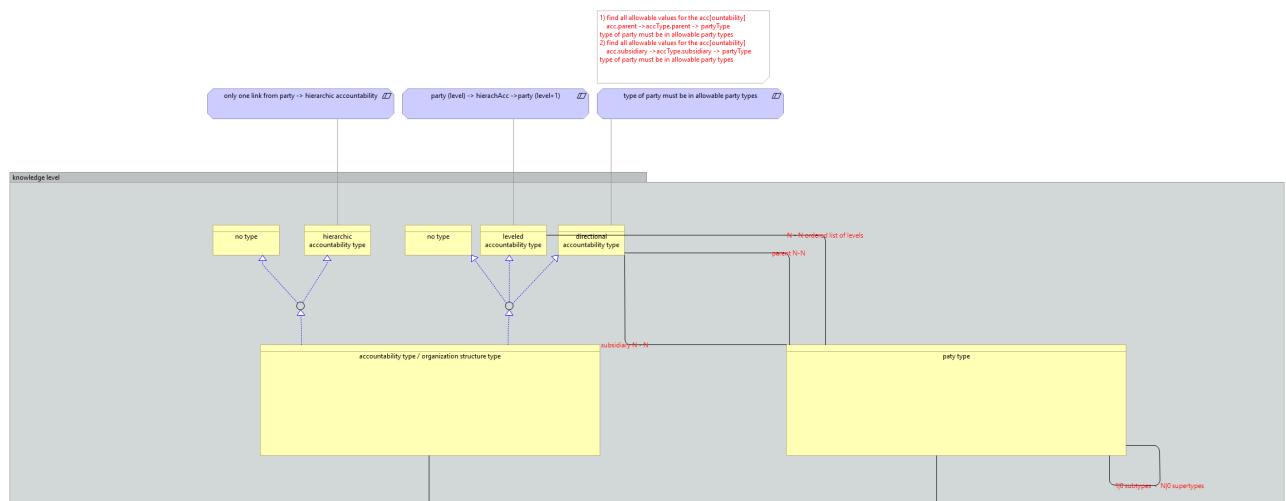
Martin Fowler

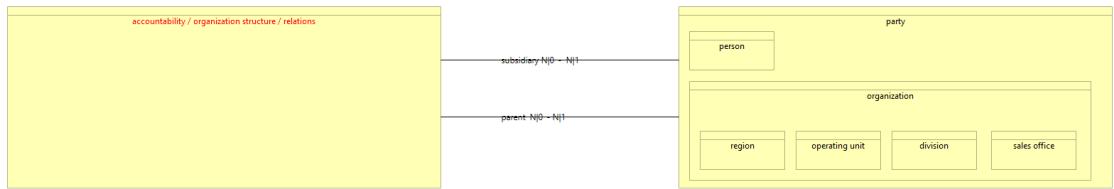


# PARTY

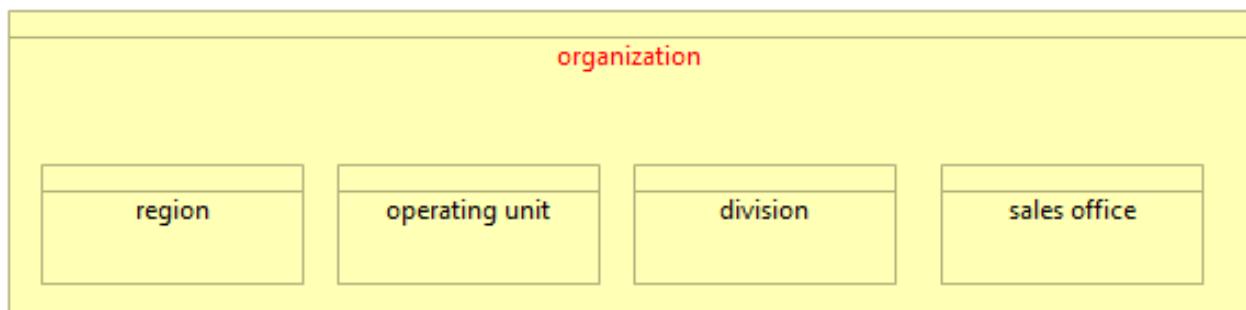


# ACCOUNTABILITY

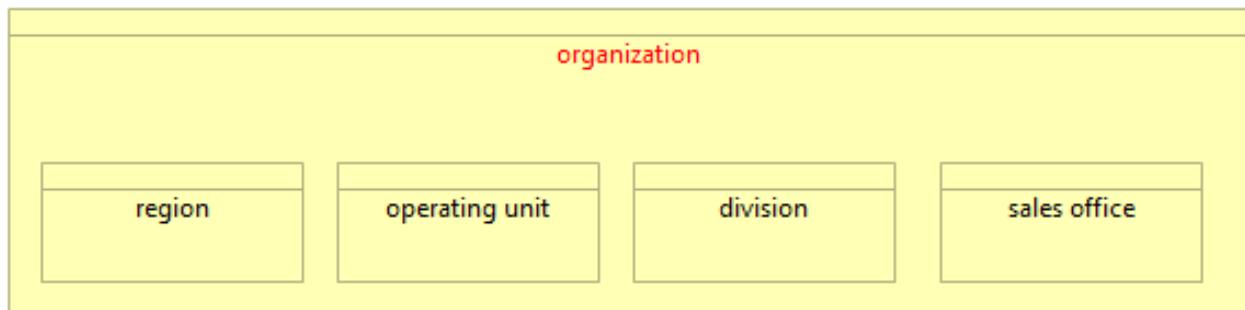




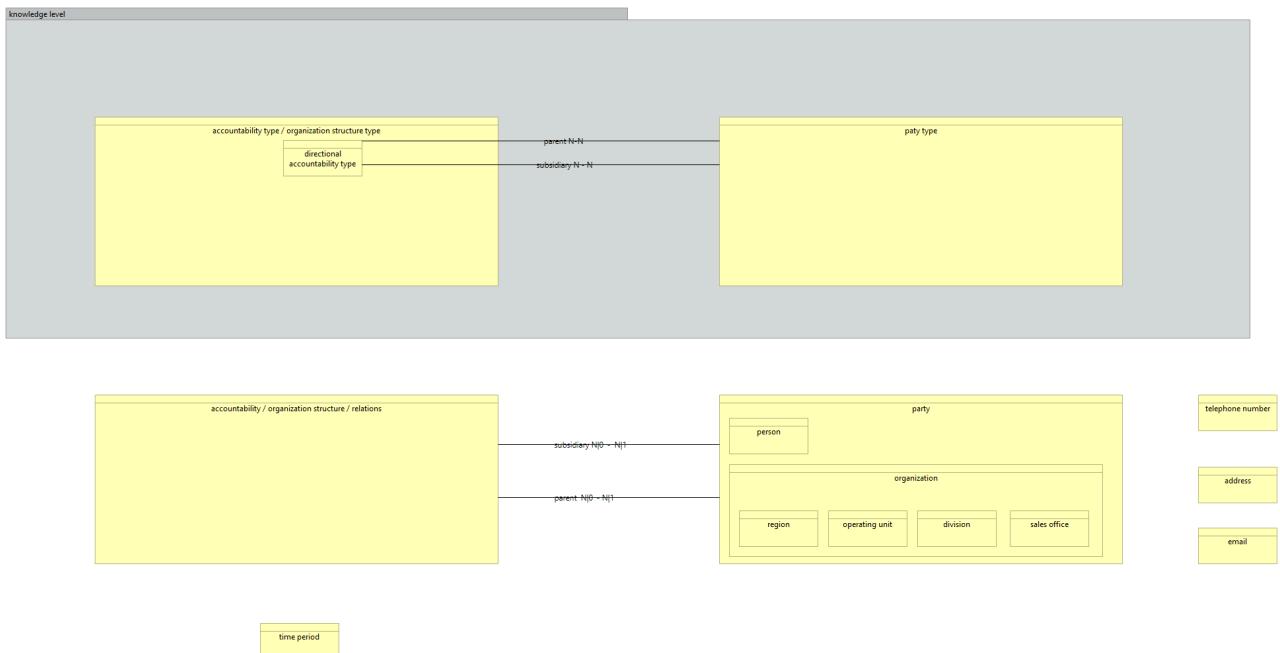
# ORGANIZATION HIERARCHIES



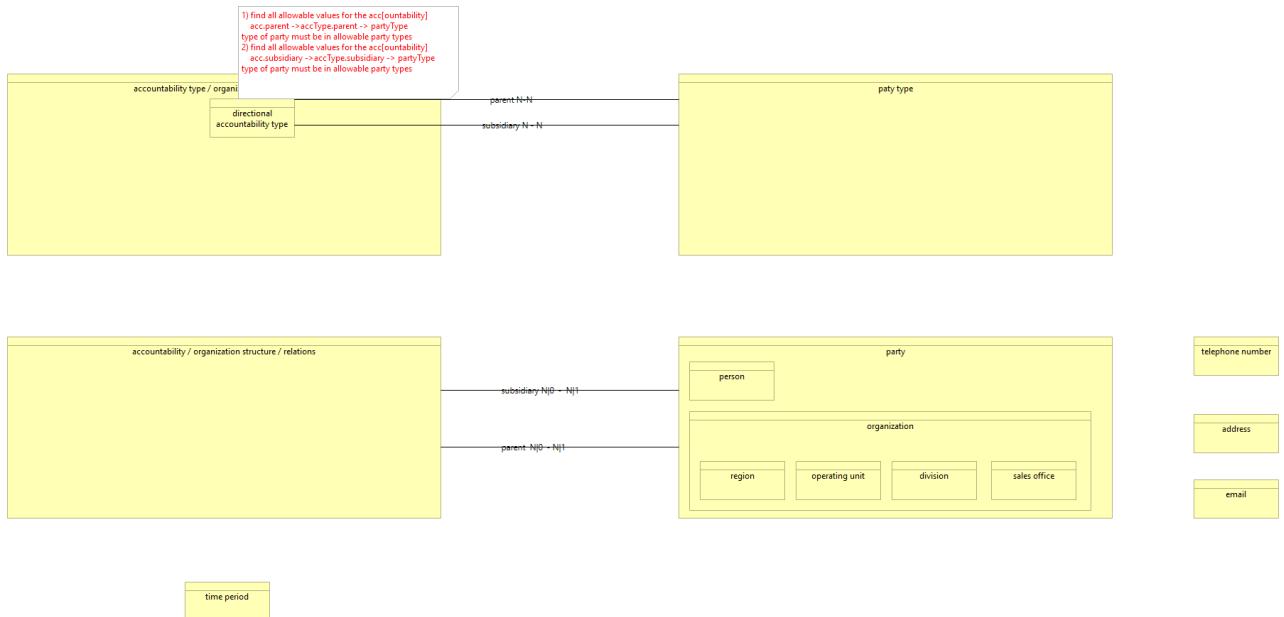
# ORGANIZATION STRUCTURE



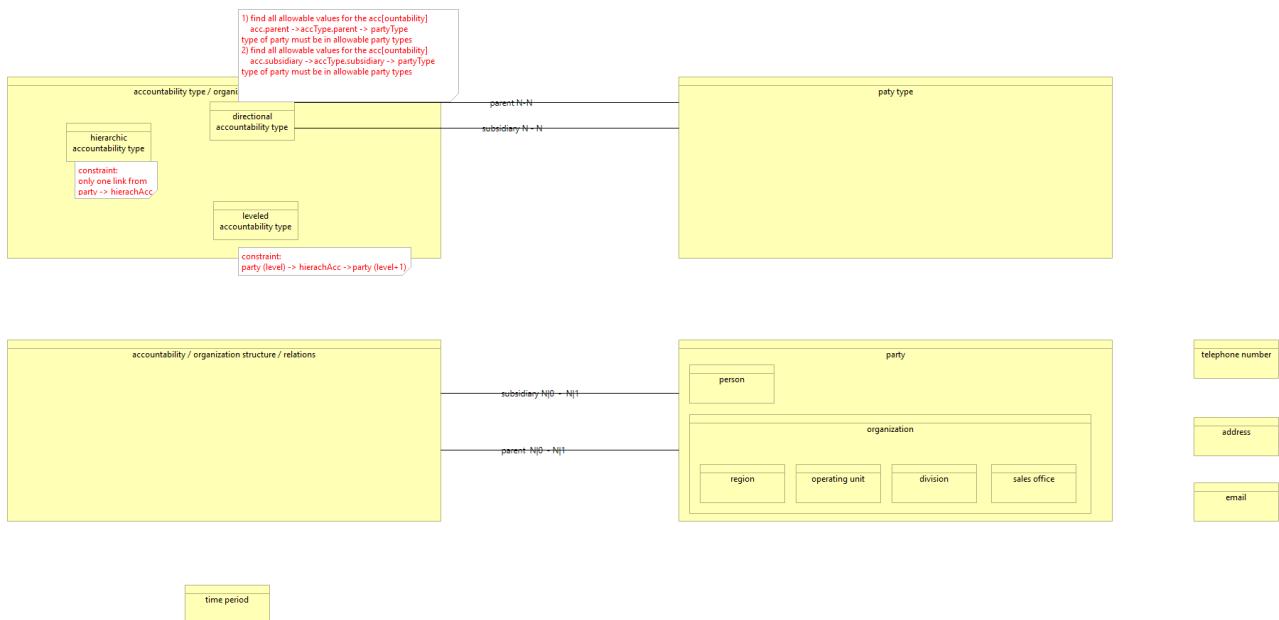
# ACCOUNTABILITY KNOWLEDGE LEVEL



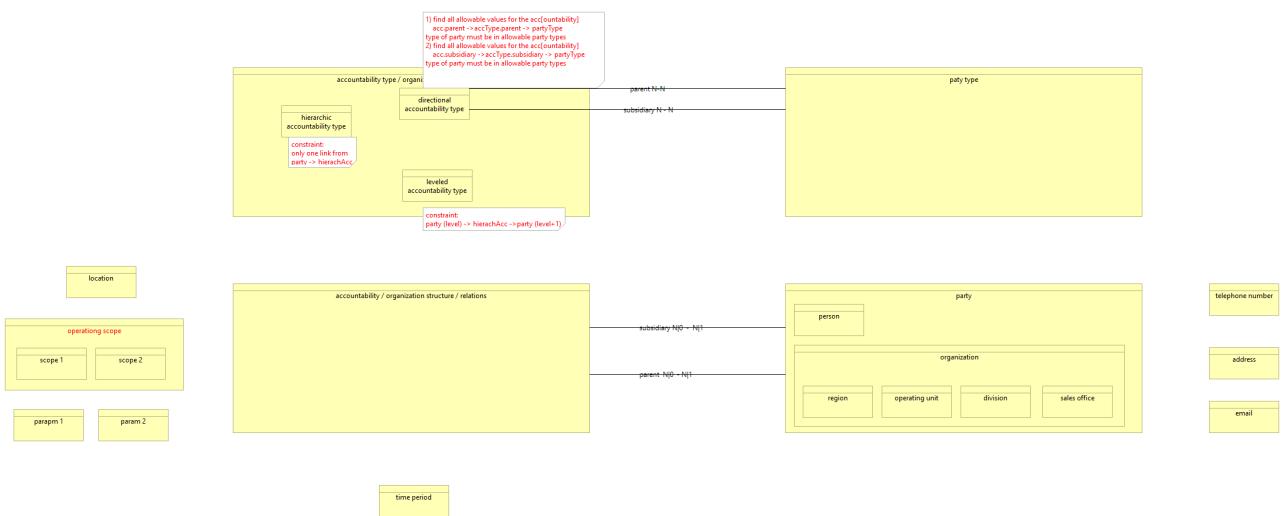
# PARTY TYPE GENERALIZATIONS



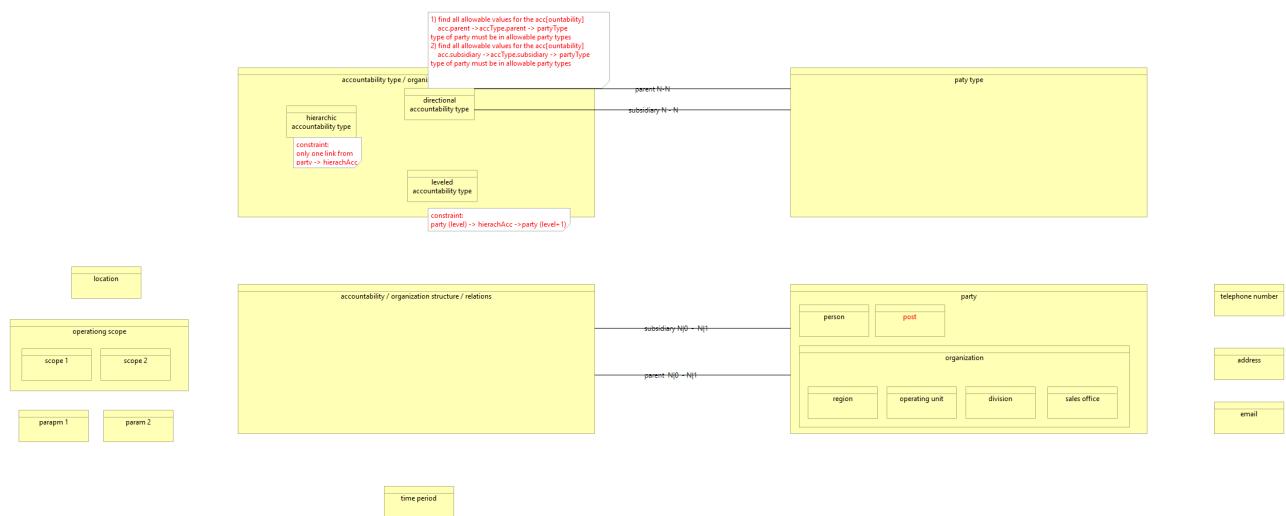
# HIERARCHIC ACCOUNTABILITY



# OPERATING SCOPES



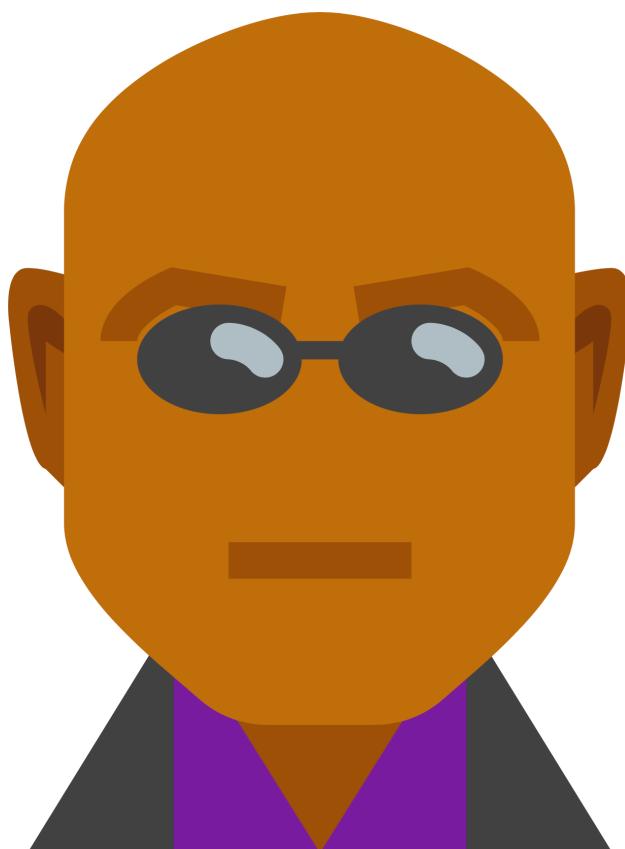
# POST



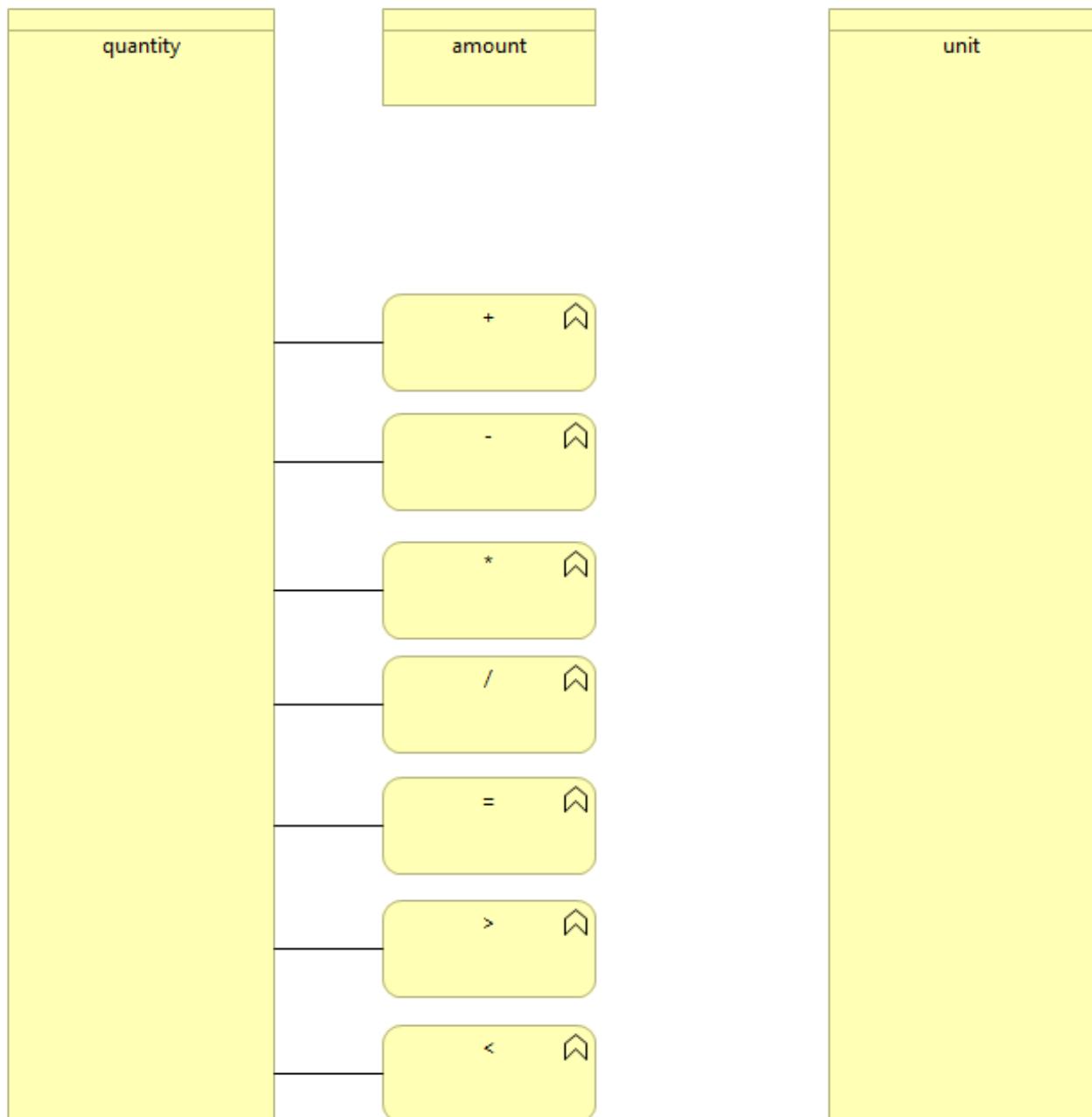
# OBSERVATIONS AND MEASUREMENTS

ANALYSIS PATTERNS

Martin Fowler



# QUANTITY

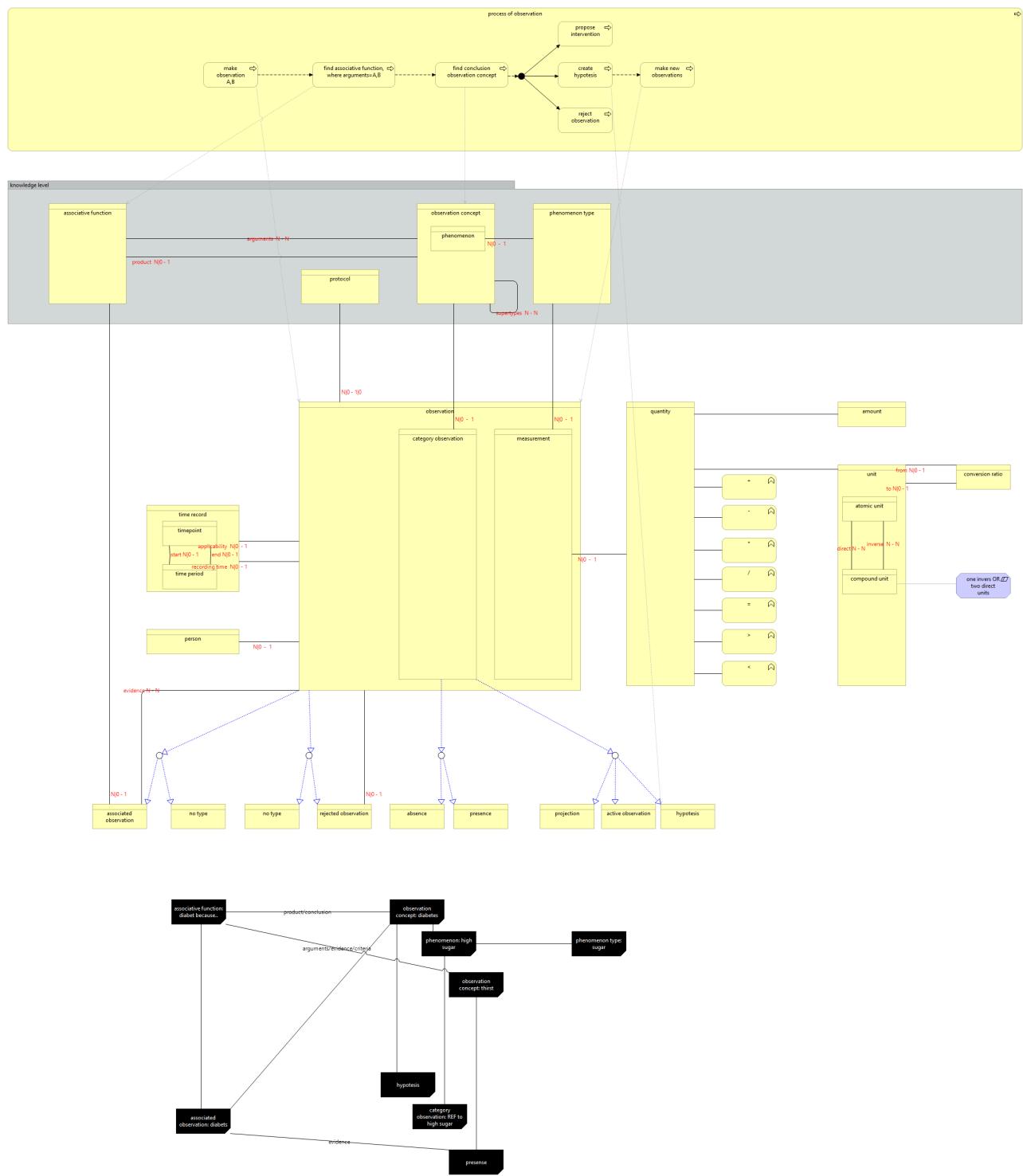


# CONVERSION RATIO

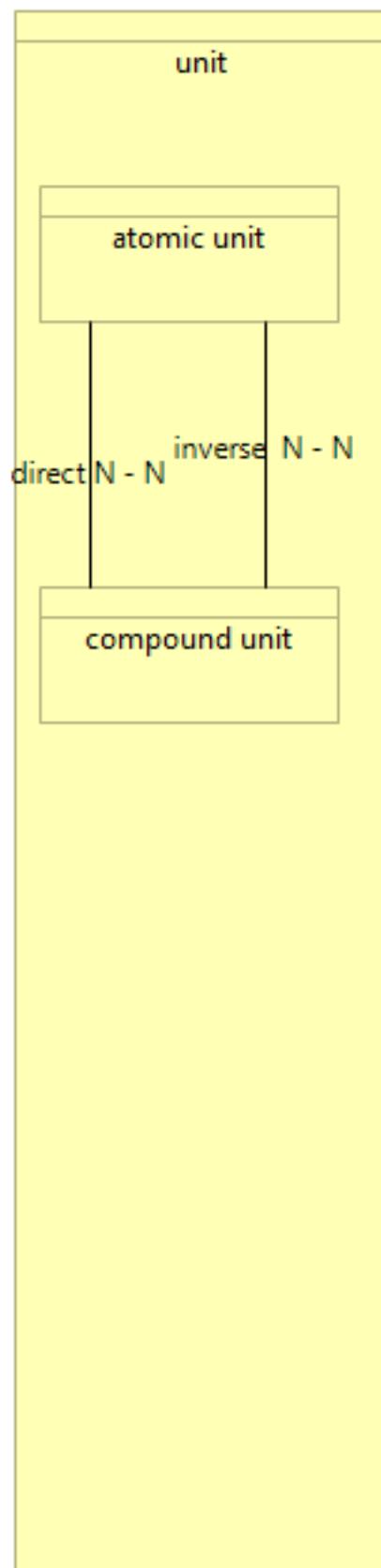
unit

conversion ratio

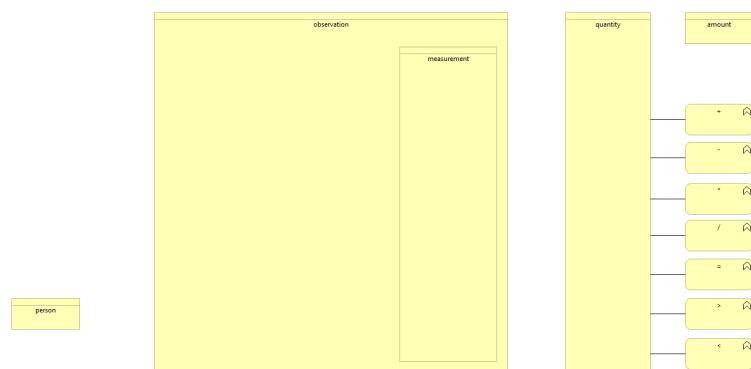
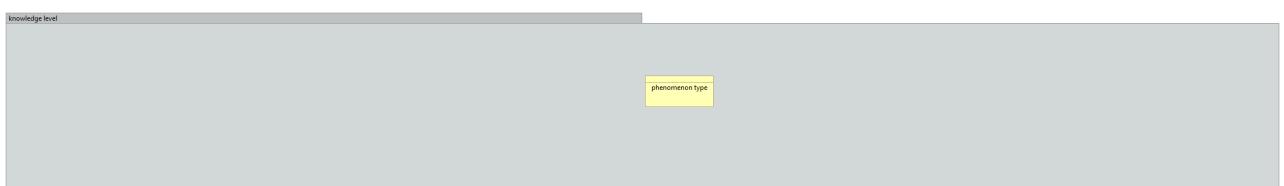
# OBSERVATIONS AND MEASUREMENTS



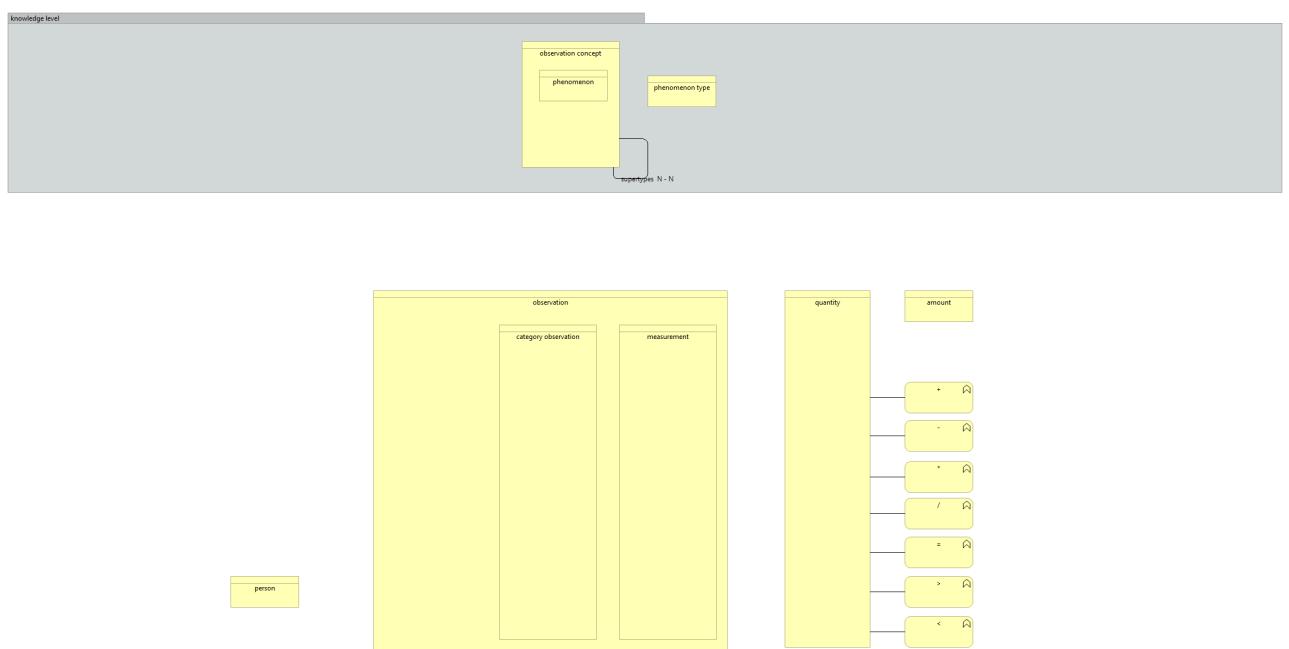
# COMPOUND UNITS



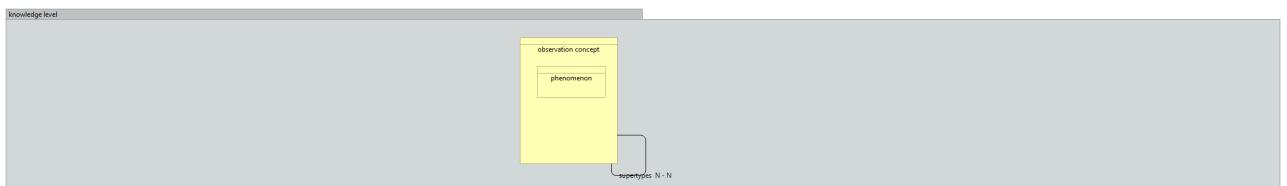
# MEASUREMENT



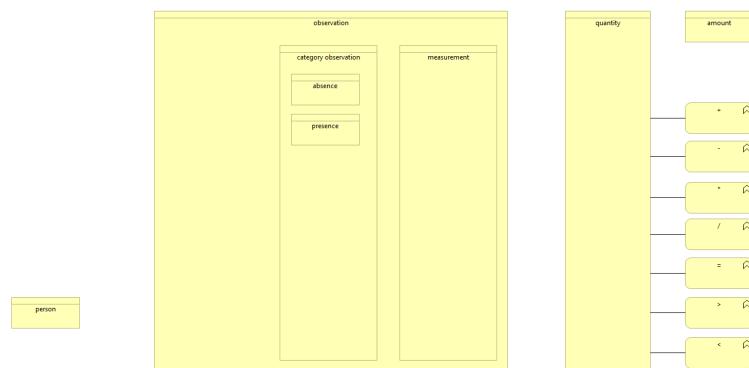
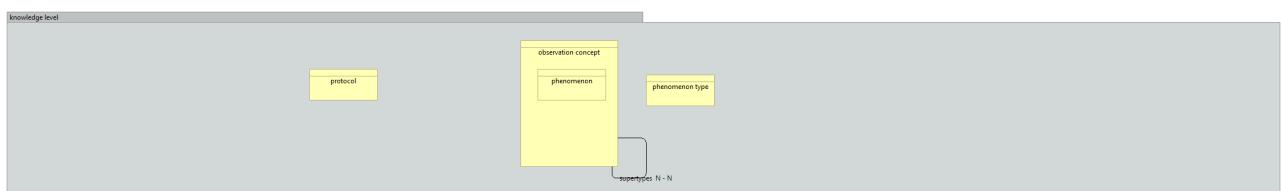
# OBSERVATION



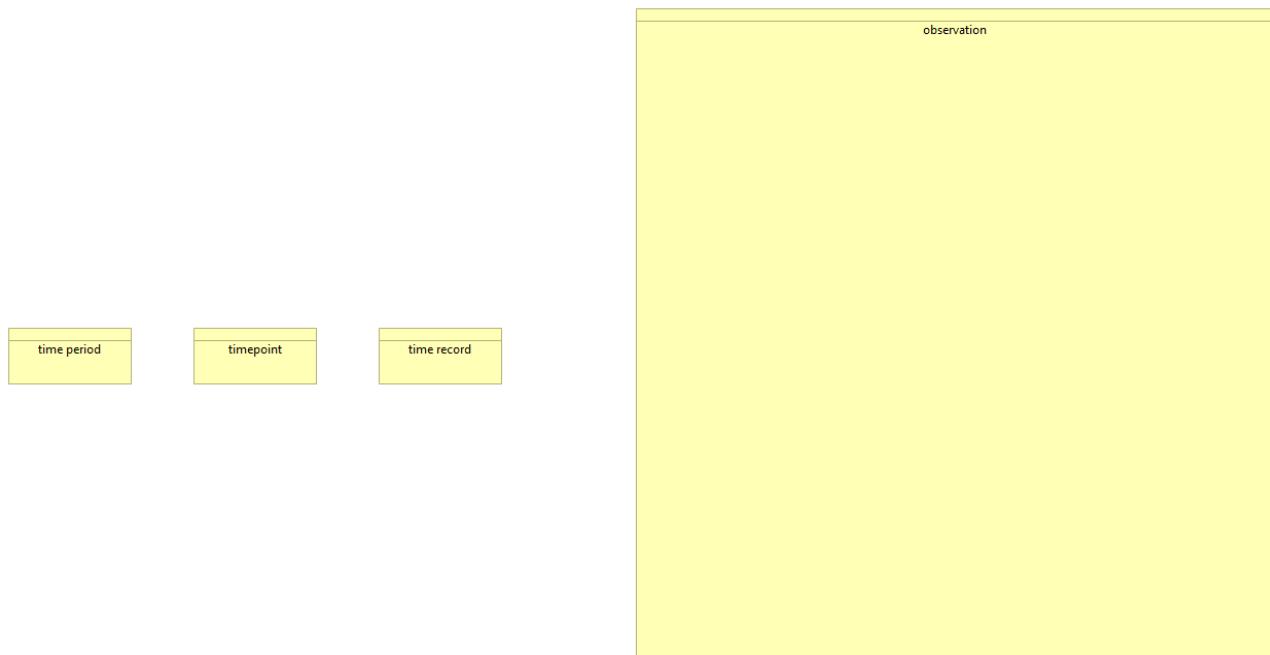
# SUBTYPING OBSERVATION CONCEPTS



# PROTOCOL



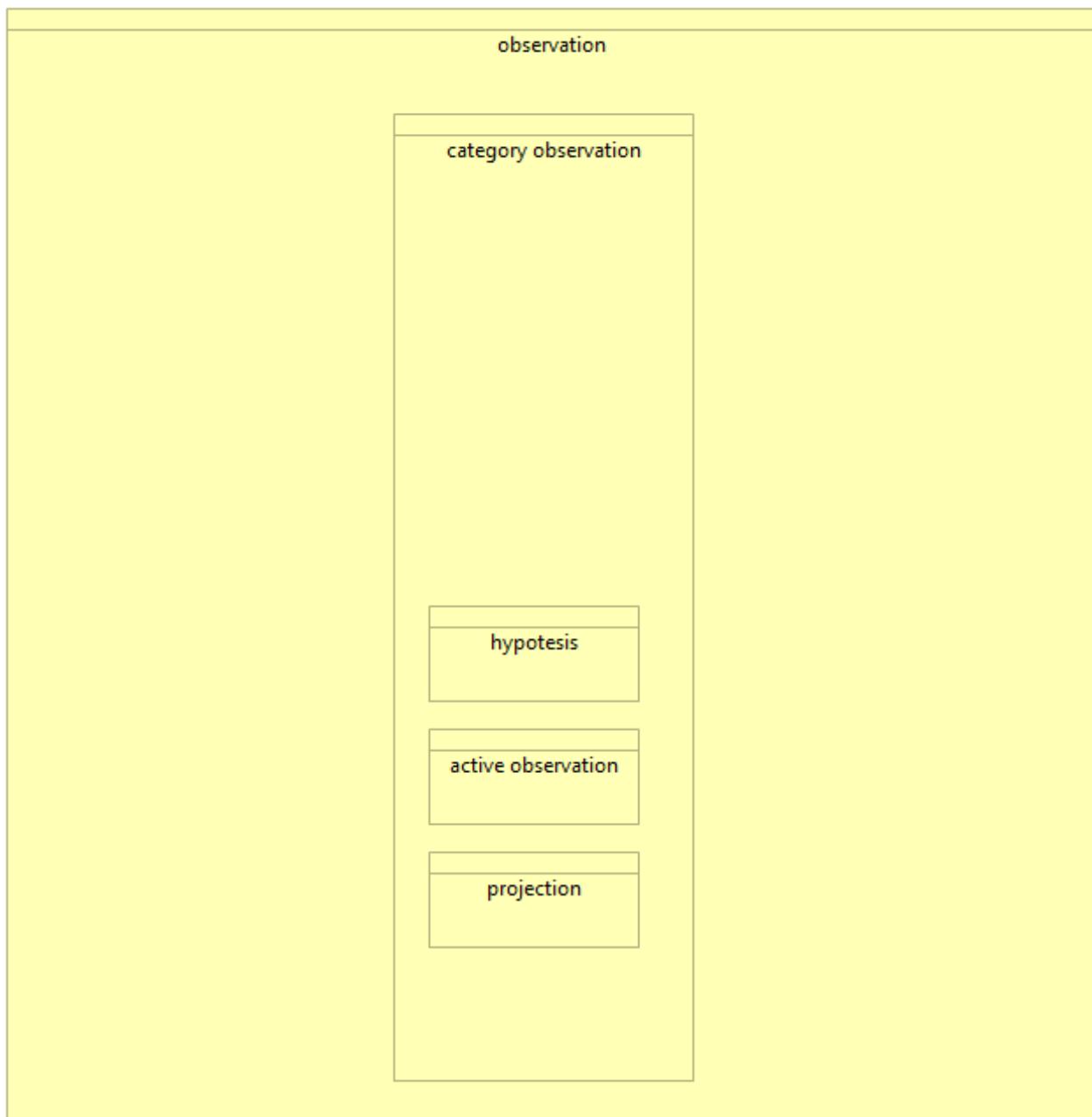
# DUAL TIME RECORD



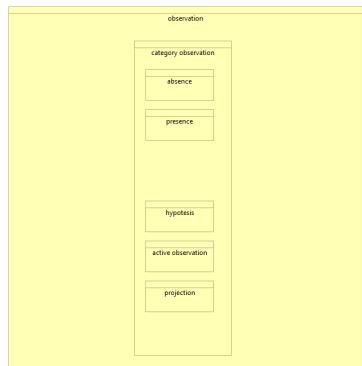
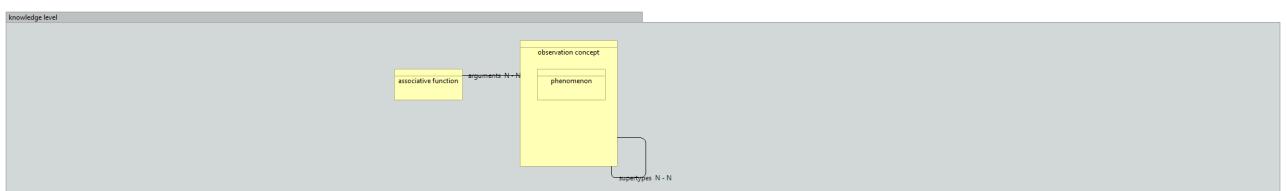
# REJECTED OBSERVATION

observation

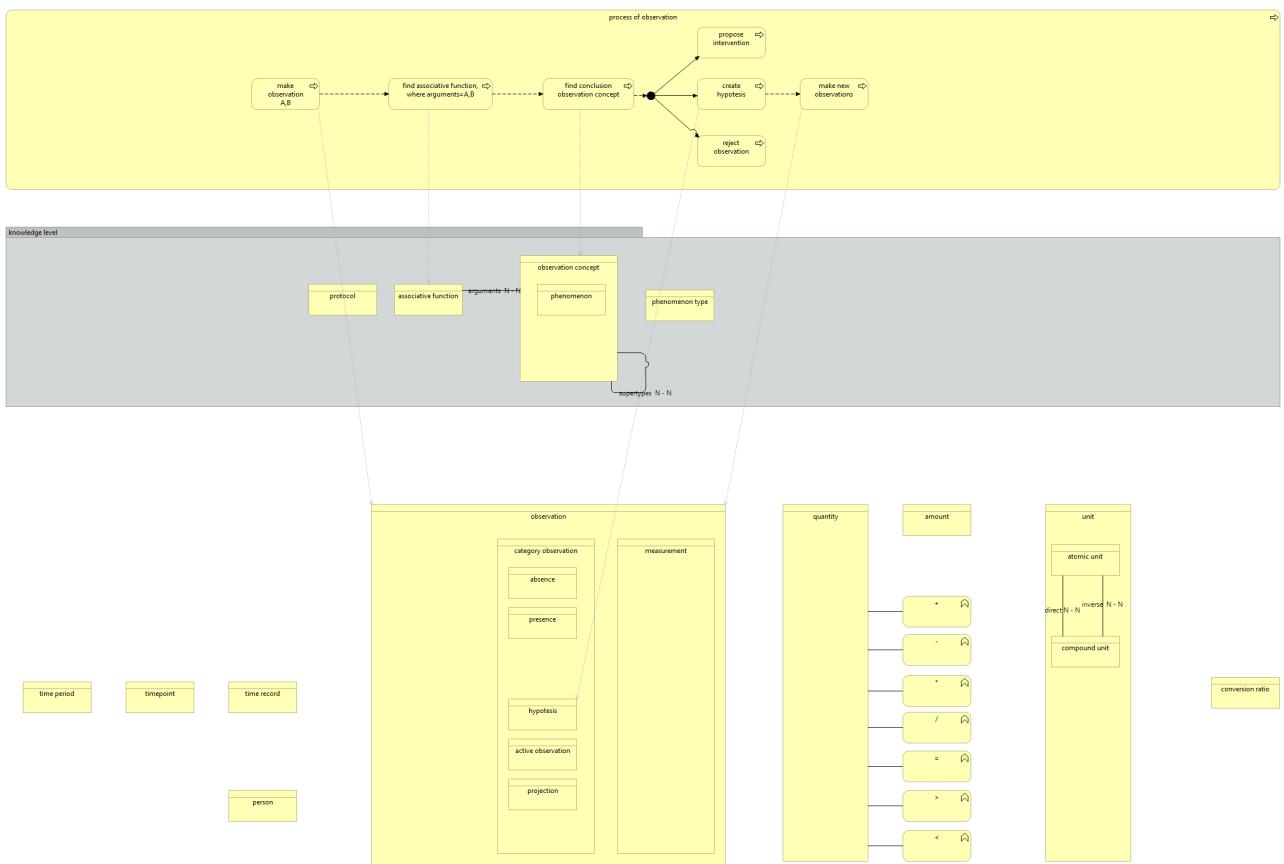
# ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION



# ASSOCIATED OBSERVATION



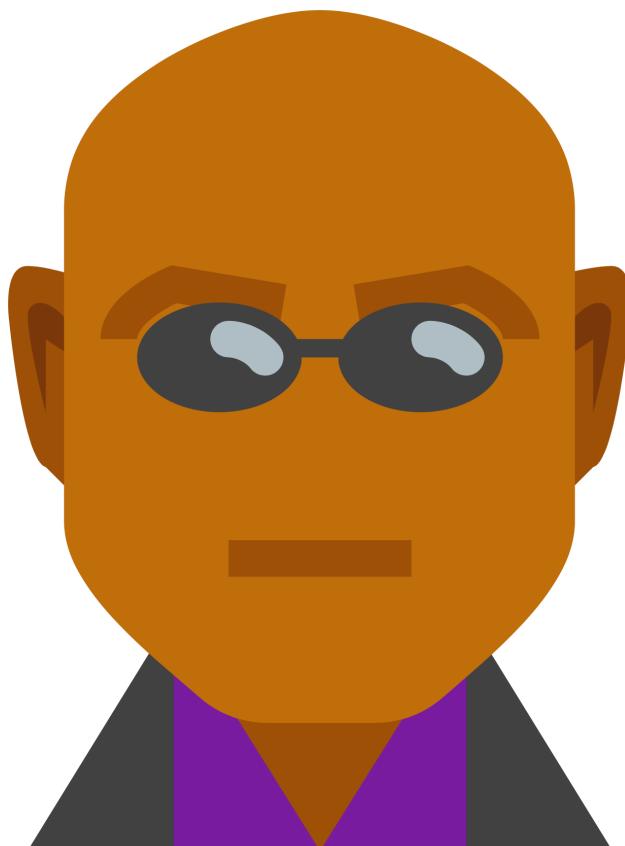
# PROCESS OF OBSERVATION



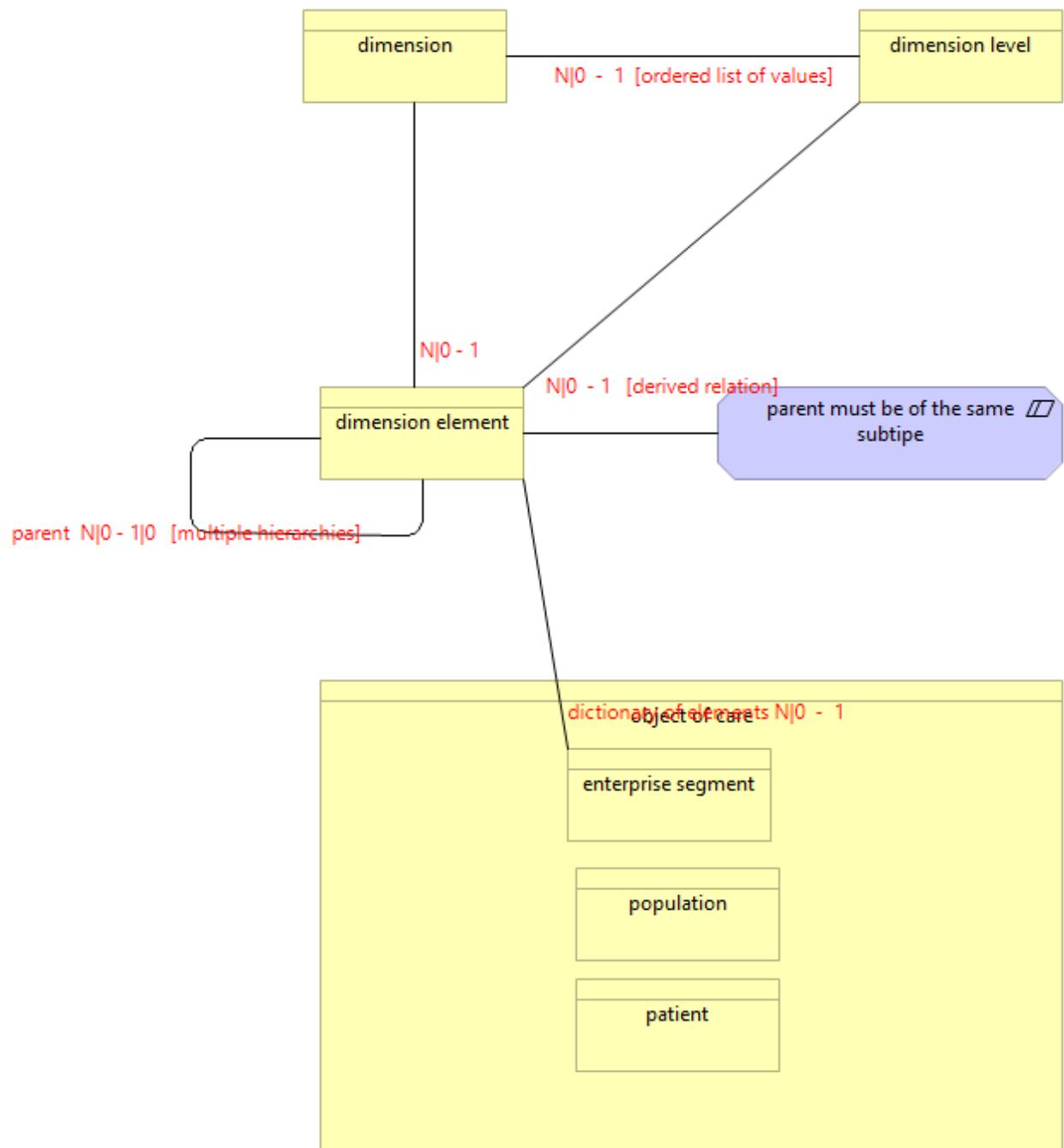
# OBSERVATIONS FOR CORPORATE FINANCE

ANALYSIS PATTERNS

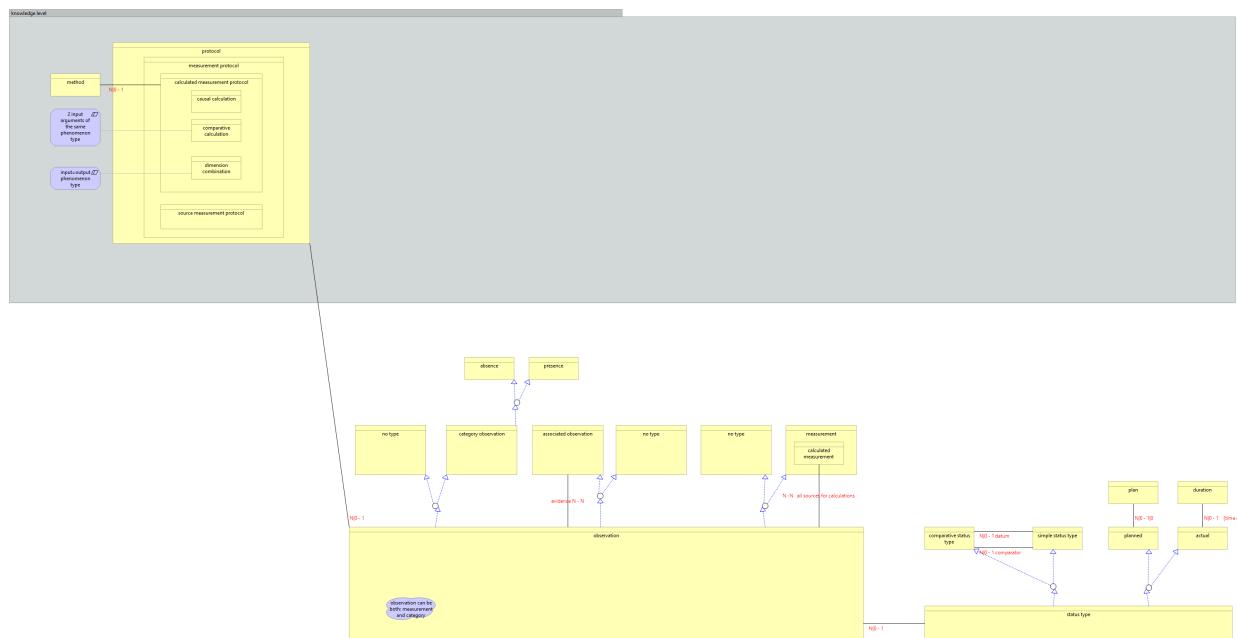
Martin Fowler



# ENTERPRISE SEGMENT



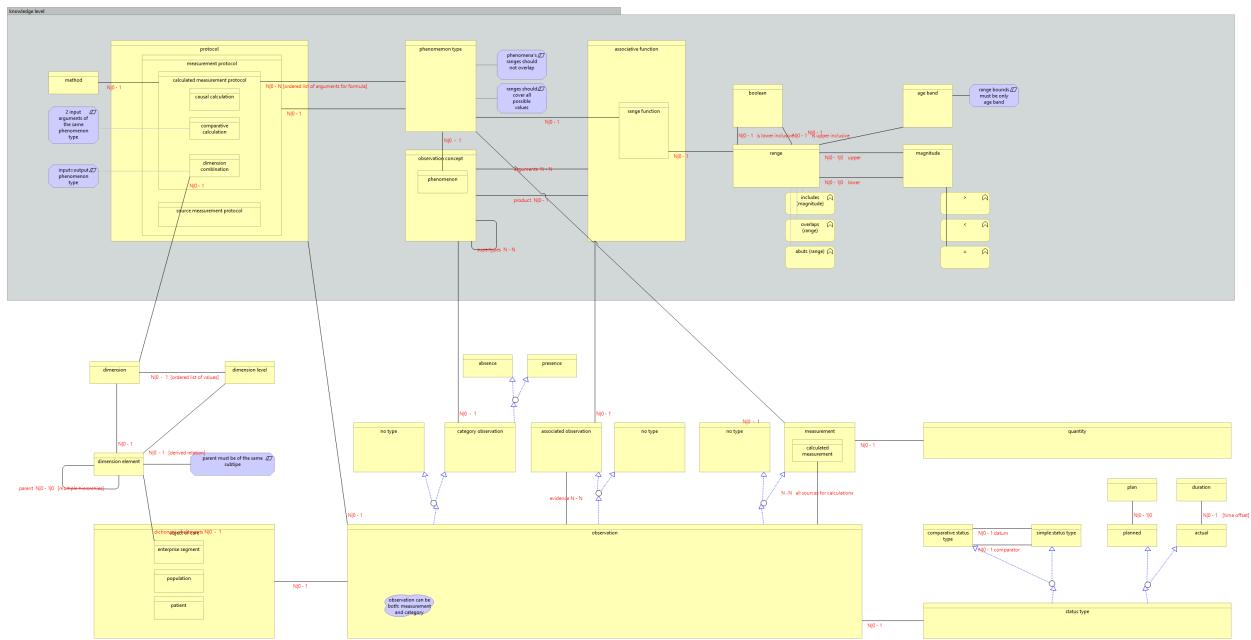
# MEASUREMENT PROTOCOL

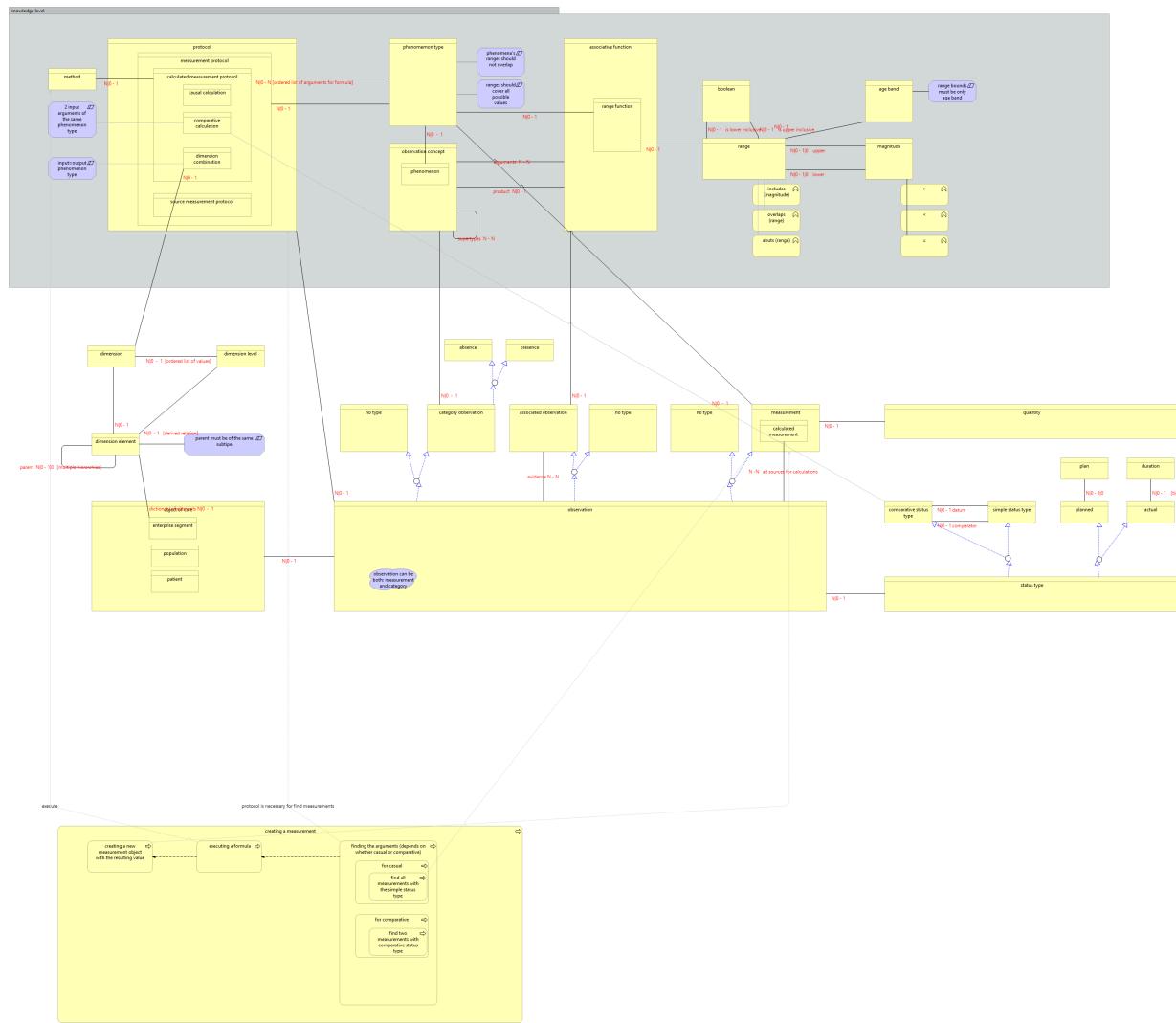


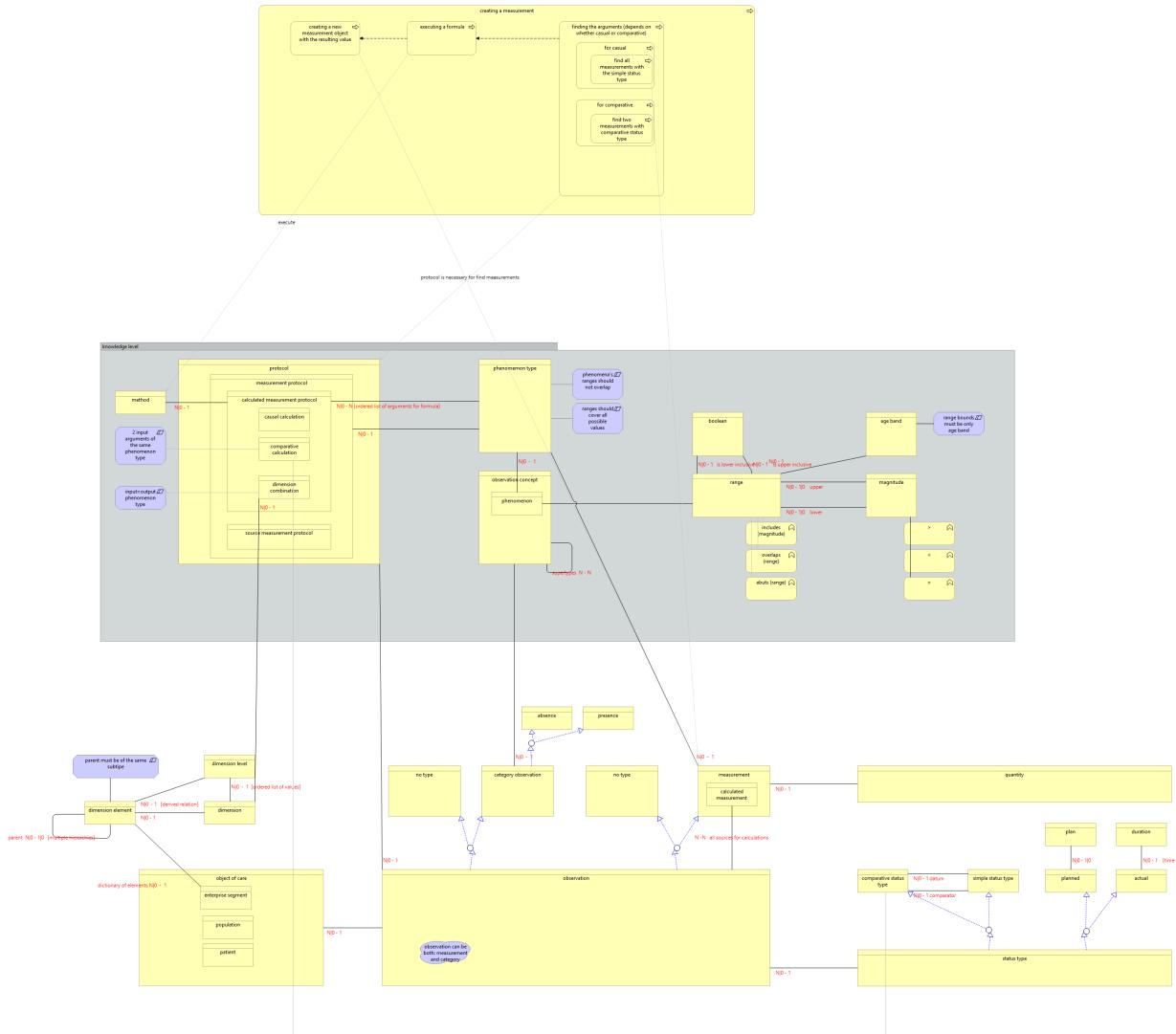
# RANGE



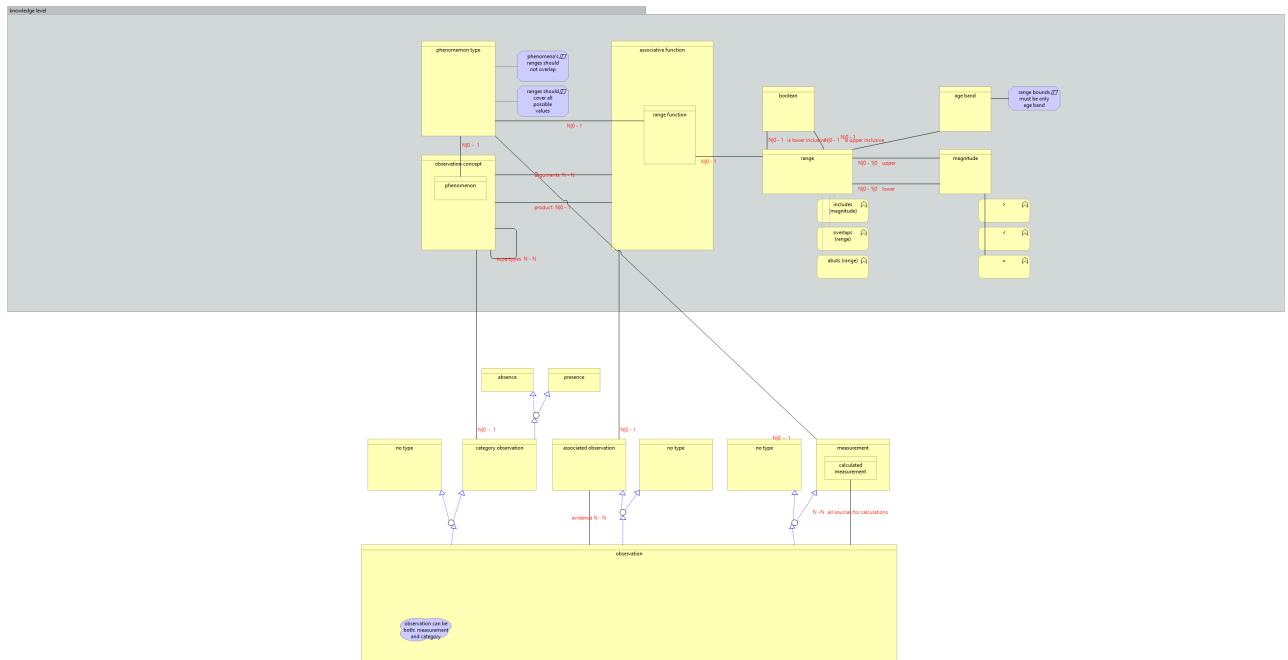
# OBSERVATIONS FOR CORPORATE FINANCE







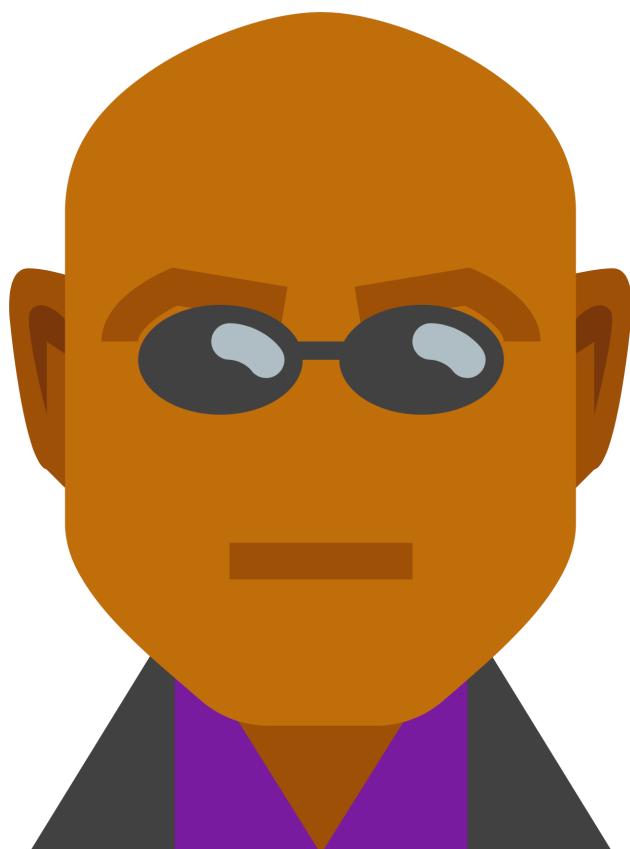
# PHENOMENON WITH RANGE



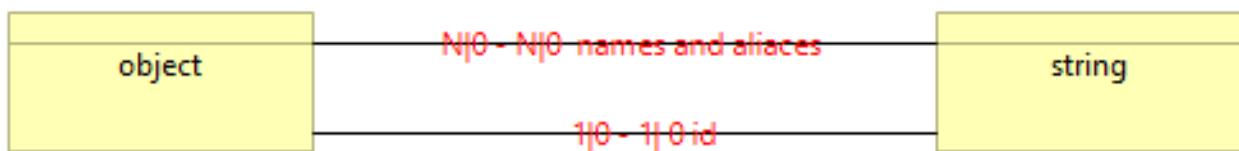
# REFERRING TO OBJECTS

ANALYSIS PATTERNS

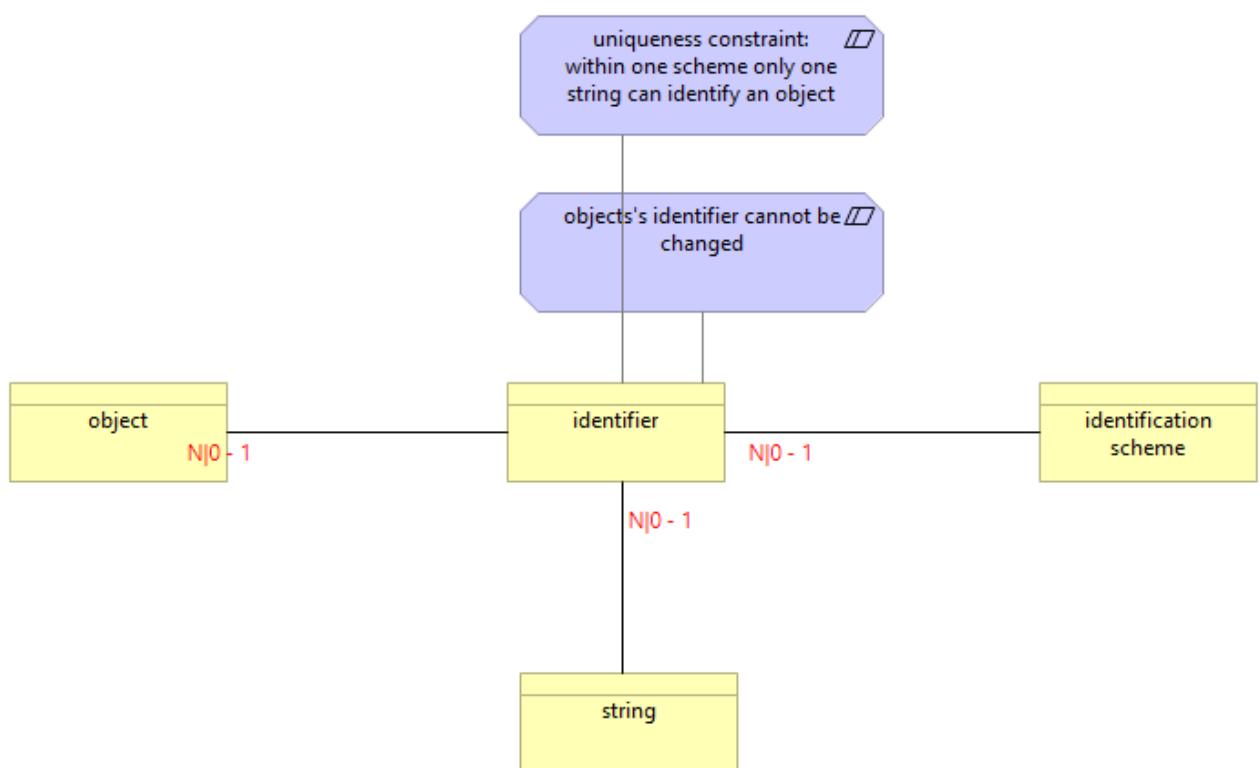
Martin Fowler



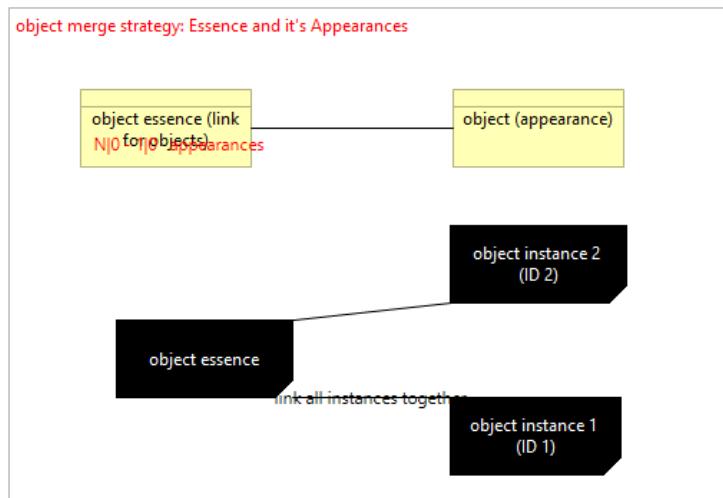
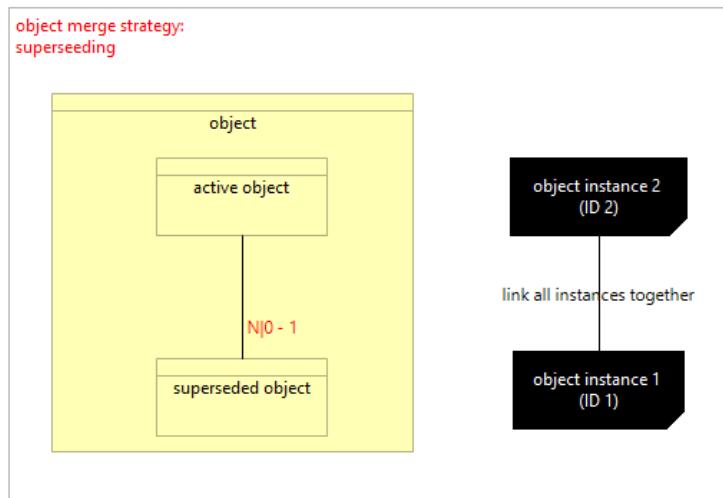
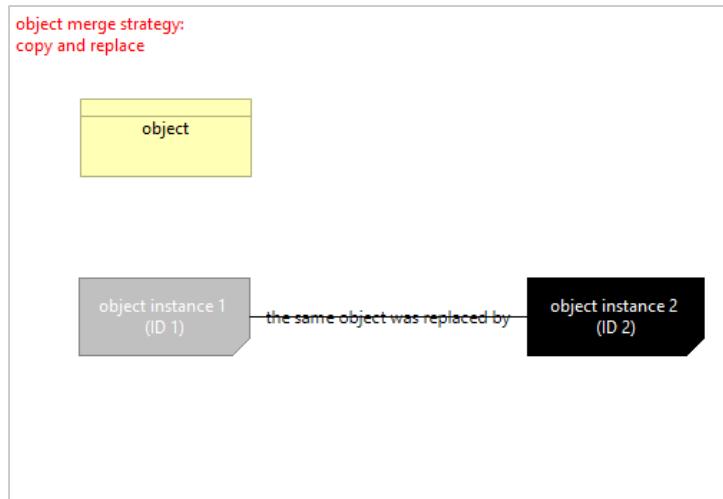
# NAME



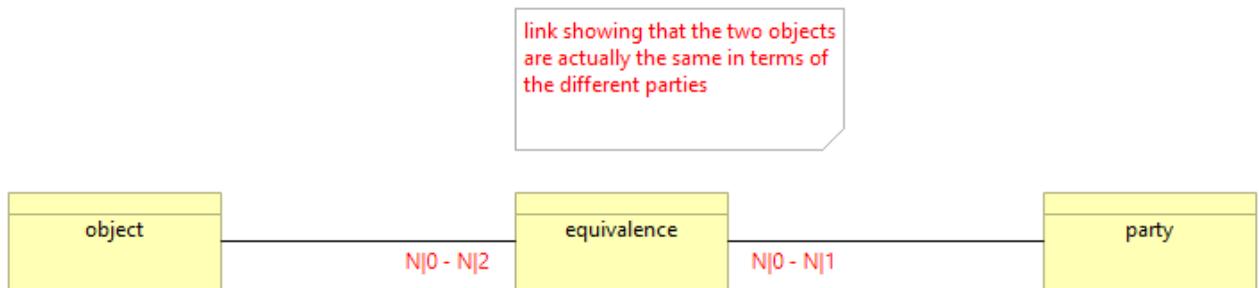
# IDENTIFICATION SCHEME



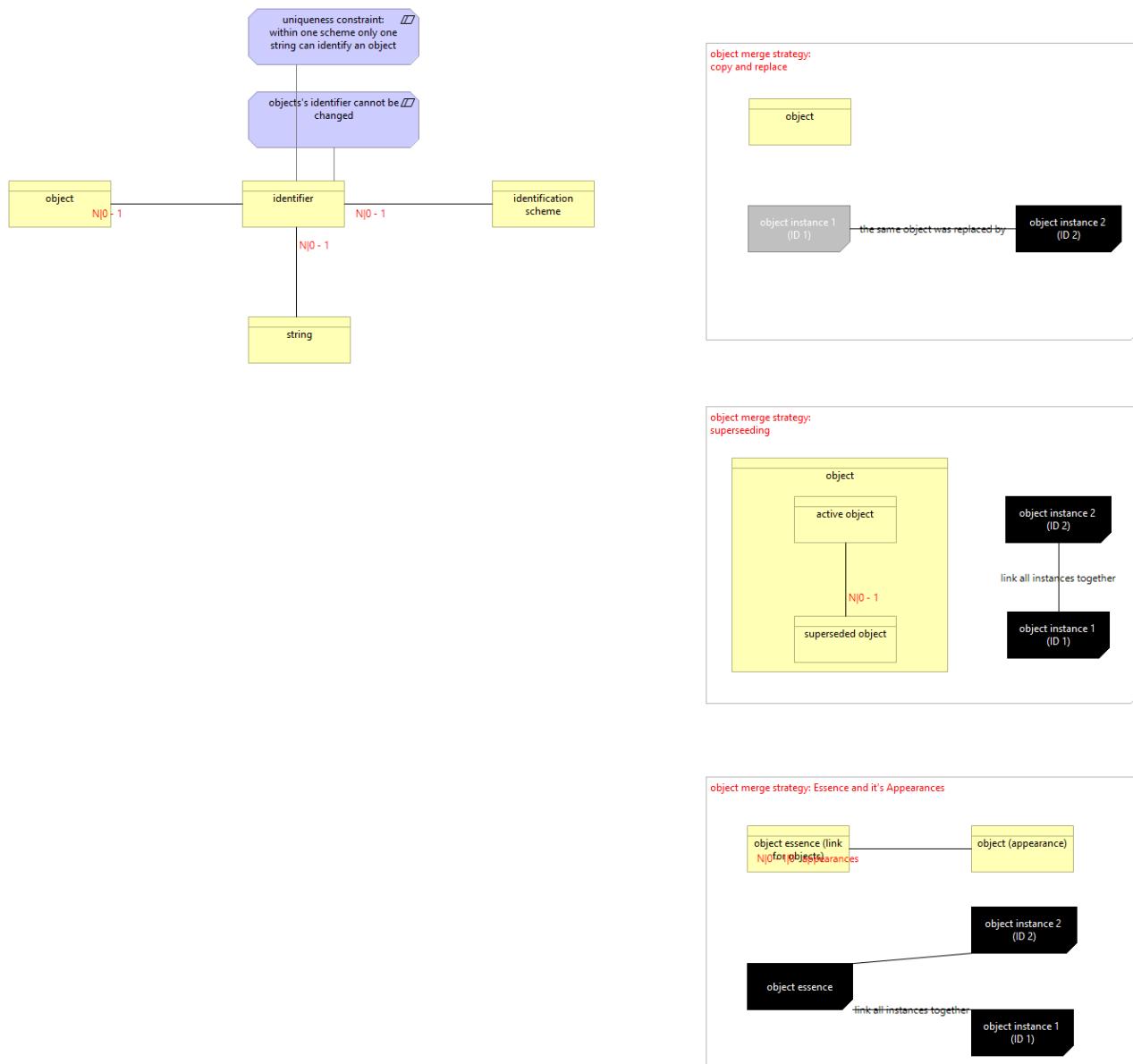
# OBJECT MERGE



# OBJECT EQUIVALENCE



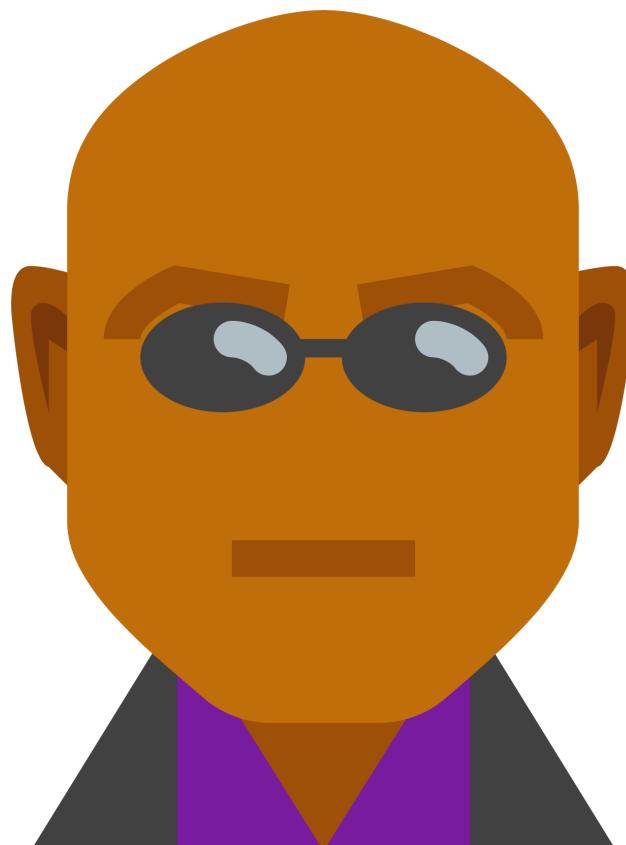
# REFERRING TO OBJECTS



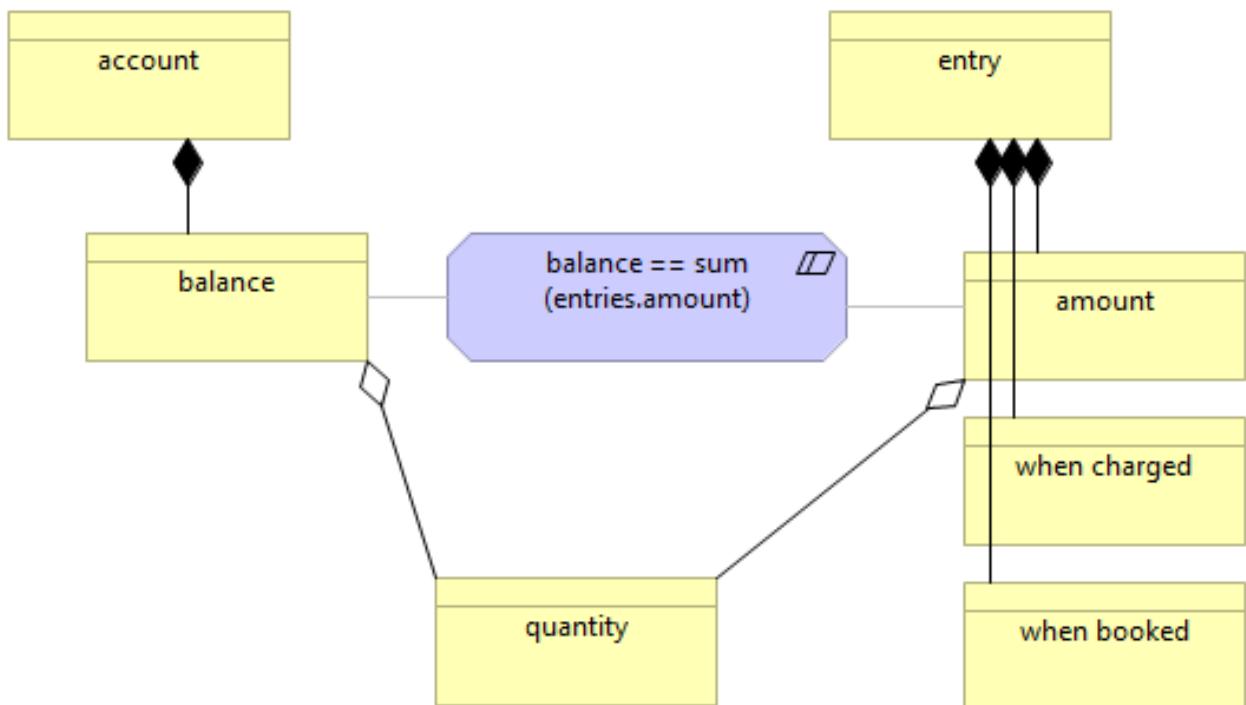
# INVENTORY AND ACCOUNTING

ANALYSIS PATTERNS

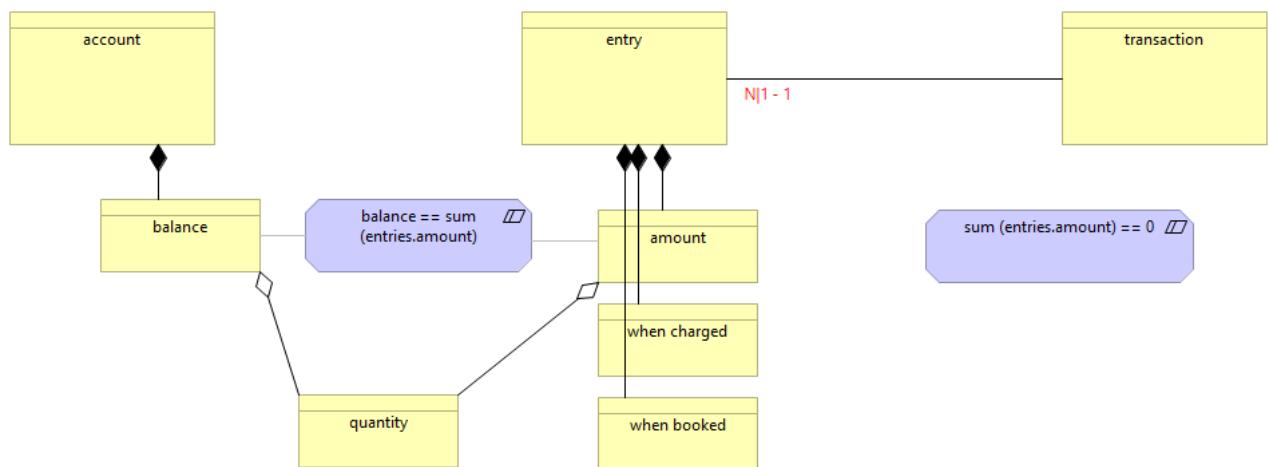
Martin Fowler



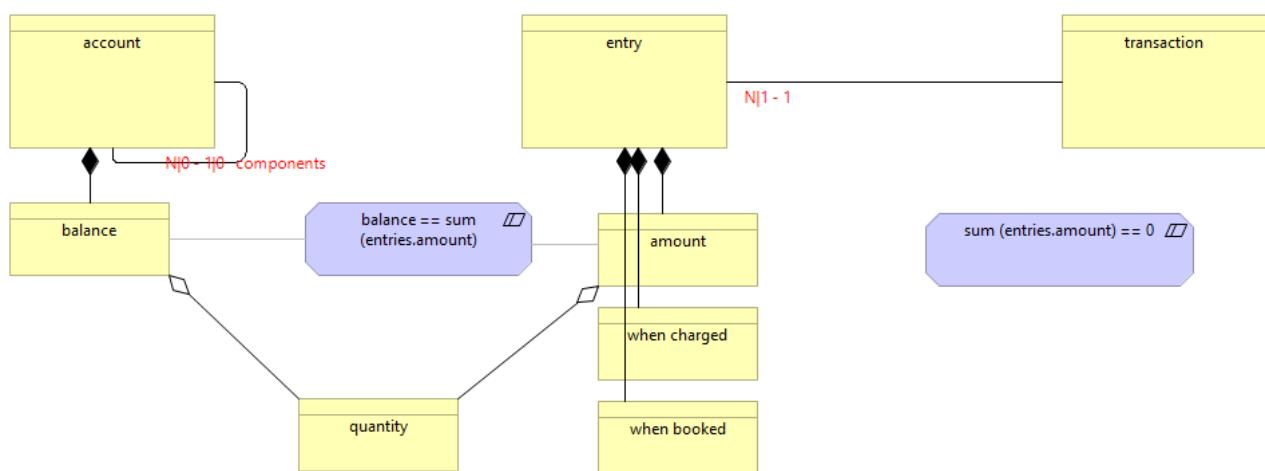
# ACCOUNT



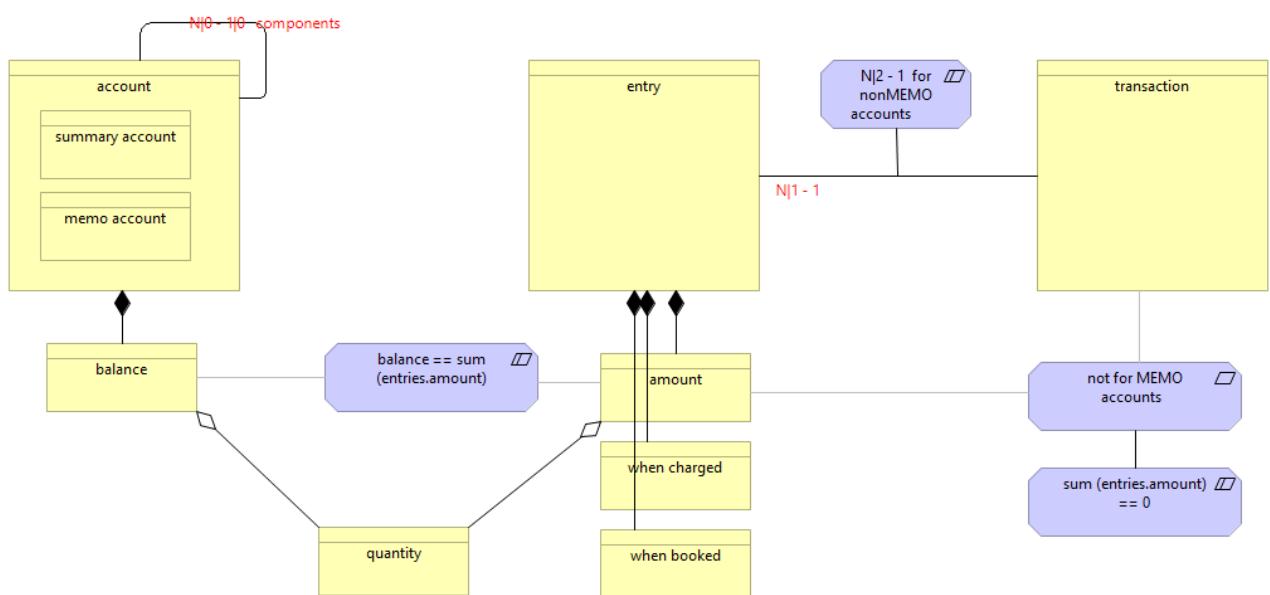
# TRANSACTIONS



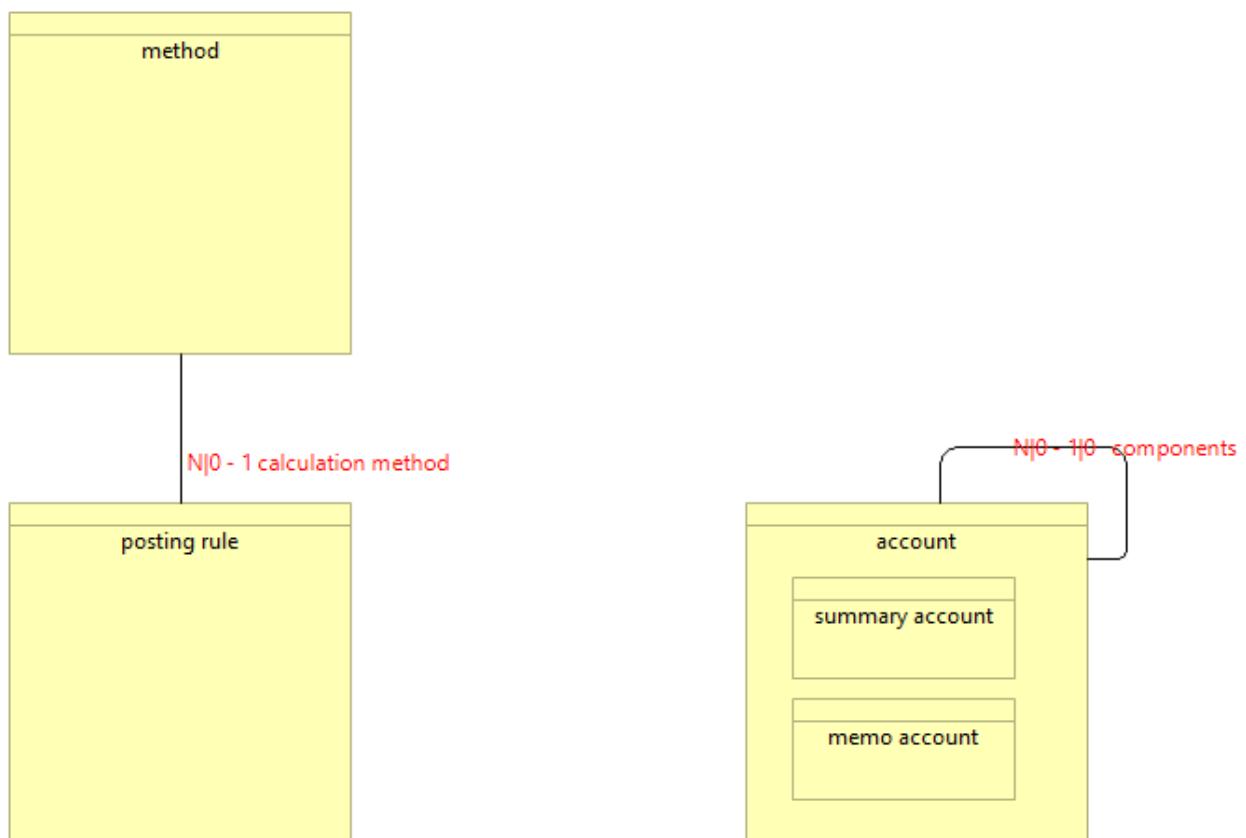
# SUMMARY ACCOUNT



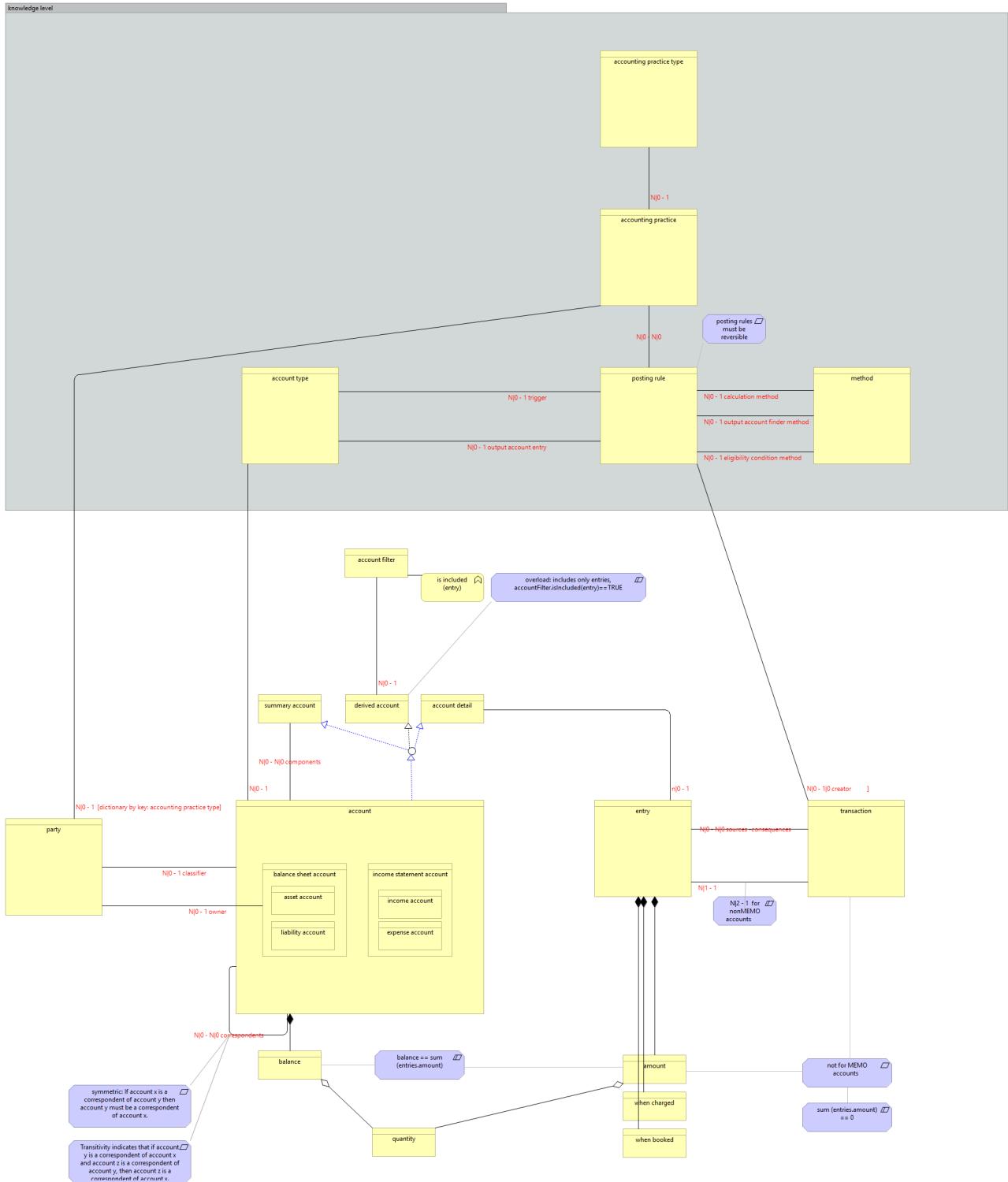
# MEMO ACCOUNT



# POSTING RULES



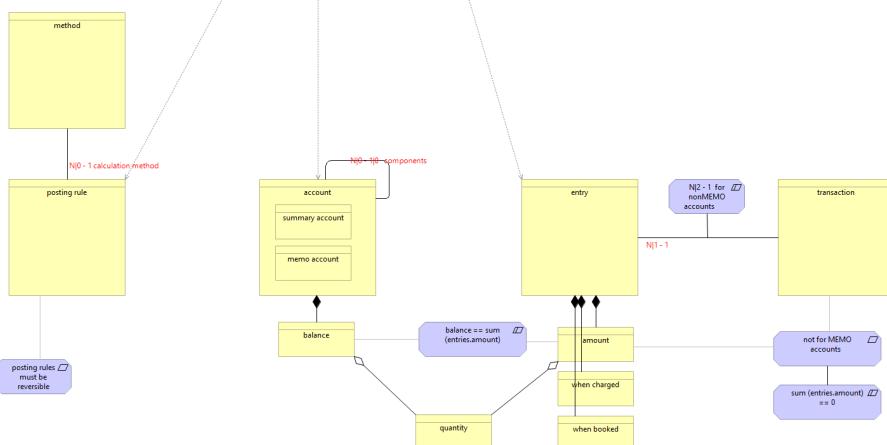
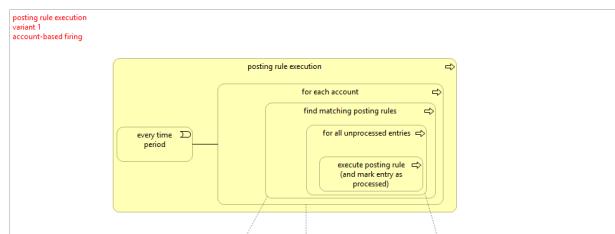
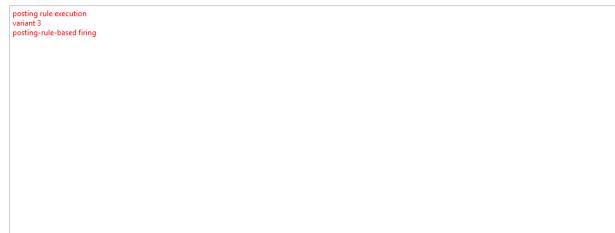
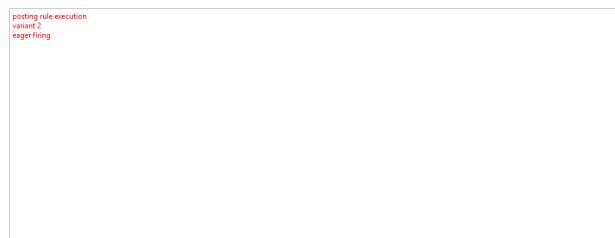
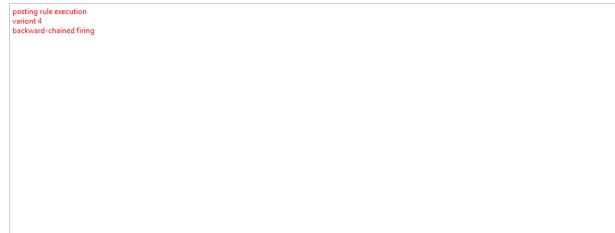
# INVENTORY AND ACCOUNTING



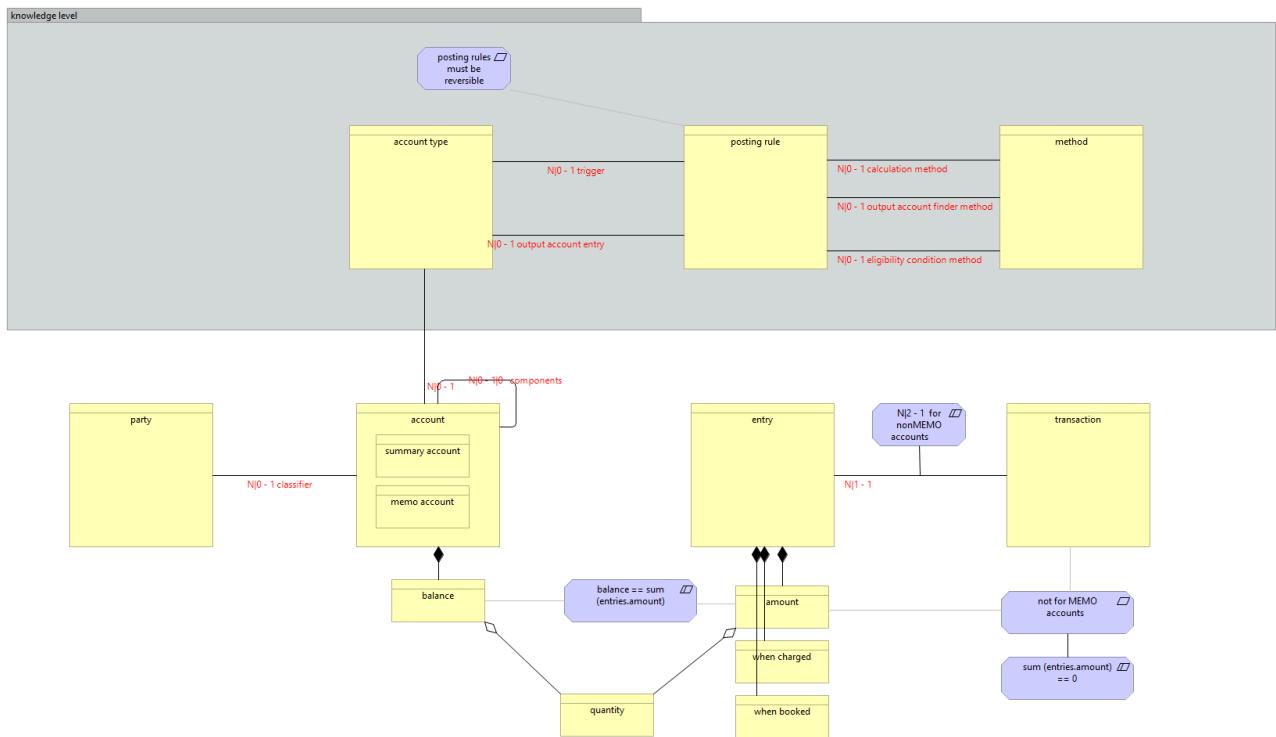
# INDIVIDUAL INSTANCE METHOD



# POSTING RULE EXECUTION



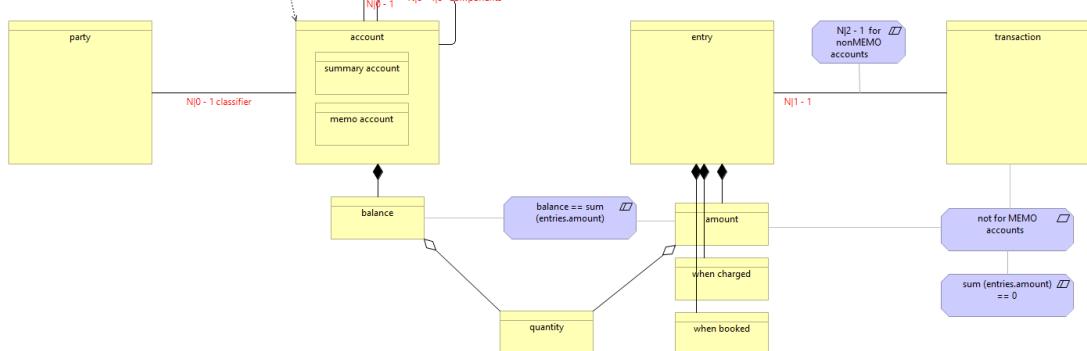
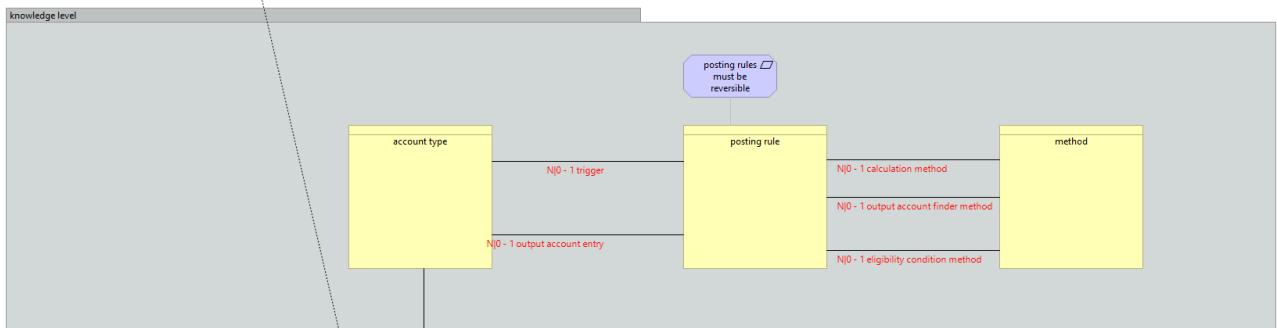
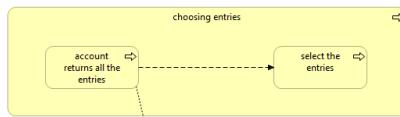
# **POSTING RULES FOR MANY ACCOUNTS**



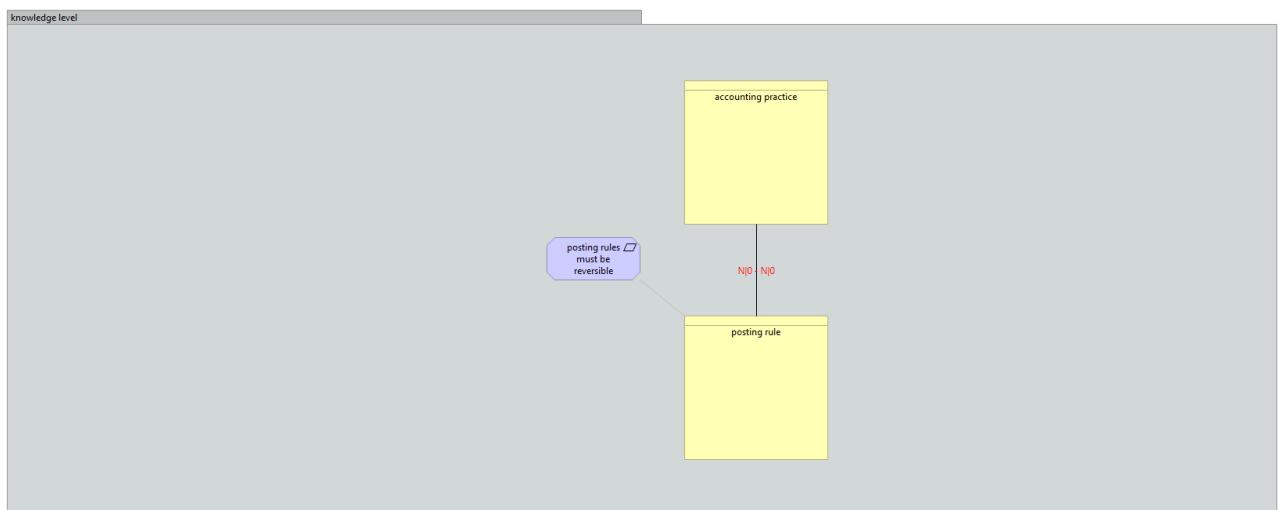
# CHOOSING ENTRIES

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

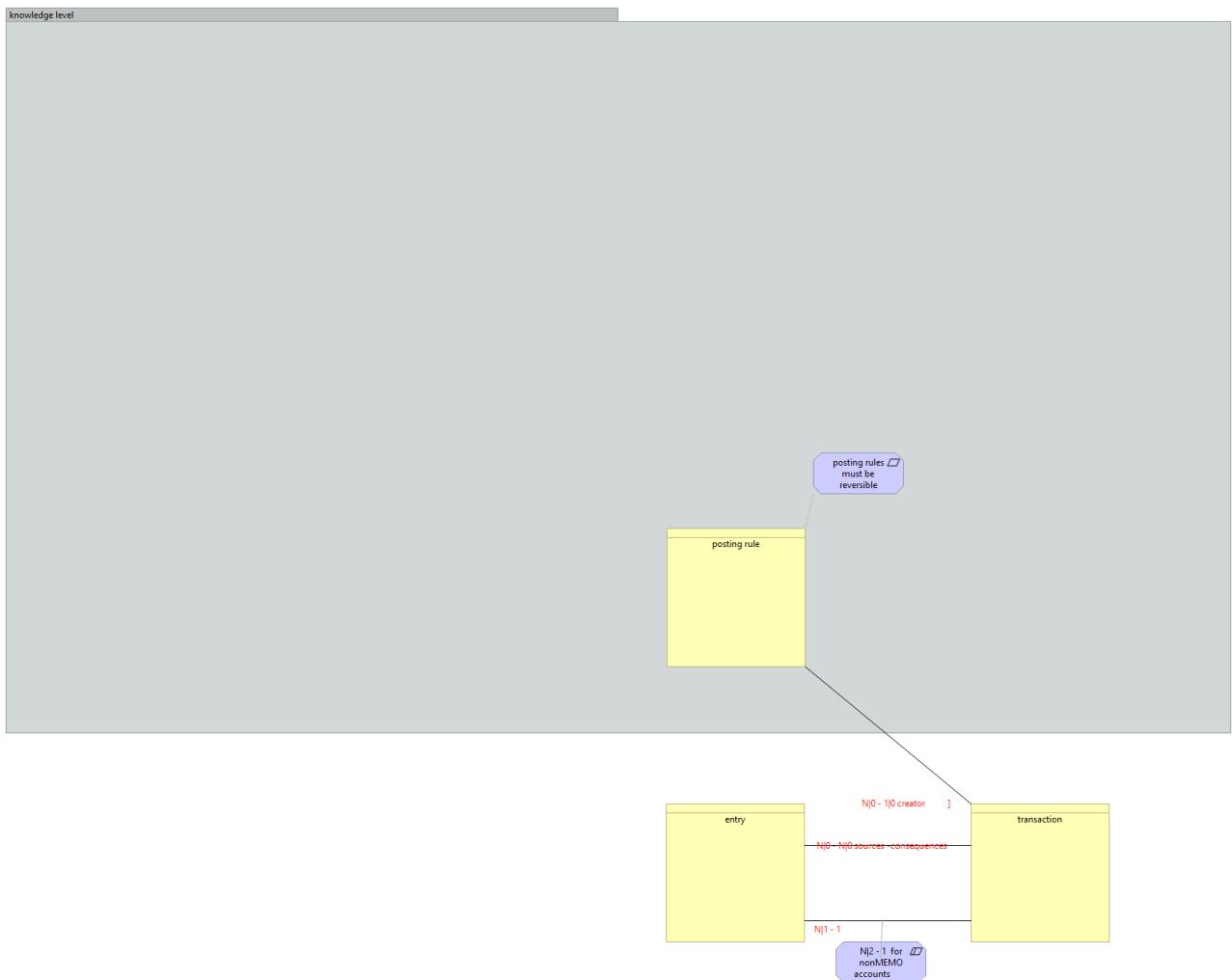
selection of entries for the application of the posing rules:  
variant 1  
The account returns all the entries, and the client selects the entries it needs.



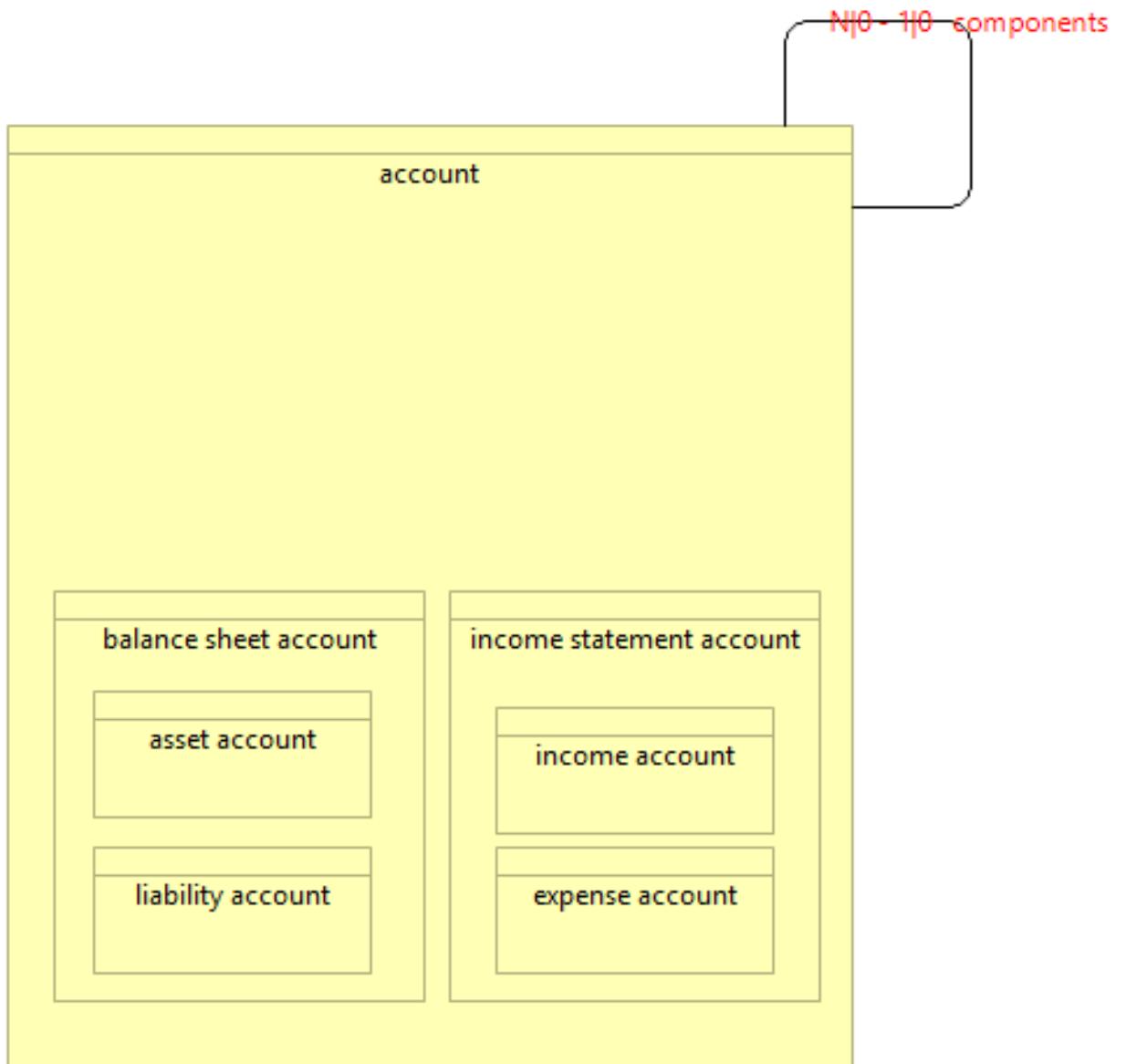
# ACCOUNTING PRACTICE



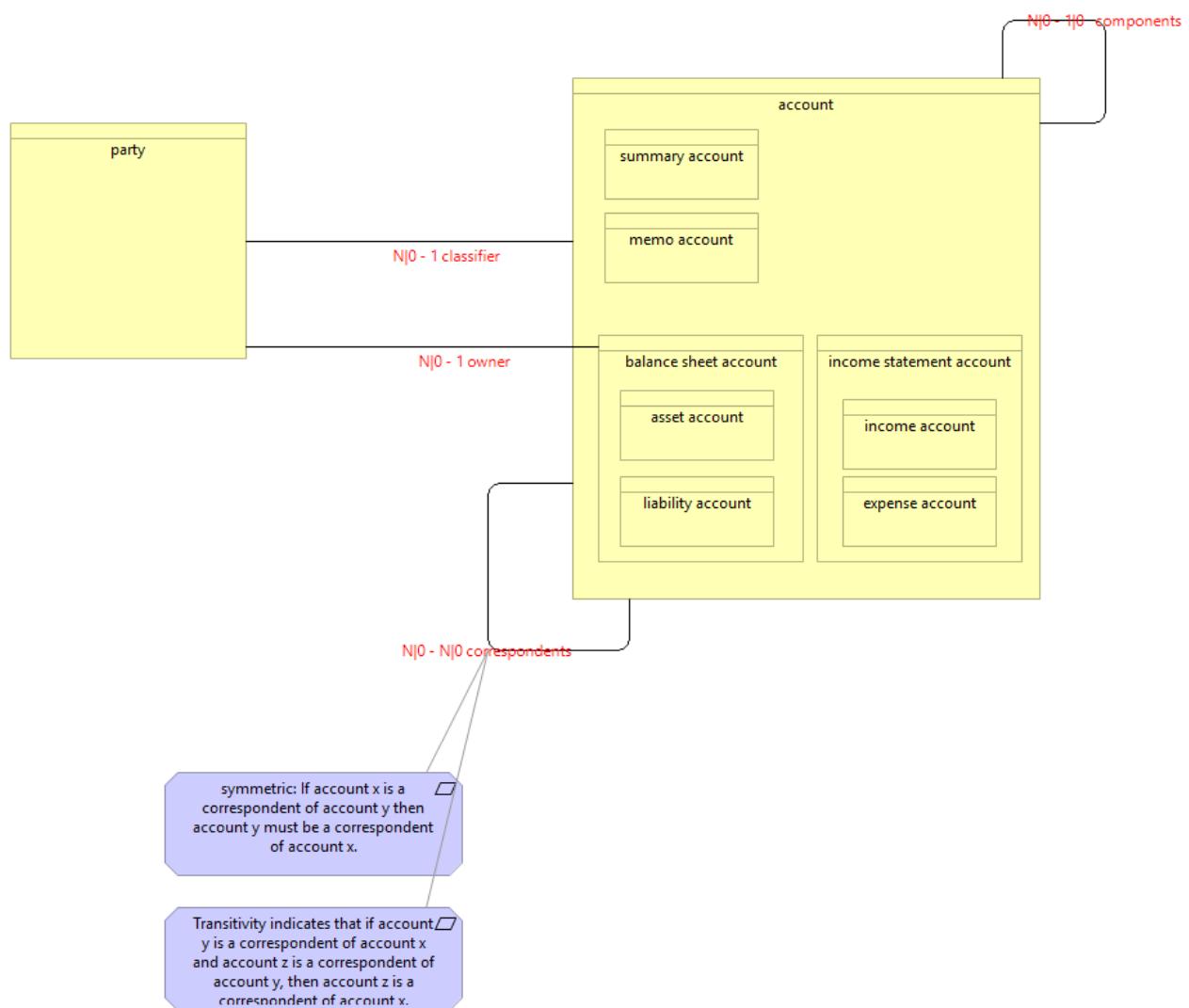
# SOURCES OF AN ENTRY



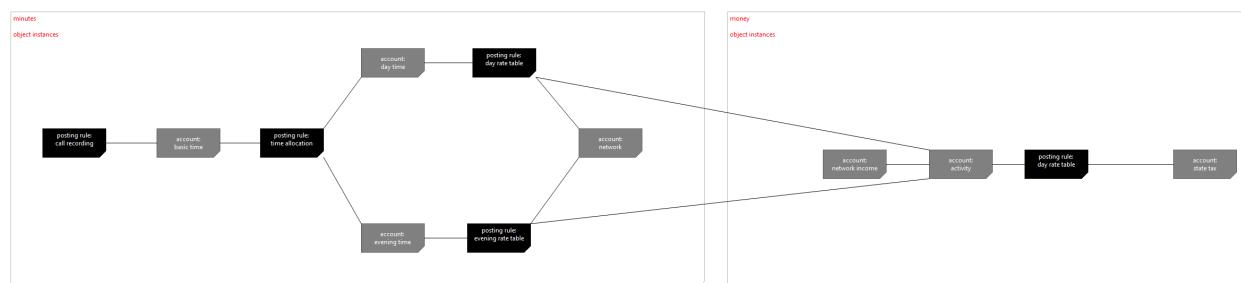
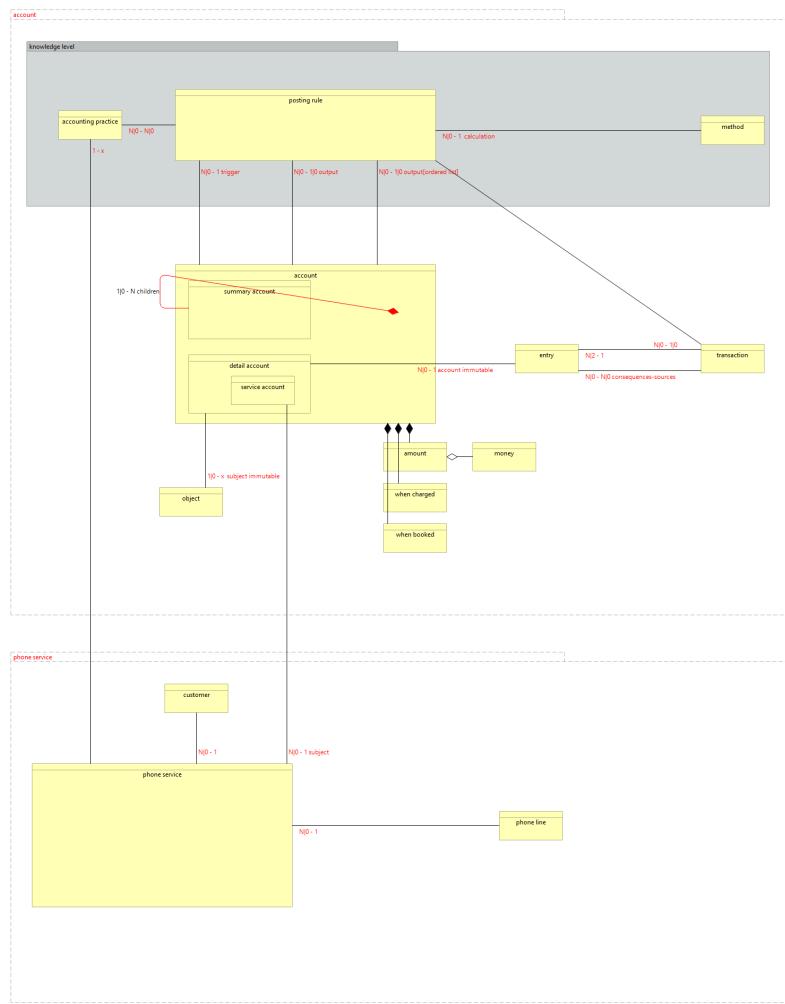
# BALANCE SHEET AND INCOME STATEMENT



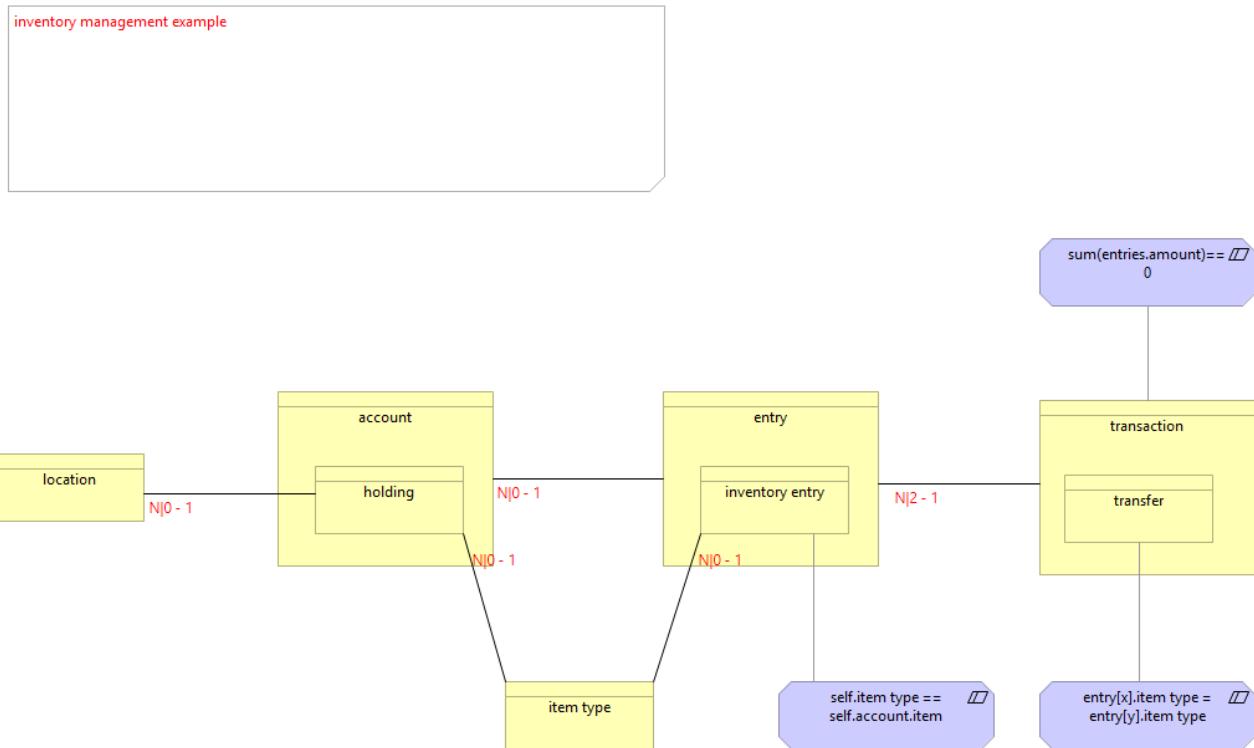
# CORRESPONDING ACCOUNT



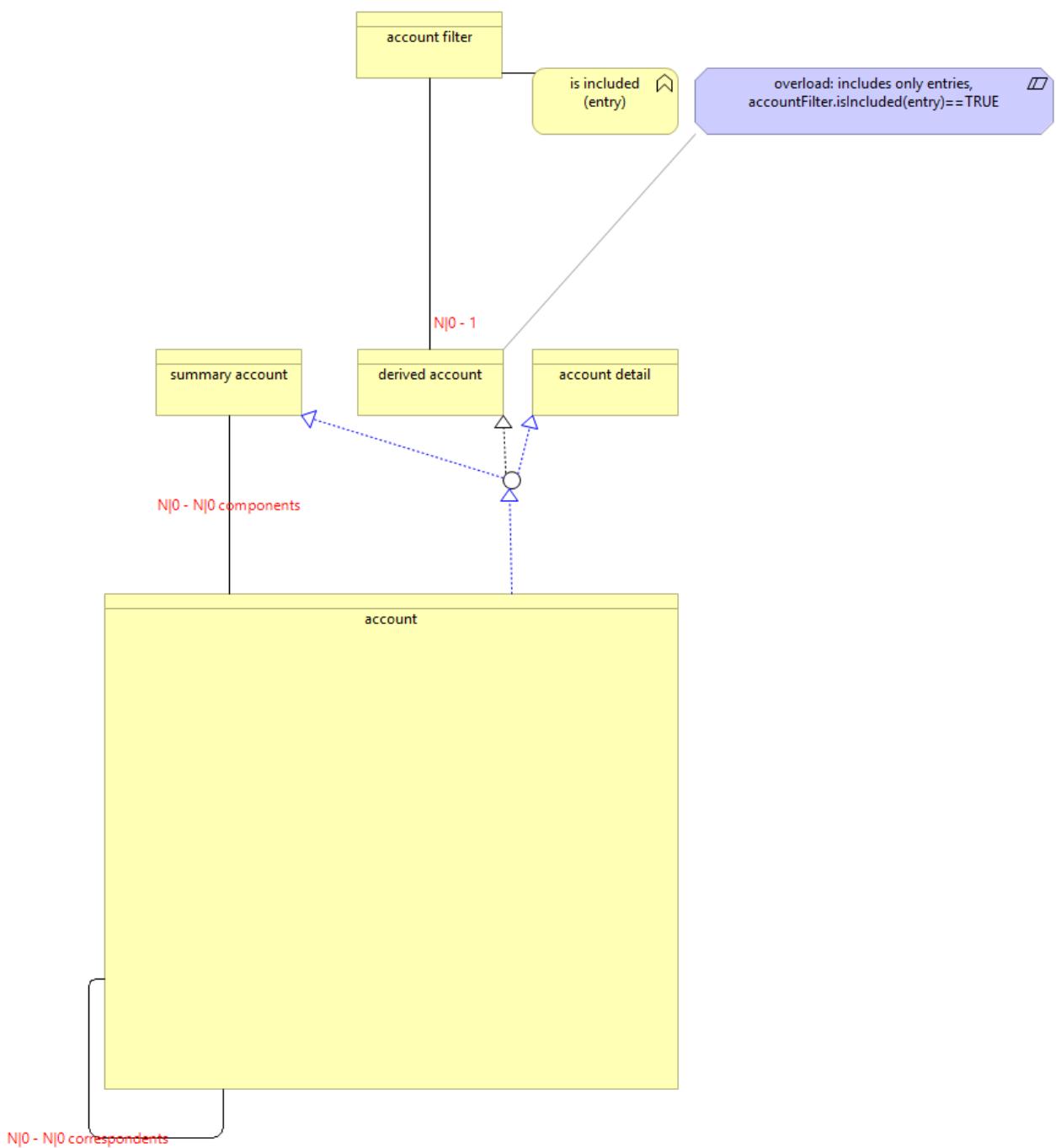
# SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)



# SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)



# BOOKING ENTRIES TO MULTIPLE ACCOUNTS

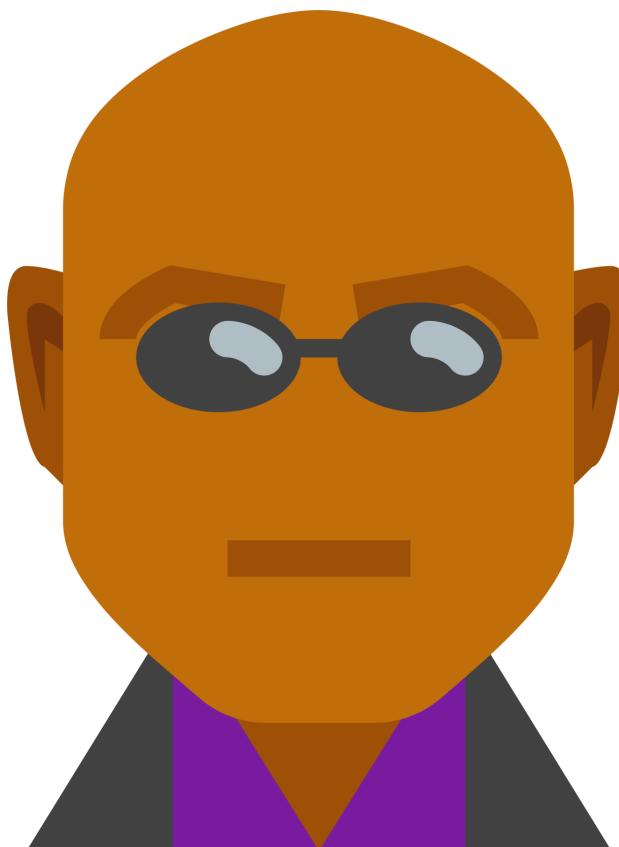


# PLANNING

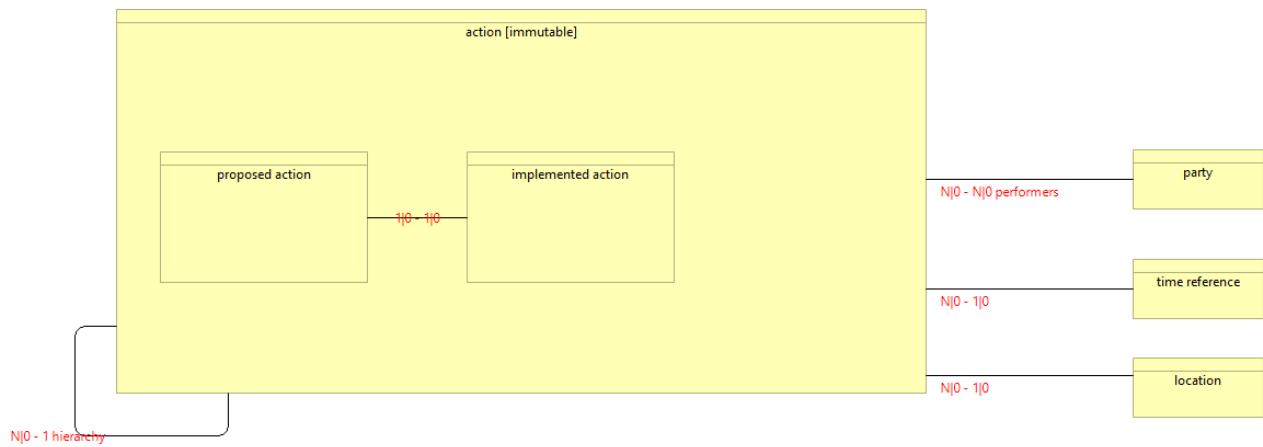
---

ANALYSIS PATTERNS

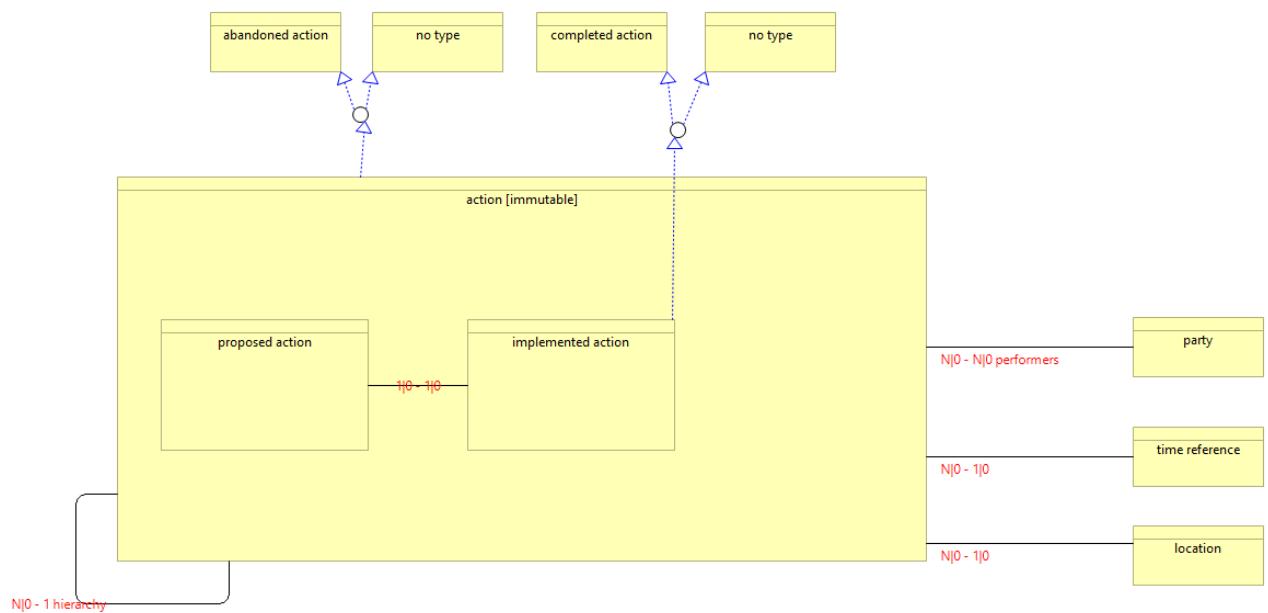
Martin Fowler



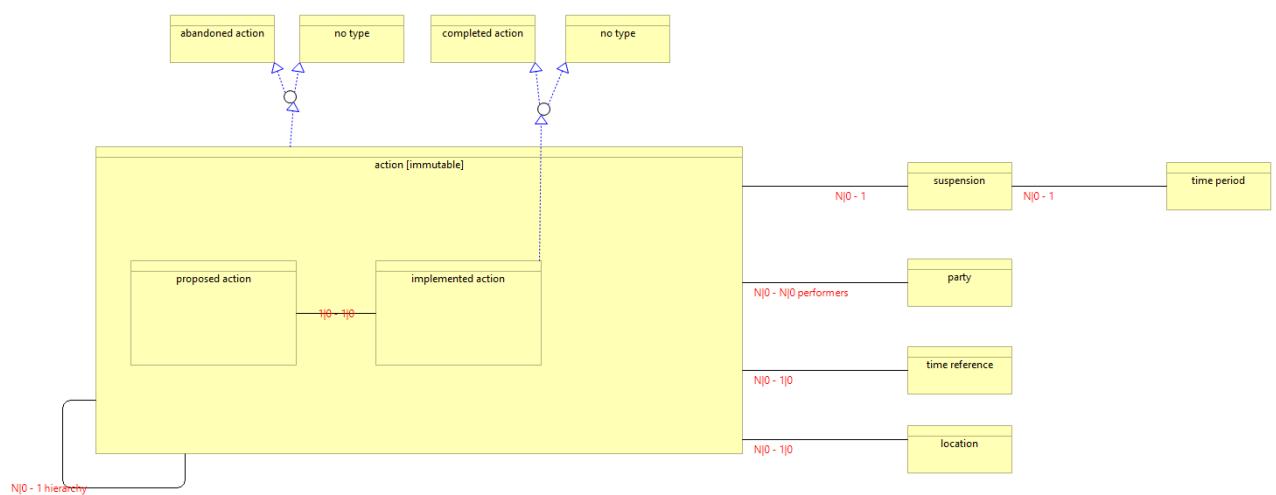
# PROPOSED AND IMPLEMENTED ACTION



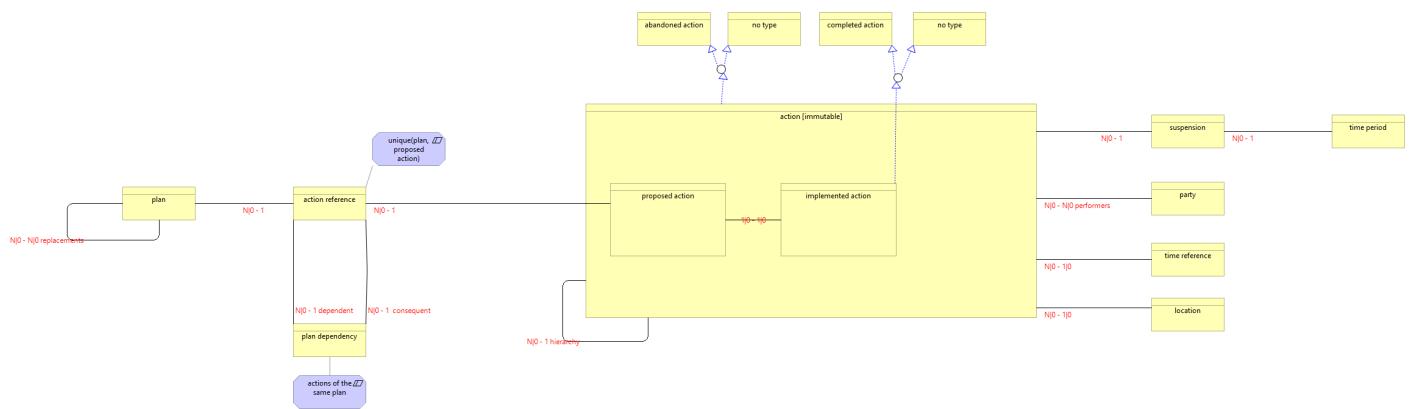
# COMPLETED AND ABANDONED ACTIONS



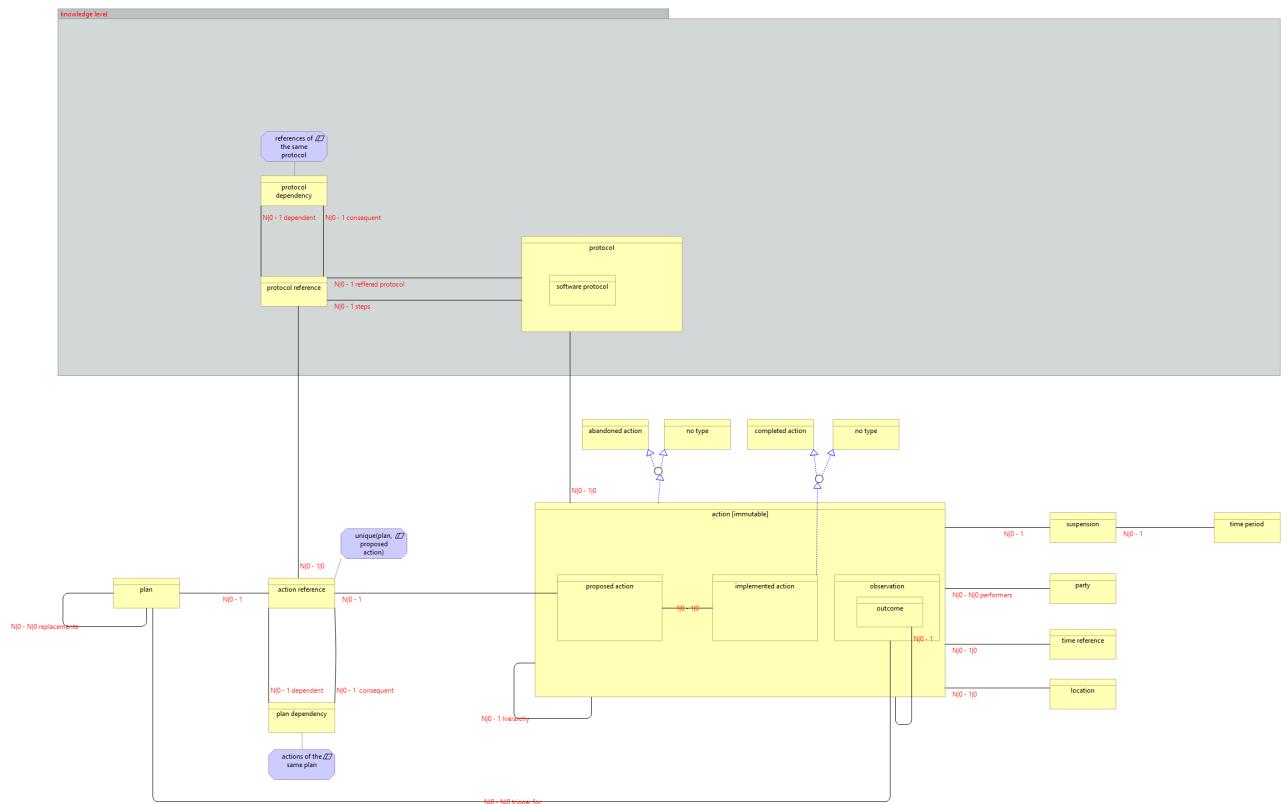
# SUSPENSION



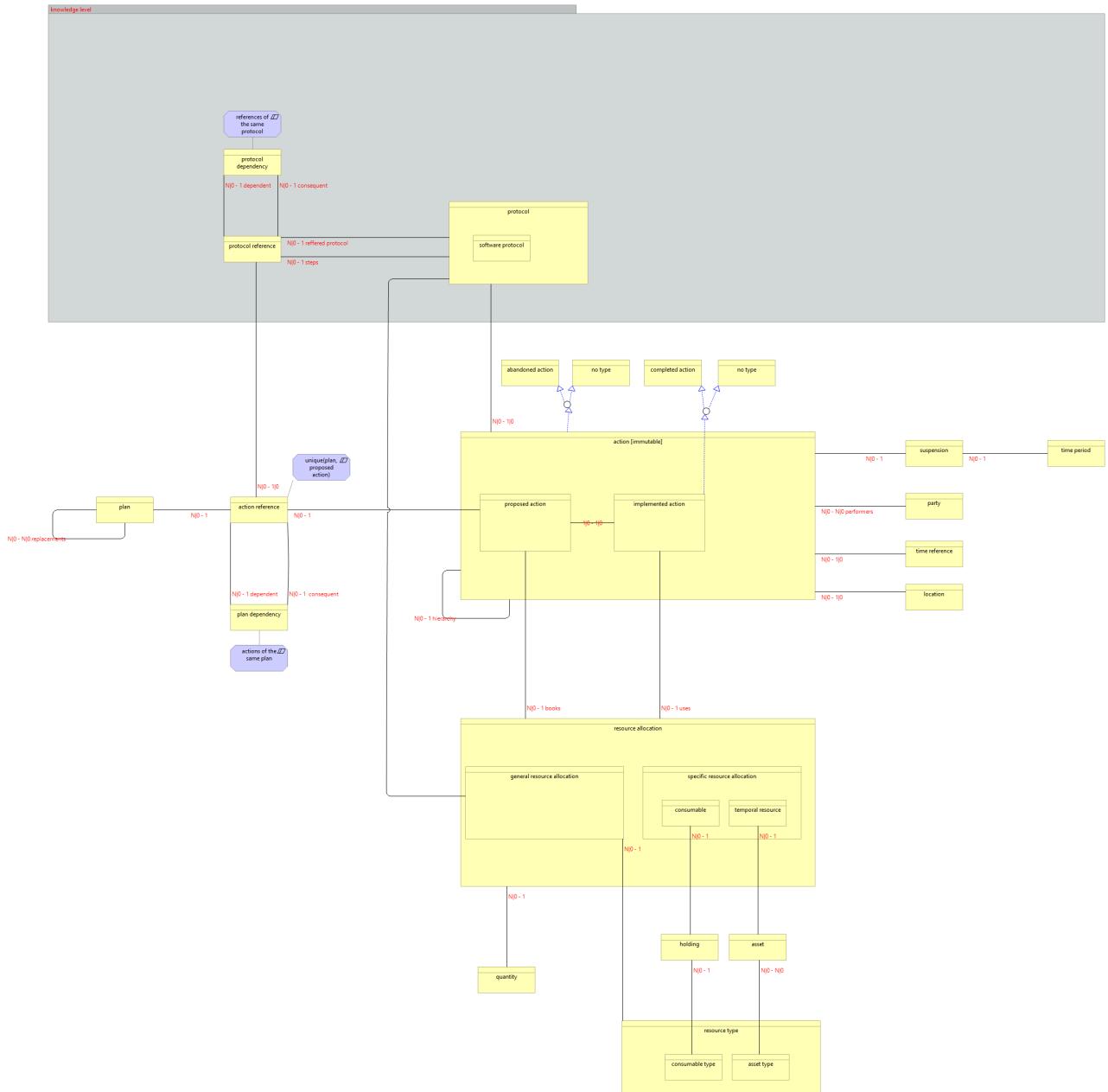
# PLAN



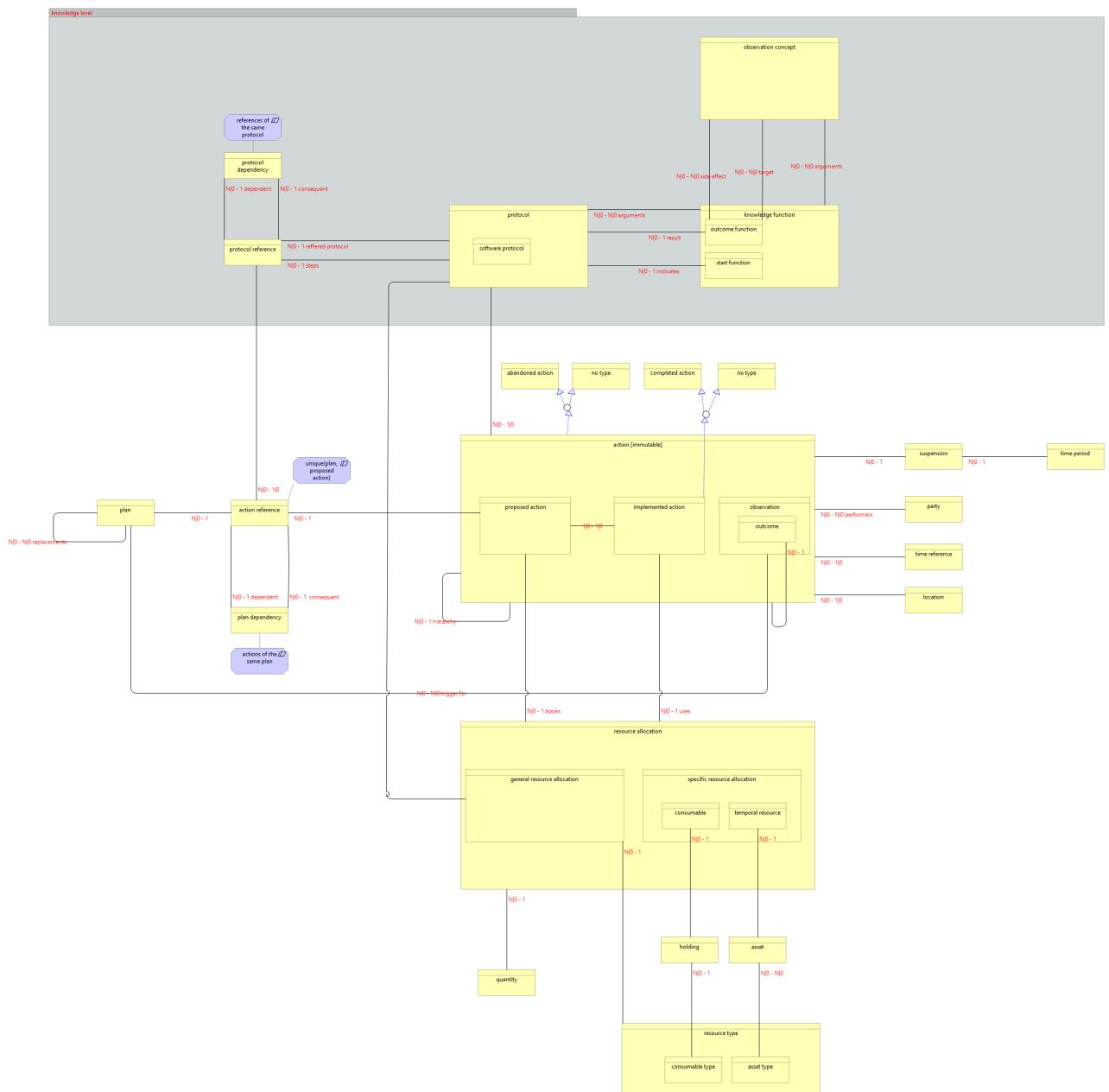
# PROTOCOL



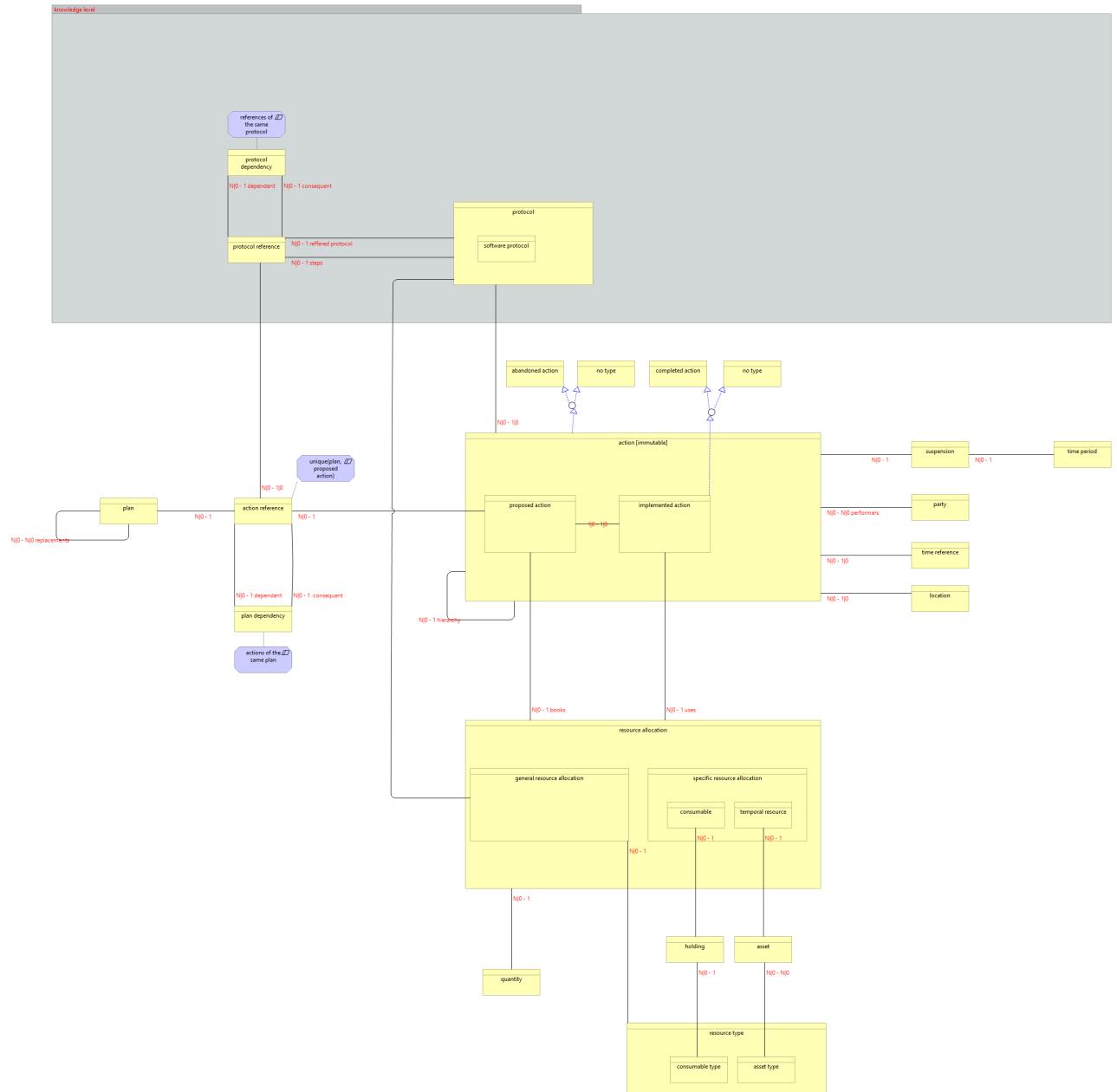
# RESOURCE ALLOCATION



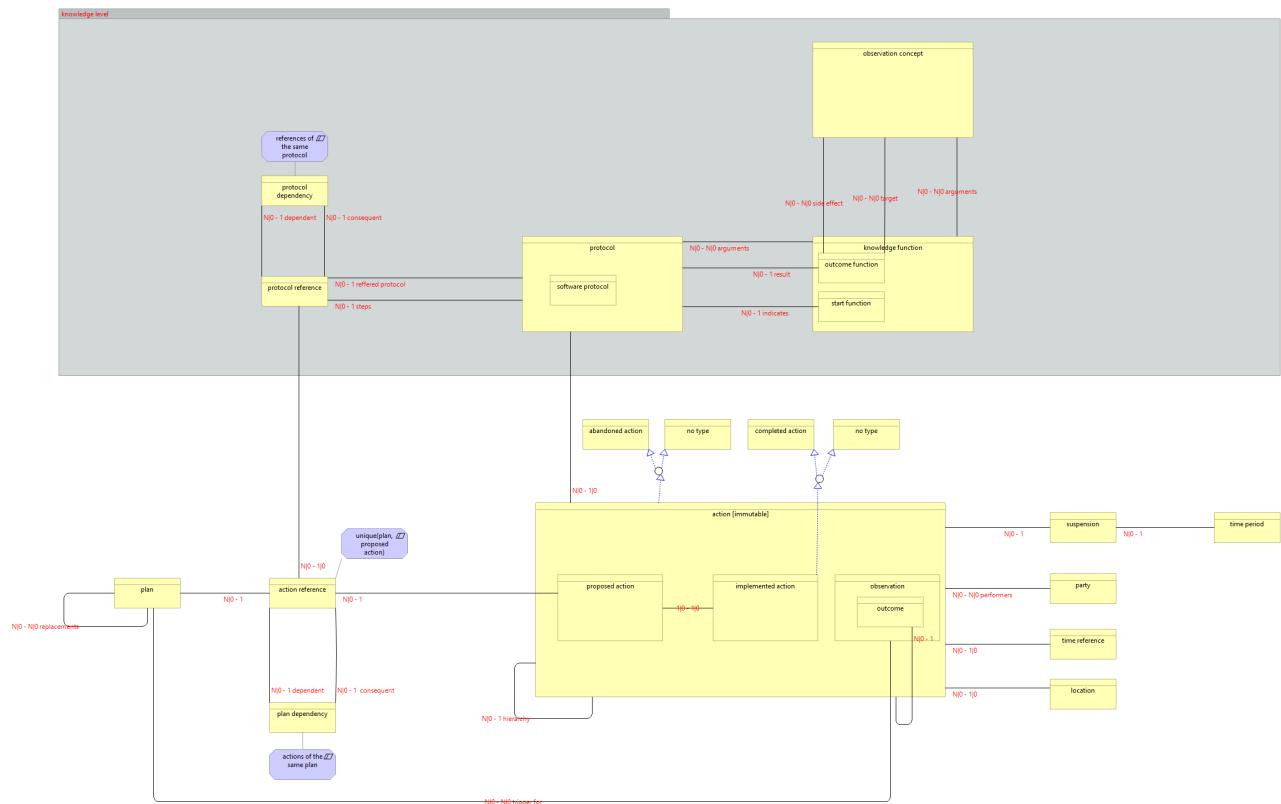
# PLANNING



# PLANNING (NO OUTCOME)



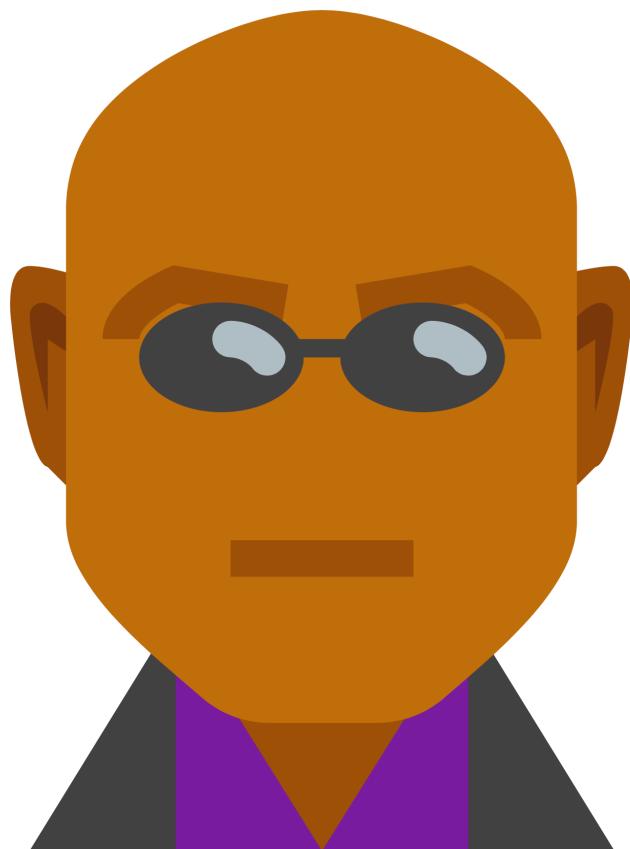
# OUTCOME AND START FUNCTIONS



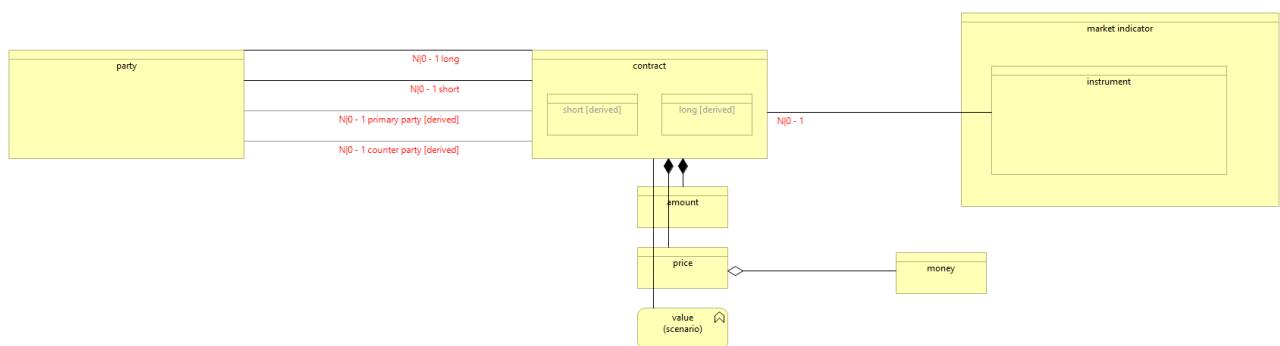
# TRADING

ANALYSIS PATTERNS

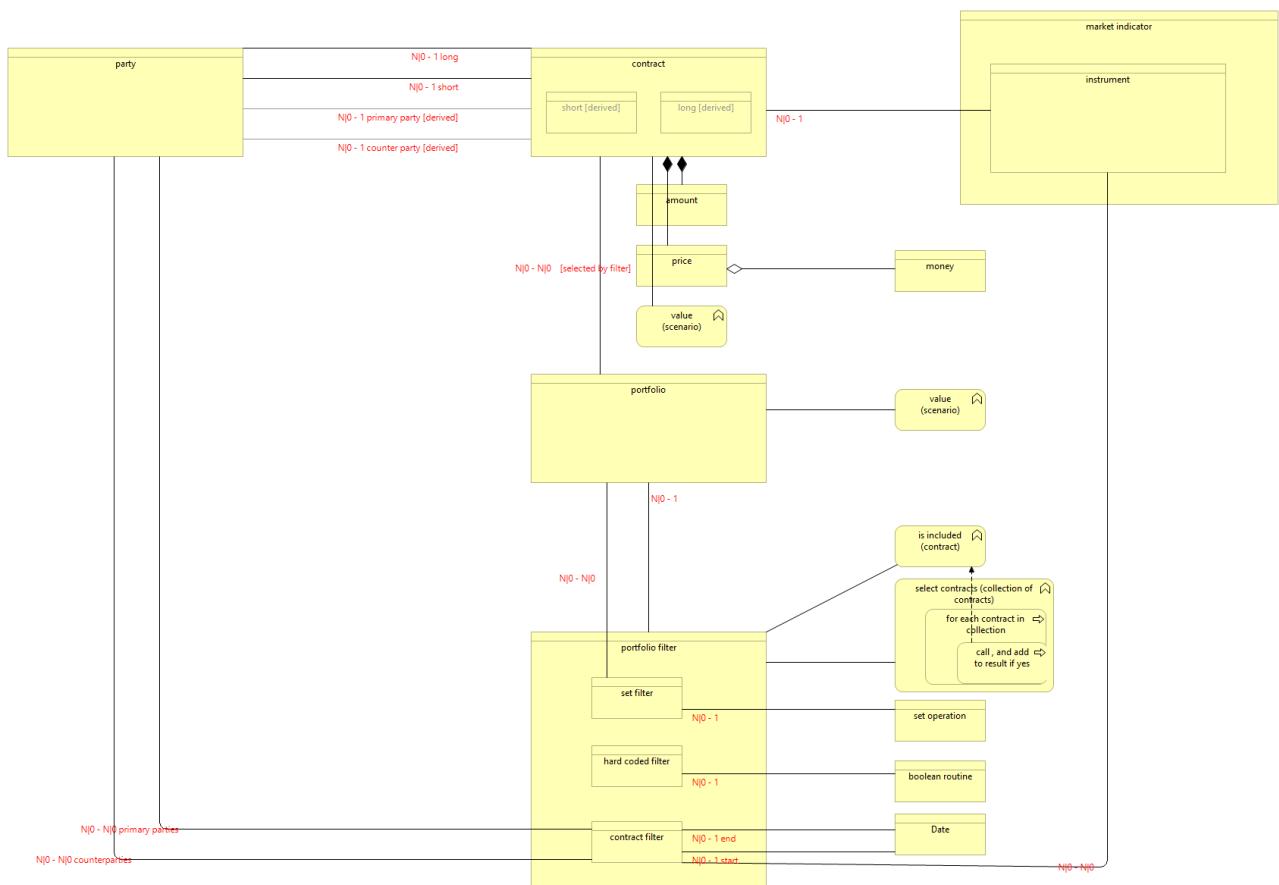
Martin Fowler



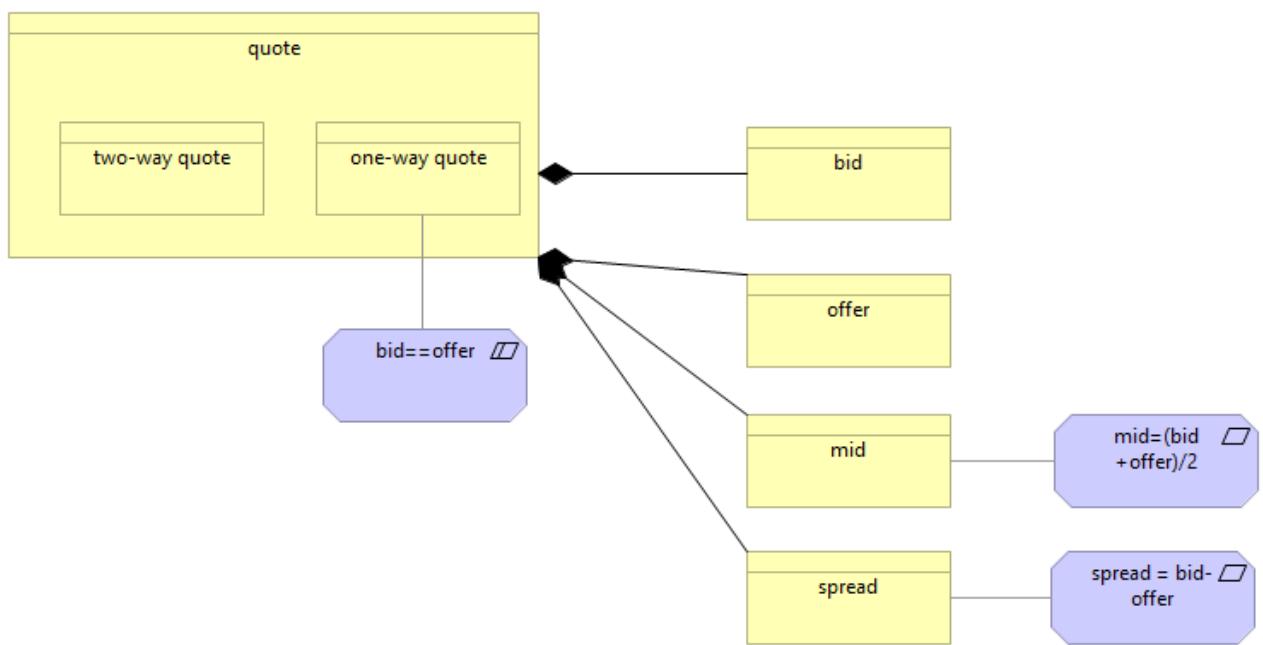
# CONTRACT



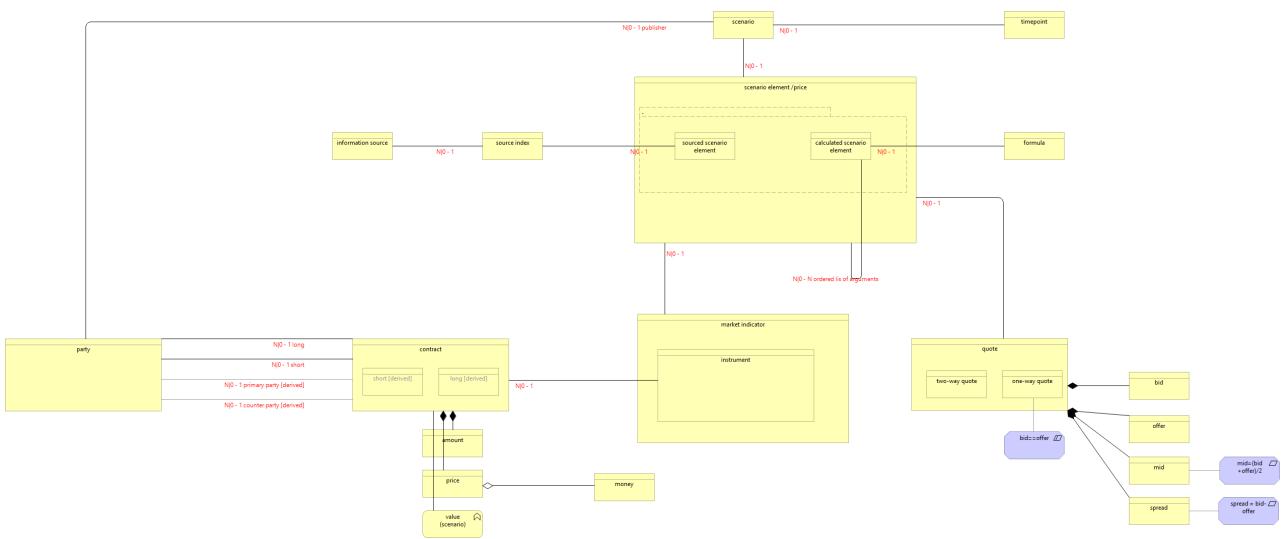
# PORTFOLIO



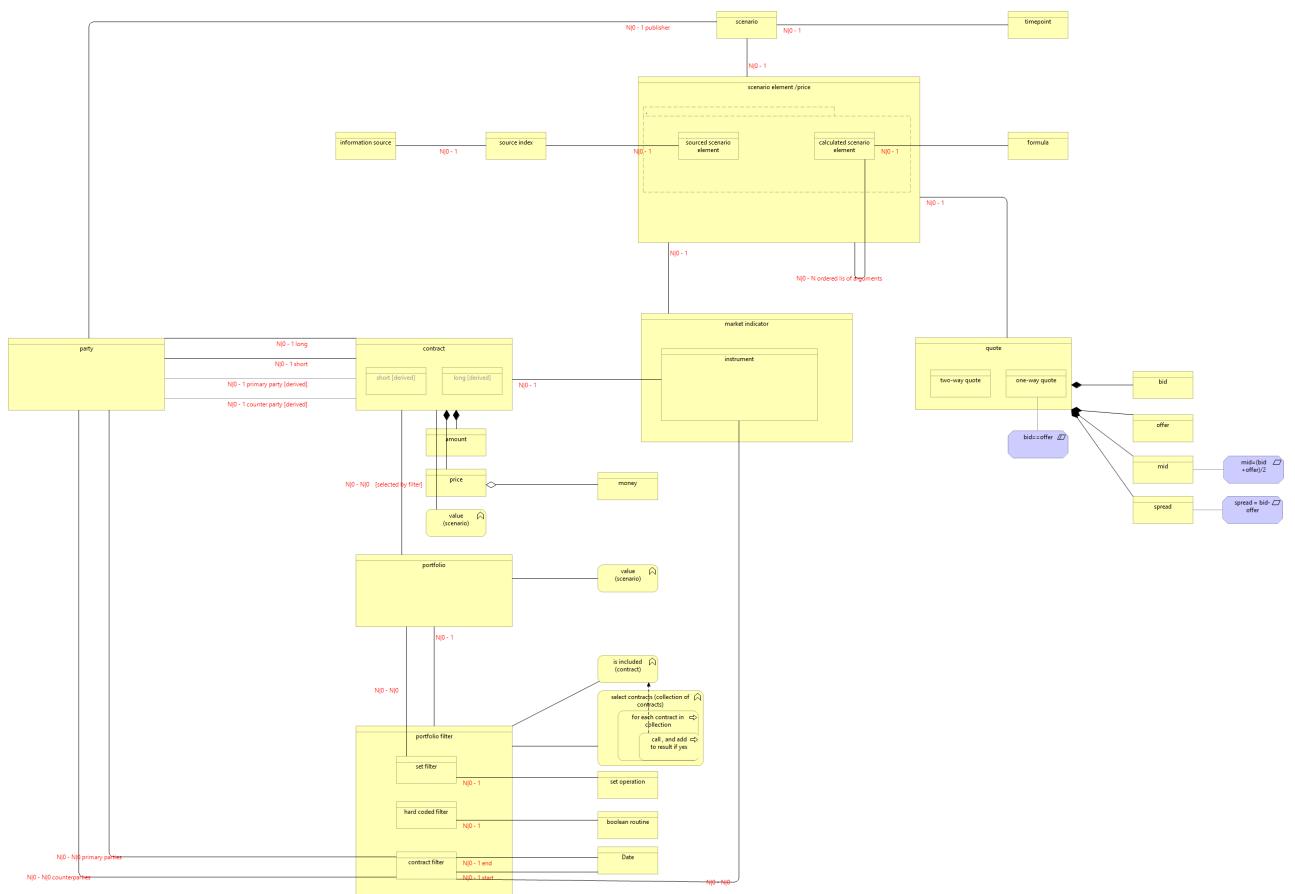
# QUOTE



# SCENARIO



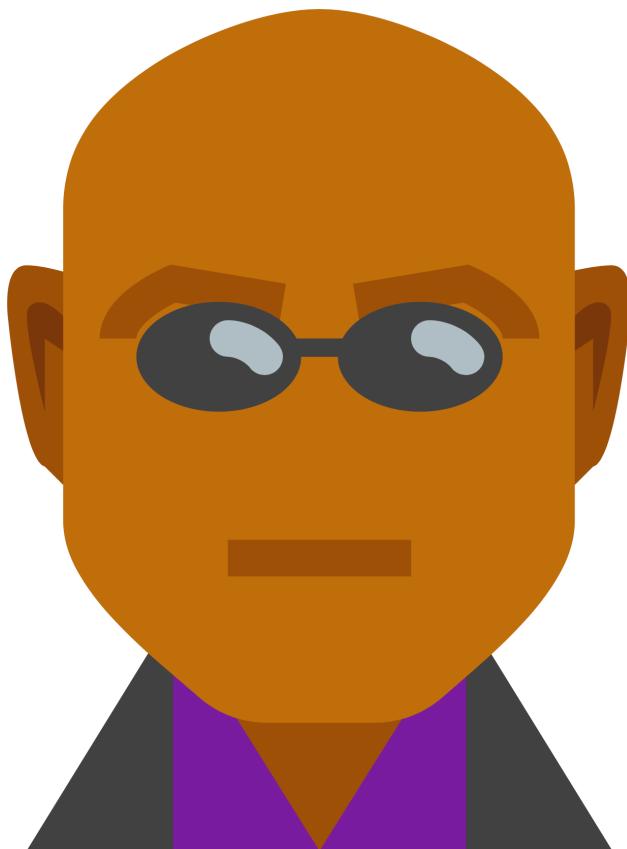
# TRADING



# DERIVATIVE CONTRACTS

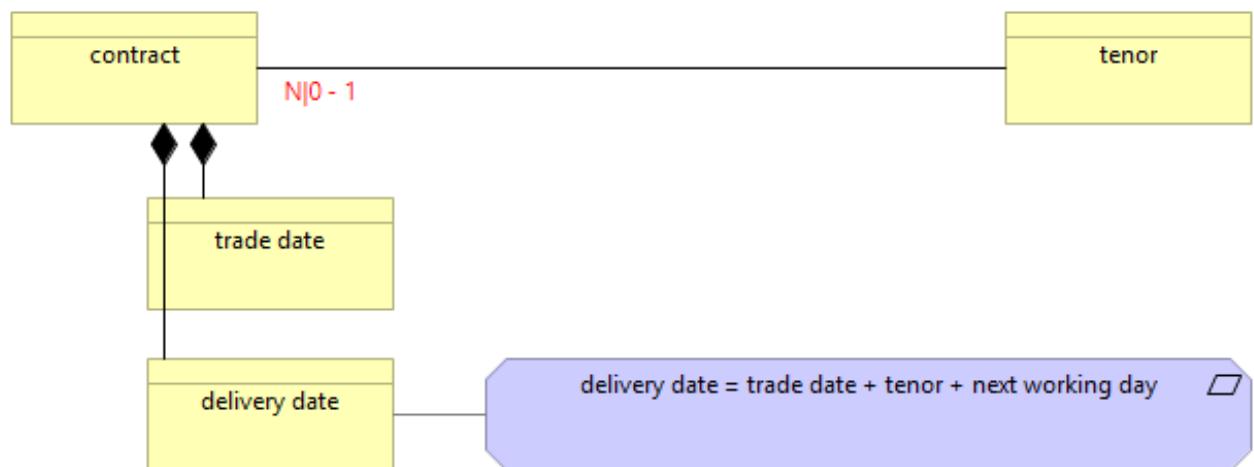
ANALYSIS PATTERNS

Martin Fowler



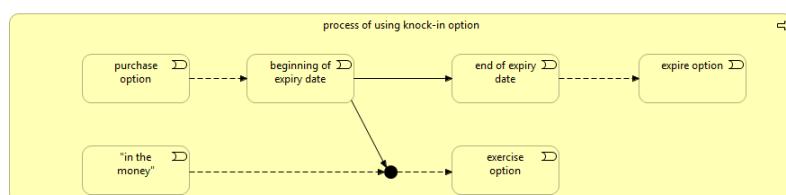
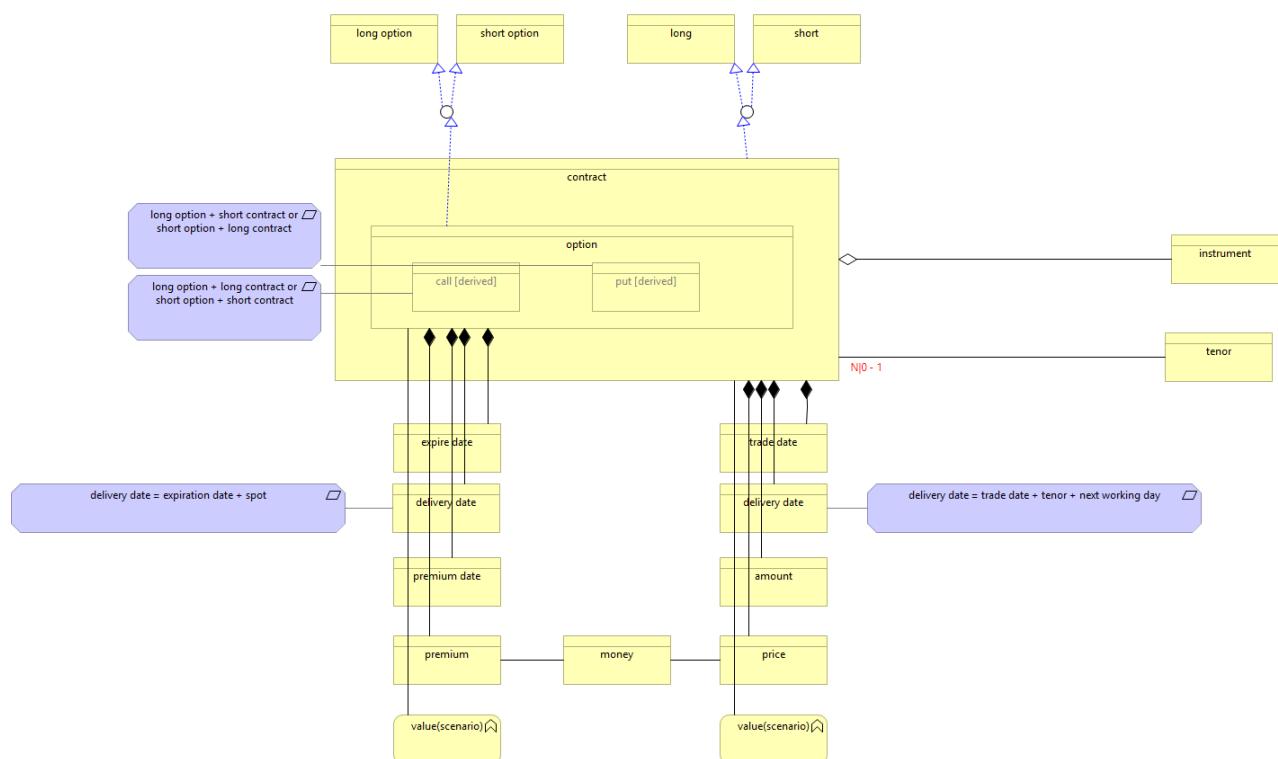
# FORWARD CONTRACTS

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



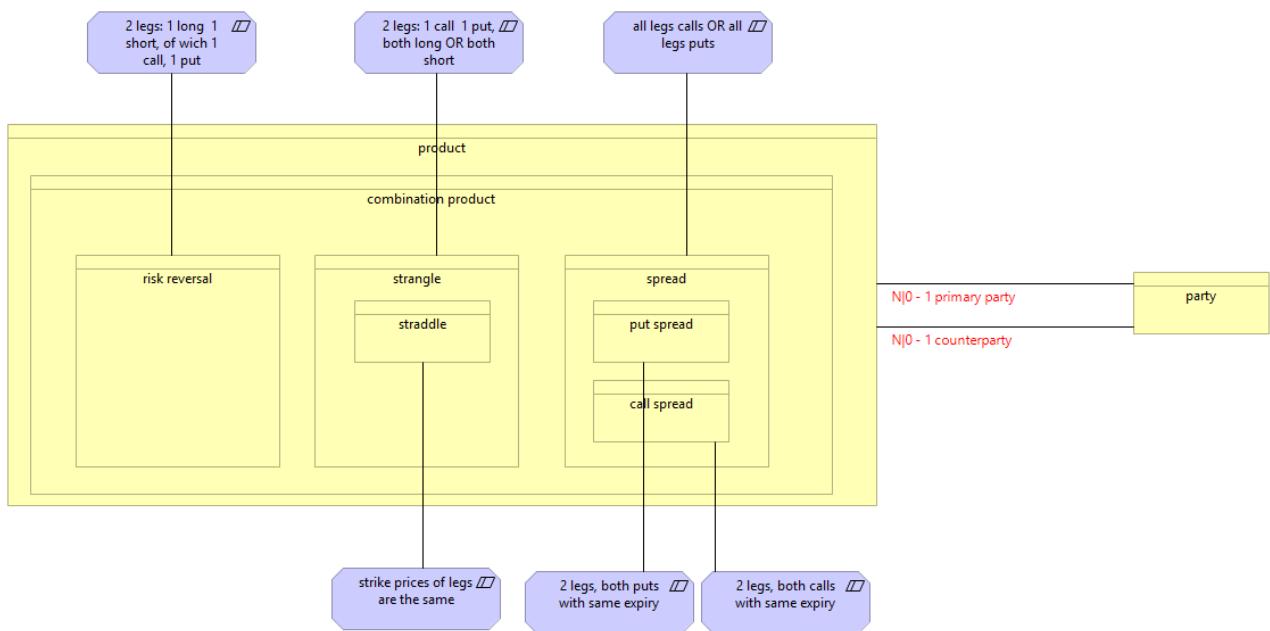
# OPTIONS

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

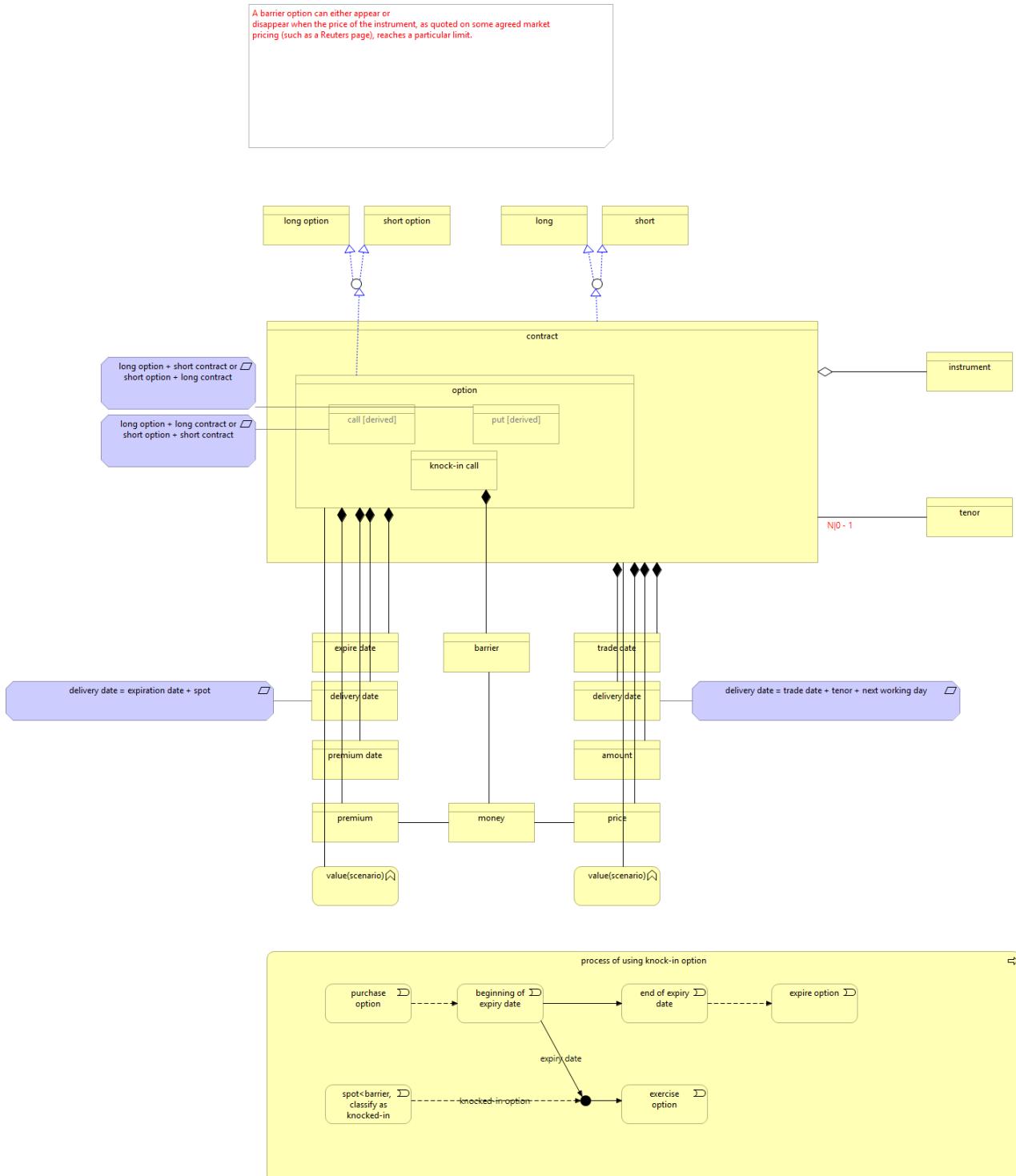


# PRODUCT

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

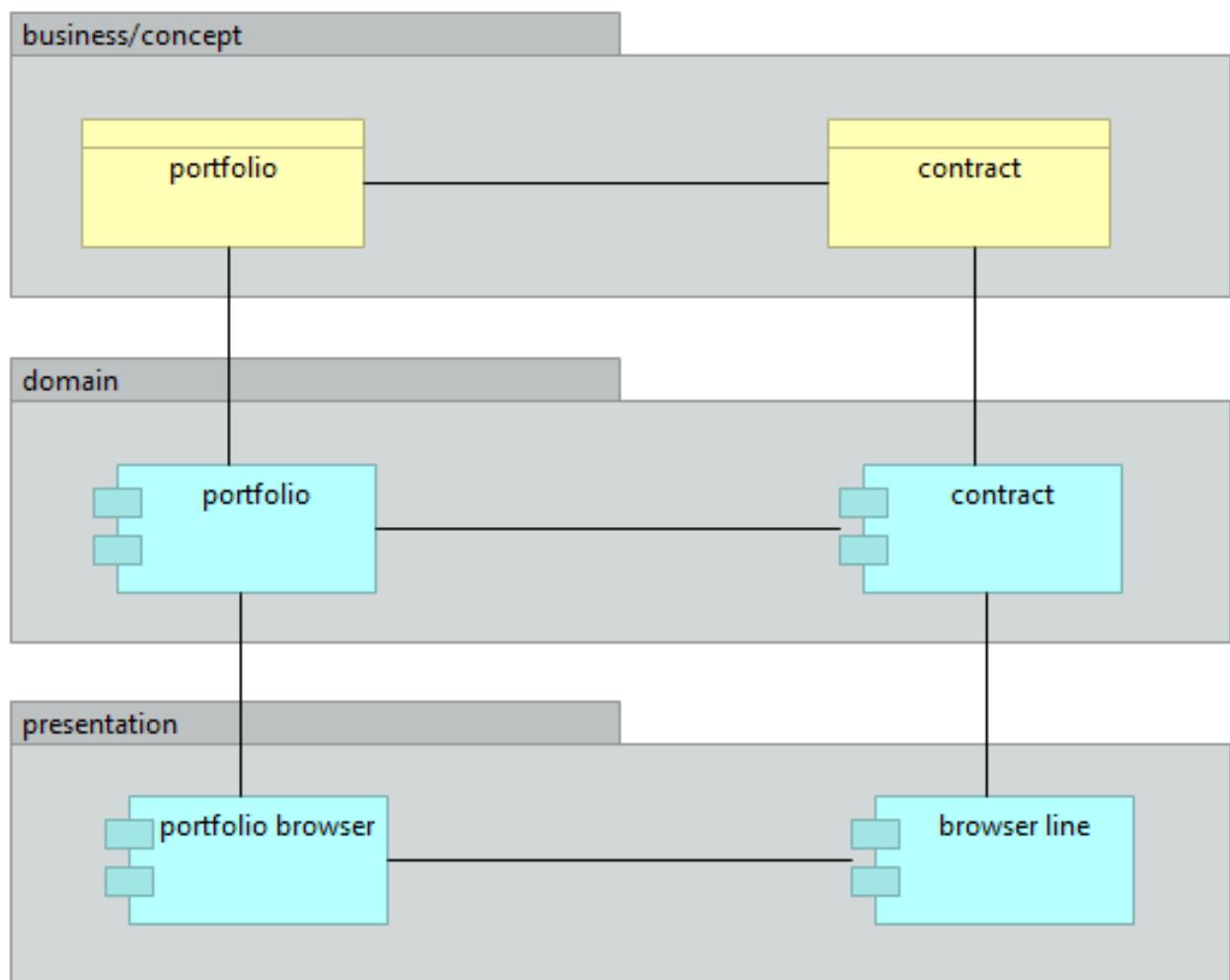


# SUBTYPE STATE MACHINES

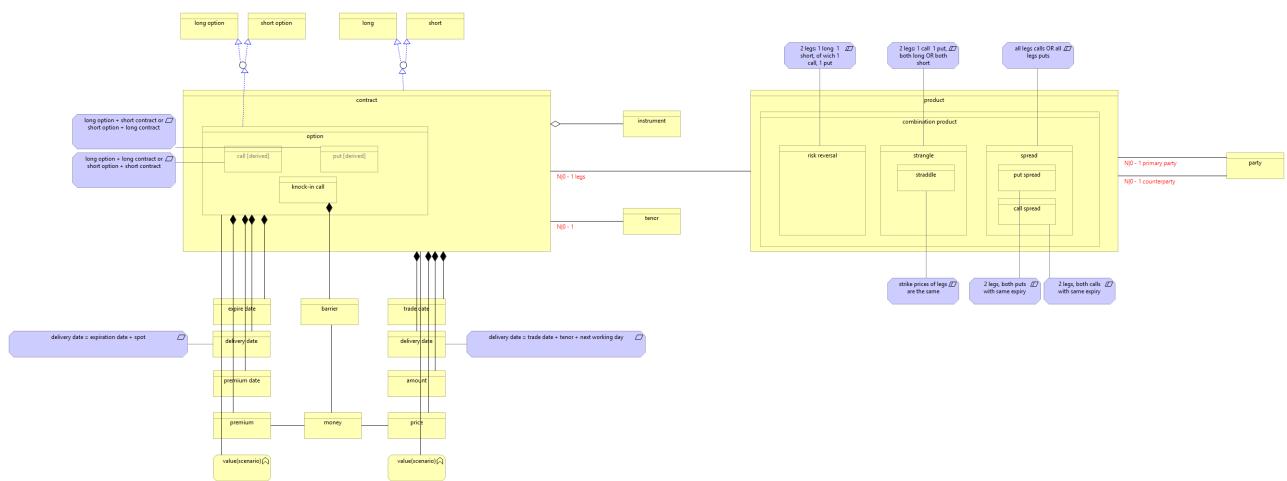


# PARALLEL APPLICATION AND DOMAIN HIERARCHIES

example, a report containing a list of contracts



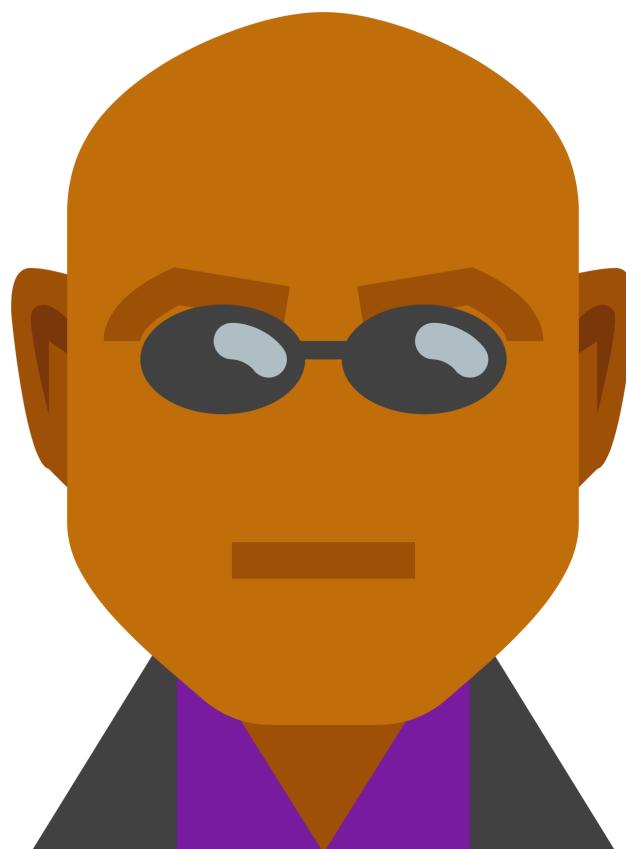
# DERIVATIVE CONTRACTS



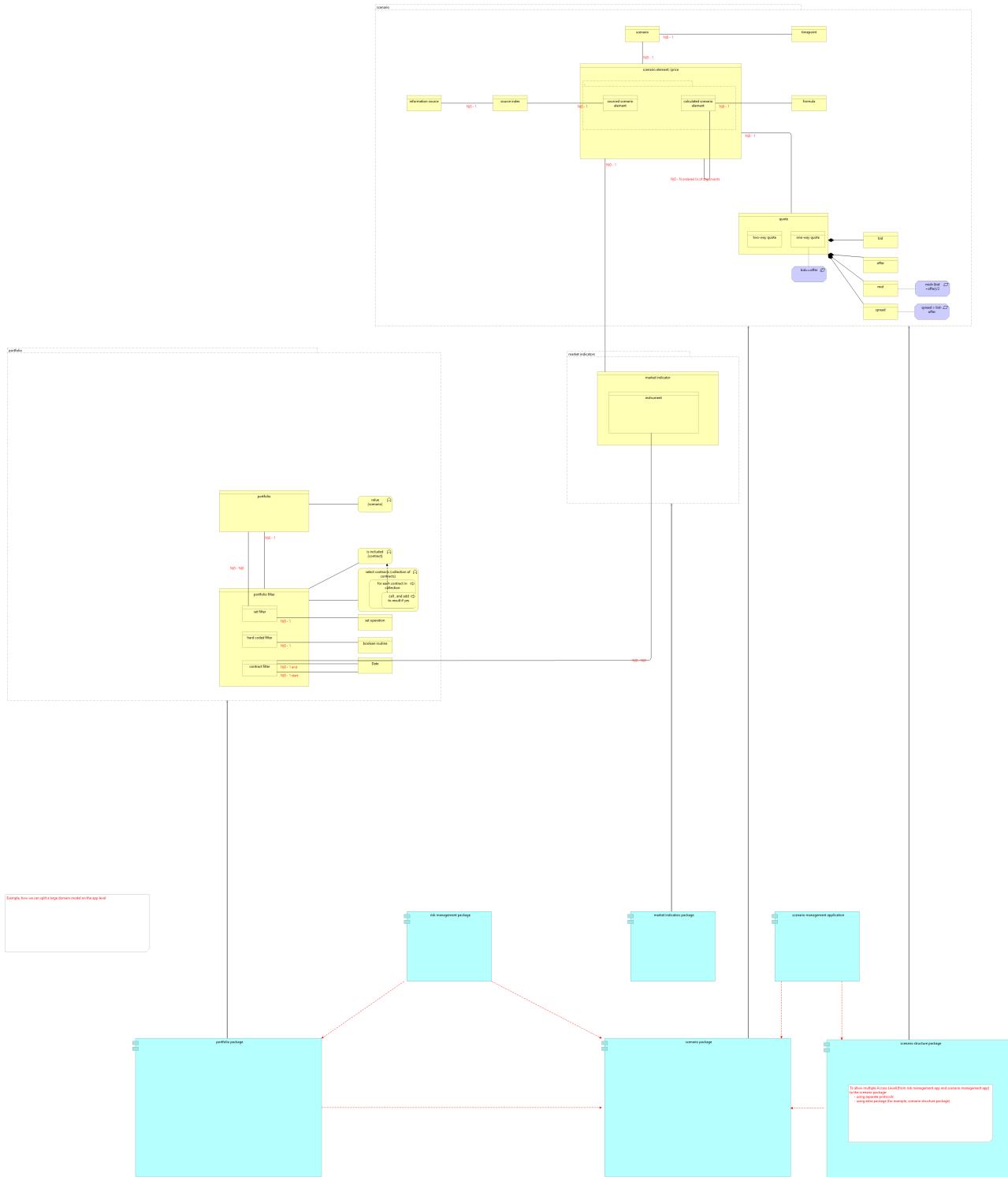
# TRADING PACKAGES

ANALYSIS PATTERNS

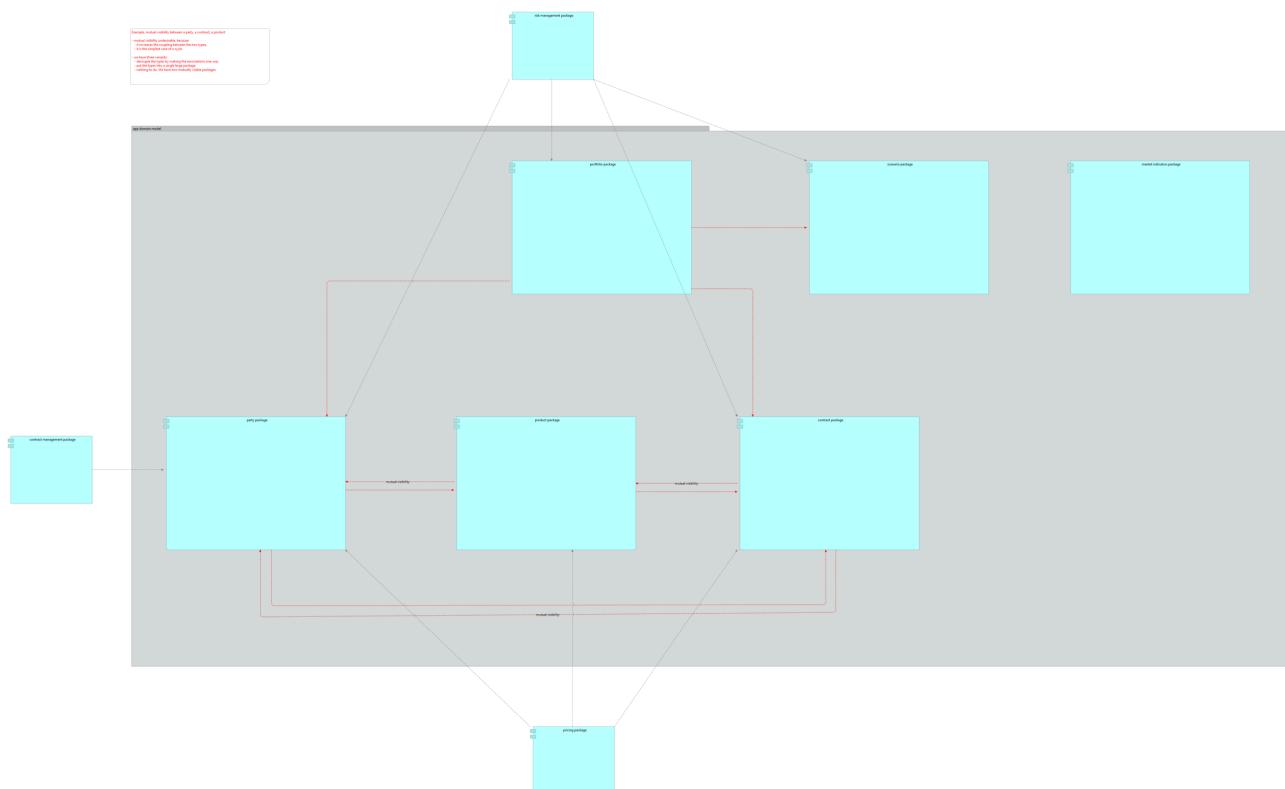
Martin Fowler



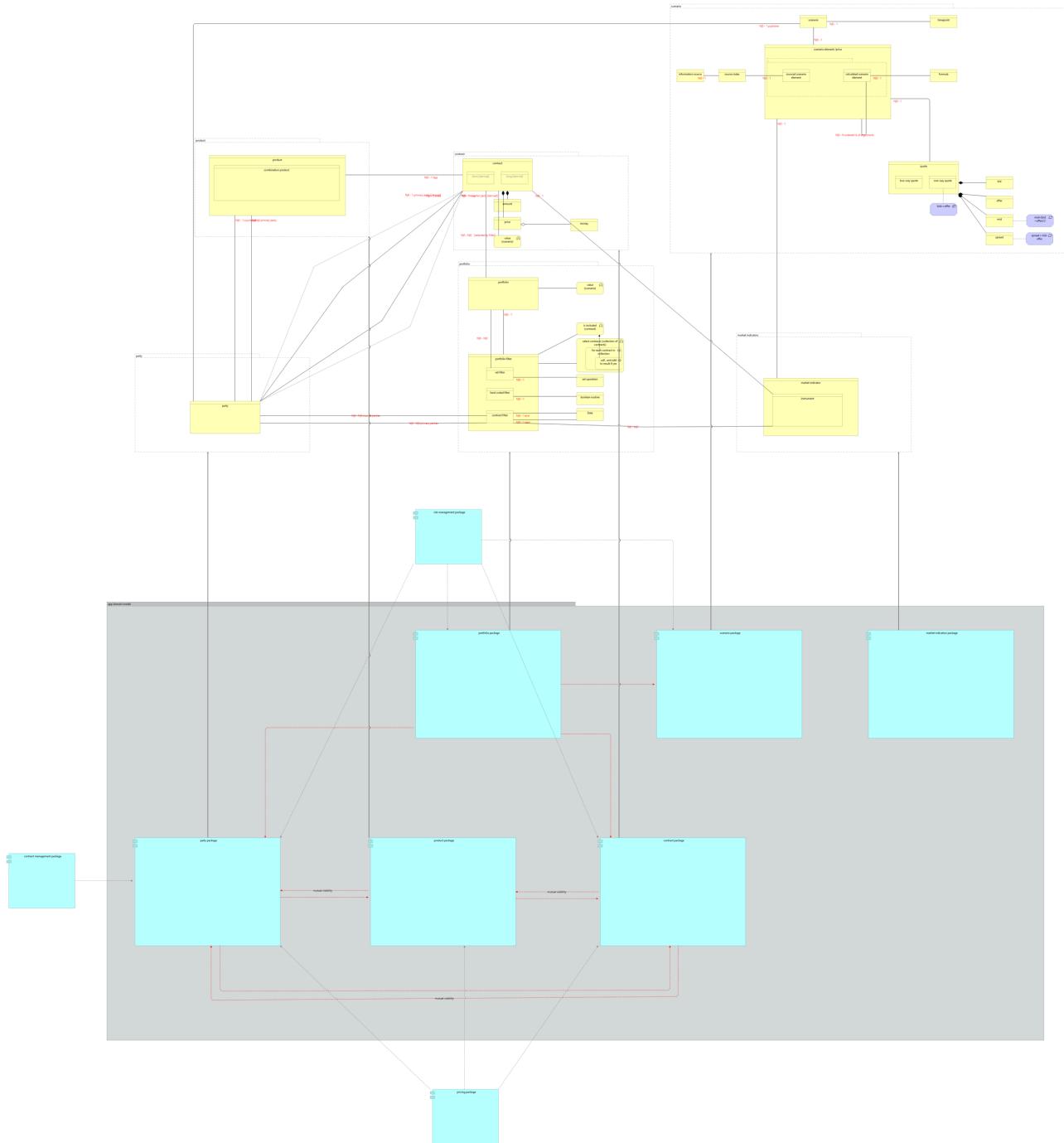
# MULTIPLE ACCESS LEVELS TO A PACKAGE



# MUTUAL VISIBILITY



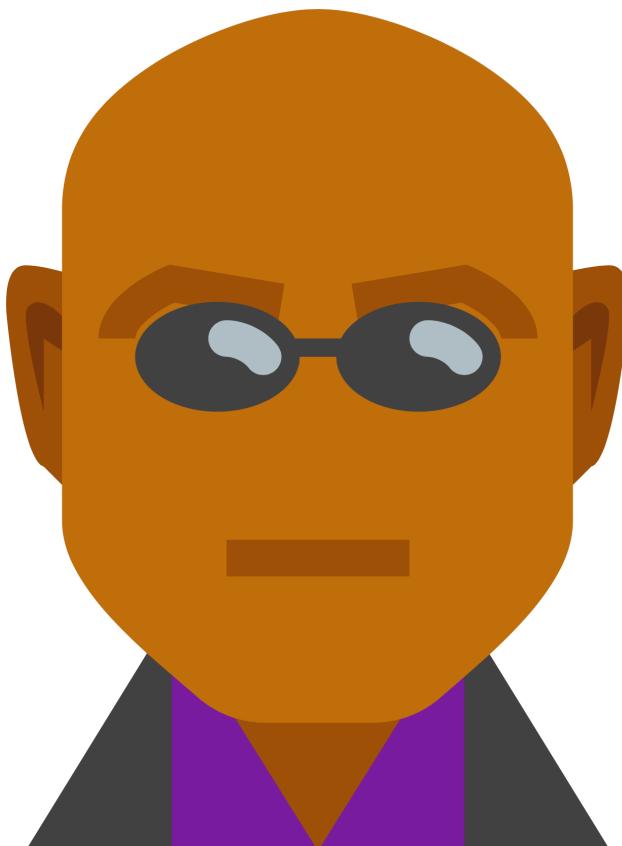
# TRADING PACKAGES



# LAYERED ARCHITECTURE

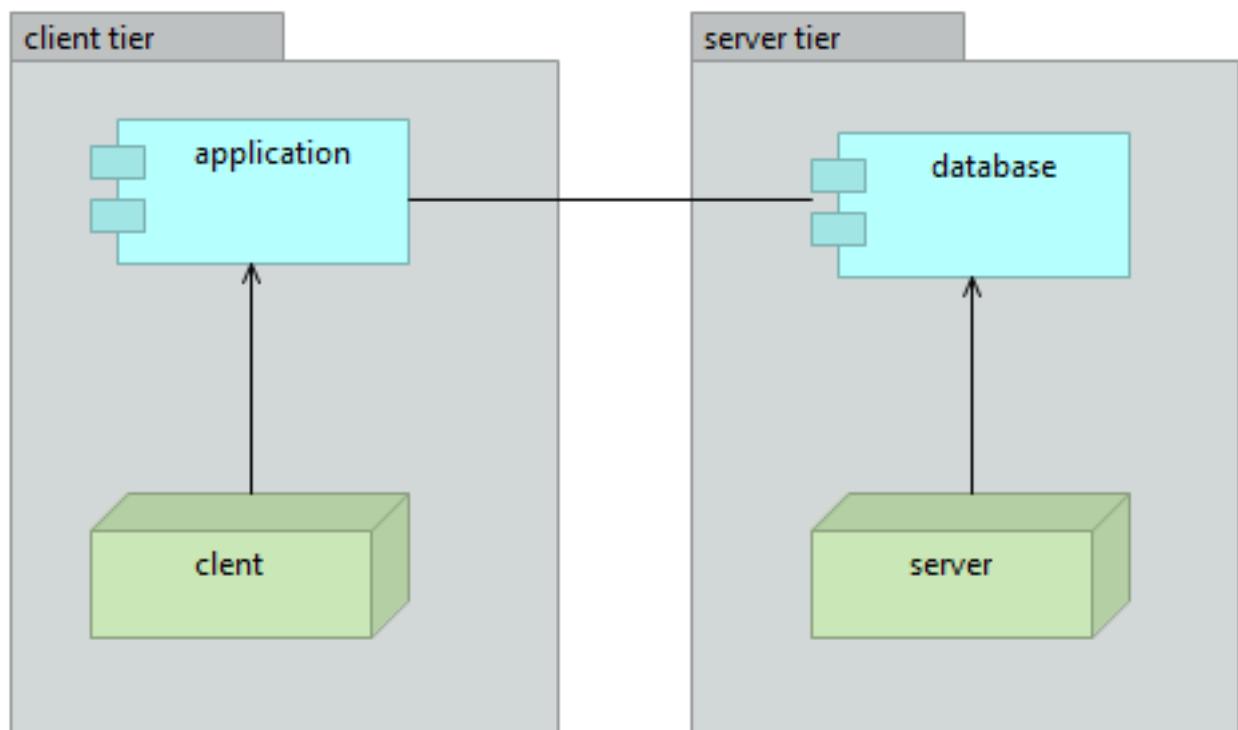
ANALYSIS PATTERNS

Martin Fowler

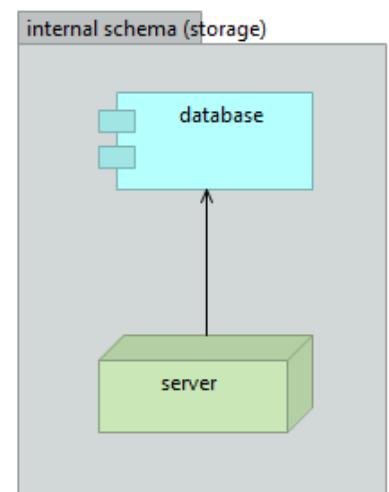
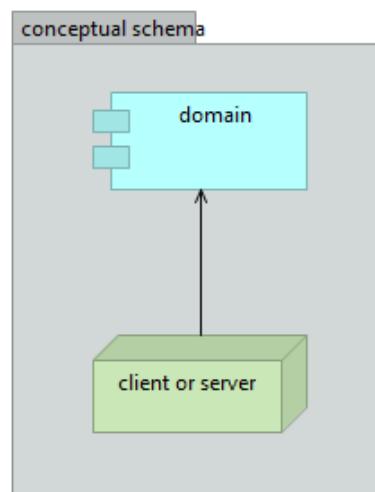
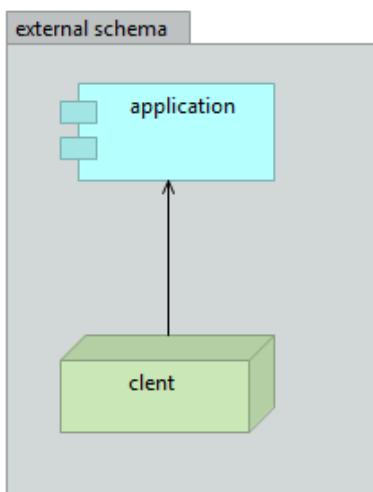


# TWO-TIER ARCHITECTURE

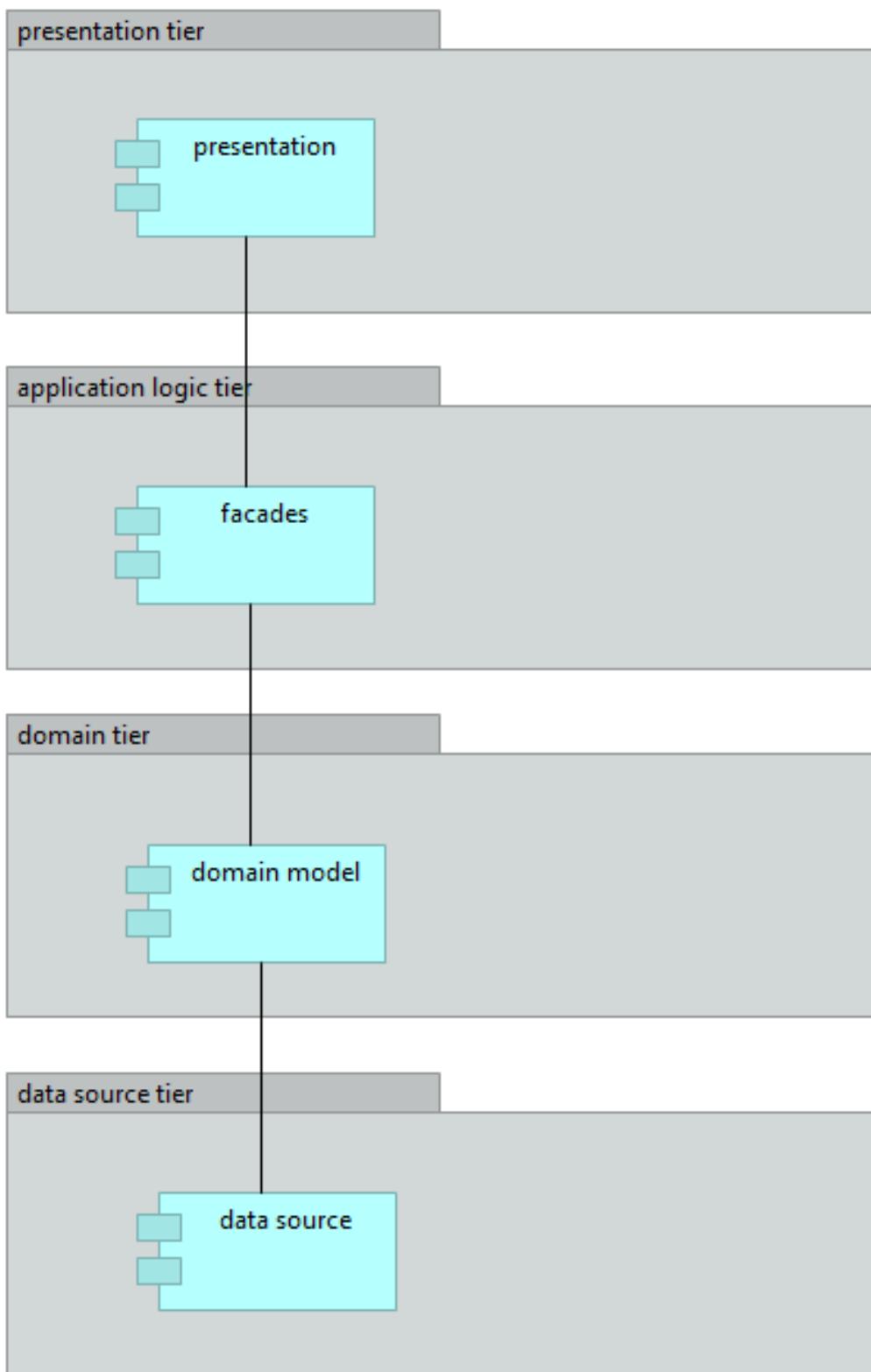
two-tier architecture



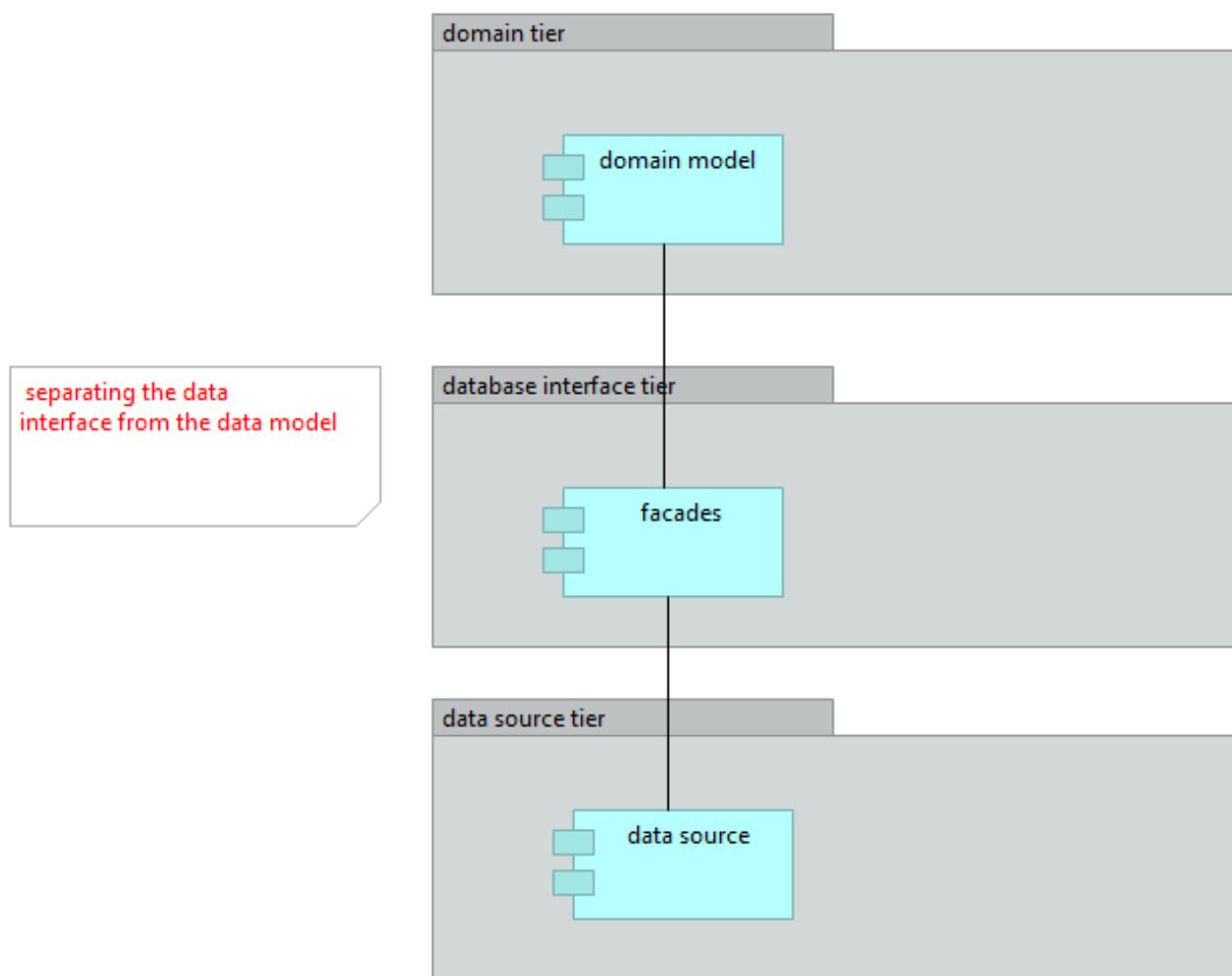
# THREE-TIER ARCHITECTURE



# PRESENTATION AND APPLICATION LOGIC



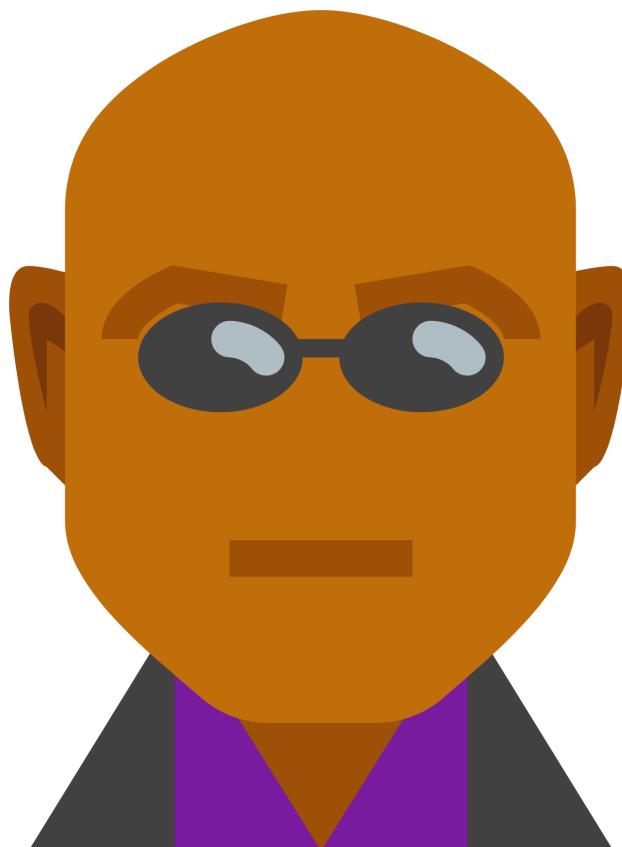
# DATABASE INTERACTION



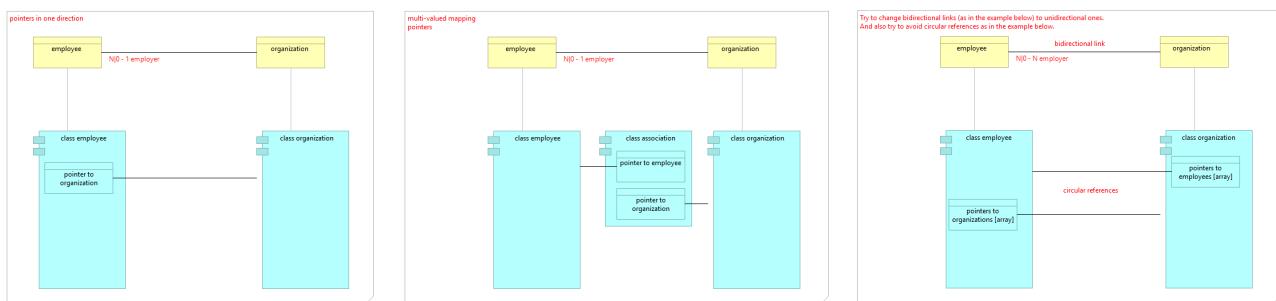
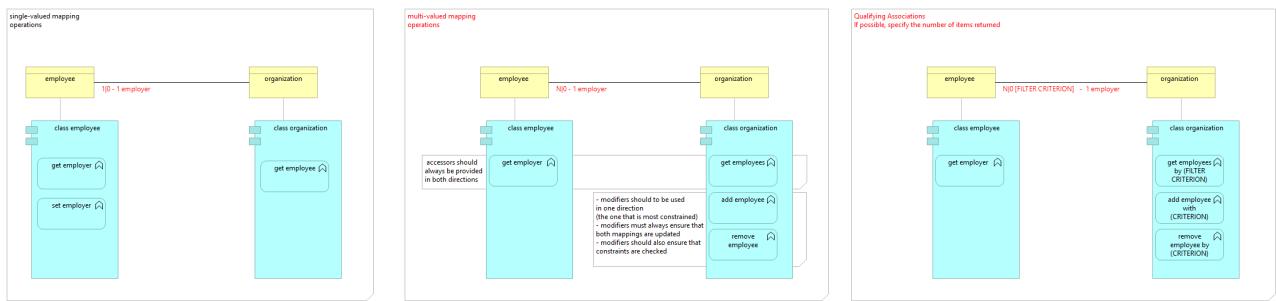
# TYPE MODEL DESIGN

ANALYSIS PATTERNS

Martin Fowler

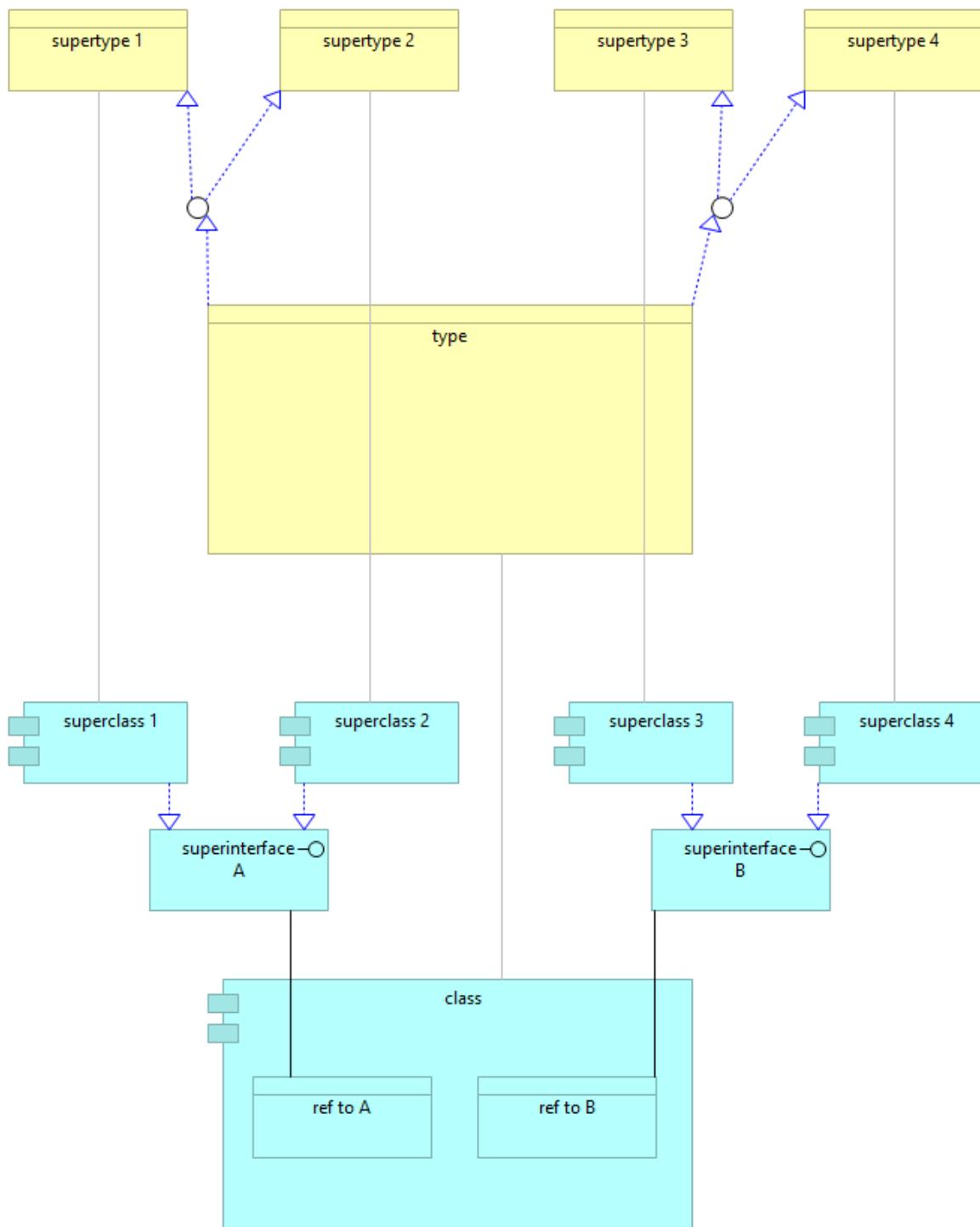


# IMPLEMENTING ASSOCIATIONS

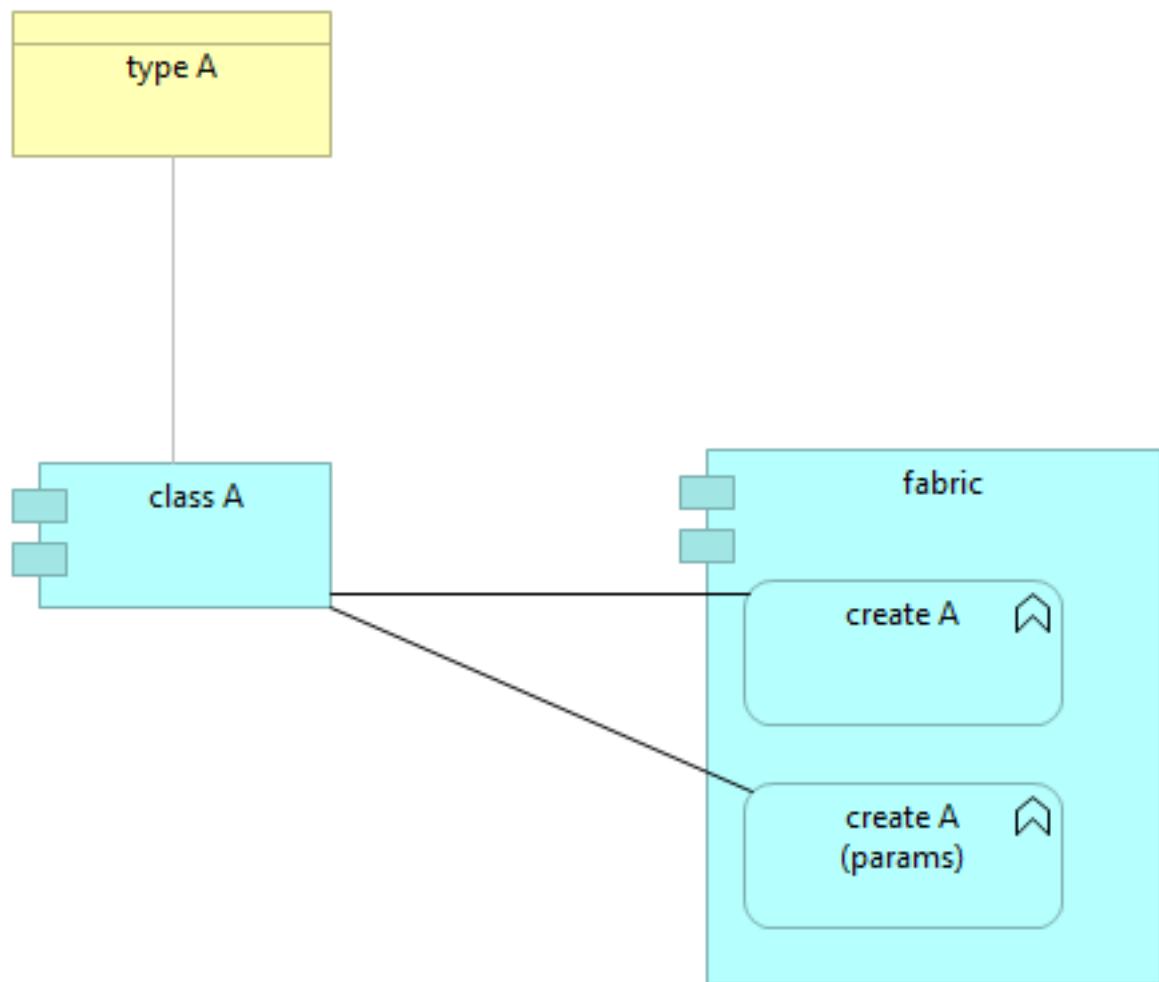


# IMPLEMENTING GENERALIZATION

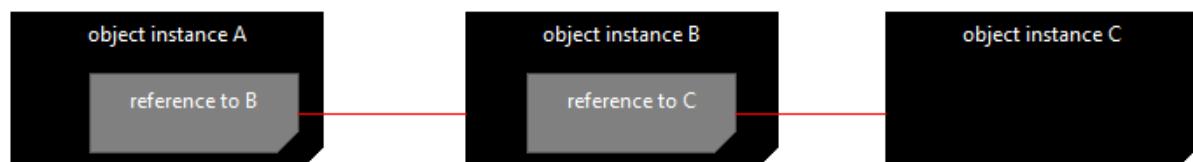
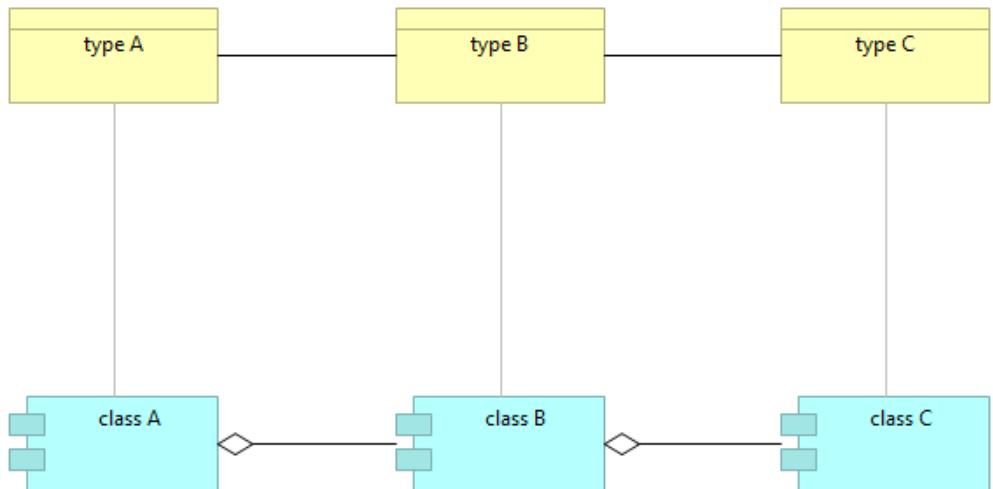
- example of implementation of multiple inheritance
- for example, the composition is applicable instead of inheritance



# OBJECT CREATION



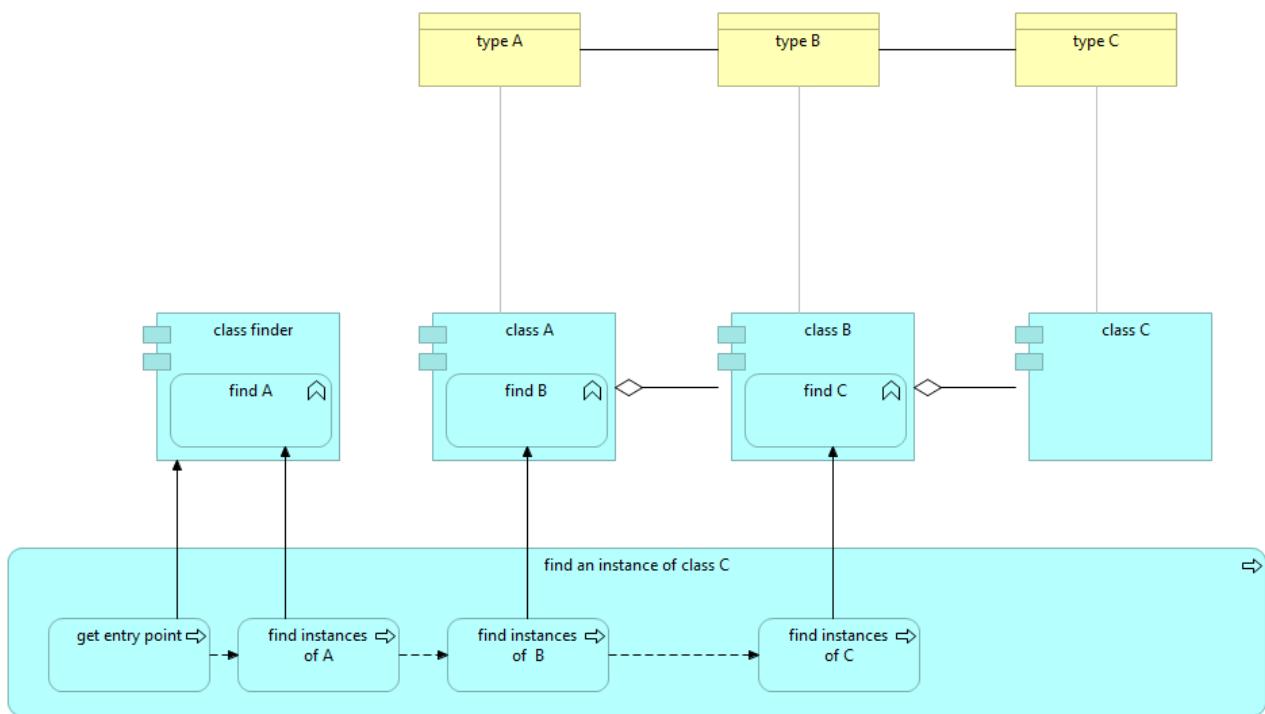
# OBJECT DESTRUCTION



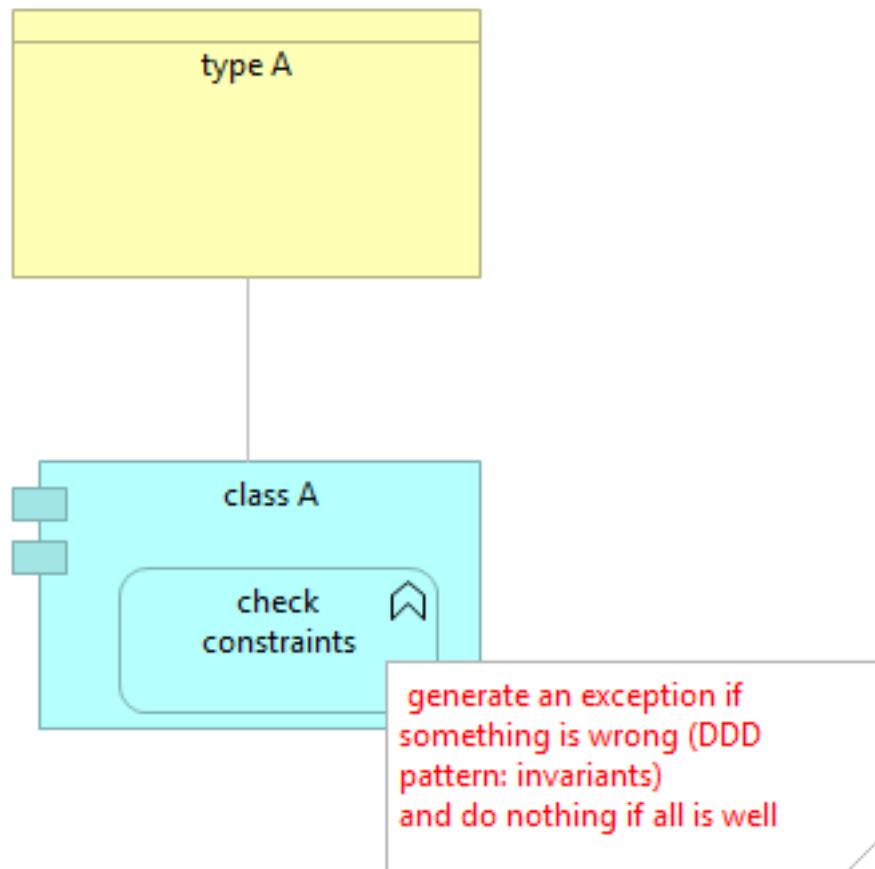
for example, delete object B  
- all constraints are met  
- no dangling references

object instance A

# ENTRY POINT.



# IMPLEMENTING CONSTRAINTS

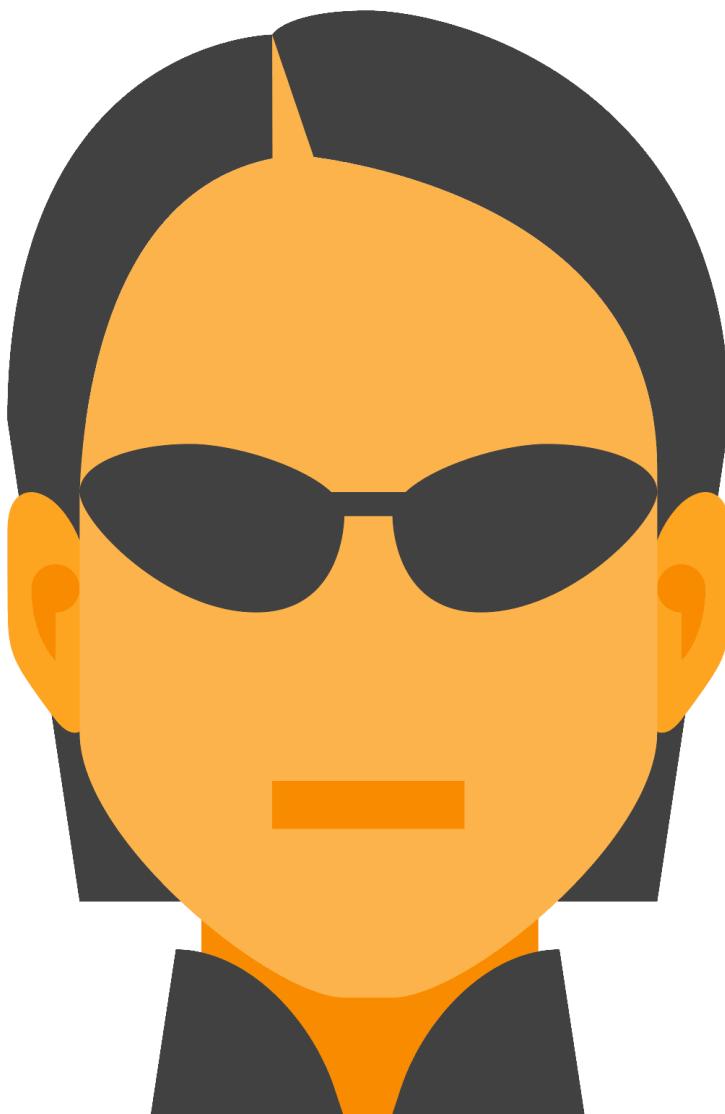


04

# Domain Driven Design

Tackling Complexity in the Heart  
of Software.

ERIC EVANS



# MODEL AND STRUCTURAL ELEMENTS

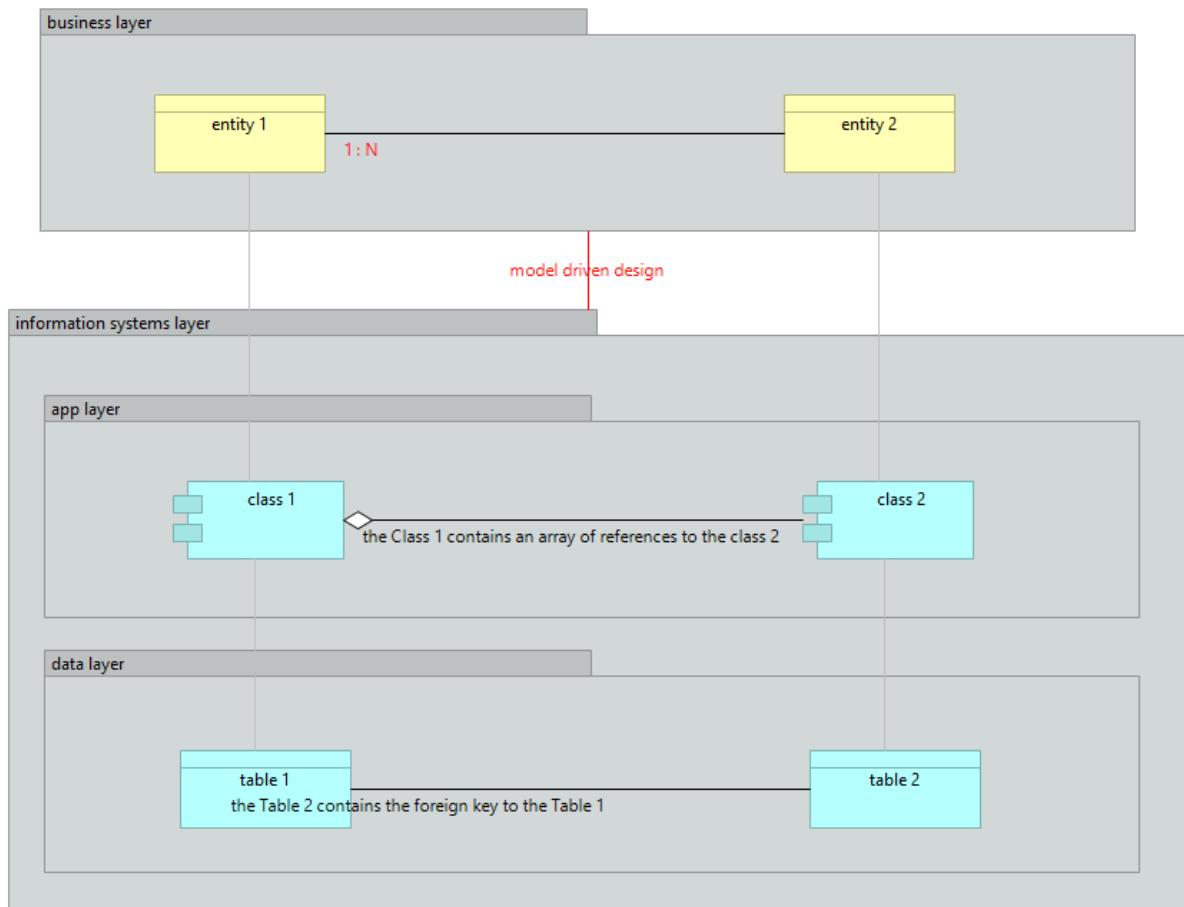
DOMAIN DRIVEN DESIGN

Eric Evans

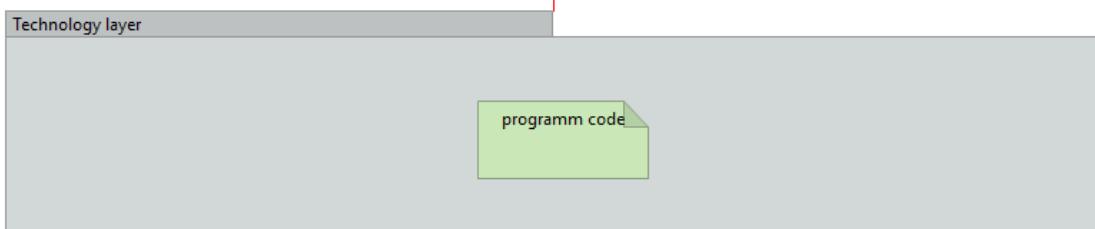


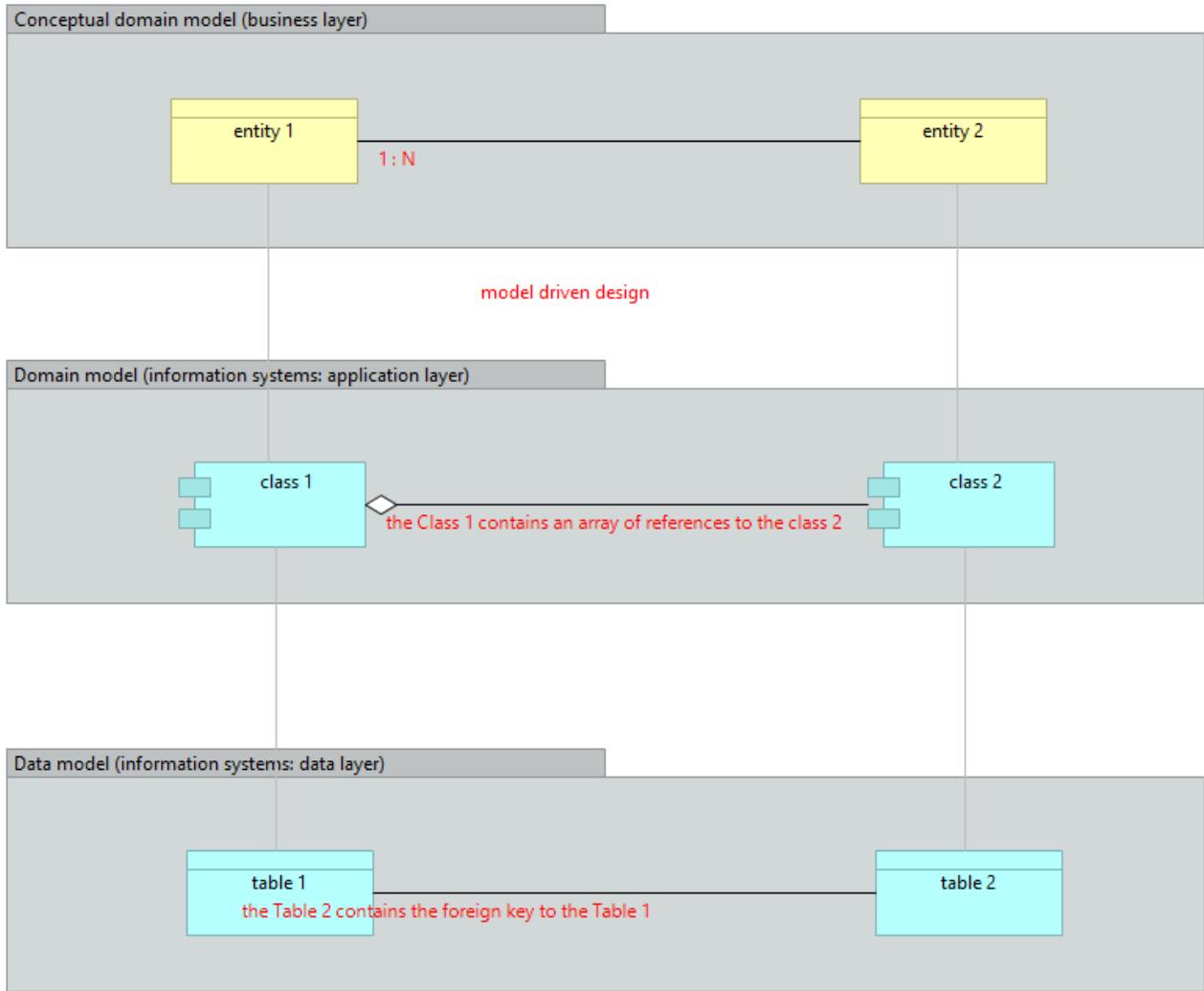
# MODEL-DRIVEN DESIGN

layers - TOGAF's architecture domains



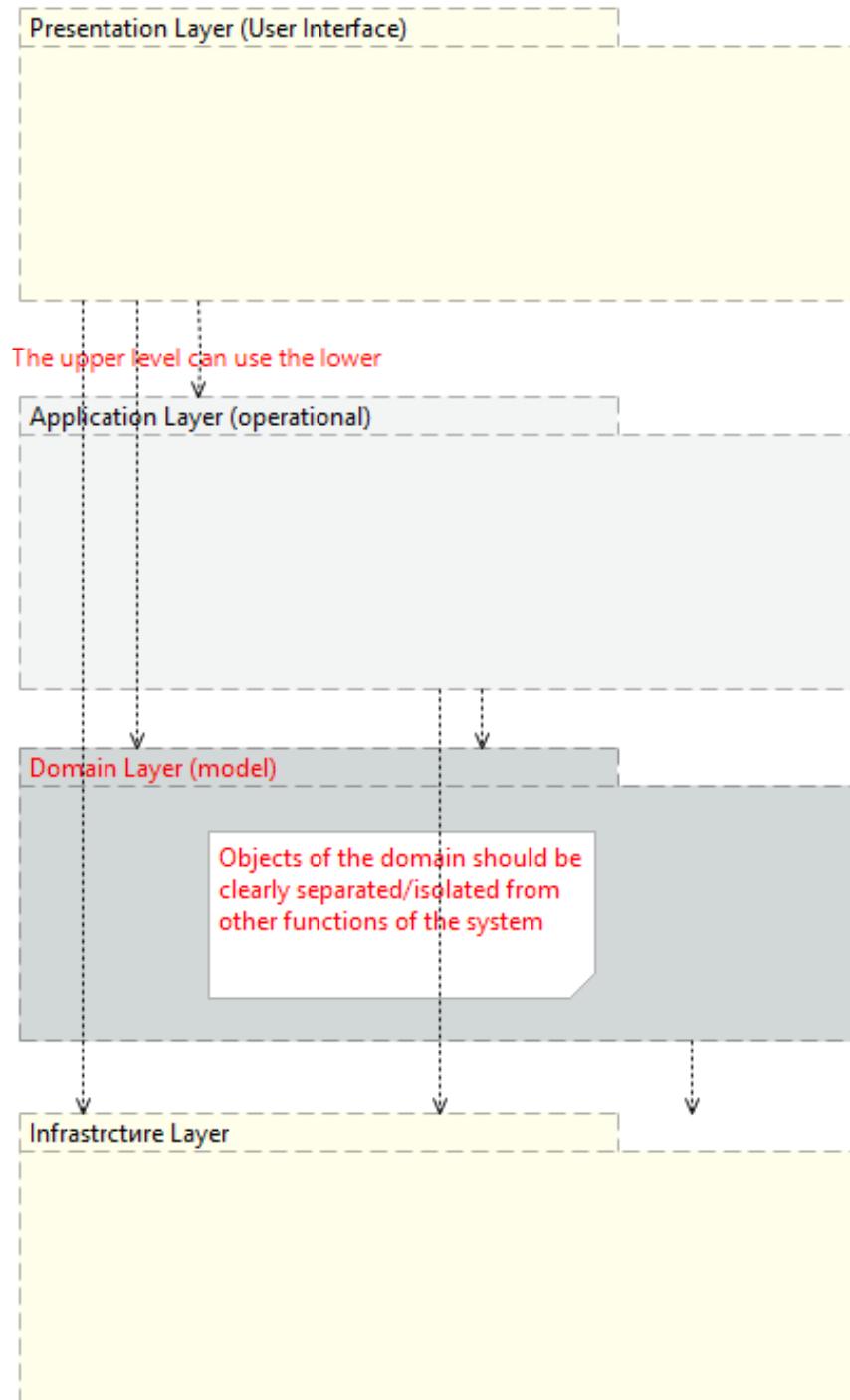
The creation of the model takes place simultaneously with the software implementation



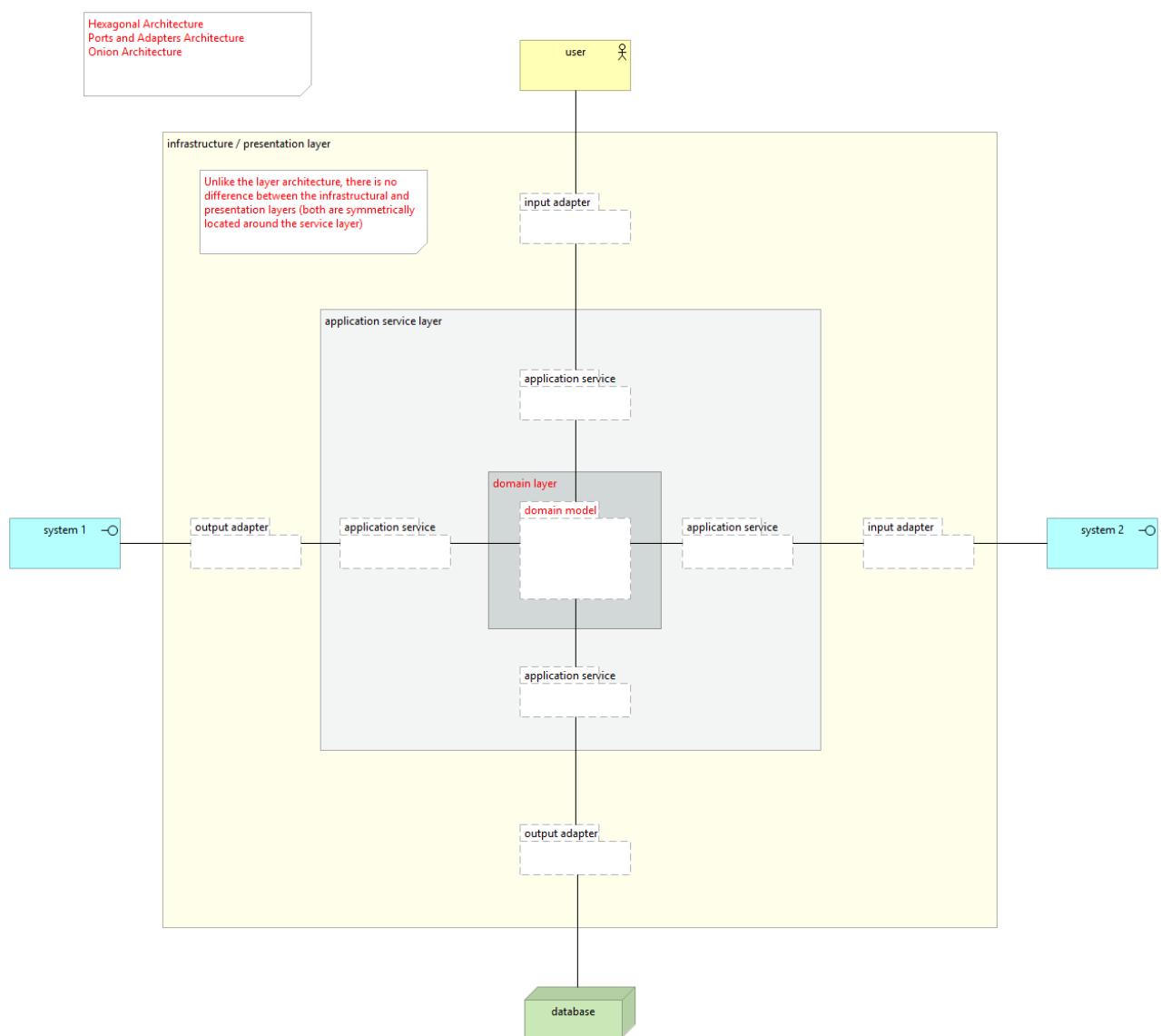


# LAYERED ARCHITECTURE (ASYMMETRIC )

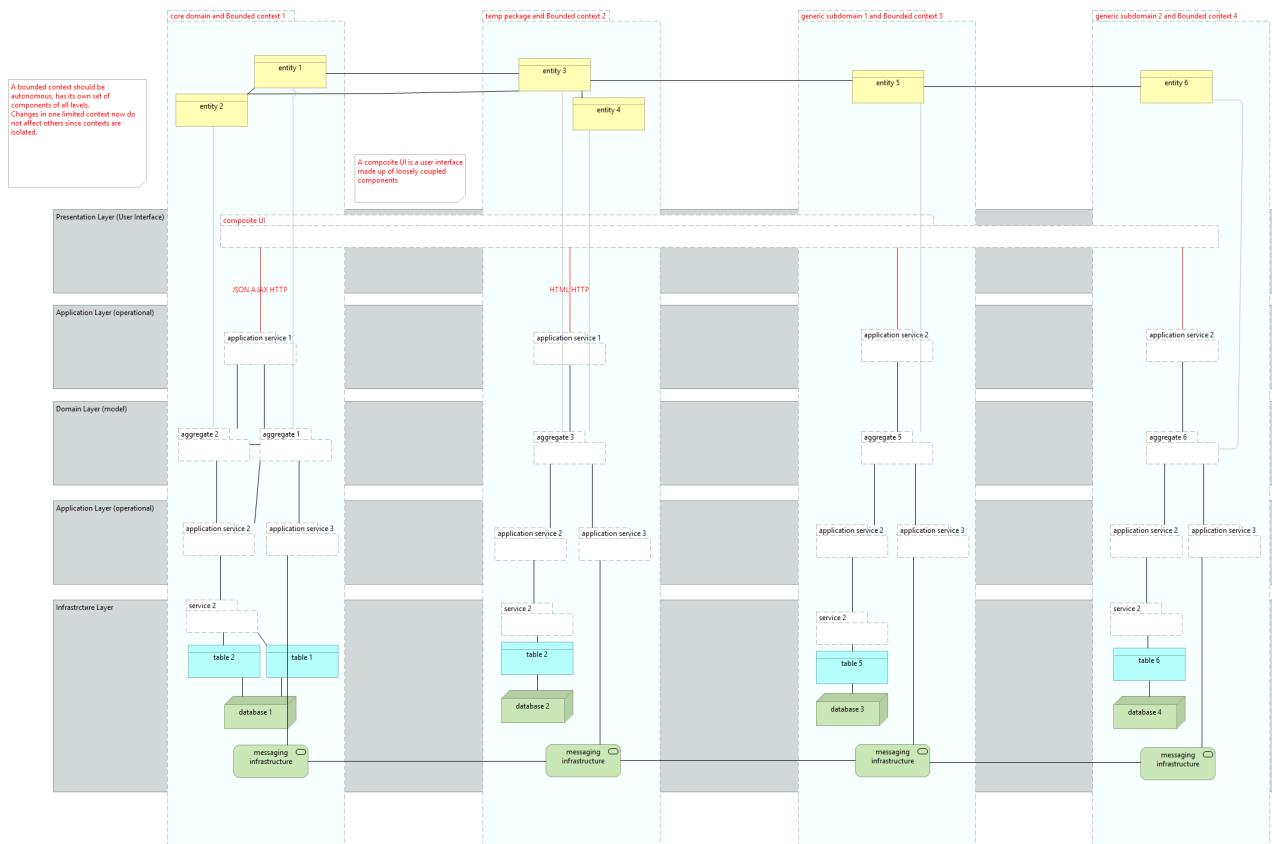
layered architecture  
Only domain layer is required



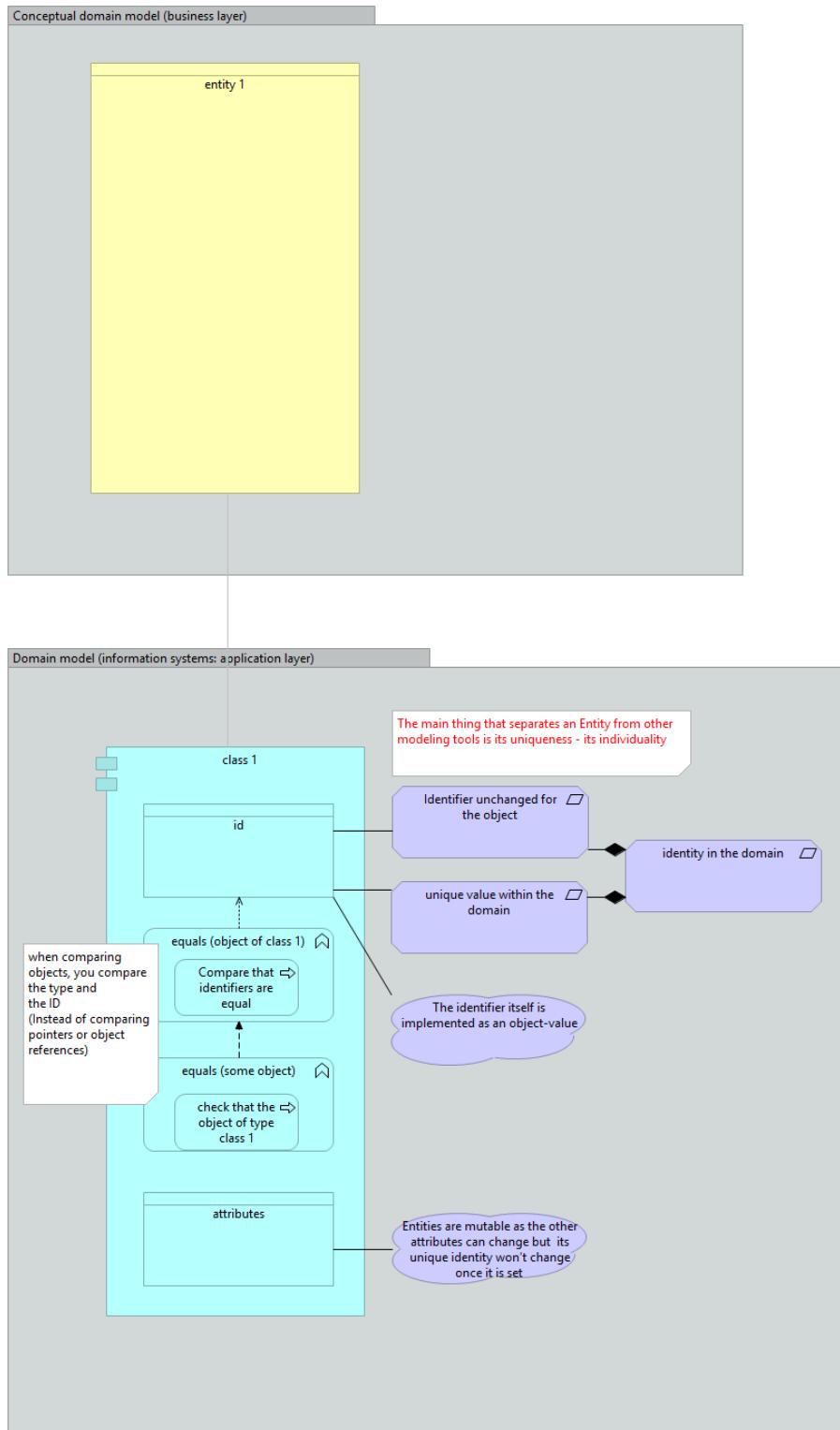
# HEXAGONAL ARCHITECTURE (SYMMETRIC)

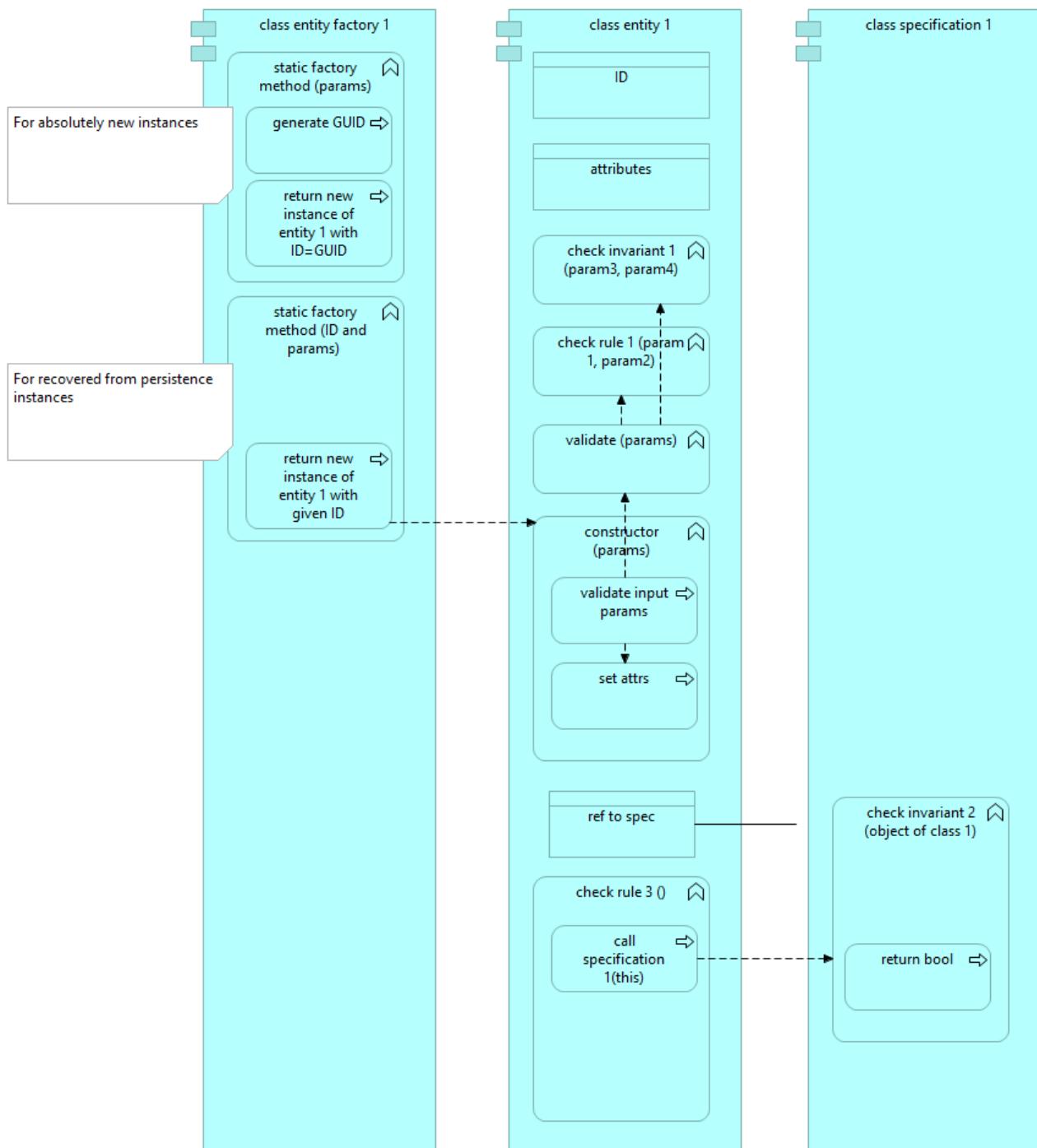


# COMPOSITE UI

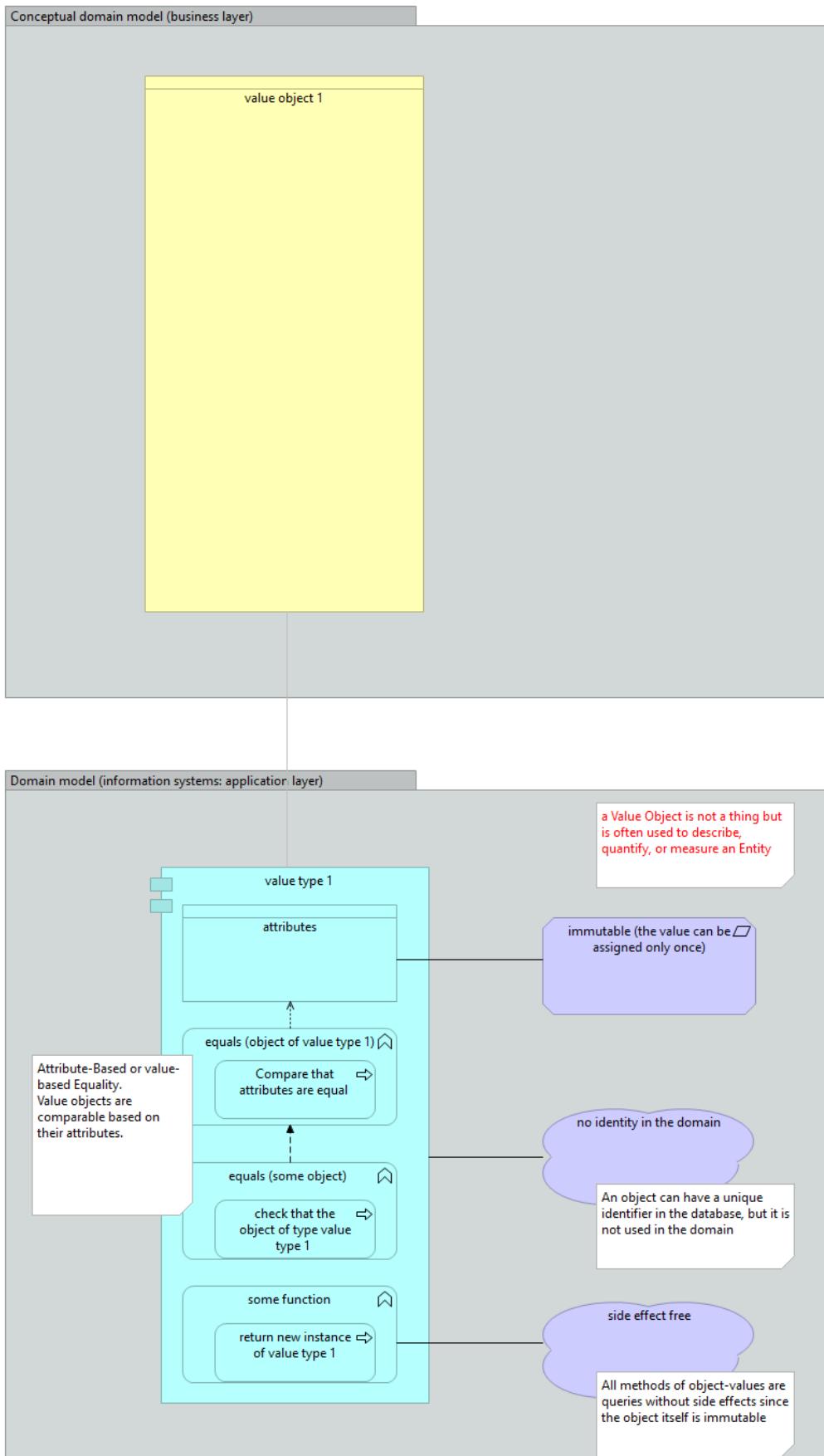


# ENTITIES

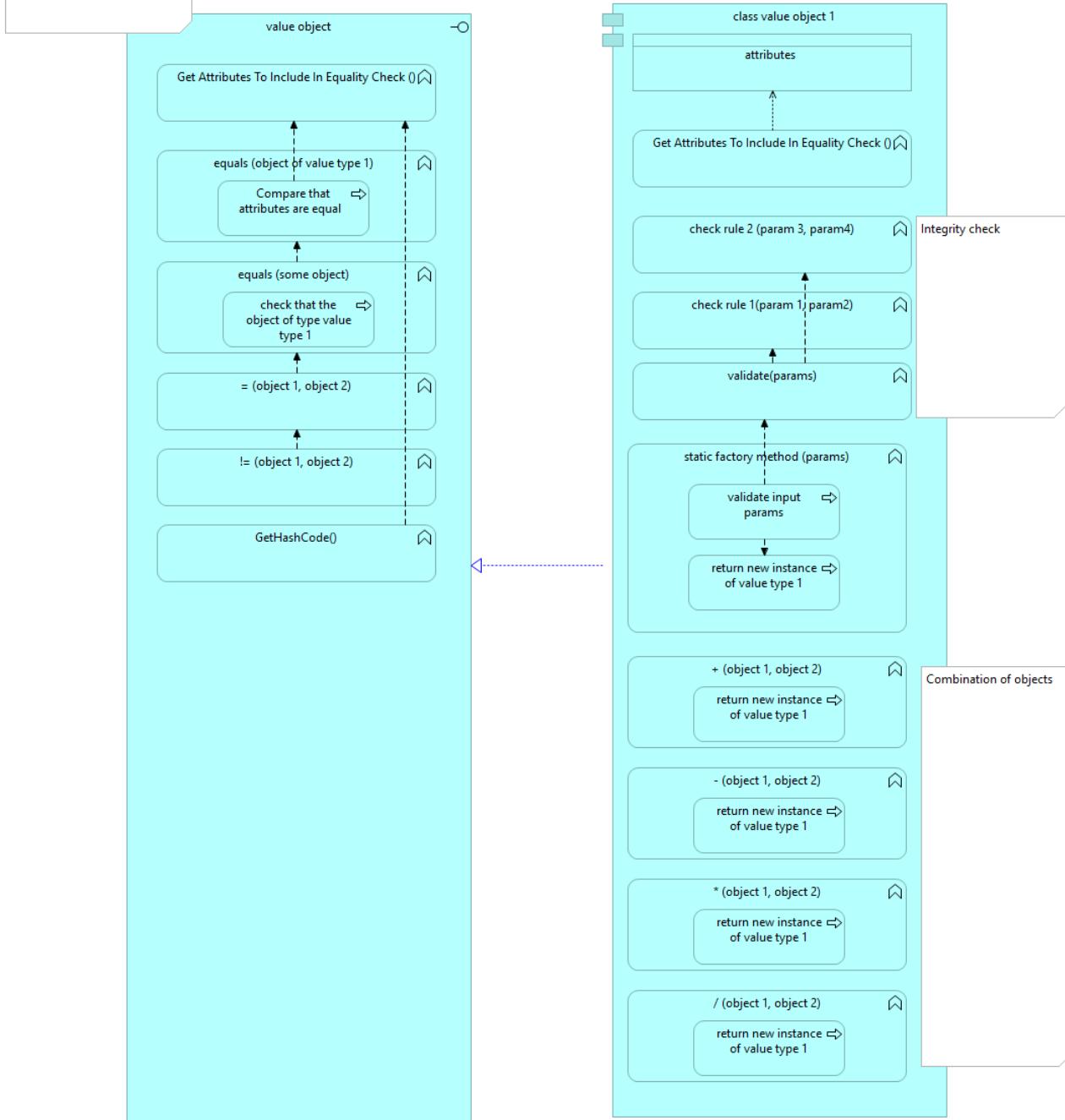




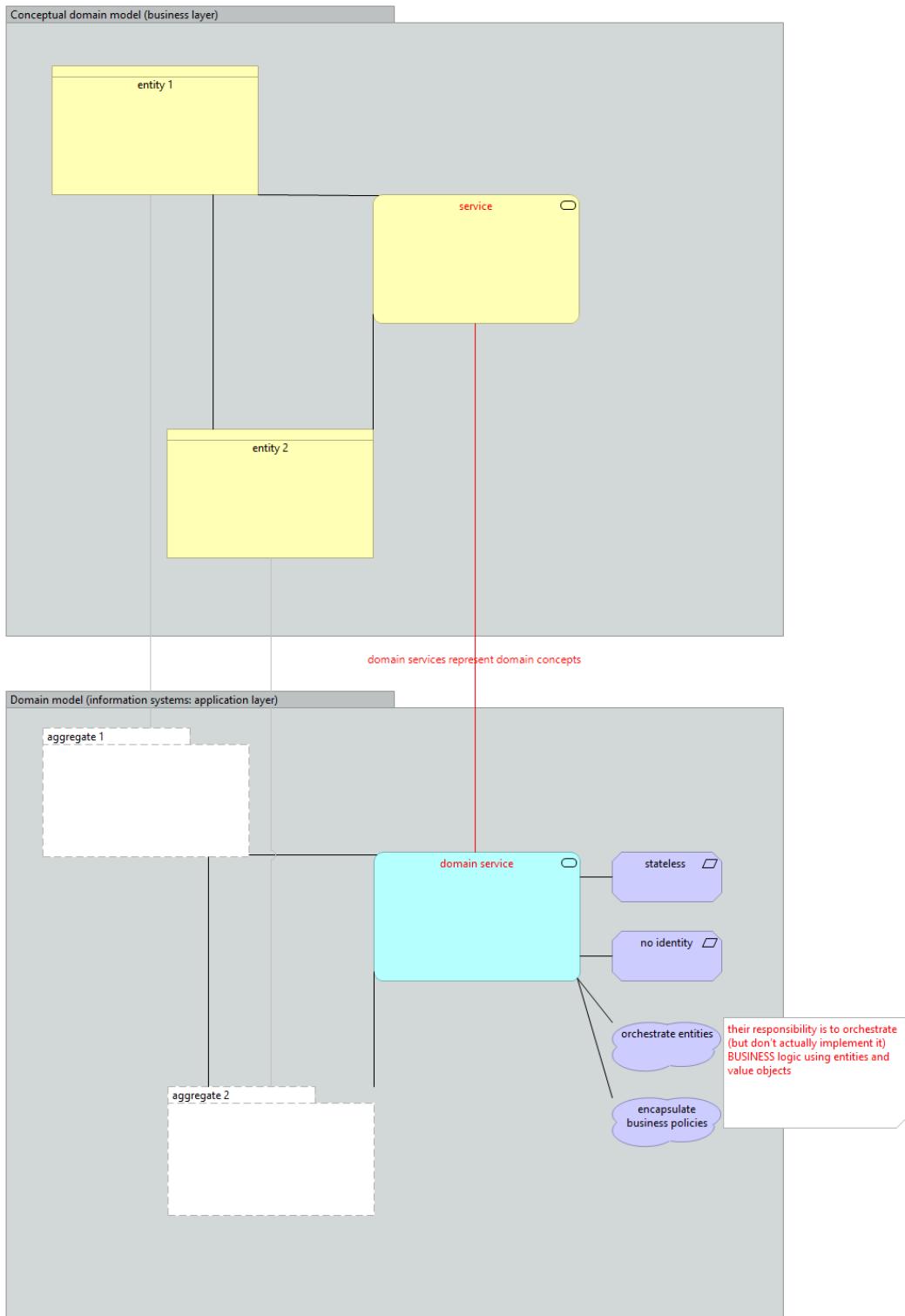
# VALUE-OBJECTS



See example: Fowler's Money

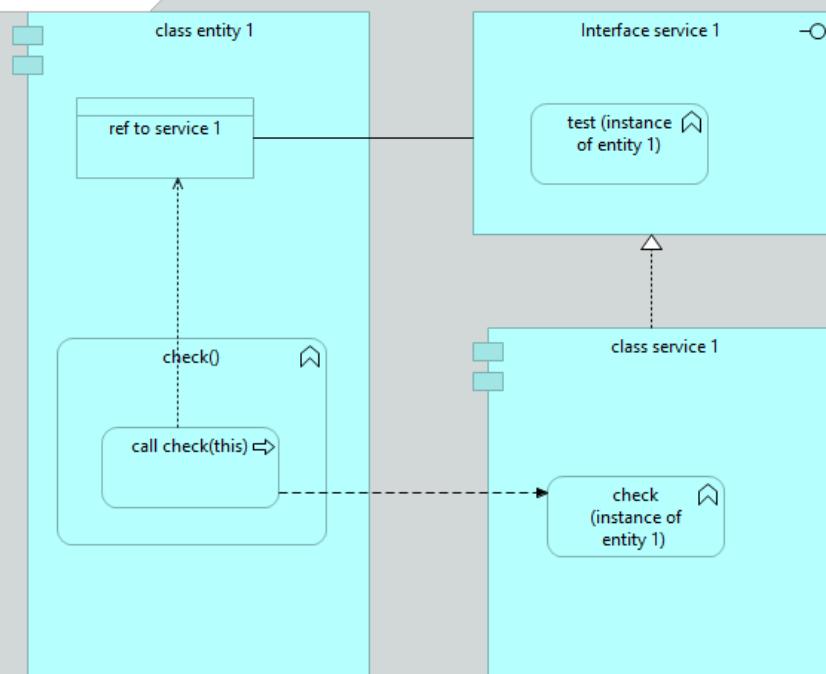


# DOMAIN SERVICES

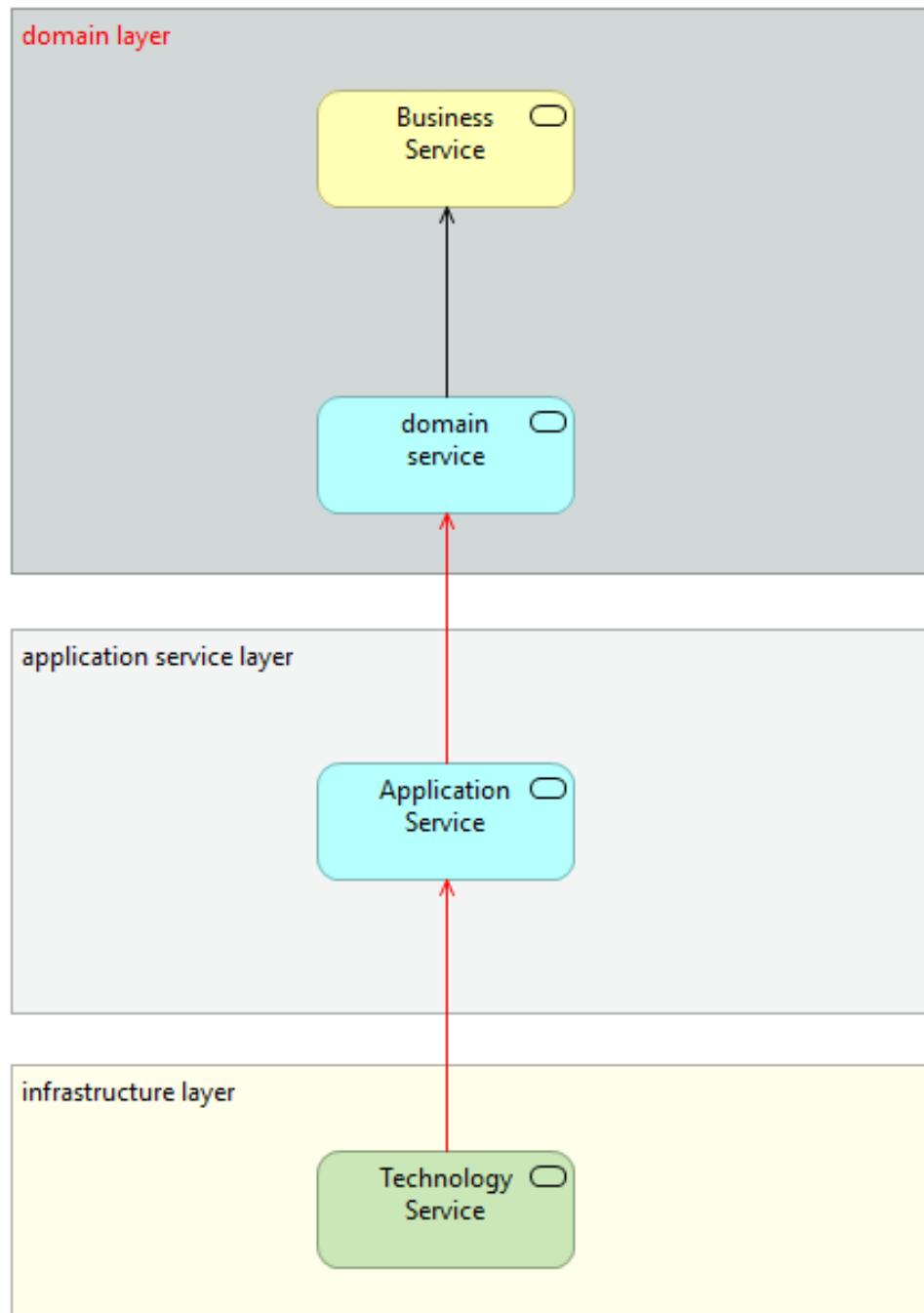


Domain model (information systems: application layer)

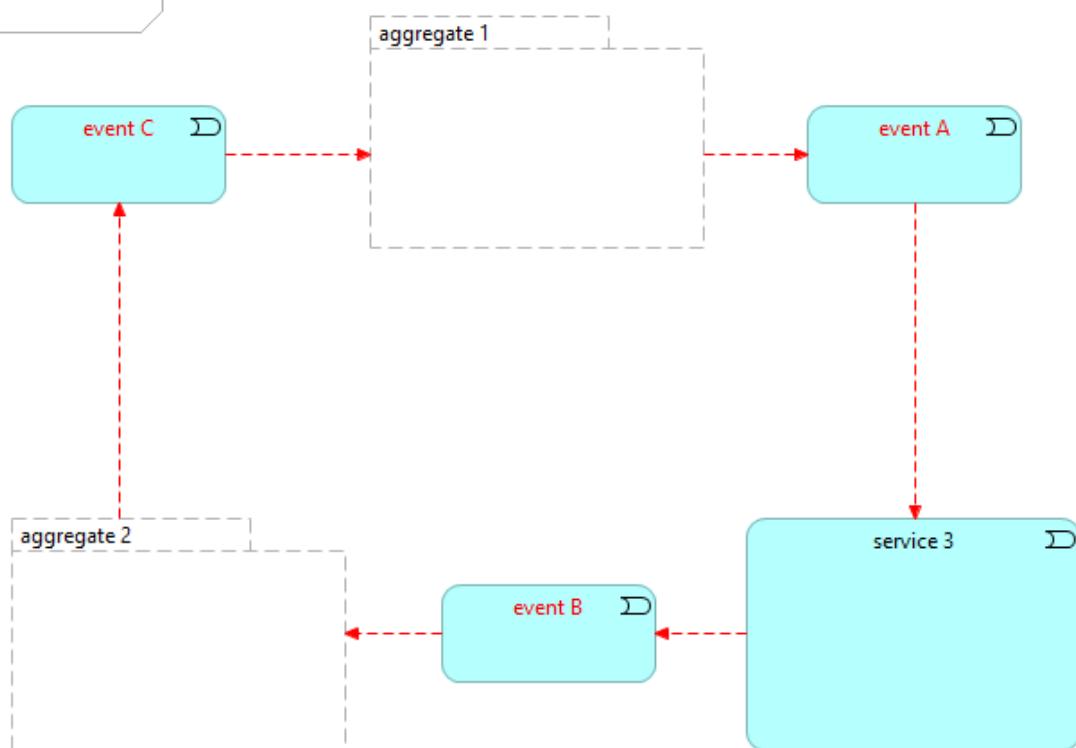
Example of communicating services with aggregates via direct service call



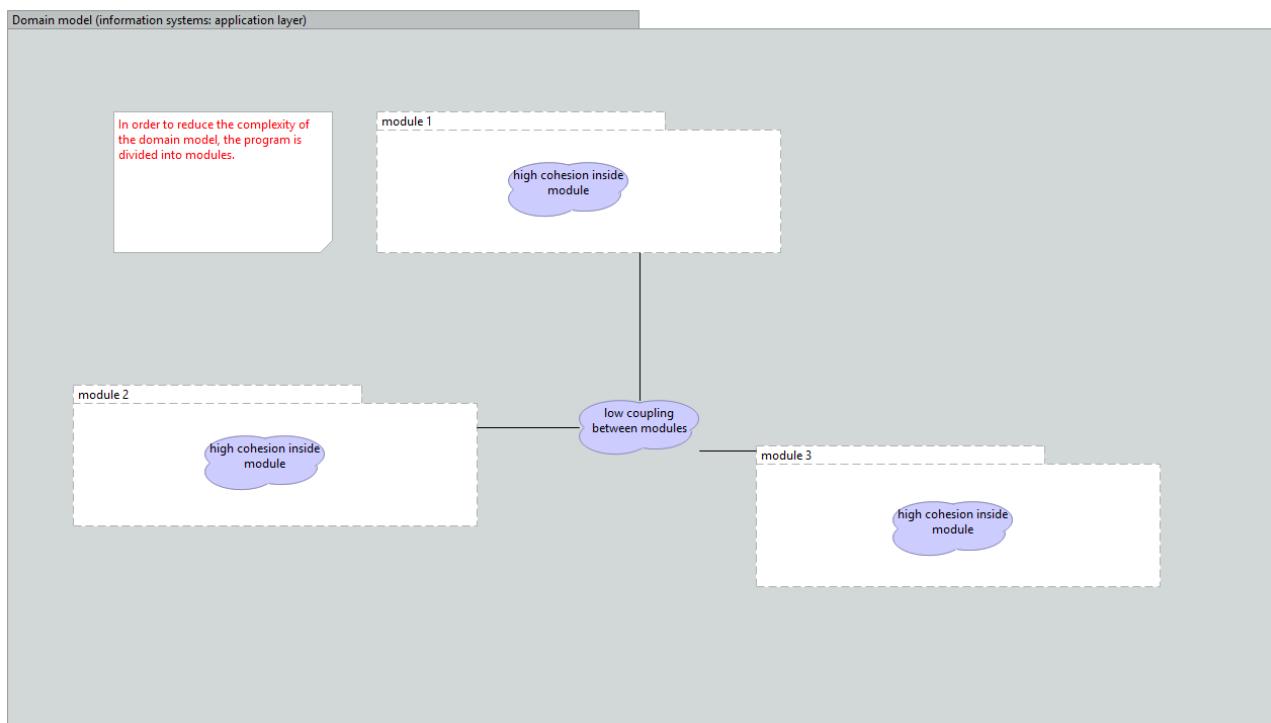
Services support each other



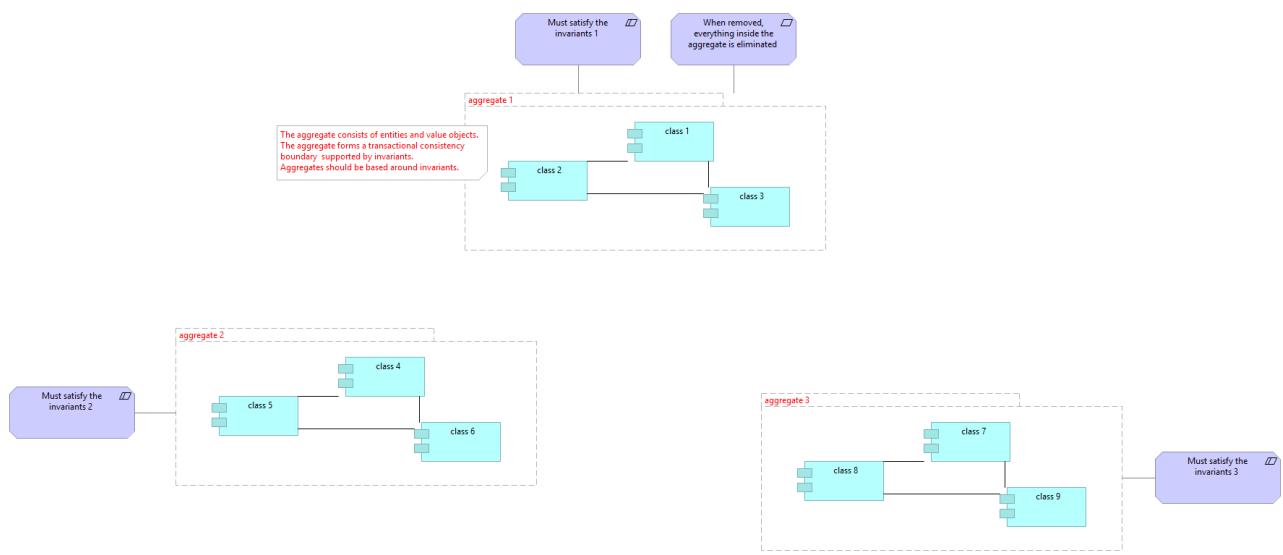
Example of communicating services with aggregates through events



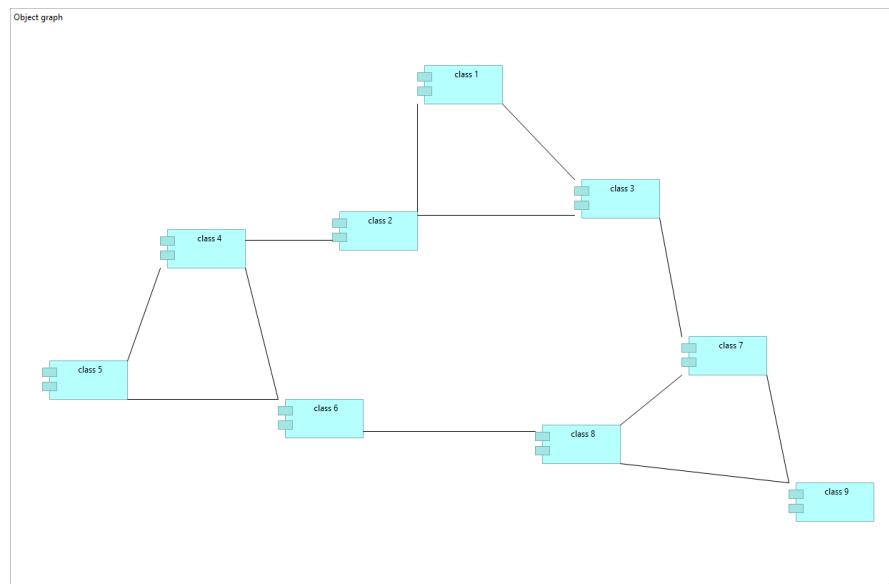
# MODULES



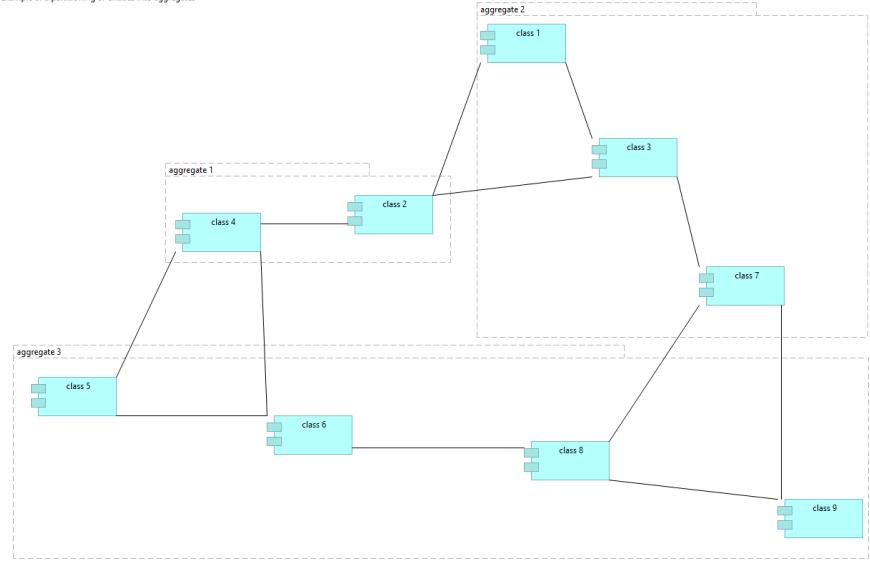
# AGGREGATES



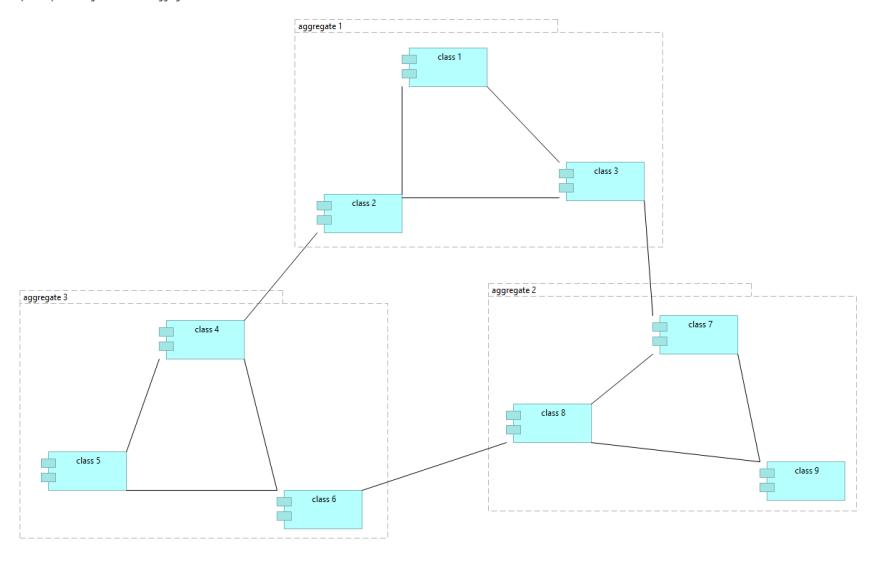
In the course of modeling, think about different ways of splitting the model into aggregates



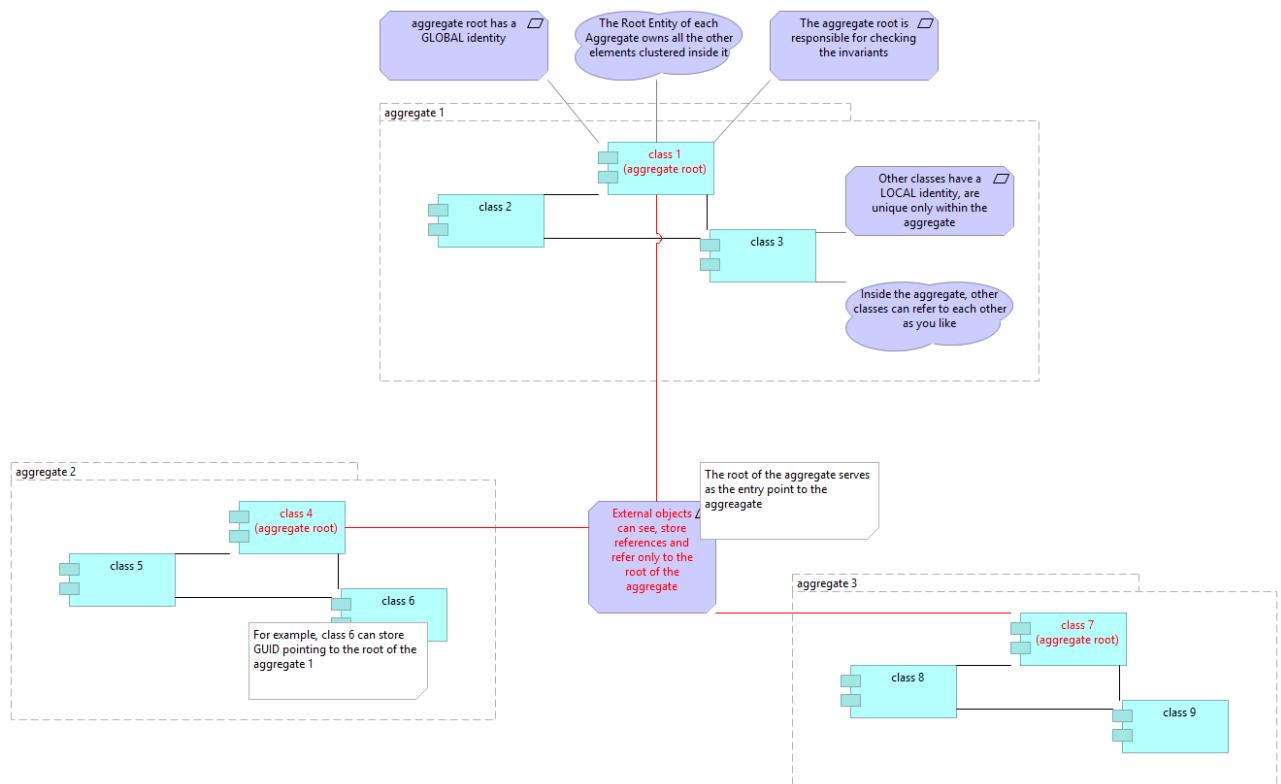
Example of a partitioning of entities into aggregates



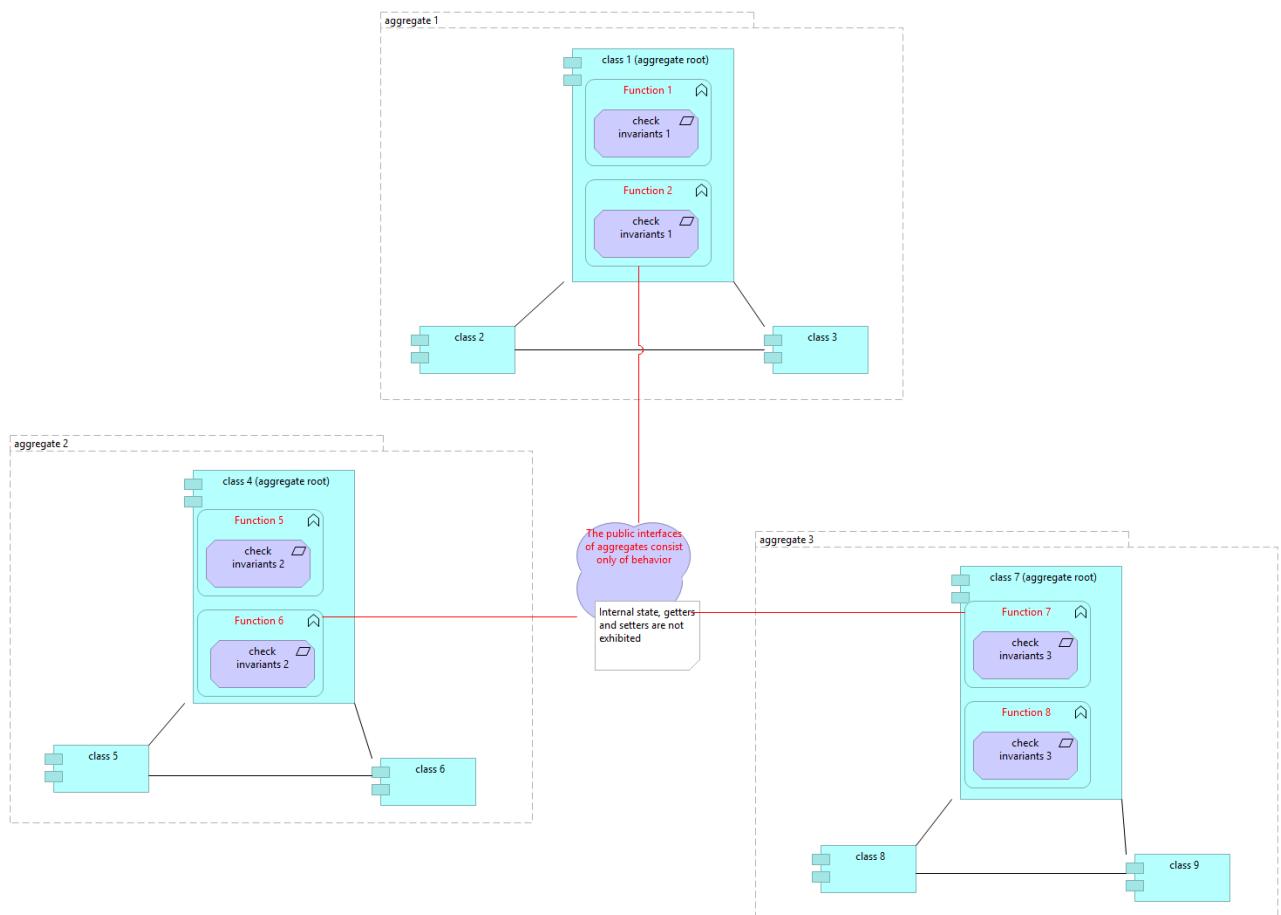
Example of a partitioning of entities into aggregates



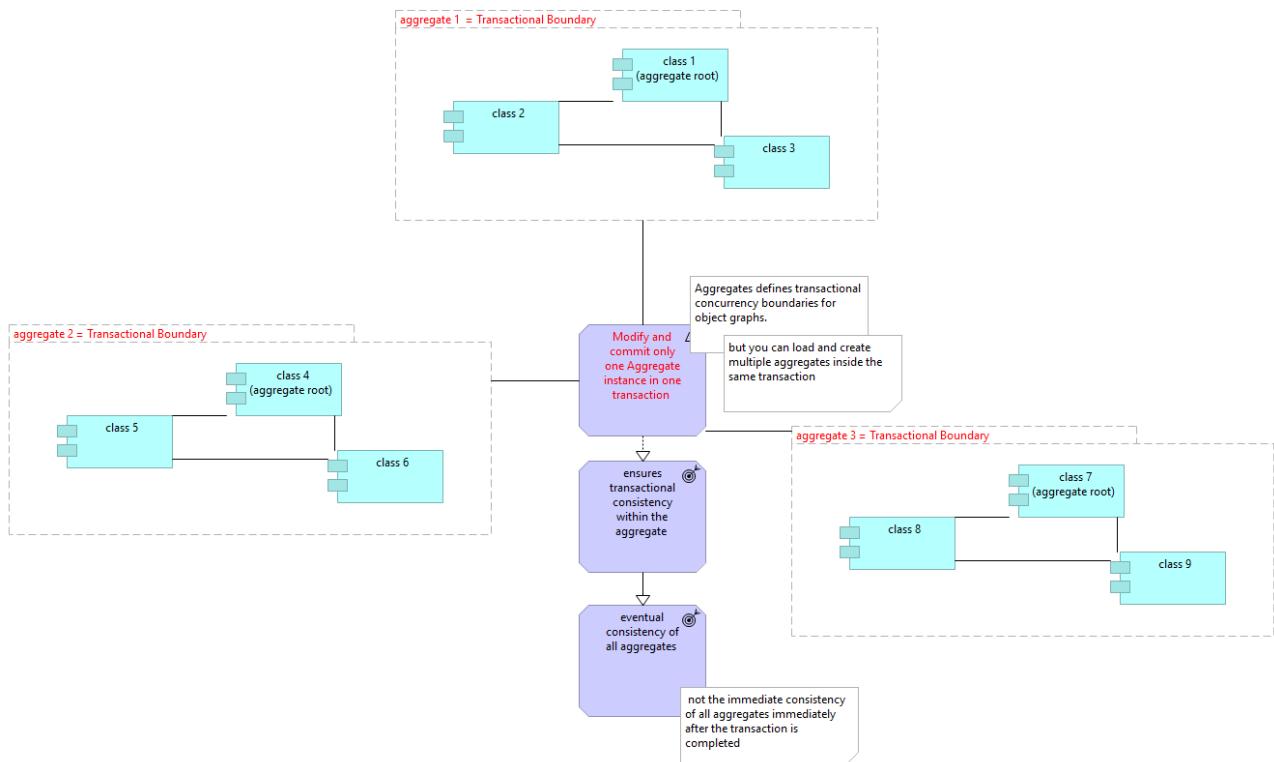
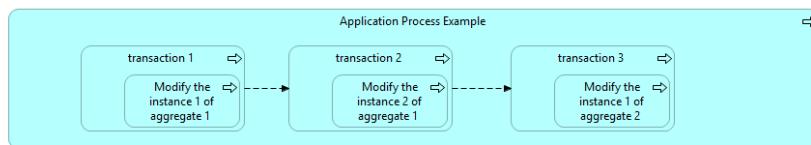
# AGGREGATE ROOT



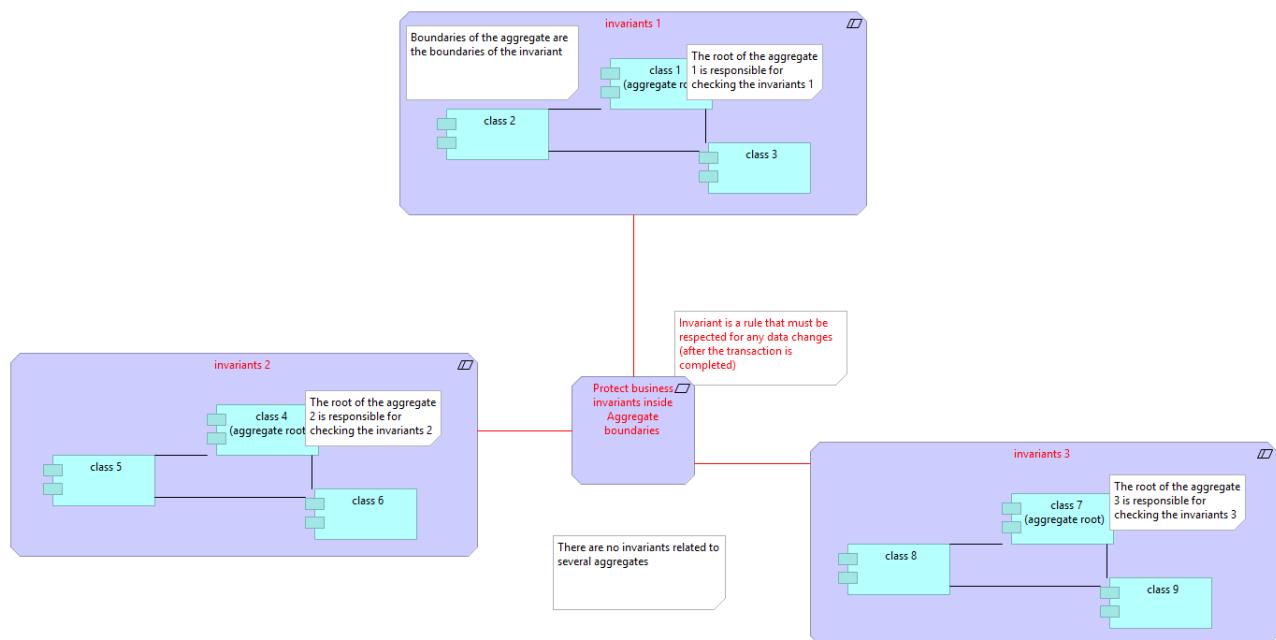
# BEHAVIOR-FOCUSED AGGREGATE ROOT



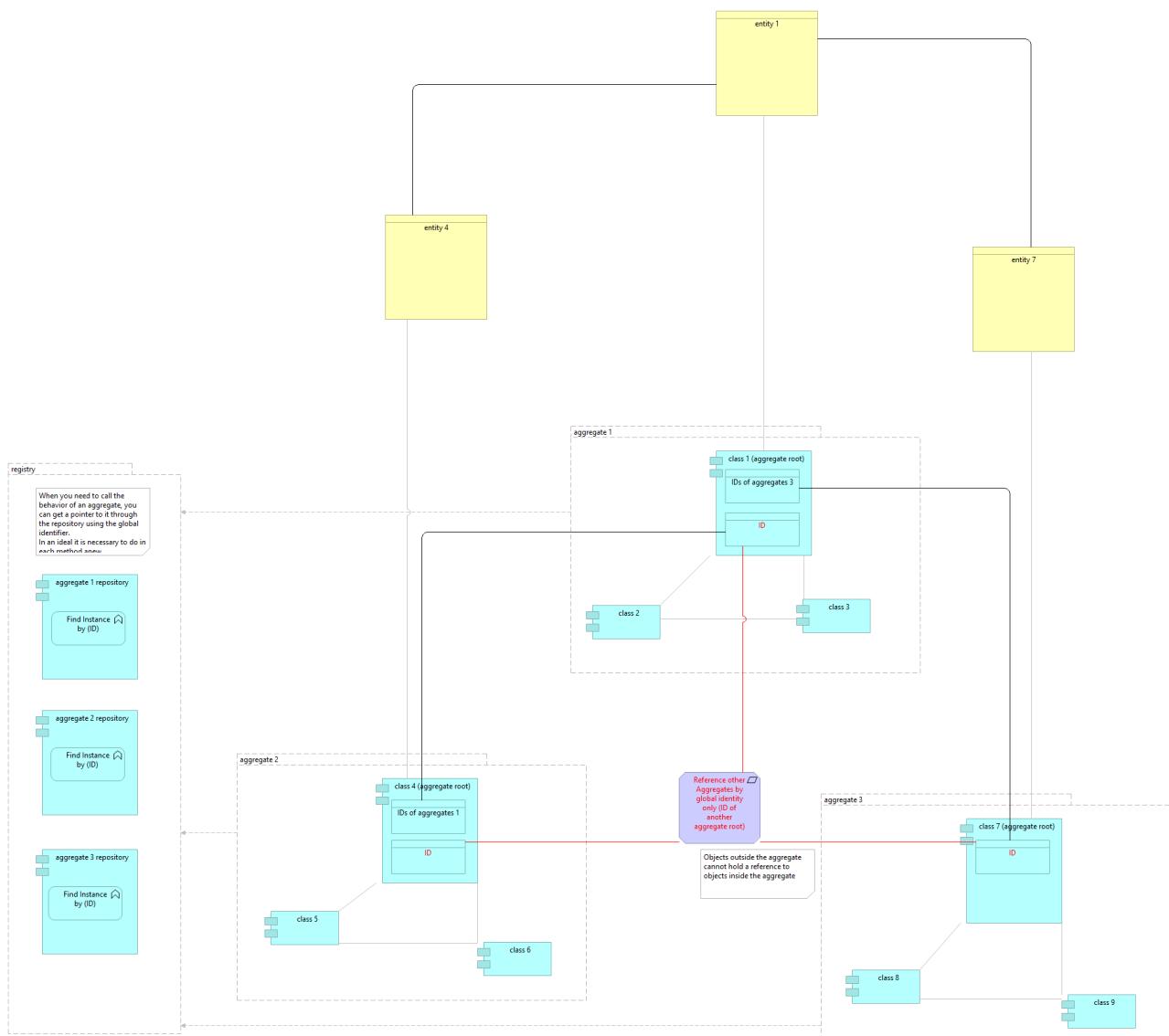
# MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION



# PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES



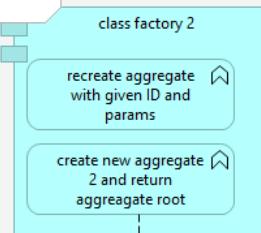
# REFERENCE OTHER AGGREGATES BY IDENTITY ONLY



# FACTORIES

application service layer

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

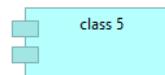


domain layer

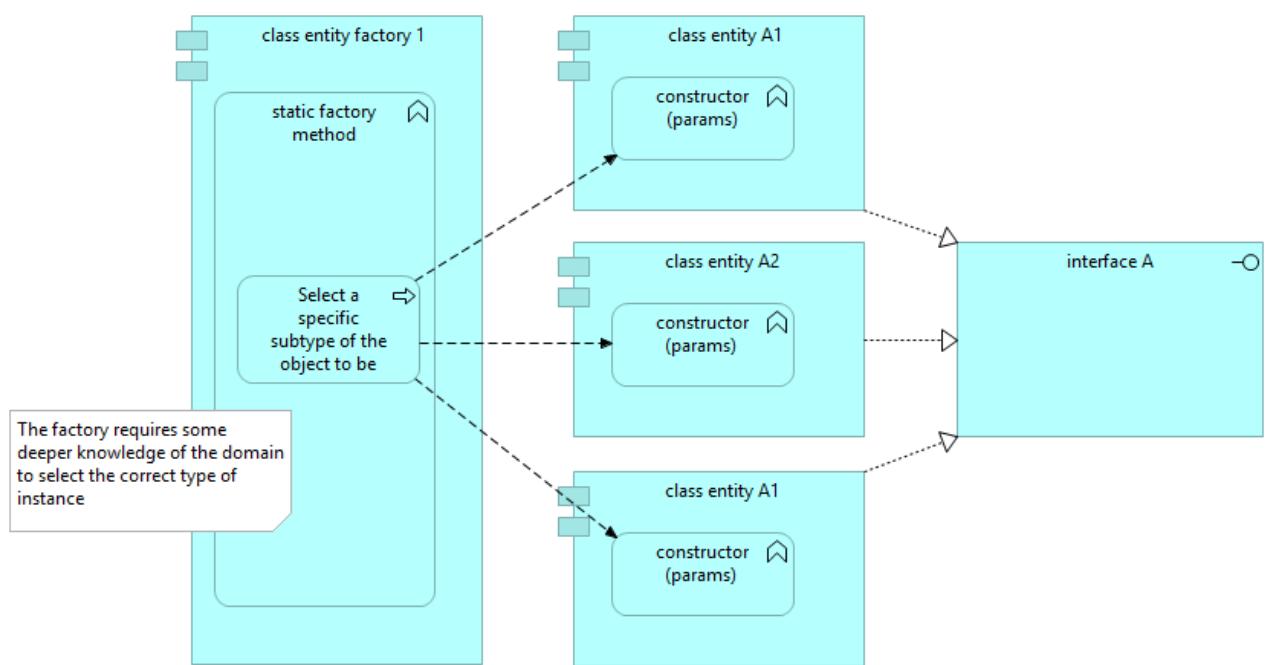
Must satisfy the invariants 2 after creation

aggregate 2

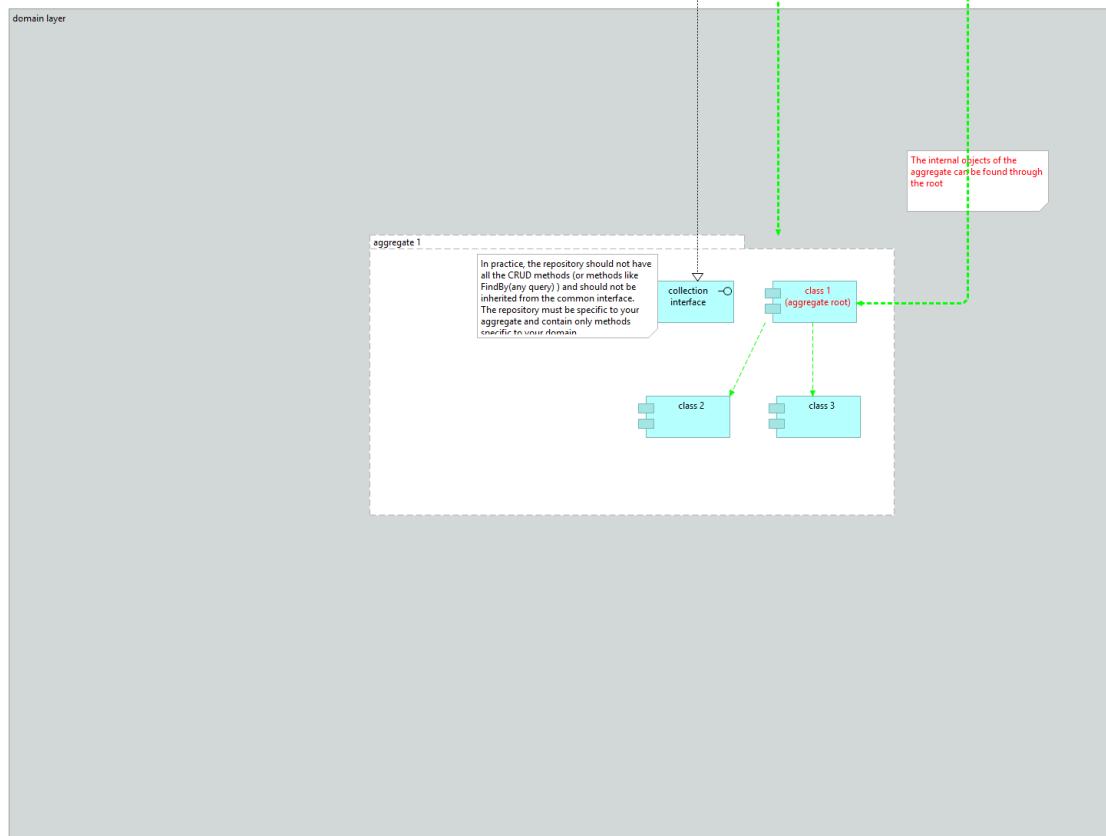
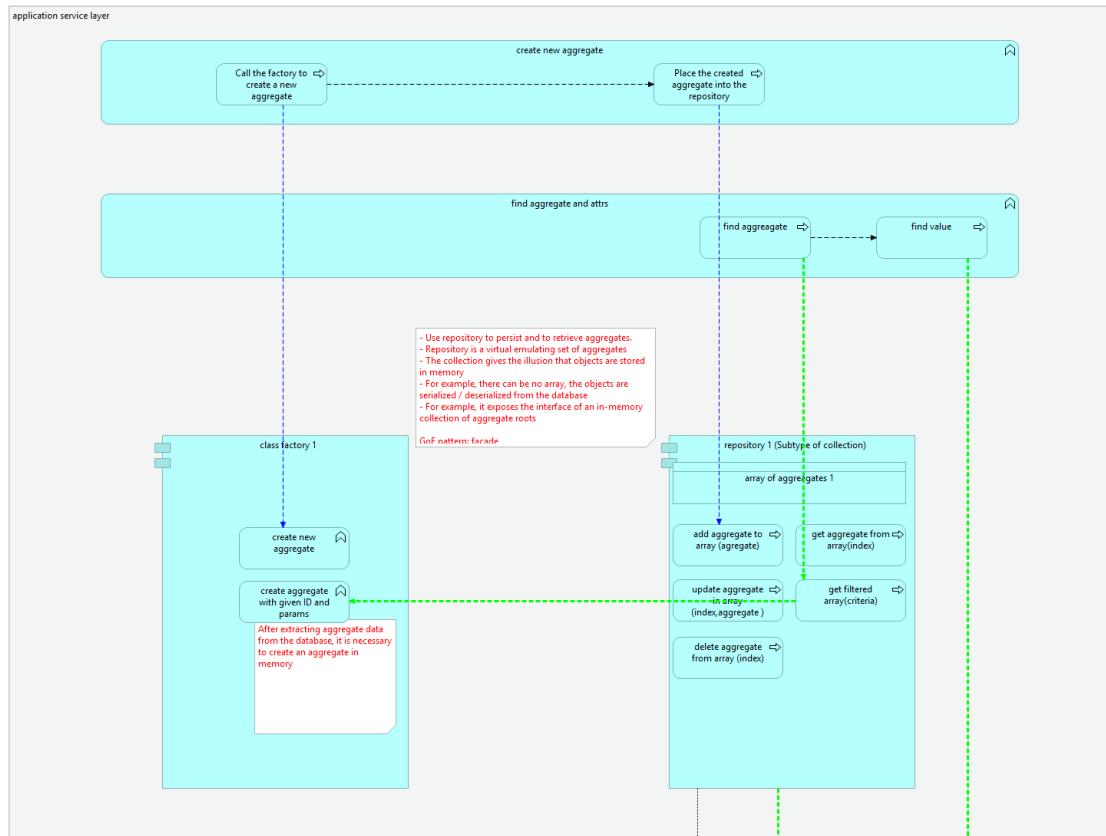
The root of the aggregate itself creates all internal objects

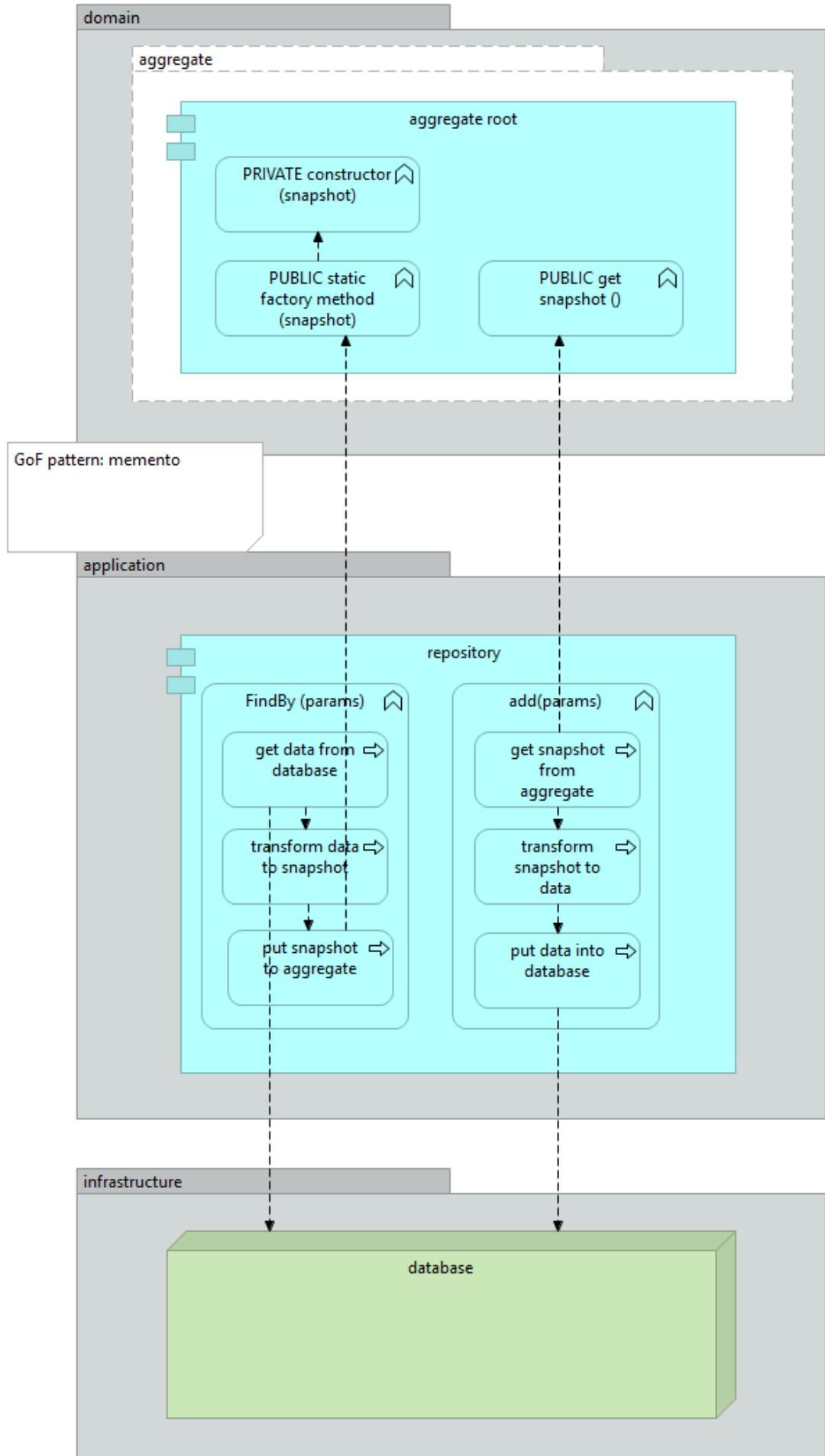


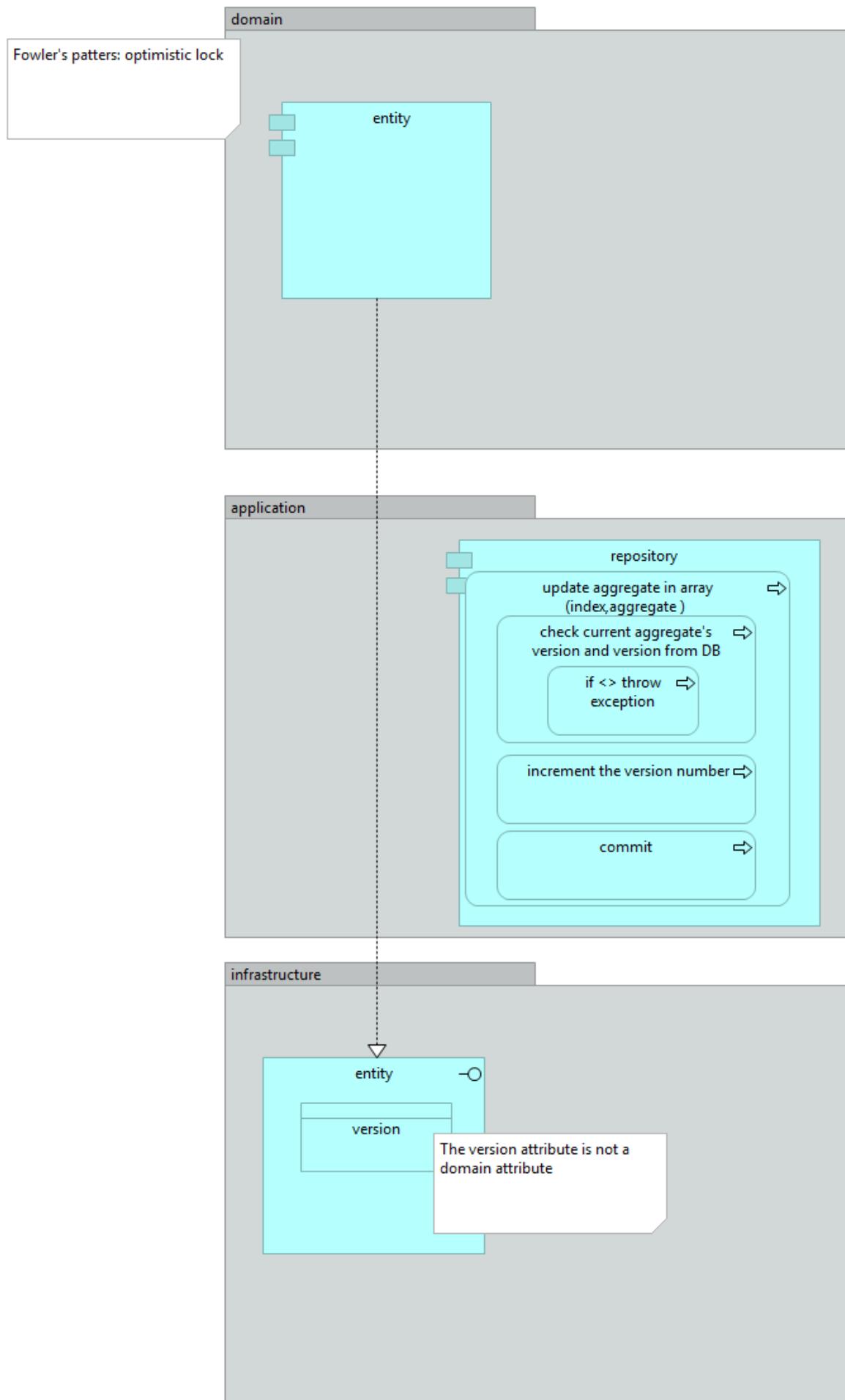
class 6



# REPOSITORIES







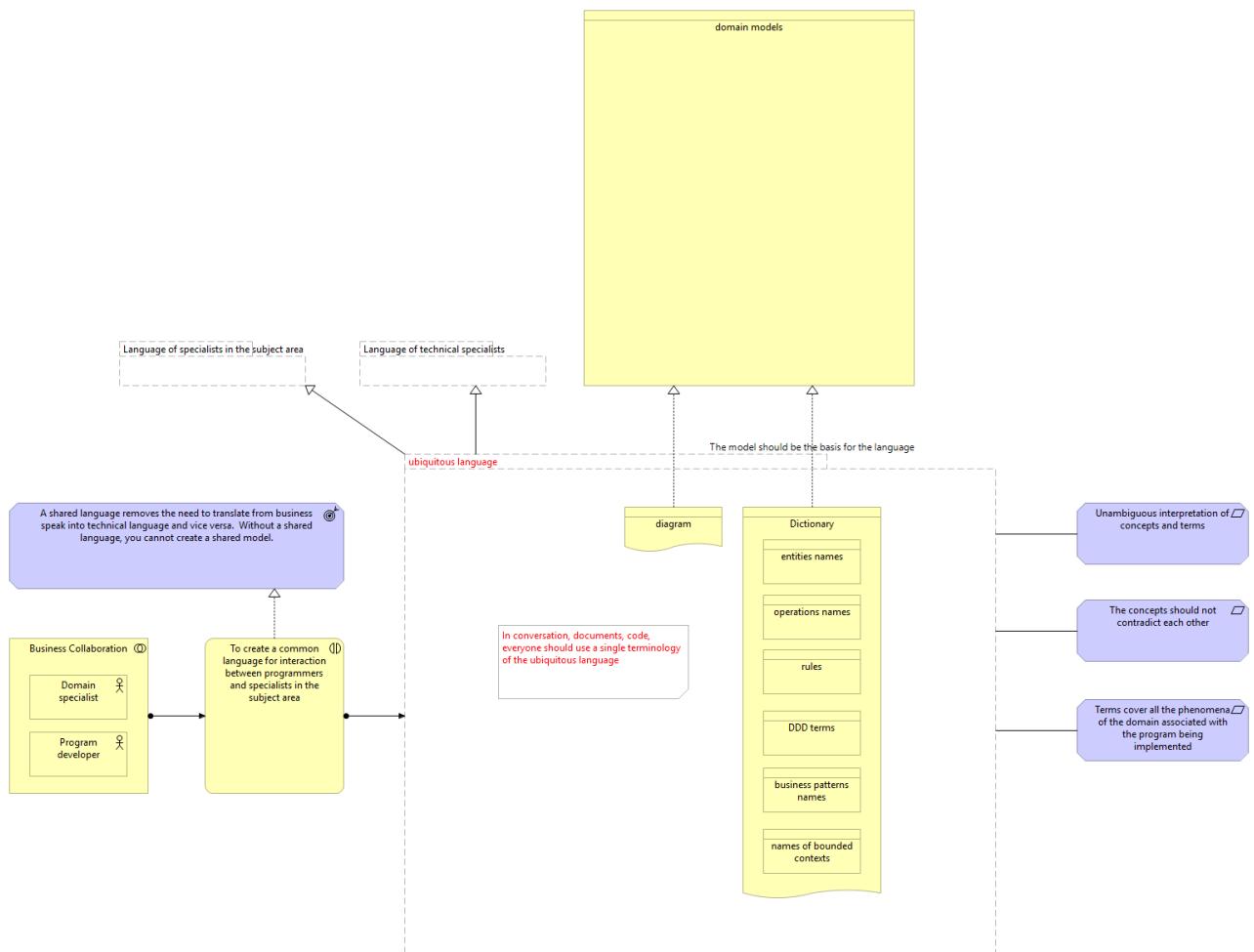
# SUPPLE DESIGN

DOMAIN DRIVEN DESIGN

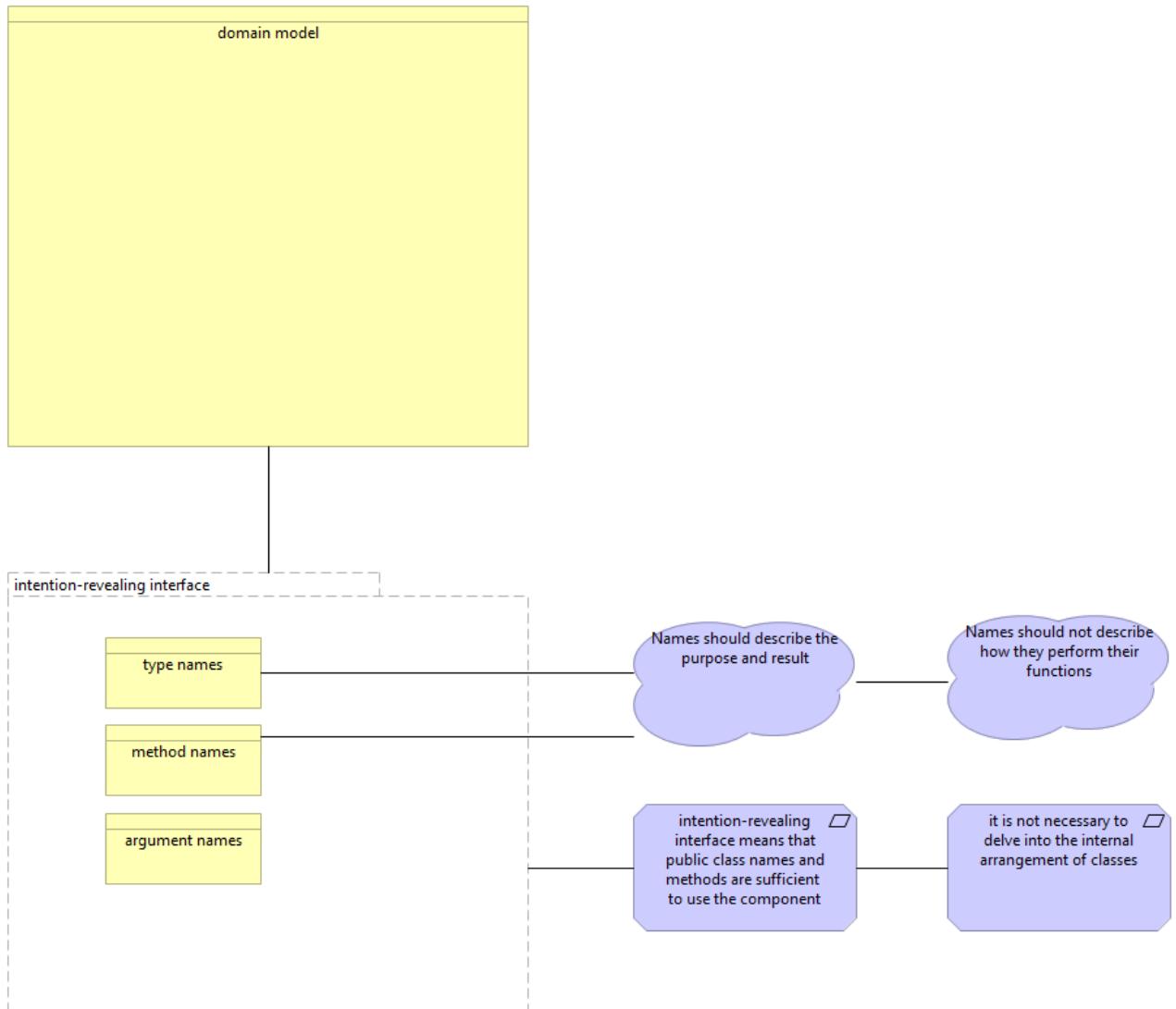
Eric Evans



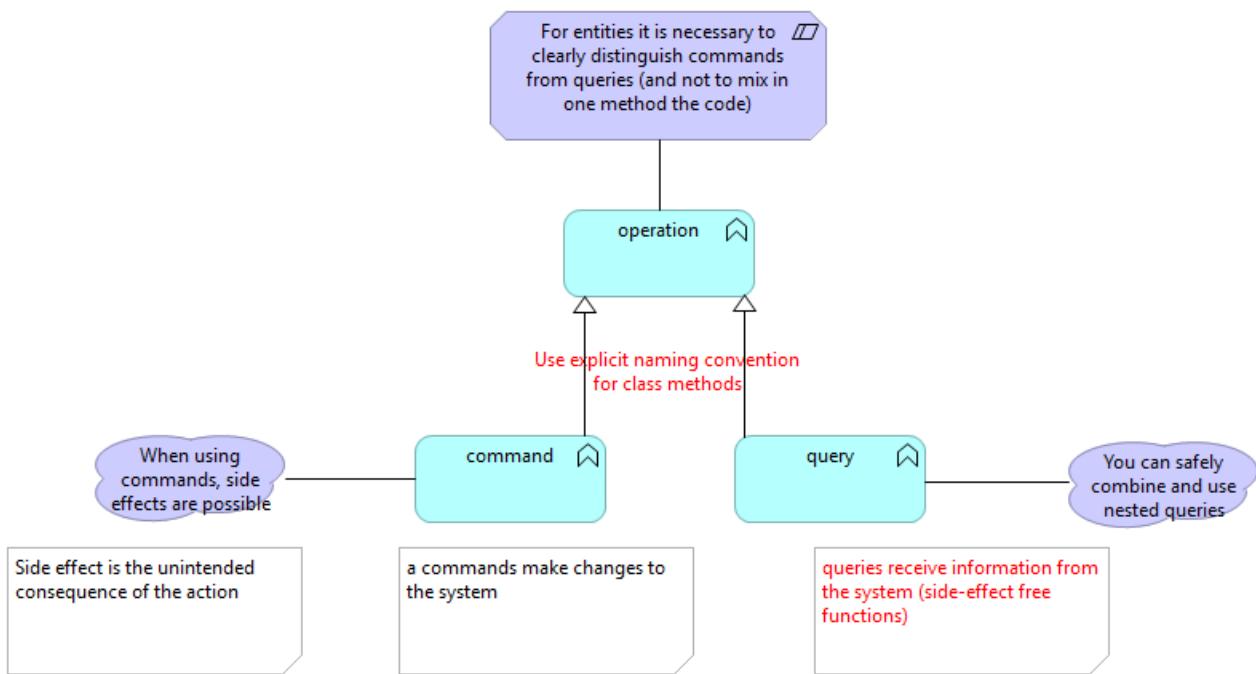
# UBIQUITOUS LANGUAGE



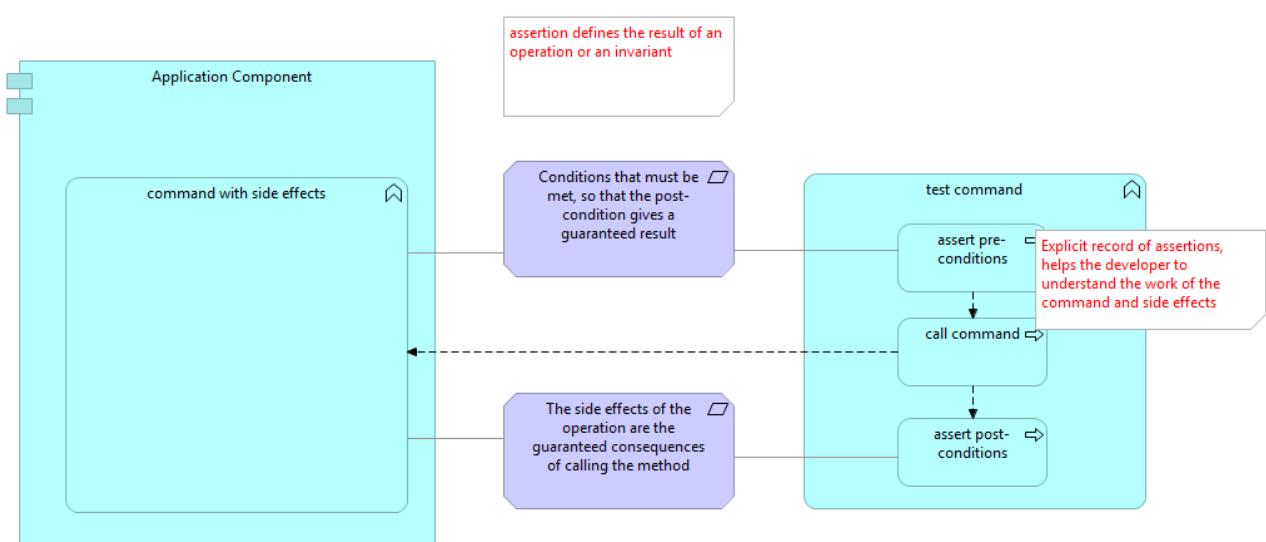
# INTENTION-REVEALING INTERFACES



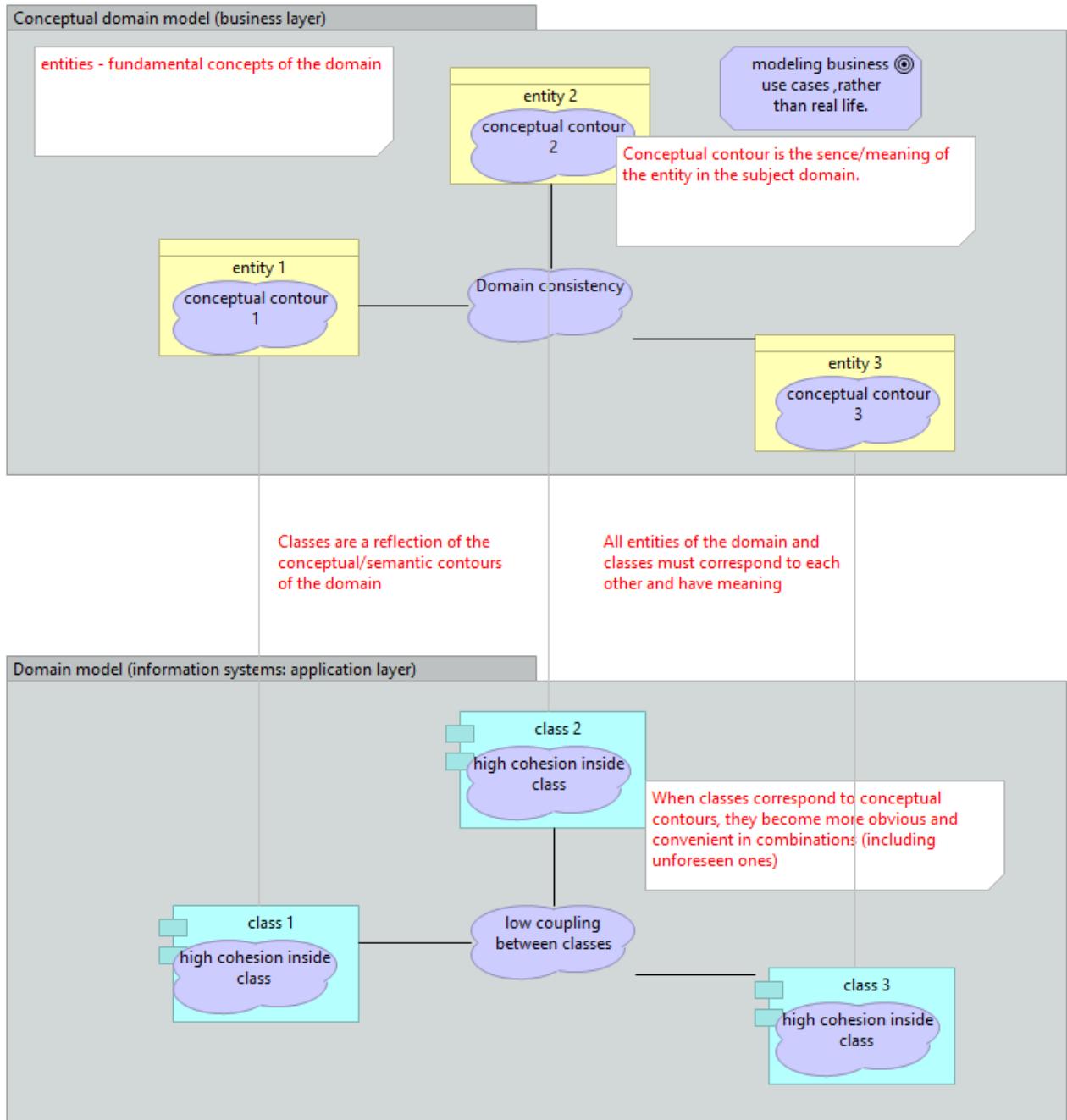
# SIDE-EFFECT FREE FUNCTIONS



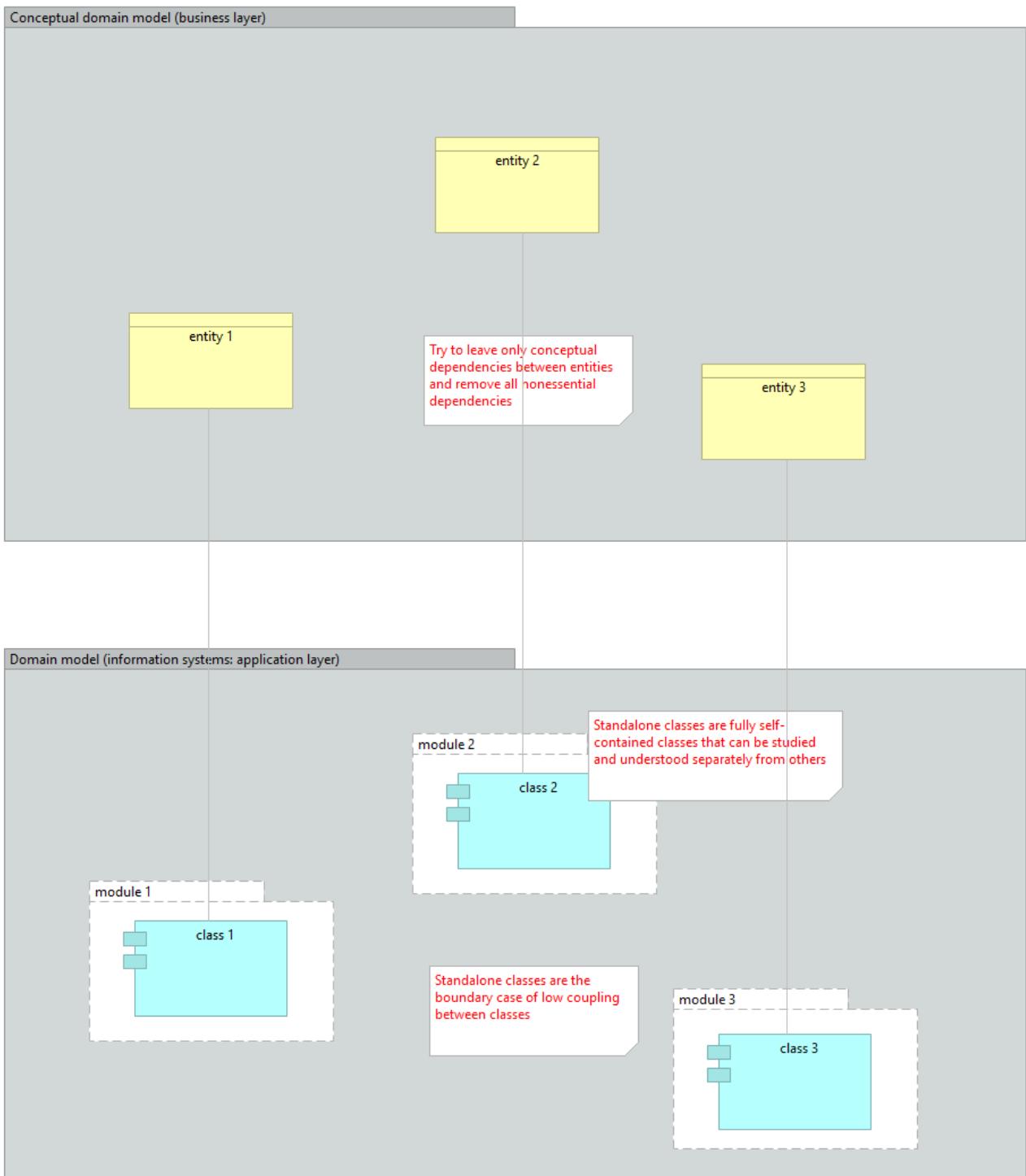
# ASSERTIONS



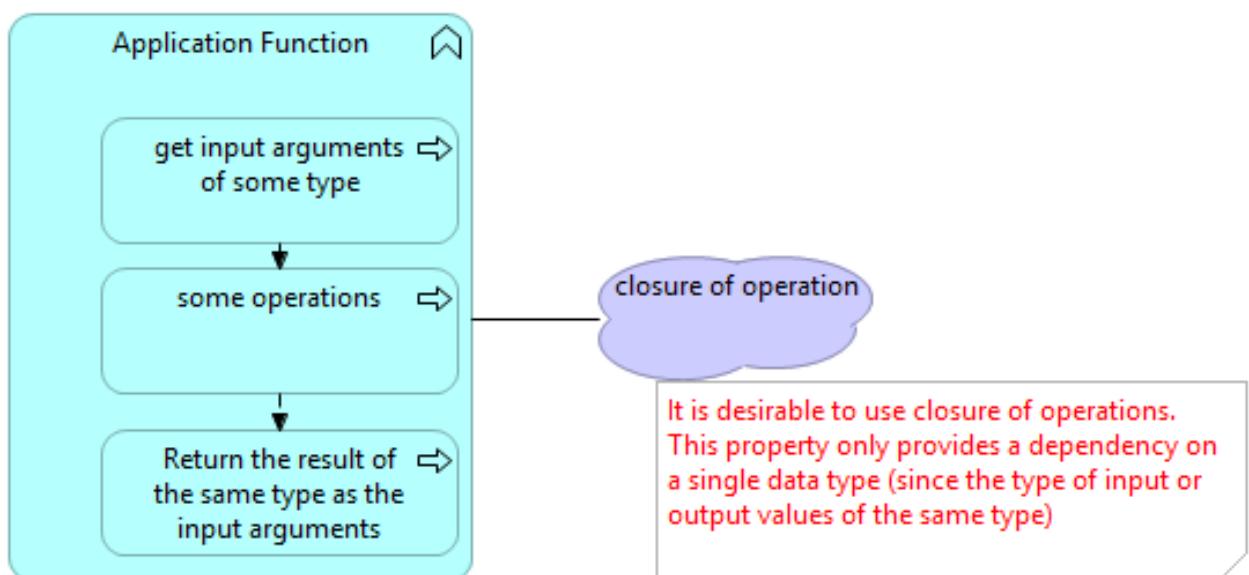
# CONCEPTUAL CONTOURS



# STANDALONE CLASSES



# CLOSURE OF OPERATIONS



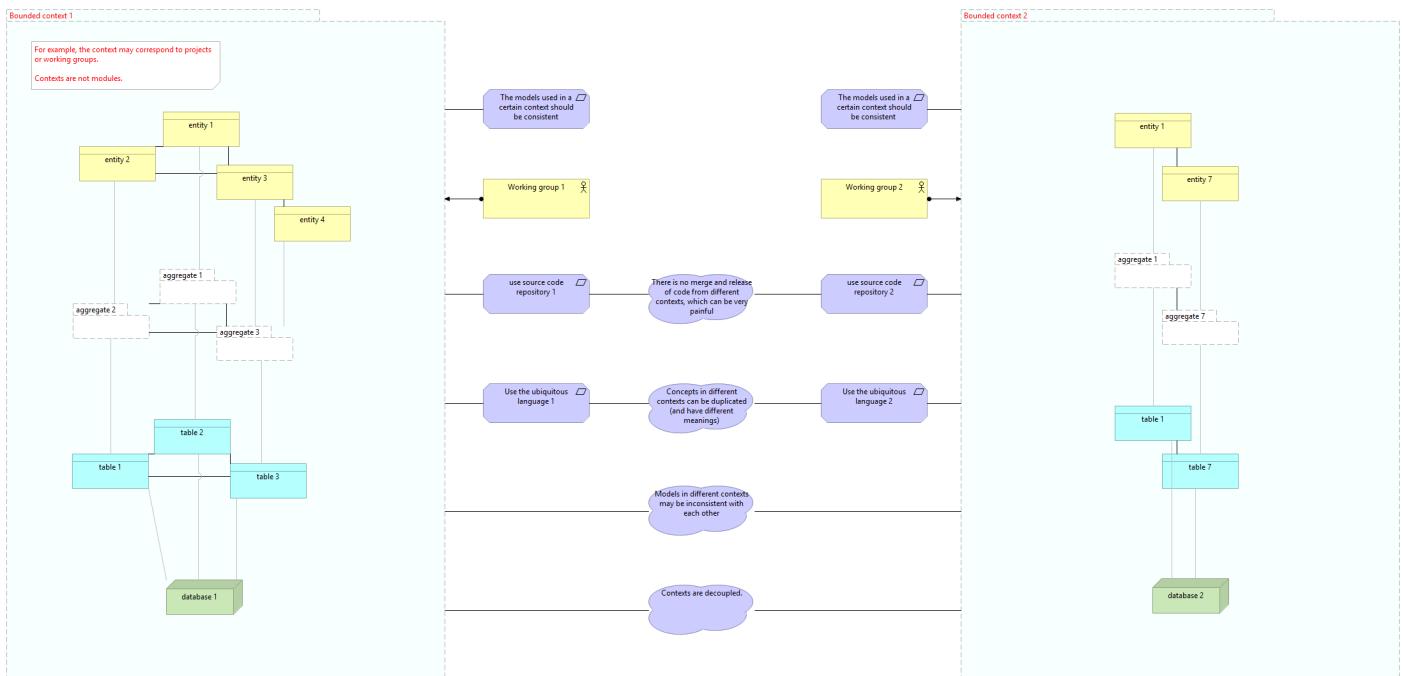
# MODEL INTEGRITY AND CONTEXT

DOMAIN DRIVEN DESIGN

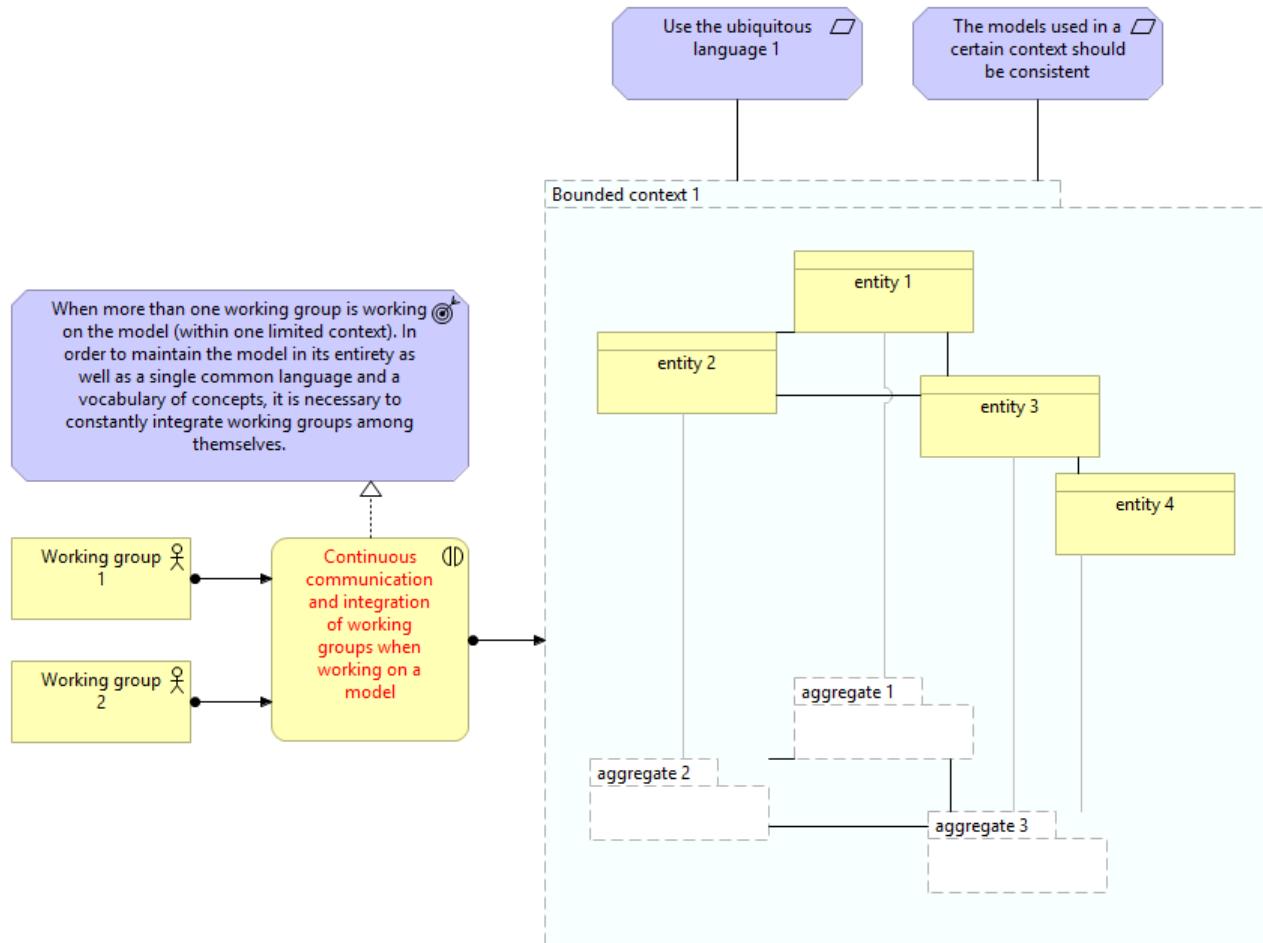
Eric Evans



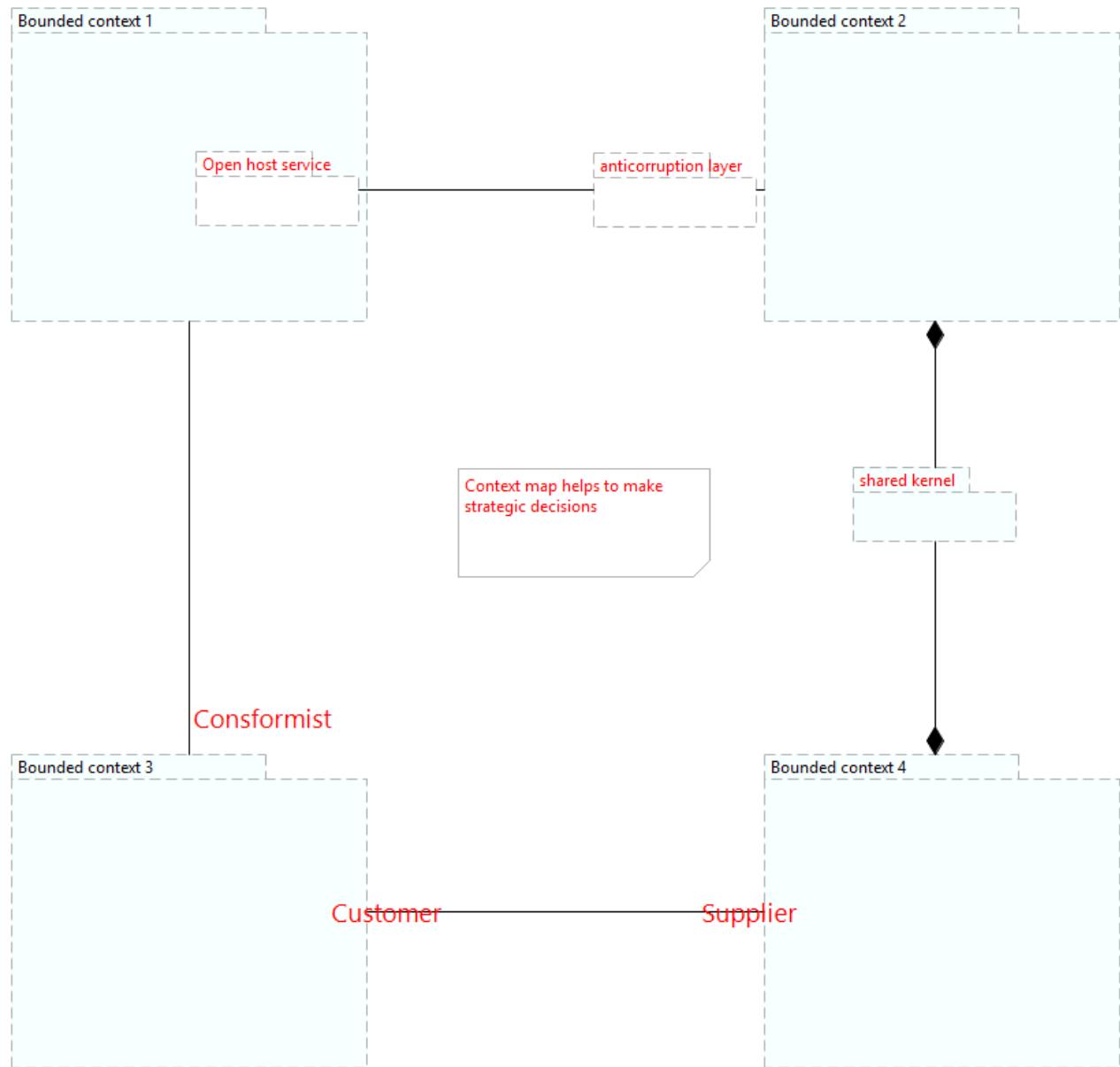
# BOUNDED CONTEXT



# CONTINUOUS INTEGRATION

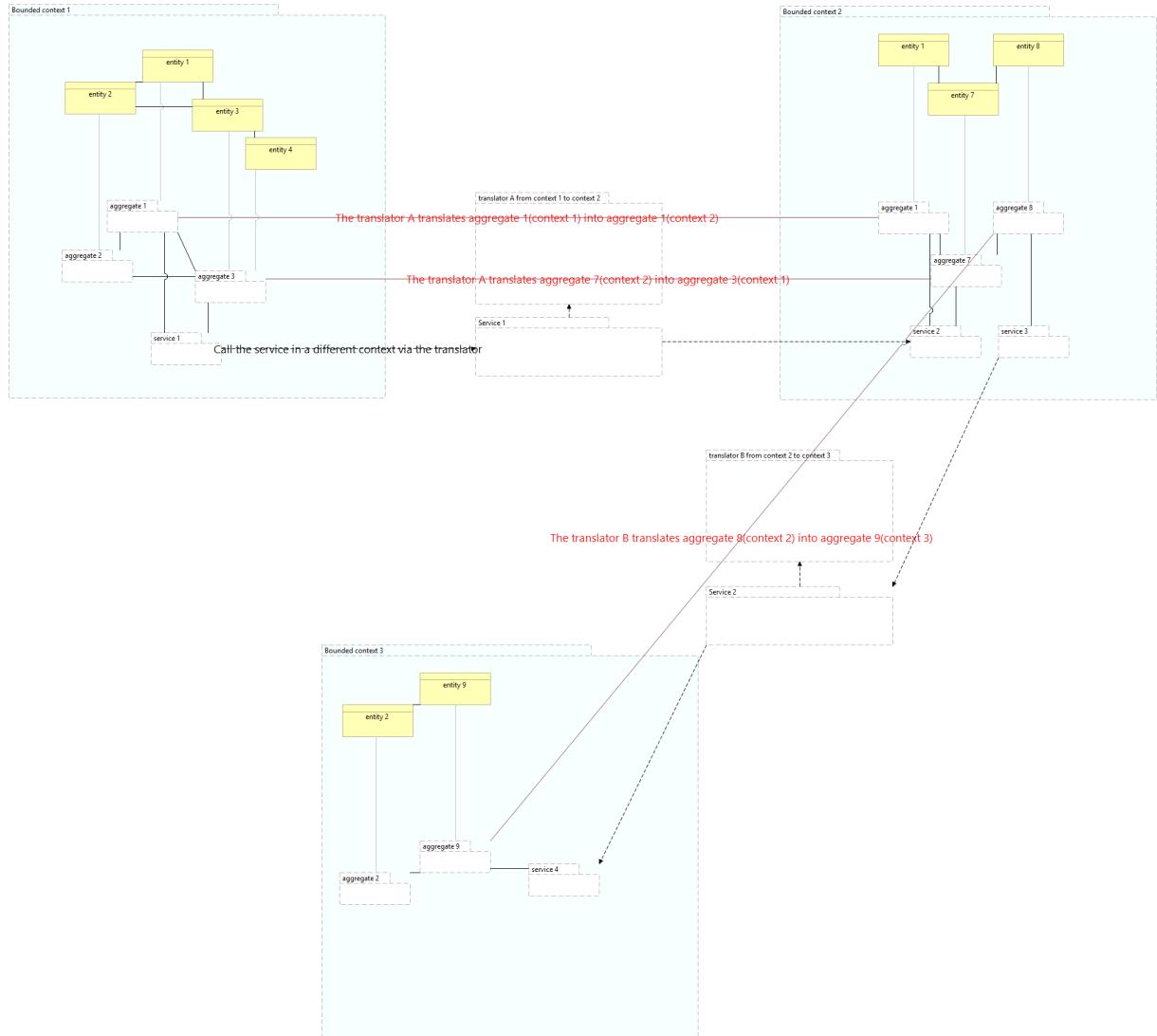


# STRATEGIC CONTEXT MAP

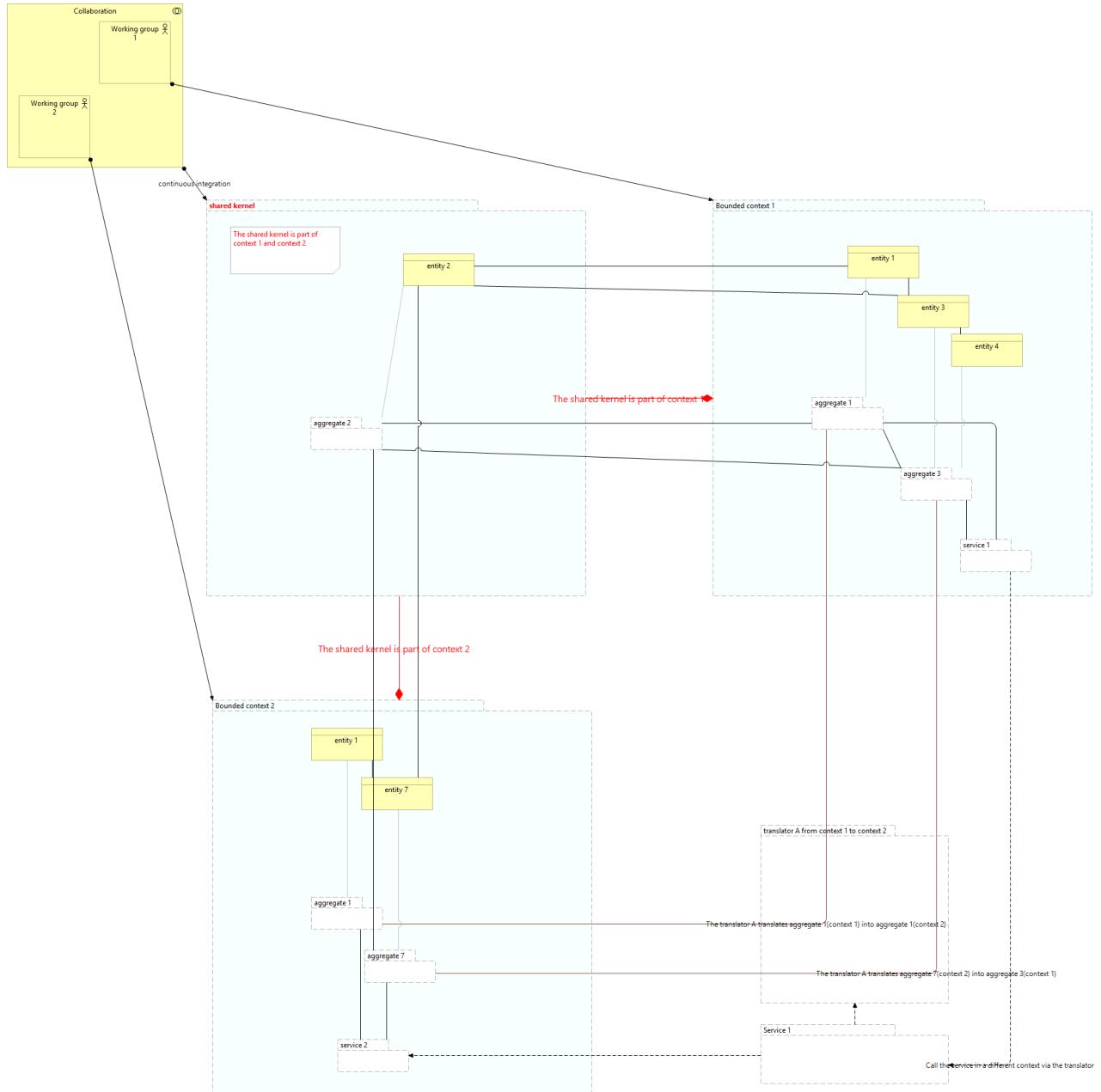


# CONTEXTUAL MAP

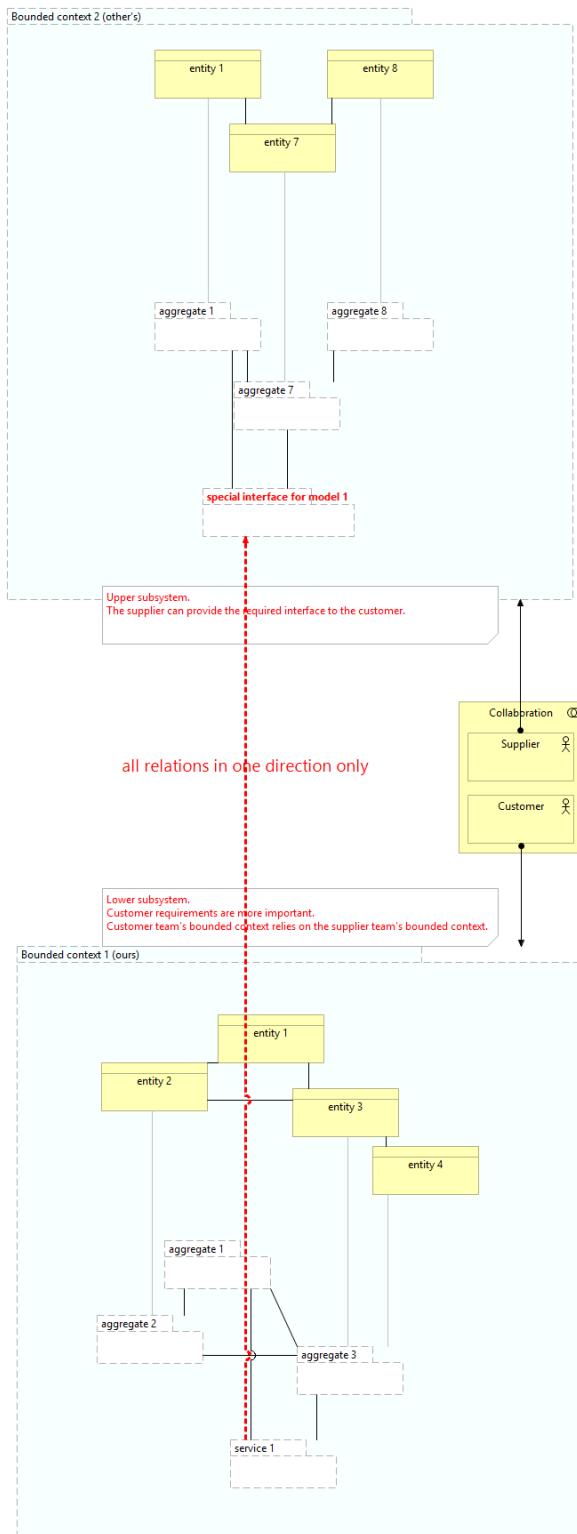
A single map of all contexts.  
Integration of all bounded contexts.



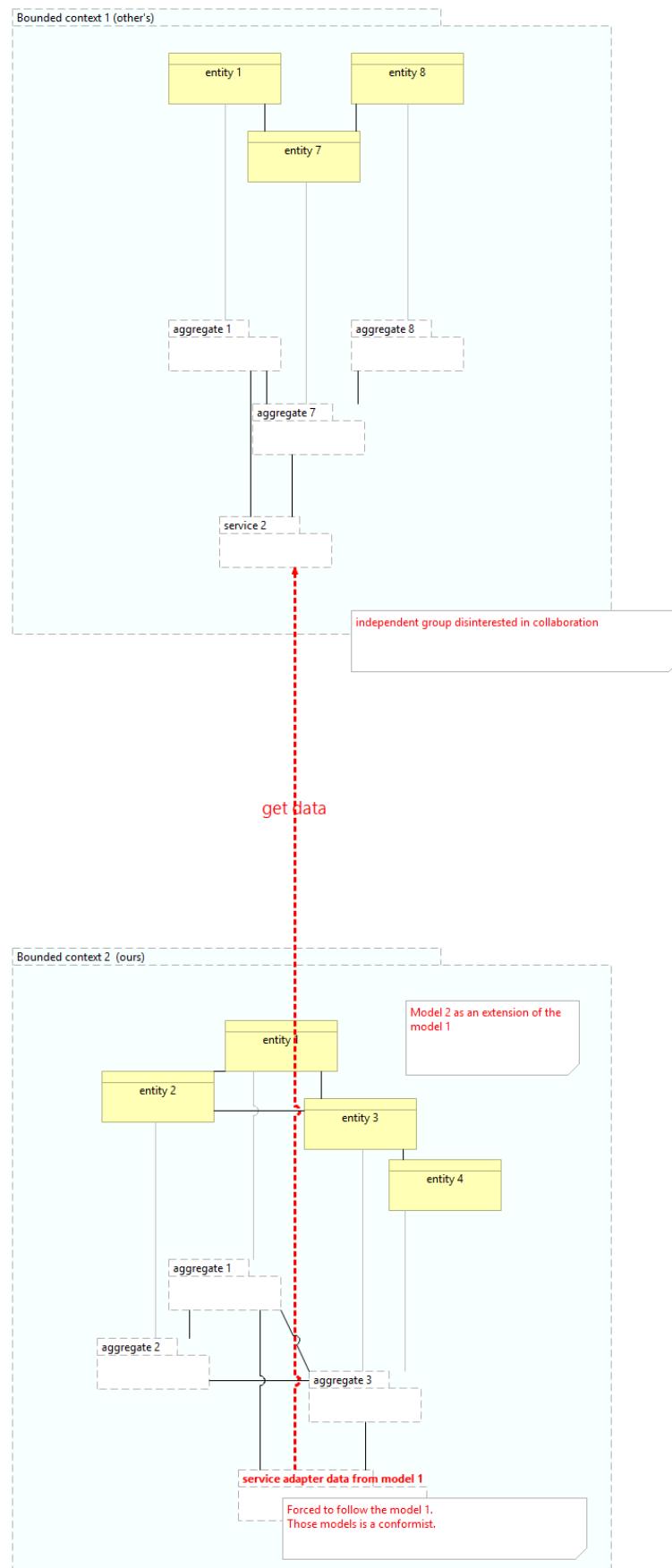
# SHARED KERNEL



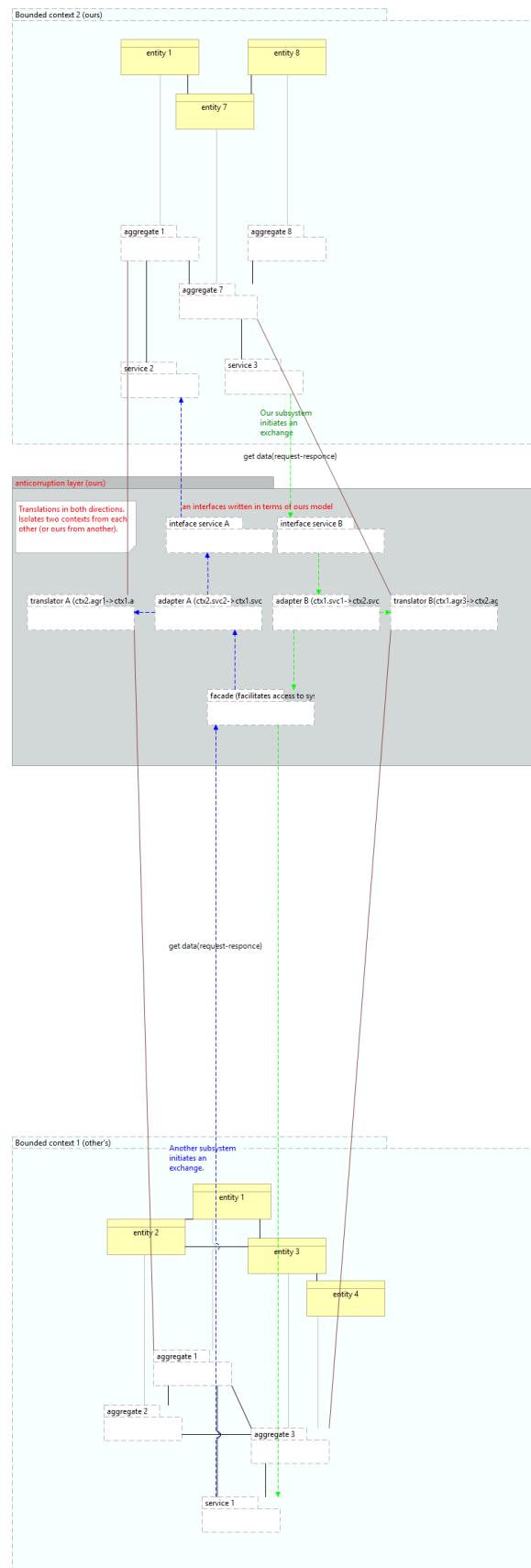
# CUSTOMER-SUPPLIER TEAMS



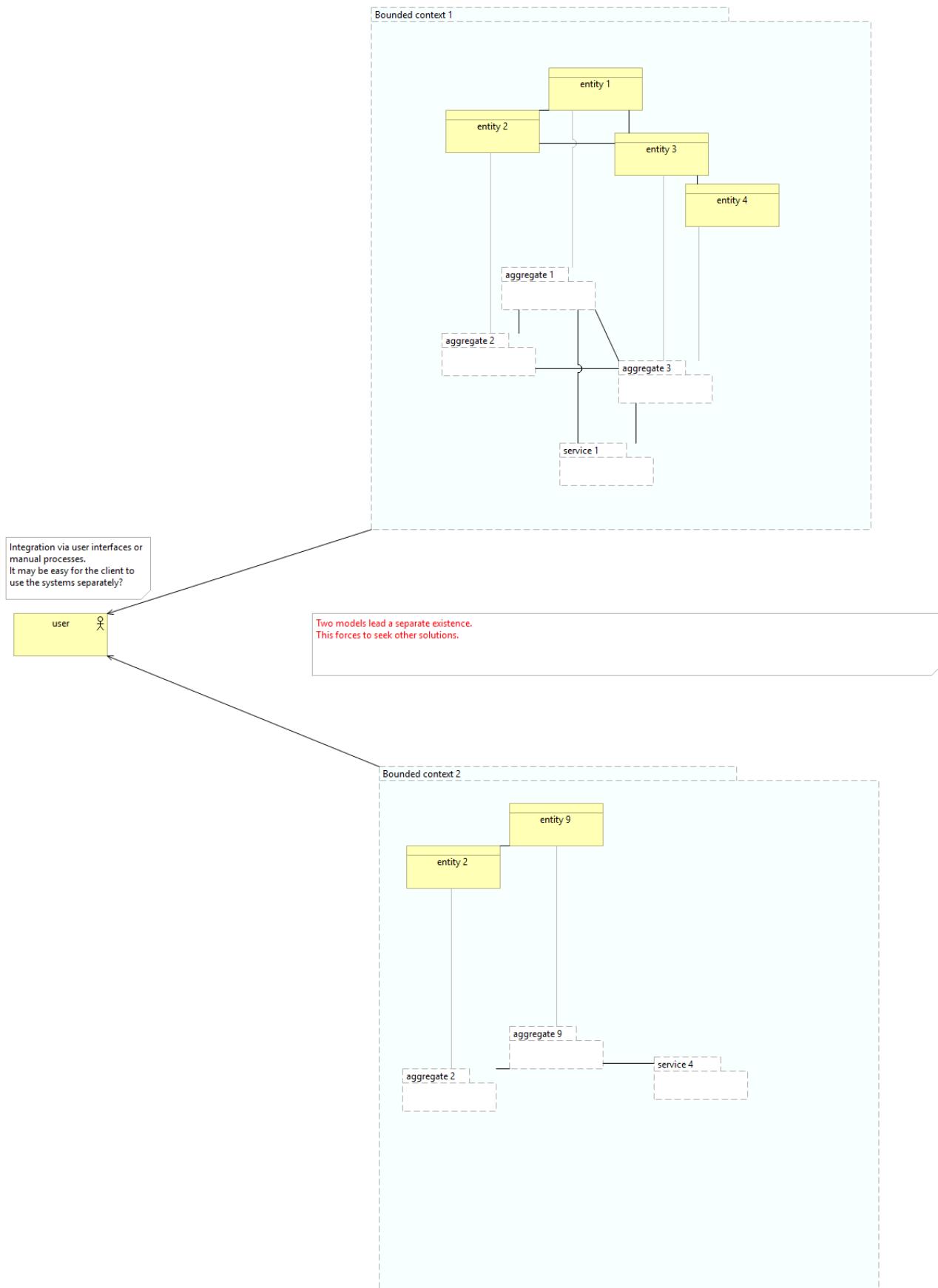
# CONFORMIST



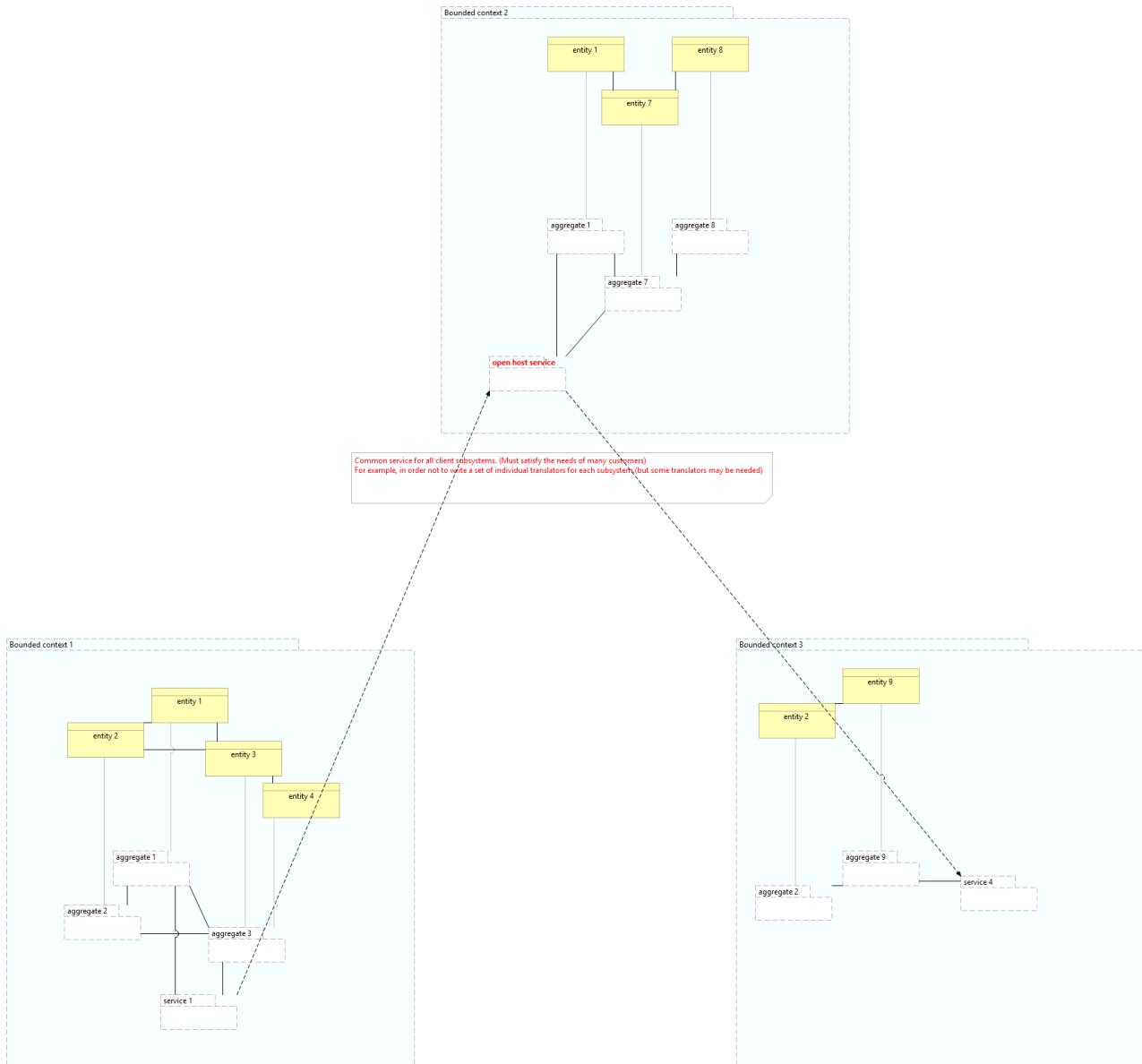
# ANTICORRUPTION LAYER



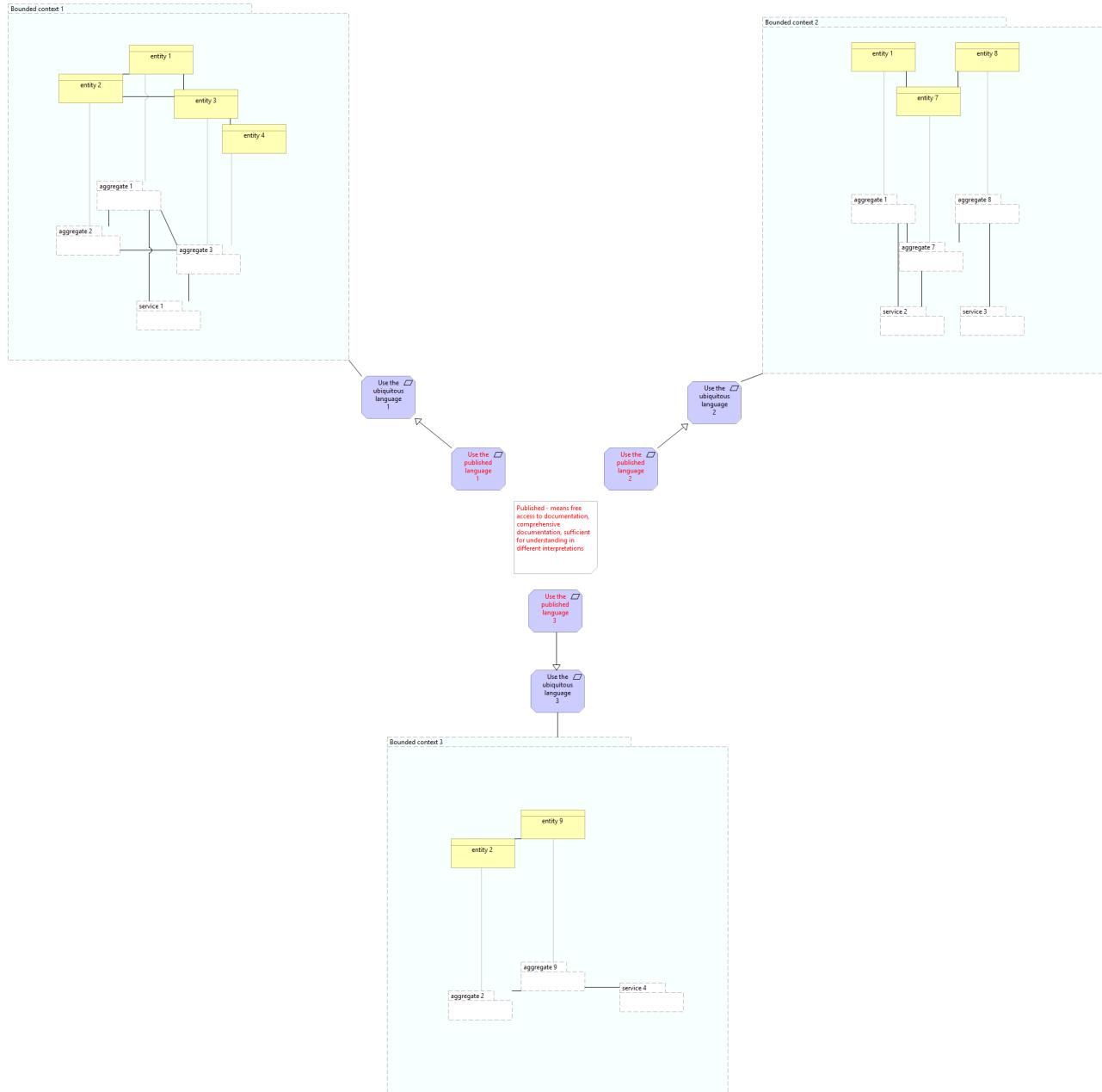
# SEPARATE WAYS



# OPEN HOST SERVICE



# PUBLISHED LANGUAGE



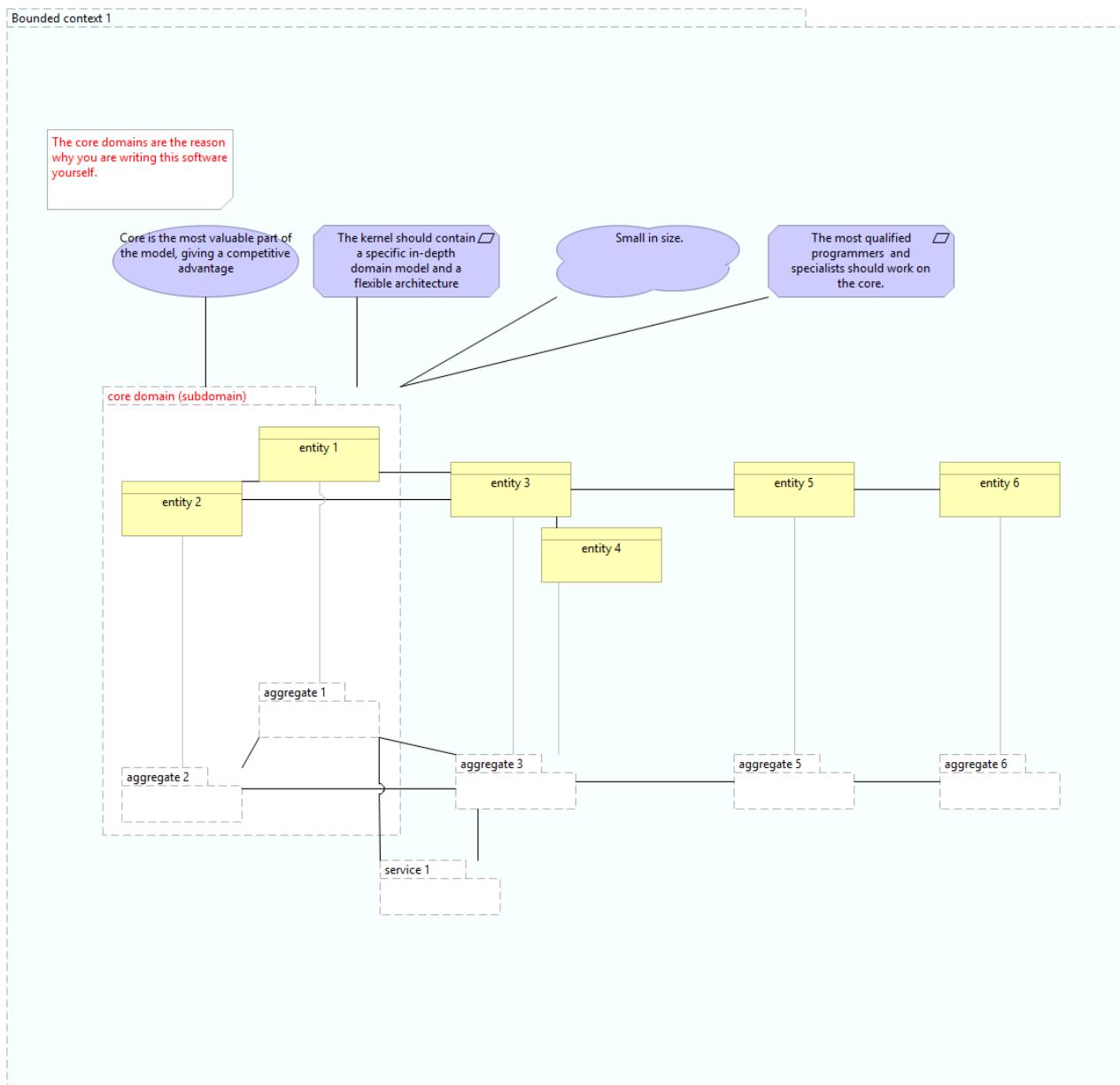
# DISTILLATION

DOMAIN DRIVEN DESIGN

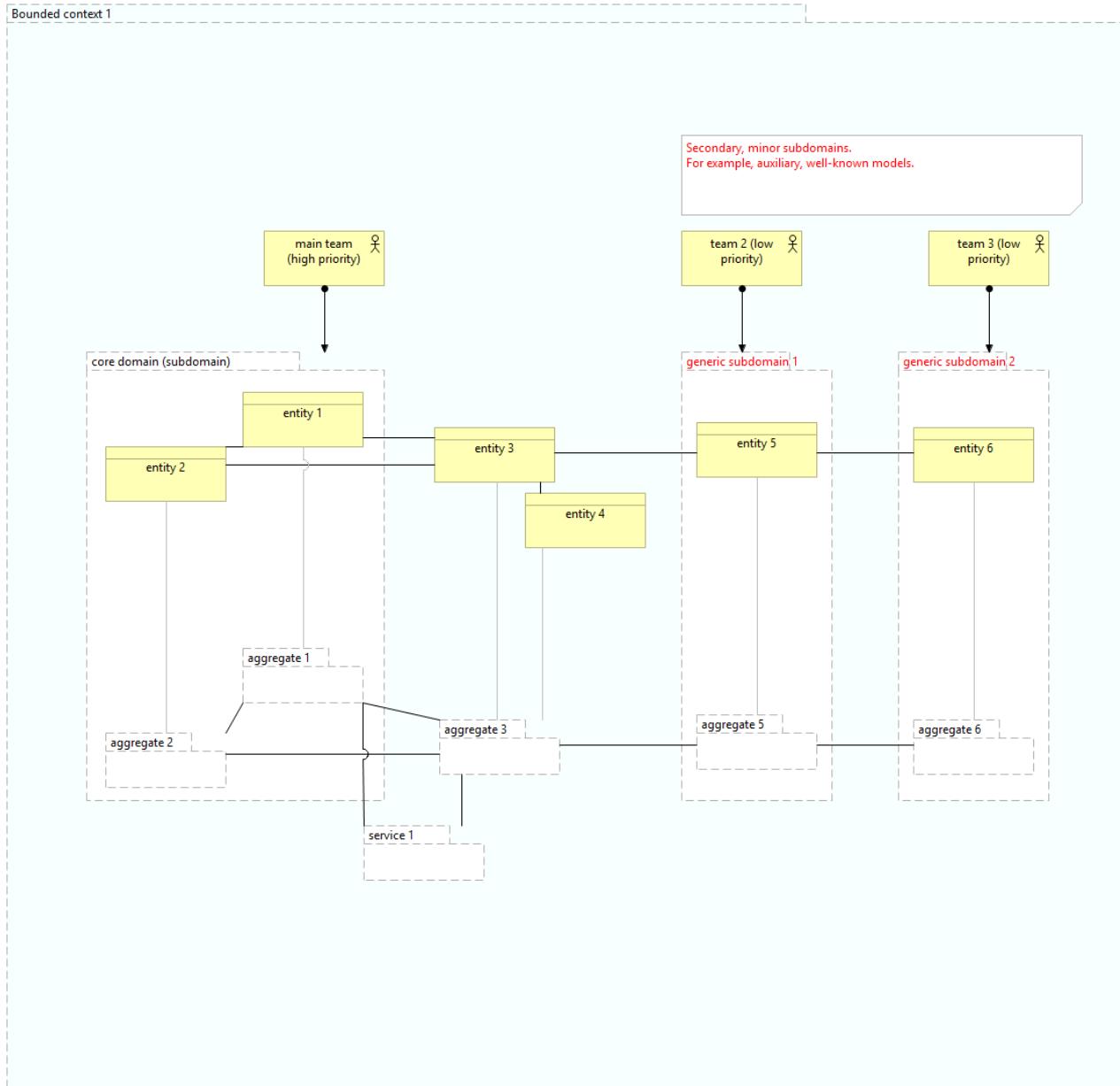
Eric Evans



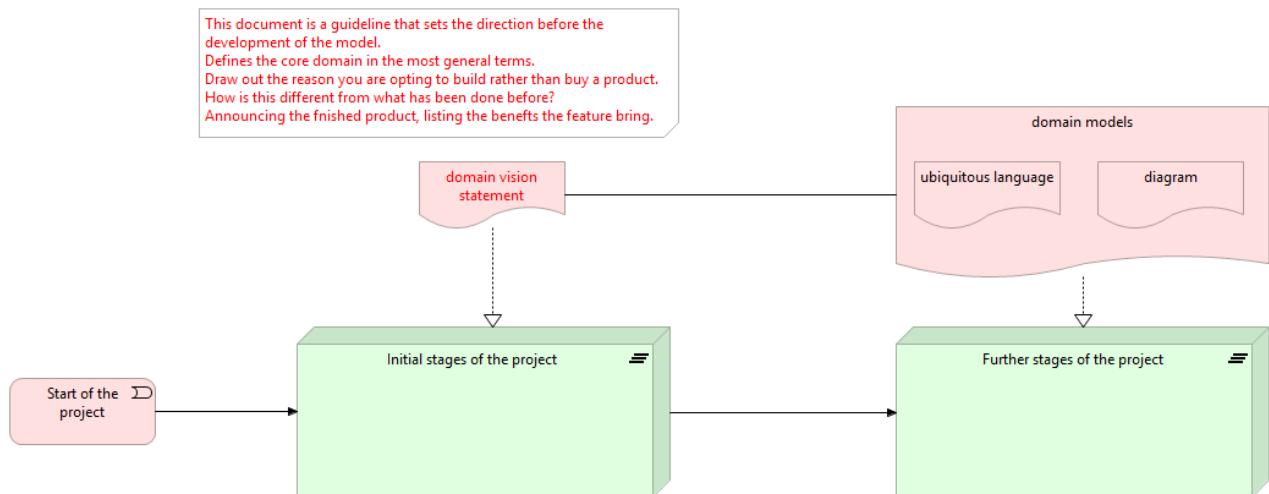
# CORE DOMAIN



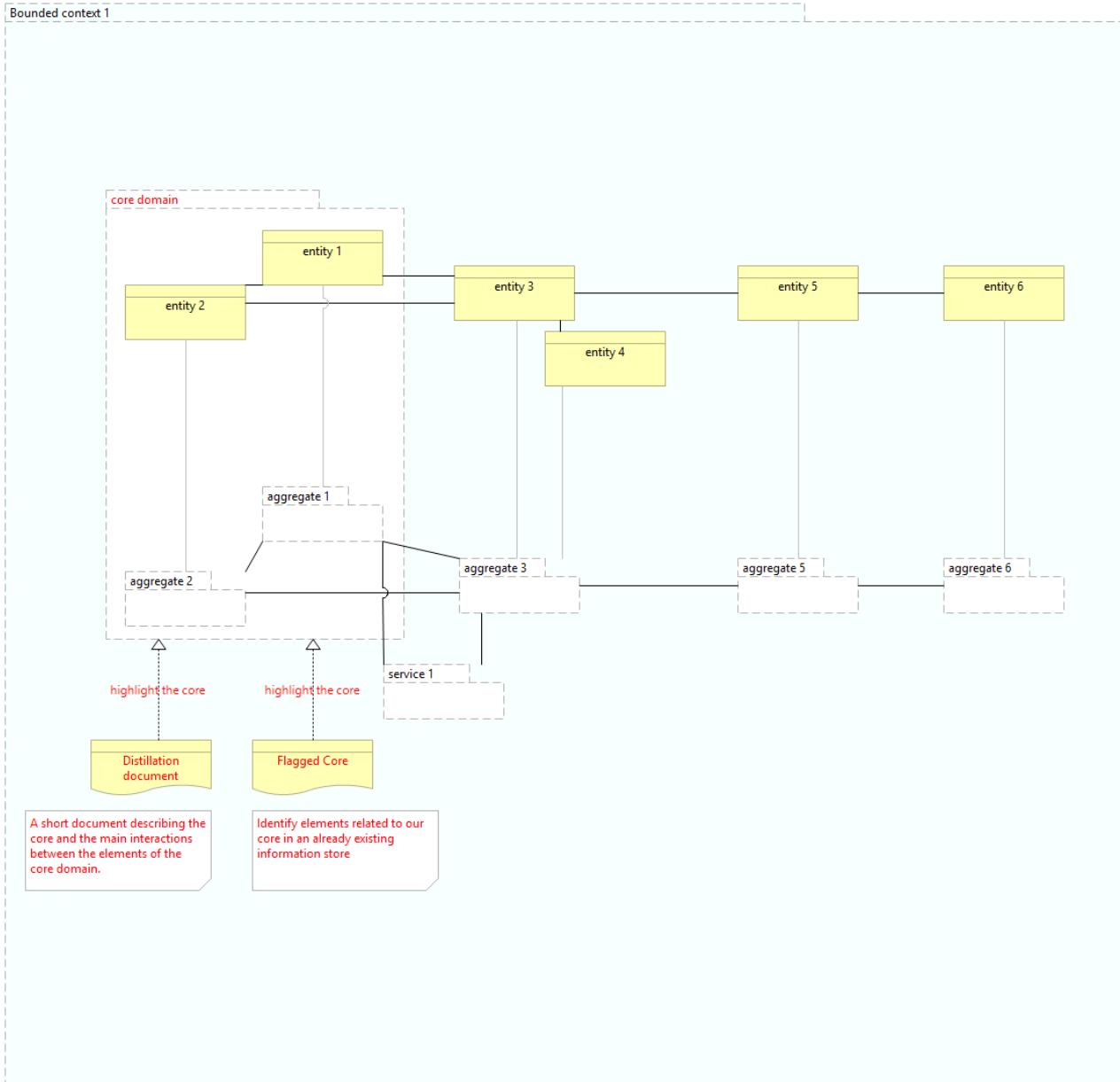
# GENERIC SUBDOMAINS



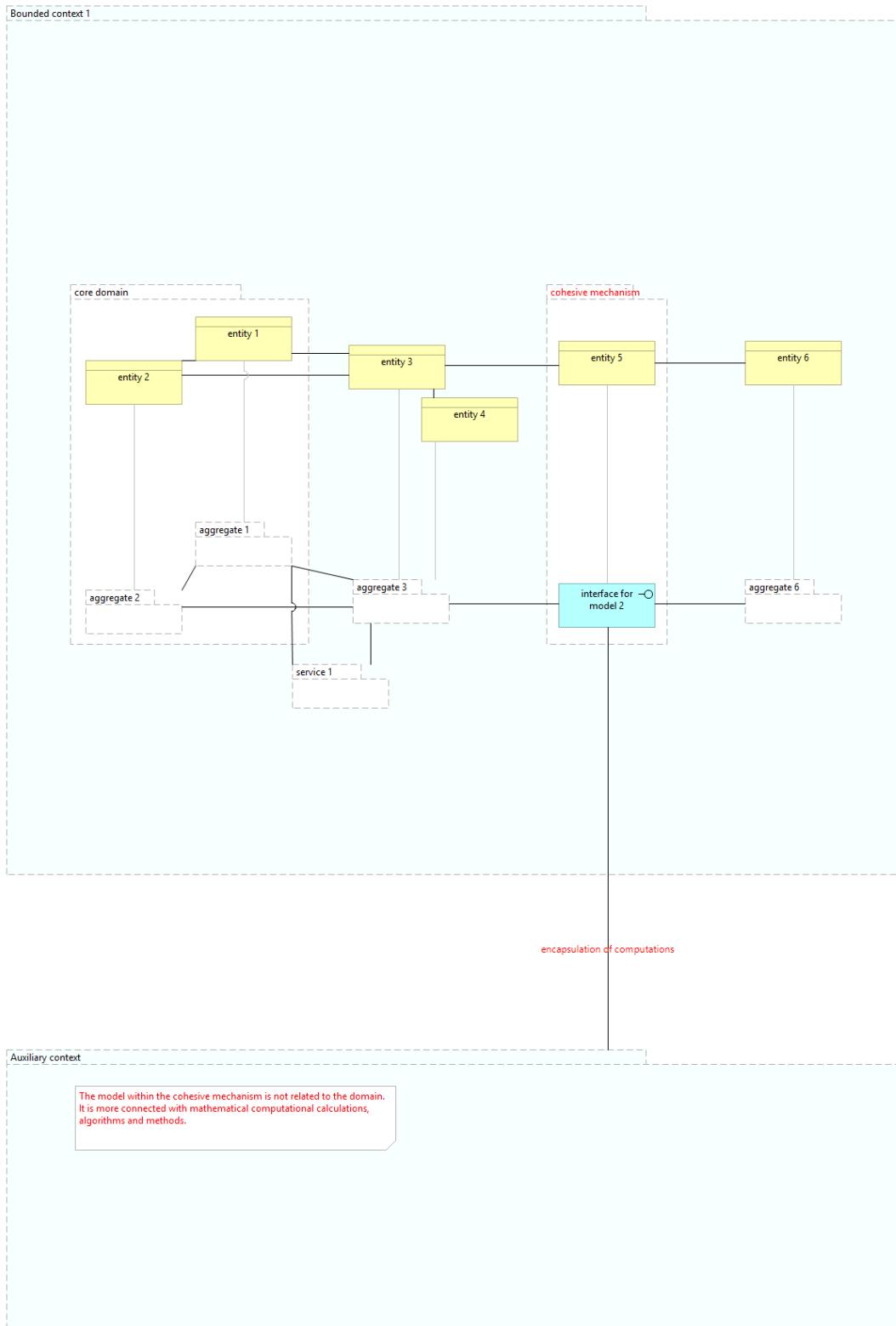
# DOMAIN VISION STATEMENT



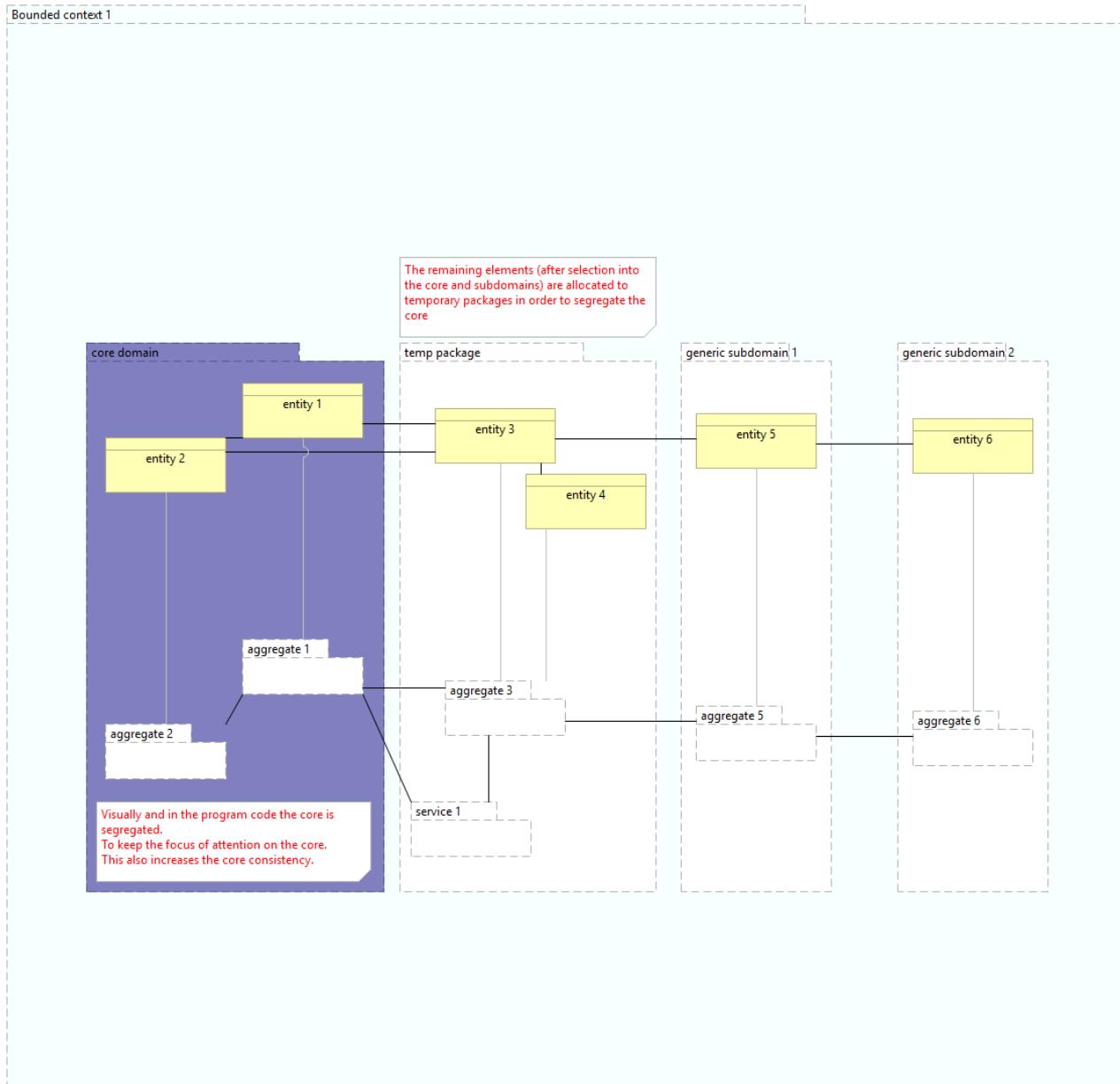
# HIGHLIGHTED CORE



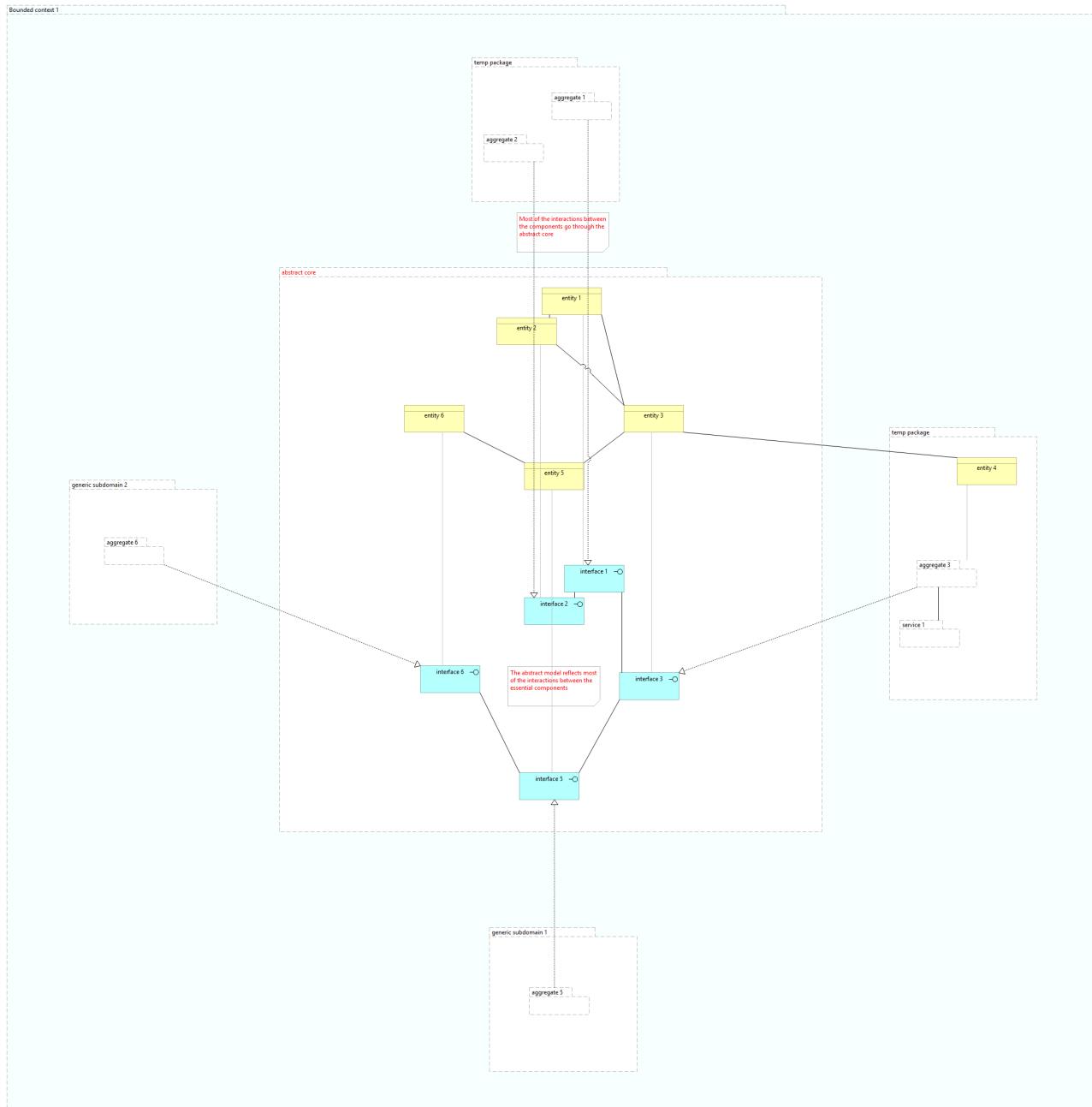
# COHESIVE MECHANISMS



# SEGREGATED CORE



# ABSTRACT CORE



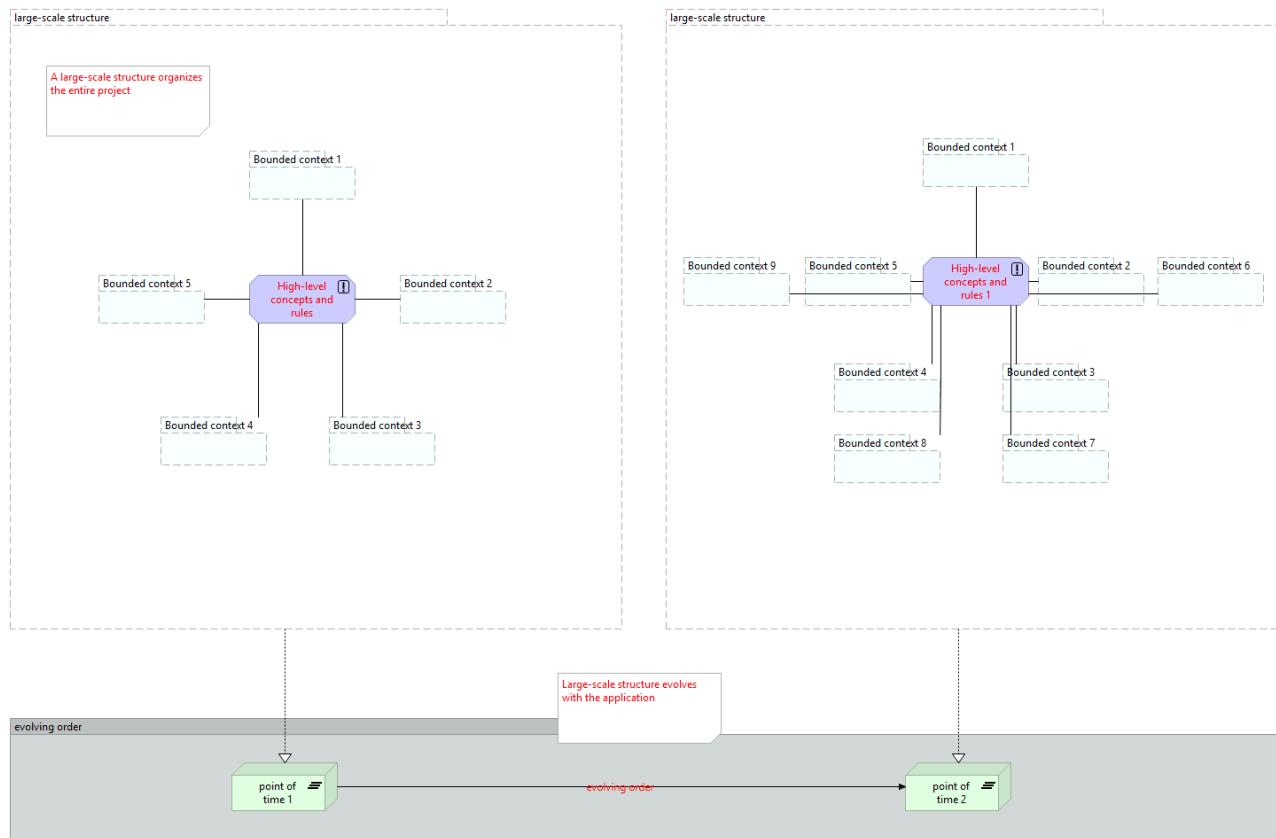
# LARGE-SCALE STRUCTURE

DOMAIN DRIVEN DESIGN

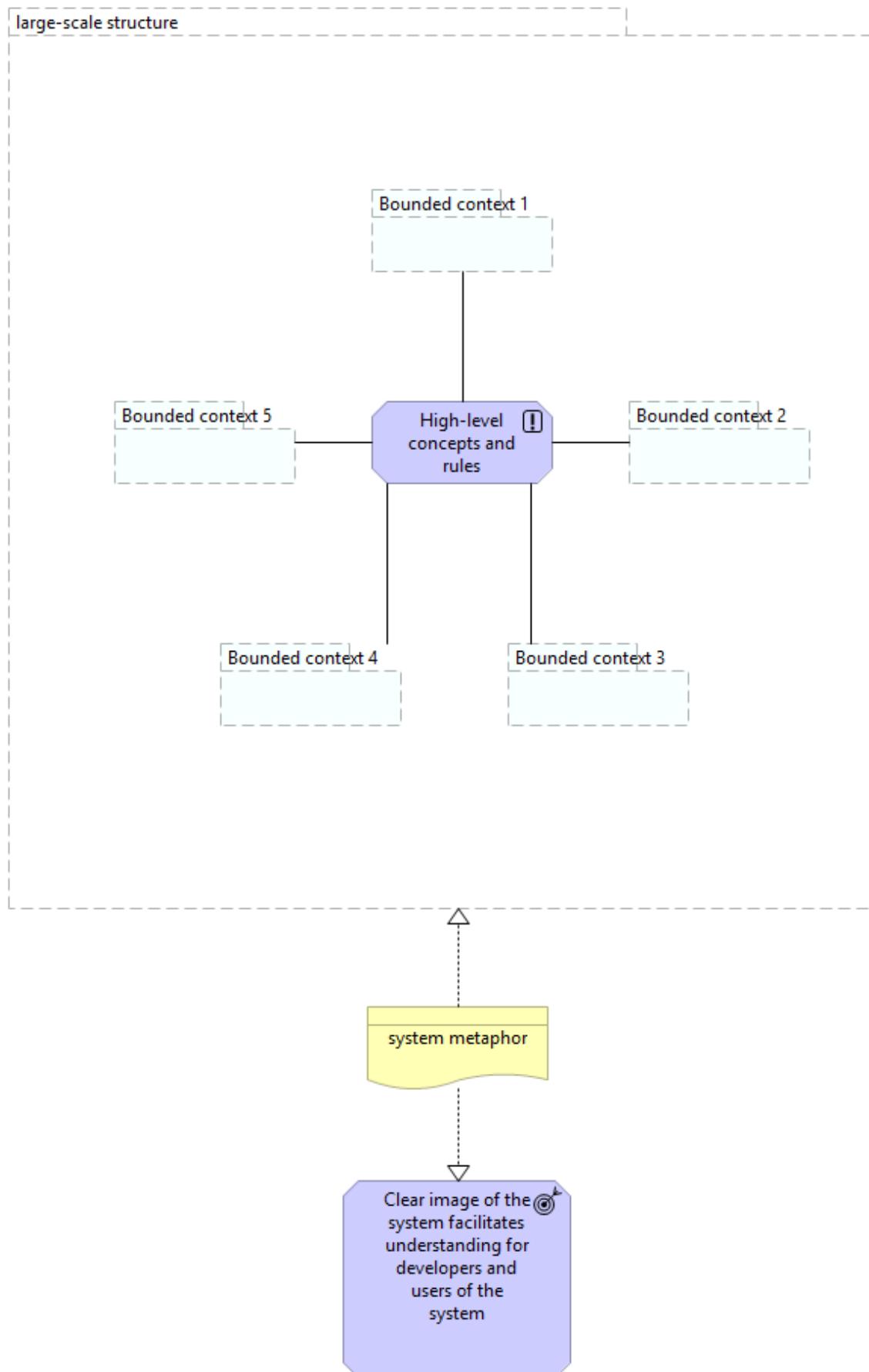
Eric Evans



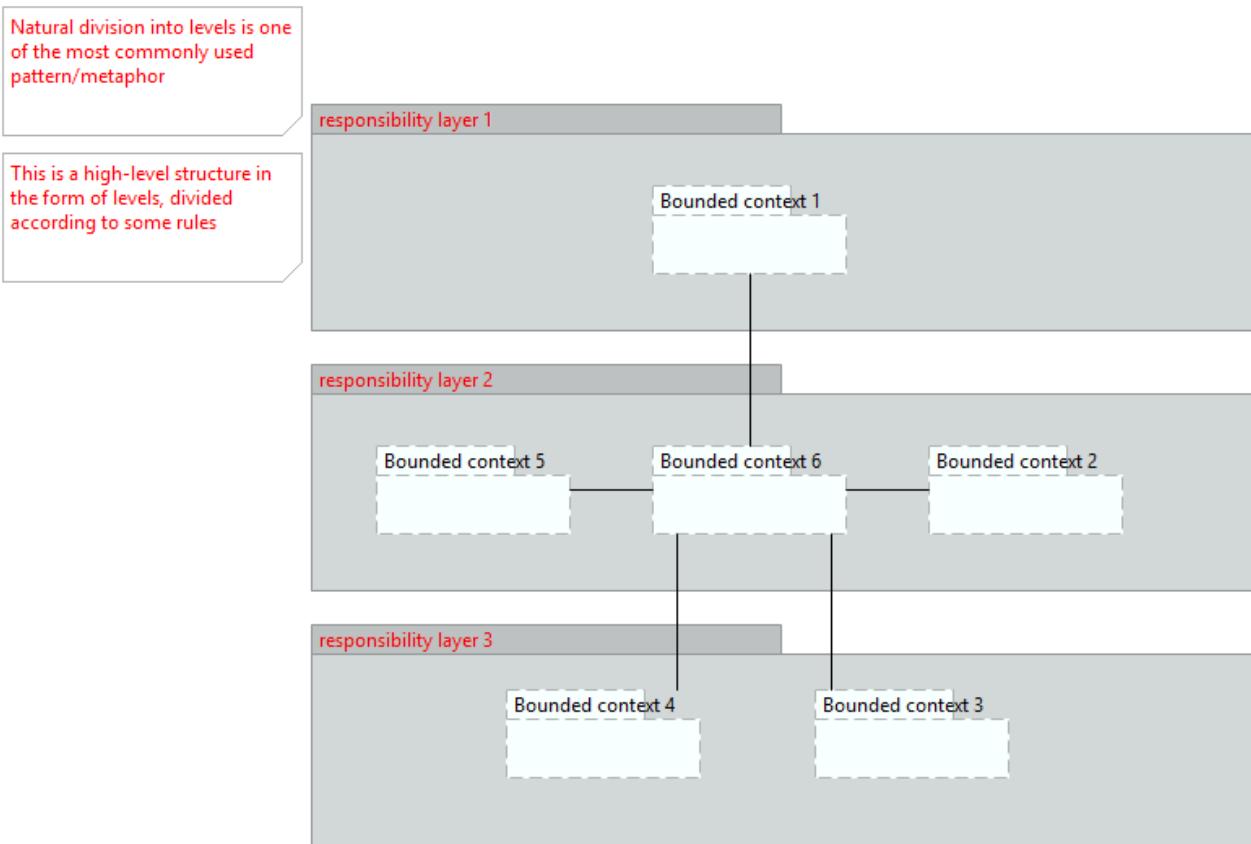
# EVOLVING ORDER

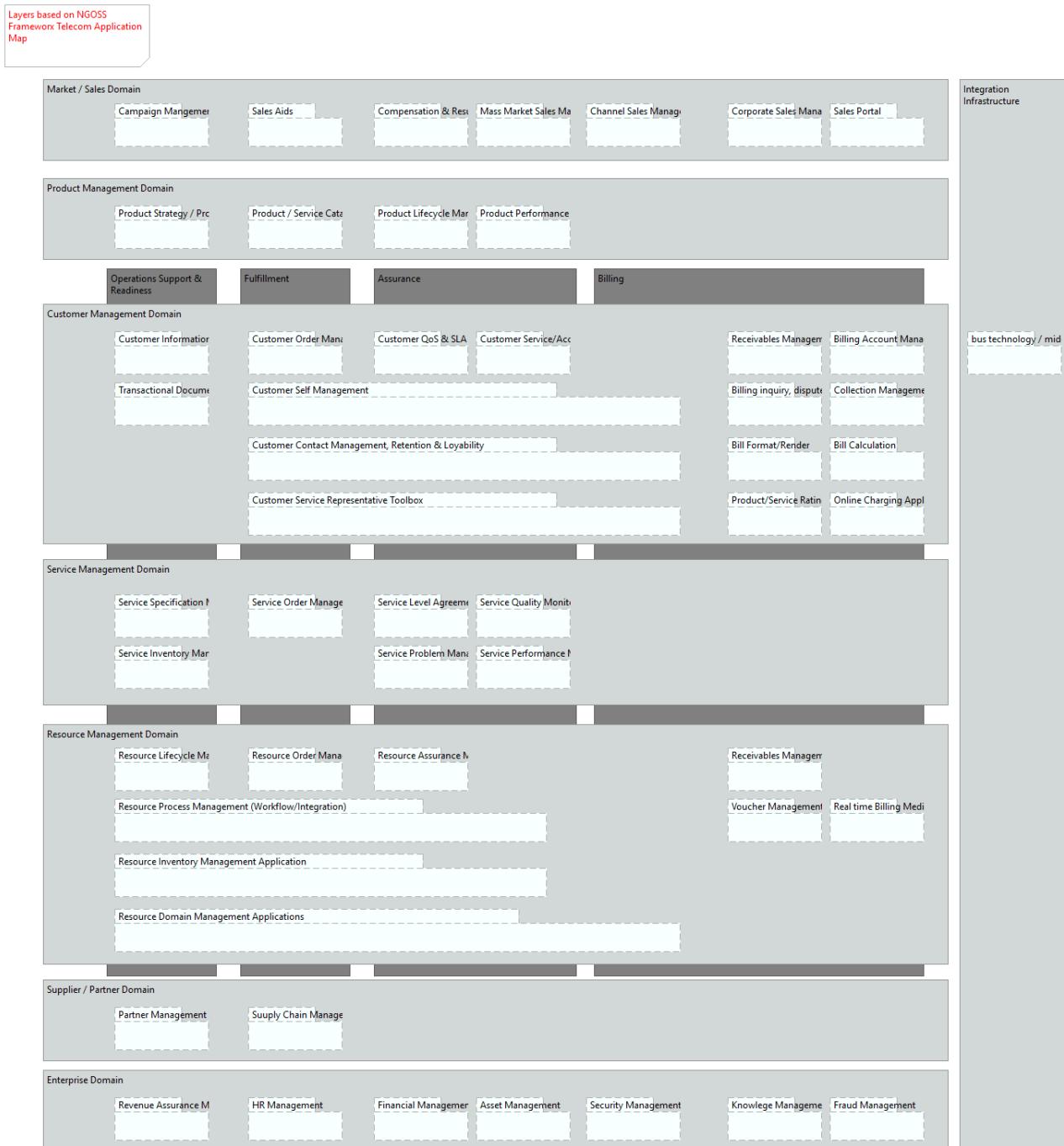


# SYSTEM METAPHOR



# RESPONSIBILITY LAYERS





Layers based on Porter's Value Chain

Firm infrastructure

Bounded context 1

Human Resources Management

Bounded context 5

Bounded context 6

Bounded context 2

Technological Development

Bounded context 4

Bounded context 3

Procurement

Bounded context 5

Bounded context 6

Inbound Logistics

Bounded context 7

Operations

Bounded context 8

Outbound Logistics

Bounded context 9

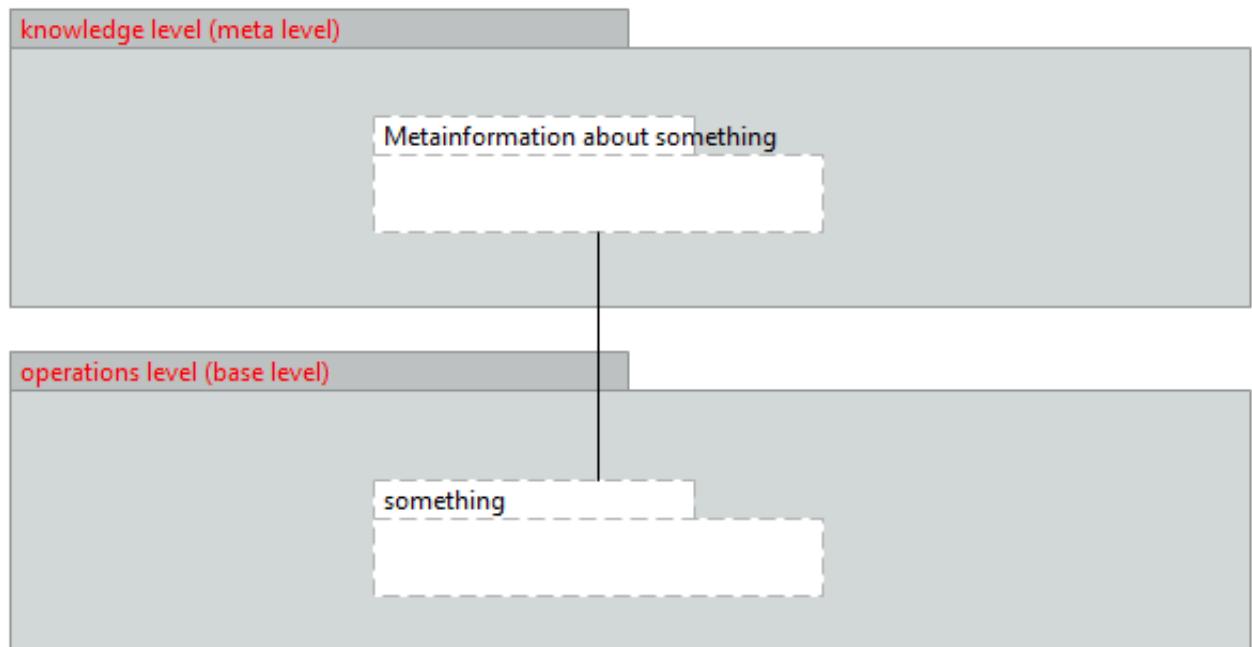
Marketing & Sales

Bounded context 10

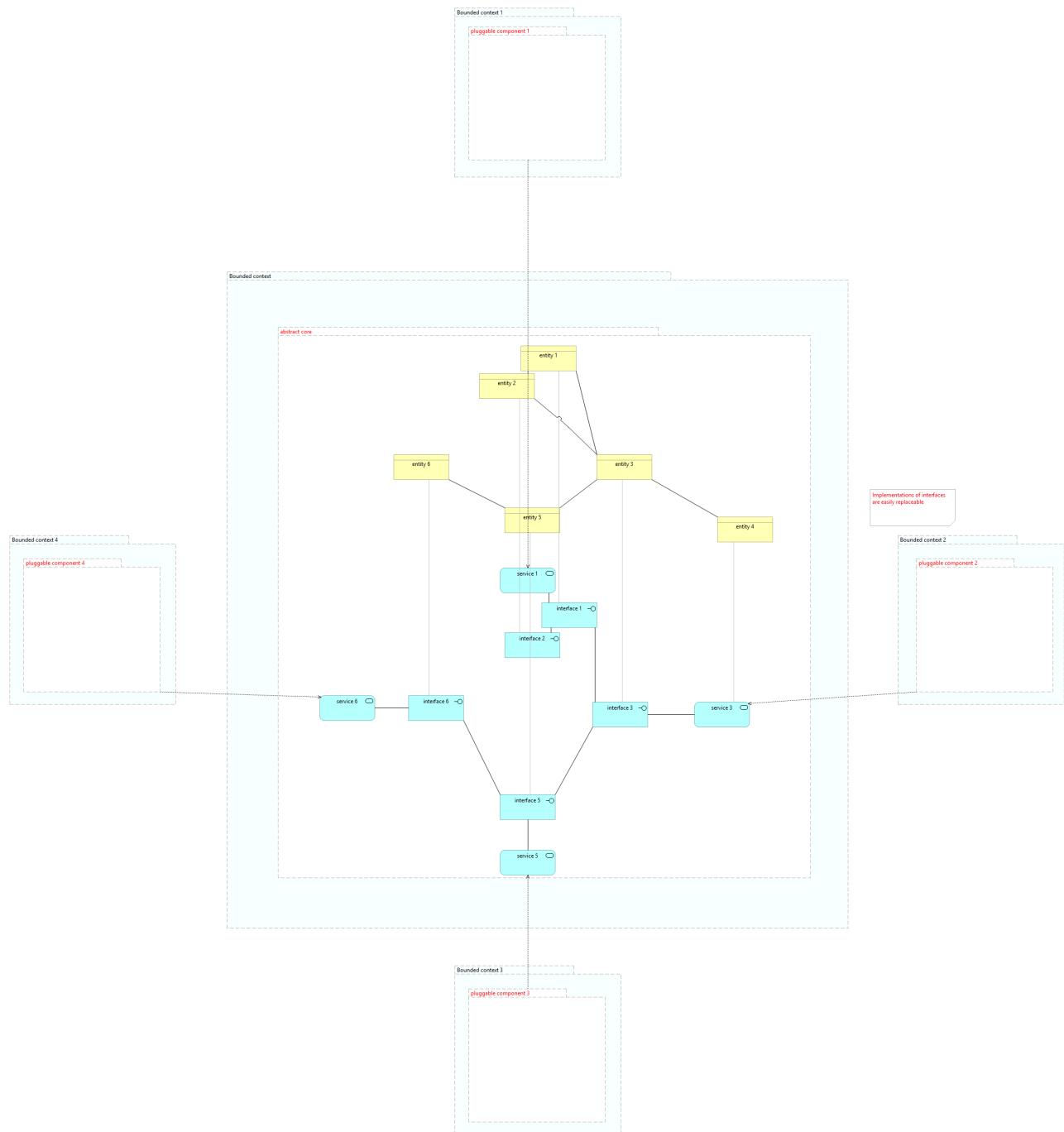
Service

Bounded context 11

# KNOWLEDGE LEVEL



# PLUGGABLE COMPONENT FRAMEWORK



# ADDITIONAL PATTERNS

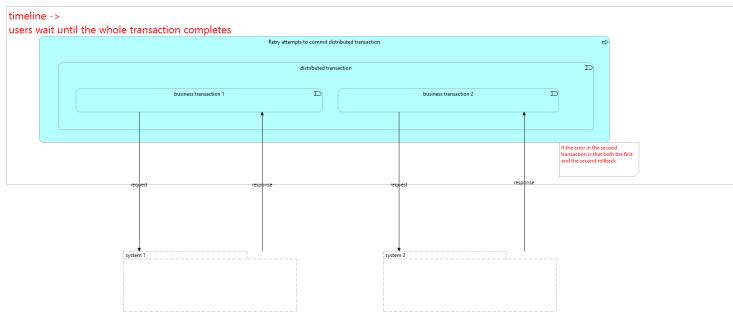
DOMAIN DRIVEN DESIGN

Eric Evans

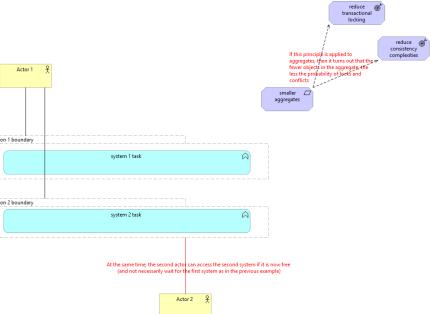
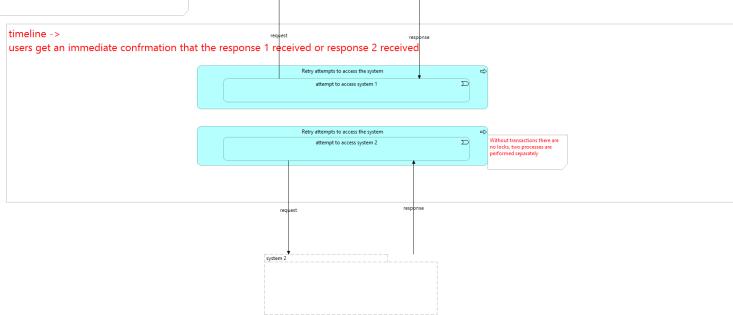


# TYPES OF CONSISTENCY

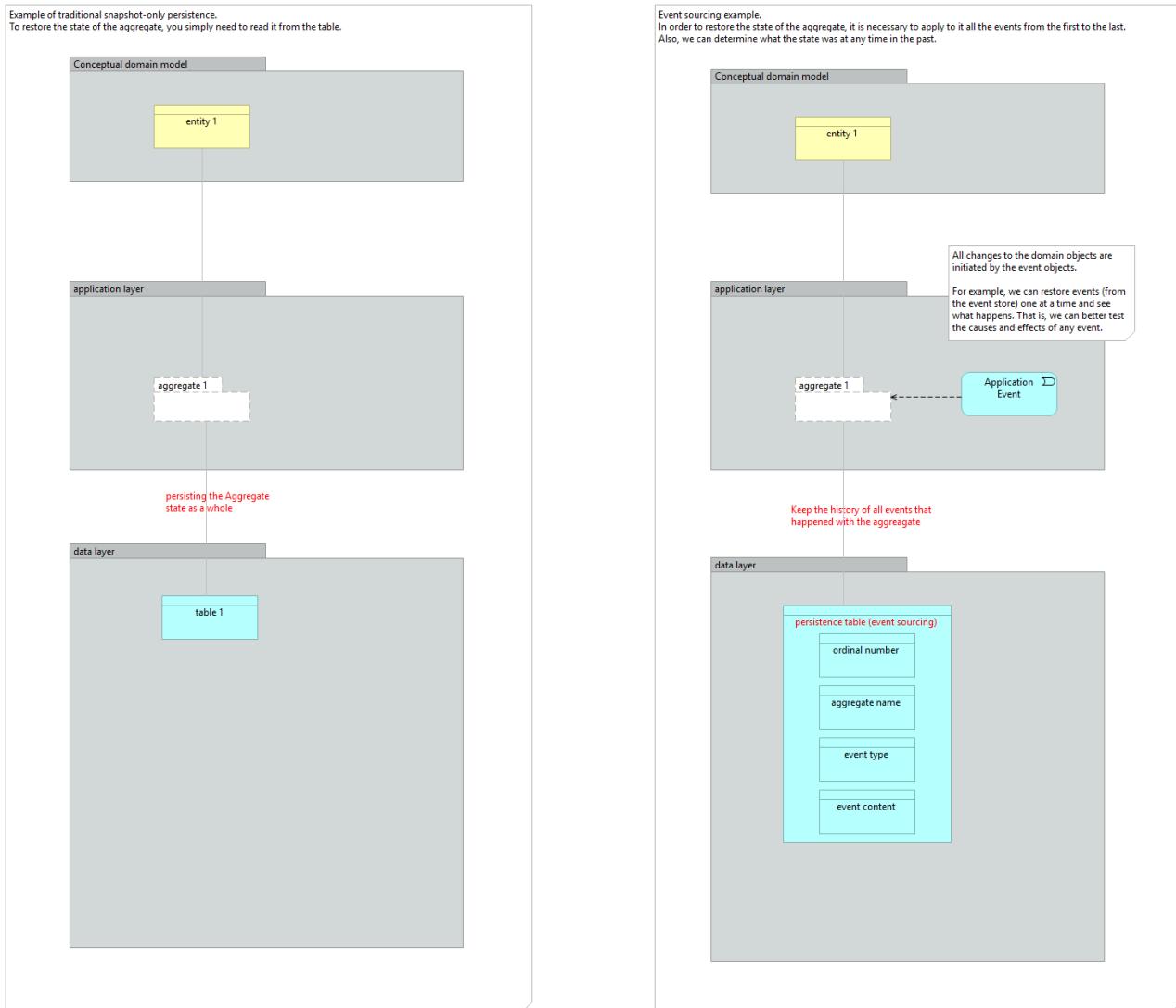
**Transactional consistency:**  
When a transaction is finished the system is in a consistent state.  
C� A distributed transaction is a workflow.  
The more objects involved in a distributed transaction, the greater the risk of potential loss and conflicts.



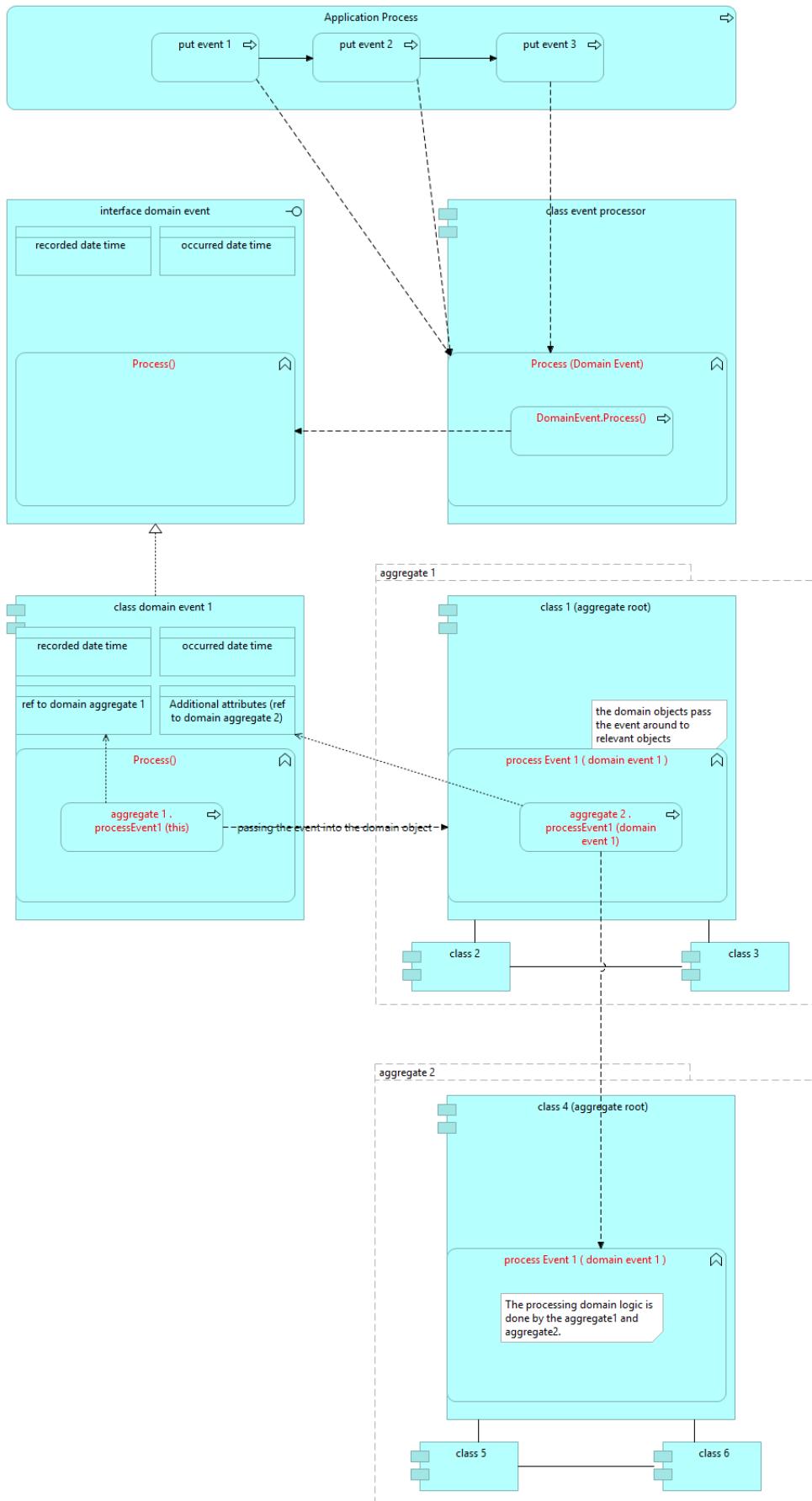
**Eventual Consistency:**  
B&D (Business Available, Soft state, Eventual consistency)



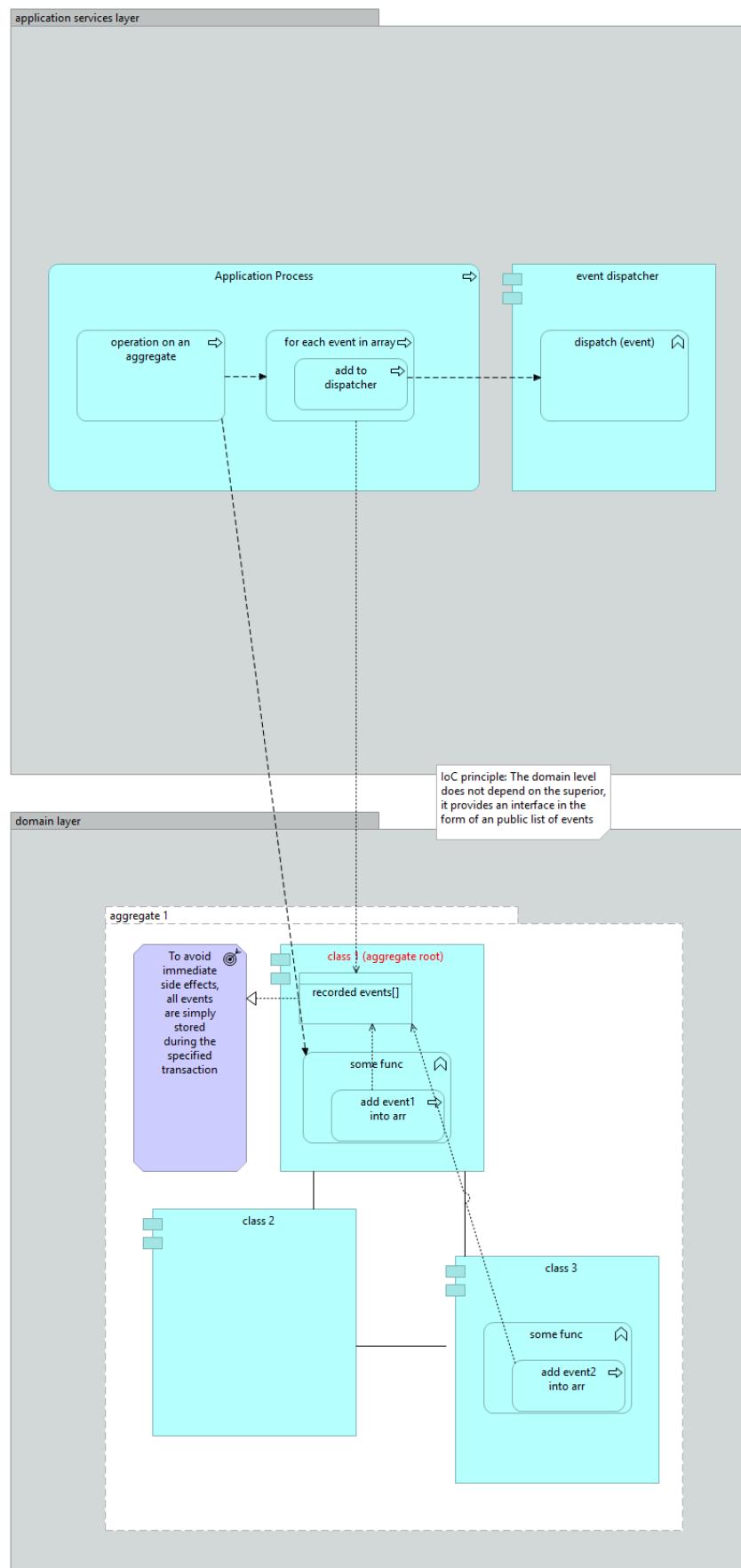
# EVENT SOURCING



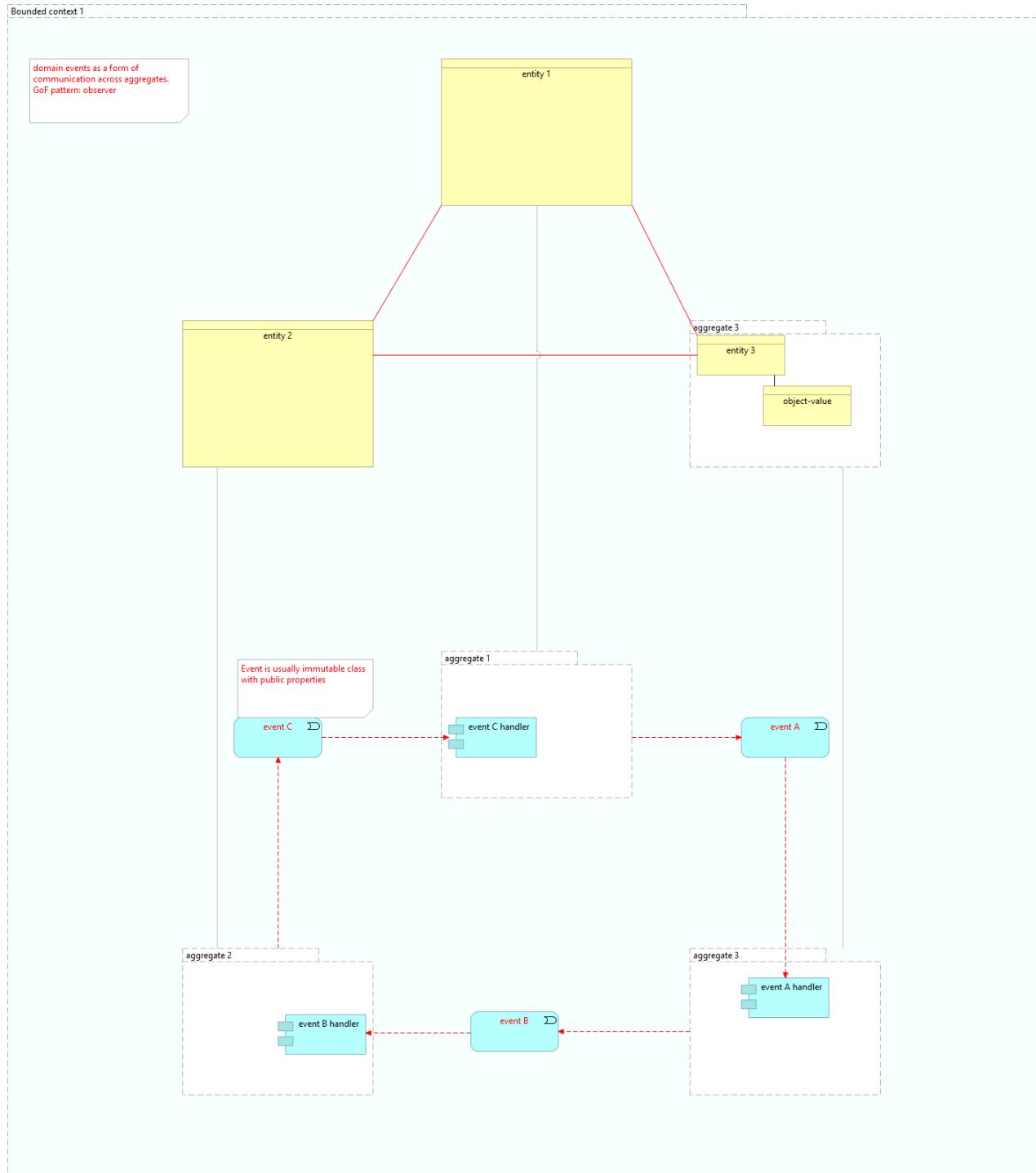
# EVENT PROCESSOR



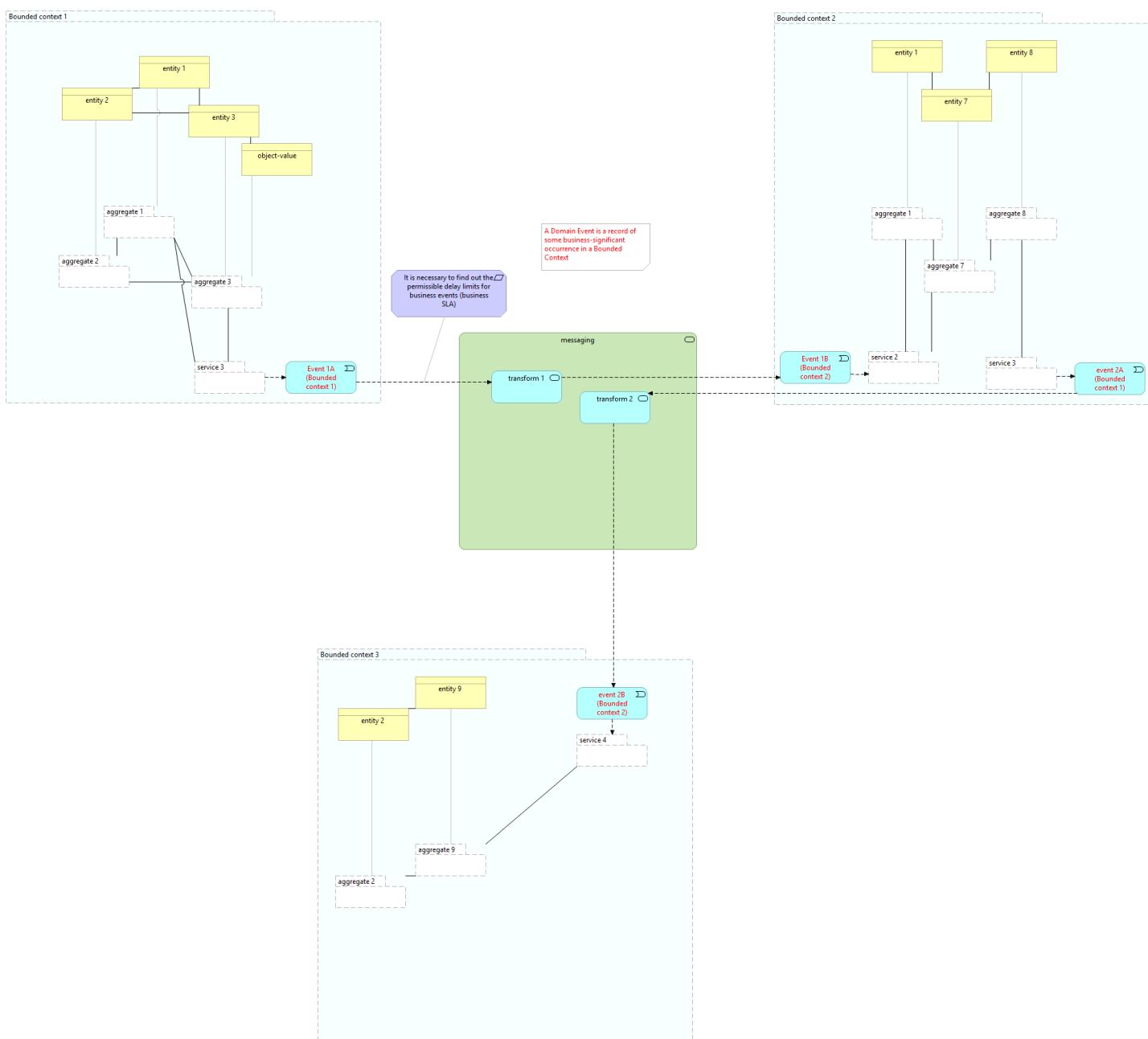
# EVENT DISPATCHER



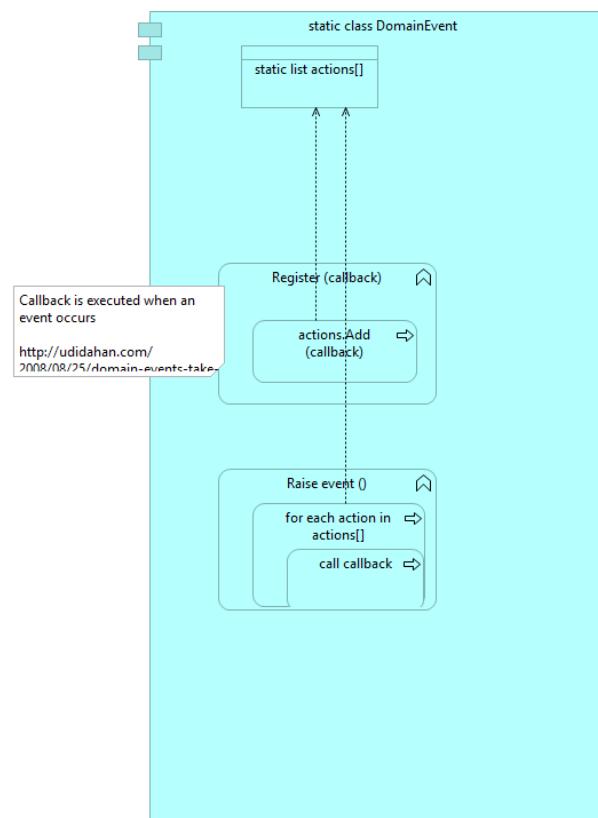
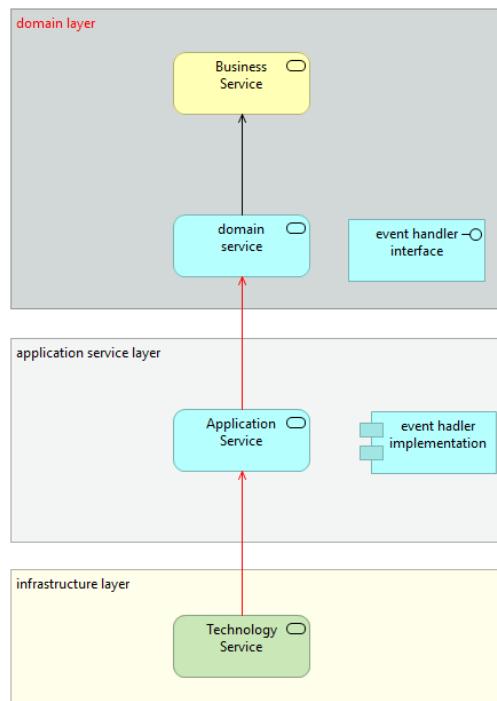
# INTERNAL DOMAIN EVENTS



# EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS

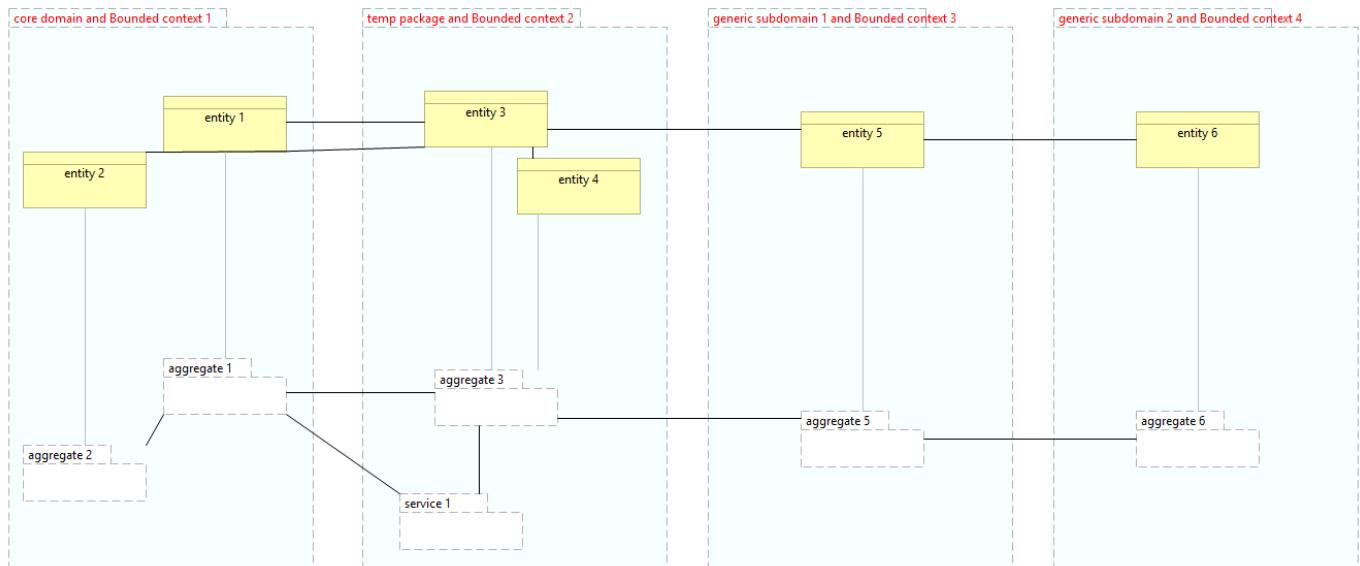


# STATIC DOMAIN EVENTS CLASS

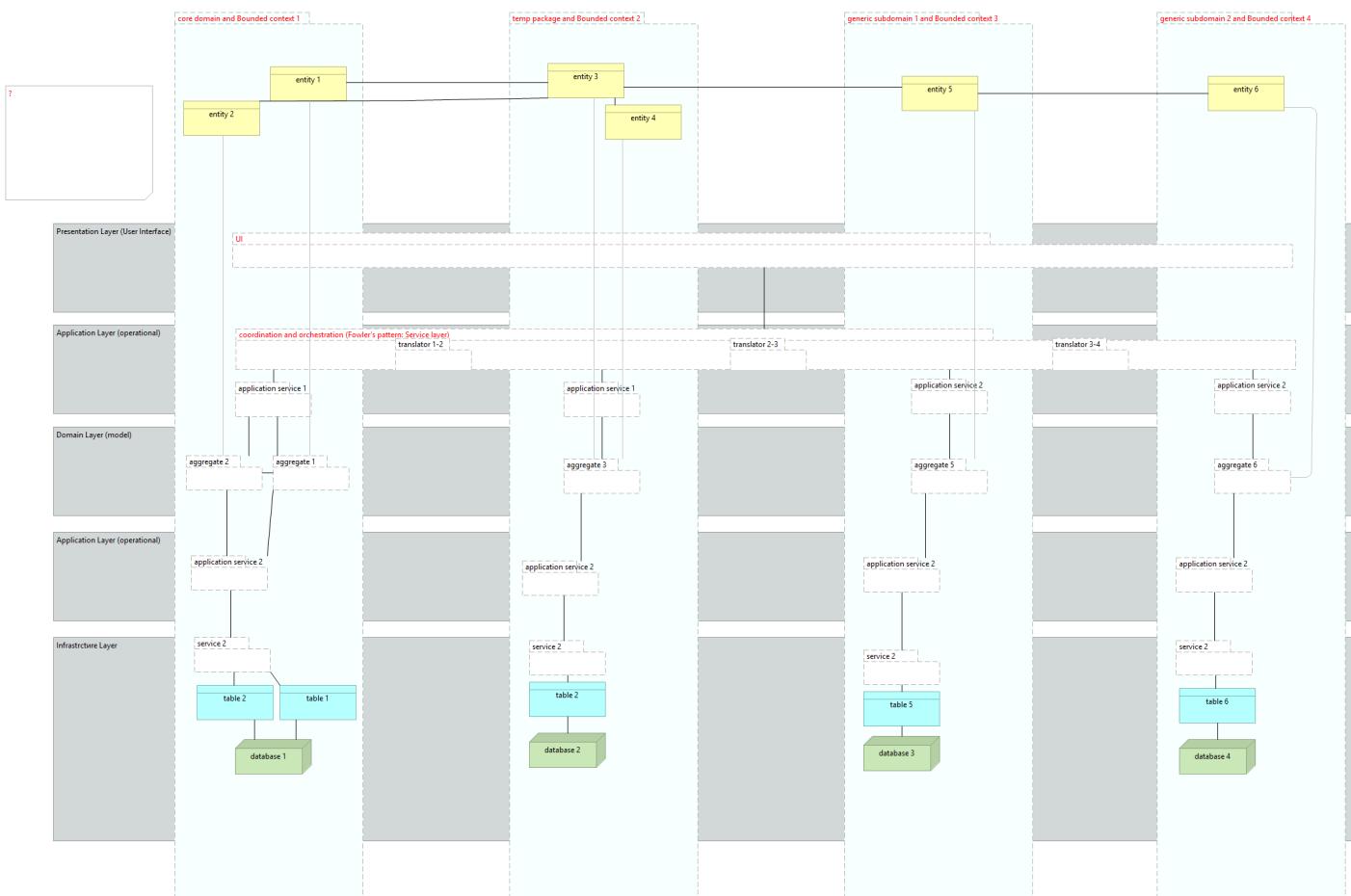


# ONE SUBDOMAIN PER BOUNDED CONTEXT

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

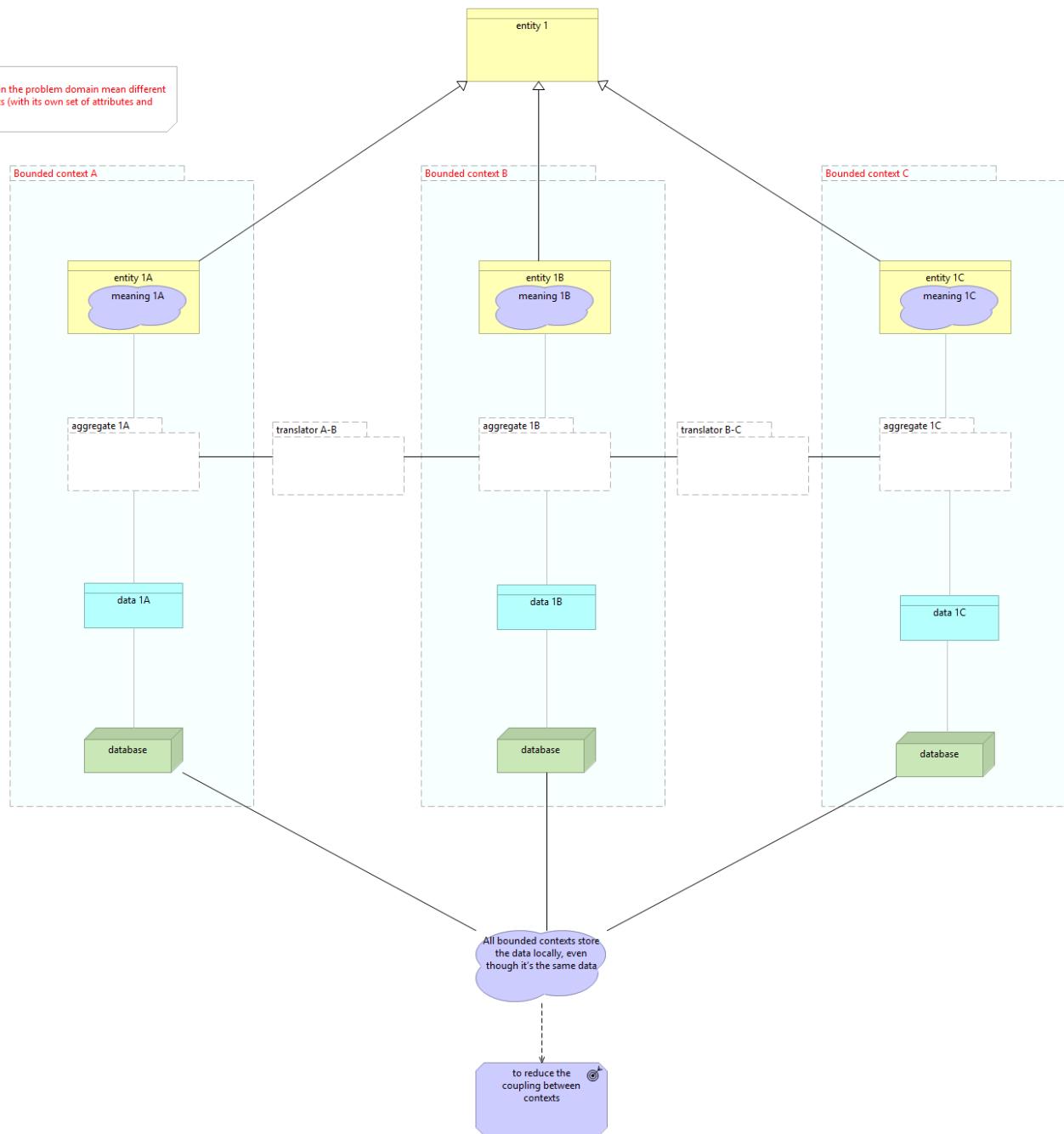


# THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS

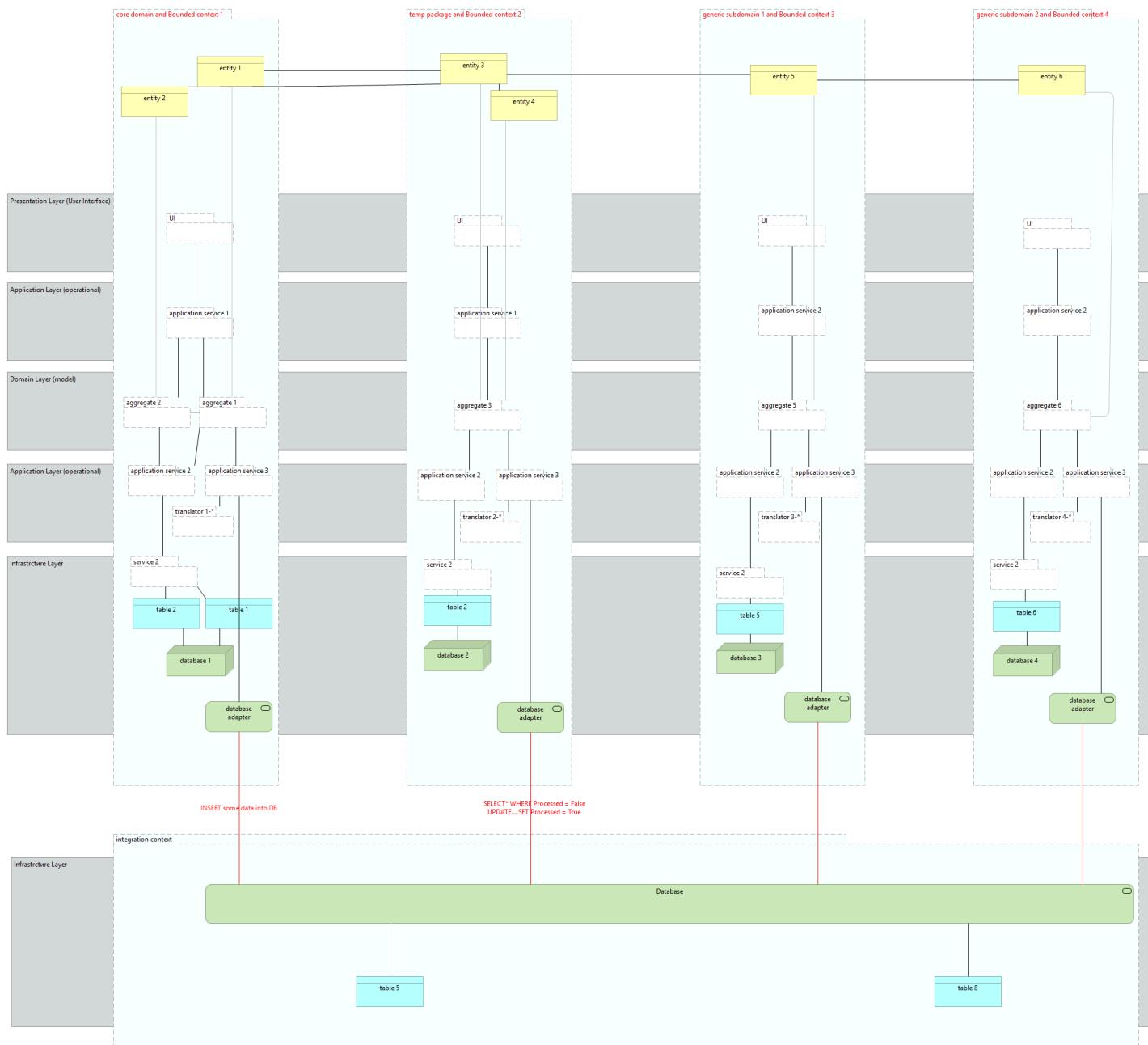


# THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS

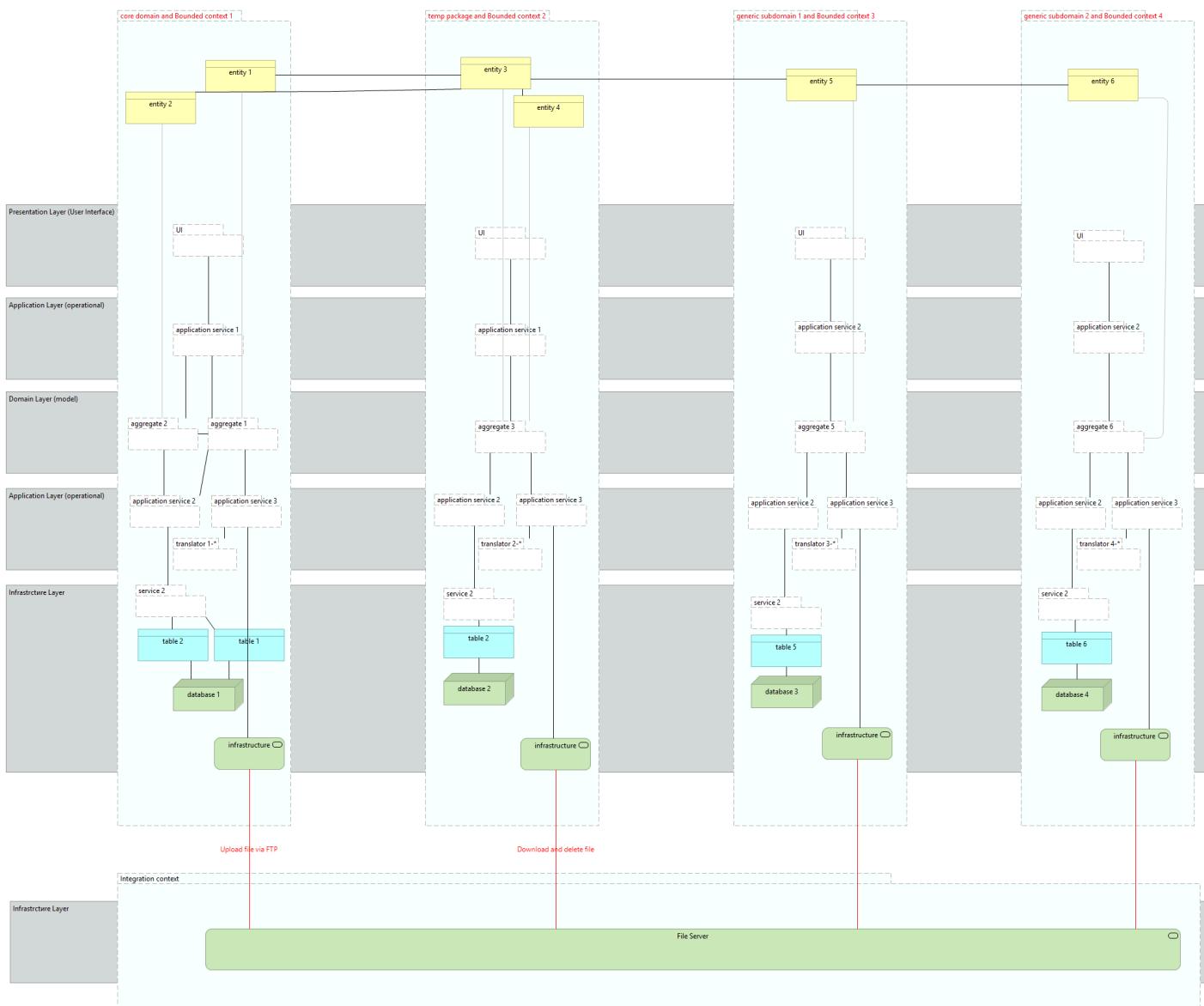
**Example**  
The same physical entity in the problem domain mean different things in different contexts (with its own set of attributes and aspects of behavior).



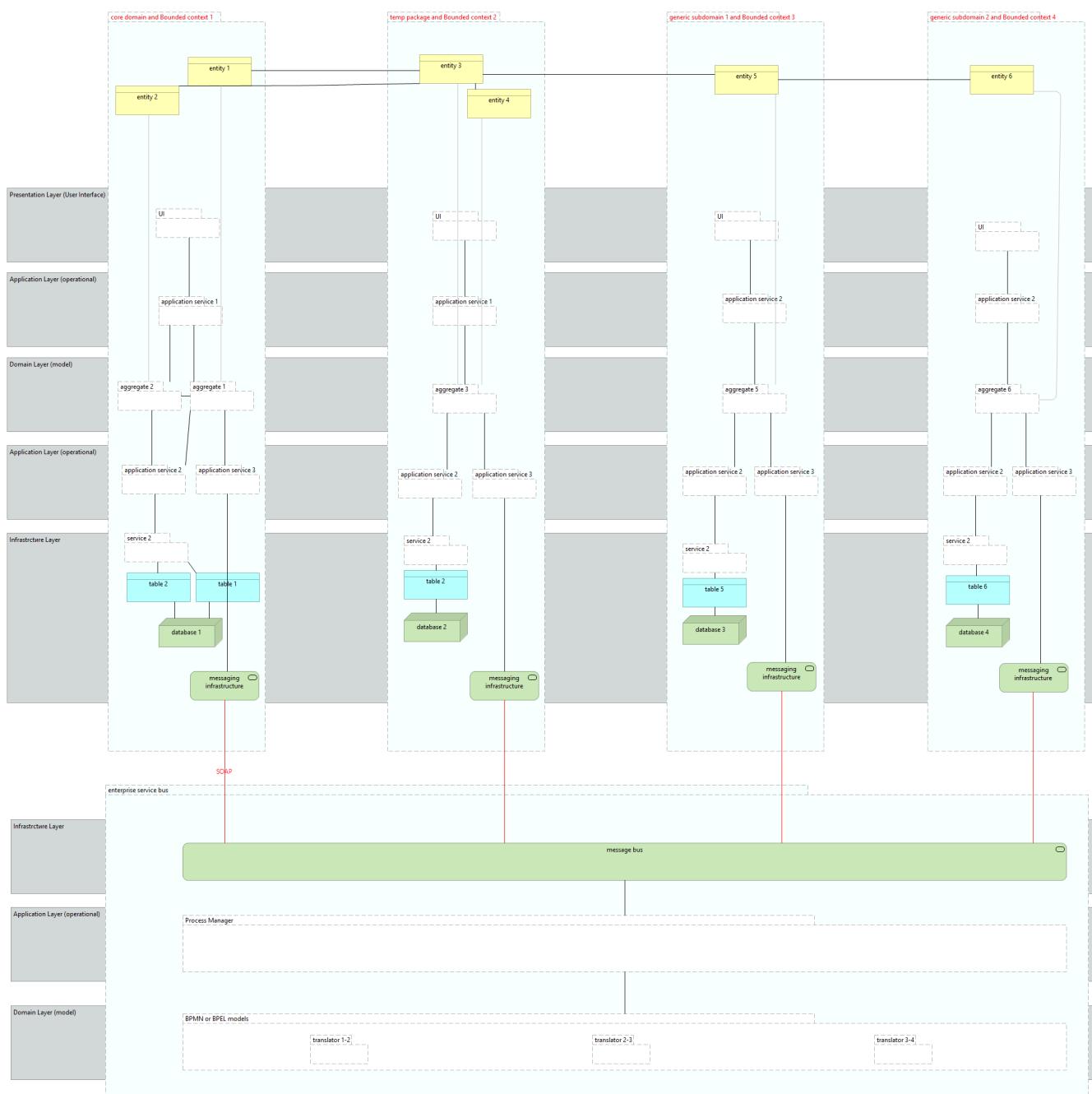
# INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE



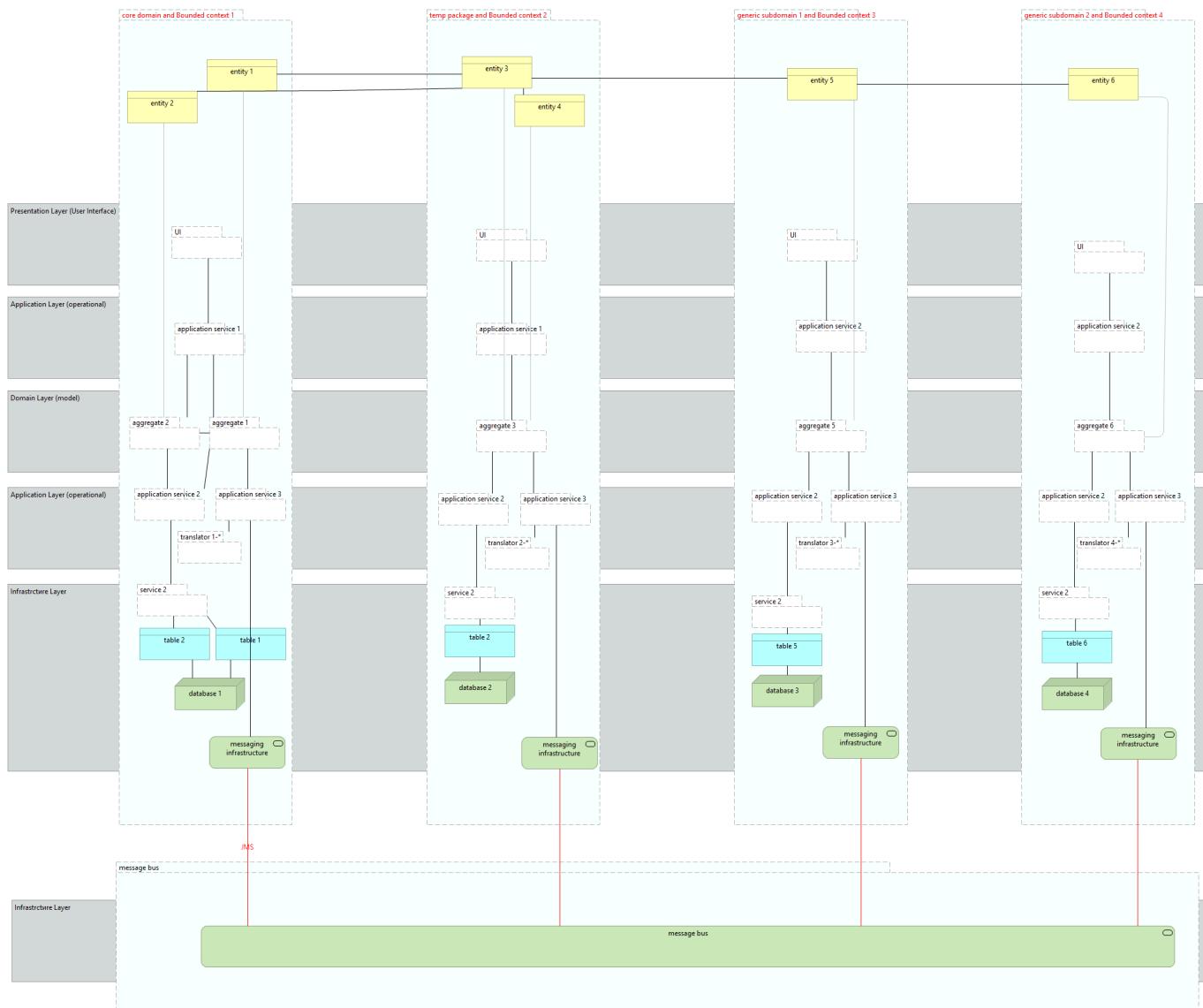
# INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES



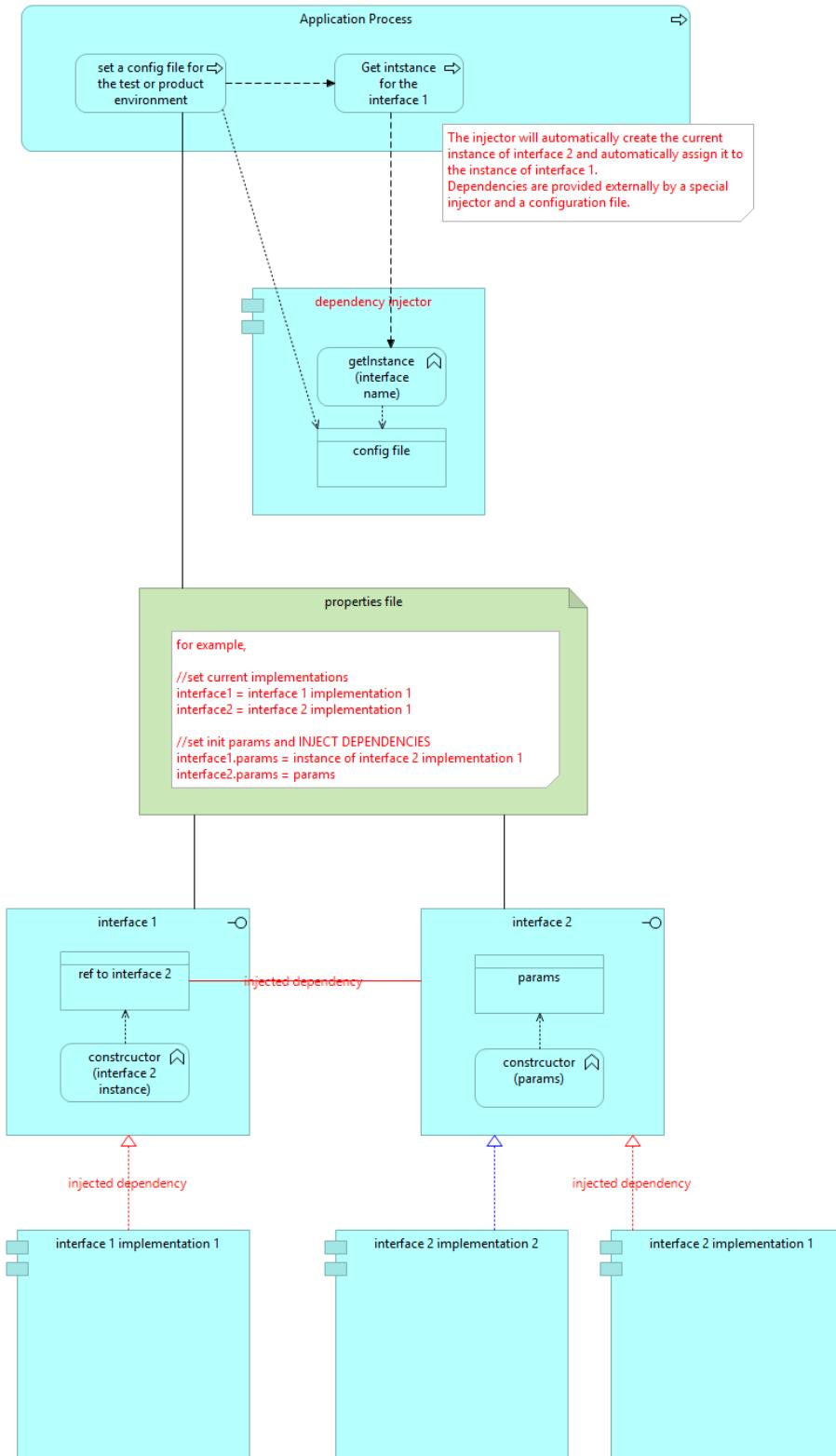
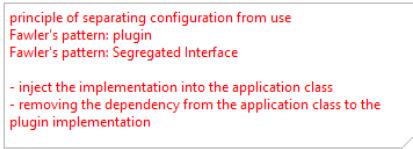
# INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS



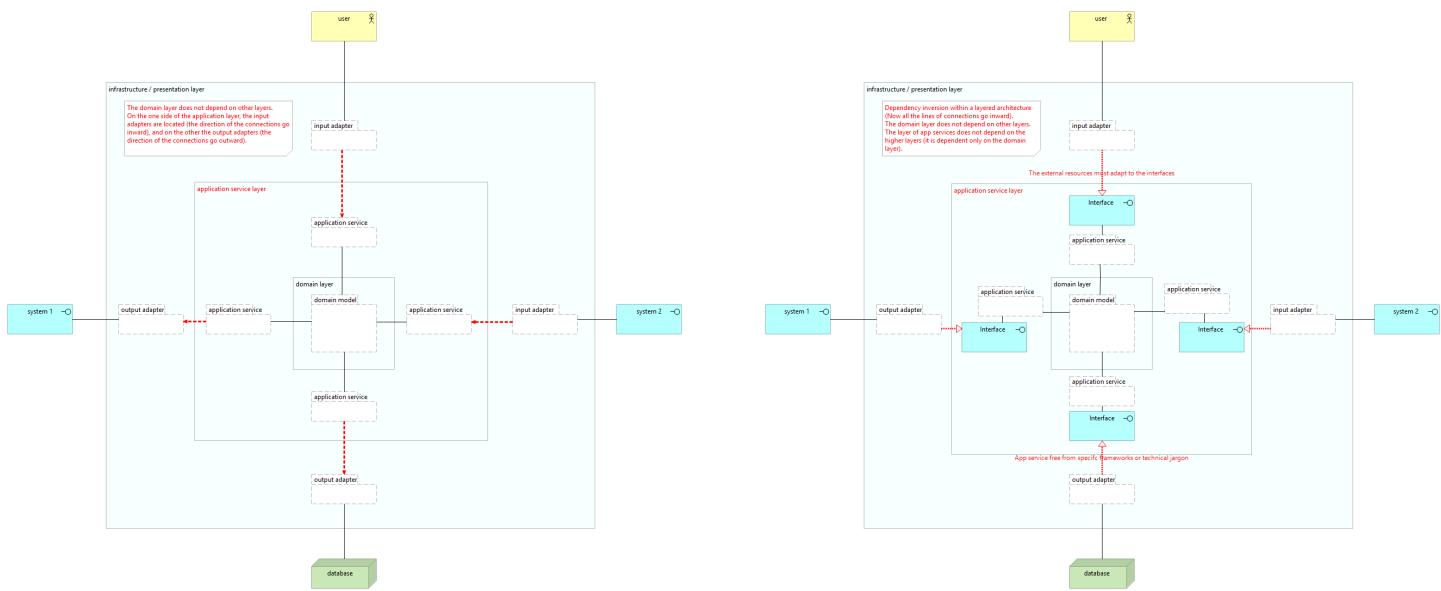
# INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE



# DEPENDENCY INJECTION



# DEPENDENCY INVERSION

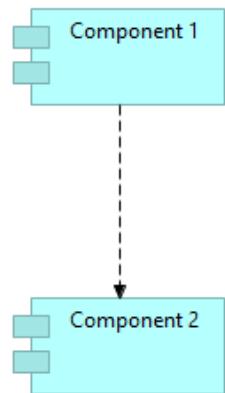


# INVERSION OF CONTROL

IoC

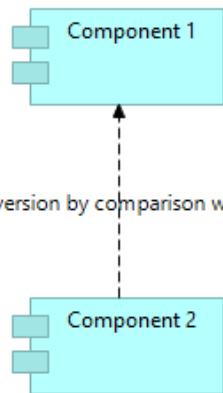
Inversion of control is about who initiates messages.

Variant 1



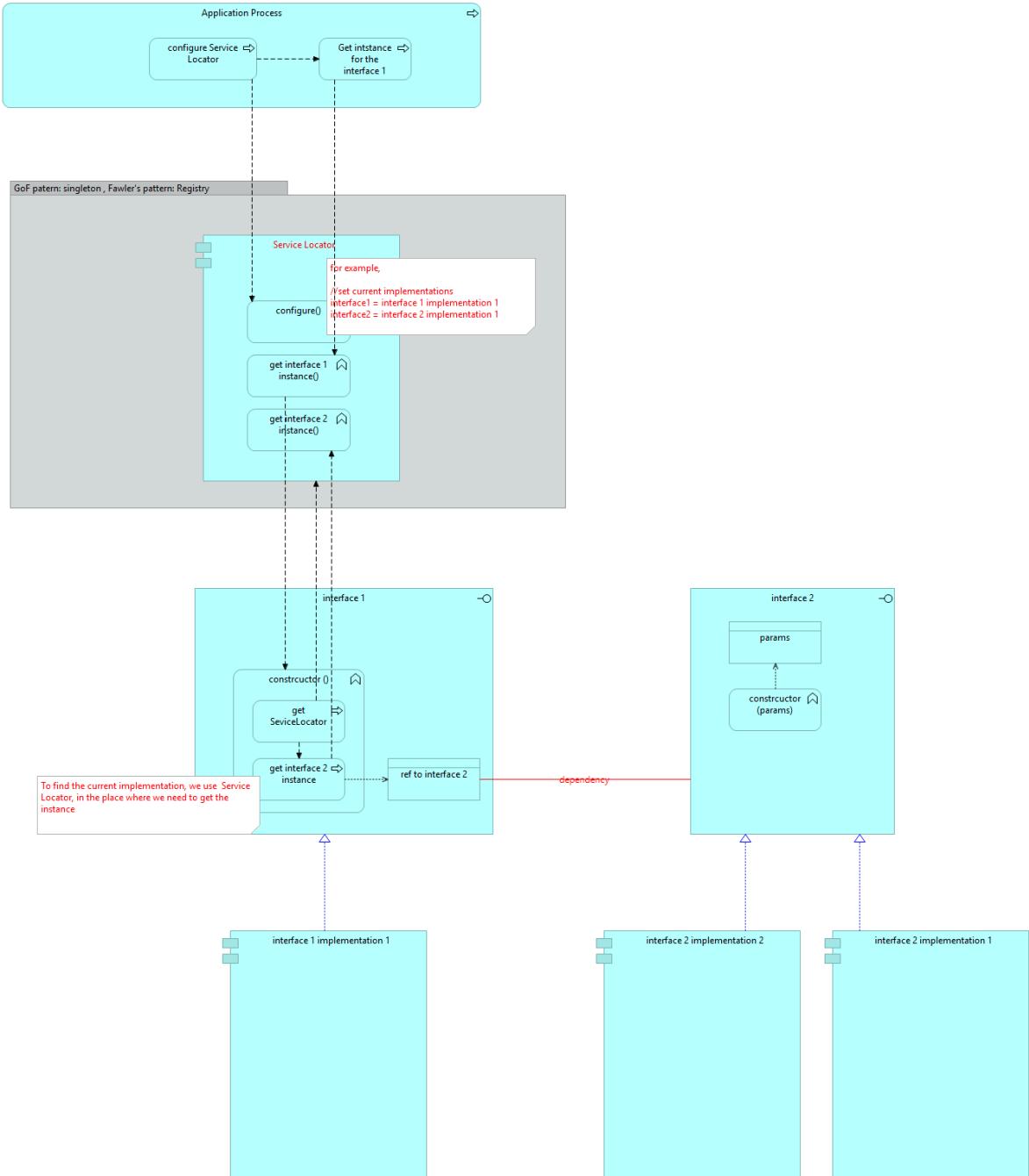
Variant 2

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





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