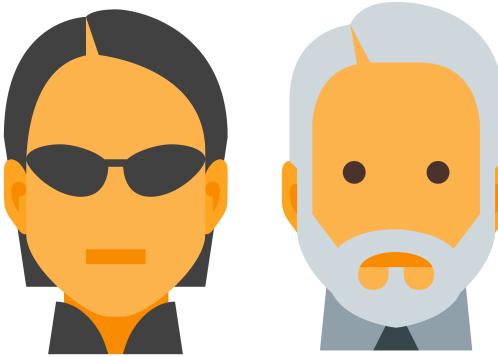


155 PATTERNS



Wilmer Krisp

278 SOFTWARE ARCHITECTURE
DESIGN MODELS

COMPLETE CATALOG OF ALL CLASSICAL
PATTERNS IN THE ARCHIMATE LANGUAGE

03

Analysis Patterns

Reusable Object Models

MARTIN FOWLER

04

Domain Driven Design

Tackling Complexity in the Heart

ERIC EVANS

01

Design Patterns

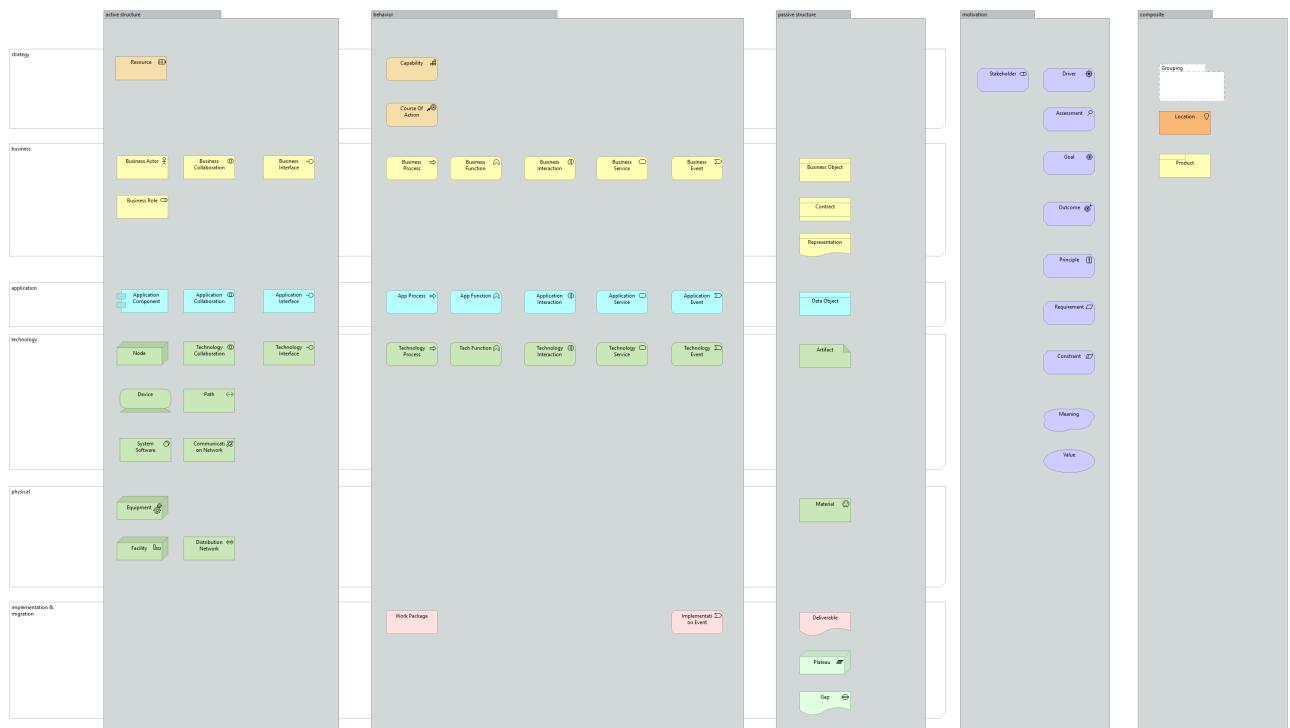
Elements of Reusable Object-

GANG OF FOUR

USED NOTATION

ARCHIMATE METAMODEL

The Open Group



ABSTRACT FACTORY	11
BUILDER	15
FACTORY METHOD	17
PROTOTYPE	19
SINGLETON	21
ADAPTER OF CLASS	24
ADAPTER OF OBJECT	26
BRIDGE	29
COMPOSITE	33
DECORATOR	35
FACADE	37
FLYWEIGHT	40
FLYWEIGHT + COMPOSITE	42
PROXY	43
CHAIN OF RESPONSIBILITY	46
COMMAND	48
INTERPRETER	50
ITERATOR	52
MEDIATOR	54
MEMENTO	56
OBSERVER	58
STATE	60
STRATEGY	62
TEMPLATE METHOD	64
VISITOR	67
DOMAIN MODEL	74
SERVICE LAYER	75
TRANSACTION SCRIPT	76
TABLE MODULE	77
ACTIVE RECORD	79
DATA MAPPER	80
ROW DATA GATEWAY	81
TABLE DATA GATEWAY	82
IDENTITY MAP	84
LAZY LOAD	85
UNIT OF WORK.	86
CLASS TABLE INHERITANCE	88
CONCRETE TABLE INHERITANCE	89
INHERITANCE MAPPERS	90
SINGLE TABLE INHERITANCE	91
ASSOCIATION TABLE MAPPING	93
DEPENDENT MAPPING	94
EMBEDDED VALUE	95
FOREIGN KEY MAPPING	96
IDENTITY FIELD	97
SERIALIZED LOB	98
METADATA MAPPING	100
QUERY OBJECT	101
REPOSITORY	102
MODEL VIEW CONTROLLER	104
APPLICATION CONTROLLER	105
FRONT CONTROLLER	106
PAGE CONTROLLER	107
TEMPLATE VIEW	109
TRANSFORM VIEW	110
TWO STEP VIEW	111
DATA TRANSFER OBJECT	113
REMOTE FAÇADE	114
COARSE-GRAINED LOCK	116
IMPLICIT LOCK	117

OPTIMISTIC OFFLINE LOCK	118
PESSIMISTIC OFFLINE LOCK	119
CLIENT SESSION STATE	121
DATABASE SESSION STATE	122
SERVER SESSION STATE	123
GATEWAY	125
LAYER SUPER TYPE	126
MAPPER	127
MONEY	128
PLUGIN	129
RECORD SET	130
REGISTRY	131
SEPARATED INTERFACE	132
SERVICE STUB	133
SPECIAL CASE	134
VALUE OBJECT	135
PARTY	139
ACCOUNTABILITY	140
ORGANIZATION HIERARCHIES	142
ORGANIZATION STRUCTURE	143
ACCOUNTABILITY KNOWLEDGE LEVEL	144
PARTY TYPE GENERALIZATIONS	145
HIERARCHIC ACCOUNTABILITY	146
OPERATING SCOPES	147
POST	148
QUANTITY	150
CONVERSION RATIO	151
OBSERVATIONS AND MEASUREMENTS	152
COMPOUND UNITS	153
MEASUREMENT	154
OBSERVATION	155
SUBTYPING OBSERVATION CONCEPTS	156
PROTOCOL	157
DUAL TIME RECORD	158
REJECTED OBSERVATION	159
ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION	160
ASSOCIATED OBSERVATION	161
PROCESS OF OBSERVATION	162
ENTERPRISE SEGMENT	164
MEASUREMENT PROTOCOL	165
RANGE	166
OBSERVATIONS FOR CORPORATE FINANCE	167
PHENOMENON WITH RANGE	170
NAME	172
IDENTIFICATION SCHEME	173
OBJECT MERGE	174
OBJECT EQUIVALENCE	175
REFERRING TO OBJECTS	176
ACCOUNT	178
TRANSACTIONS	179
SUMMARY ACCOUNT	180
MEMO ACCOUNT	181
POSTING RULES	182
INVENTORY AND ACCOUNTING	183
INDIVIDUAL INSTANCE METHOD	184
POSTING RULE EXECUTION	185
POSTING RULES FOR MANY ACCOUNTS	186
CHOOSING ENTRIES	187
ACCOUNTING PRACTICE	188
SOURCES OF AN ENTRY	189

BALANCE SHEET AND INCOME STATEMENT	190
CORRESPONDING ACCOUNT	191
SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)	192
SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)	193
BOOKING ENTRIES TO MULTIPLE ACCOUNTS	194
PROPOSED AND IMPLEMENTED ACTION	196
COMPLETED AND ABANDONED ACTIONS	197
SUSPENSION	198
PLAN	199
PROTOCOL	200
RESOURCE ALLOCATION	201
PLANNING	202
PLANNING (NO OUTCOME)	203
OUTCOME AND START FUNCTIONS	204
CONTRACT	206
PORTFOLIO	207
QUOTE	208
SCENARIO	209
TRADING	210
FORWARD CONTRACTS	212
OPTIONS	213
PRODUCT	214
SUBTYPE STATE MACHINES	215
PARALLEL APPLICATION AND DOMAIN HIERARCHIES	216
DERIVATIVE CONTRACTS	217
MULTIPLE ACCESS LEVELS TO A PACKAGE	219
MUTUAL VISIBILITY	220
TRADING PACKAGES	221
TWO-TIER ARCHITECTURE	223
THREE-TIER ARCHITECTURE	224
PRESENTATION AND APPLICATION LOGIC	225
DATABASE INTERACTION	226
IMPLEMENTING ASSOCIATIONS	228
IMPLEMENTING GENERALIZATION	229
OBJECT CREATION	230
OBJECT DESTRUCTION	231
ENTRY POINT.	232
IMPLEMENTING CONSTRAINTS	233
MODEL-DRIVEN DESIGN	236
LAYERED ARCHITECTURE (ASYMMETRIC)	239
HEXAGONAL ARCHITECTURE (SYMMETRIC)	241
COMPOSITE UI	245
ENTITIES	246
VALUE-OBJECTS	248
DOMAIN SERVICES	250
MODULES	255
AGGREGATES	256
AGGREGATE ROOT	258
BEHAVIOR-FOCUSED AGGREGATE ROOT	259
MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION	260
PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES	261
REFERENCE OTHER AGGREGATES BY IDENTITY ONLY	262
FACTORIES	263
REPOSITORIES	265
UBIQUITOUS LANGUAGE	270
INTENTION-REVEALING INTERFACES	271
SIDE-EFFECT FREE FUNCTIONS	272
ASSERTIONS	273
CONCEPTUAL CONTOURS	274
STANDALONE CLASSES	275

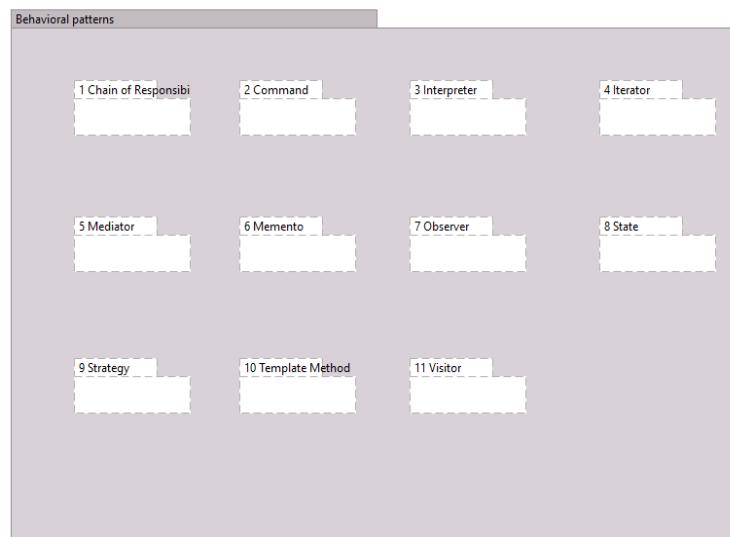
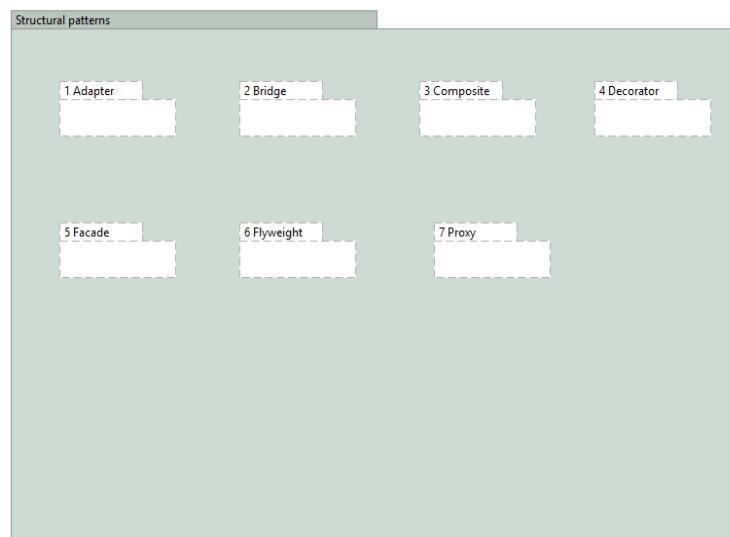
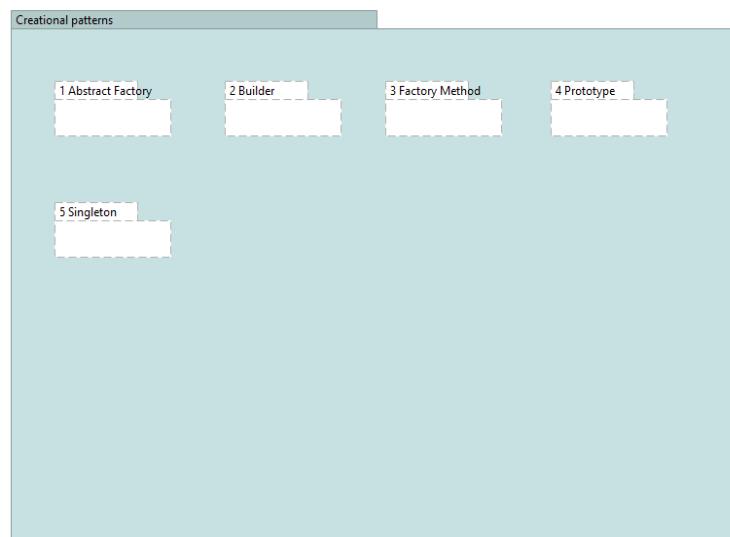
CLOSURE OF OPERATIONS	276
BOUNDED CONTEXT	278
CONTINUOUS INTEGRATION	279
STRATEGIC CONTEXT MAP	280
CONTEXTUAL MAP	281
SHARED KERNEL	282
CUSTOMER-SUPPLIER TEAMS	283
CONFORMIST	284
ANTICORRUPTION LAYER	285
SEPARATE WAYS	286
OPEN HOST SERVICE	287
PUBLISHED LANGUAGE	288
CORE DOMAIN	290
GENERIC SUBDOMAINS	291
DOMAIN VISION STATEMENT	292
HIGHLIGHTED CORE	293
COHESIVE MECHANISMS	294
SEGREGATED CORE	295
ABSTRACT CORE	296
EVOLVING ORDER	298
SYSTEM METAPHOR	299
RESPONSIBILITY LAYERS	300
KNOWLEDGE LEVEL	304
PLUGGABLE COMPONENT FRAMEWORK	305
TYPES OF CONSISTENCY	307
EVENT SOURCING	308
EVENT PROCESSOR	309
EVENT DISPATCHER	310
INTERNAL DOMAIN EVENTS	311
EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS	312
STATIC DOMAIN EVENTS CLASS	313
ONE SUBDOMAIN PER BOUNDED CONTEXT	314
THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS	315
THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS	316
INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE	317
INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES	318
INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS	319
INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE	320
DEPENDENCY INJECTION	321
DEPENDENCY INVERSION	323
INVERSION OF CONTROL	324
SERVICE LOCATOR	325
CQRS	326
CQS	327
WRAP LOW-LEVEL EXCEPTIONS	328
EXTRACT DEPENDENCY FROM INTERFACE TO COSNTRUCTOR	329
INTERFACE SEGREGATION	330
CLEAN ARCHITECTURE	331
ARCHITECTURAL STYLES	337

01

Design Patterns

Elements of Reusable Object-

GANG OF FOUR



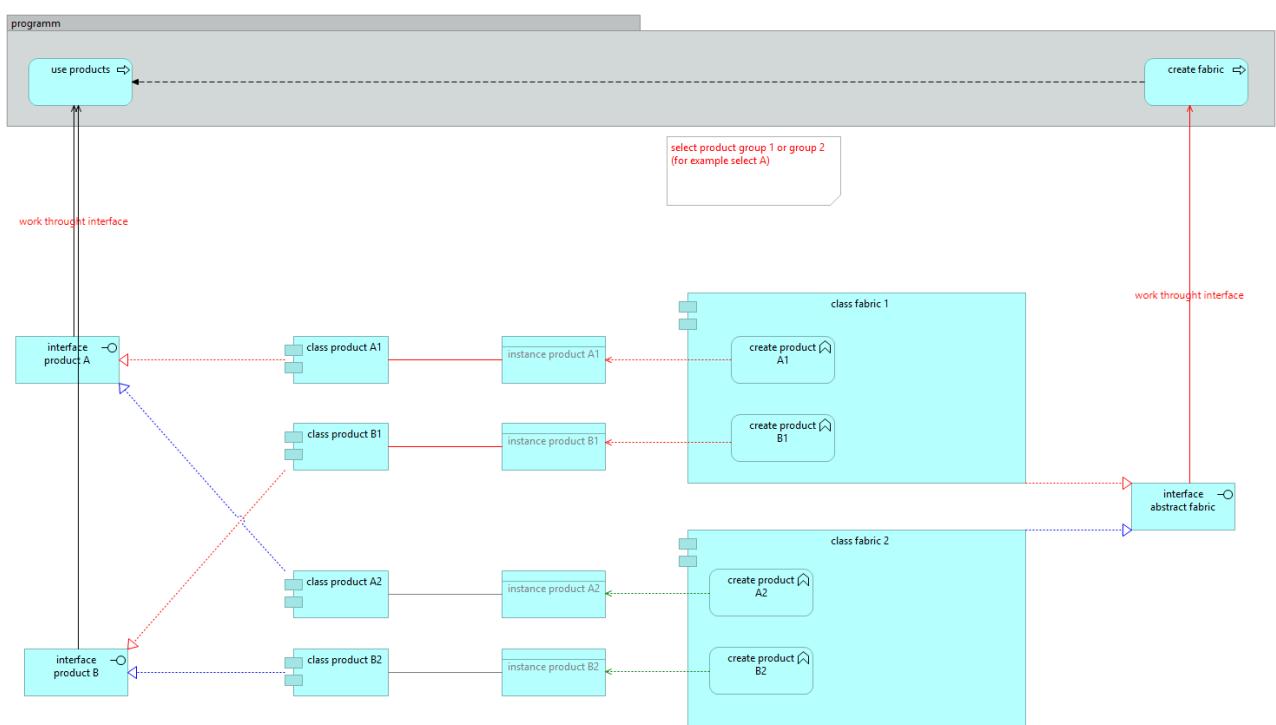
CREATIONAL PATTERNS

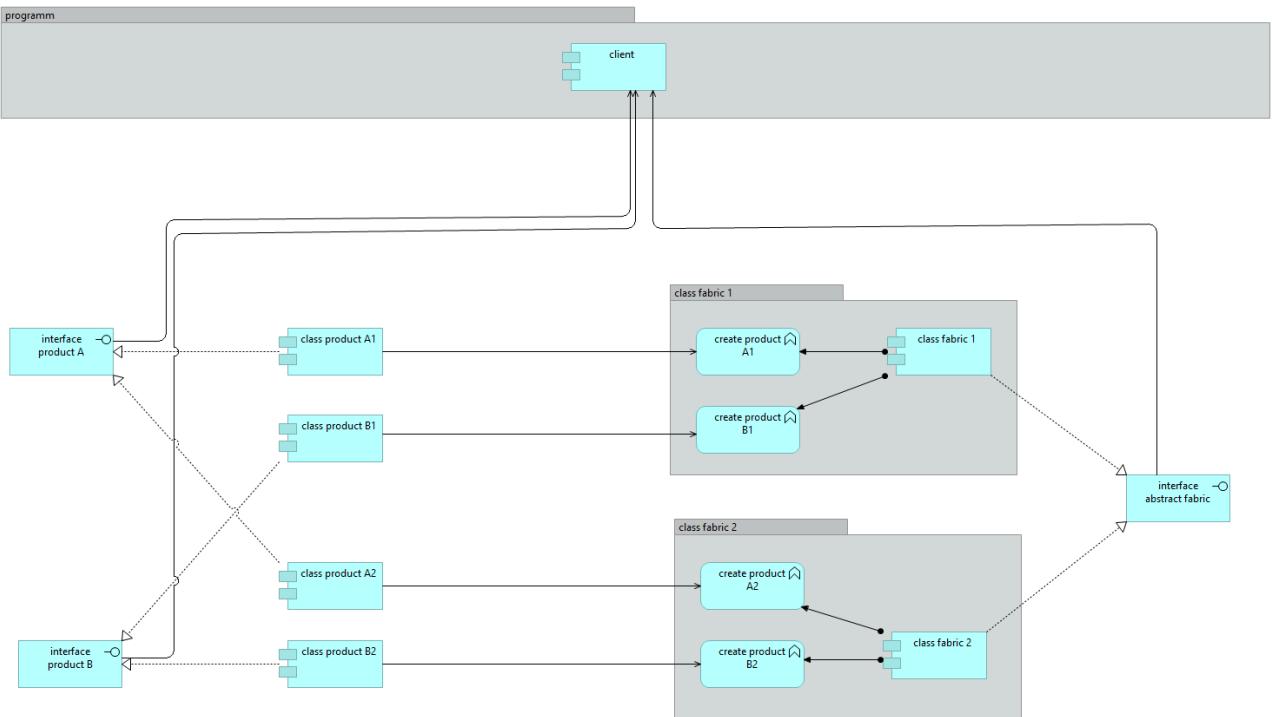
DESIGN PATTERNS

GoF

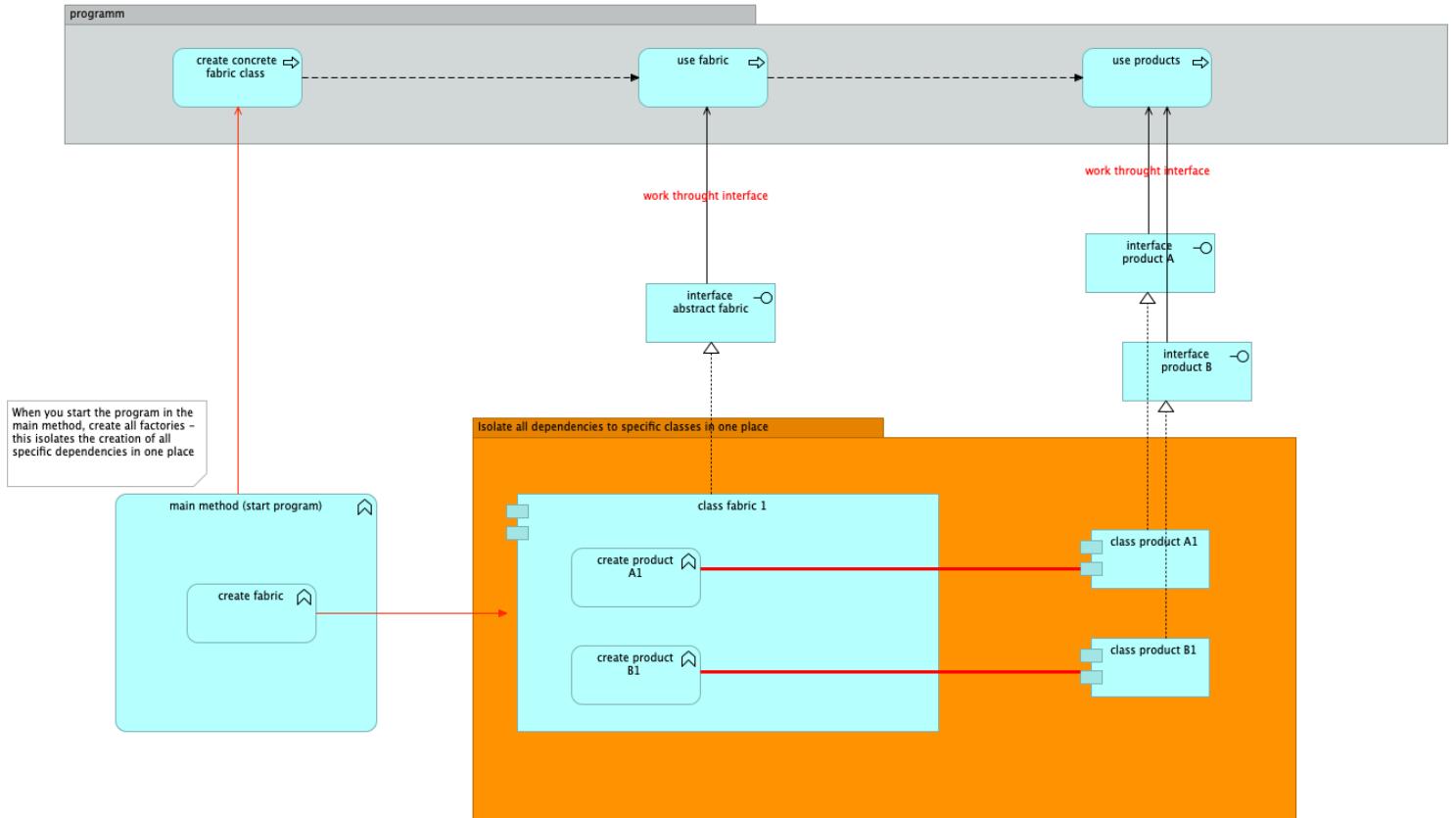


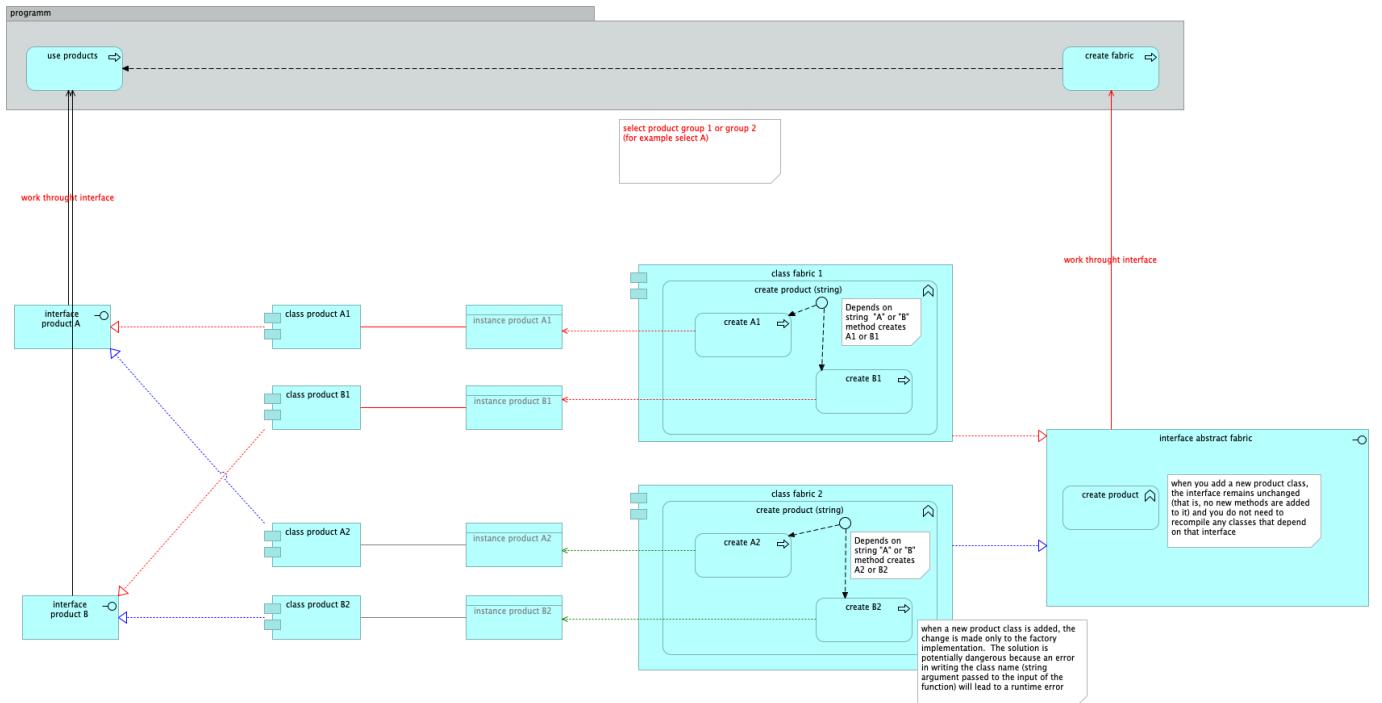
ABSTRACT FACTORY



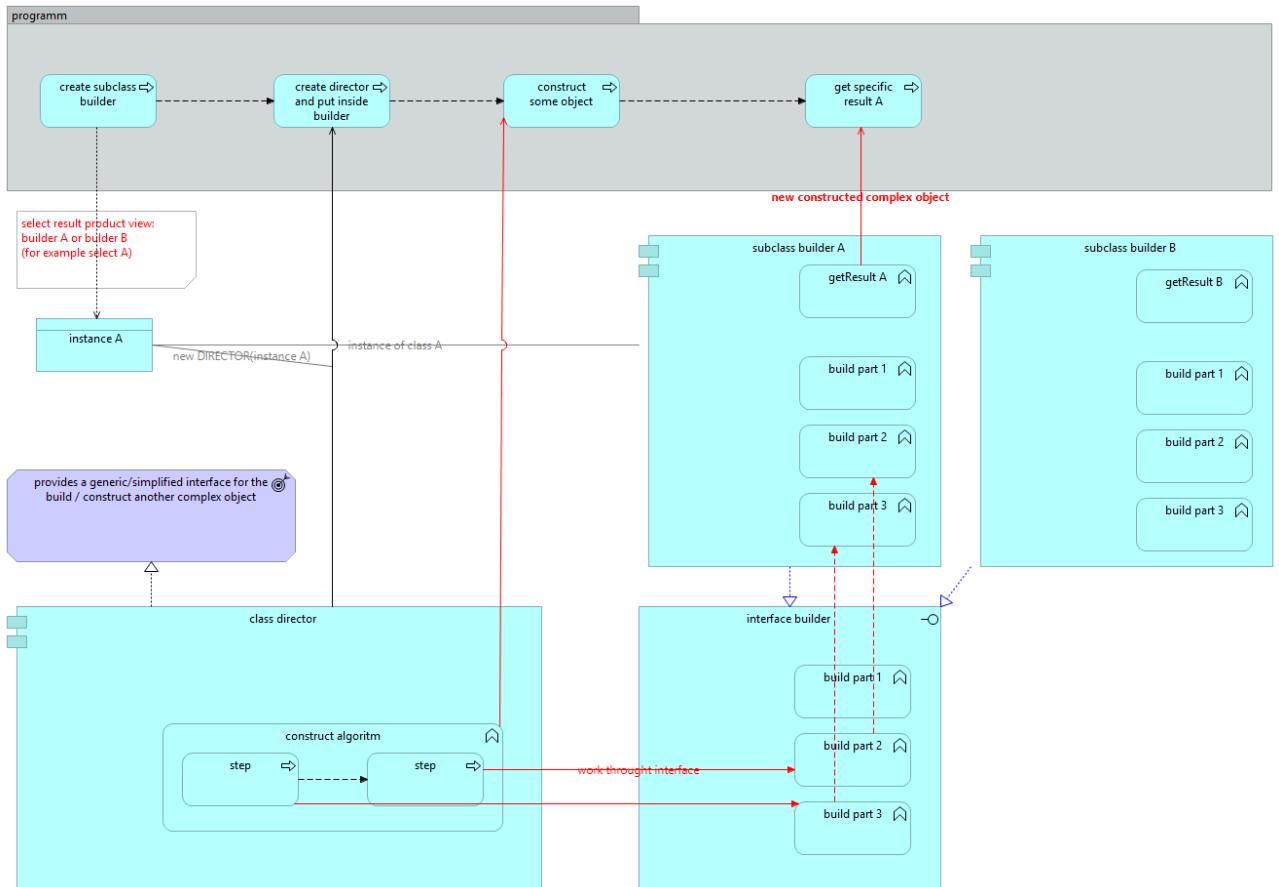


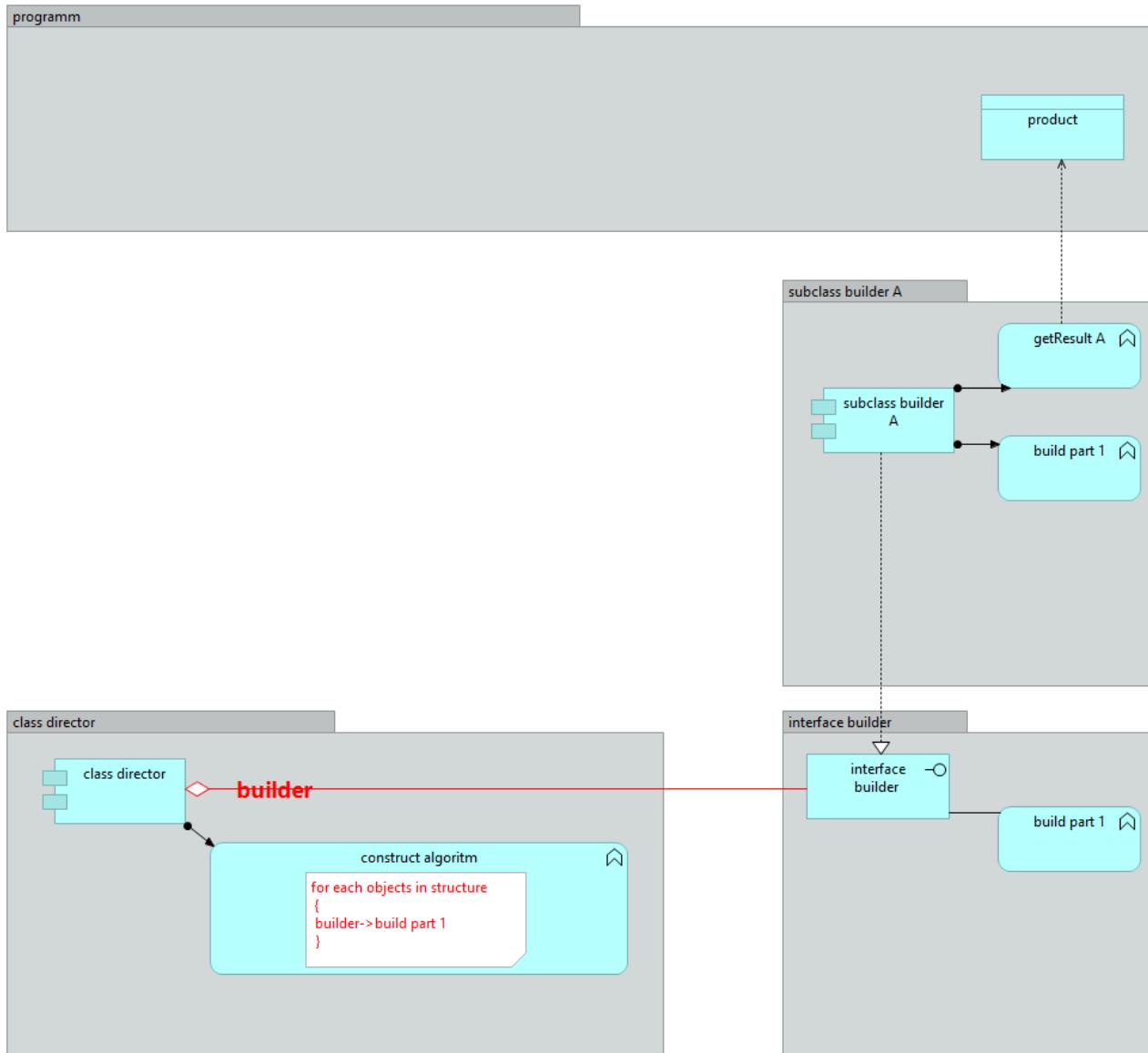
In the whole program (except for the factory inside and one factory creation) we work only with interfaces (and do not refer to specific classes)



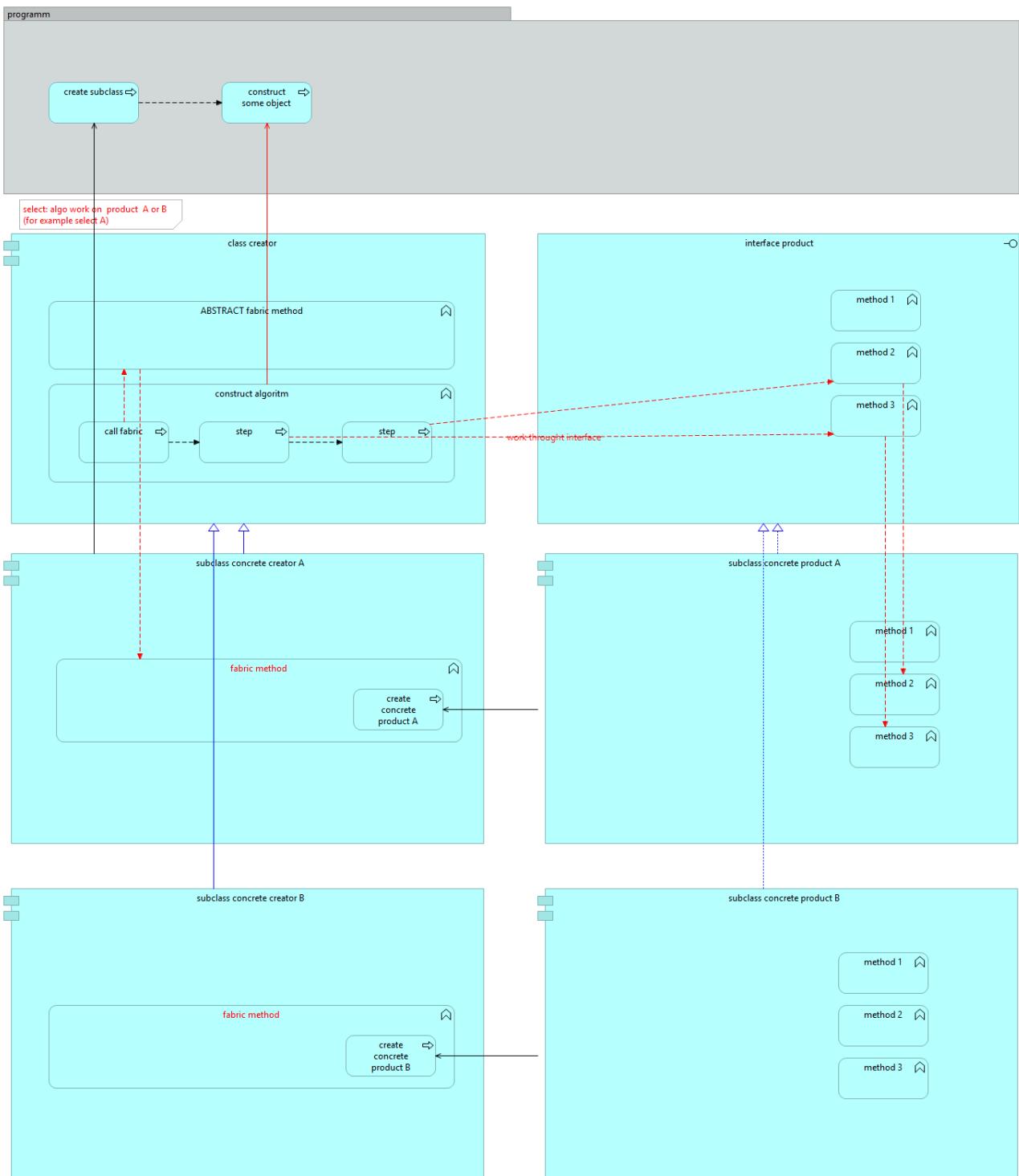


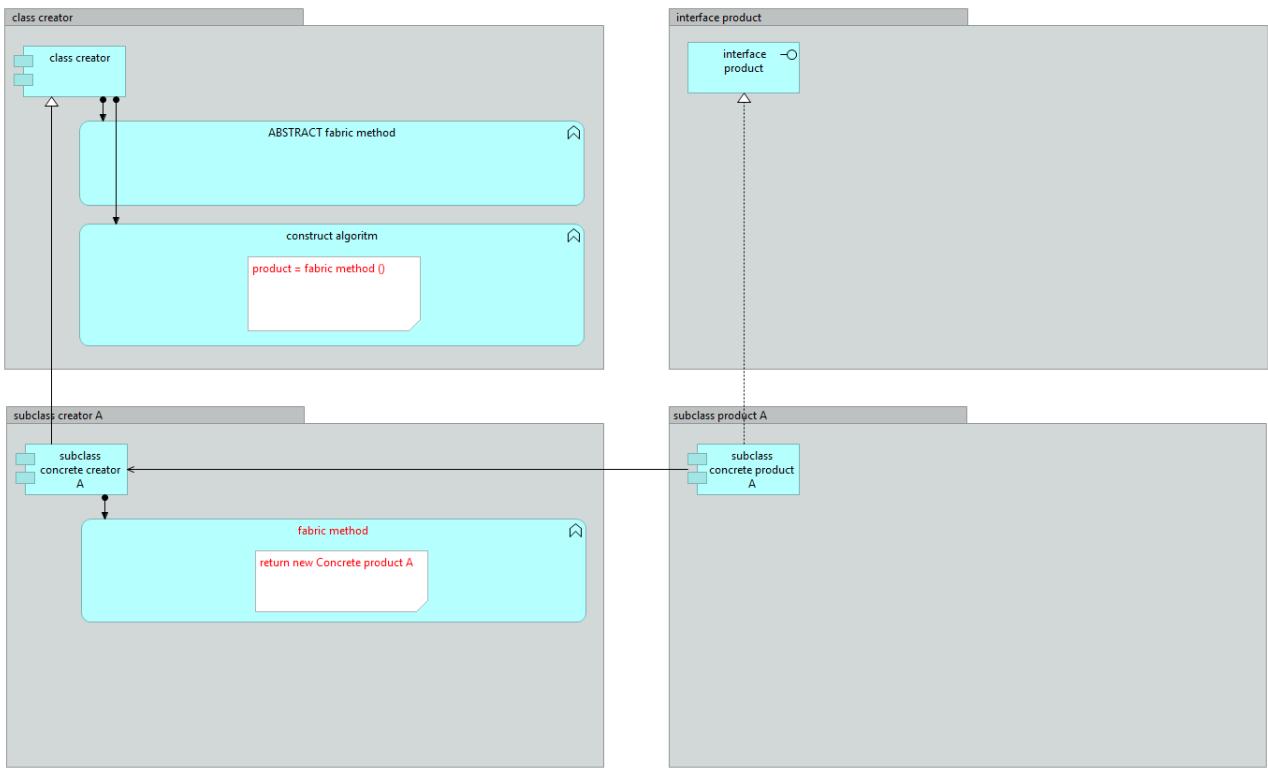
BUILDER



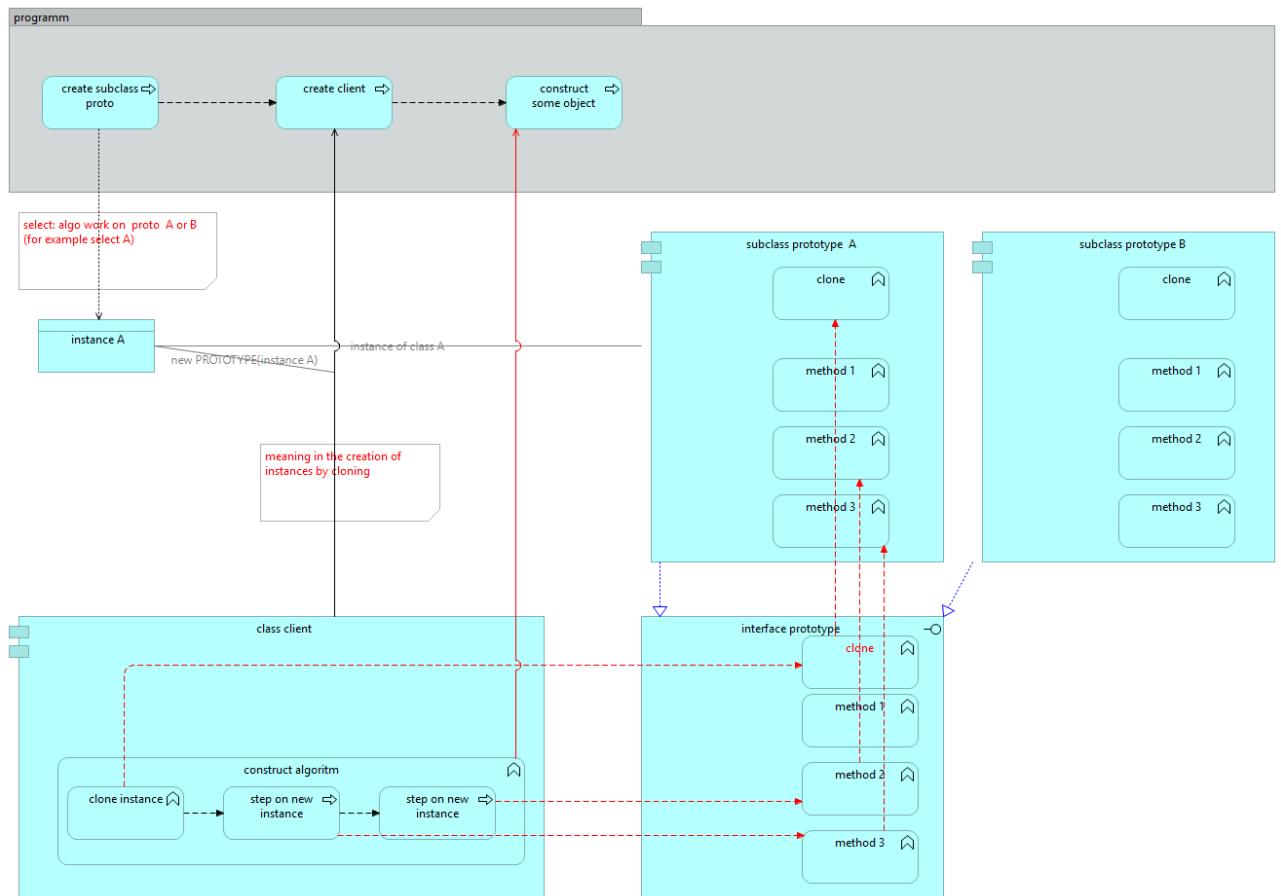


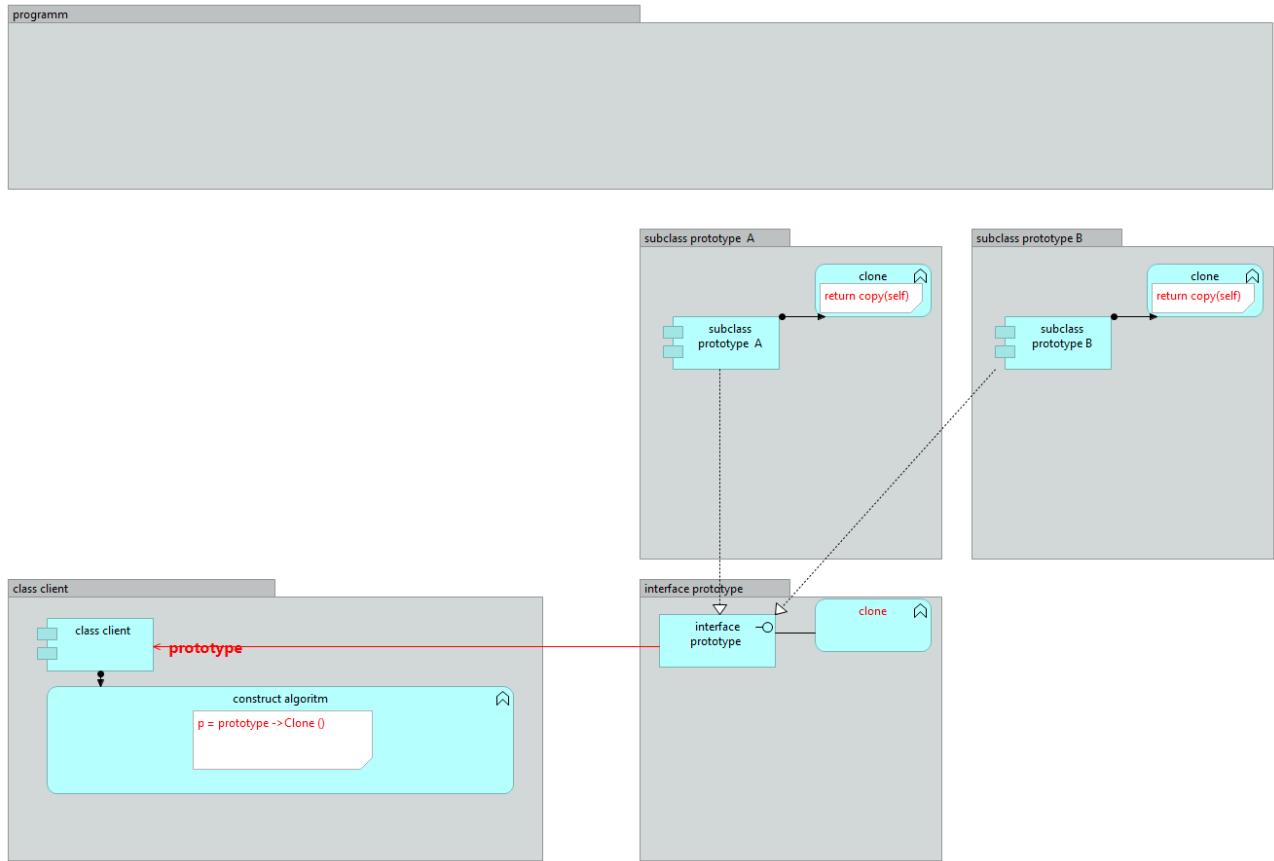
FACTORY METHOD



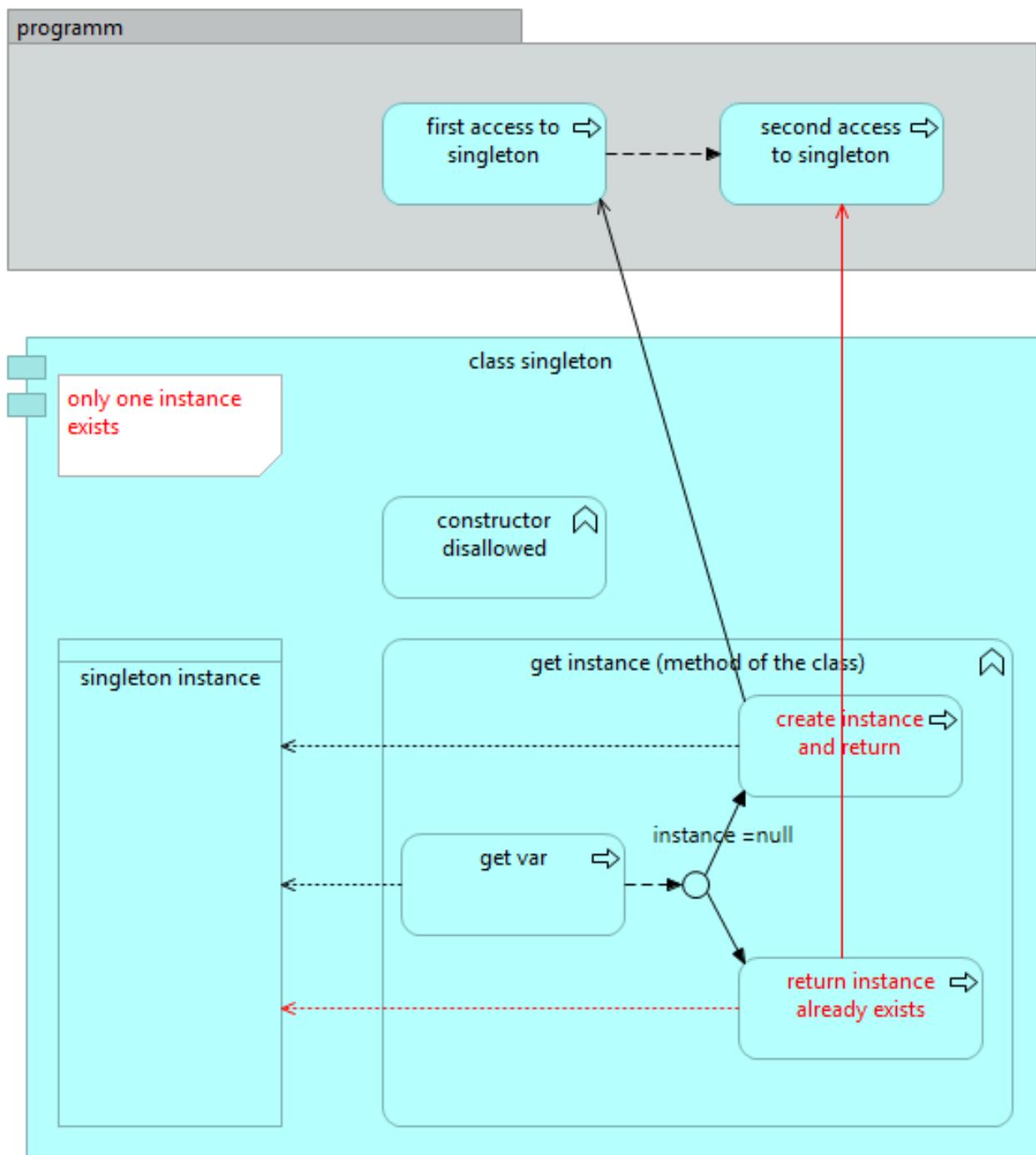


PROTOTYPE



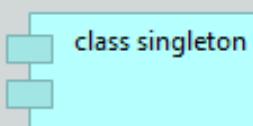


SINGLETON



programm

class singleton



singleton instance

get instance (method of the class)



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

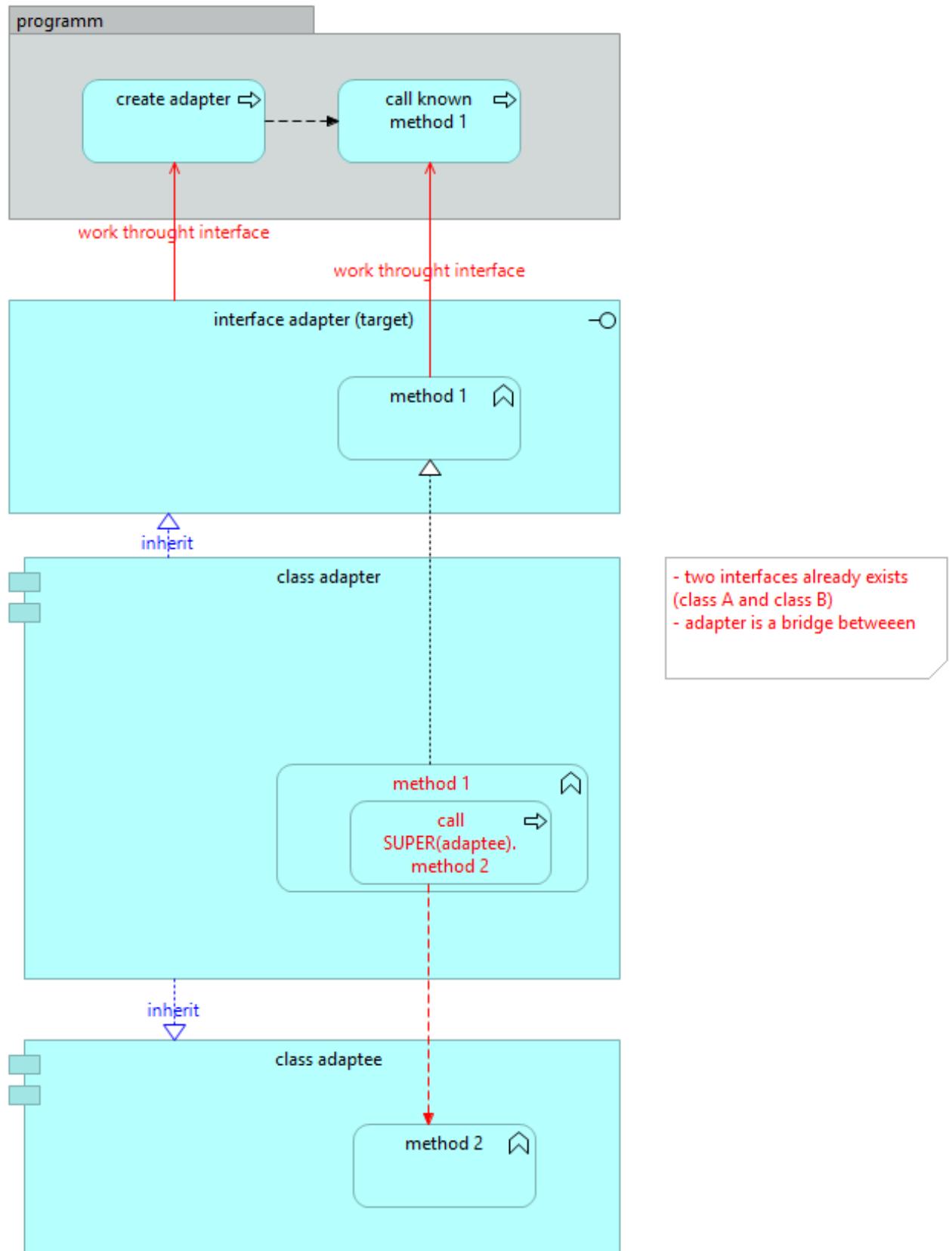
STRUCTURAL PATTERNS

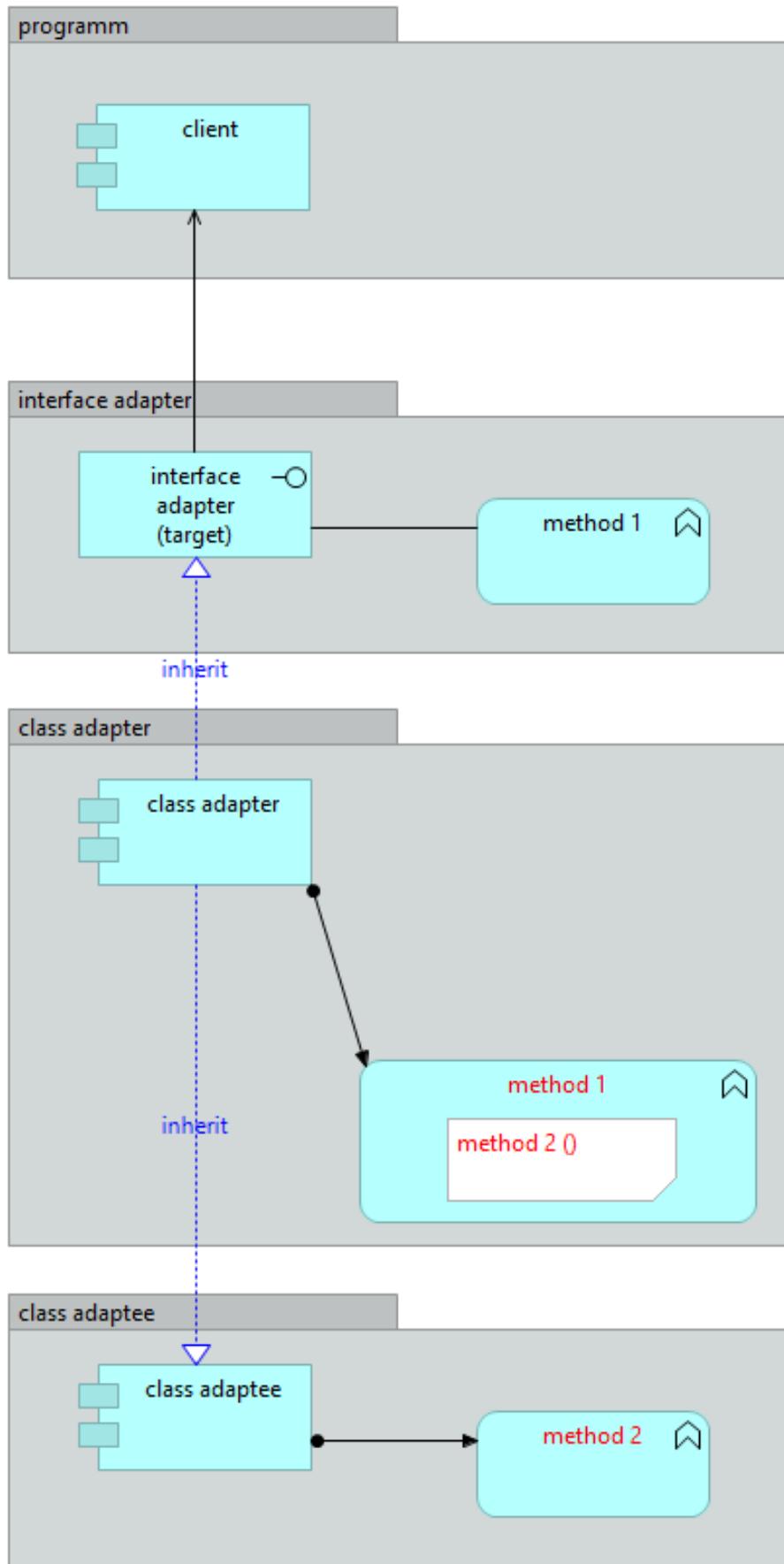
DESIGN PATTERNS

GoF

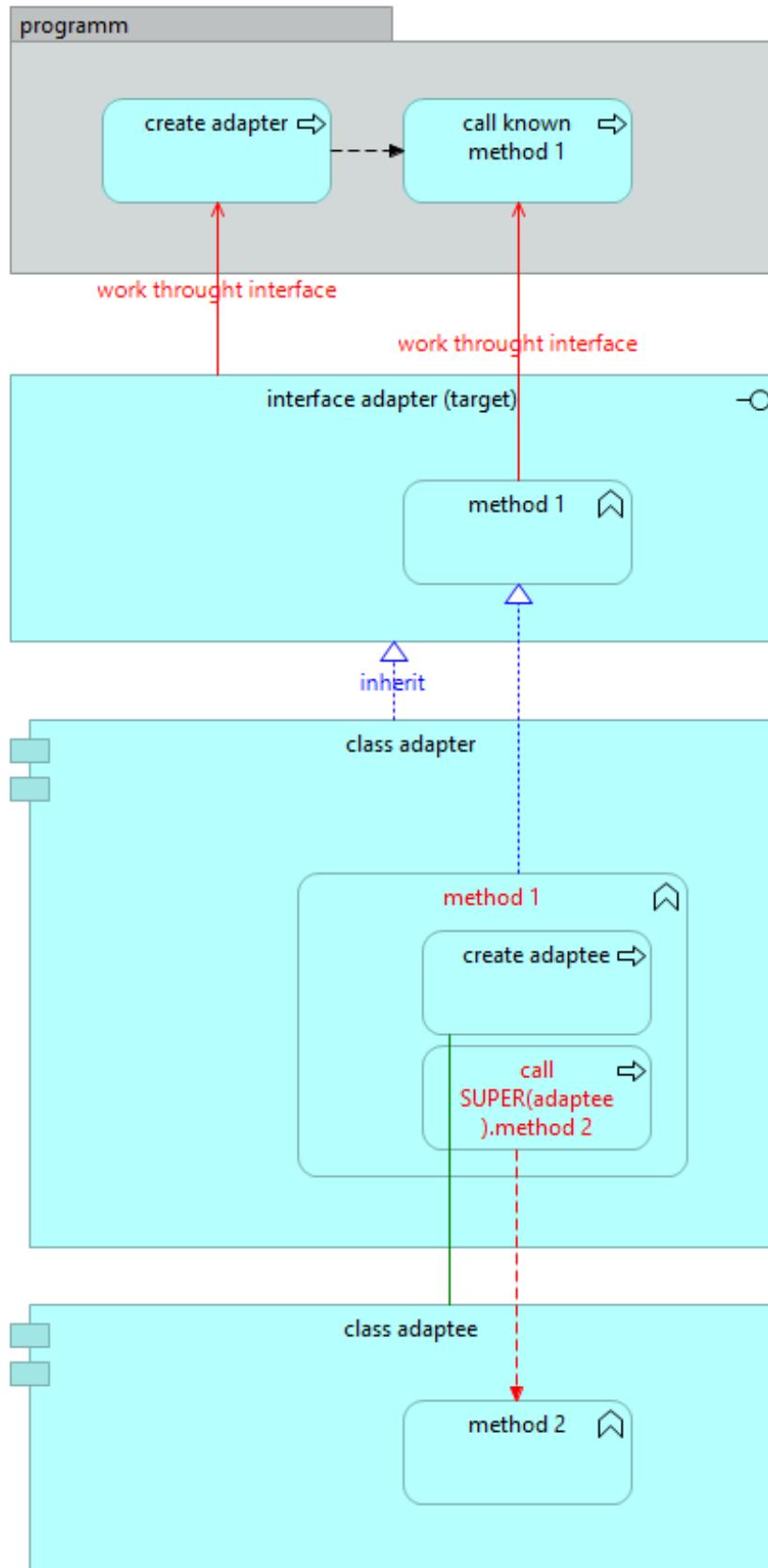


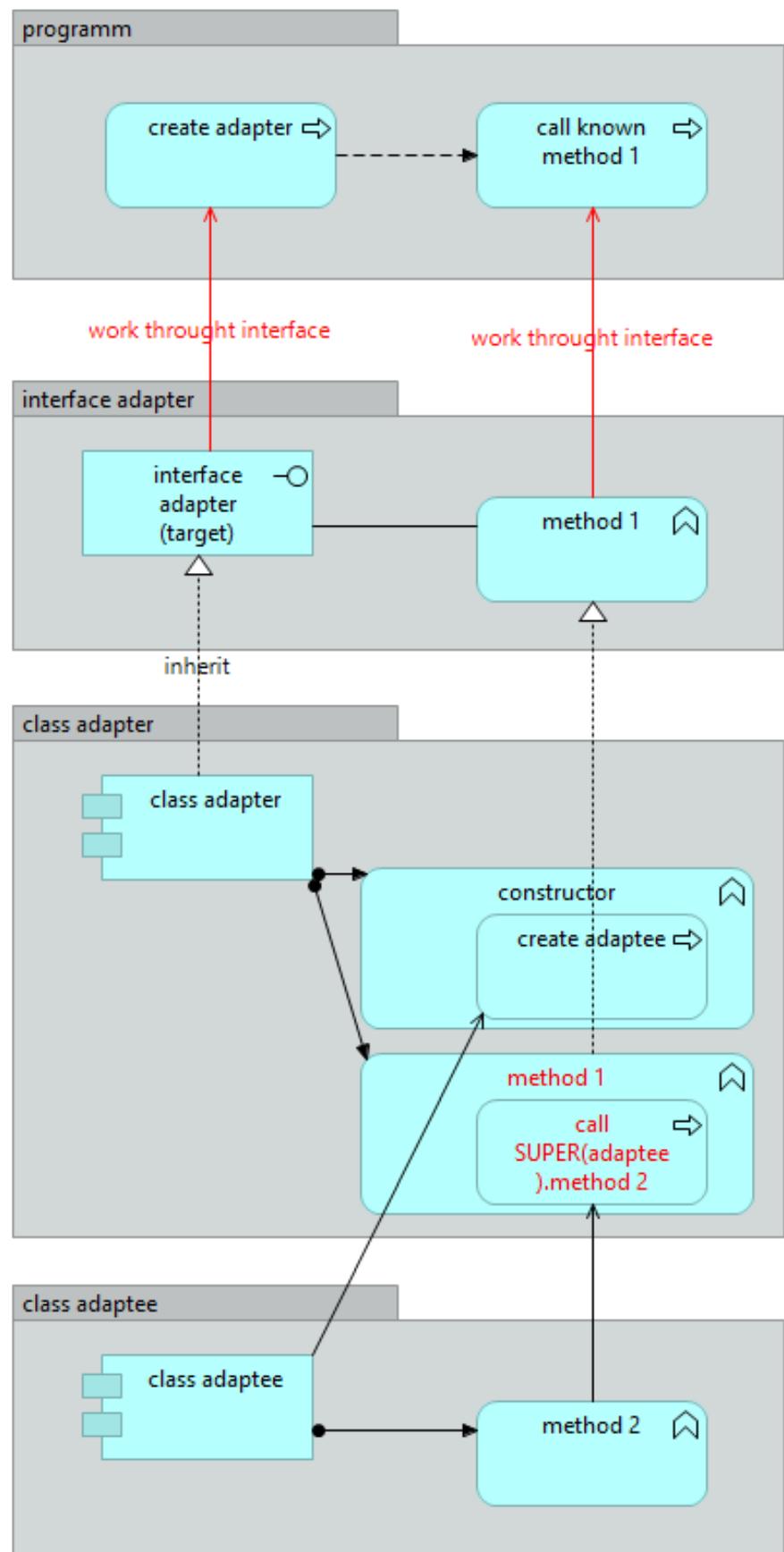
ADAPTER OF CLASS

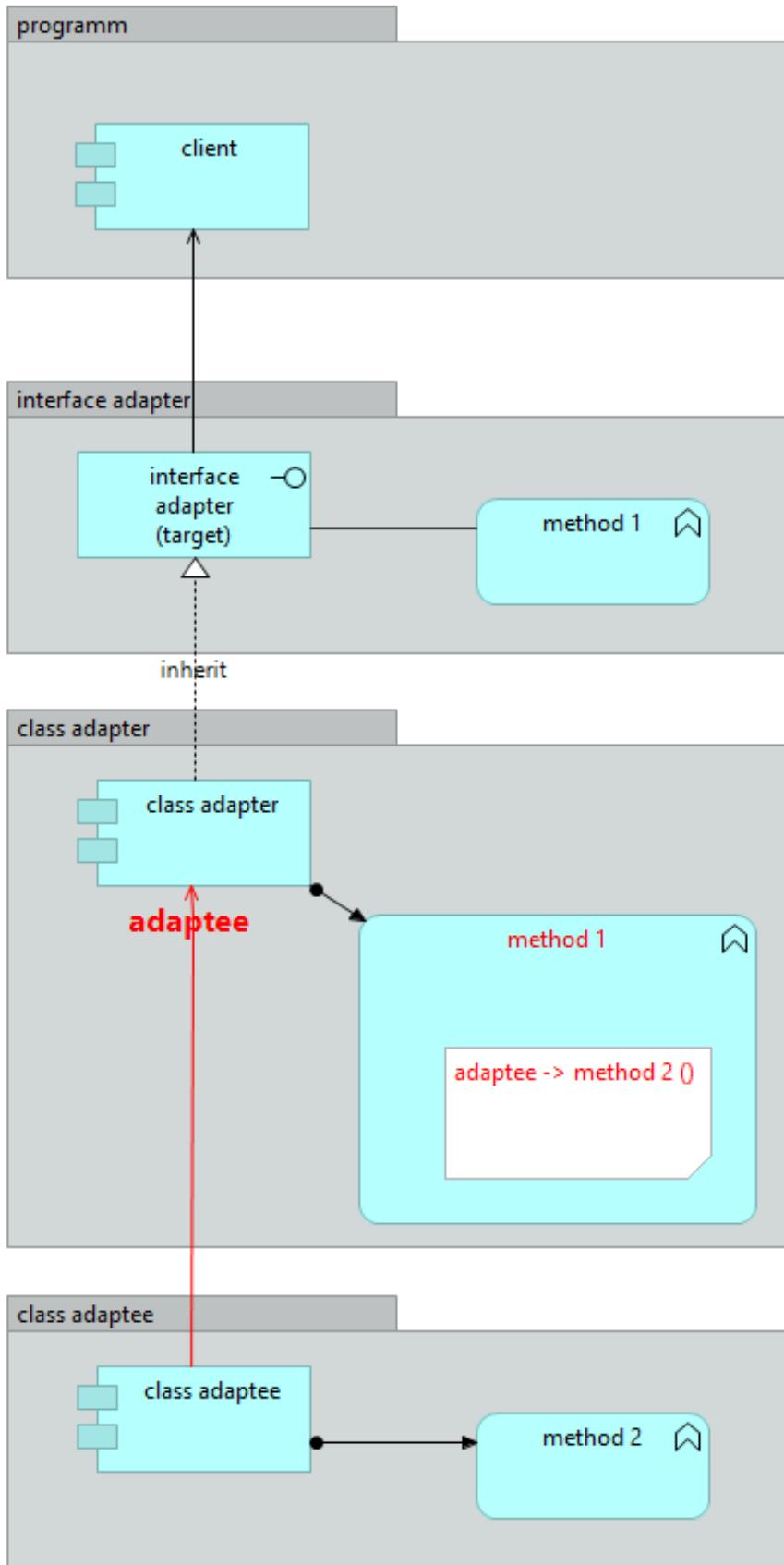




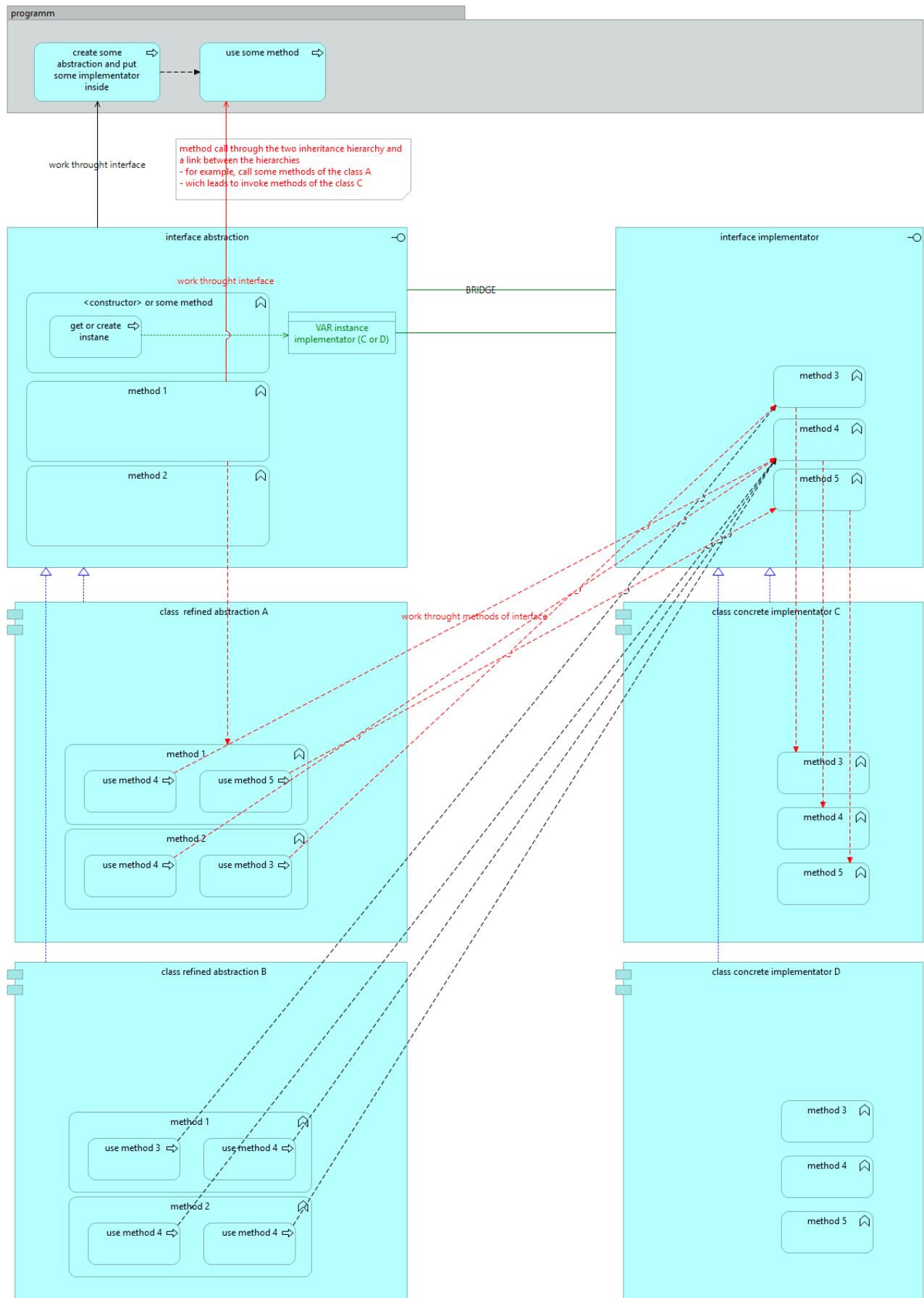
ADAPTER OF OBJECT

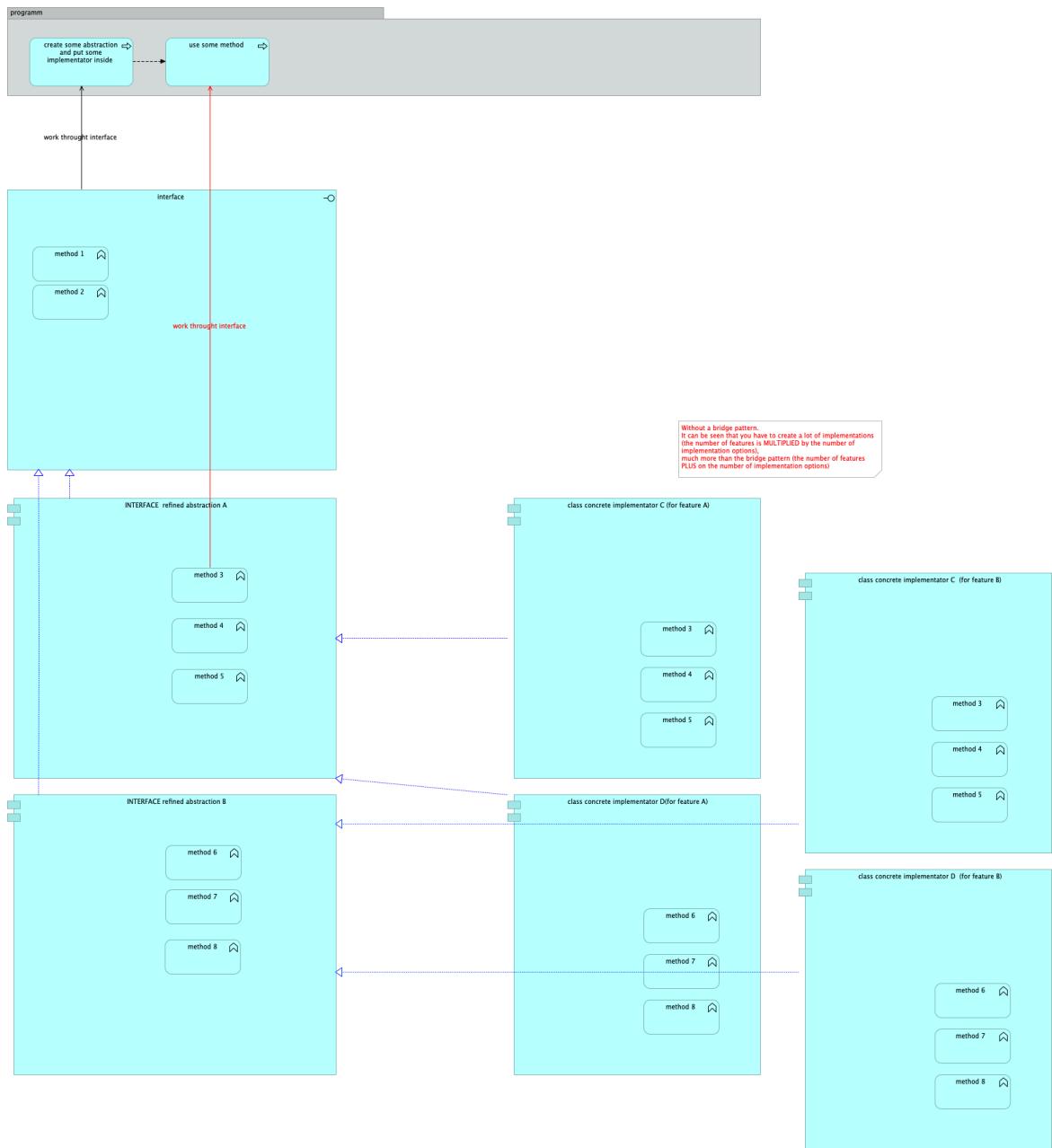


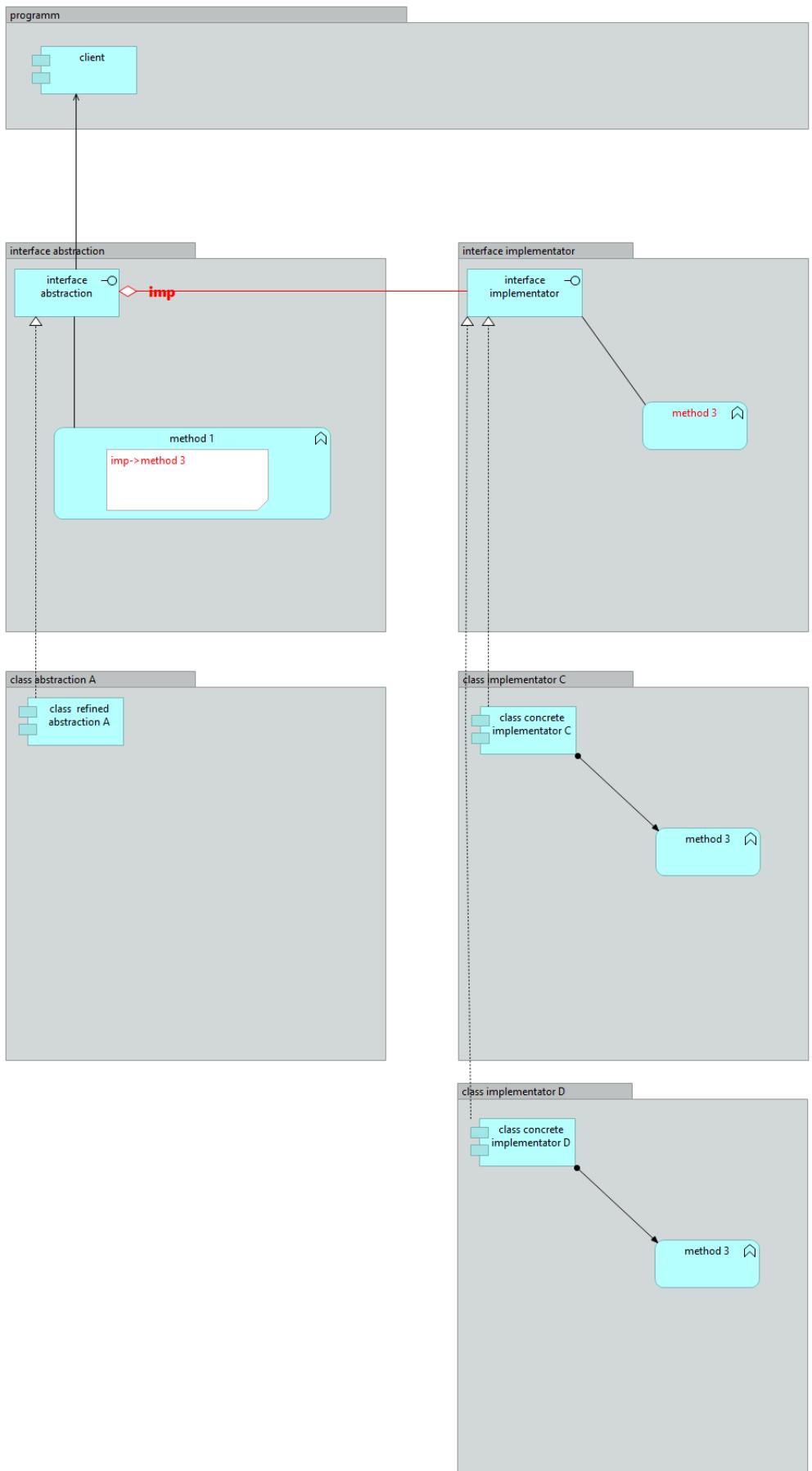


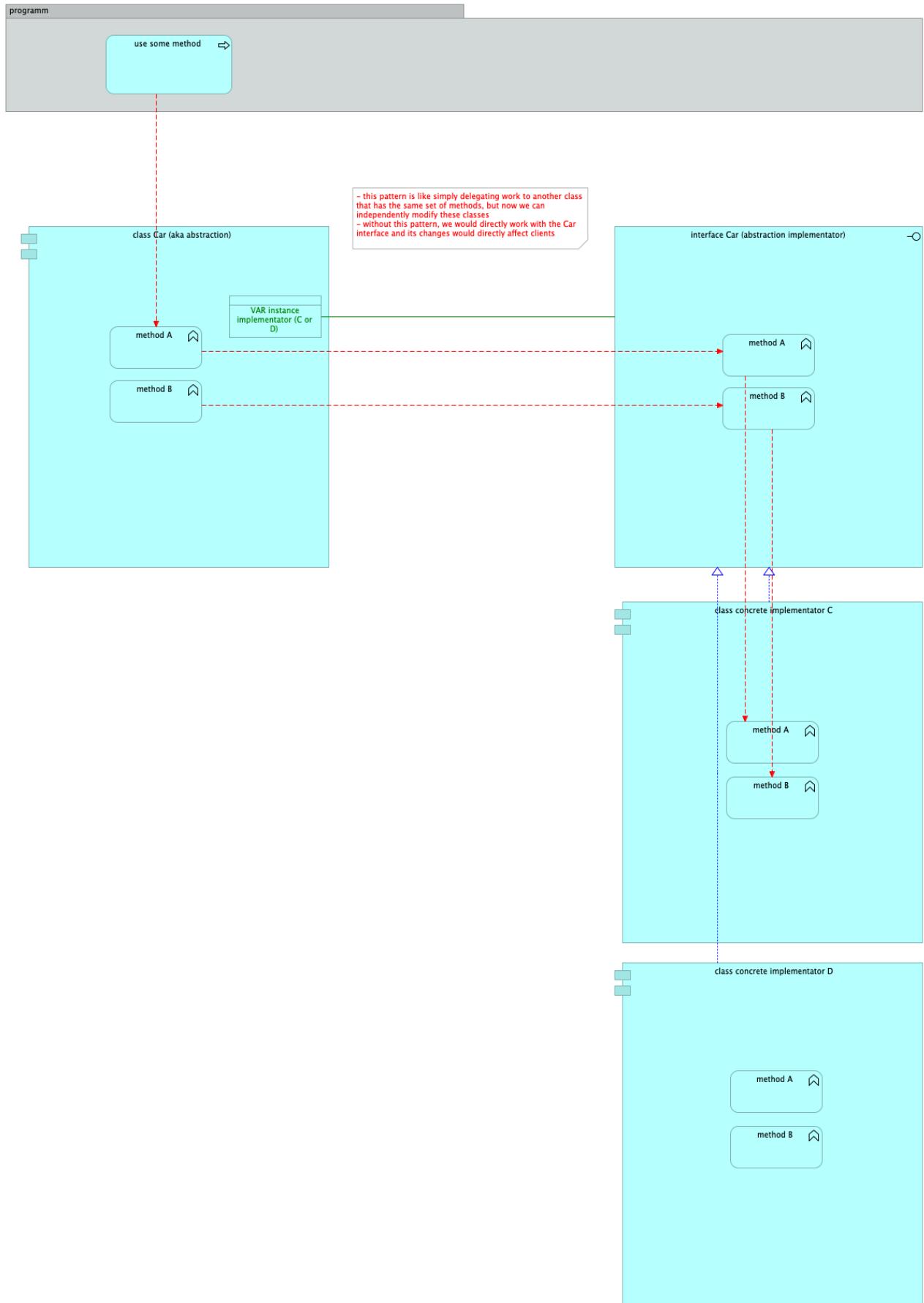


BRIDGE

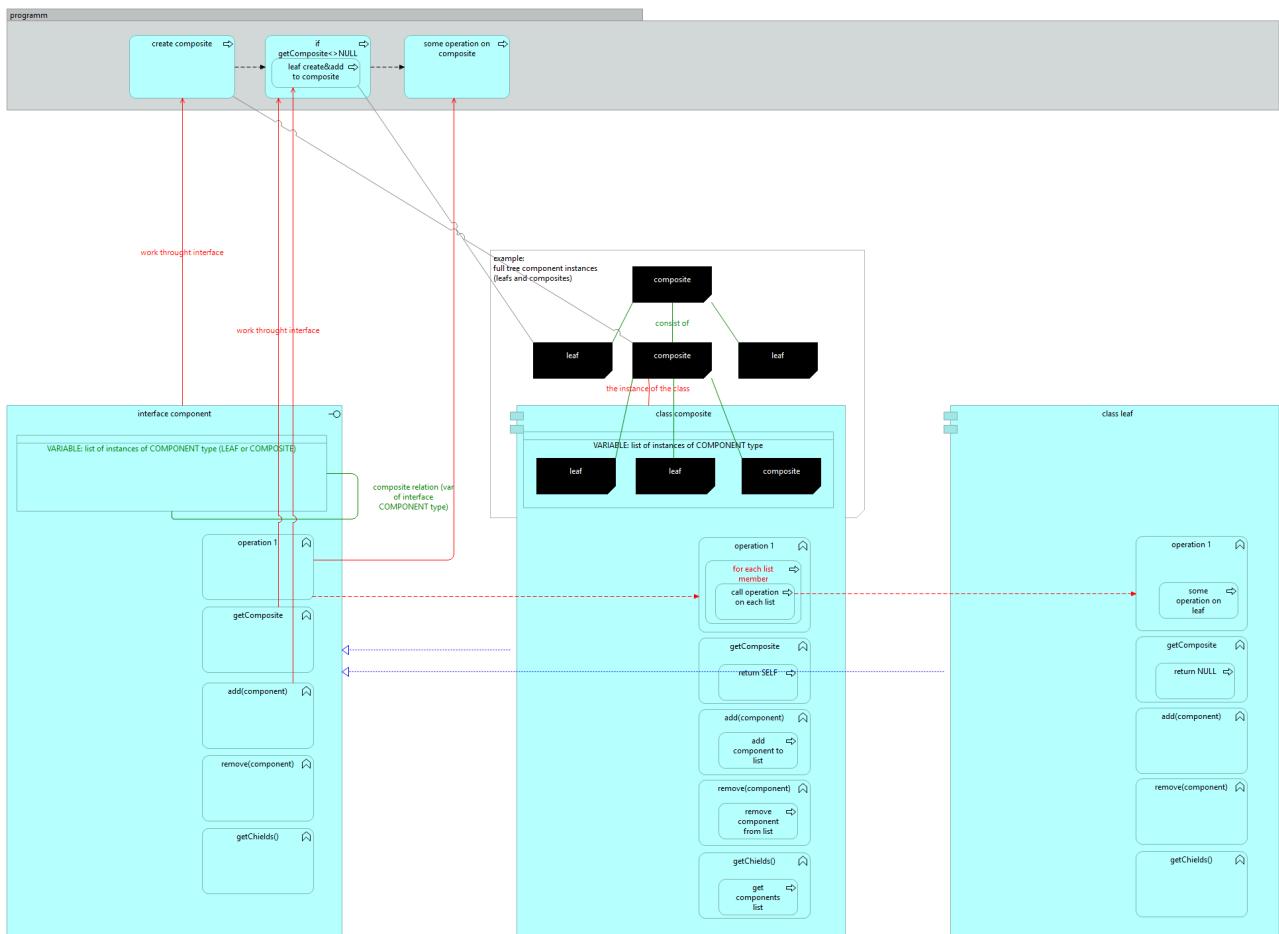


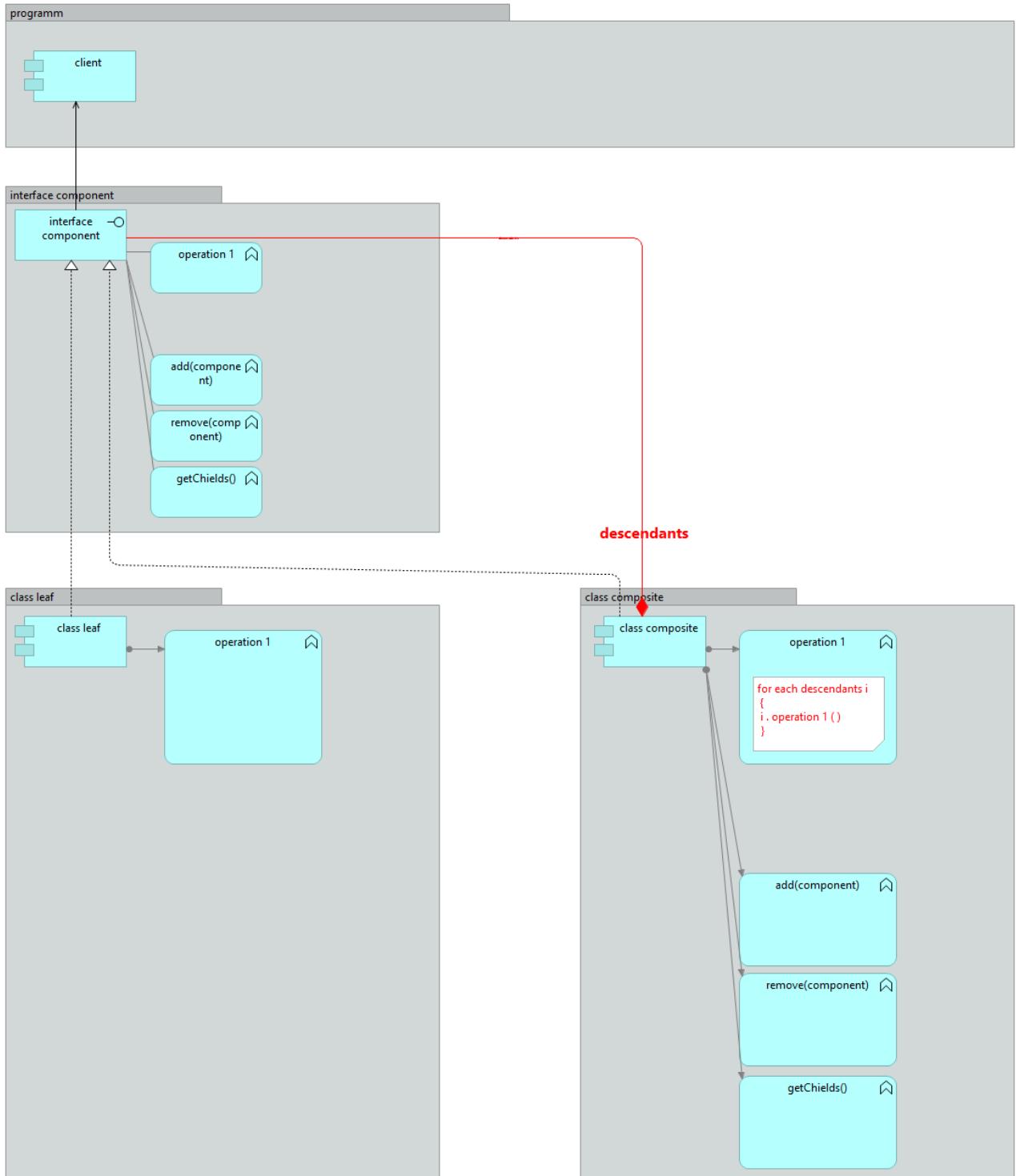




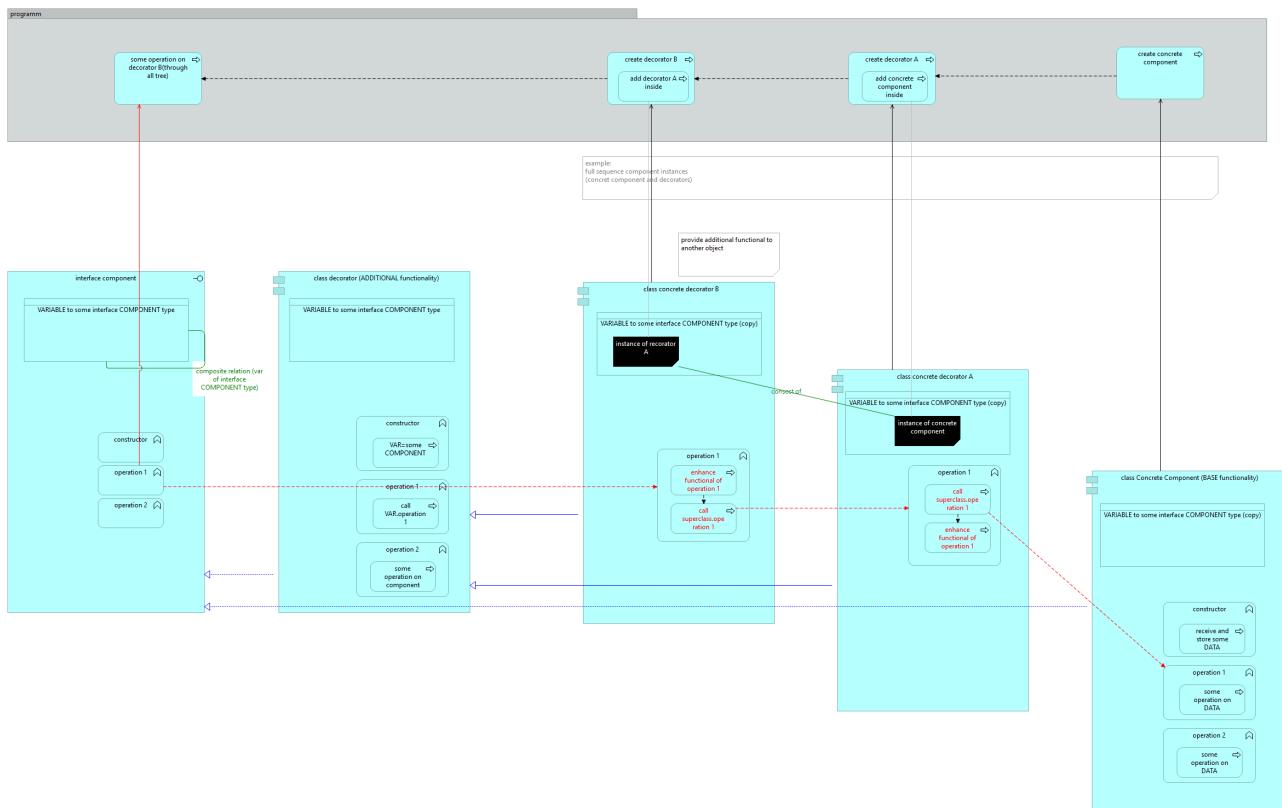


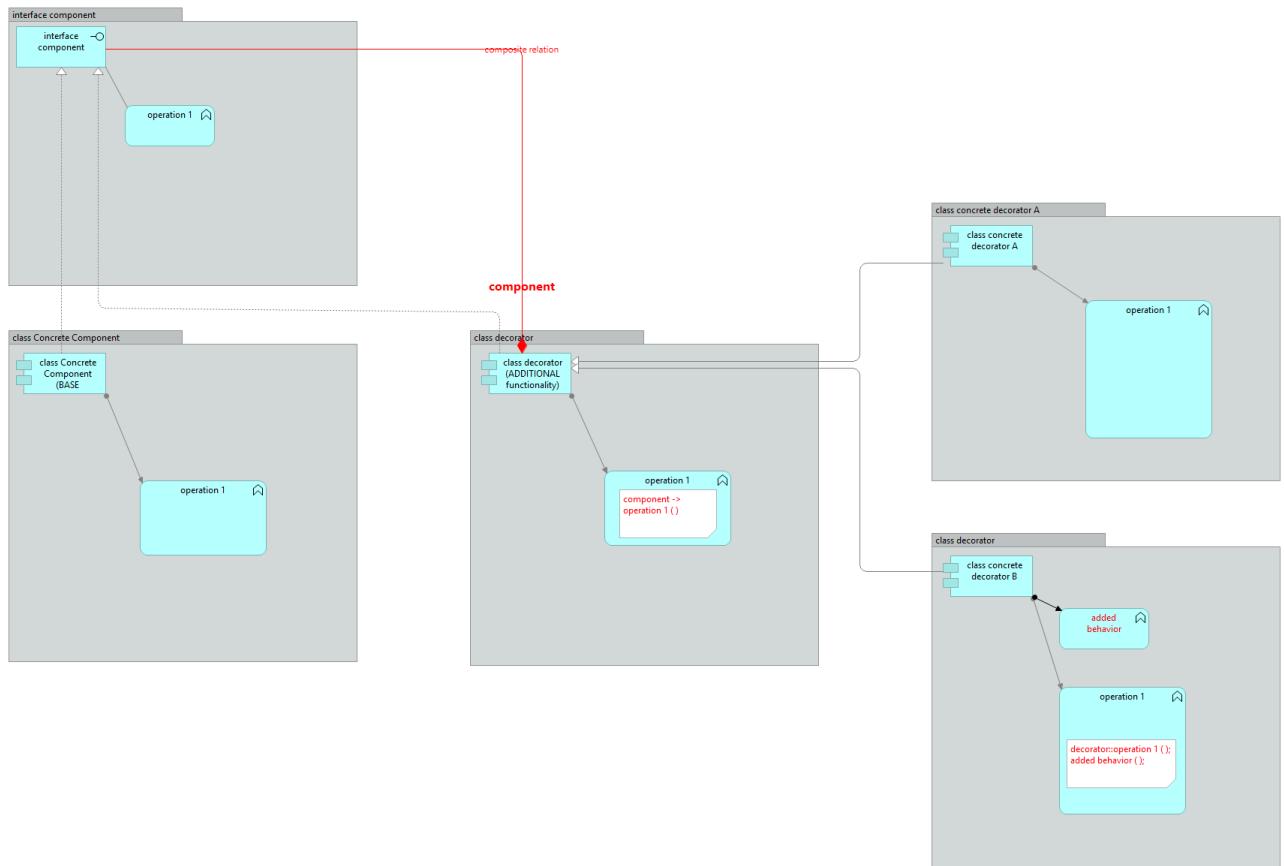
COMPOSITE



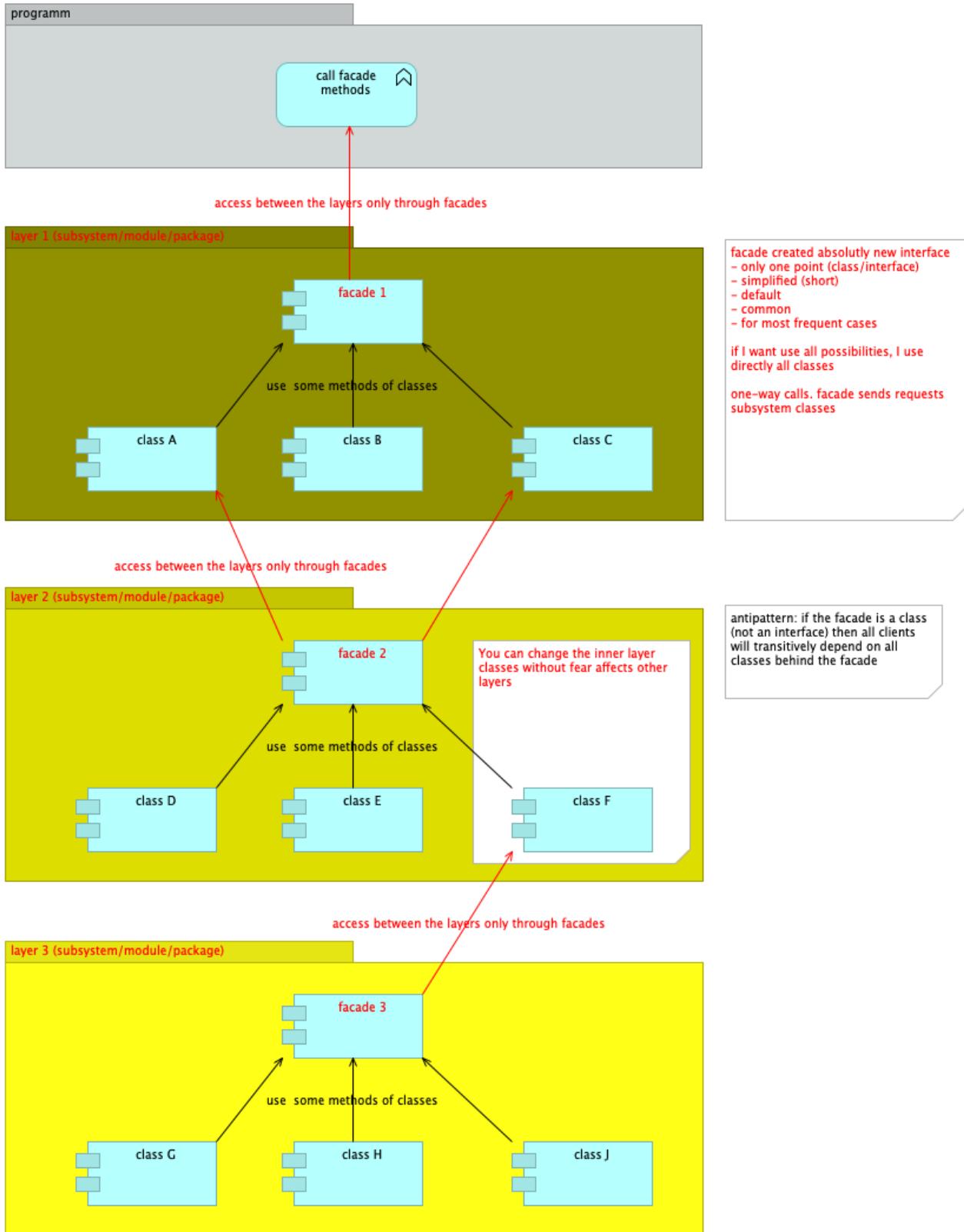


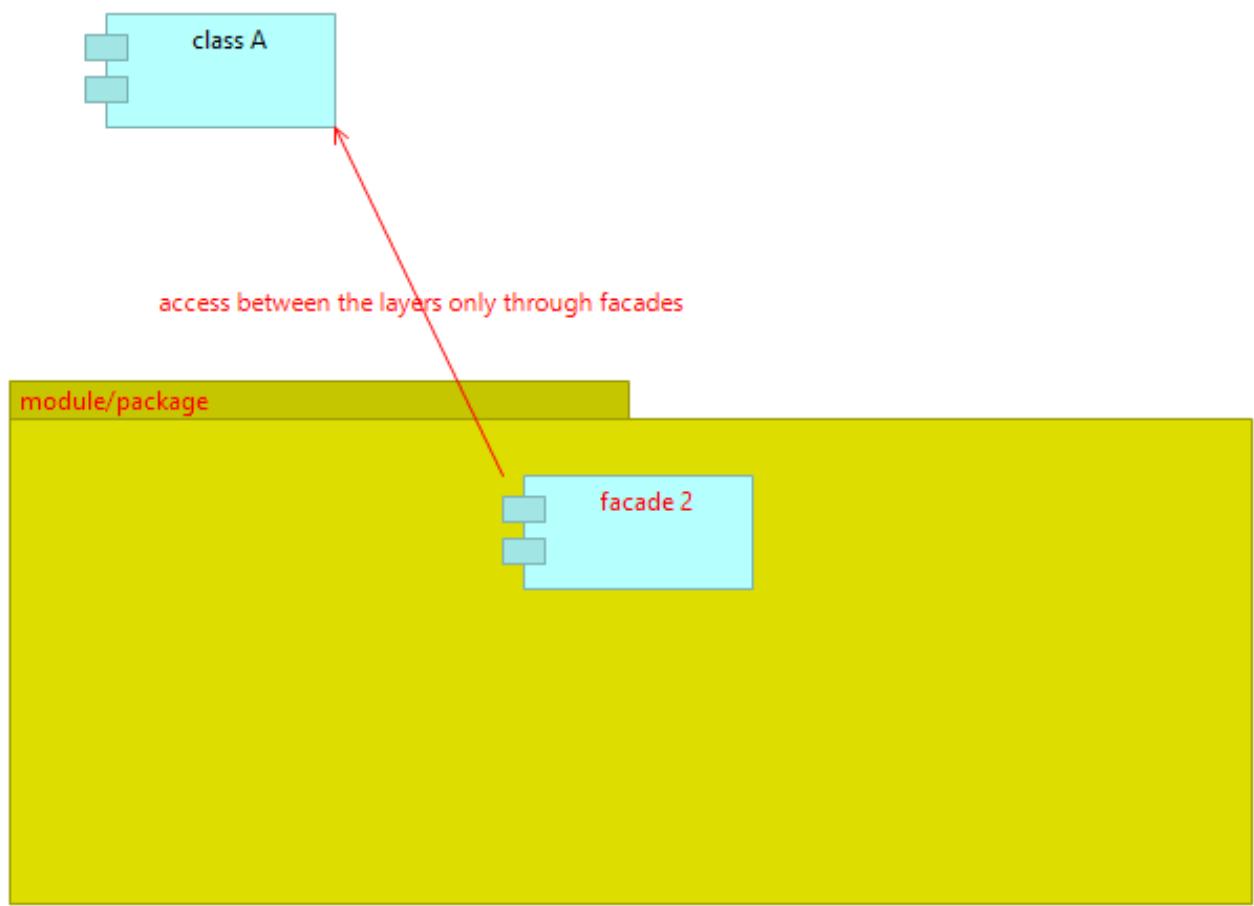
DECORATOR

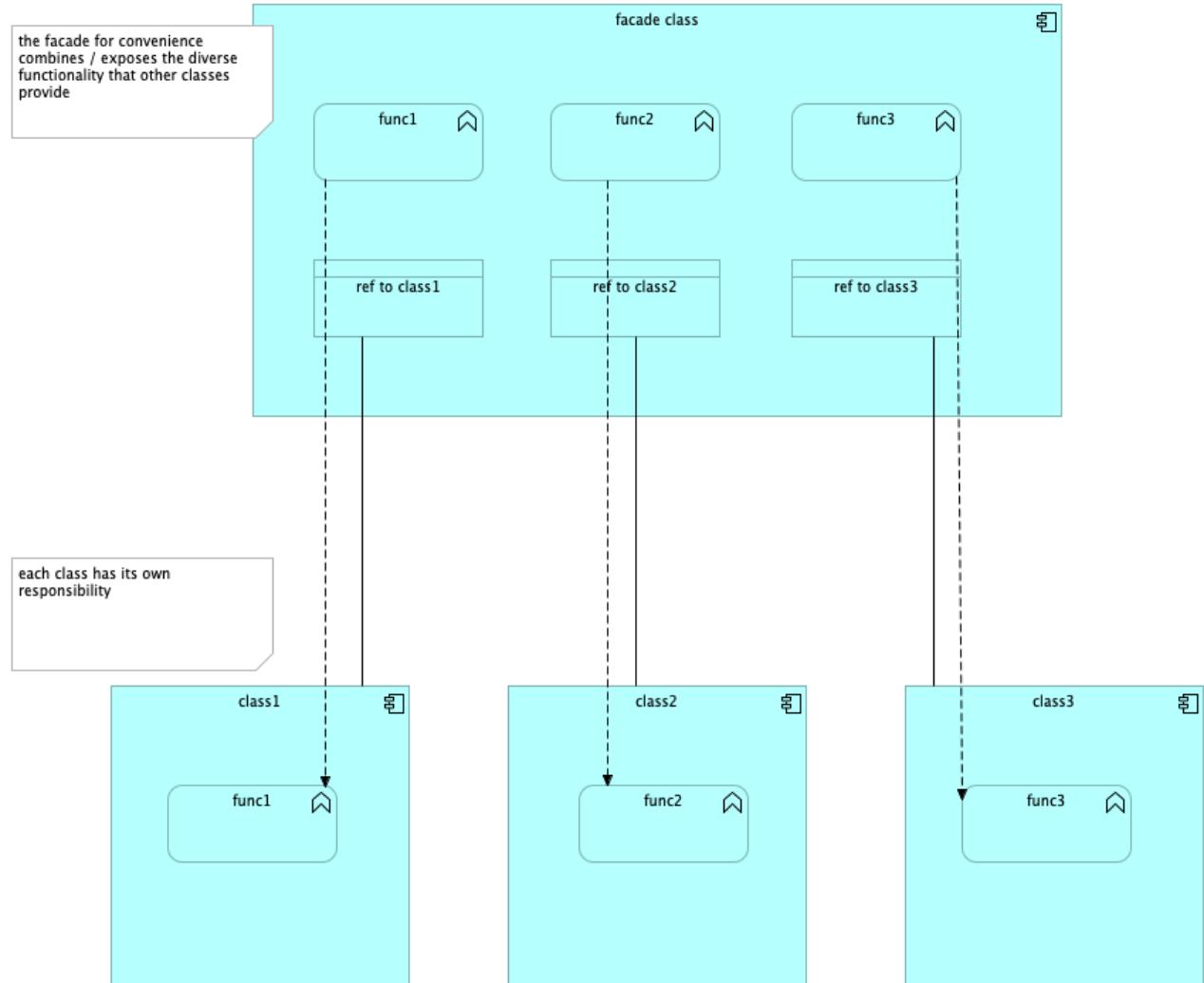




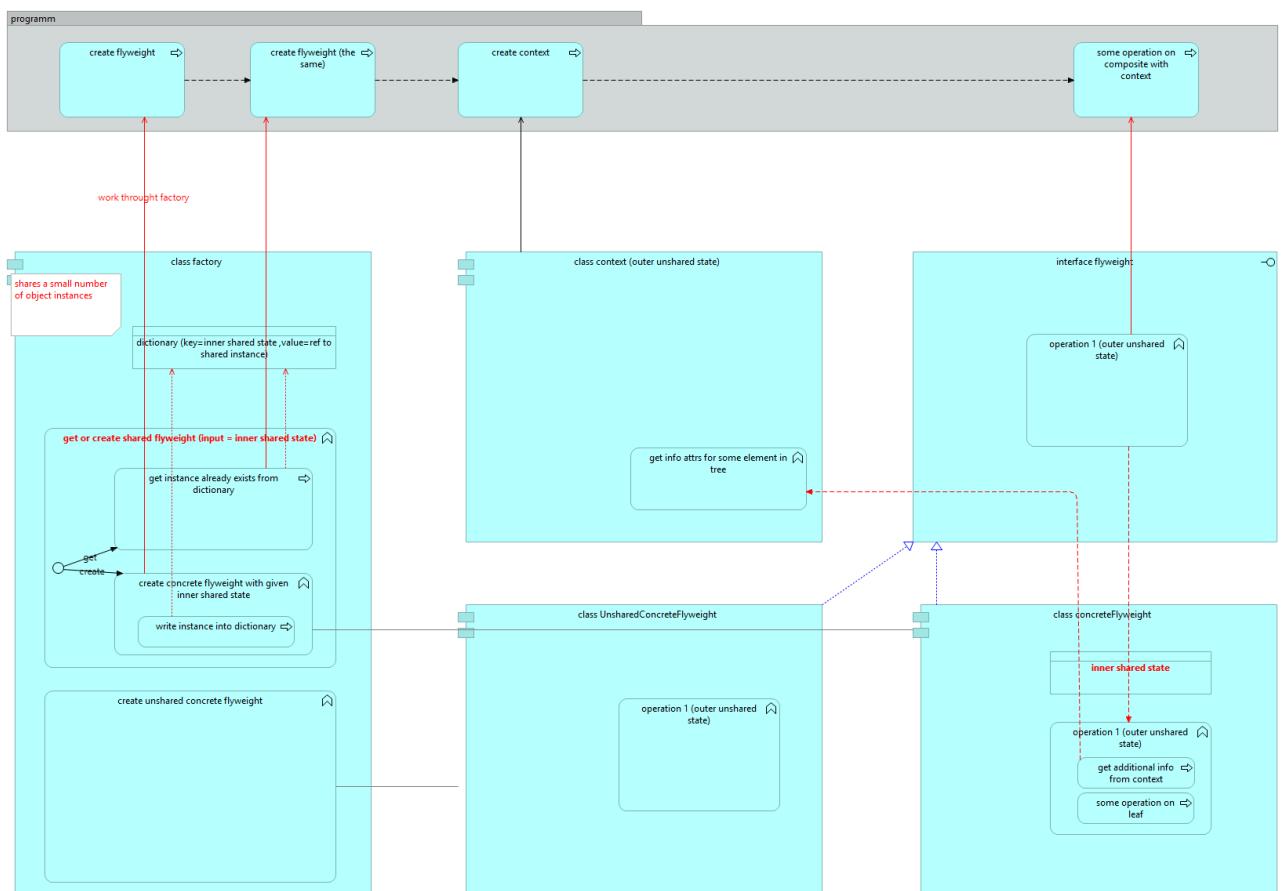
FACADE

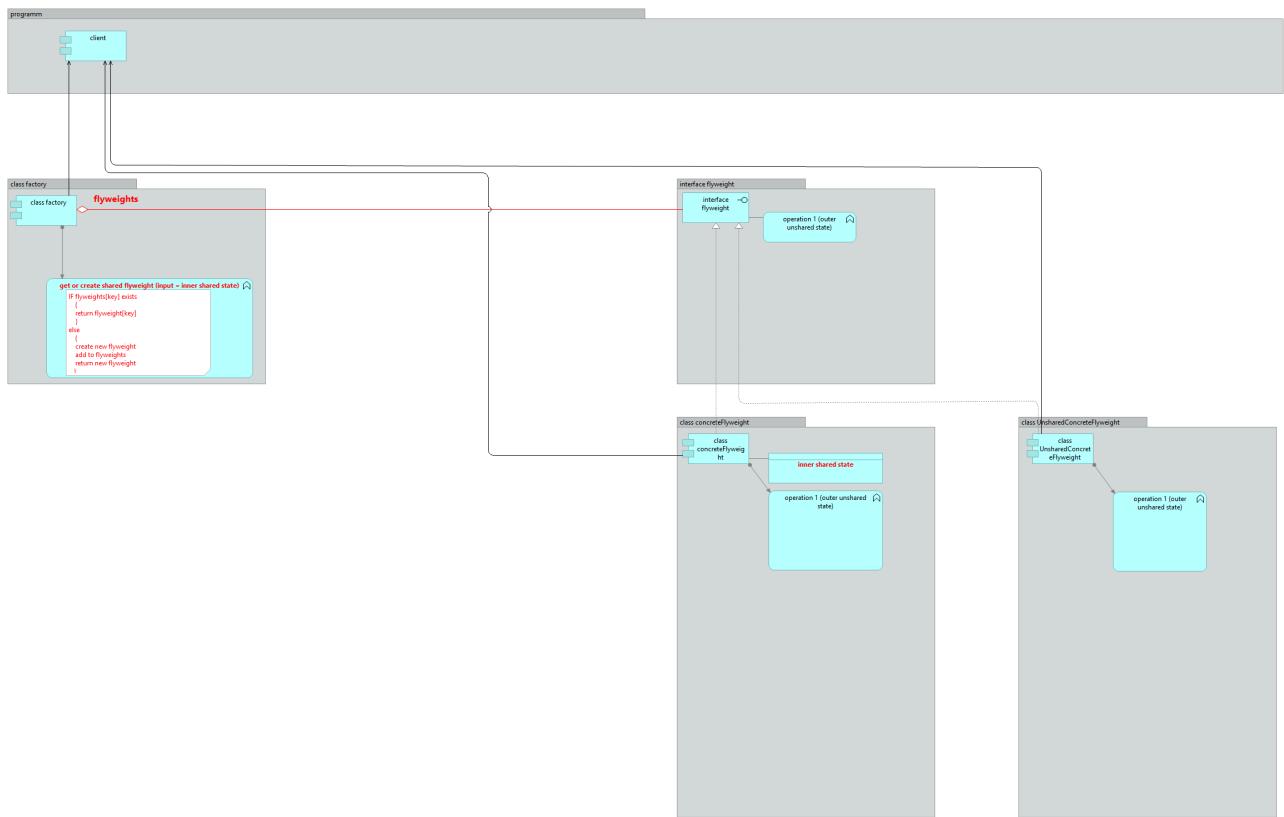




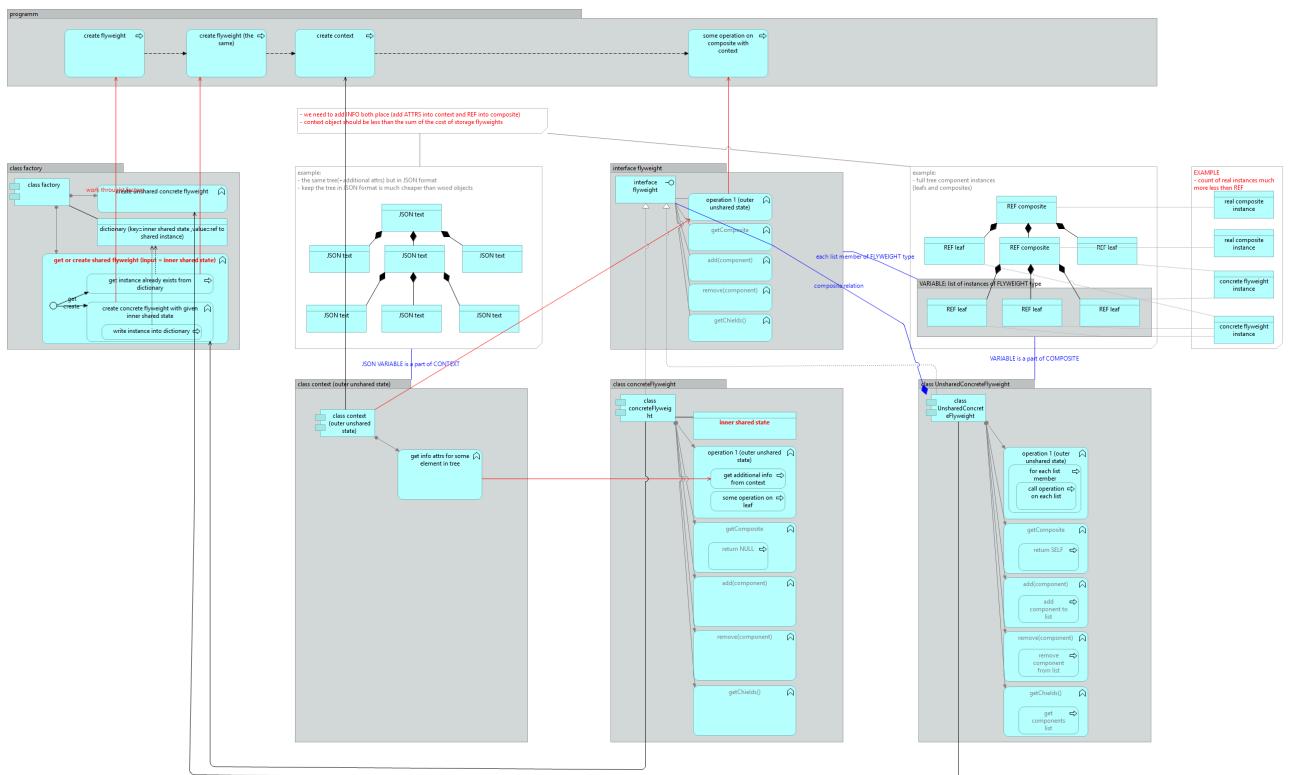


FLYWEIGHT

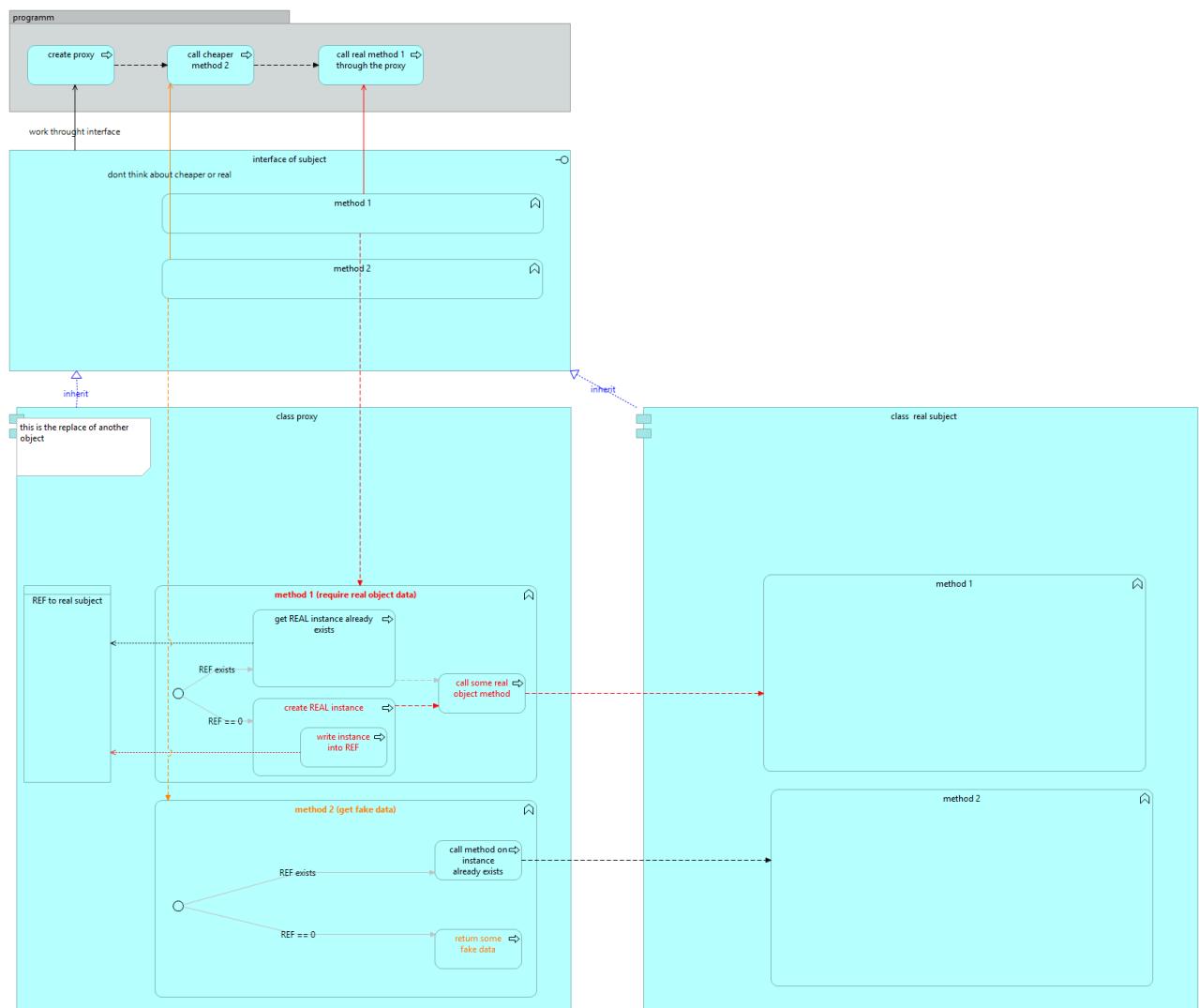


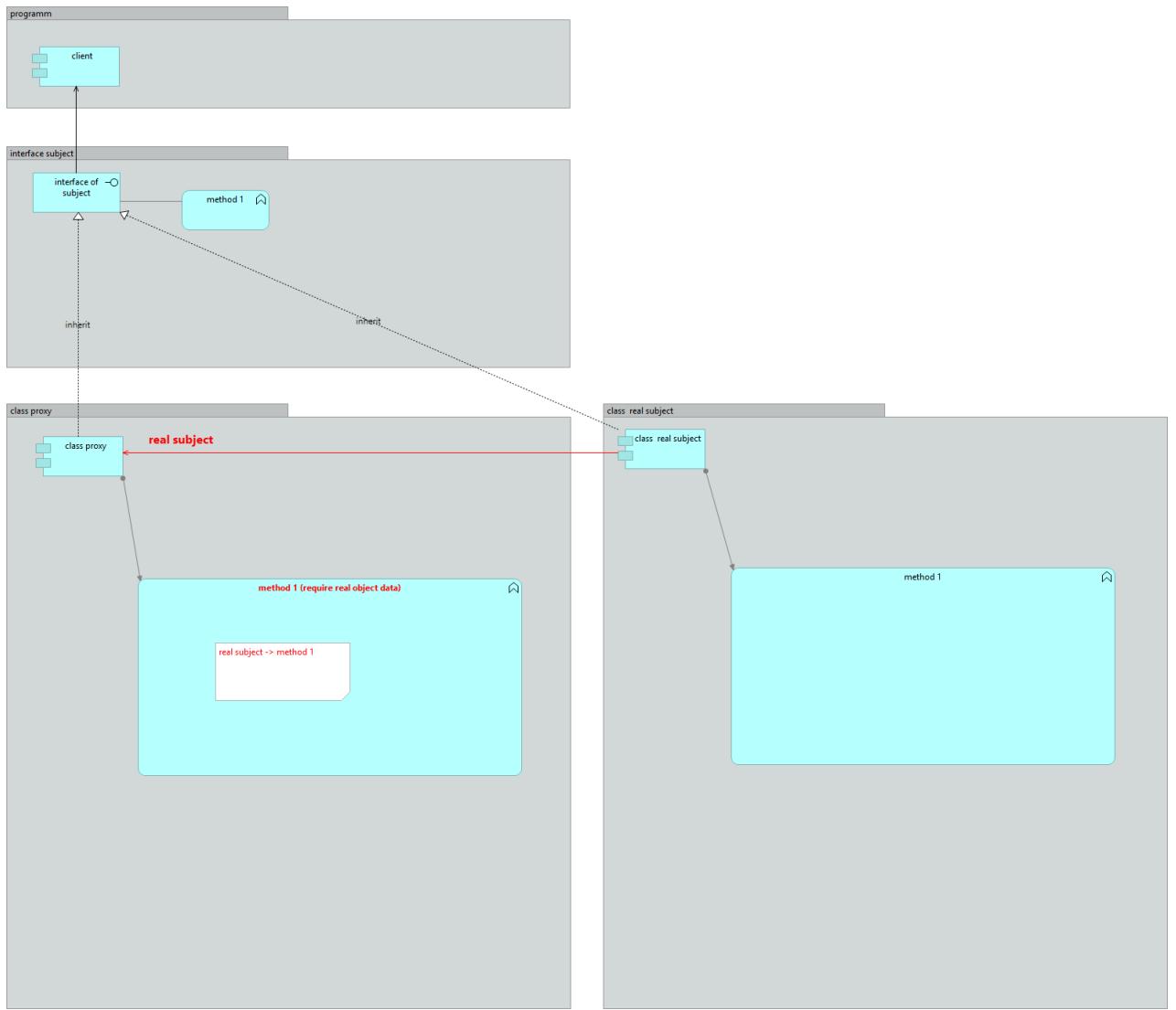


FLYWEIGHT + COMPOSITE



PROXY





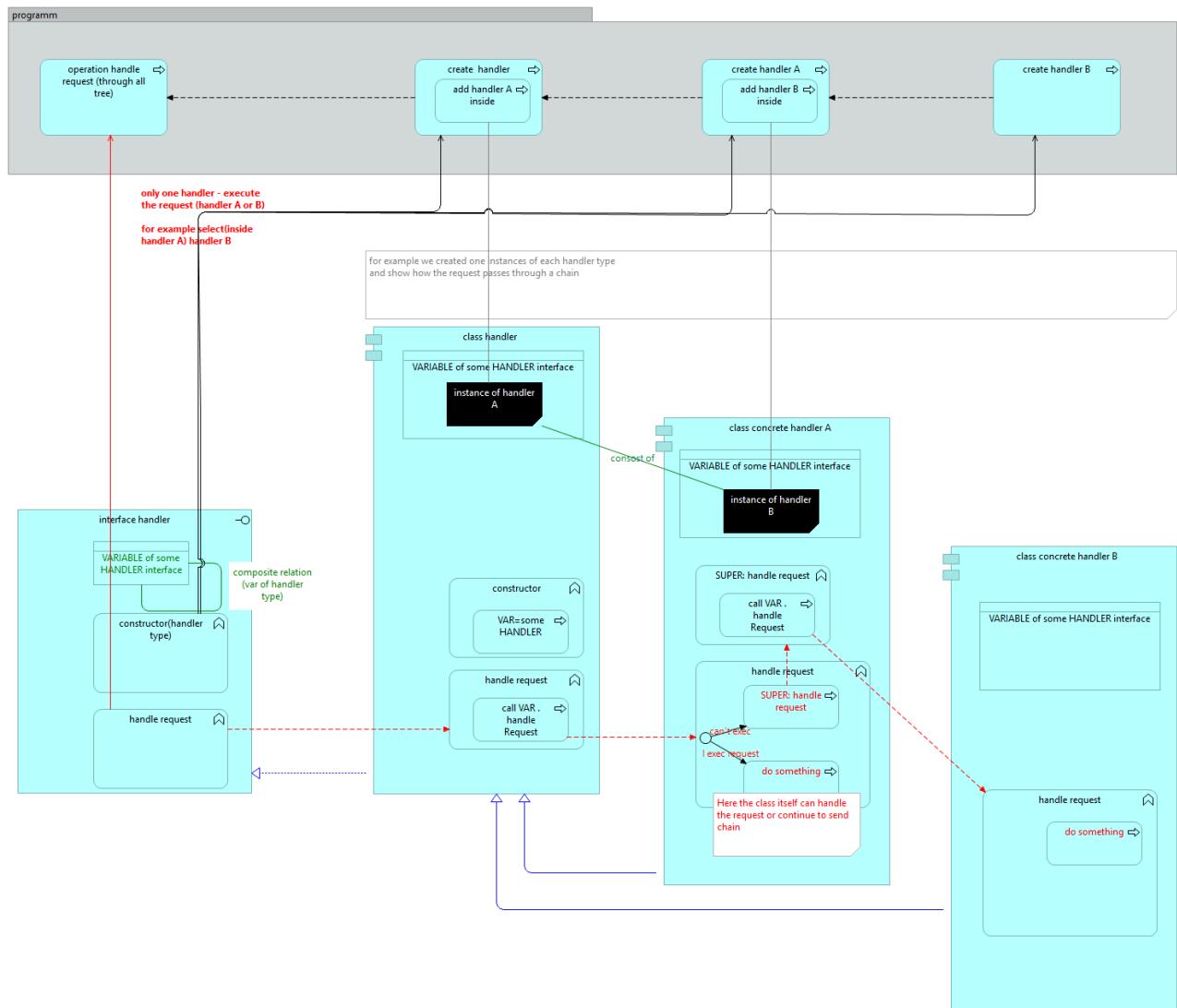
BEHAVIORAL PATTERNS

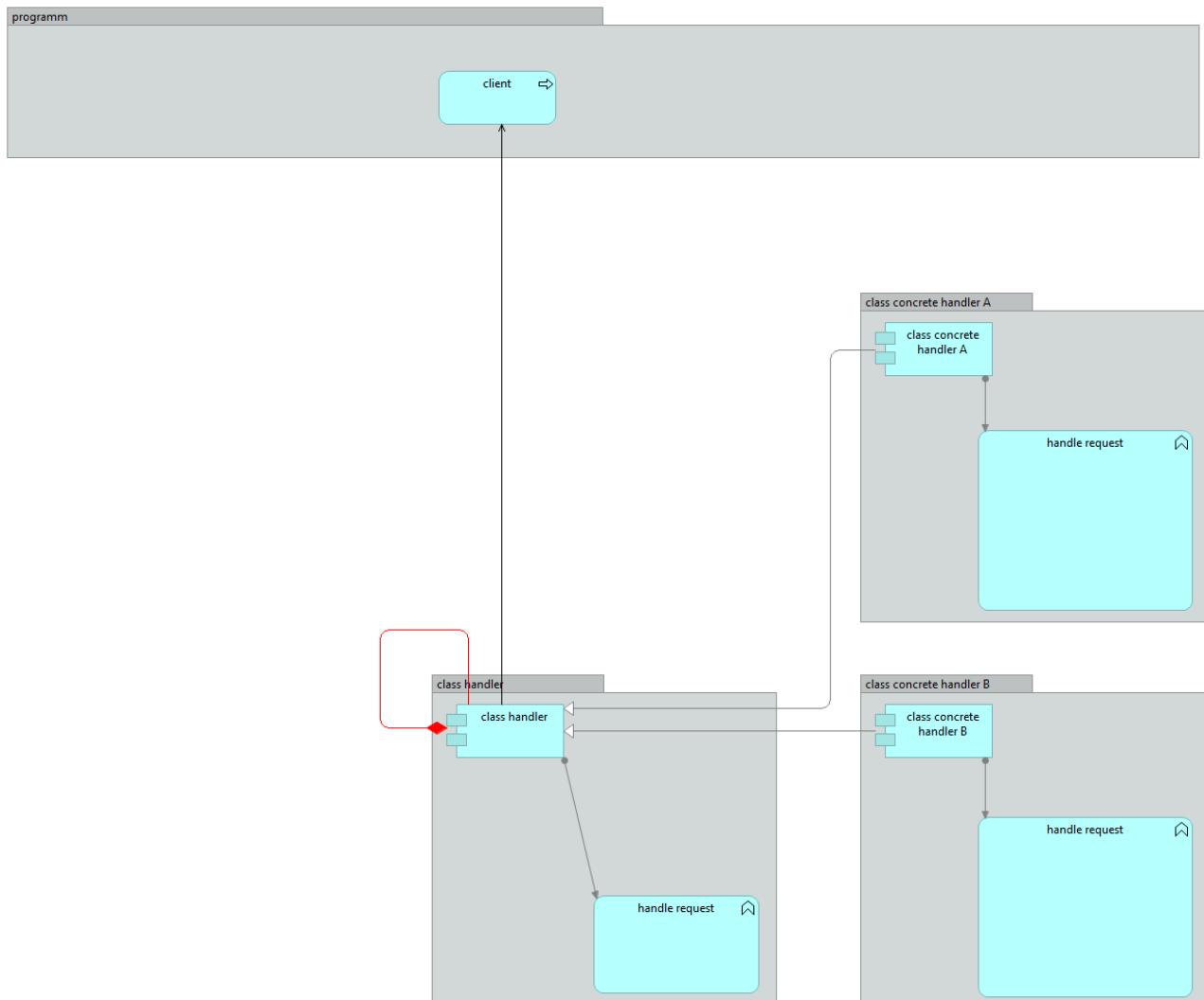
DESIGN PATTERNS

GoF

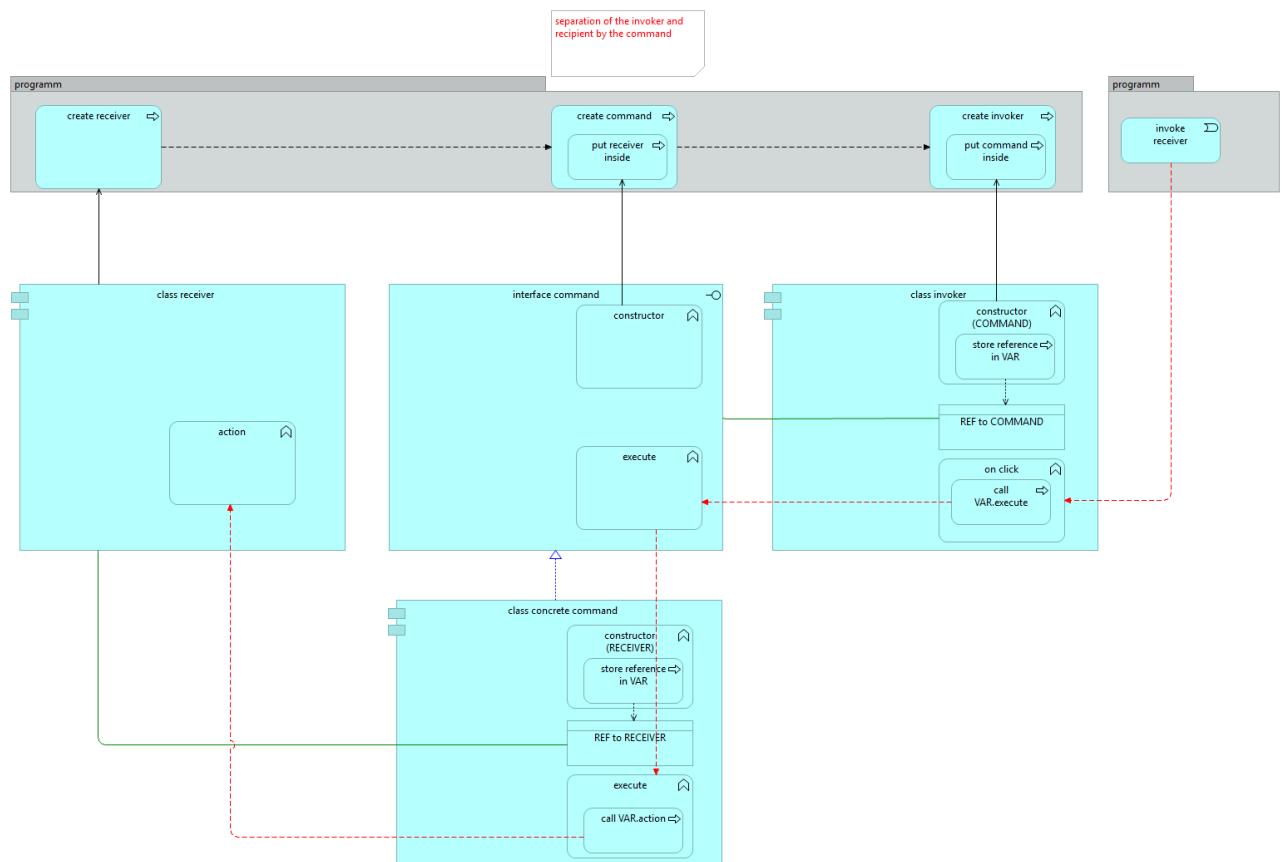


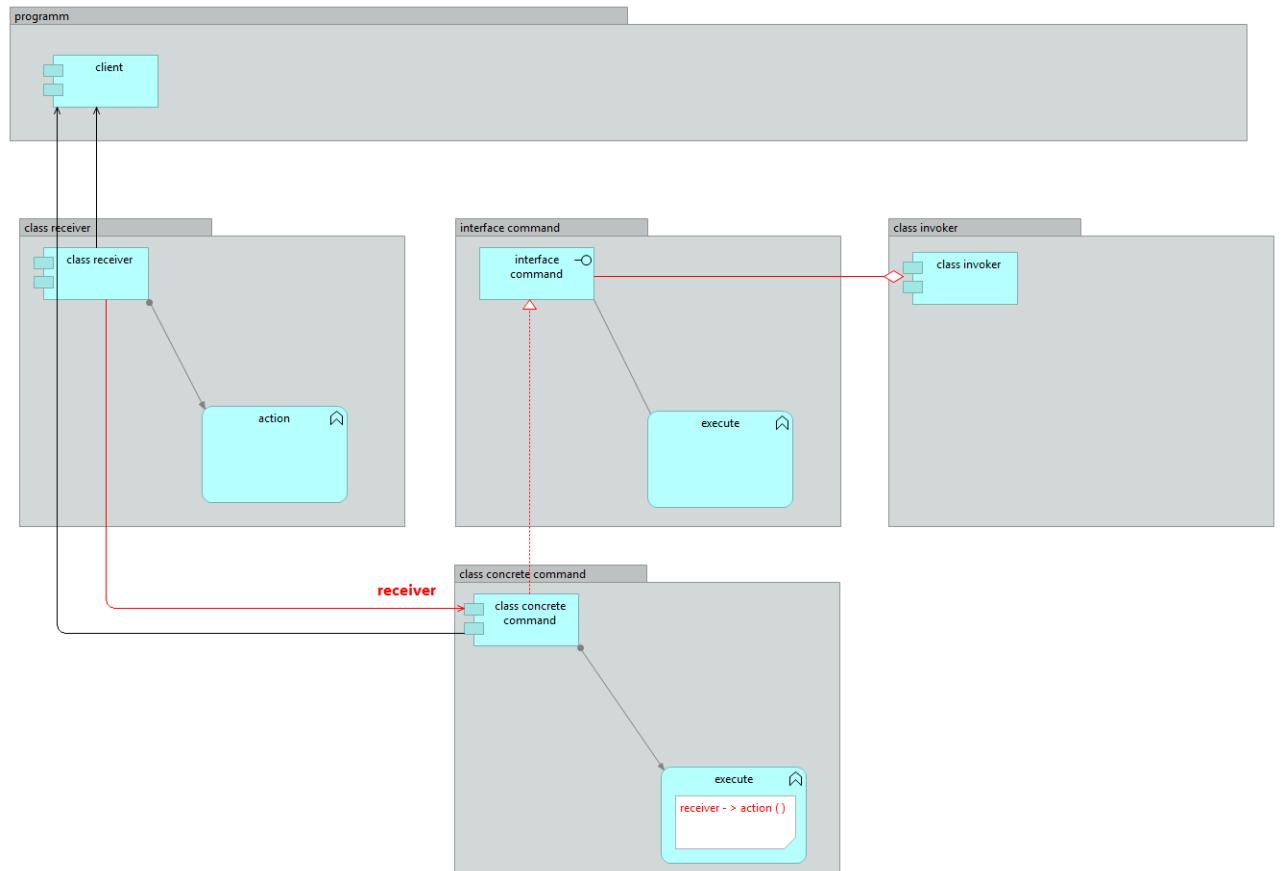
CHAIN OF RESPONSIBILITY



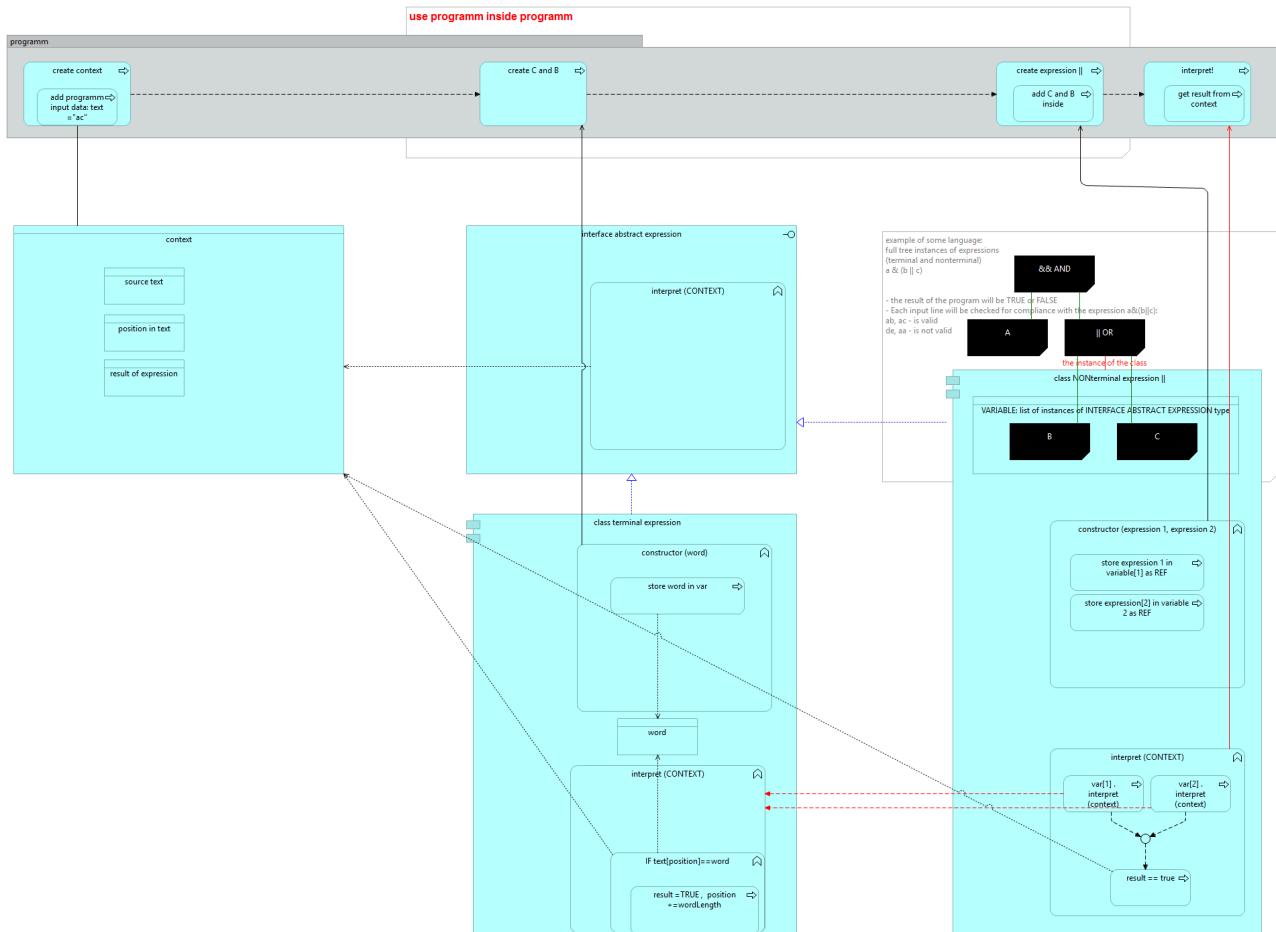


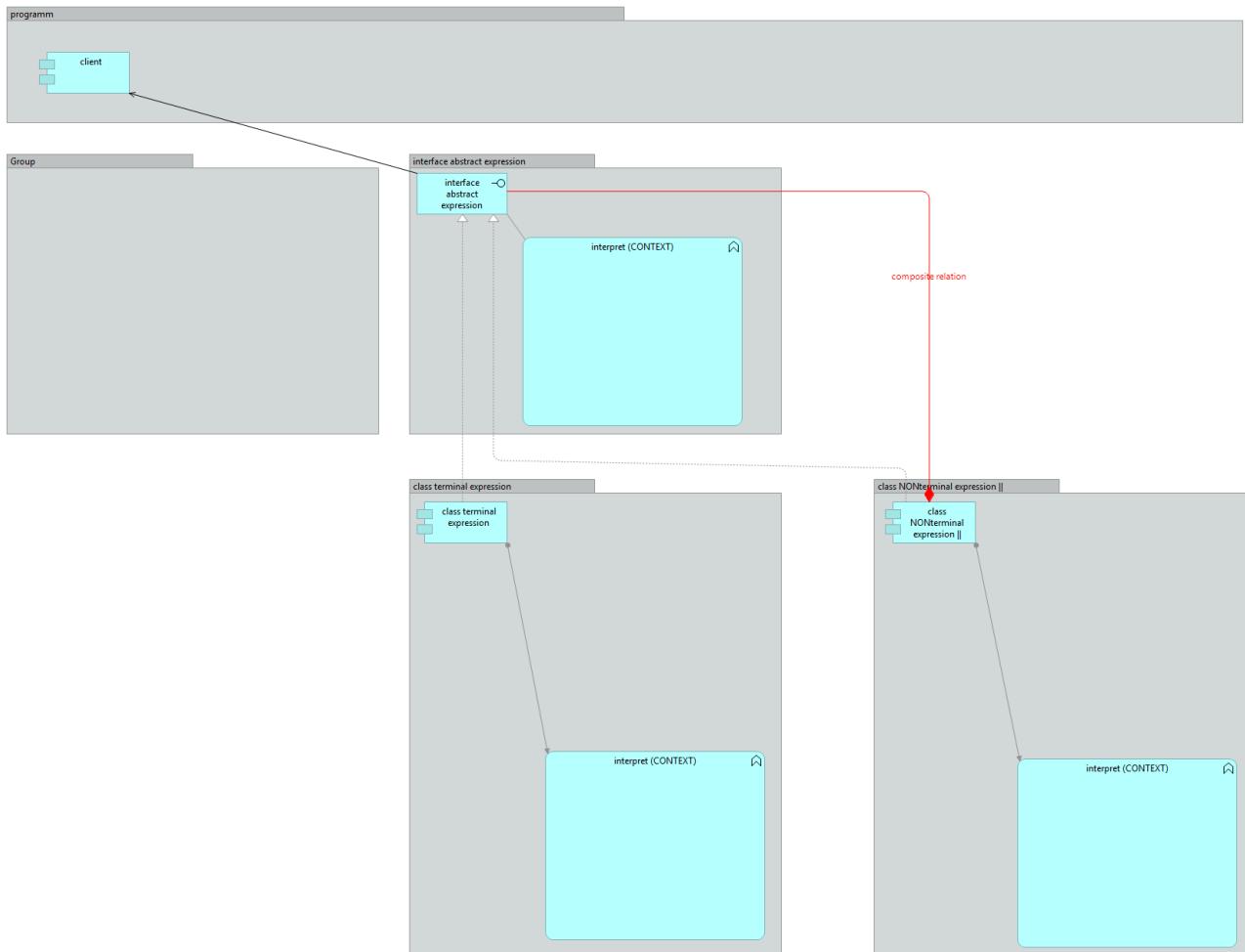
COMMAND



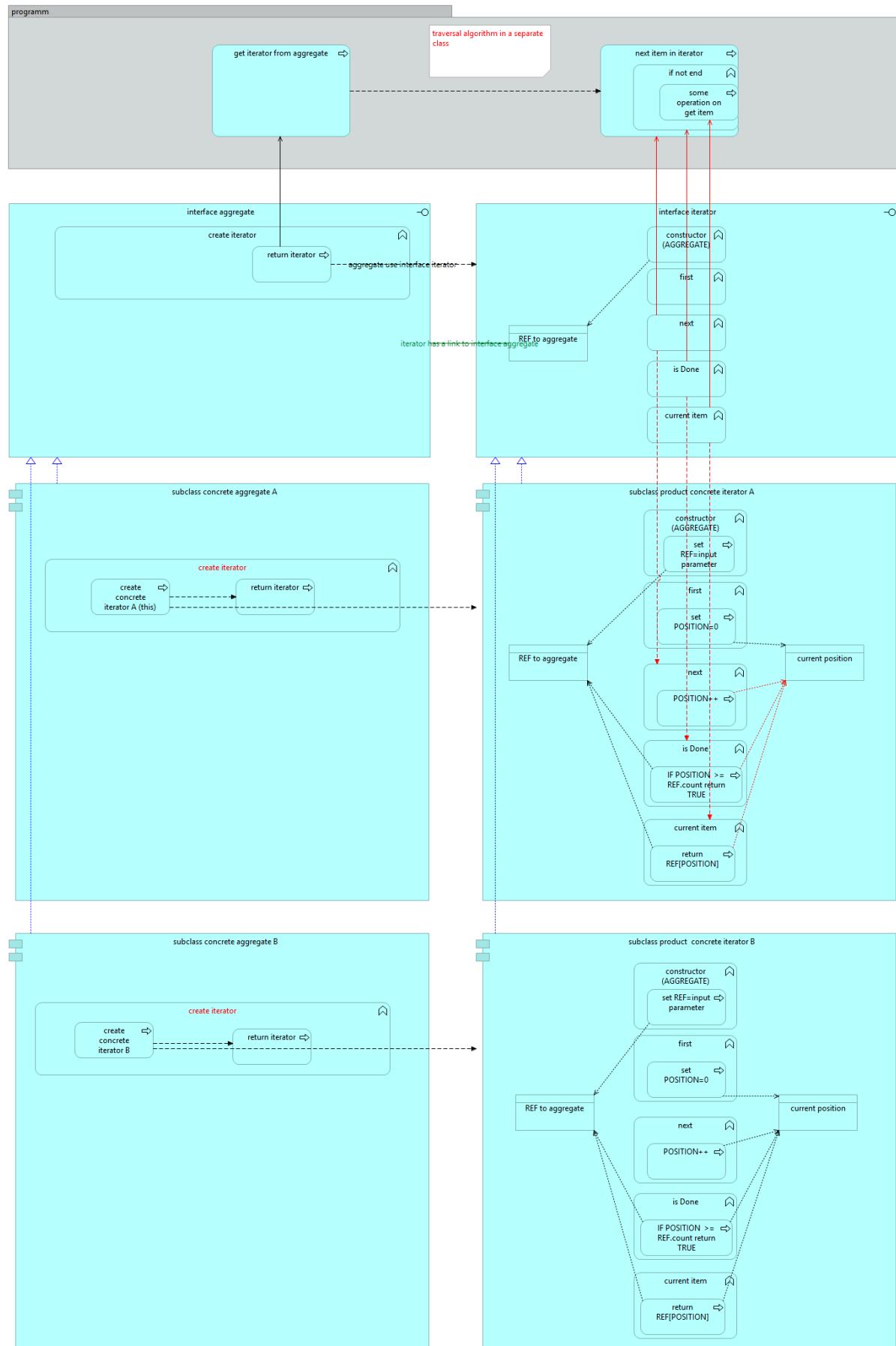


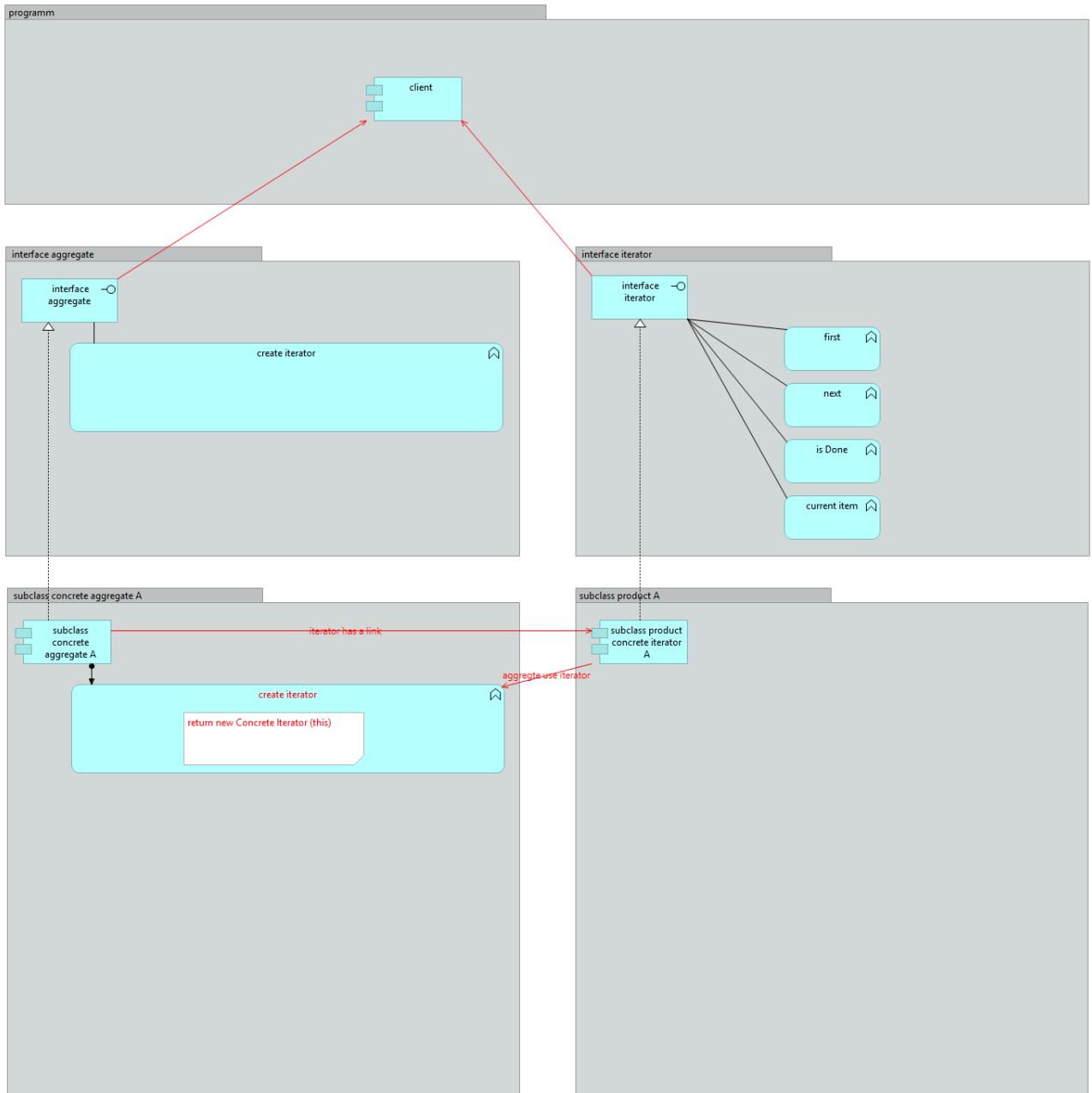
INTERPRETER



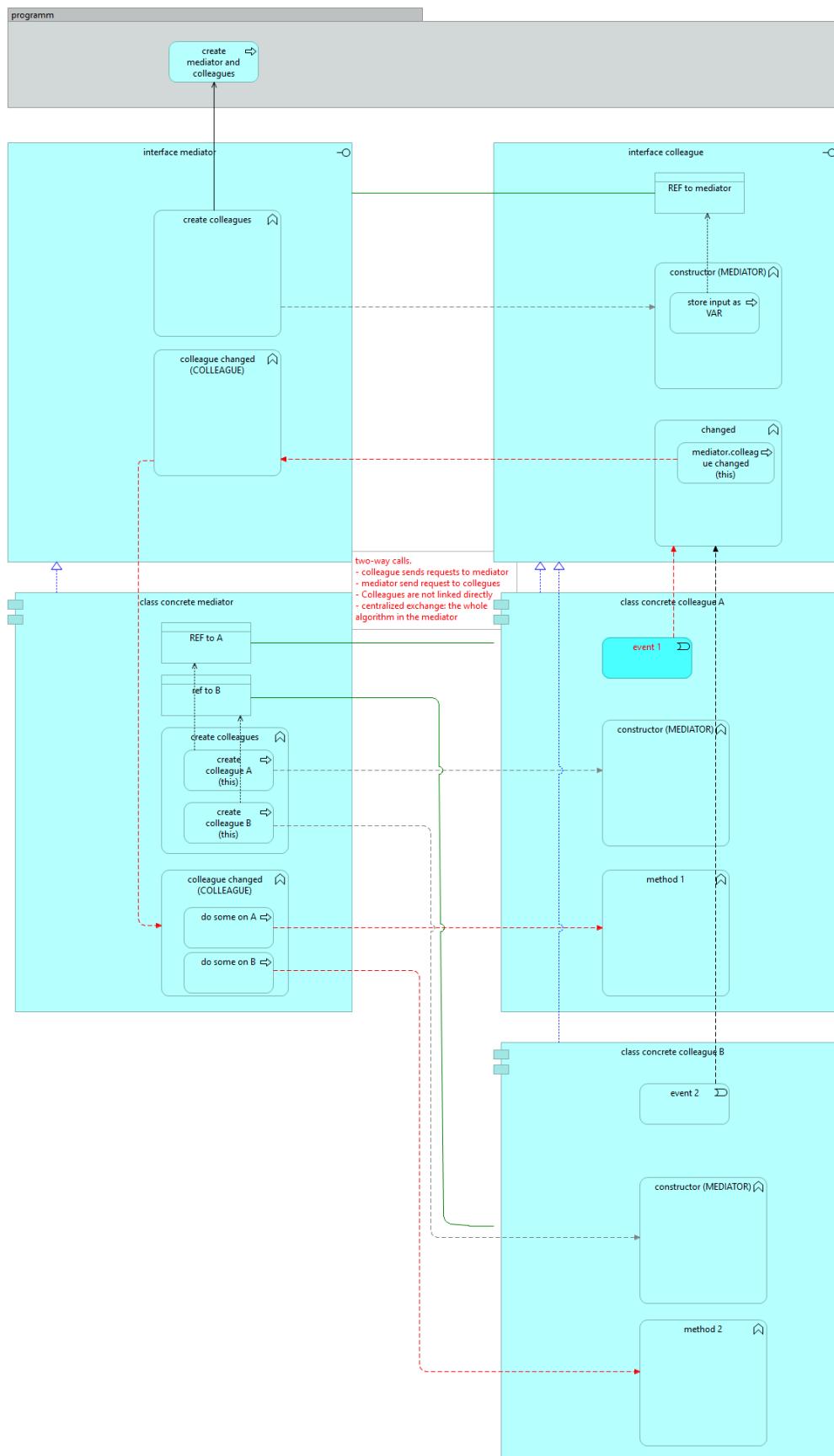


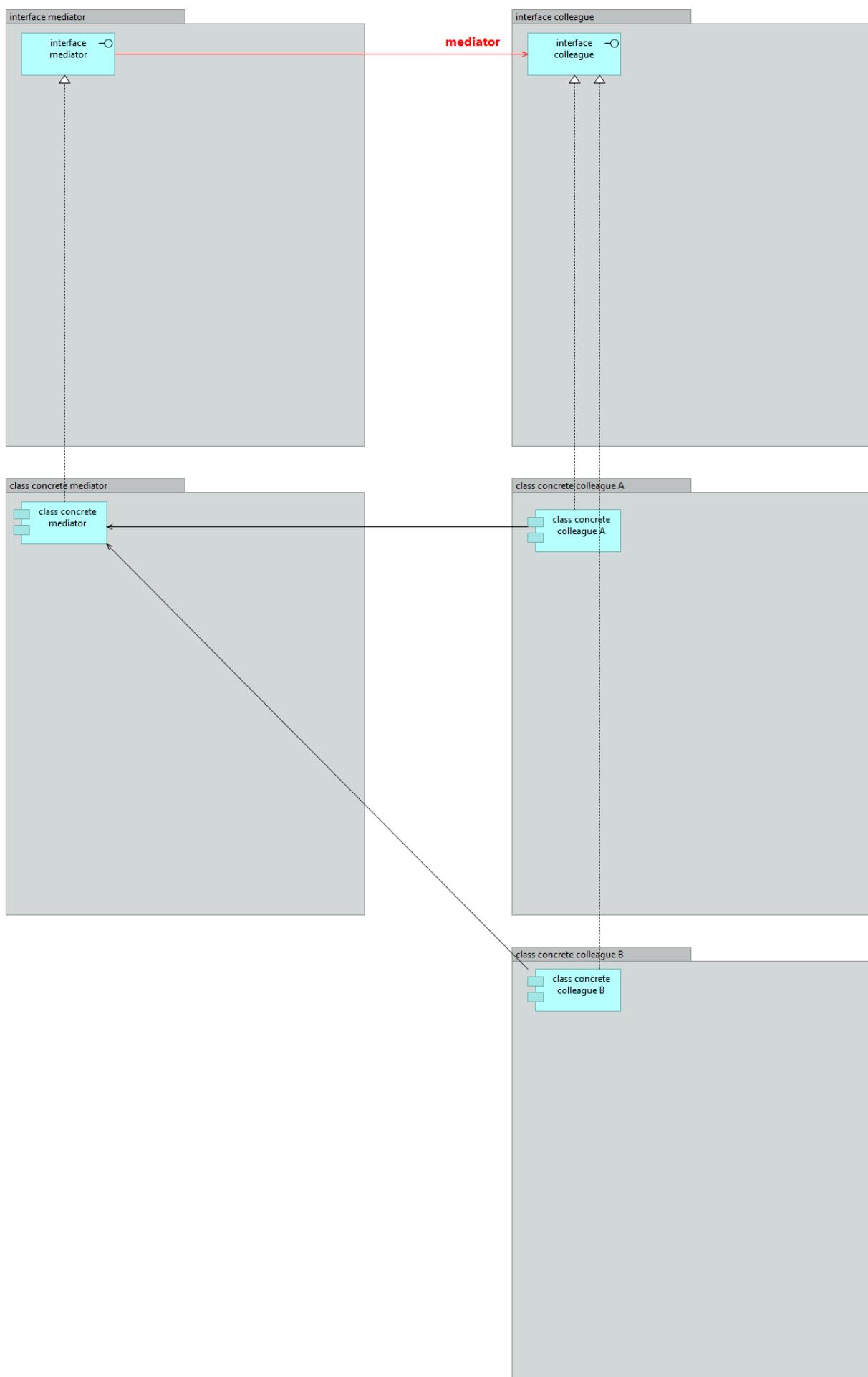
ITERATOR



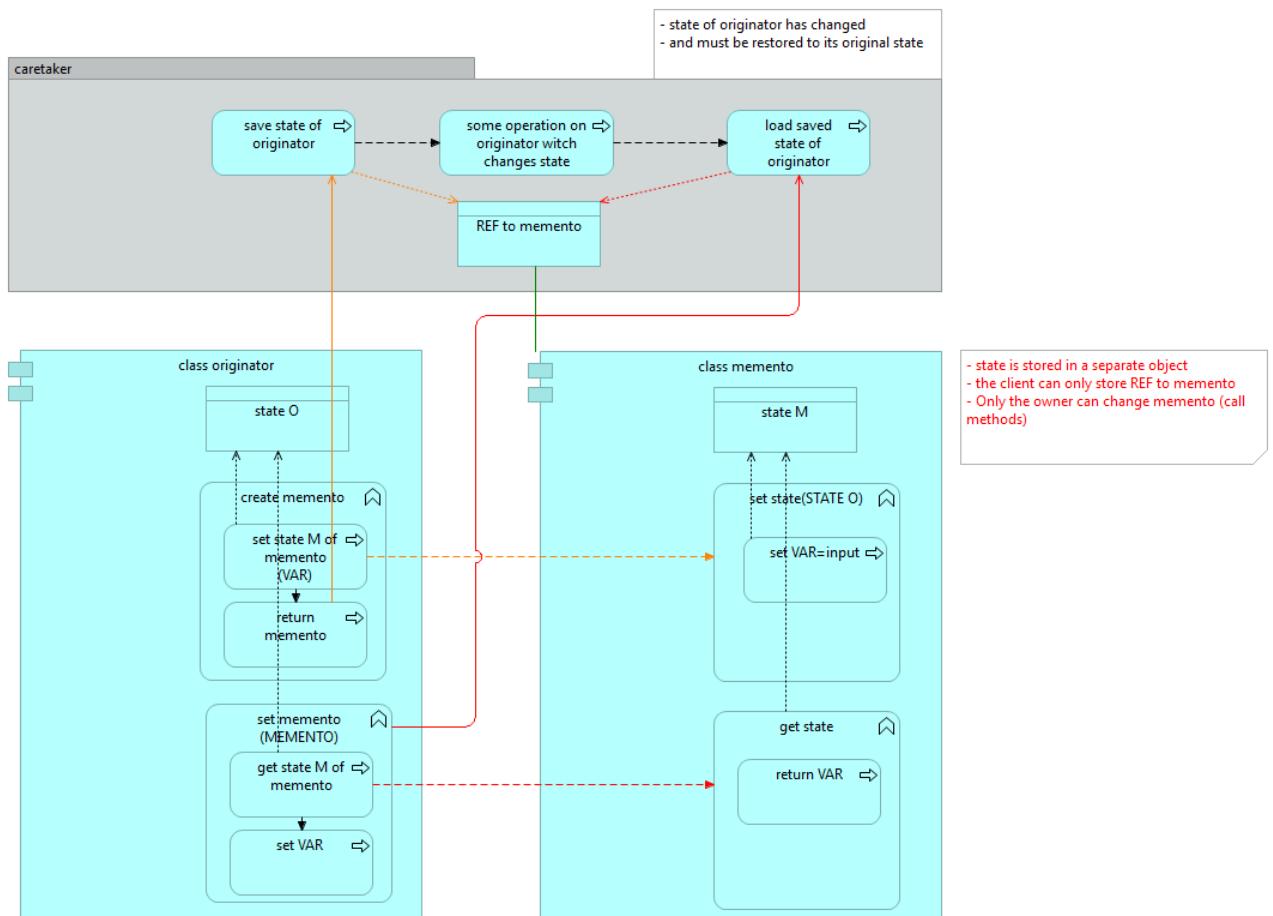


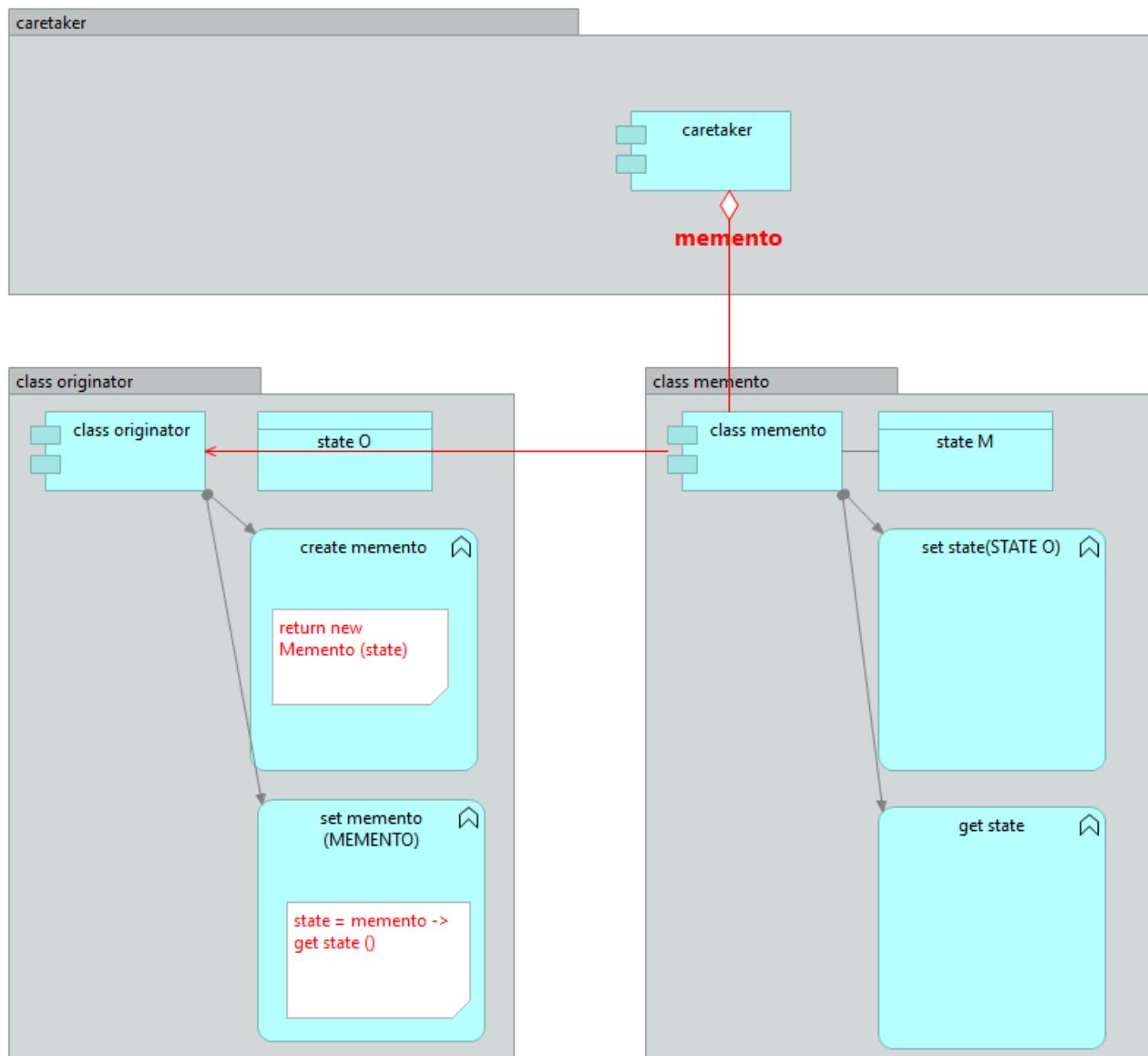
MEDIATOR



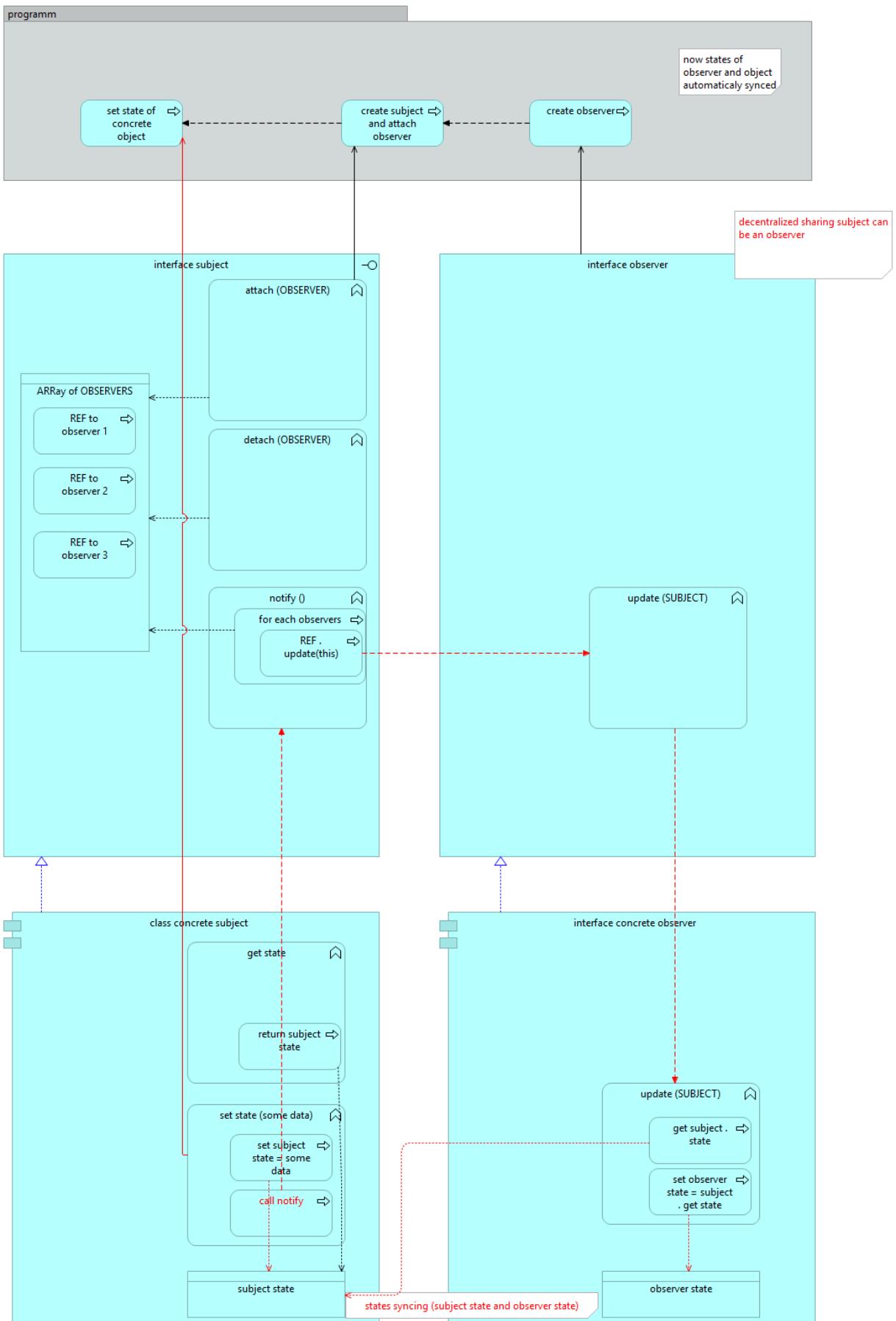


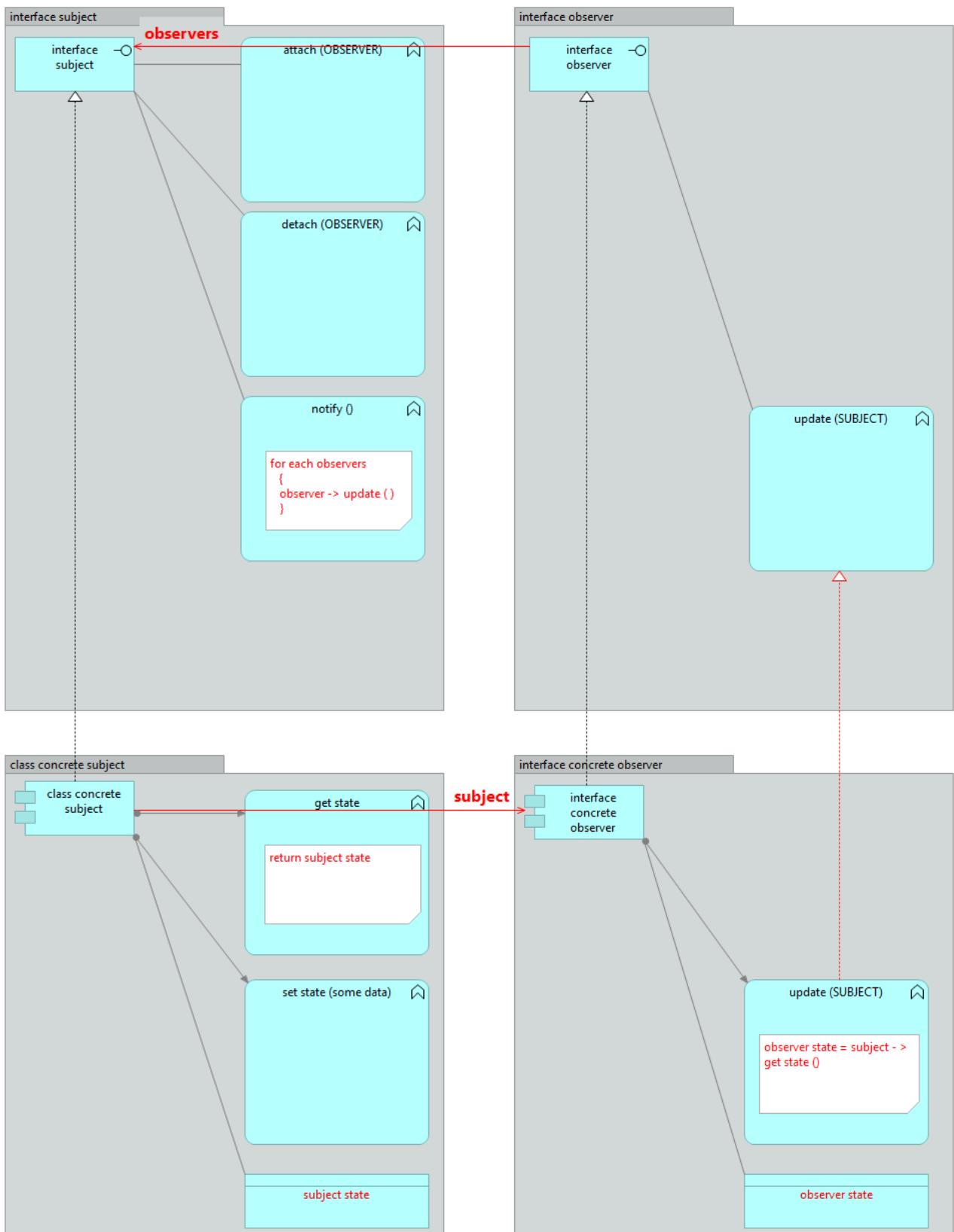
MEMENTO



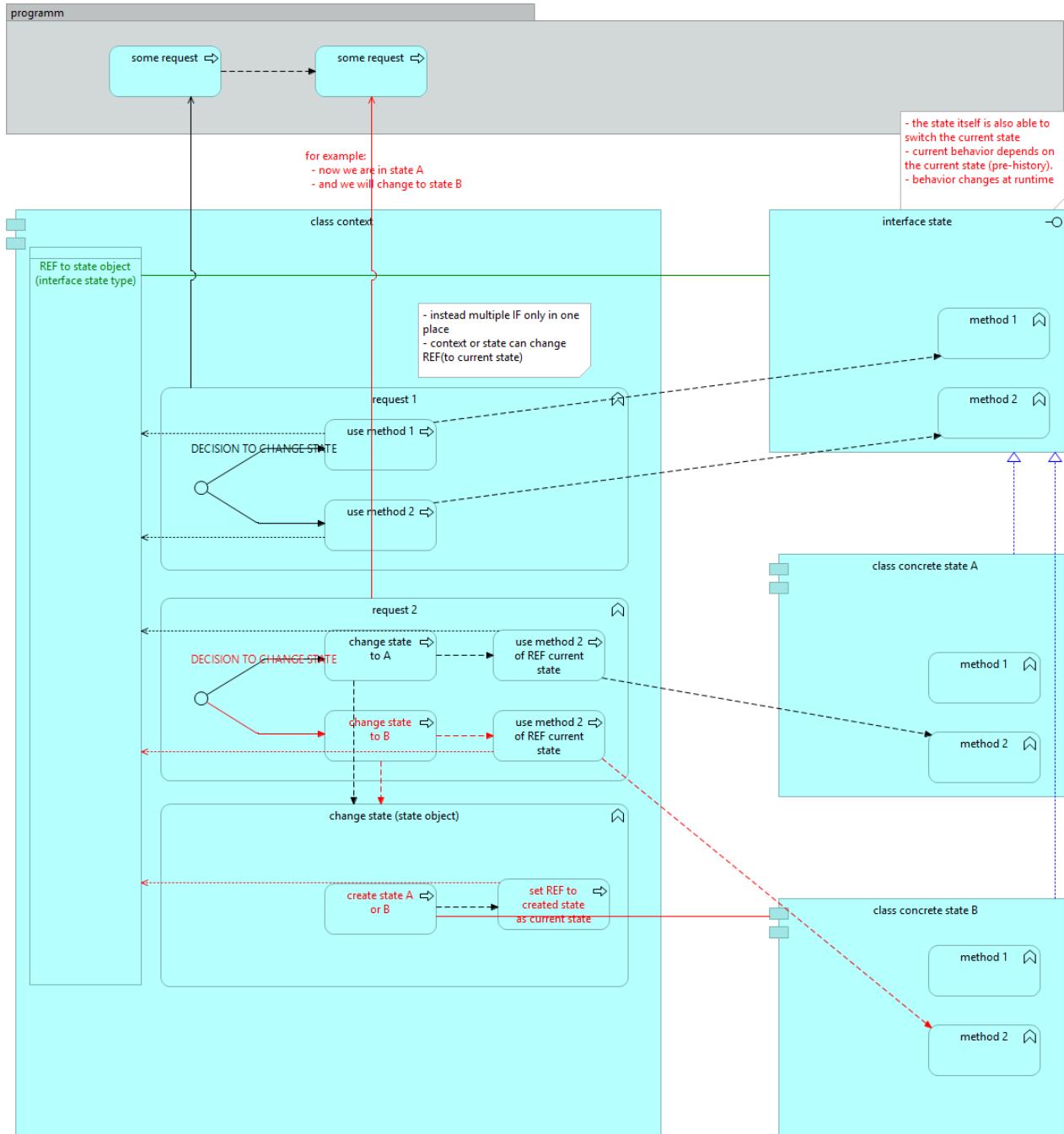


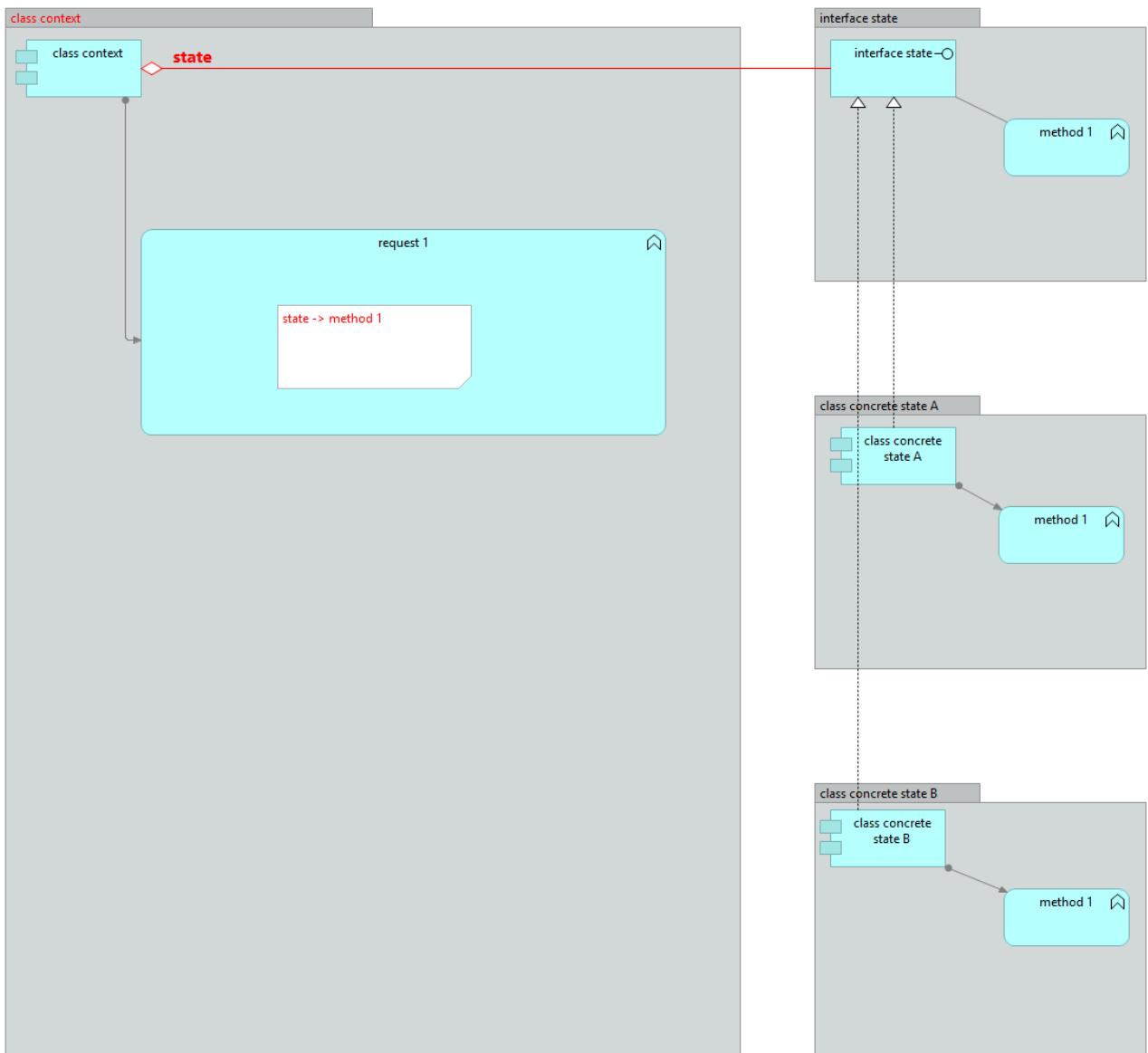
OBSERVER



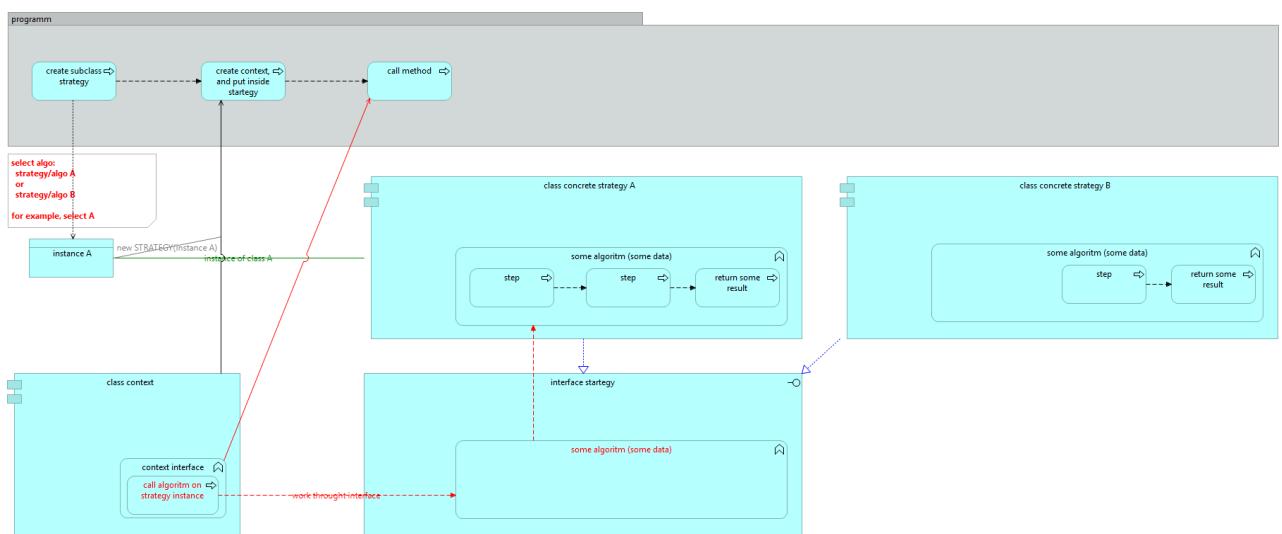


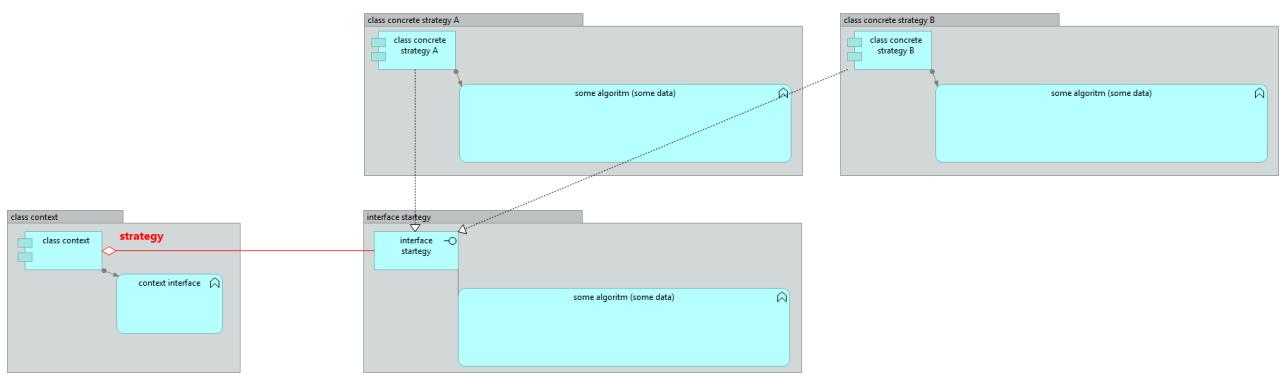
STATE



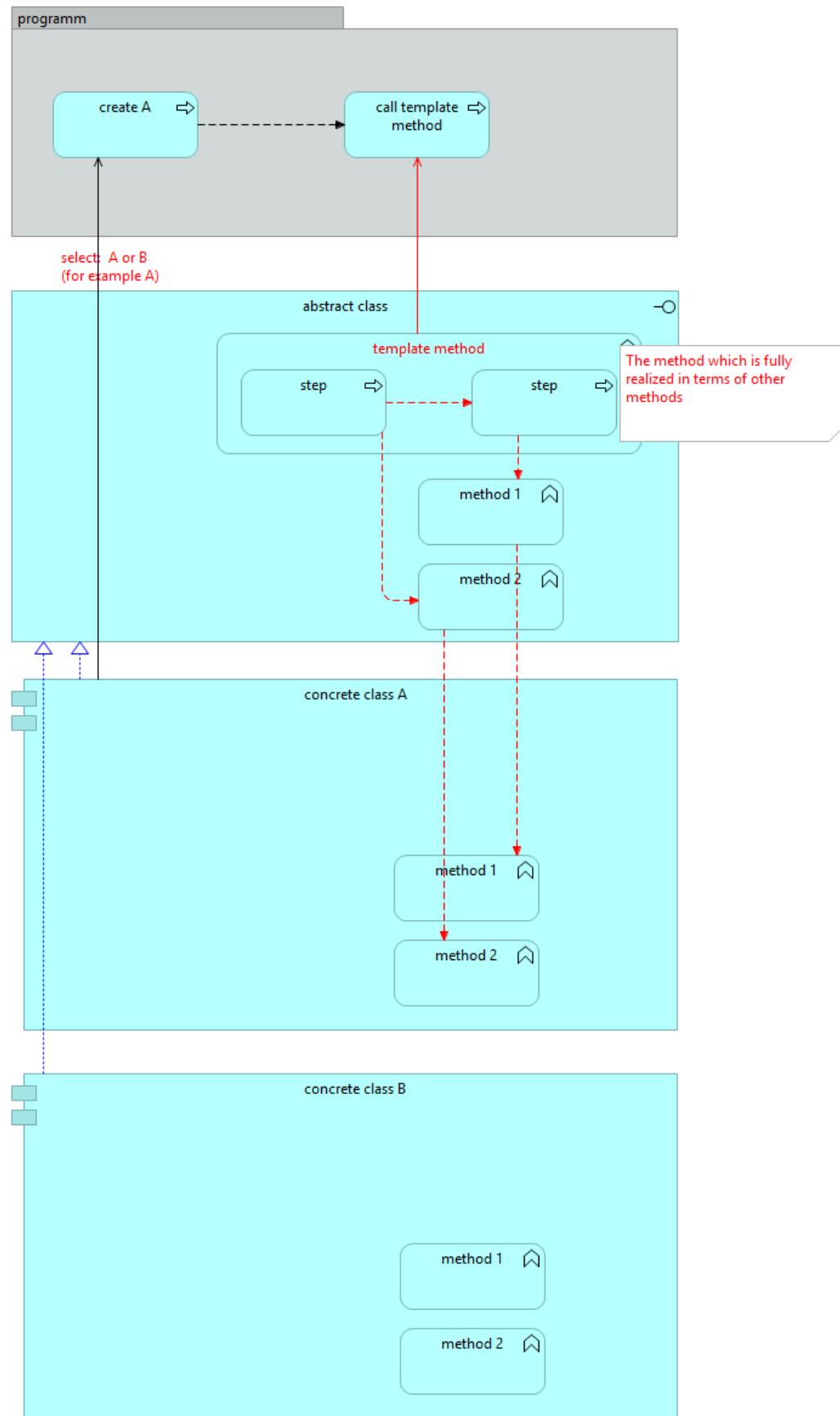


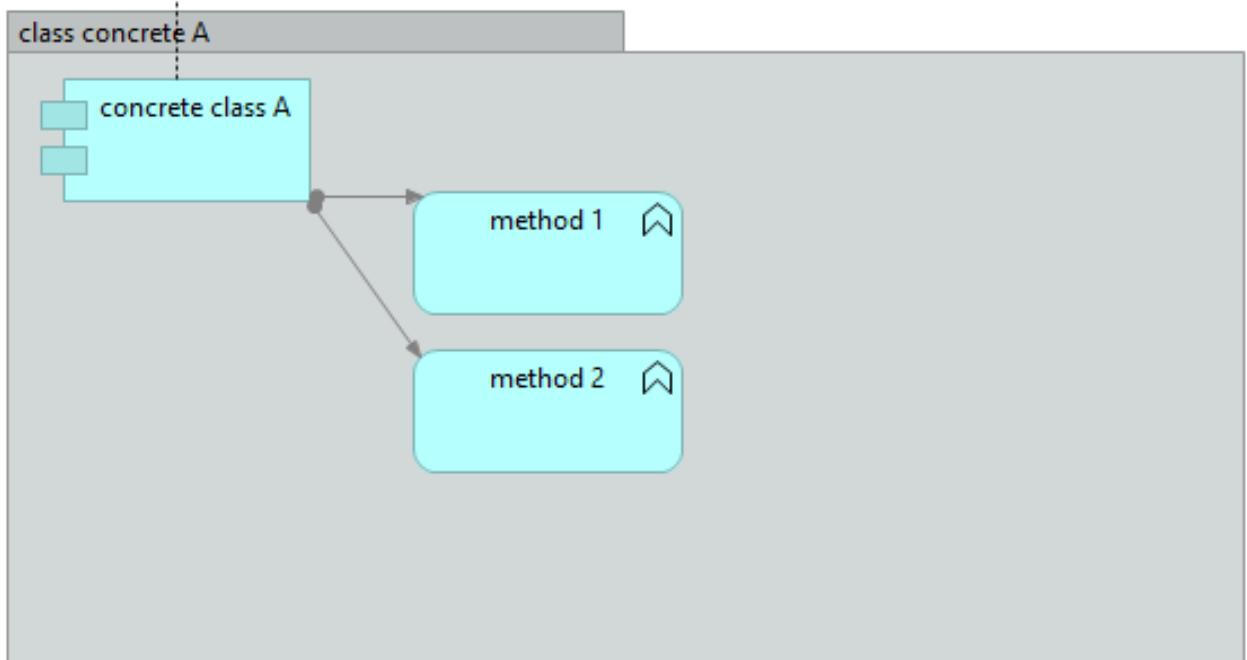
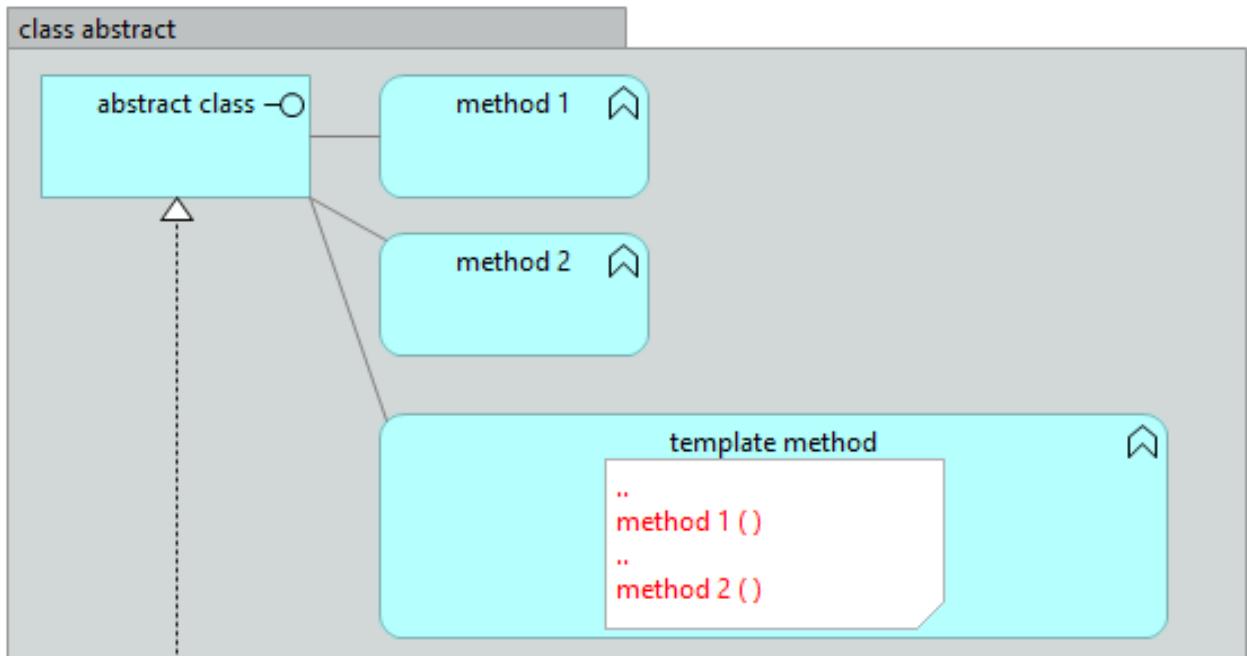
STRATEGY



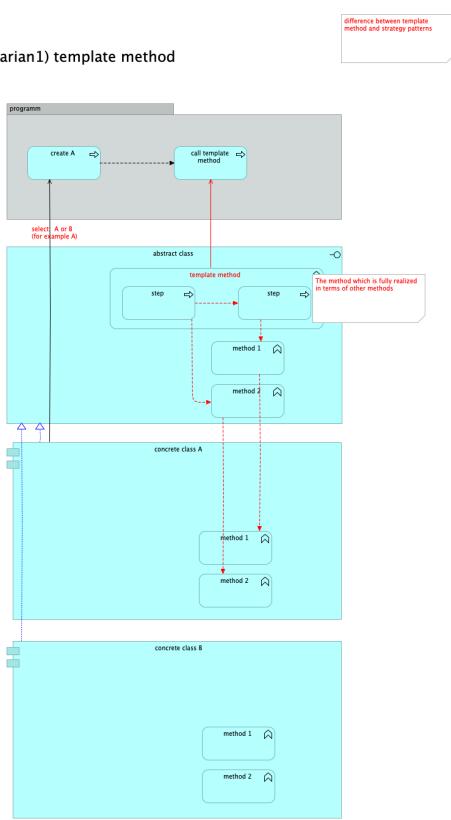


TEMPLATE METHOD

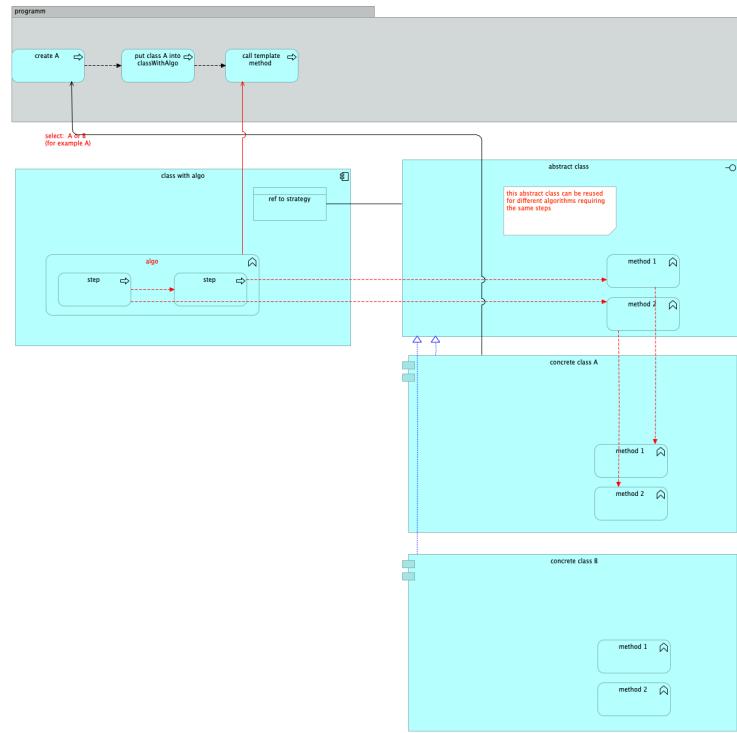




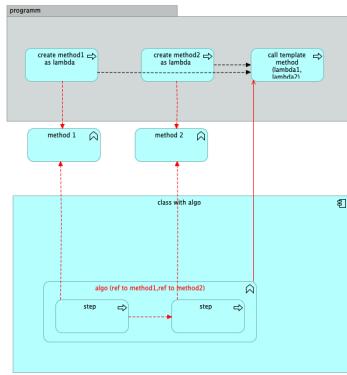
Varian1) template method



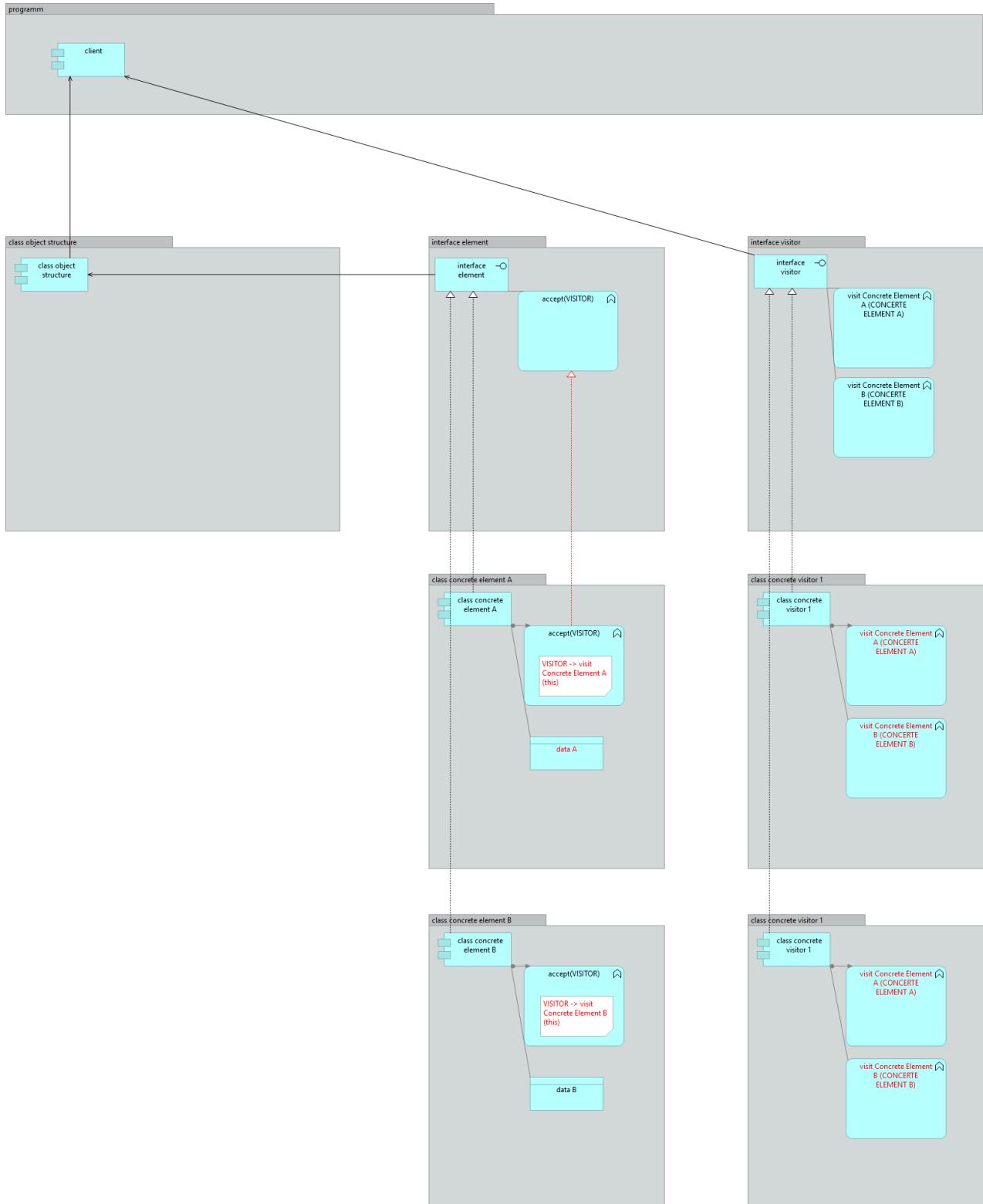
Varian2) strategy

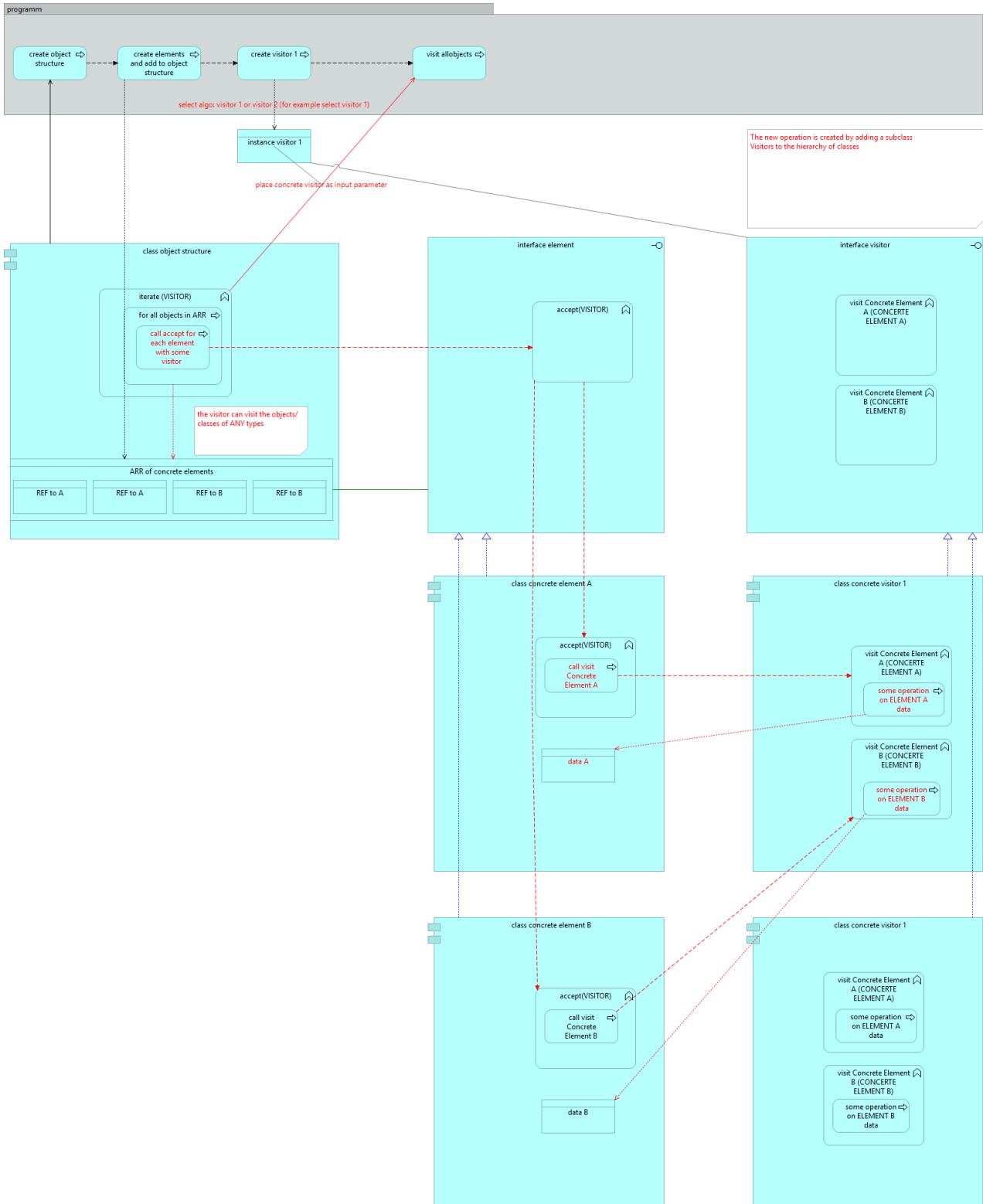


Varian3) lambdas as input parameters

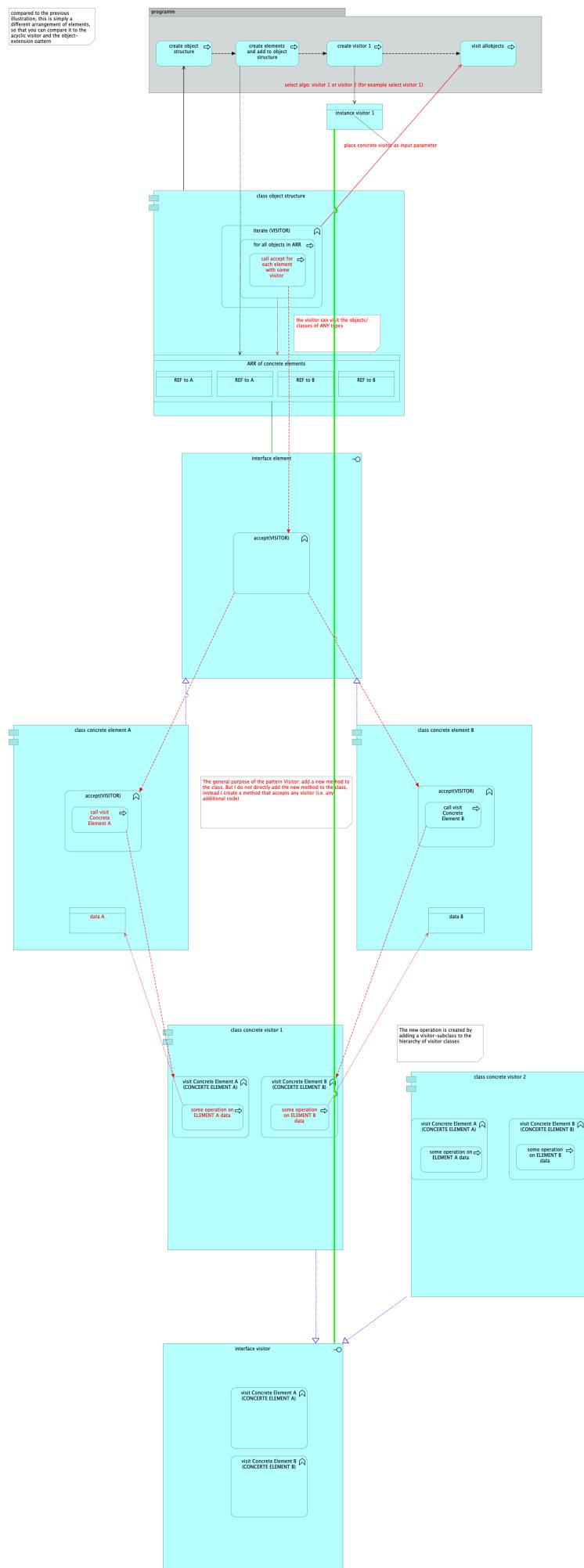


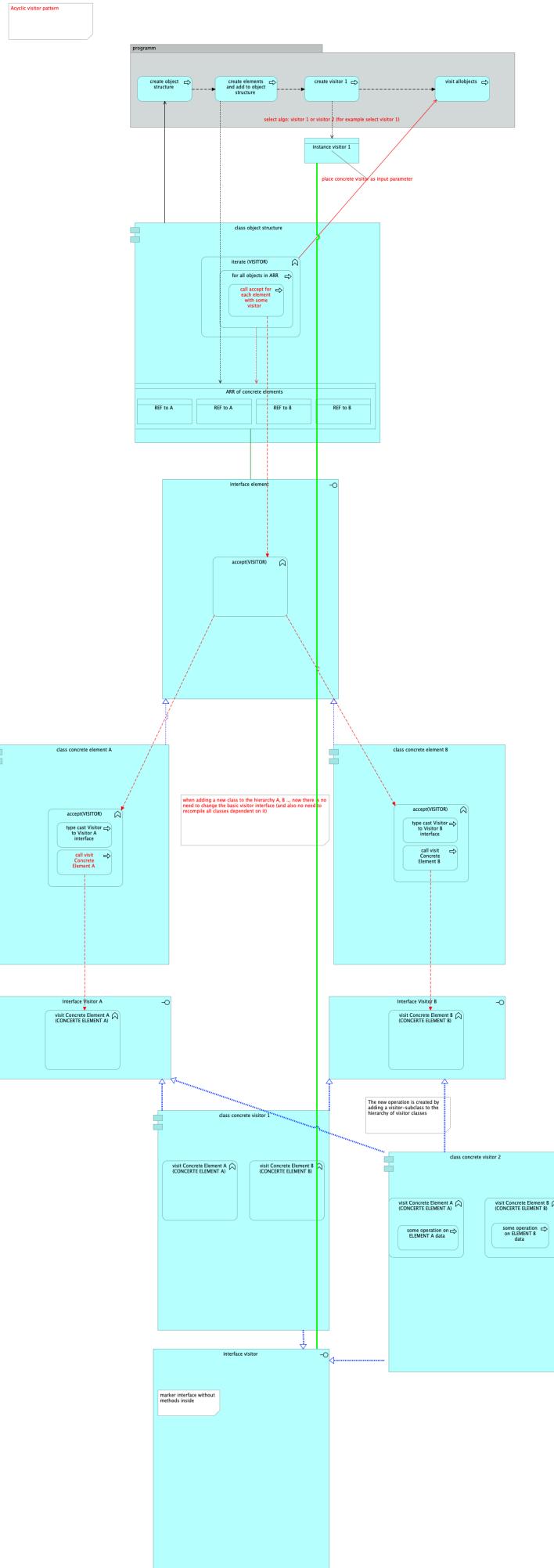
VISITOR



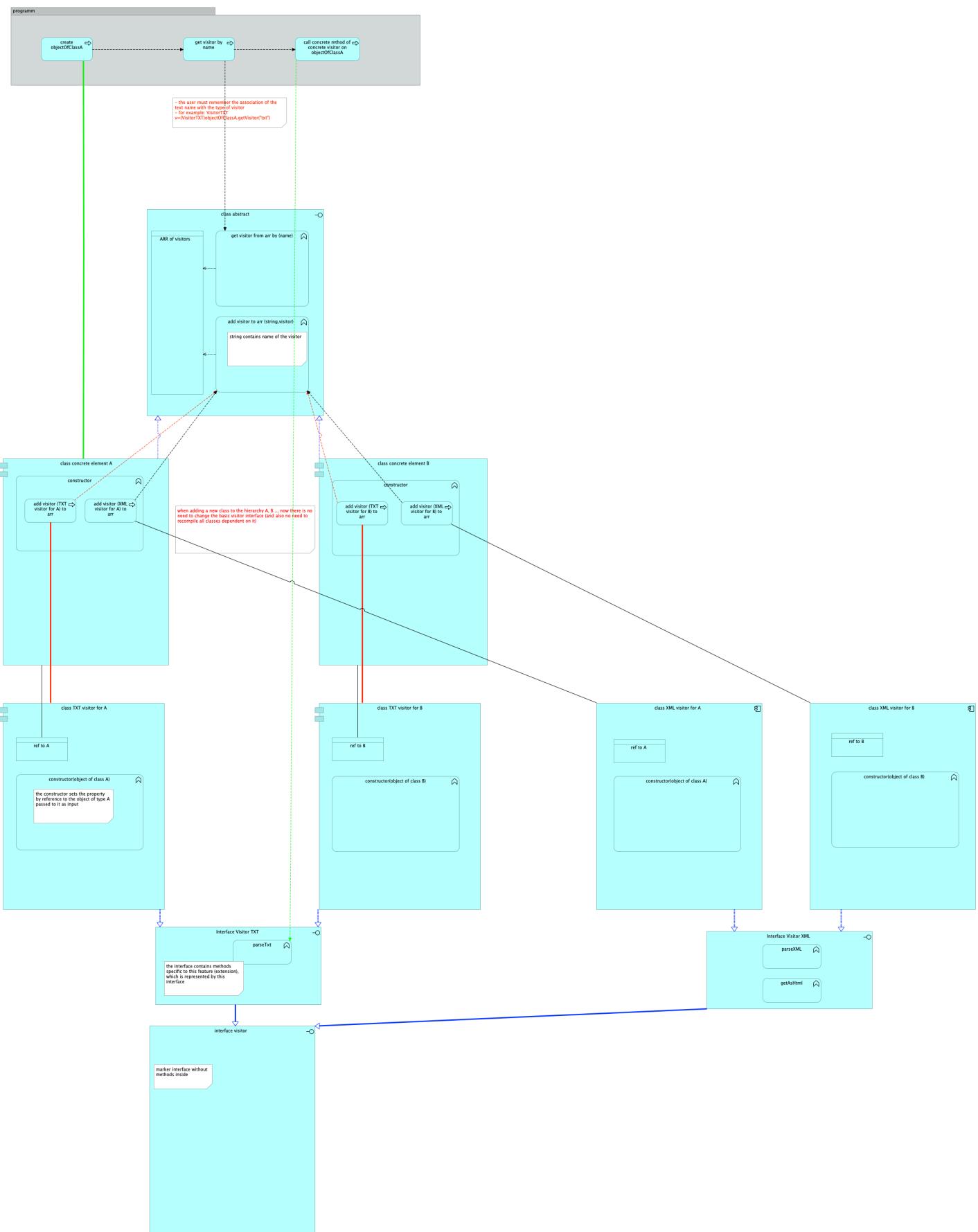


compared to the previous illustration, this one shows a different arrangement of elements, so that you can compare it to the application of the object-extension pattern.





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



02

Enterprise Patterns

Catalog of Patterns of Enterprise

MARTIN FOWLER



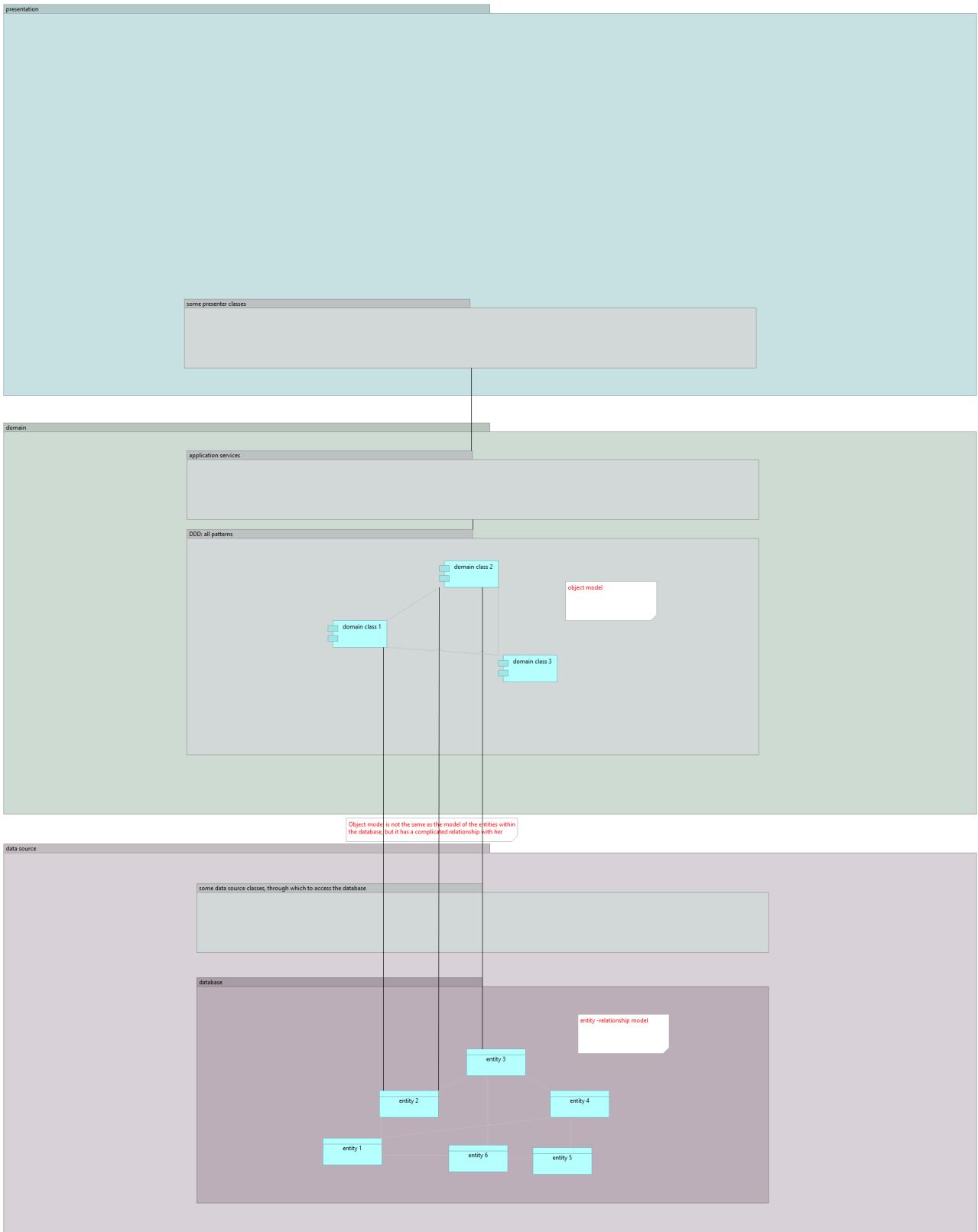
BUSINESS LOGIC

ENTERPRISE PATTERNS

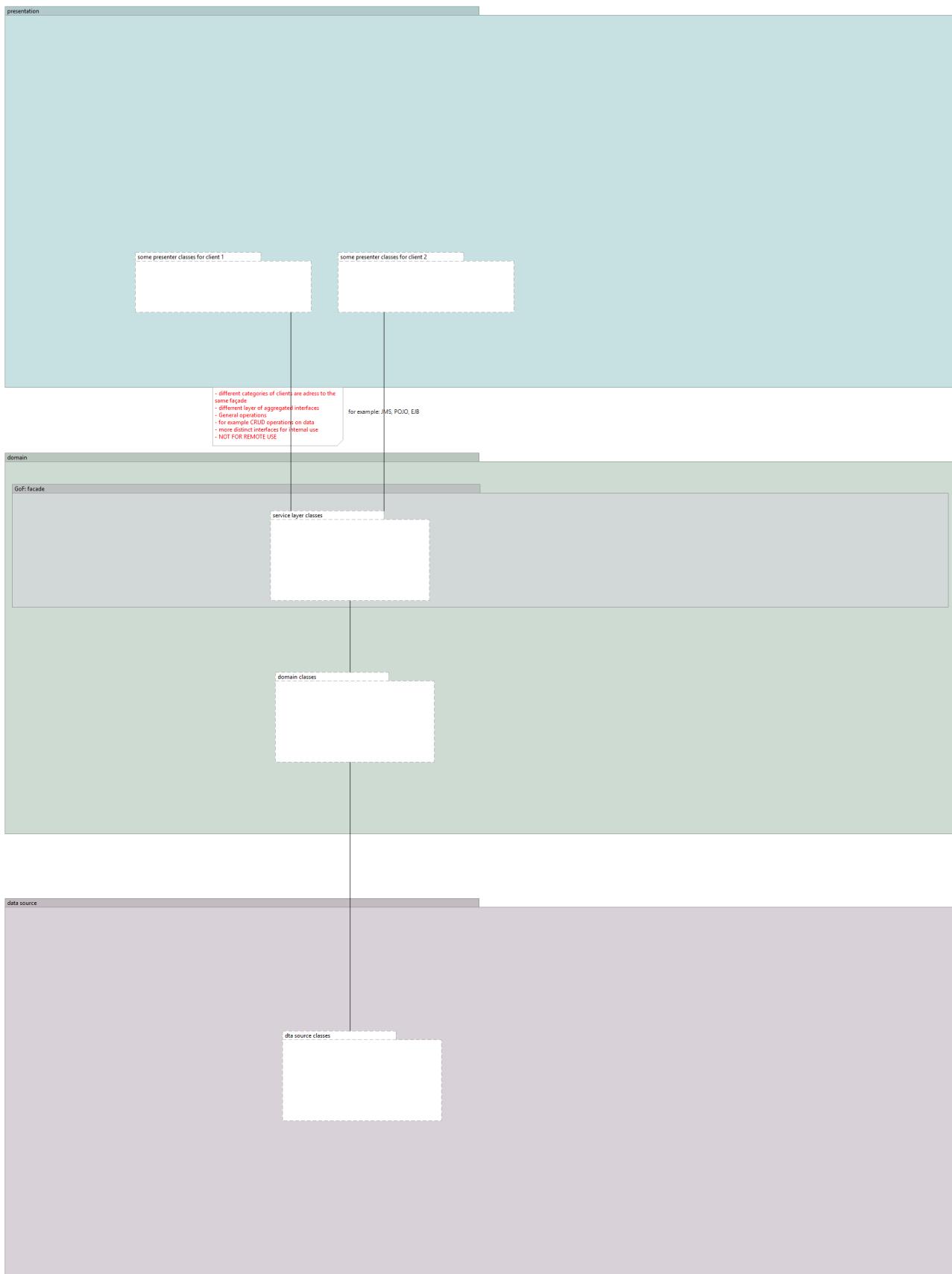
Martin Fowler



DOMAIN MODEL



SERVICE LAYER



TRANSACTION SCRIPT

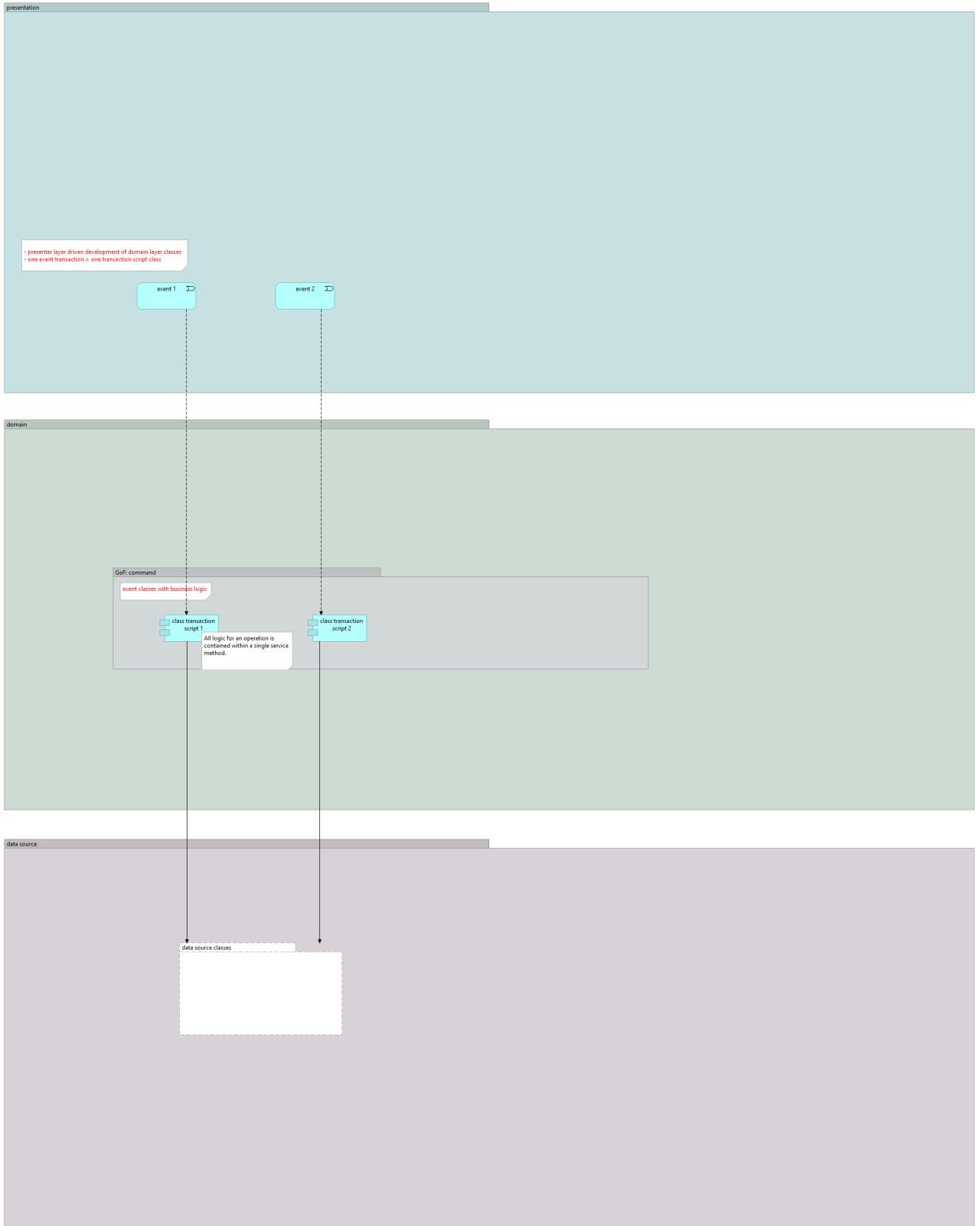
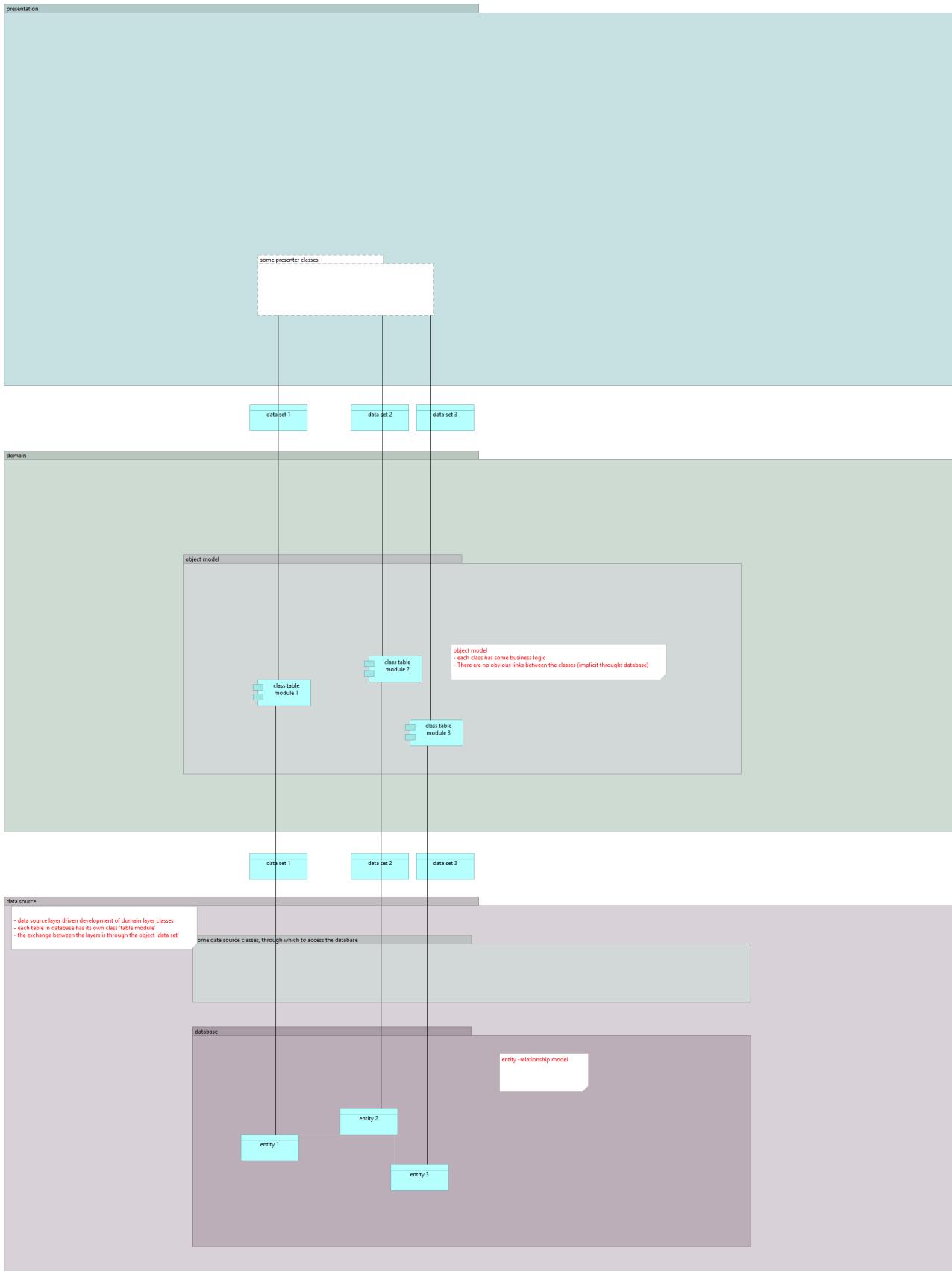


TABLE MODULE



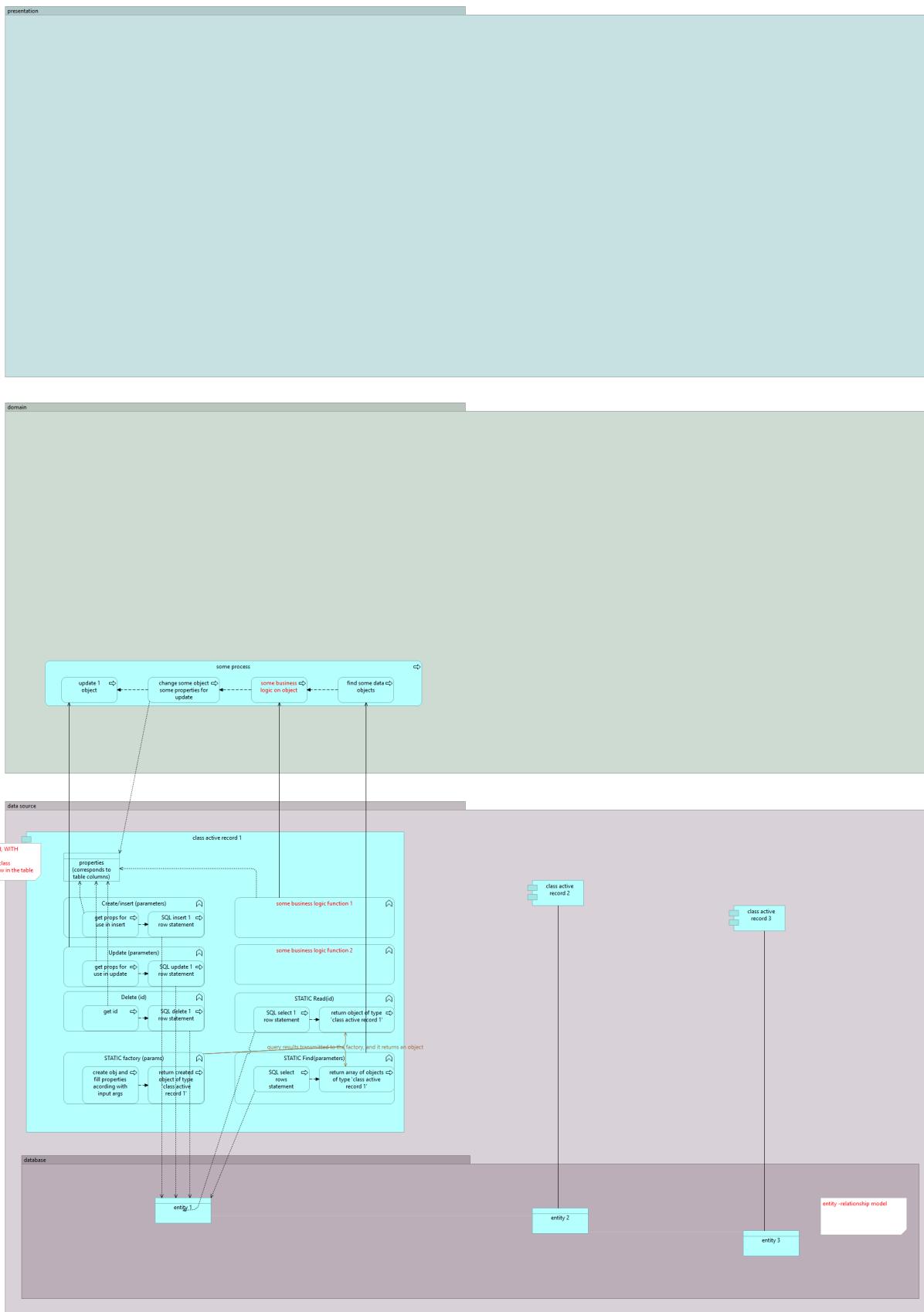
DATA SOURCES

ENTERPRISE PATTERNS

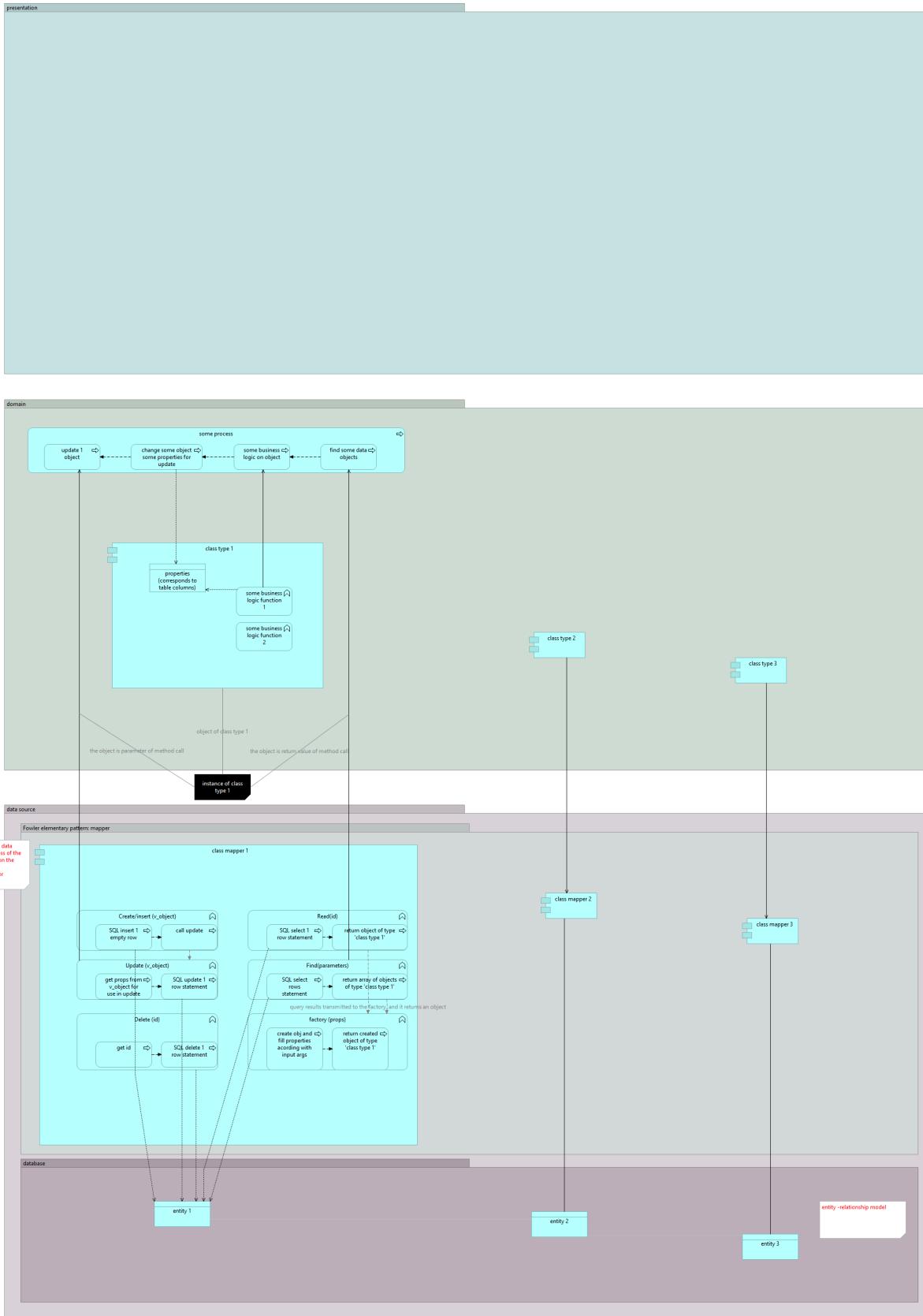
Martin Fowler



ACTIVE RECORD



DATA MAPPER



ROW DATA GATEWAY

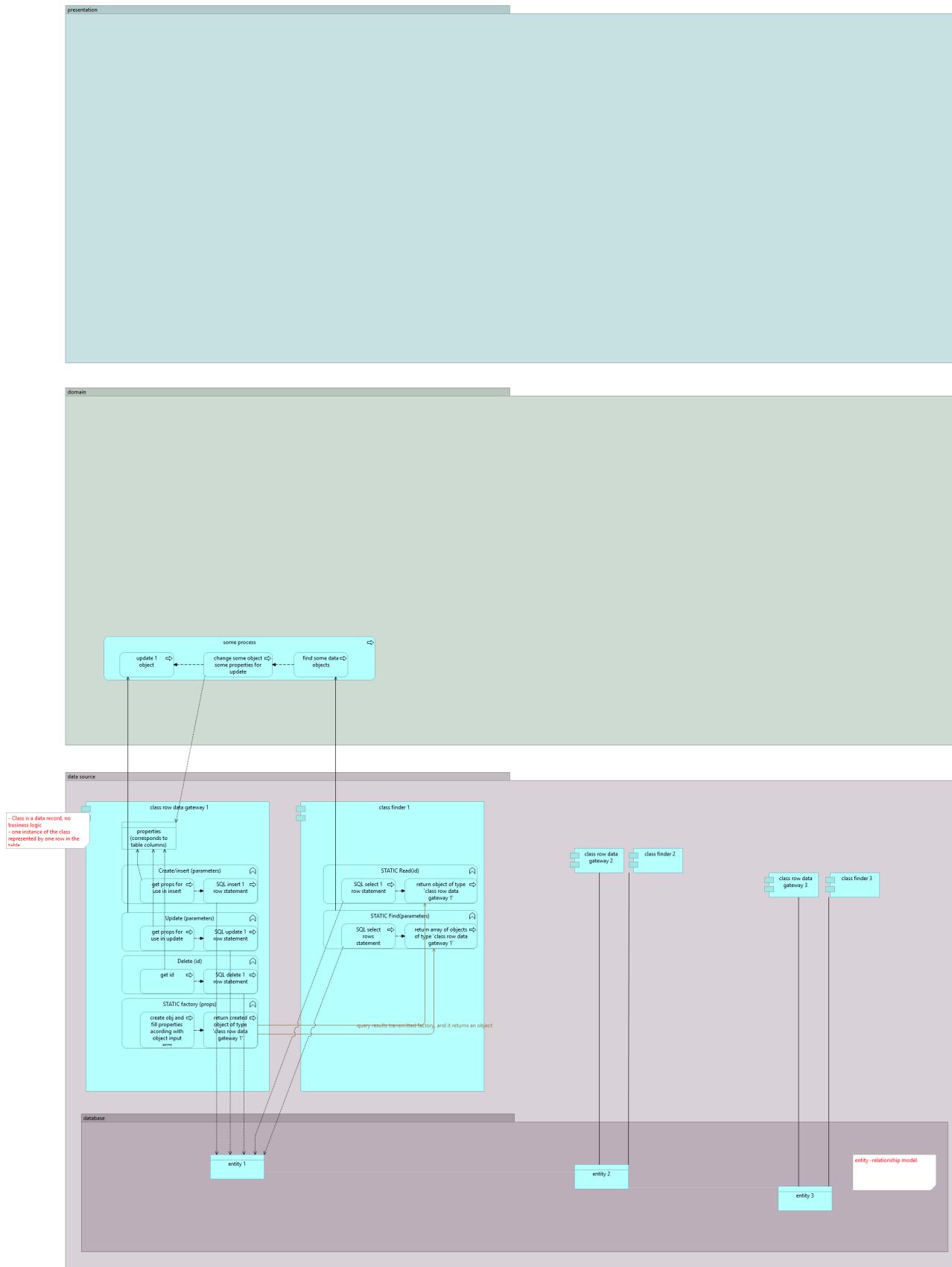
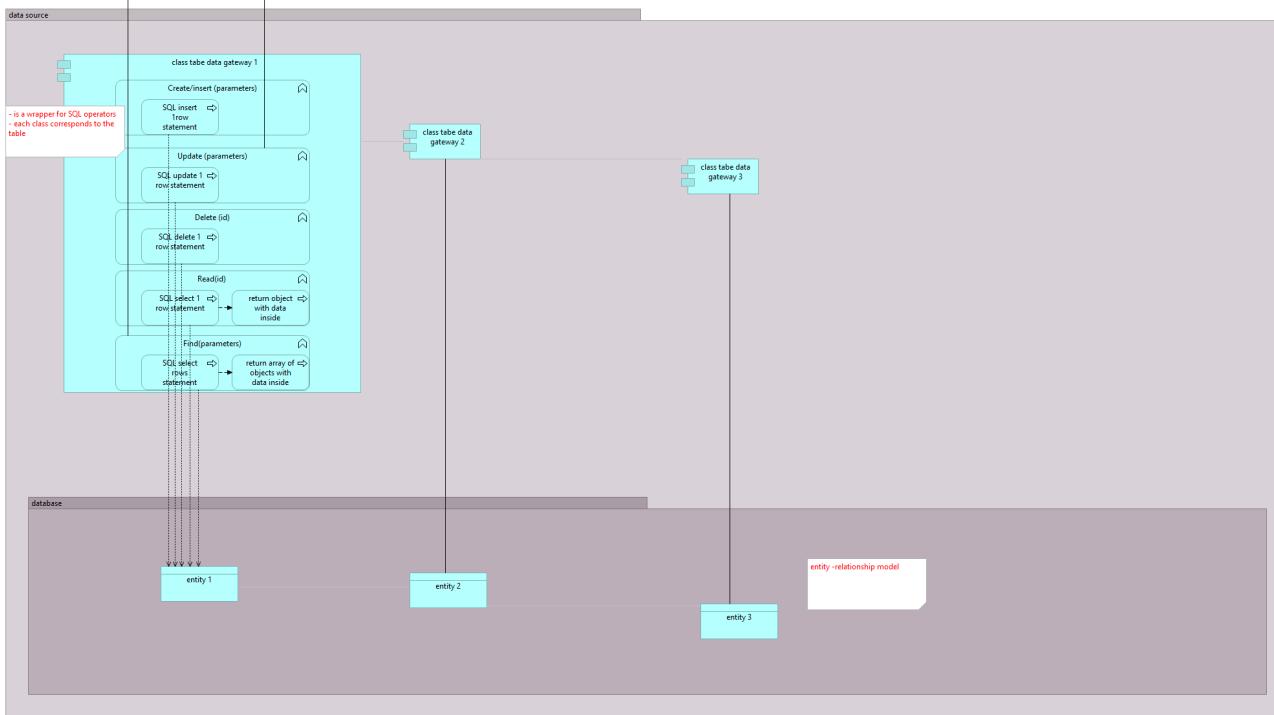
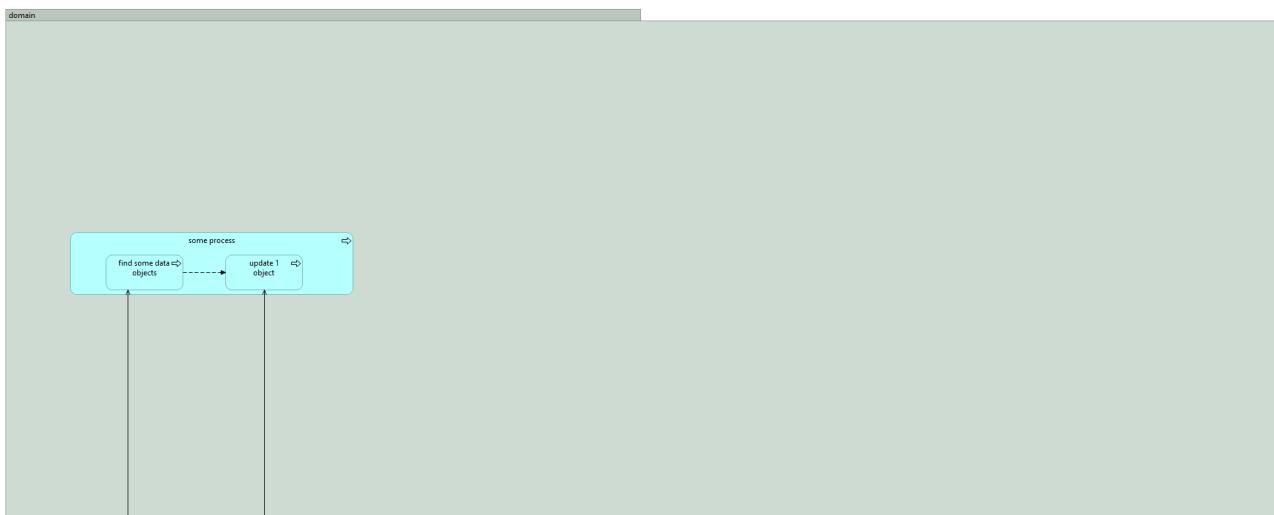
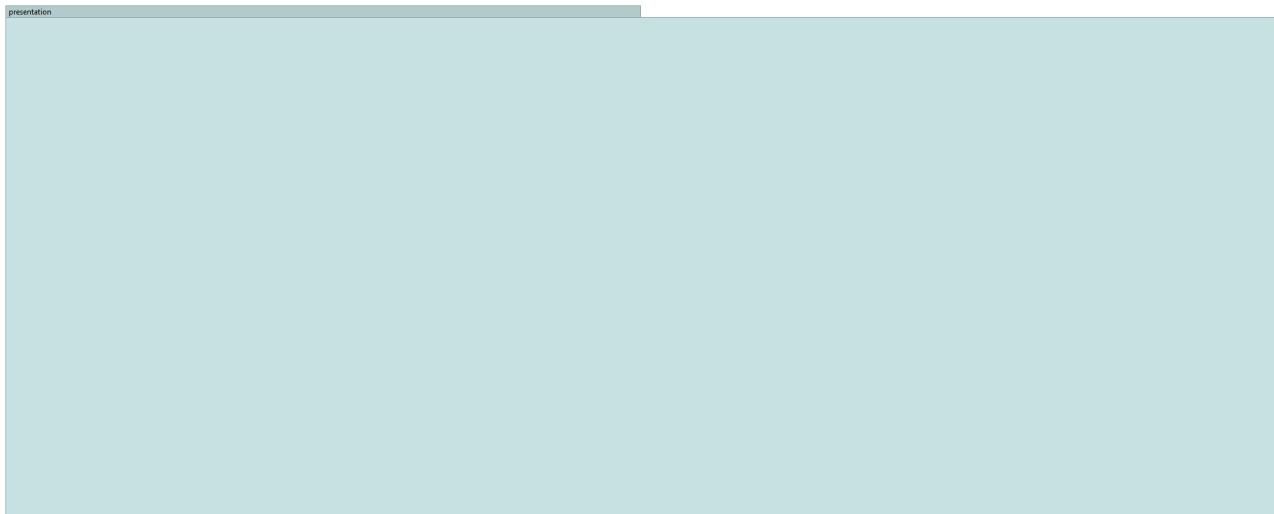


TABLE DATA GATEWAY



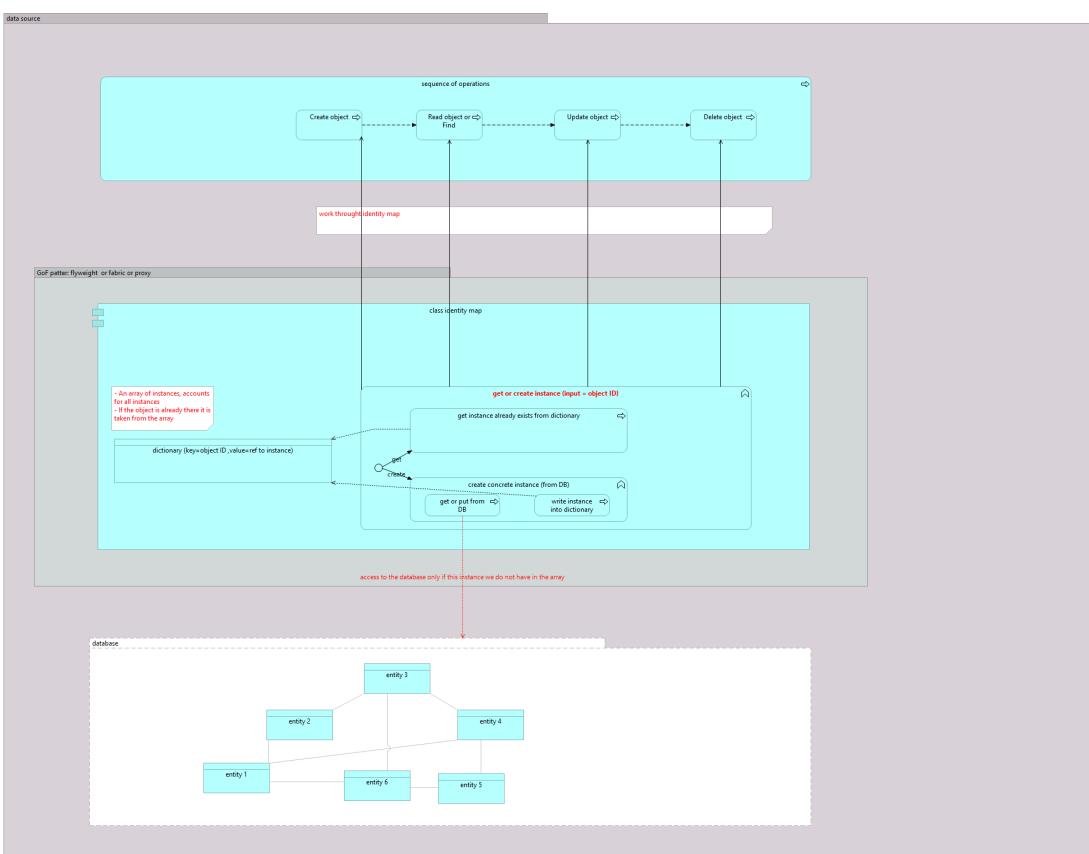
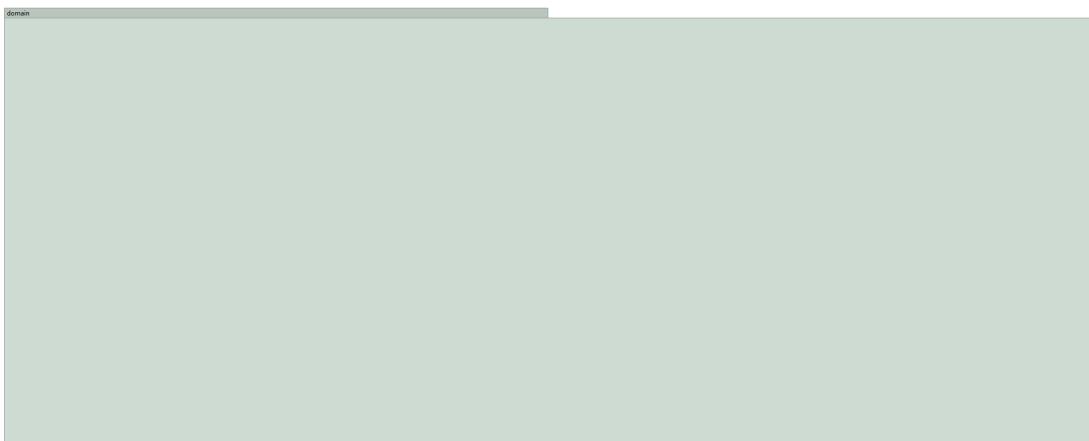
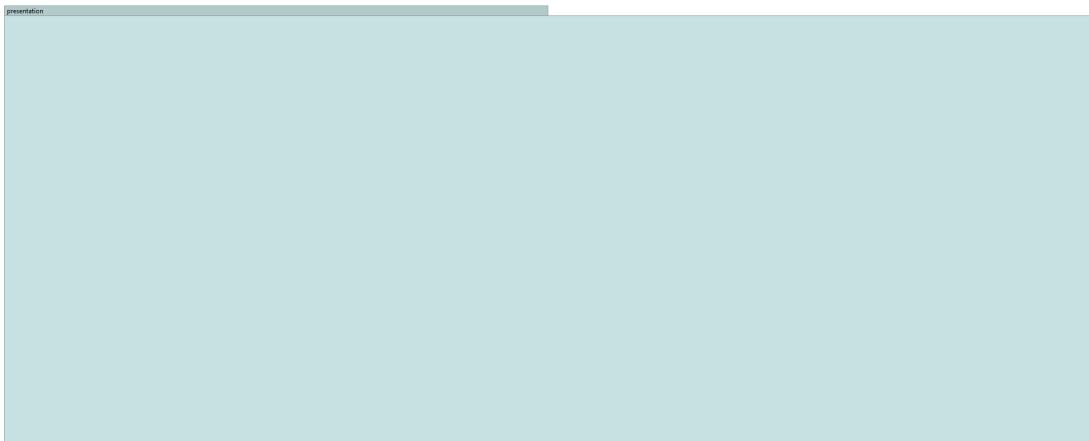
MODELING BEHAVIOR

ENTERPRISE PATTERNS

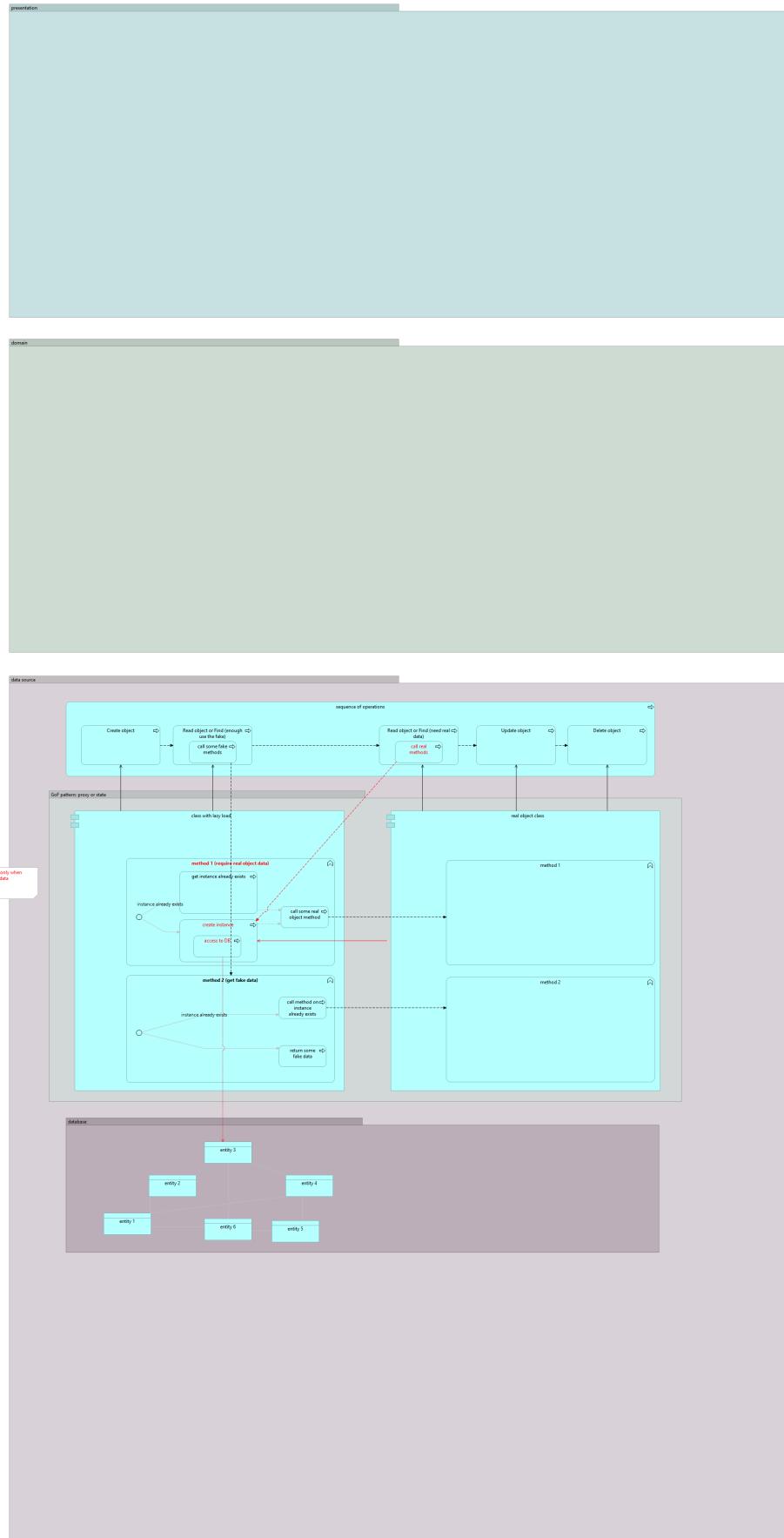
Martin Fowler



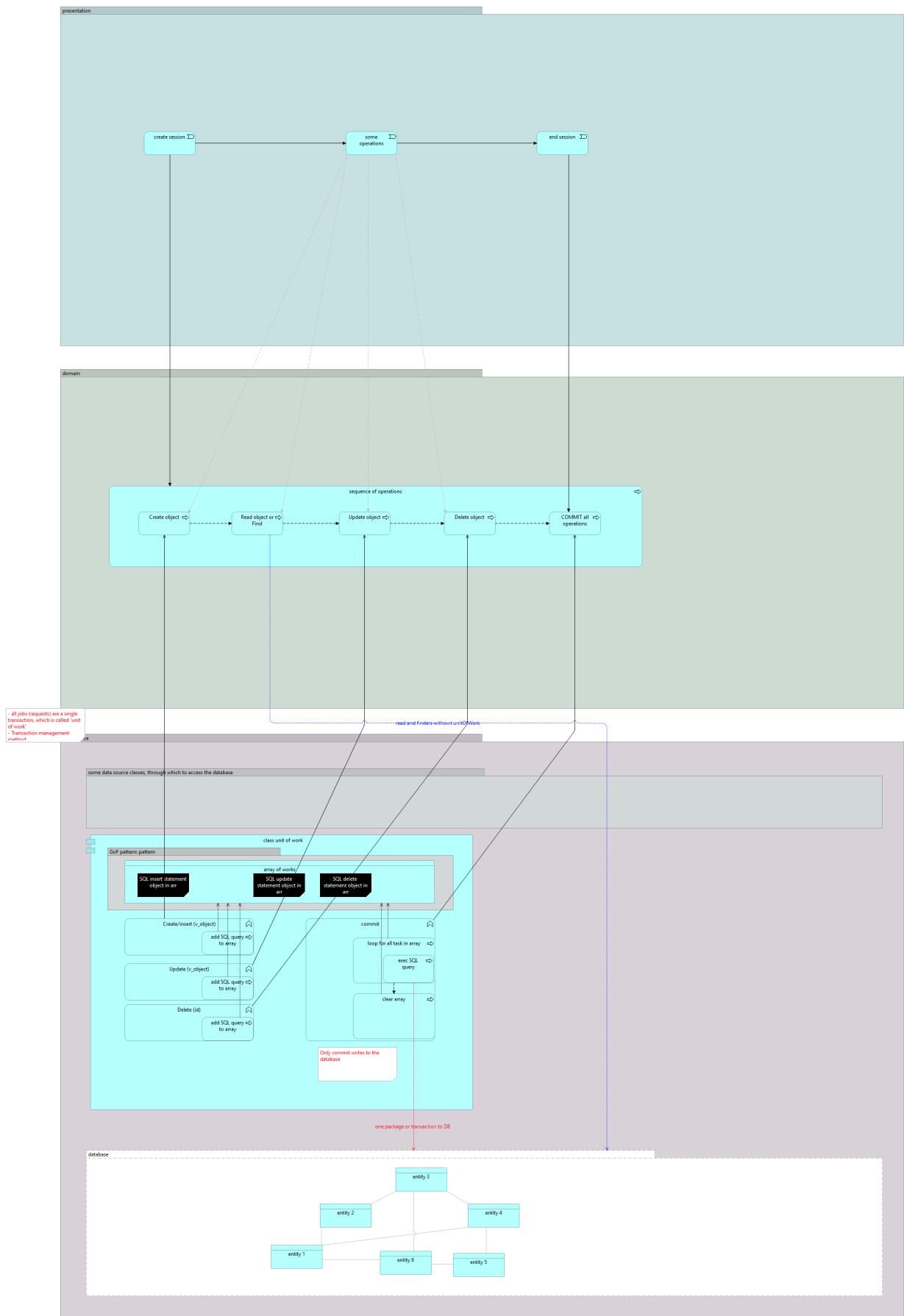
IDENTITY MAP



LAZY LOAD



UNIT OF WORK.



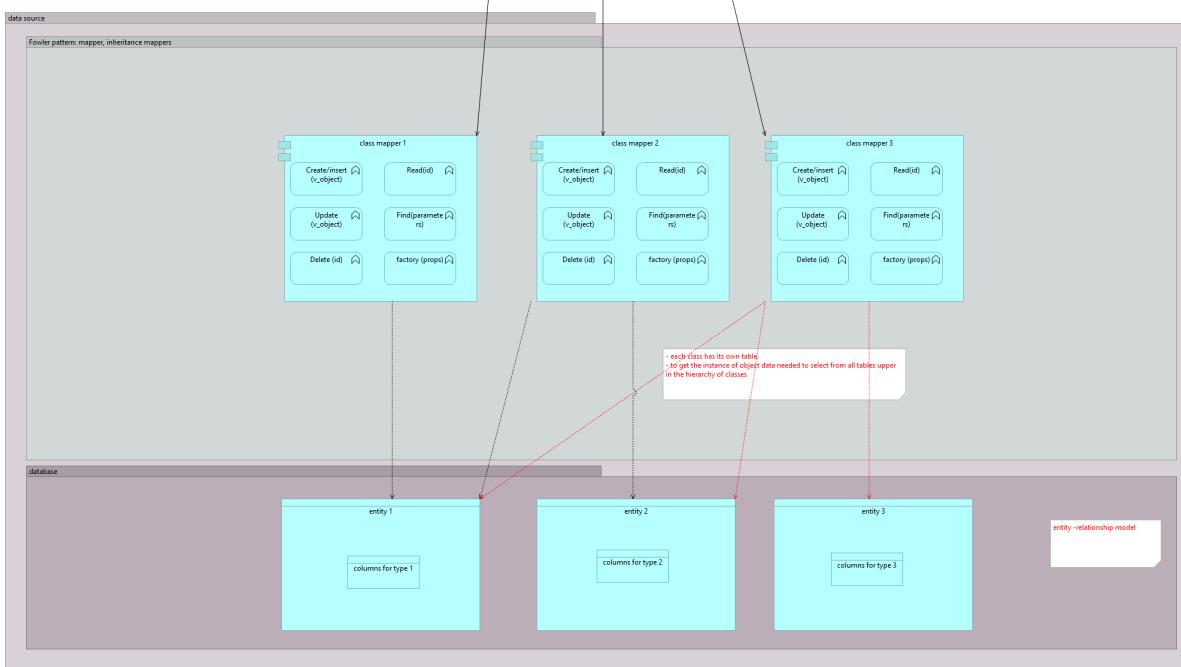
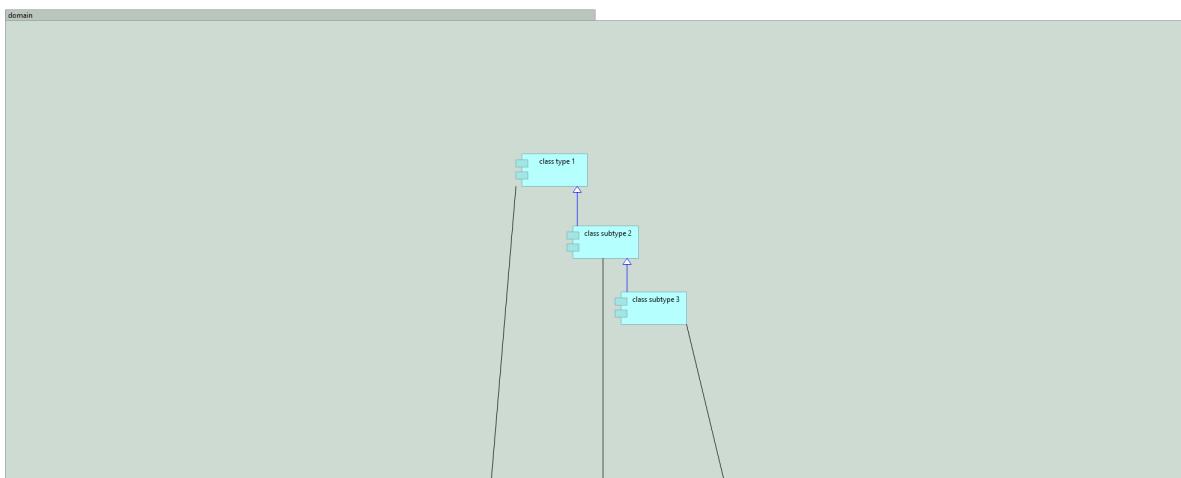
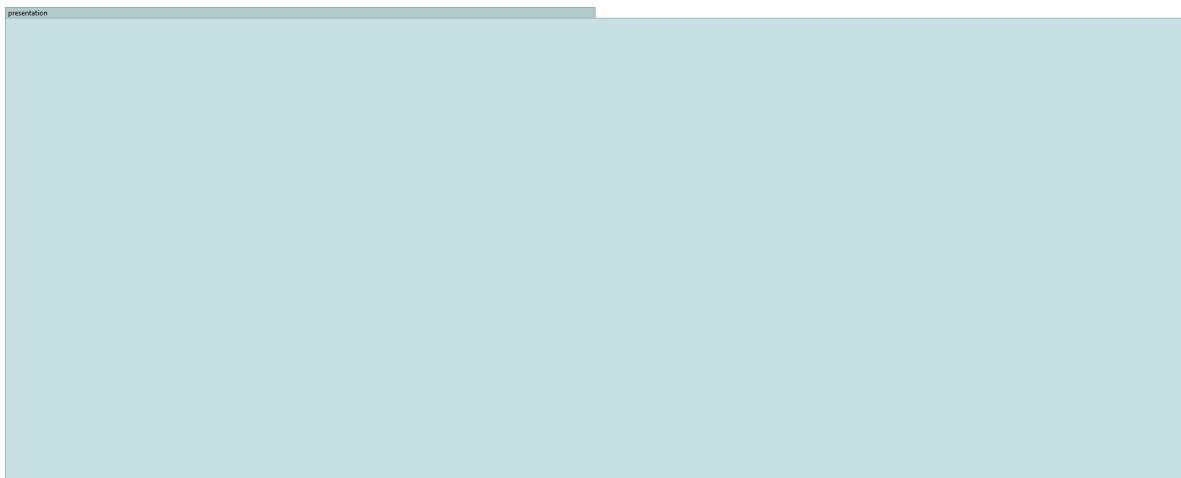
MODELING STRUCTURE HIERARCHY

ENTERPRISE PATTERNS

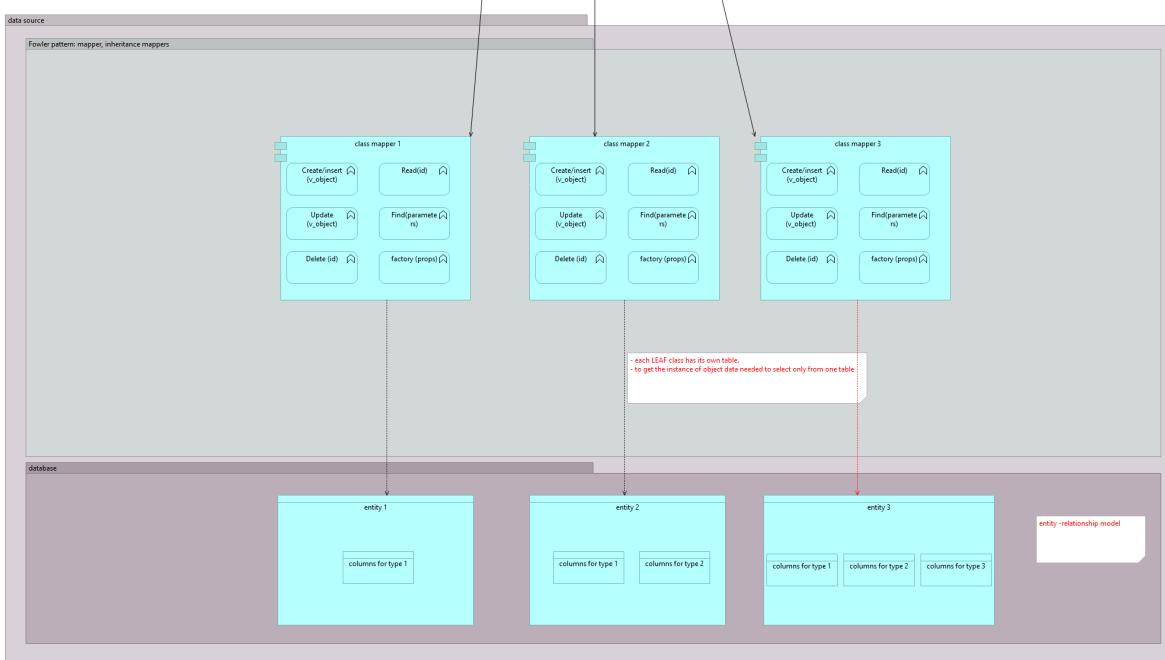
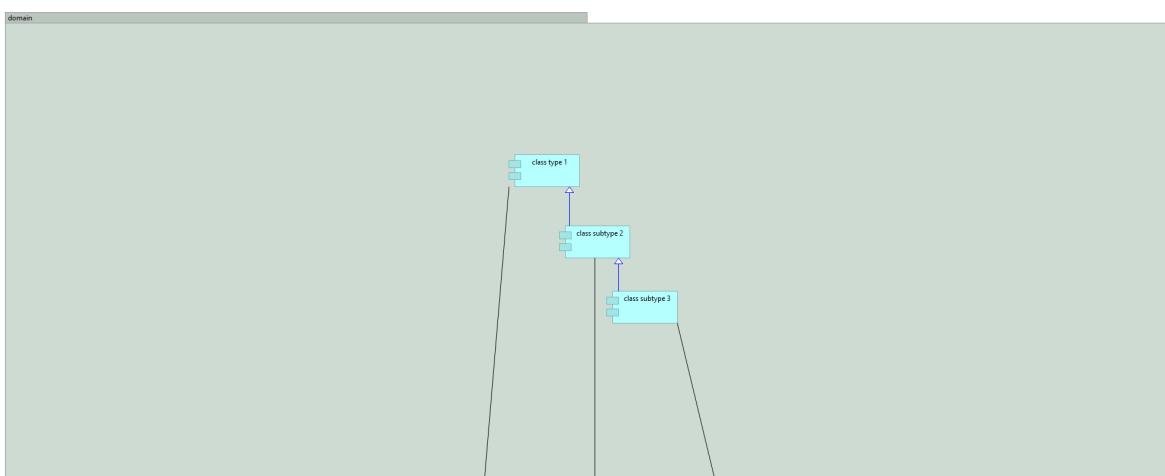
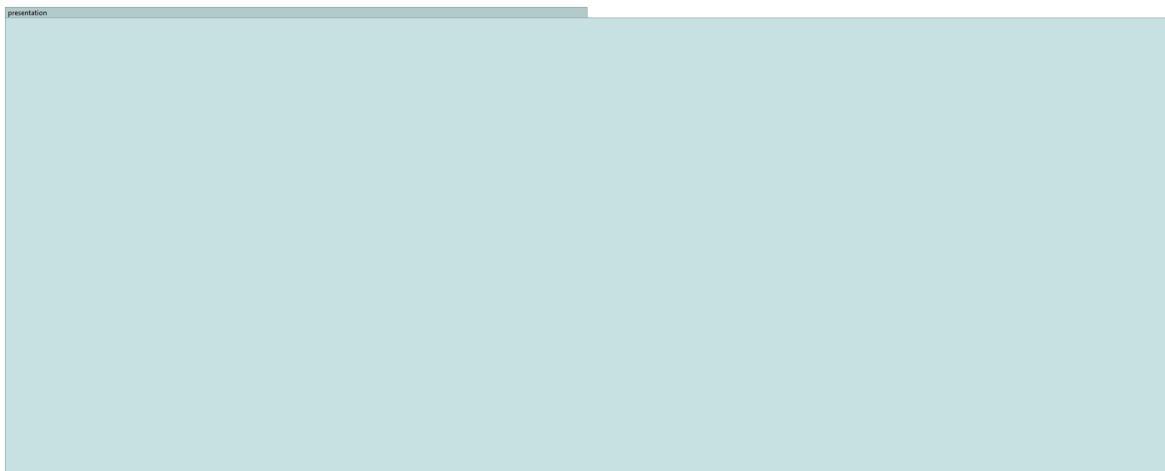
Martin Fowler



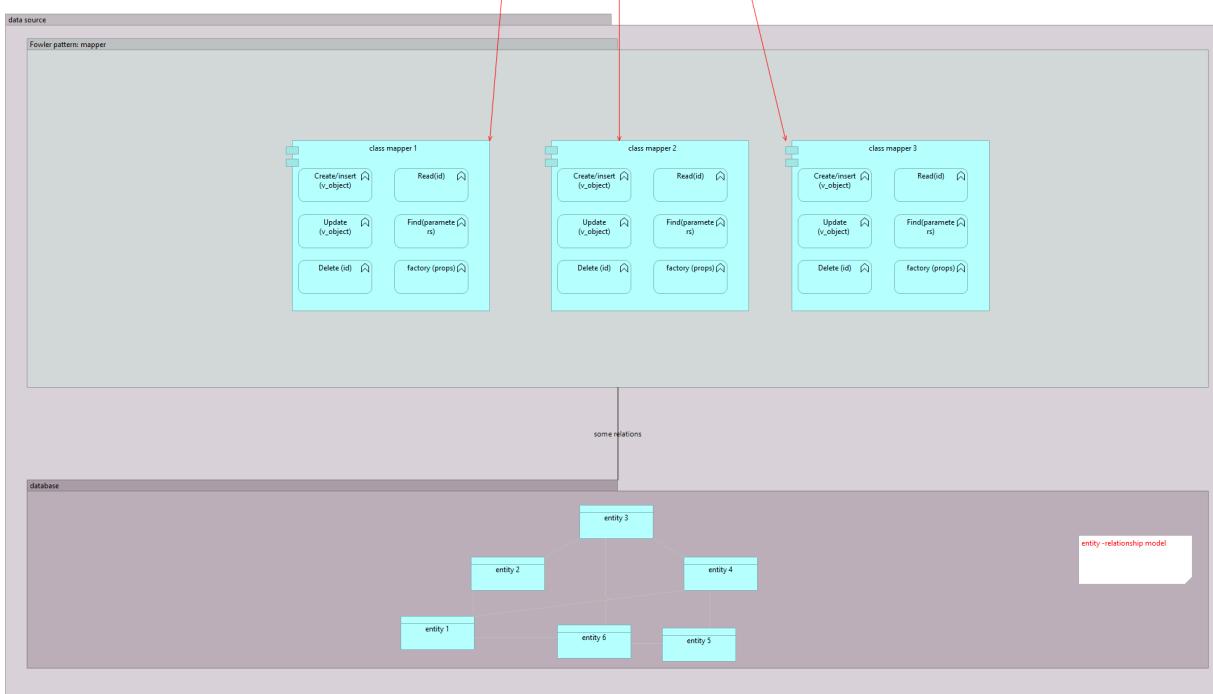
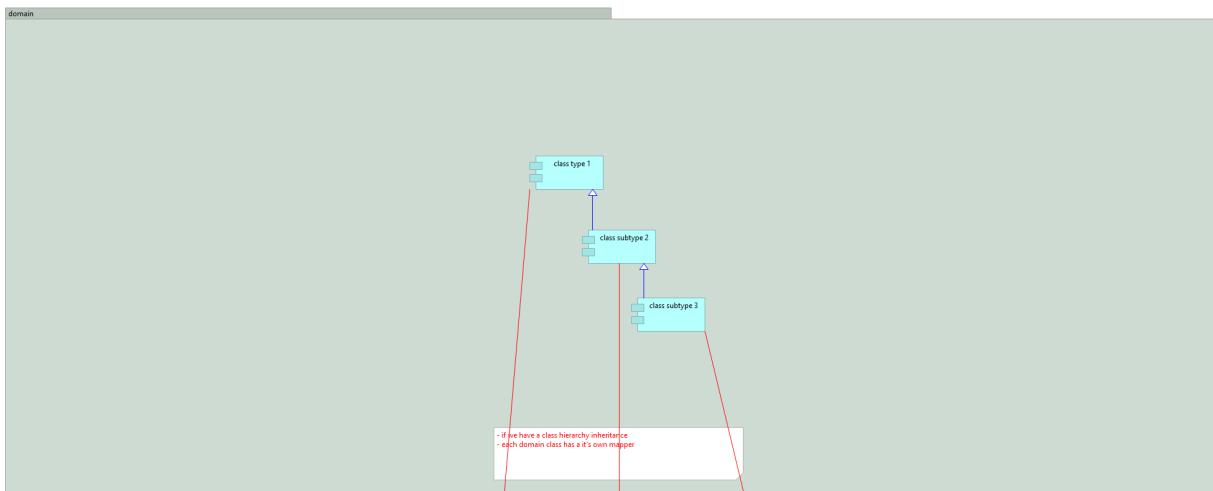
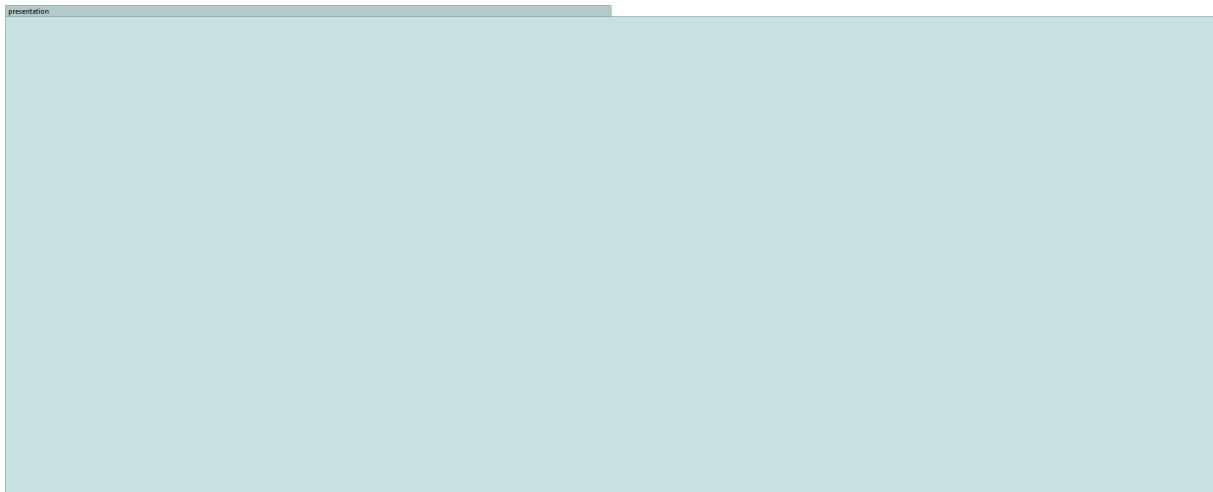
CLASS TABLE INHERITANCE



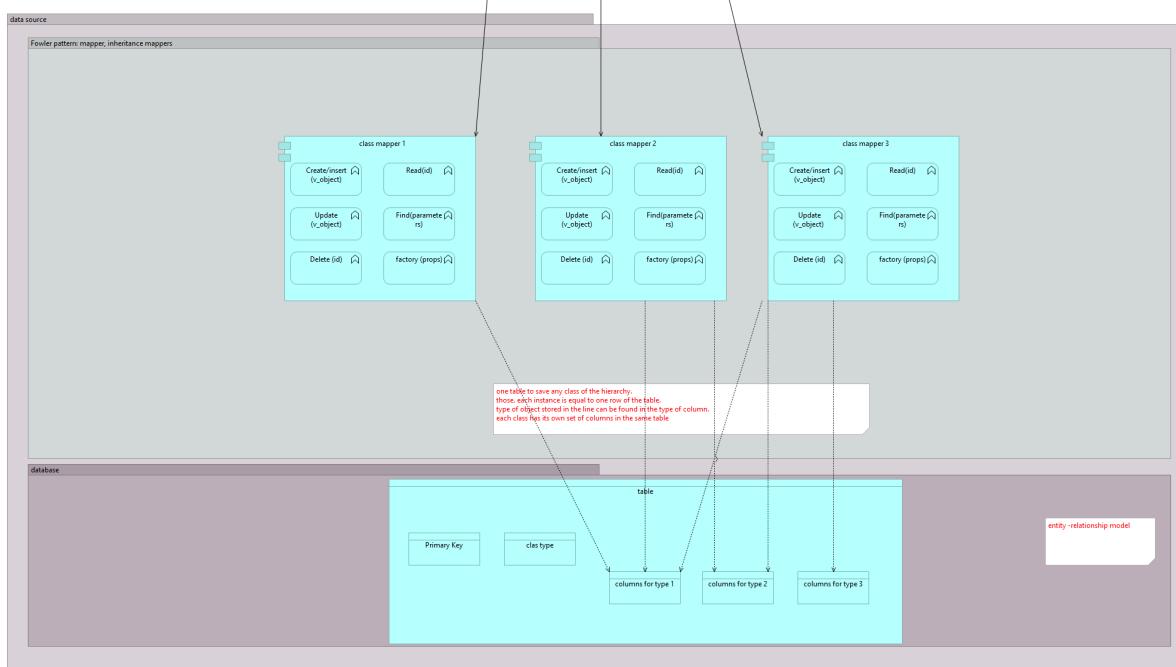
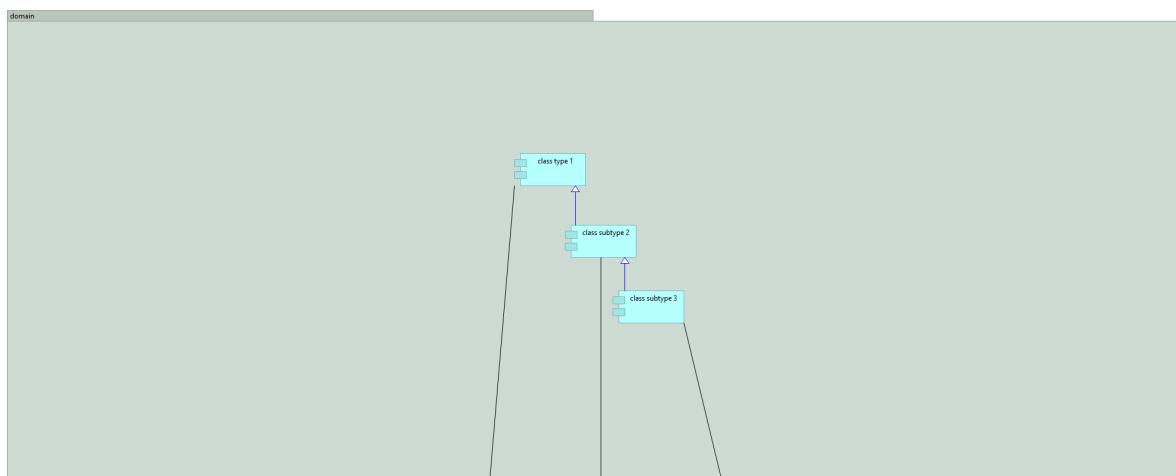
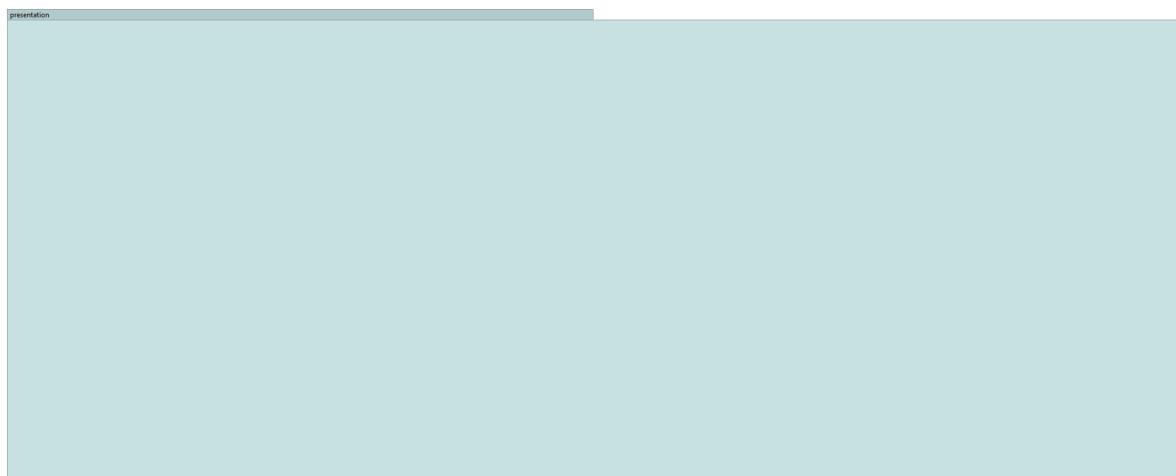
CONCRETE TABLE INHERITANCE



INHERITANCE MAPPERS



SINGLE TABLE INHERITANCE



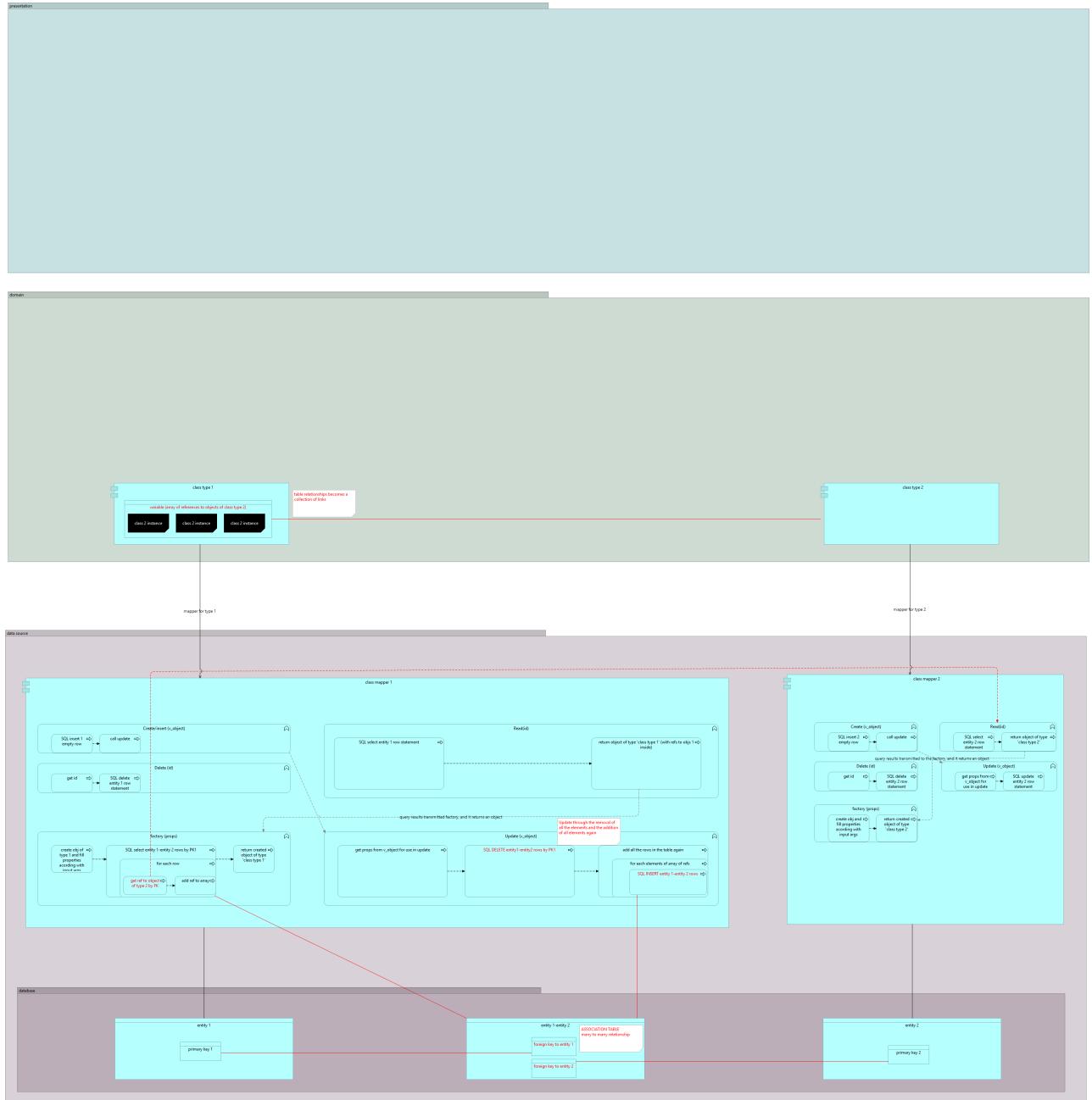
MODELING STRUCTURE RELATIONS

ENTERPRISE PATTERNS

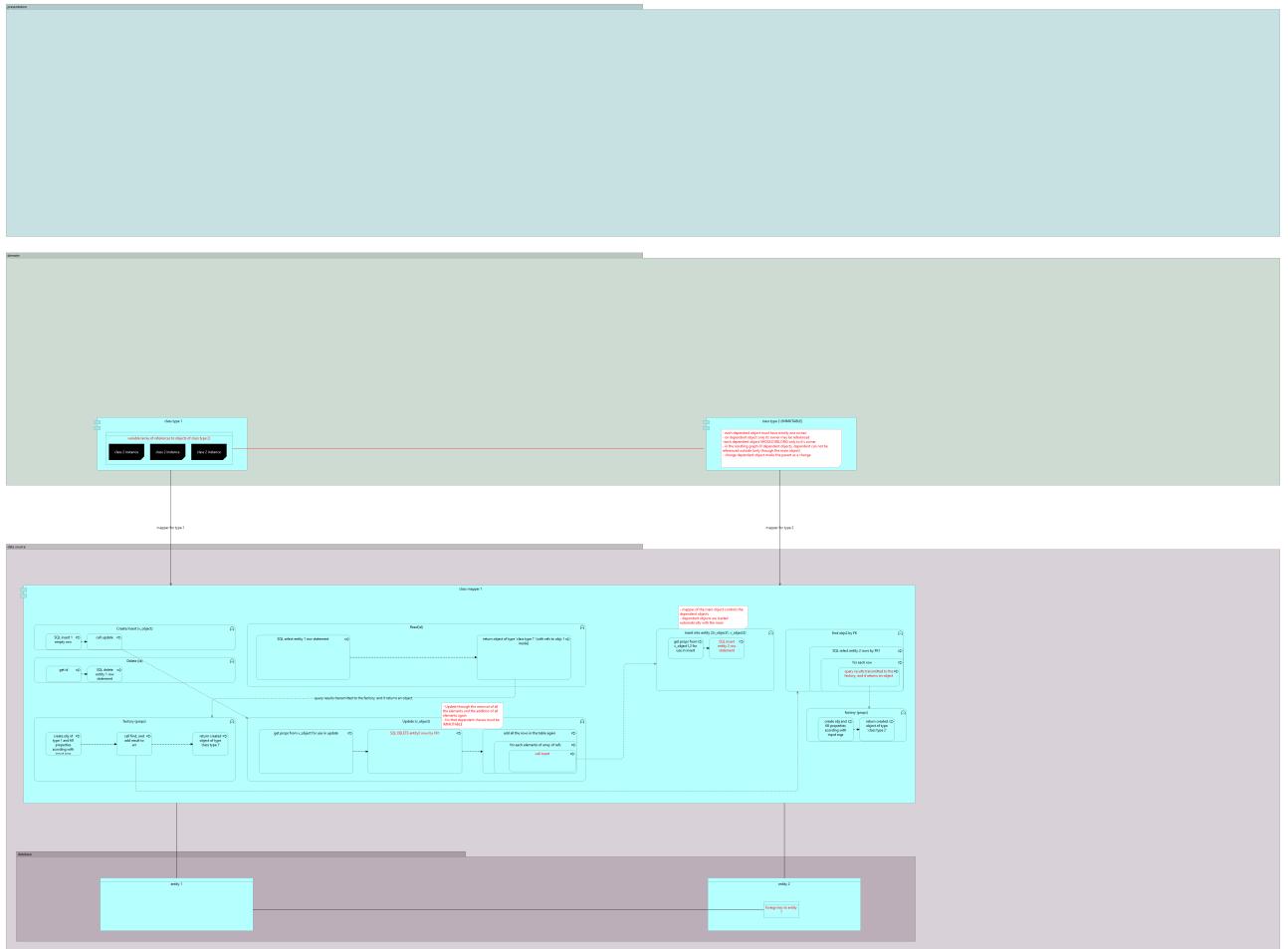
Martin Fowler



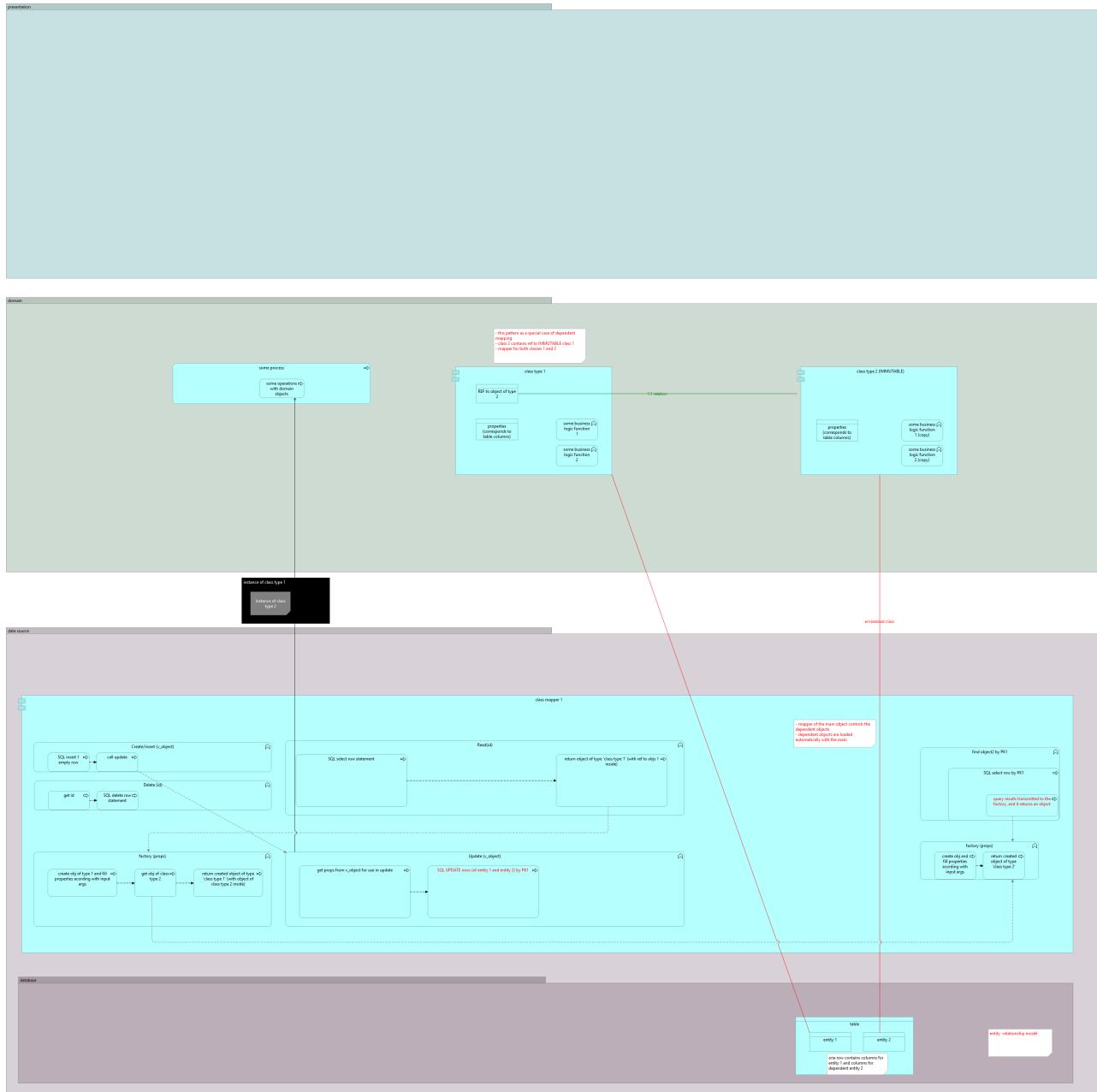
ASSOCIATION TABLE MAPPING



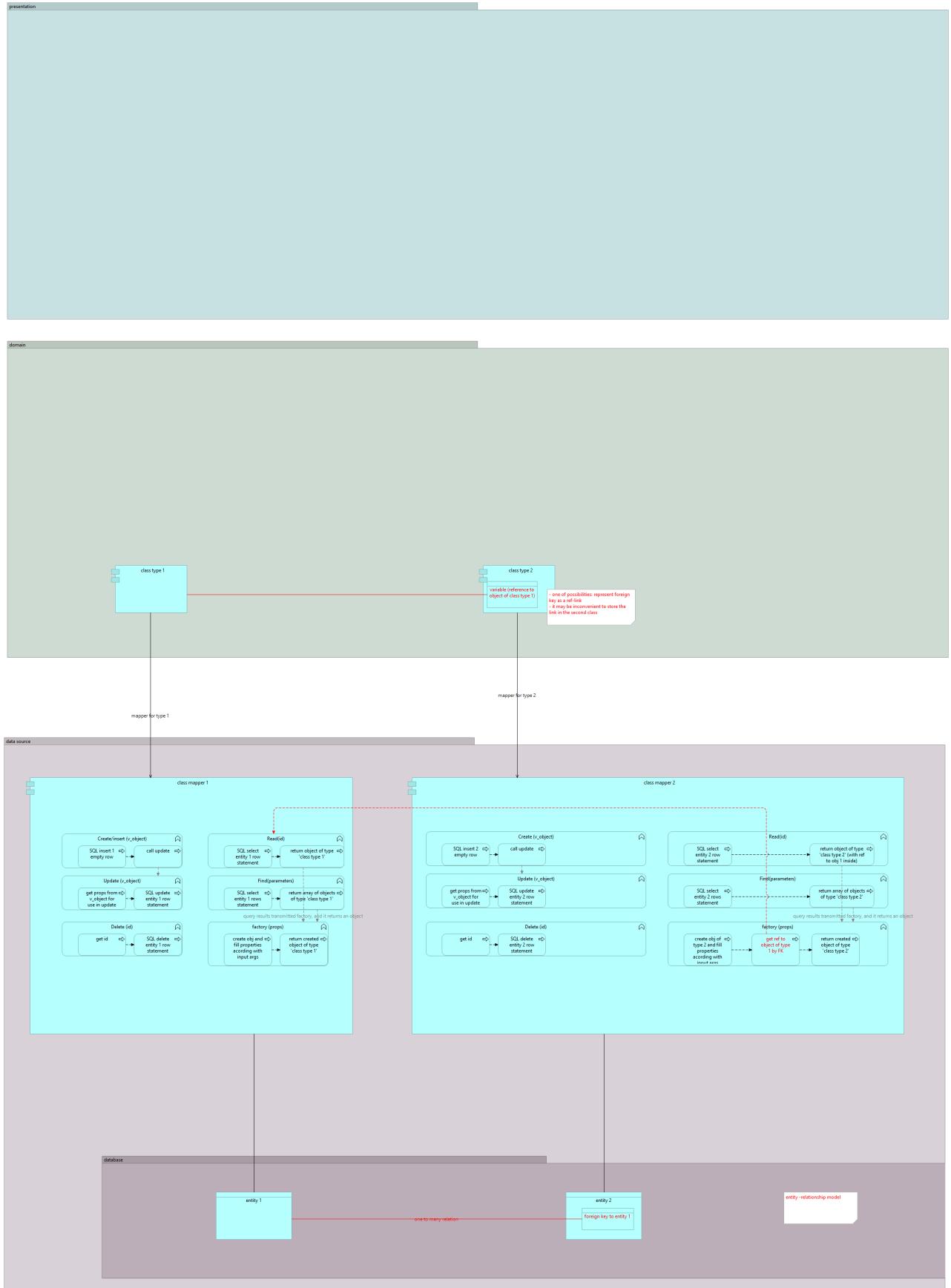
DEPENDENT MAPPING



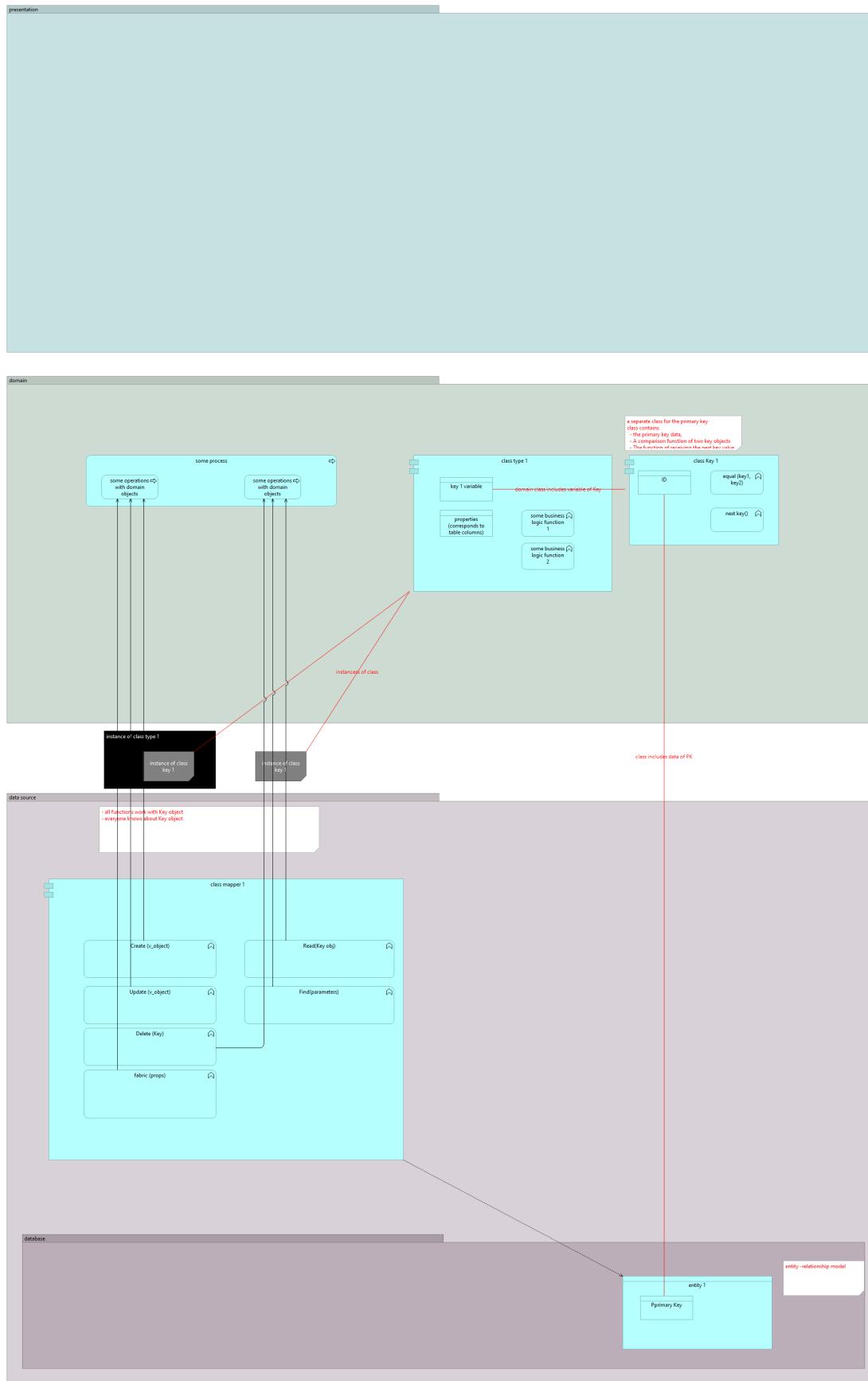
EMBEDDED VALUE



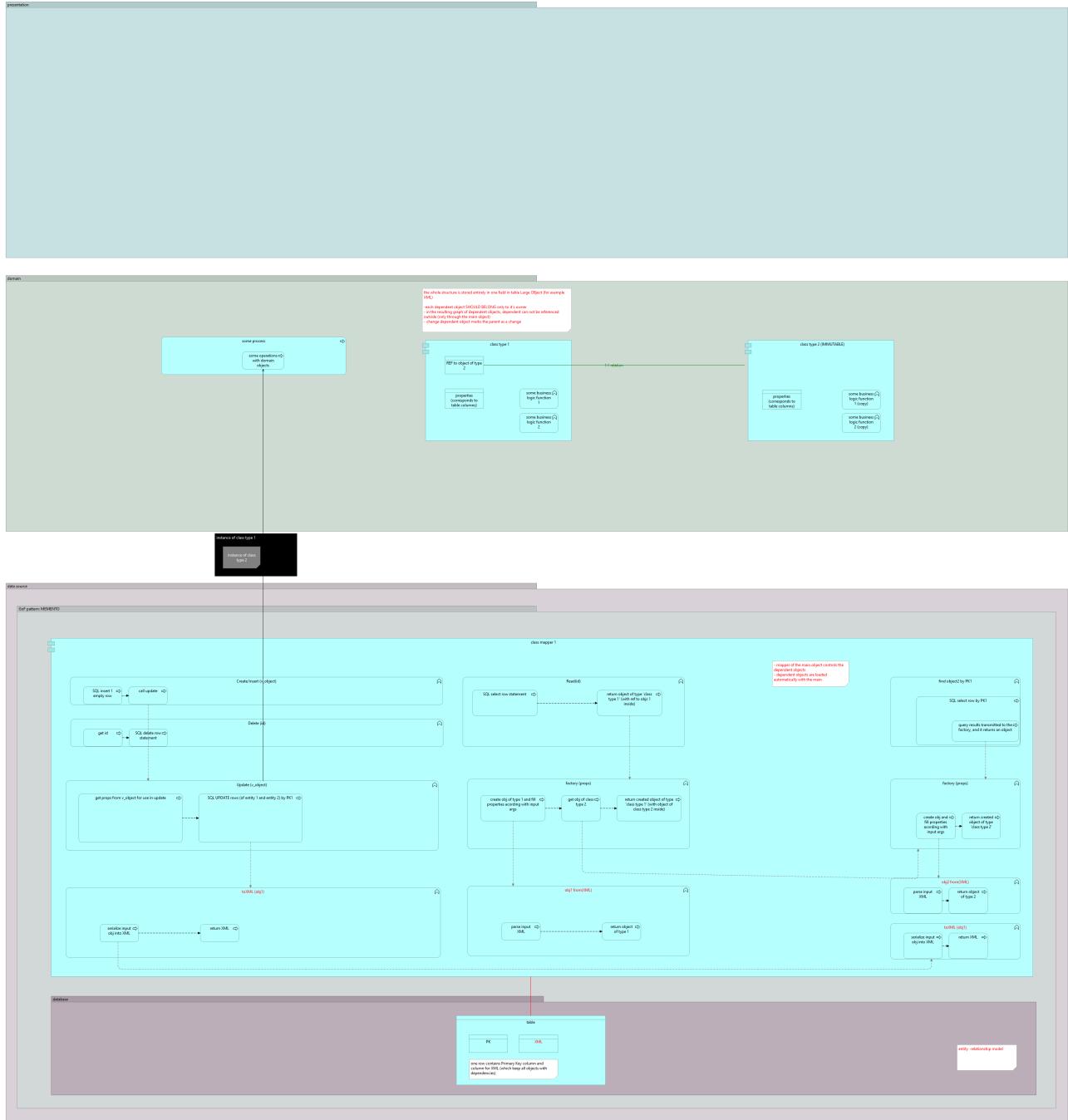
FOREIGN KEY MAPPING



IDENTITY FIELD



SERIALIZED LOB



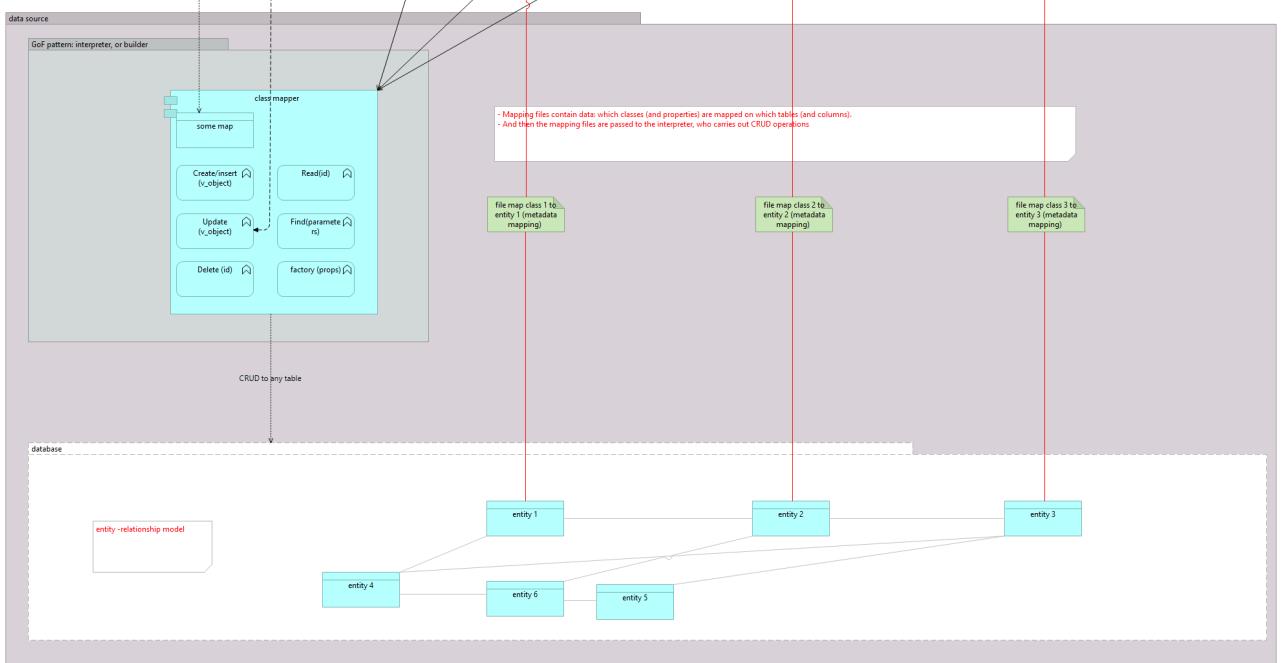
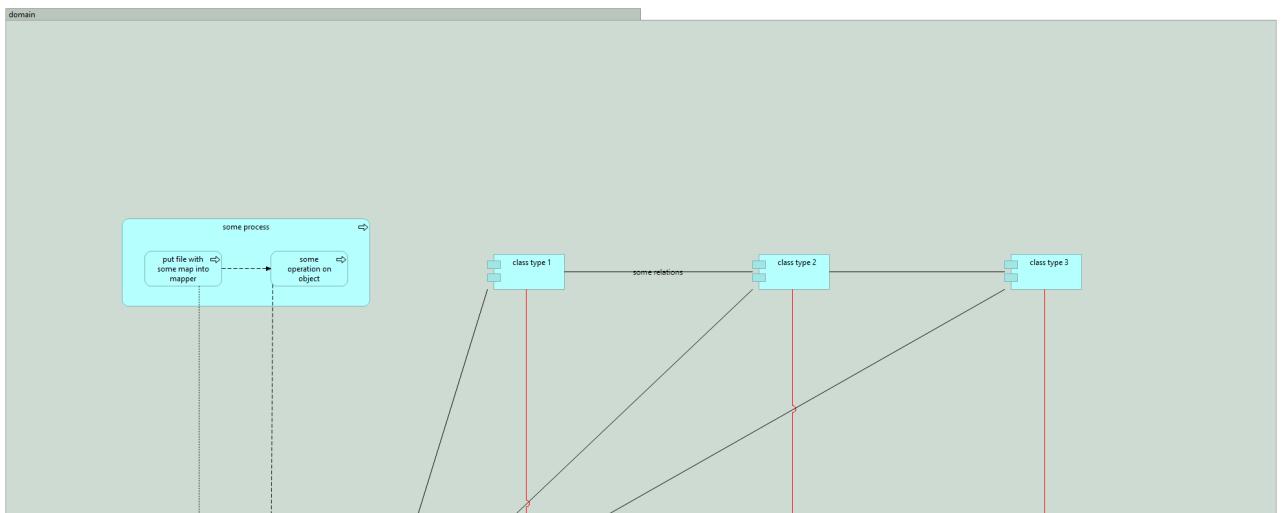
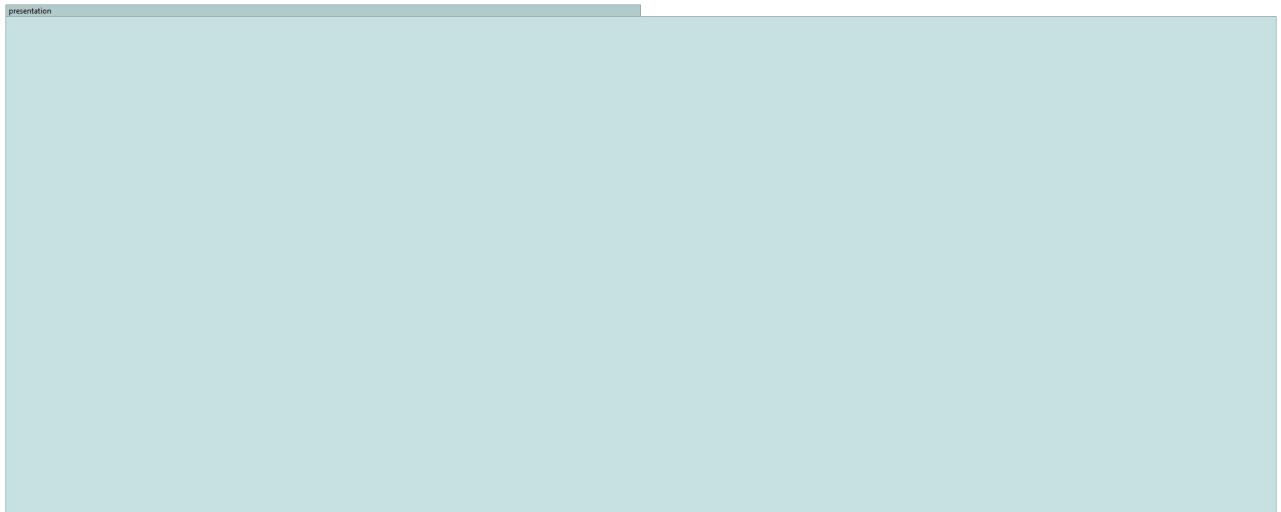
METADATA

ENTERPRISE PATTERNS

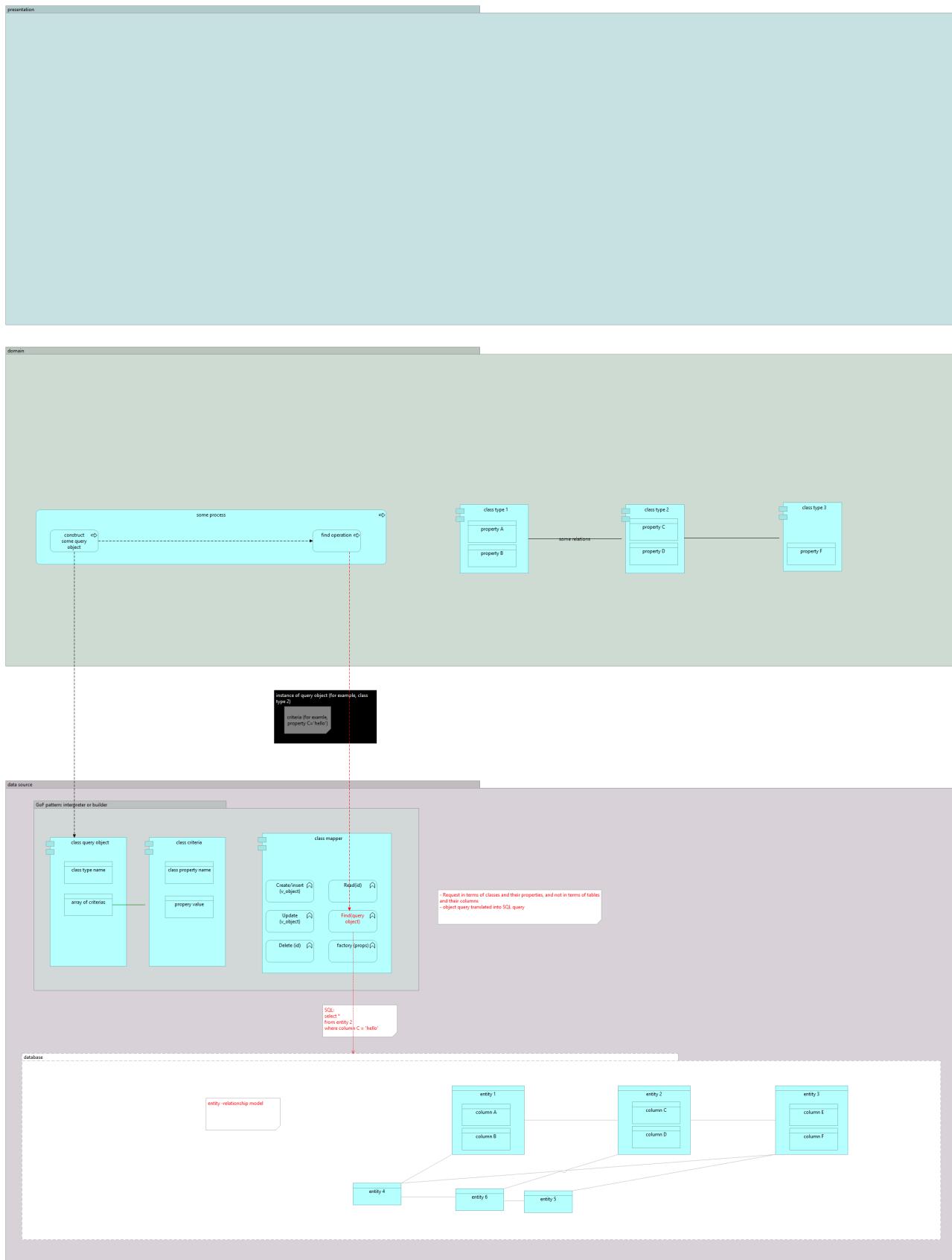
Martin Fowler



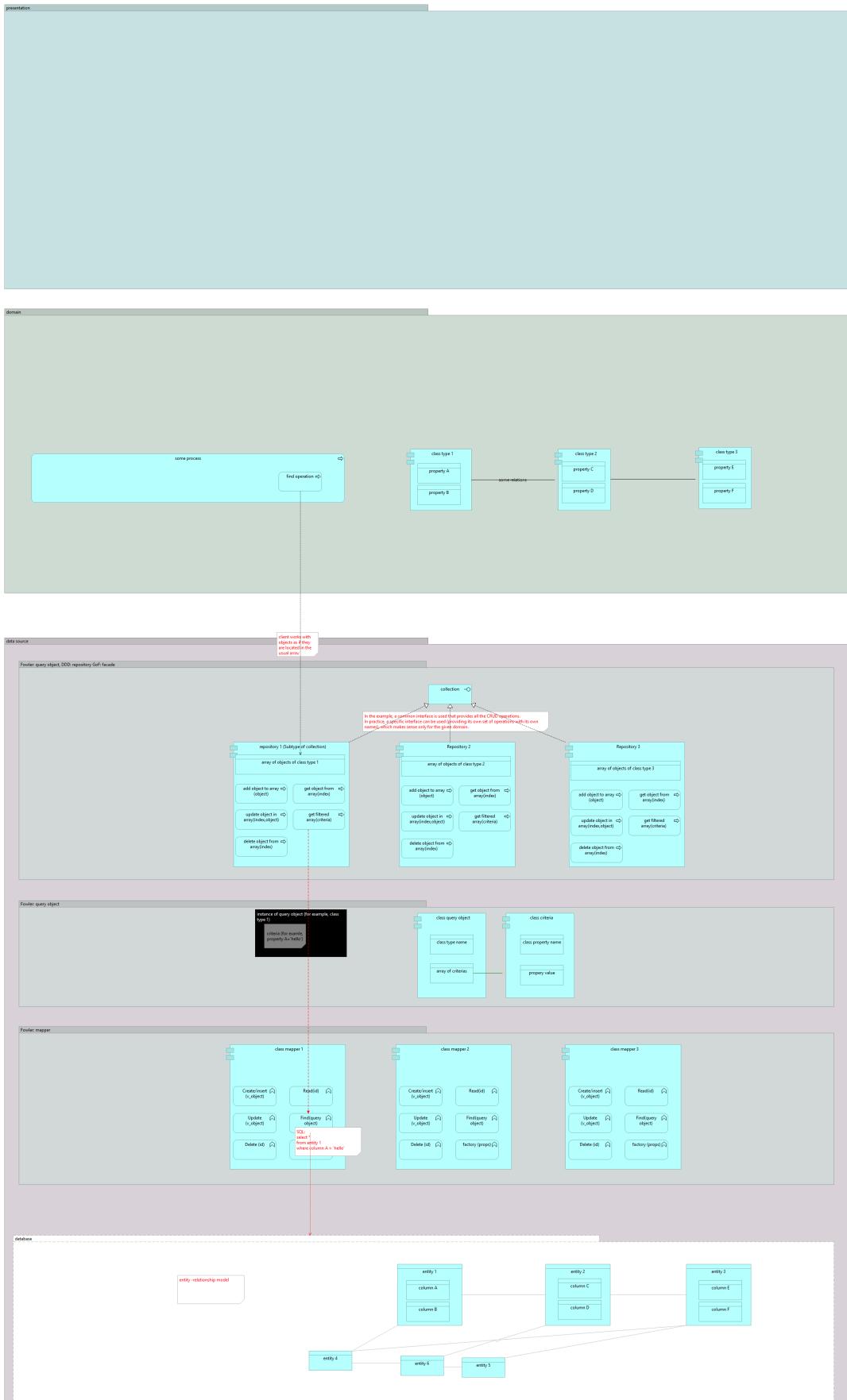
METADATA MAPPING



QUERY OBJECT



REPOSITORY



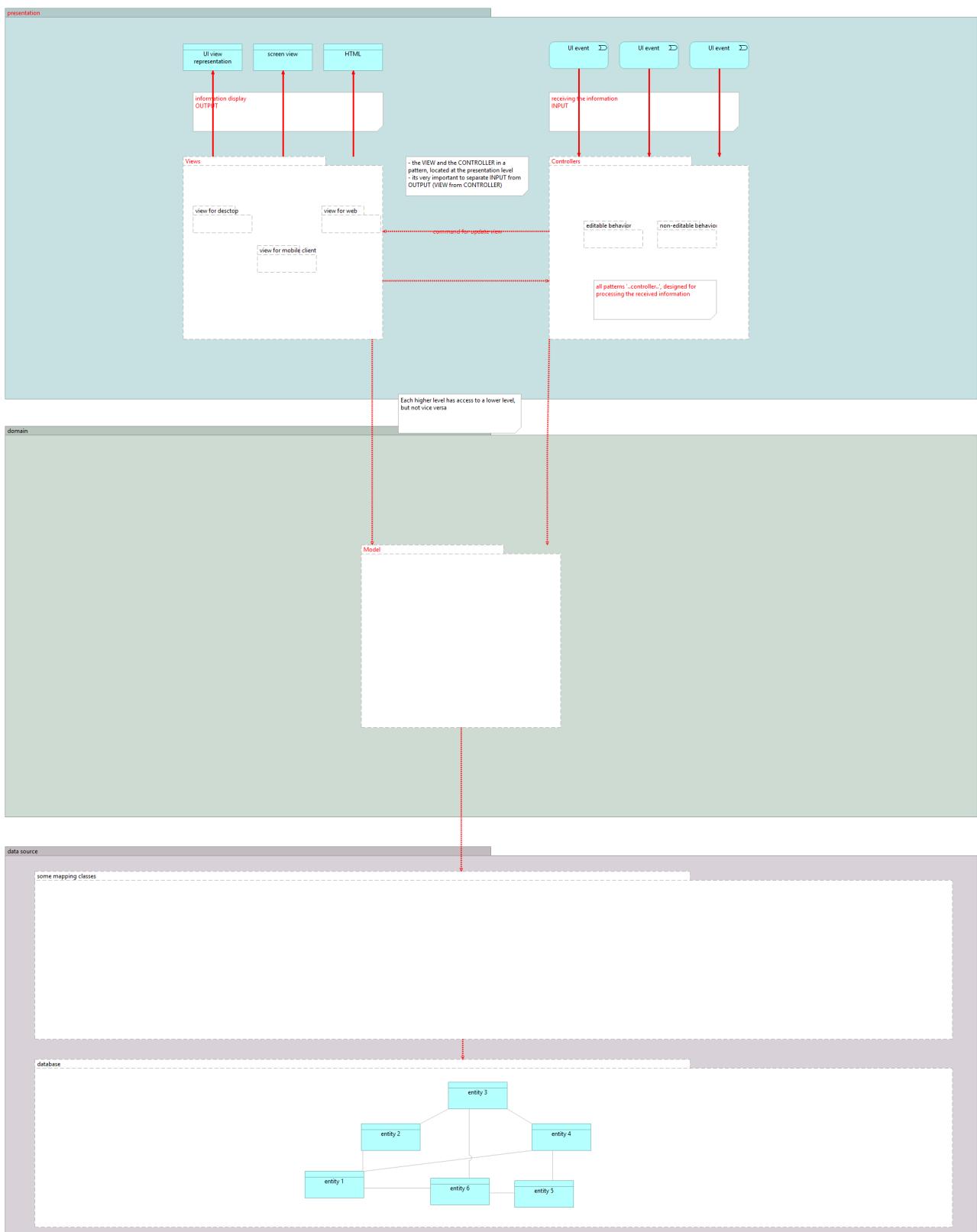
WEB REPRESENTATION CONTROLLER

ENTERPRISE PATTERNS

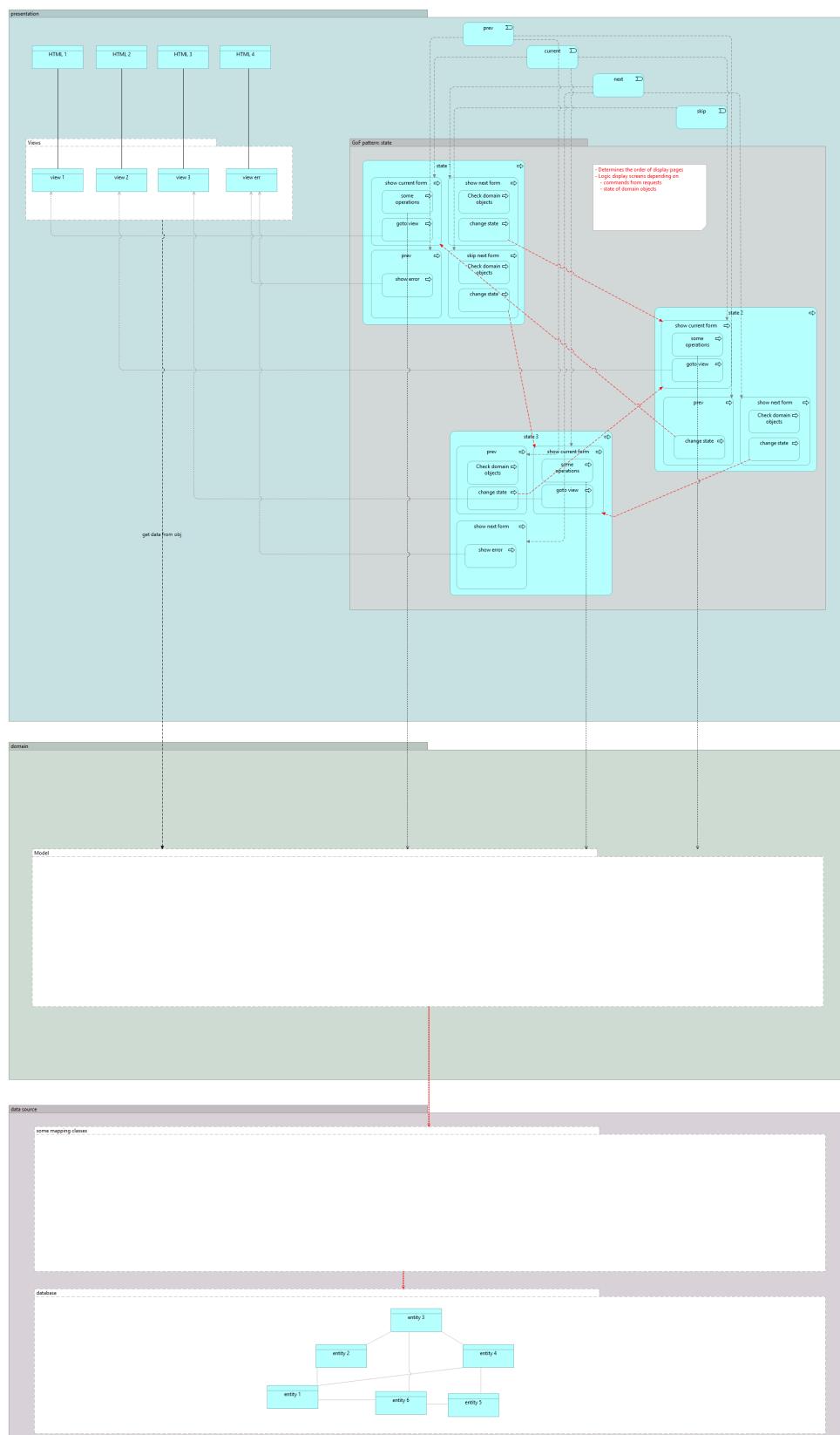
Martin Fowler



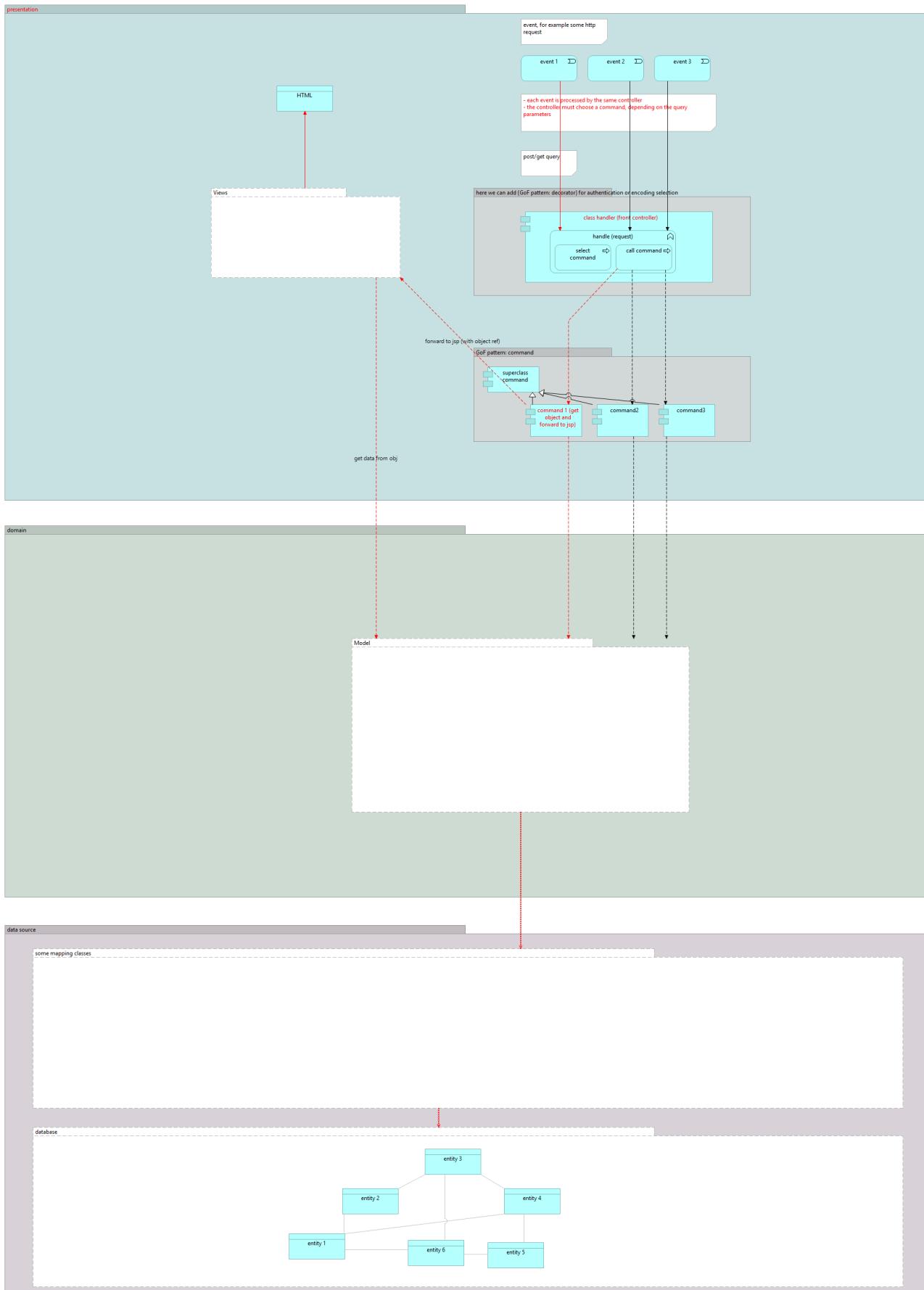
MODEL VIEW CONTROLLER



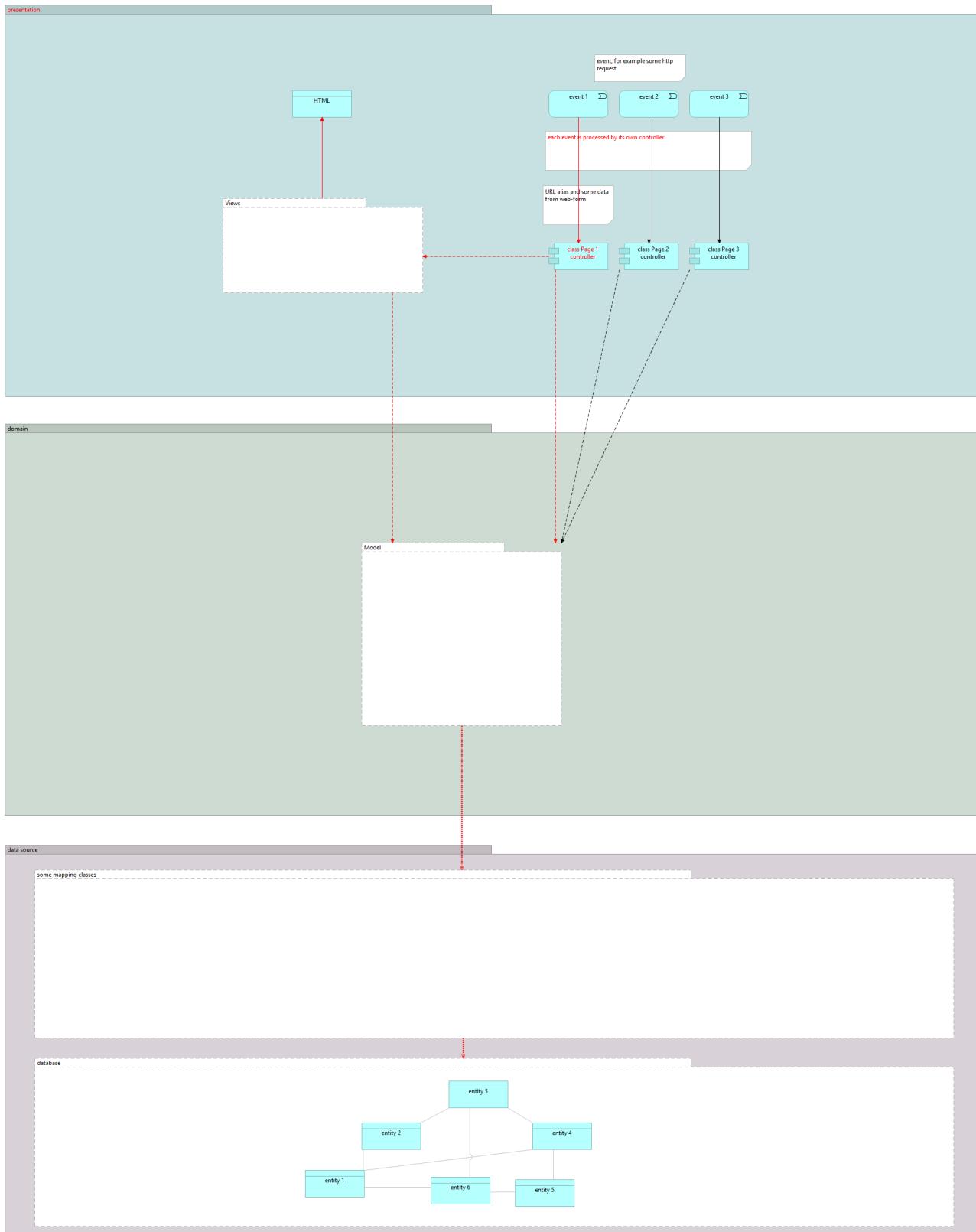
APPLICATION CONTROLLER



FRONT CONTROLLER



PAGE CONTROLLER



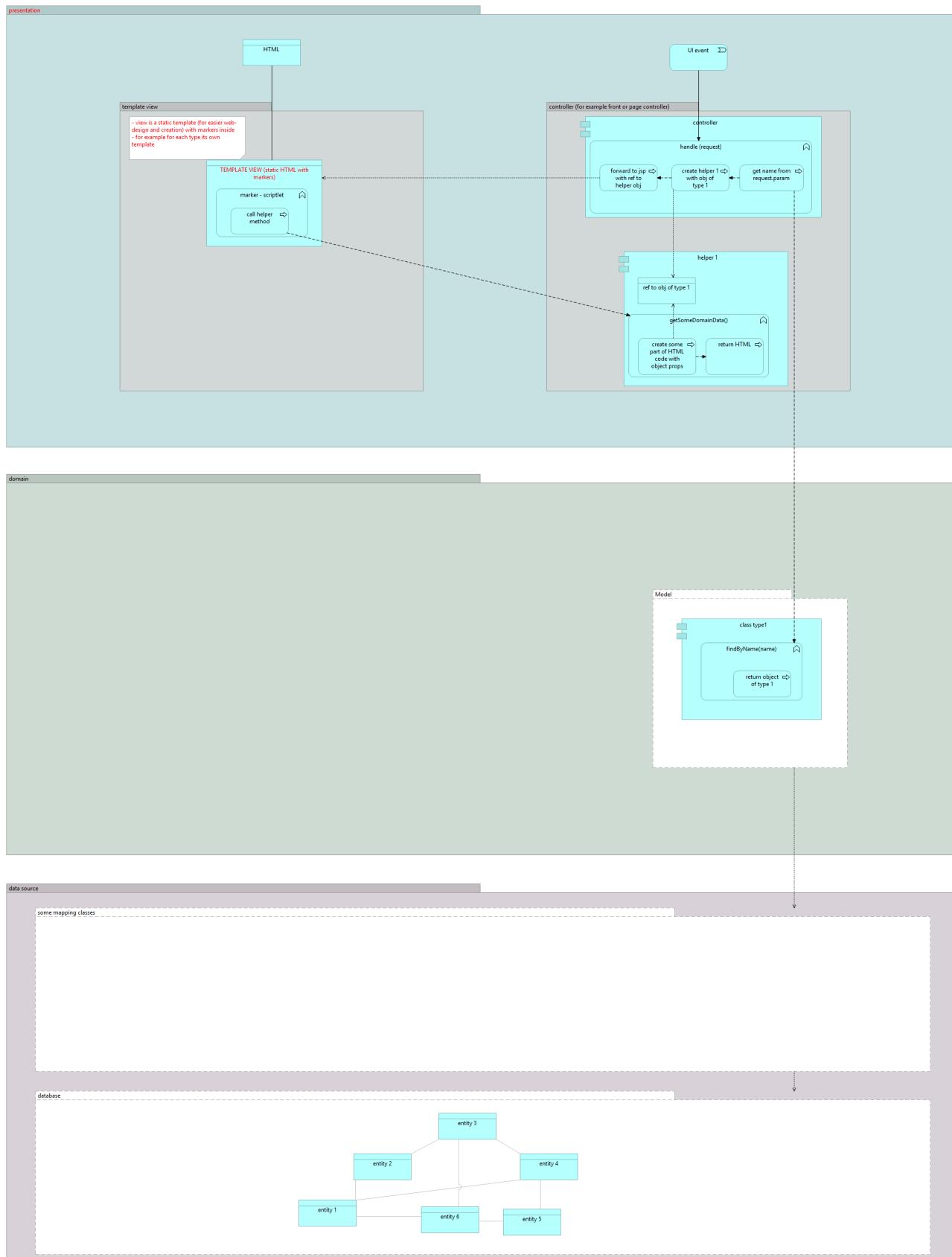
WEB REPRESENTATION VIEW

ENTERPRISE PATTERNS

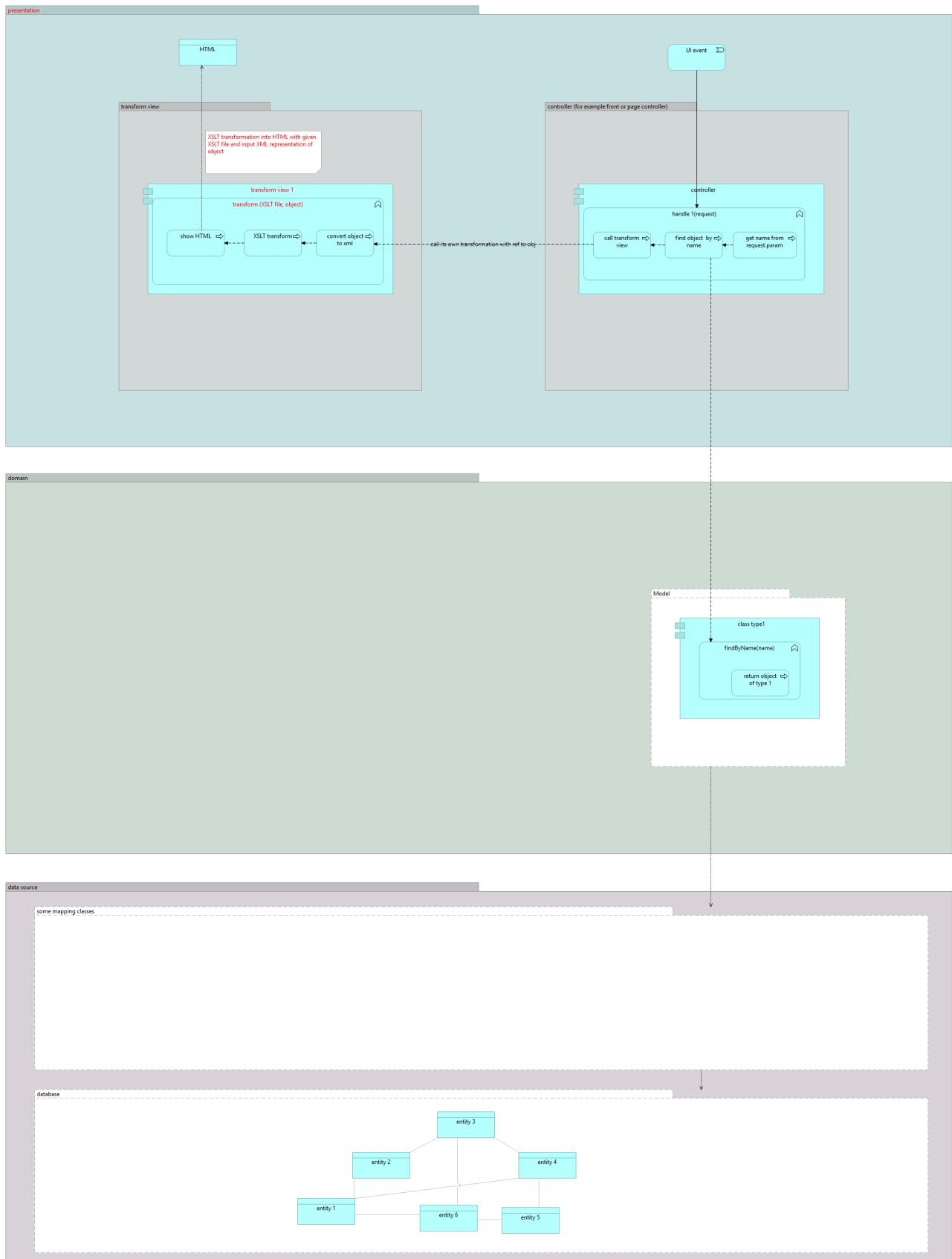
Martin Fowler



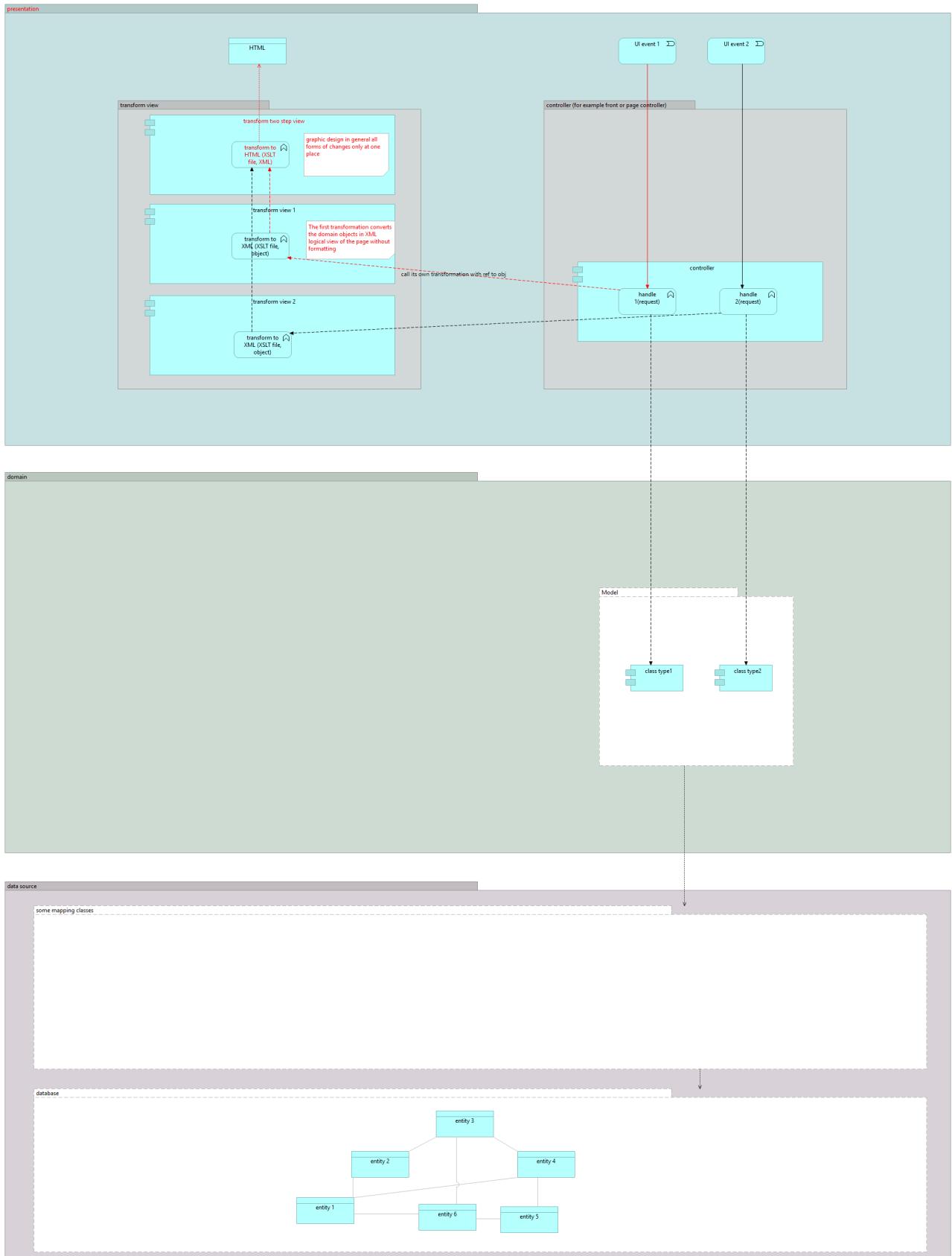
TEMPLATE VIEW



TRANSFORM VIEW



TWO STEP VIEW



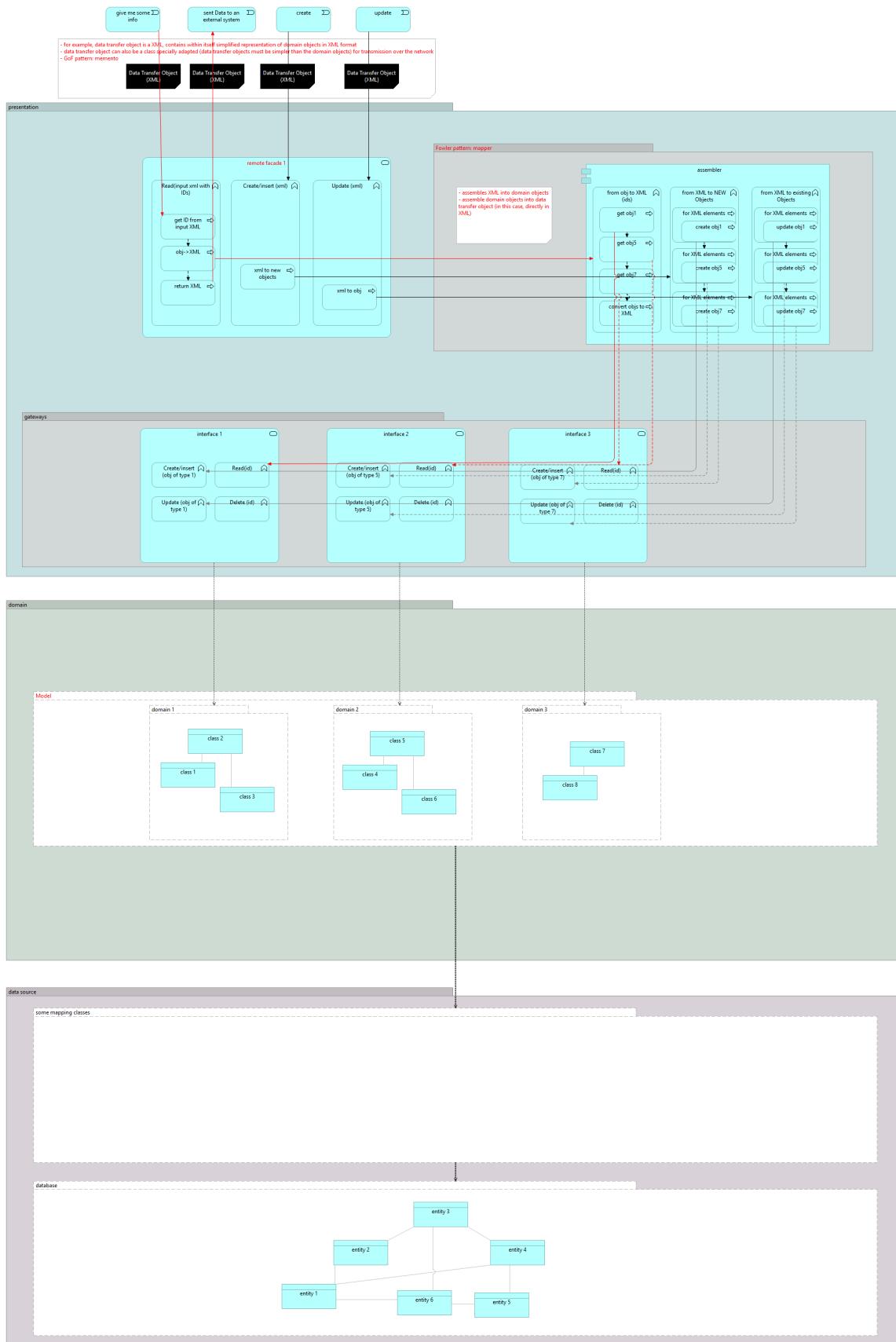
DISTRIBUTED PROCESSING

ENTERPRISE PATTERNS

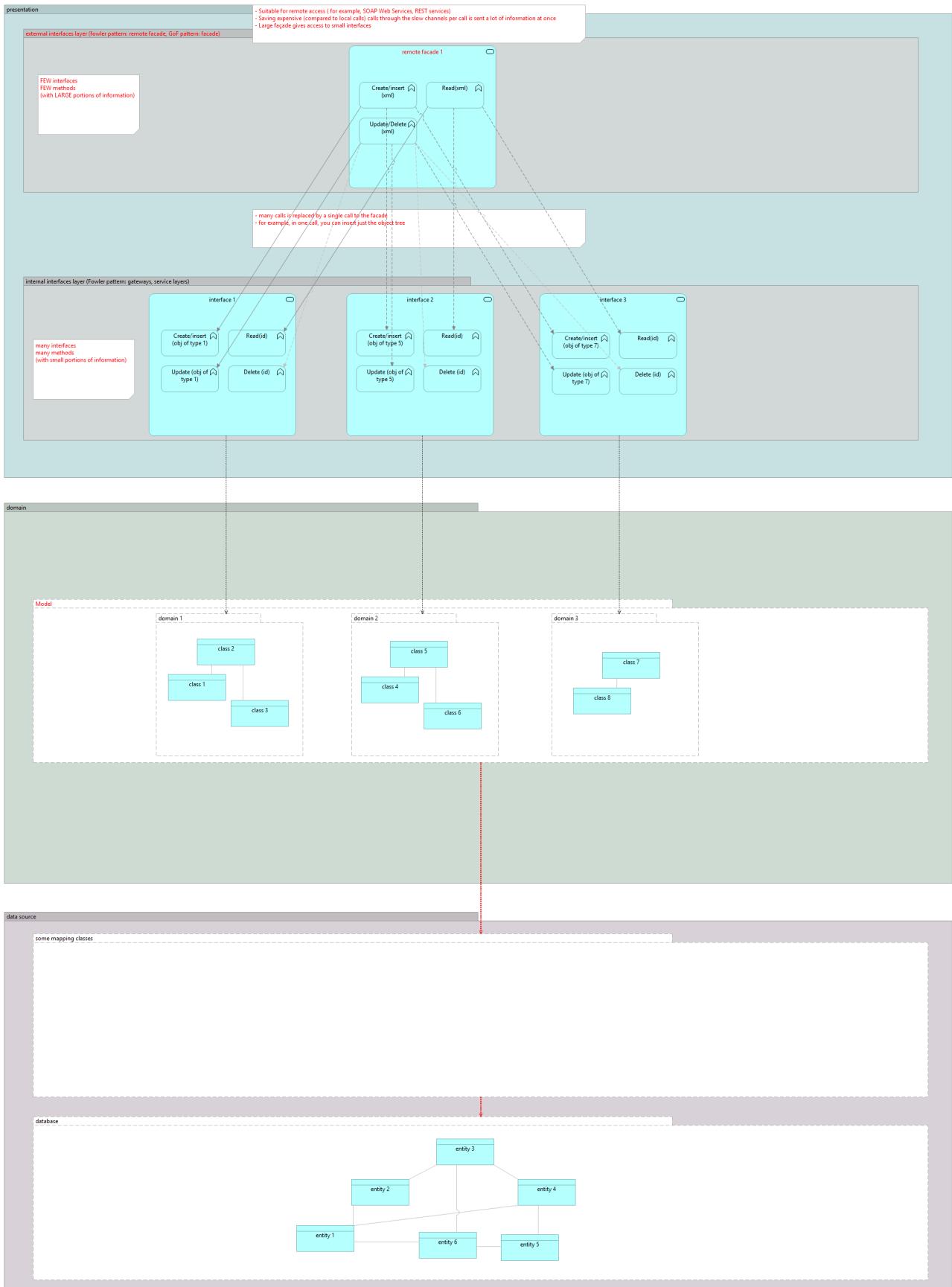
Martin Fowler



DATA TRANSFER OBJECT



REMOTE FAÇADE



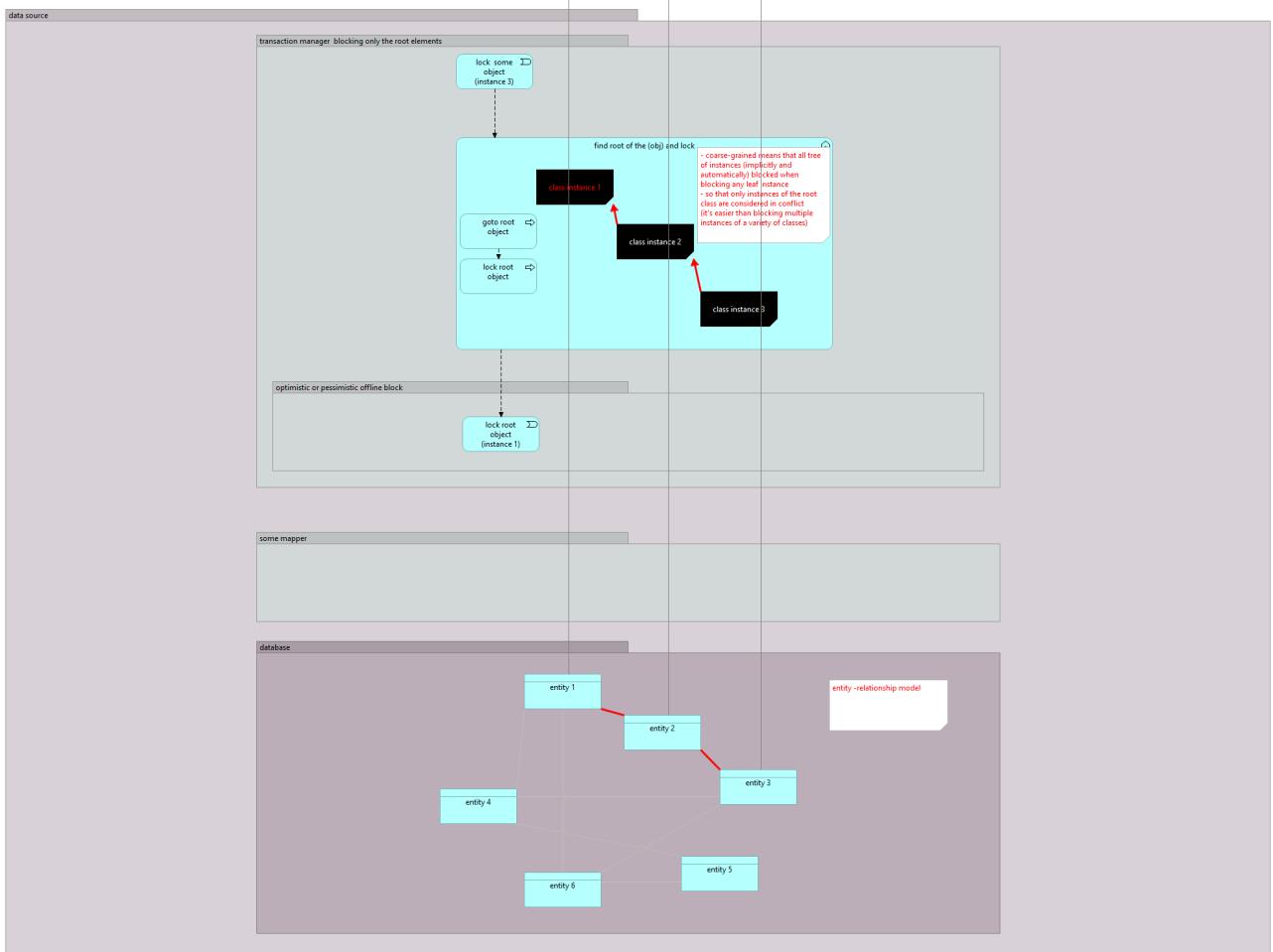
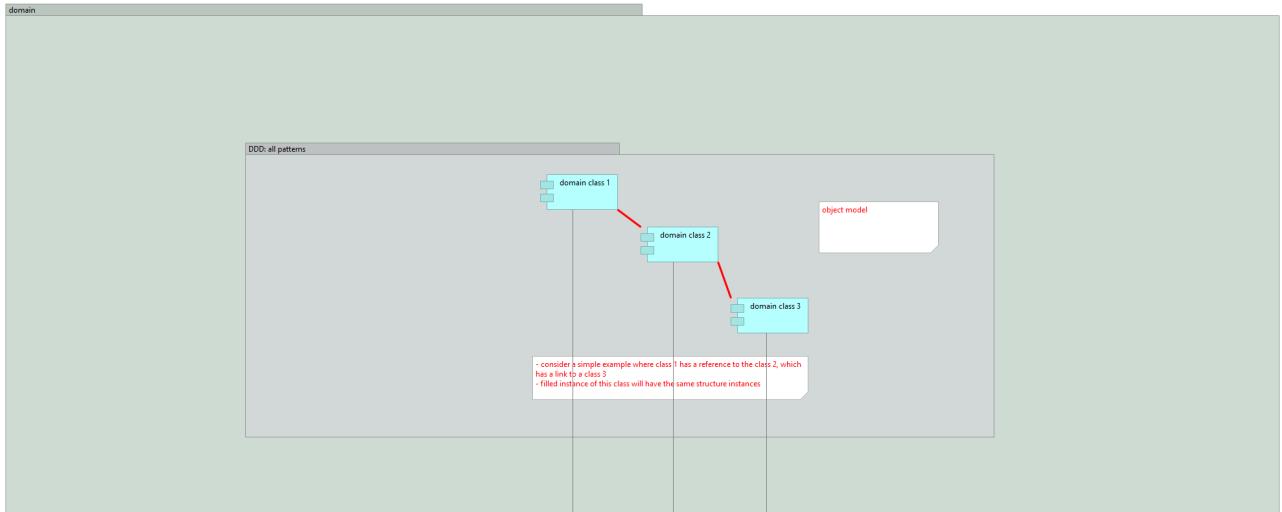
PARALLEL PROCESSING

ENTERPRISE PATTERNS

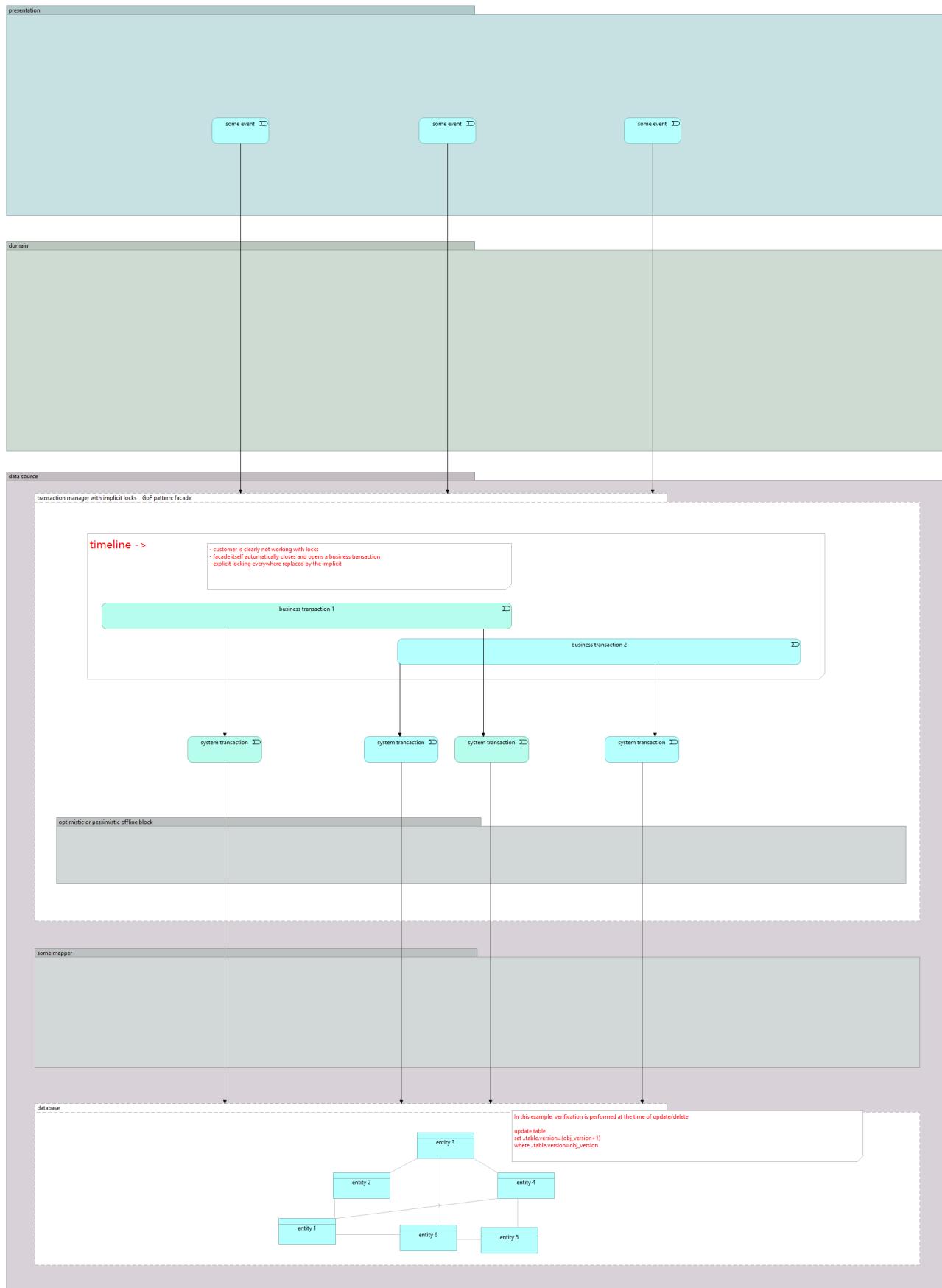
Martin Fowler



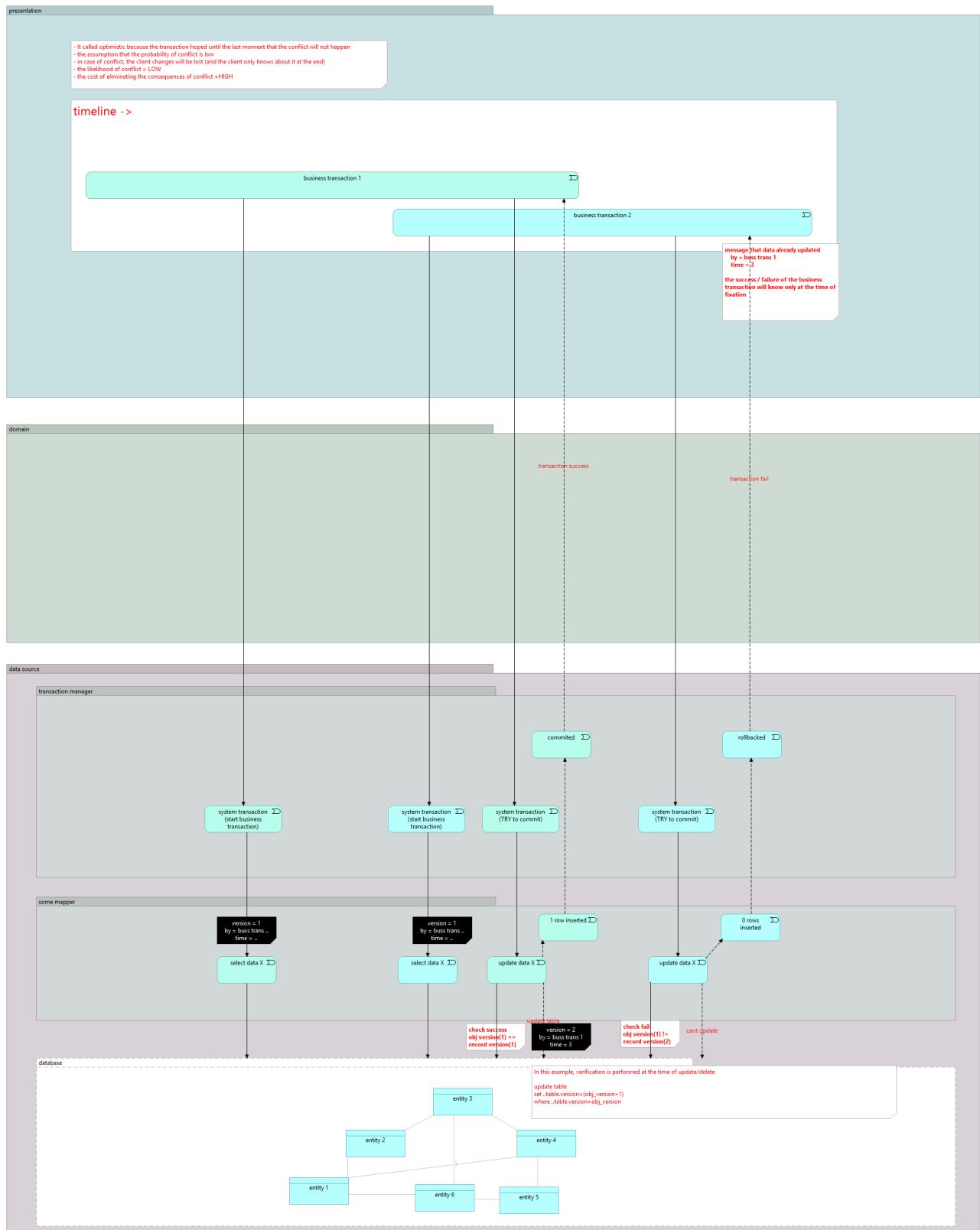
COARSE-GRAINED LOCK



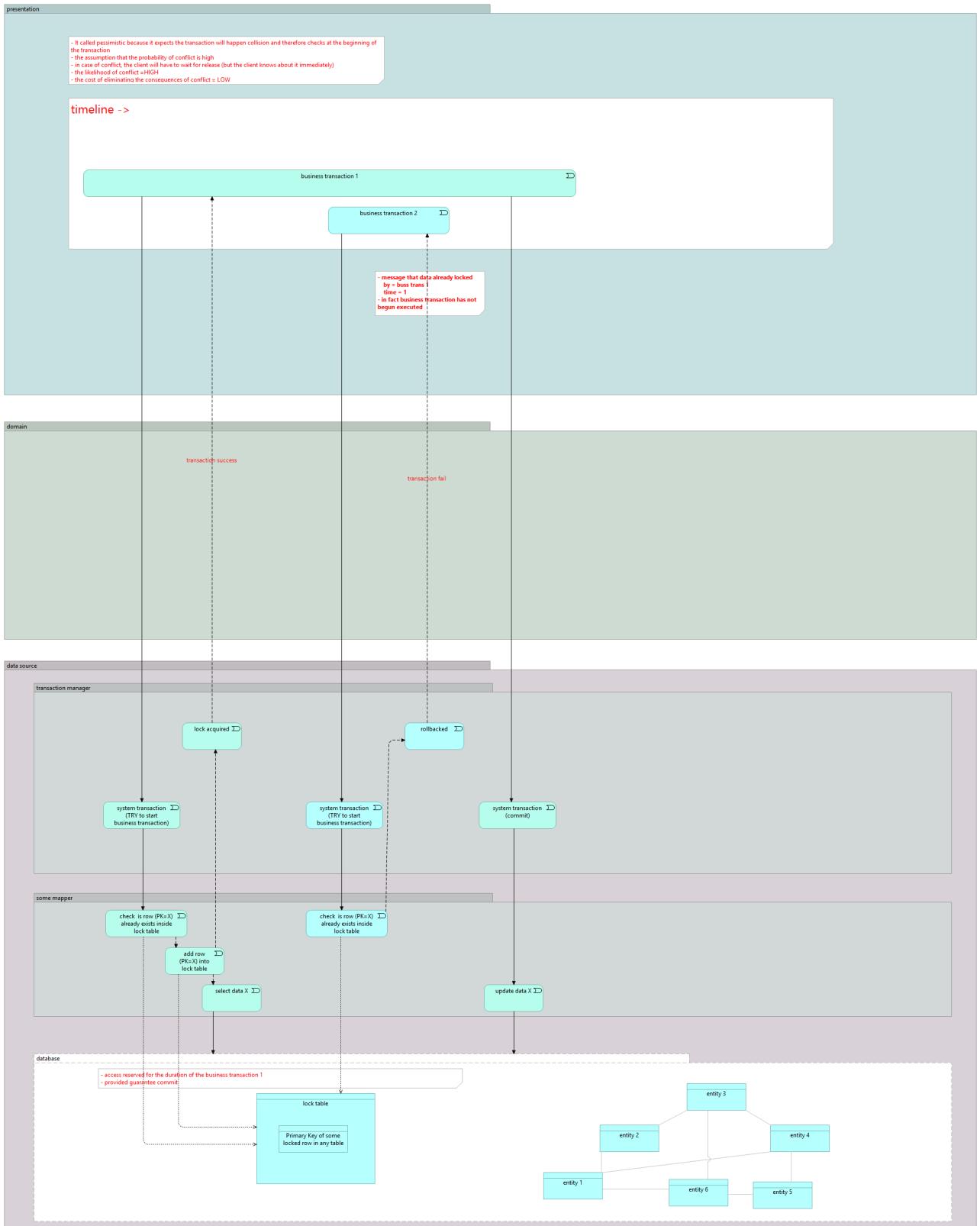
IMPLICIT LOCK



OPTIMISTIC OFFLINE LOCK



PESSIMISTIC OFFLINE LOCK



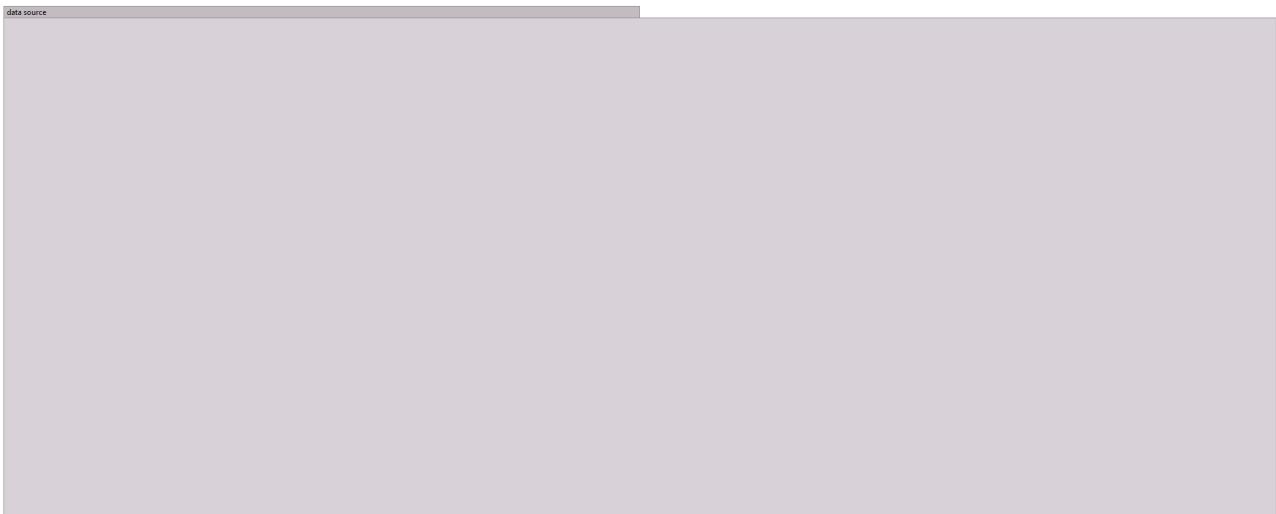
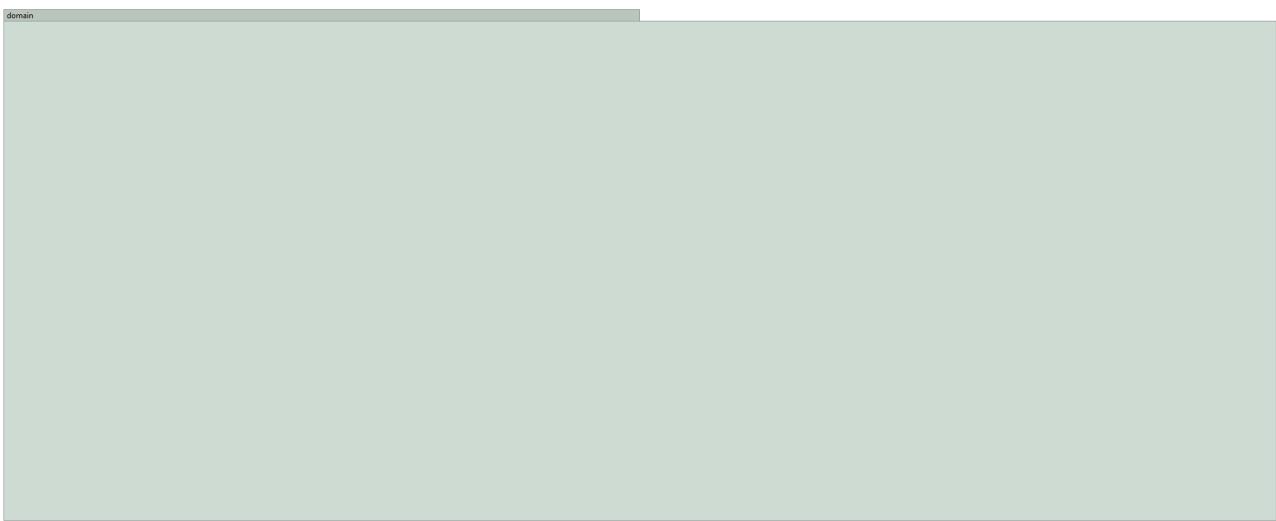
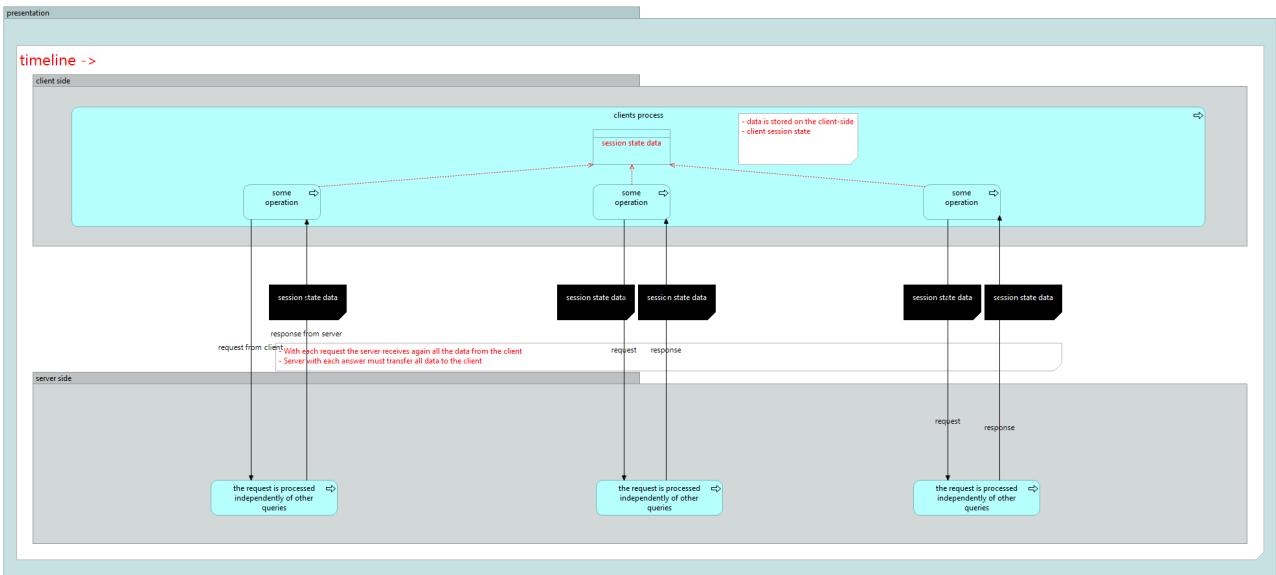
SESSION STATE

ENTERPRISE PATTERNS

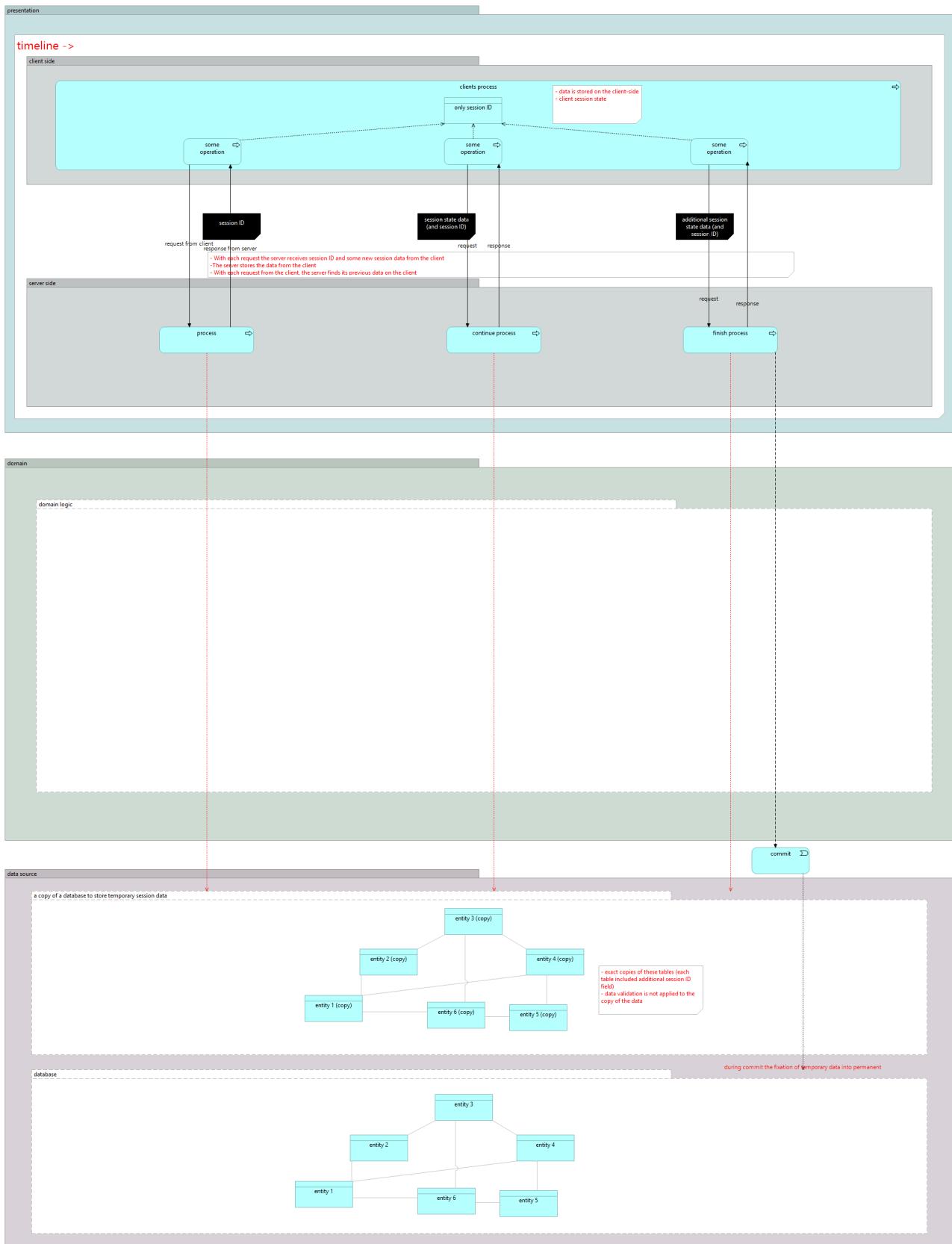
Martin Fowler



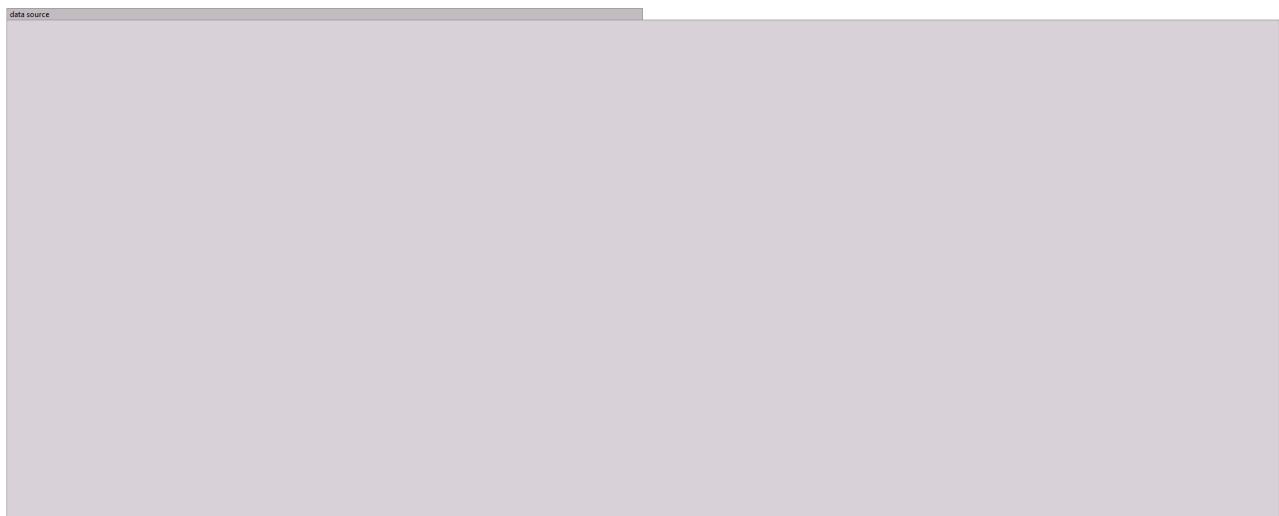
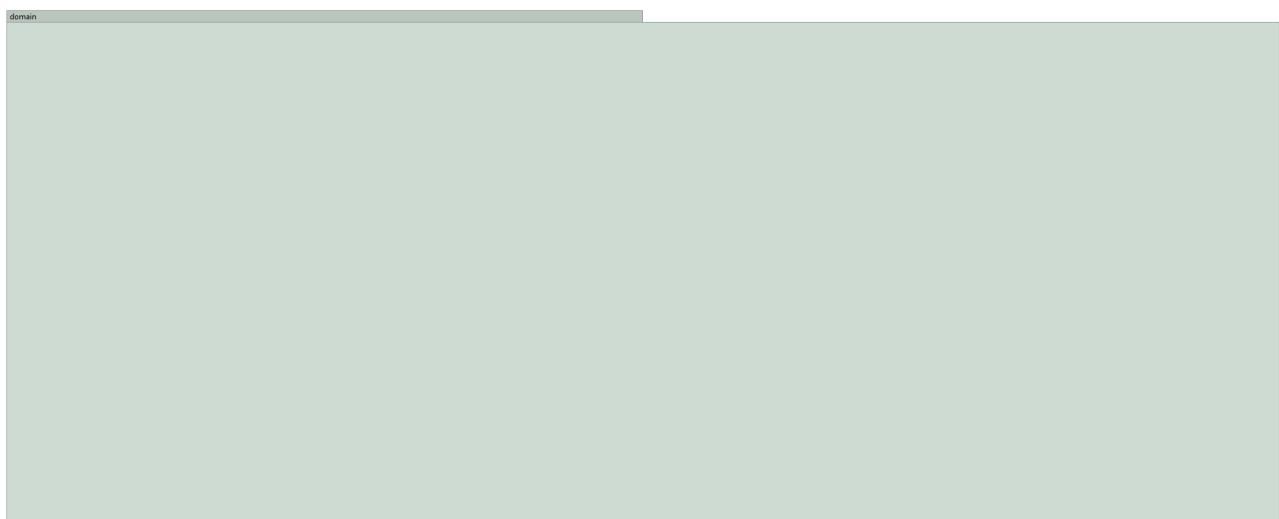
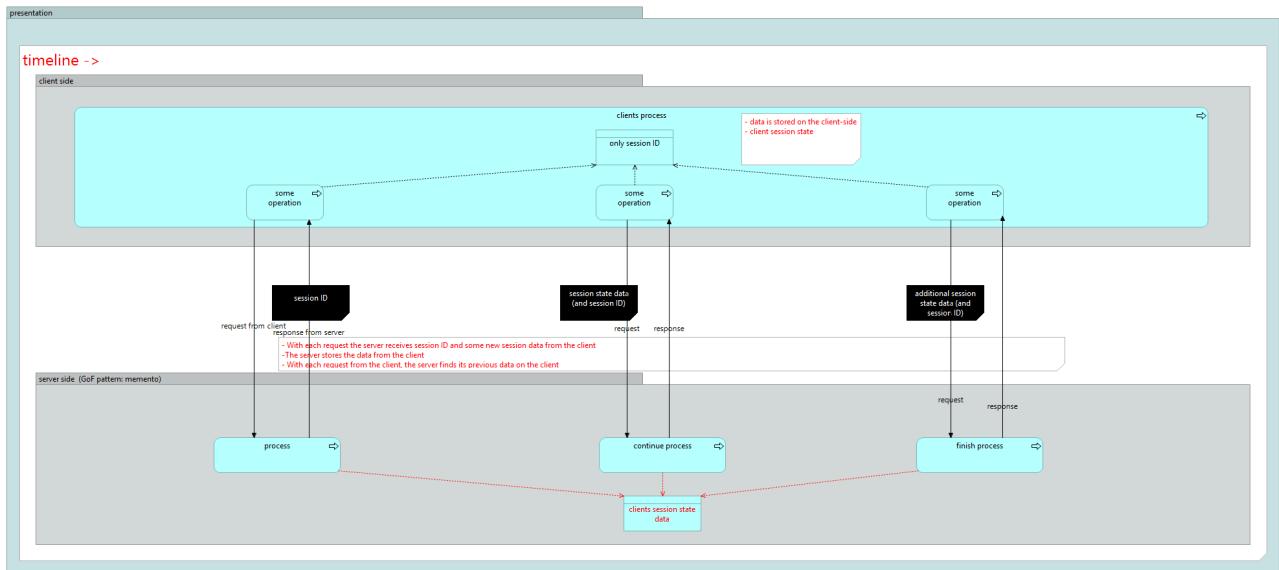
CLIENT SESSION STATE



DATABASE SESSION STATE



SERVER SESSION STATE



COMMON PATTERNS

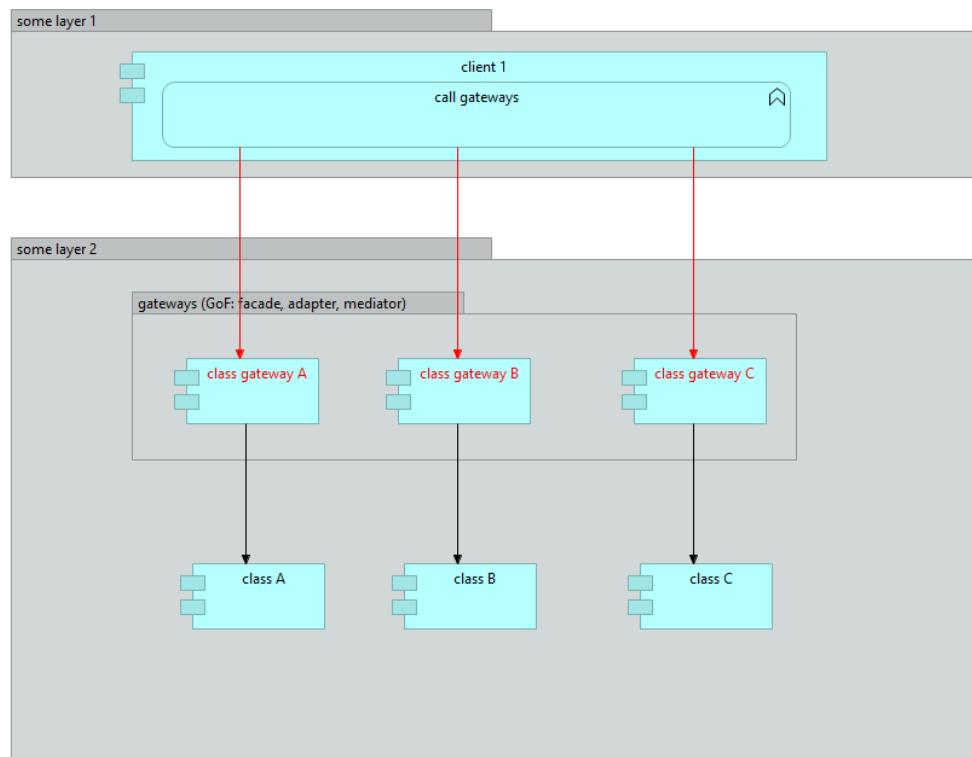
ENTERPRISE PATTERNS

Martin Fowler

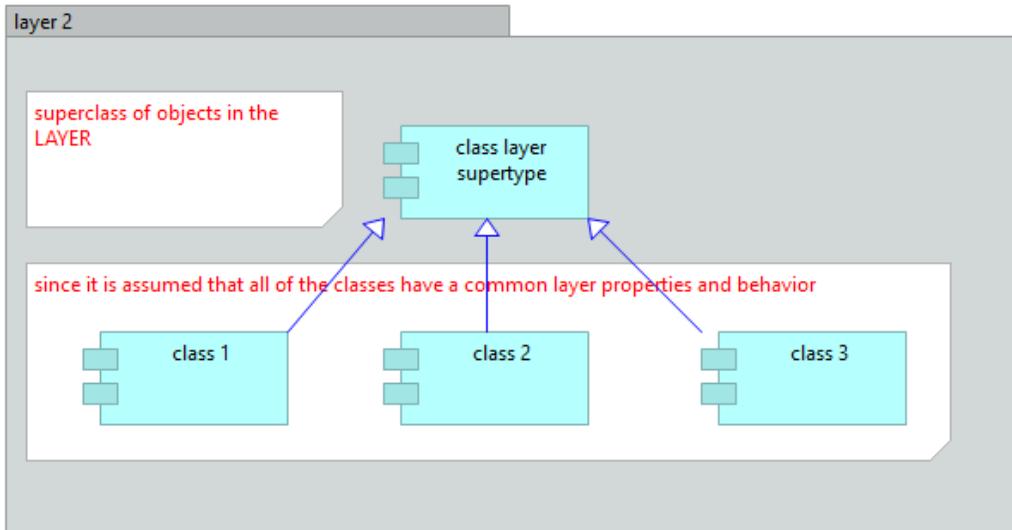


GATEWAY

- 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



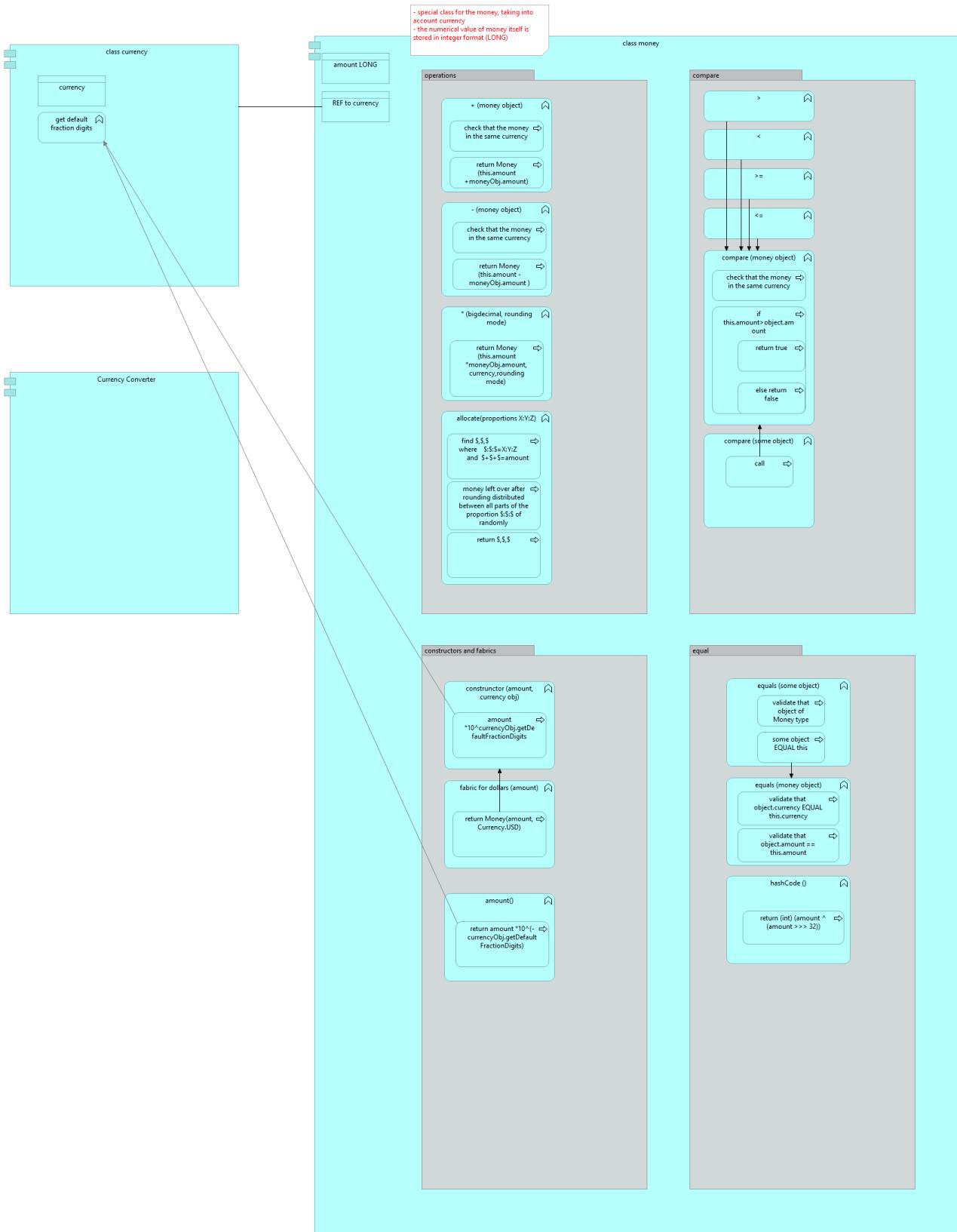
LAYER SUPERTYPE



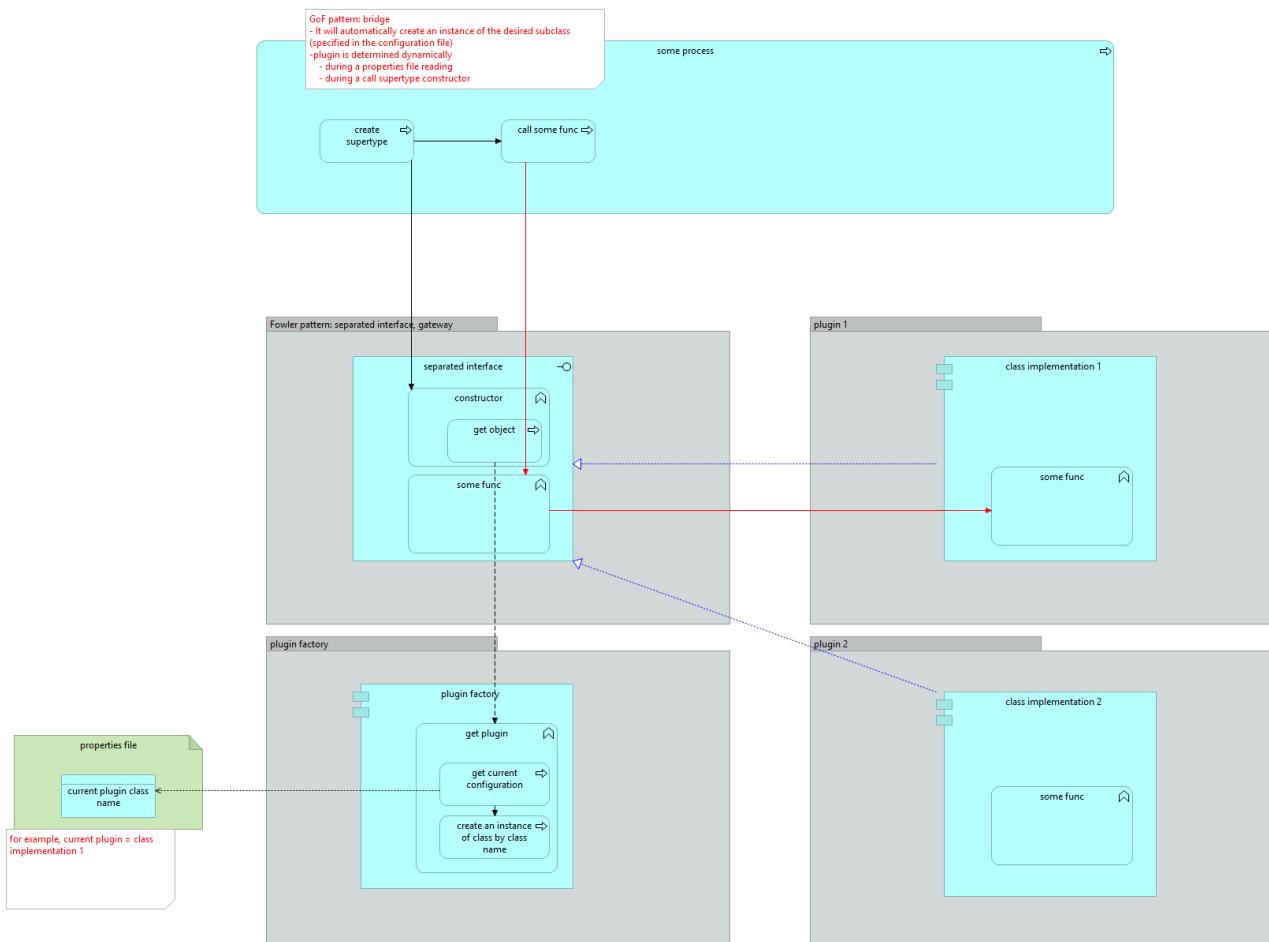
MAPPER



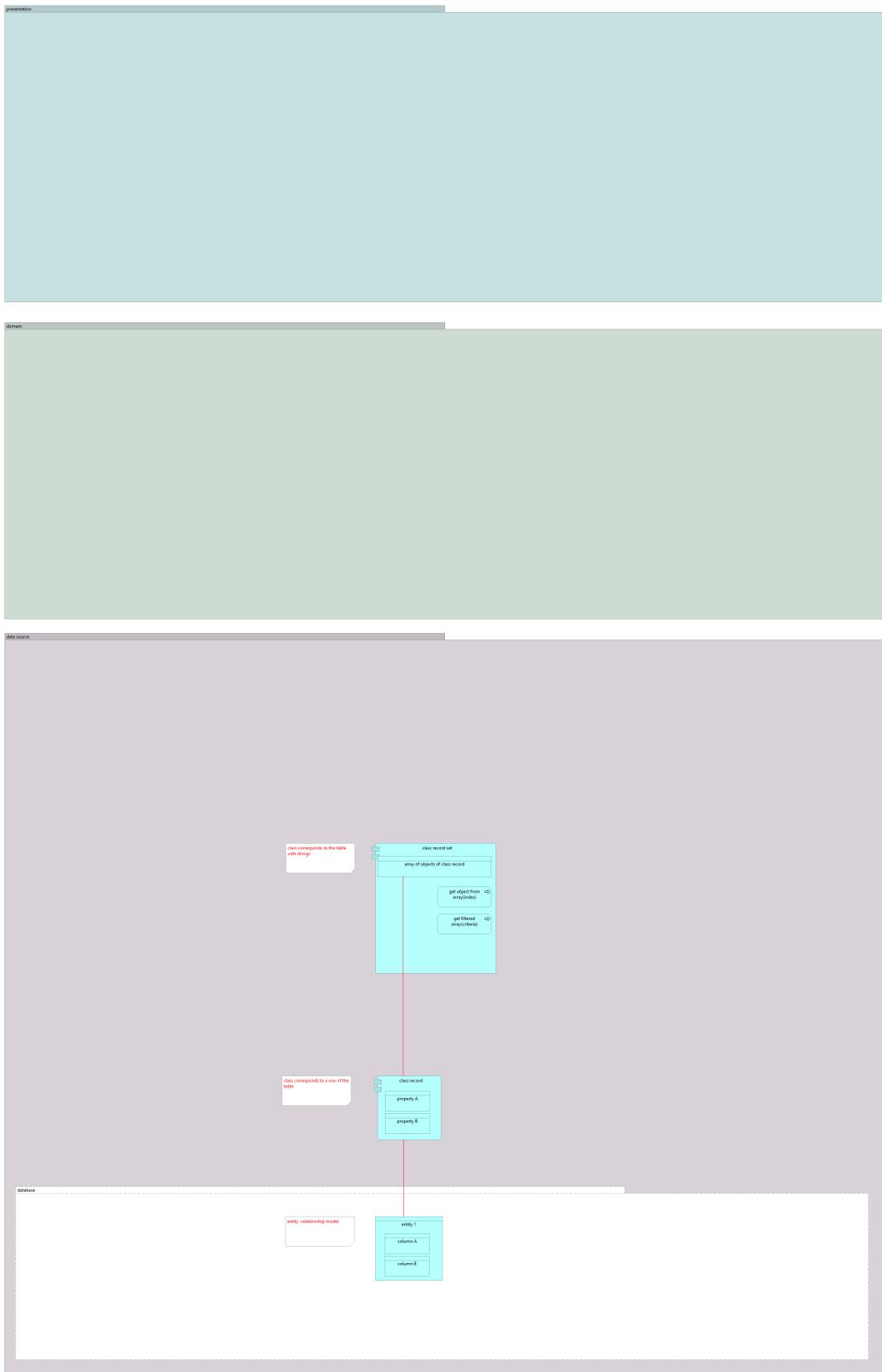
MONEY



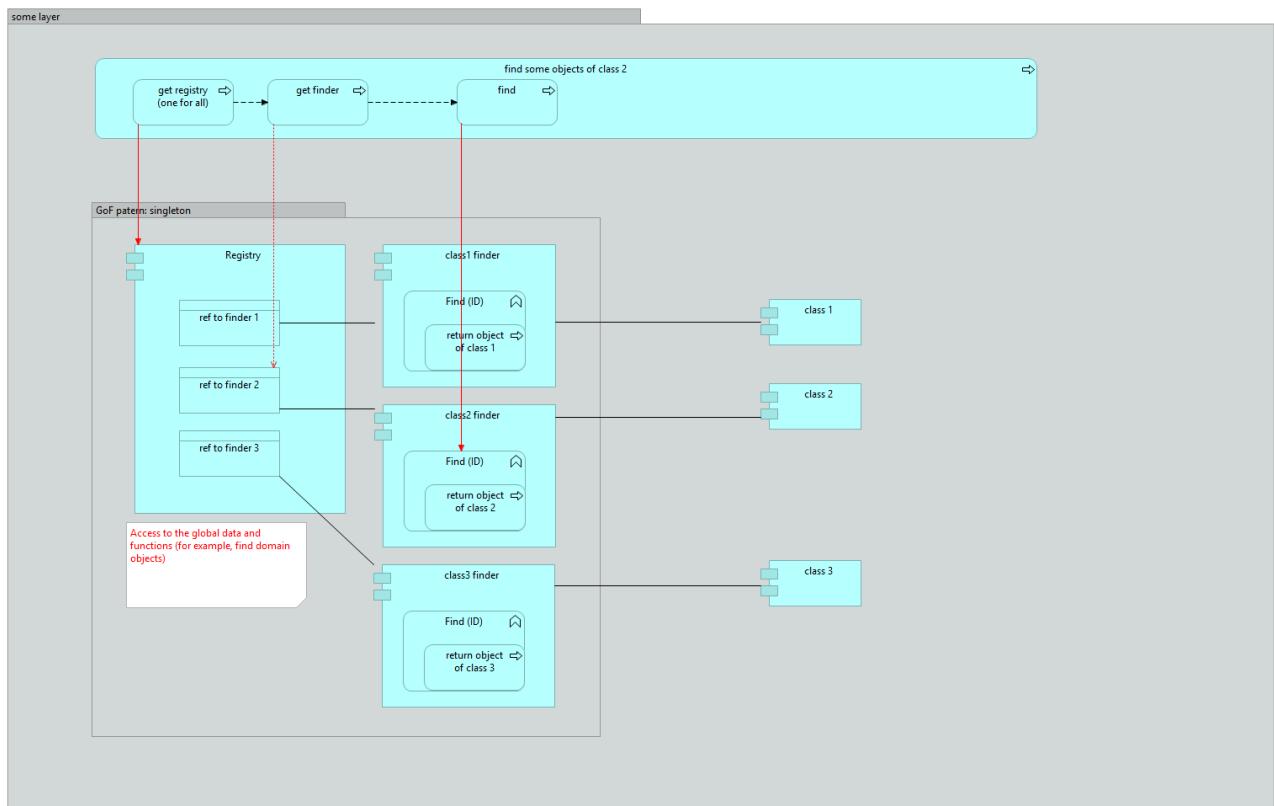
PLUGIN



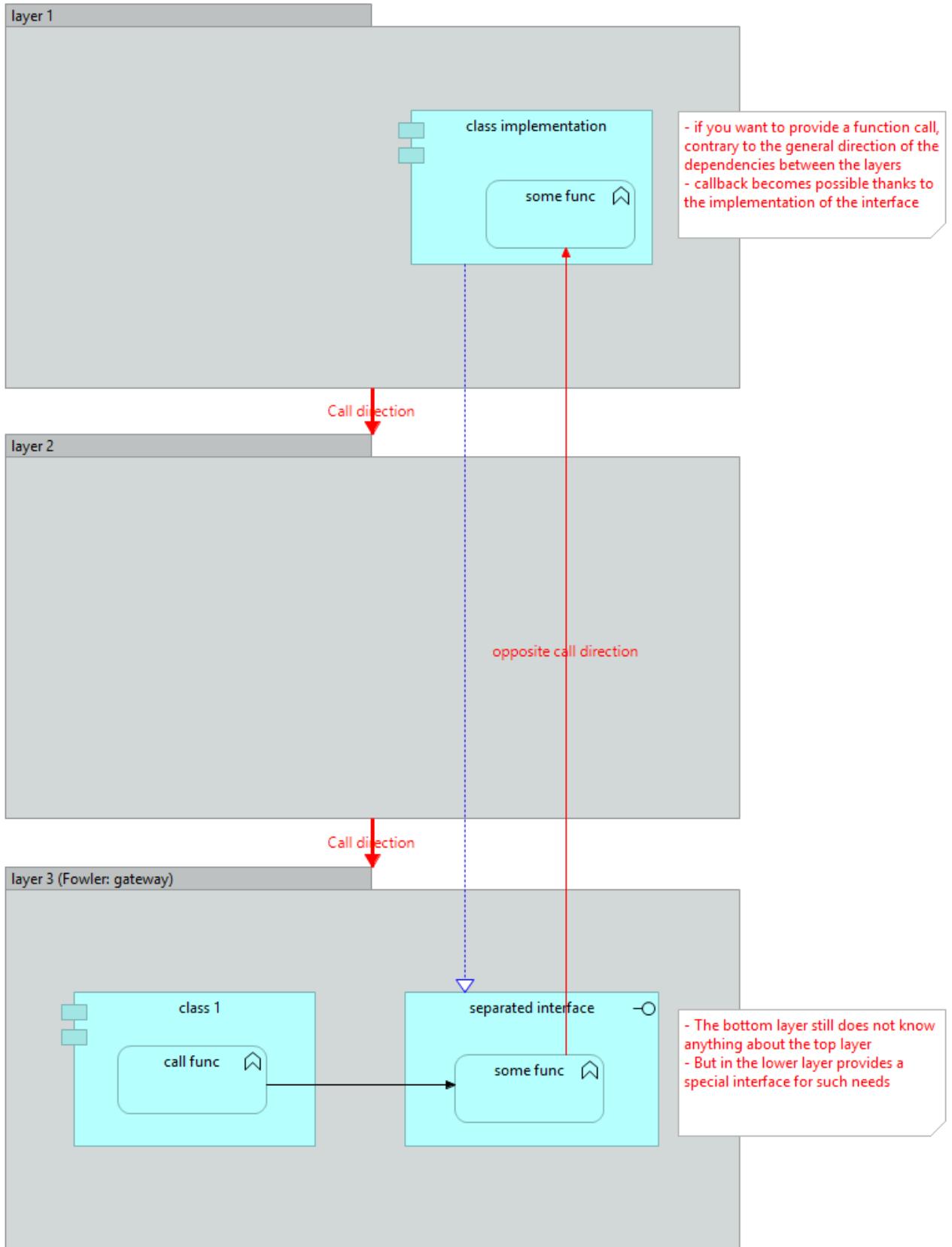
RECORD SET



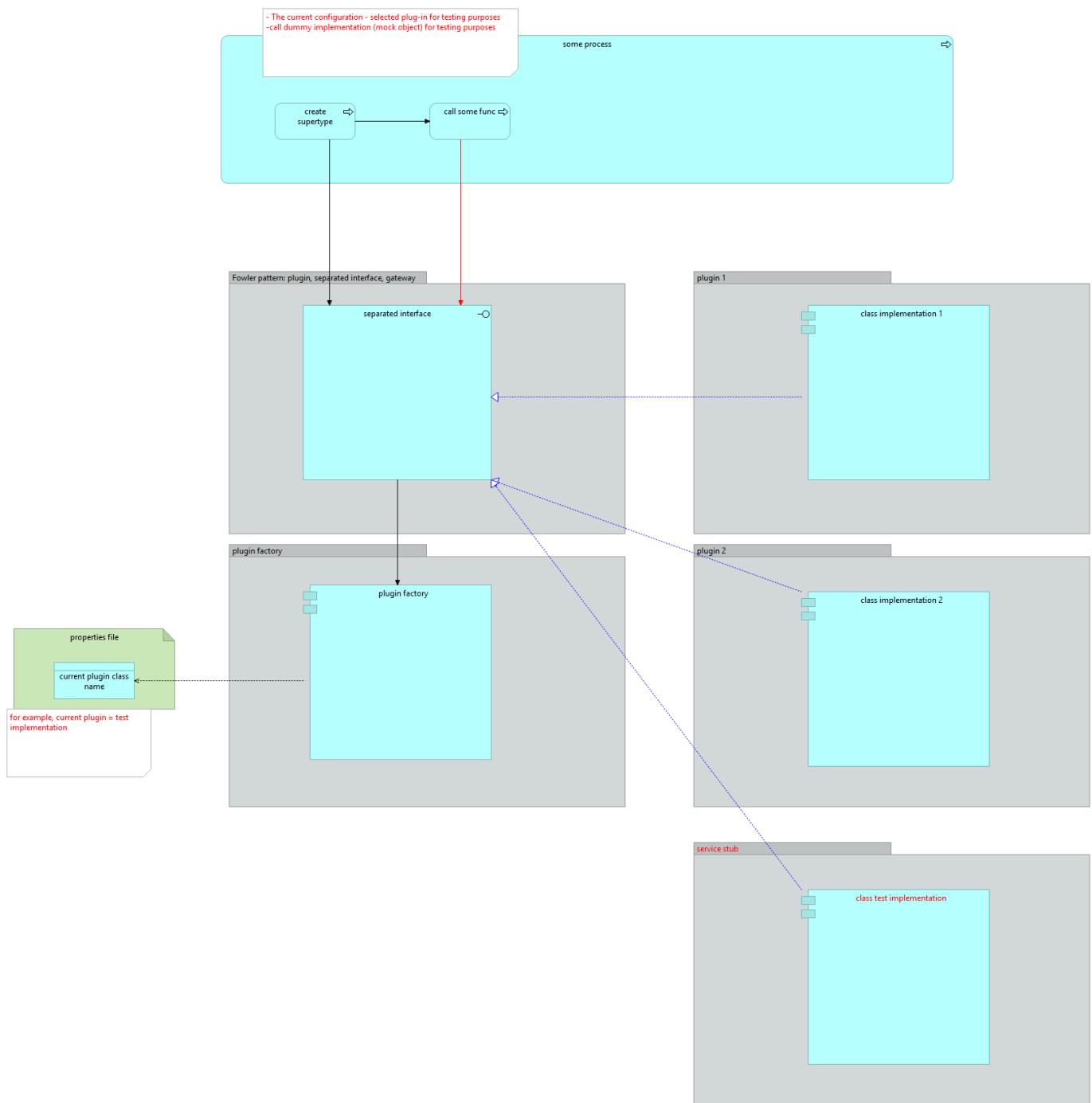
REGISTRY



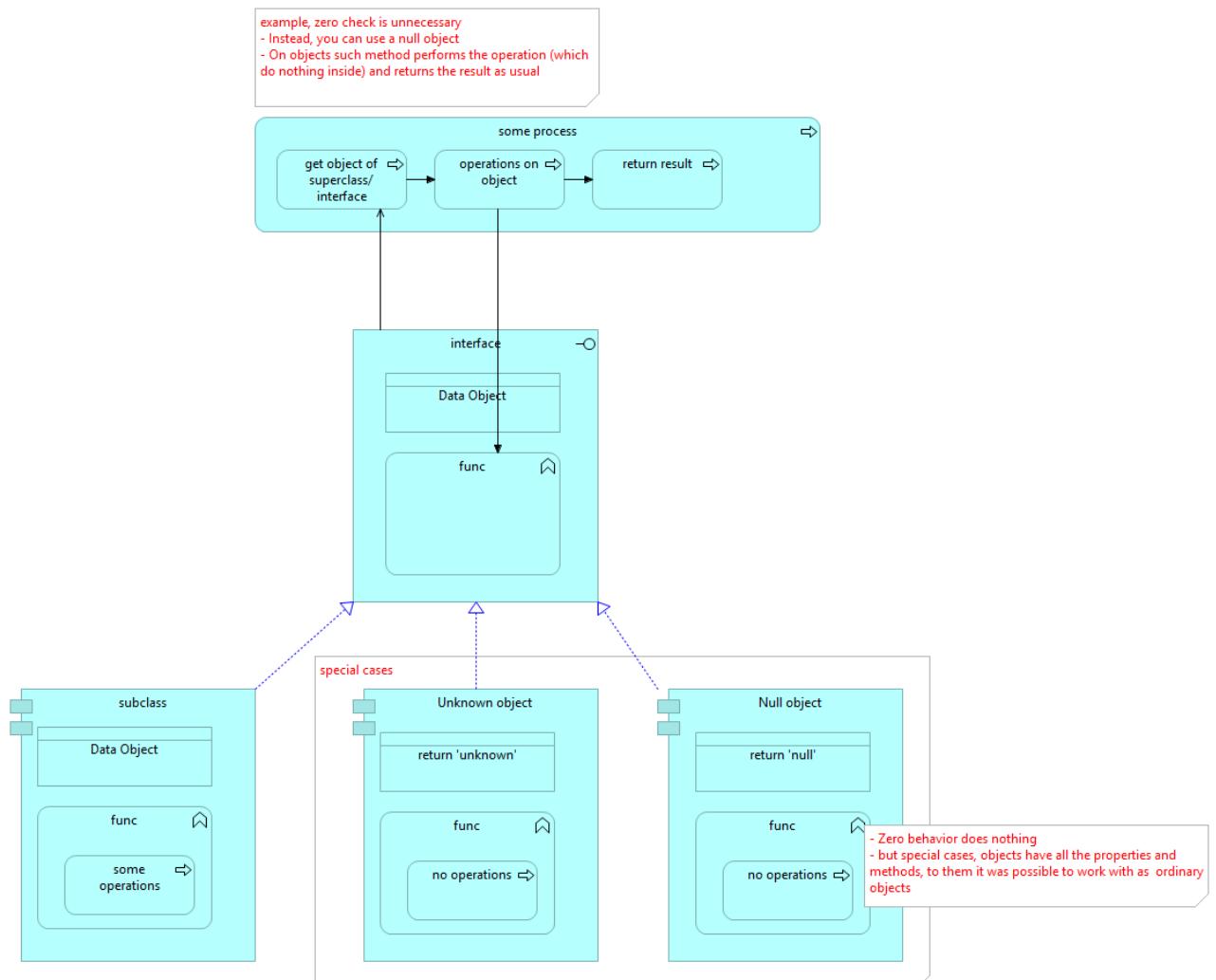
SEPARATED INTERFACE



SERVICE STUB

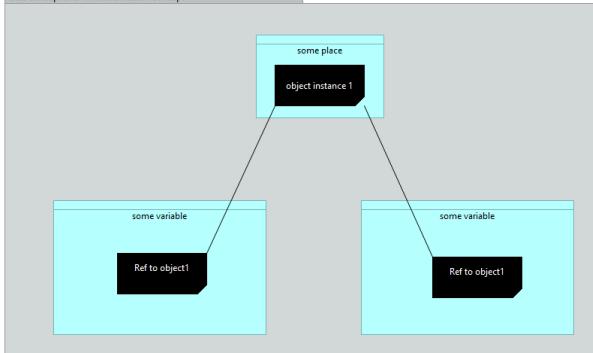


SPECIAL CASE

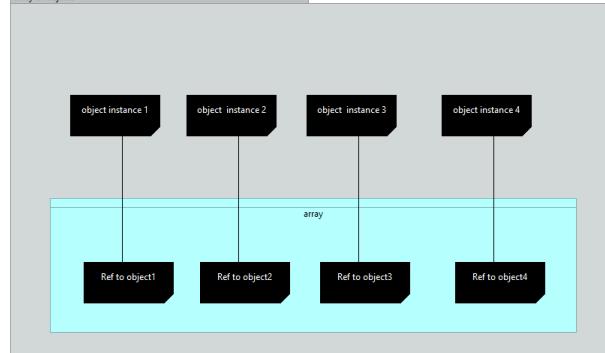


VALUE OBJECT

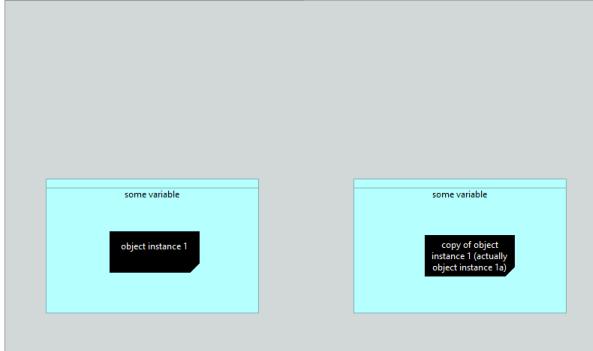
class concept and link transmission concept



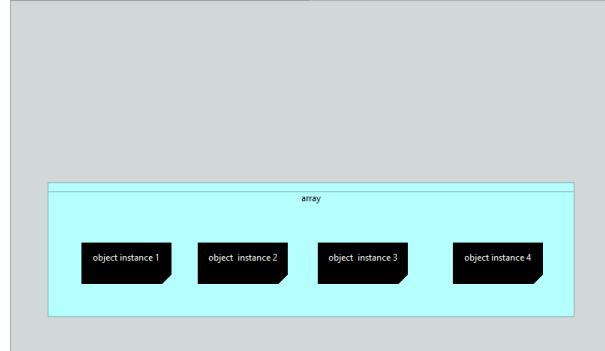
array of objects



value object concept and by value transfer concept



array of value objects

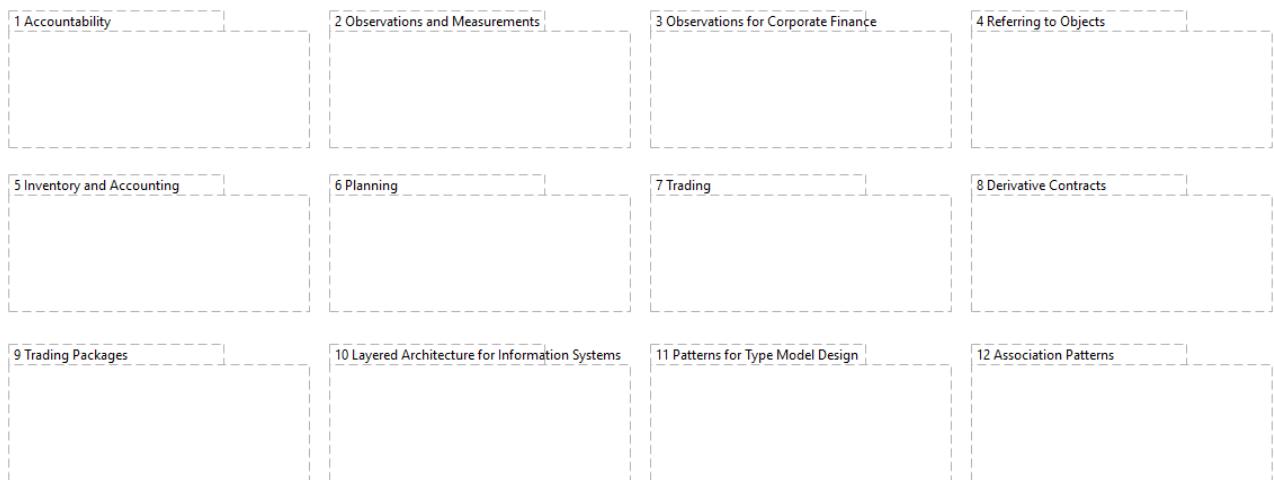


03

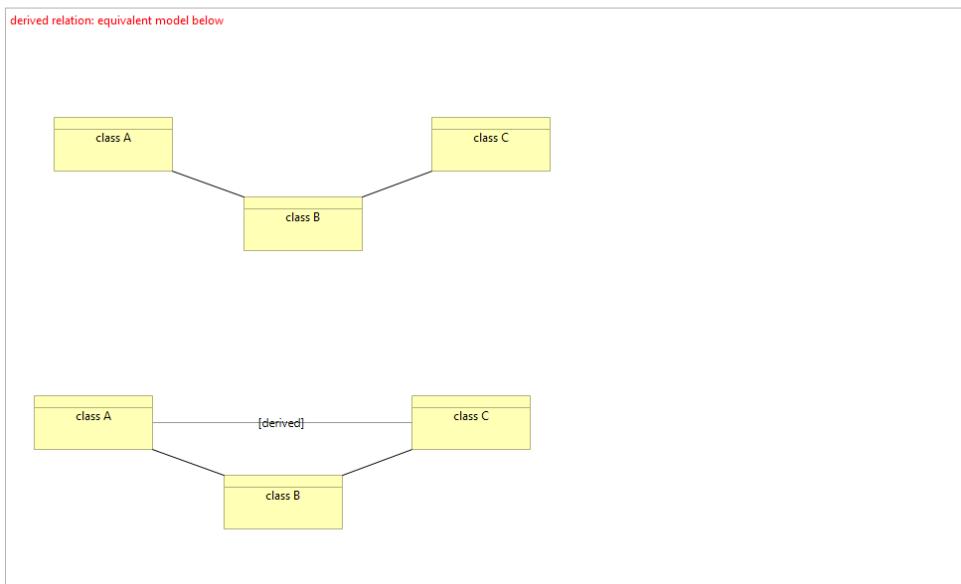
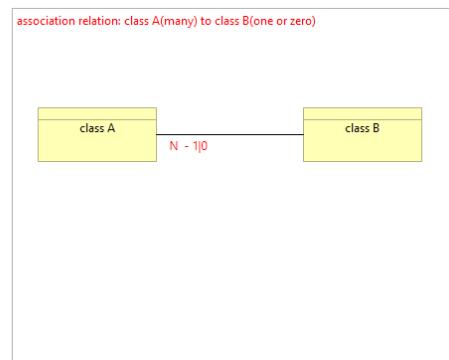
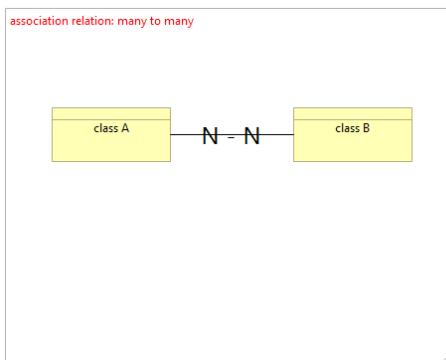
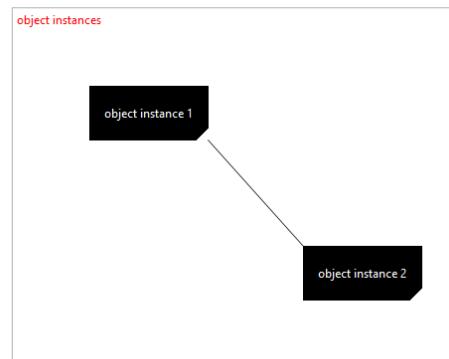
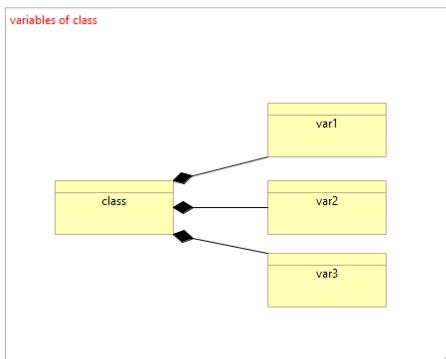
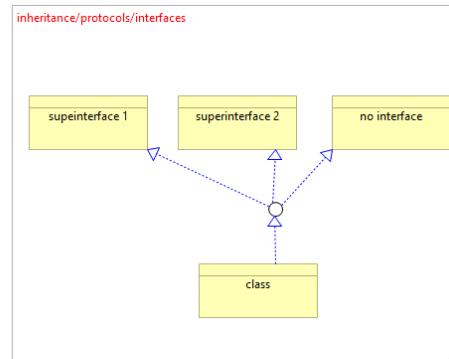
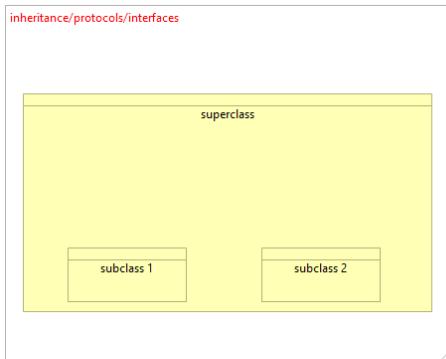
Analysis Patterns

Reusable Object Models

MARTIN FOWLER



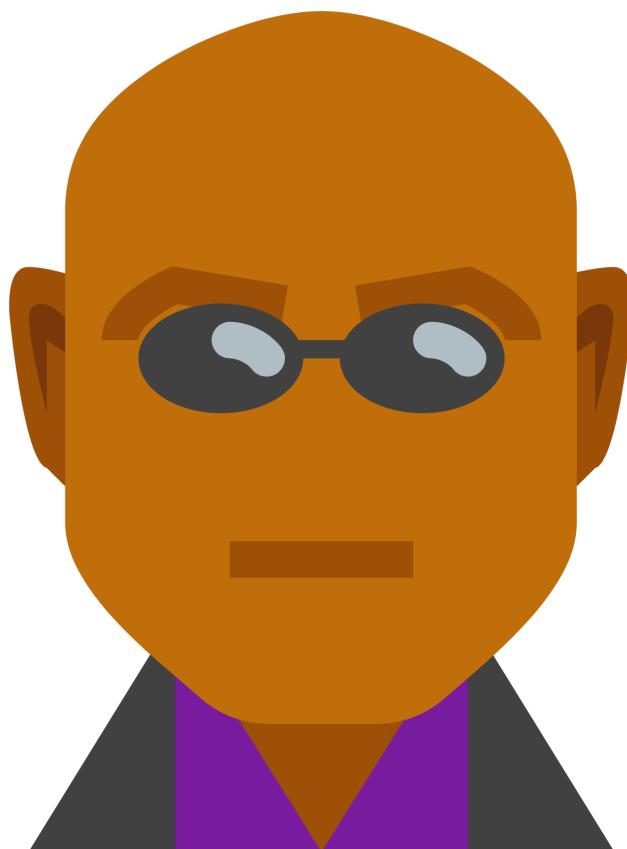
USED NOTATION



ACCOUNTABILITY

ANALYSIS PATTERNS

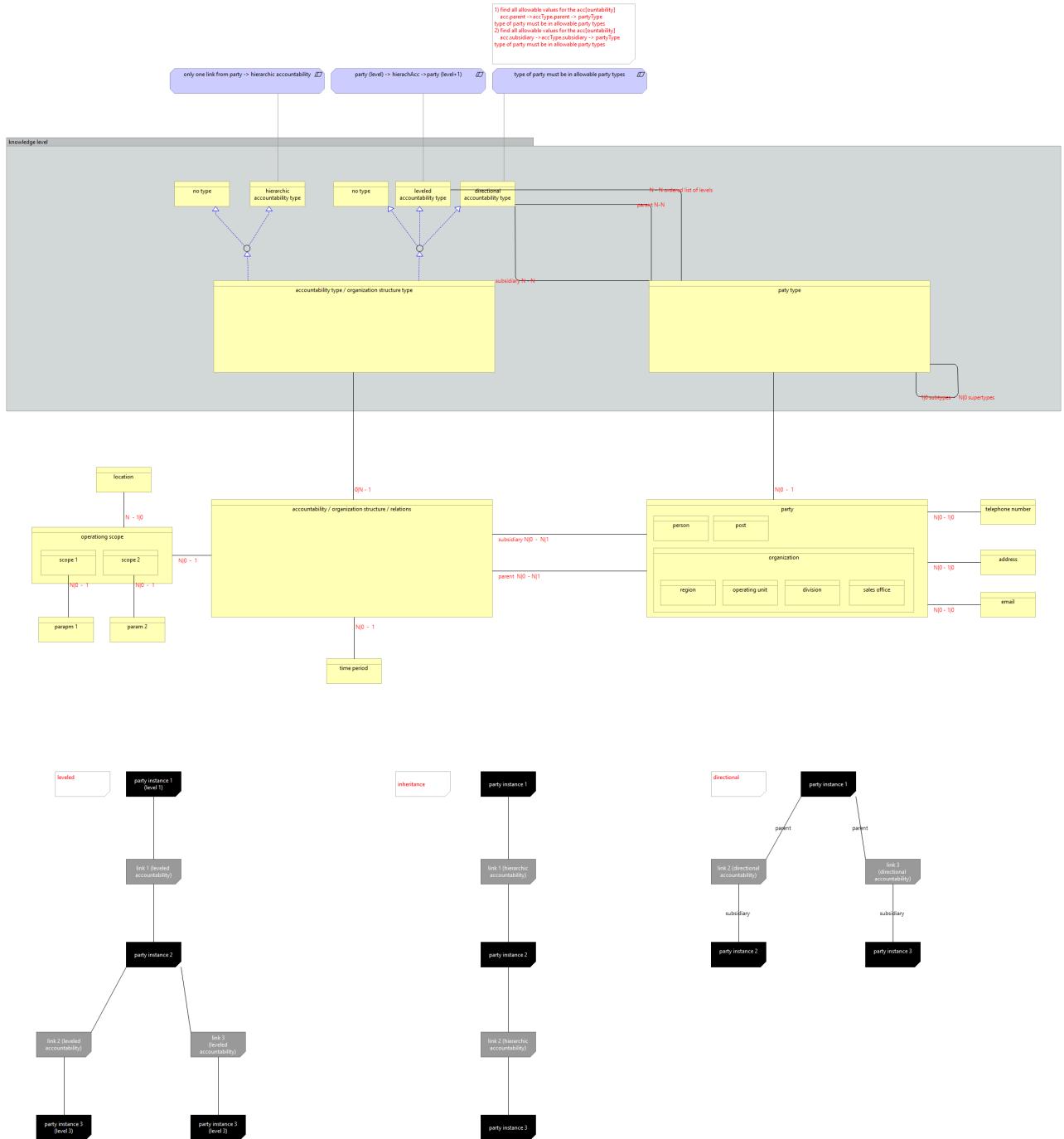
Martin Fowler

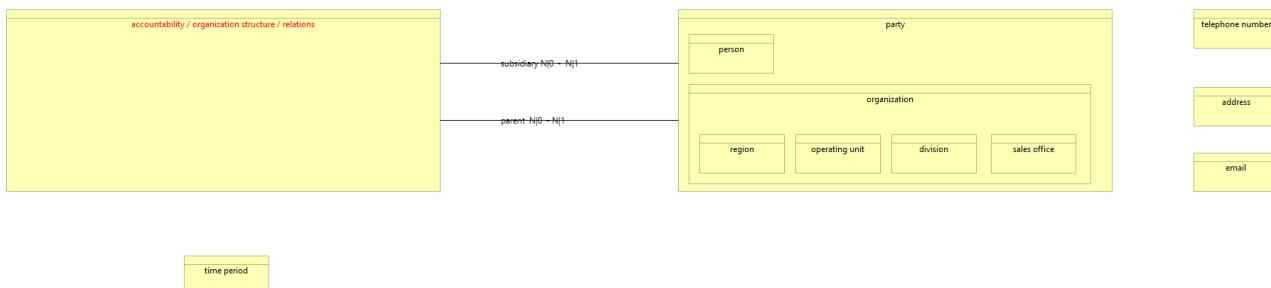
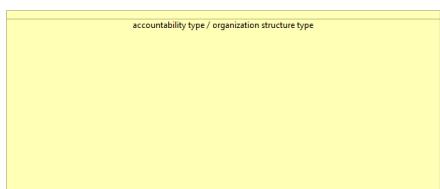


PARTY

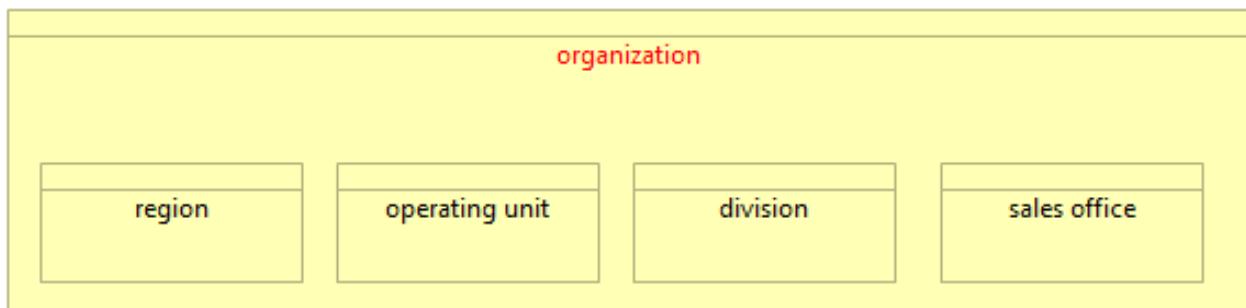


ACCOUNTABILITY

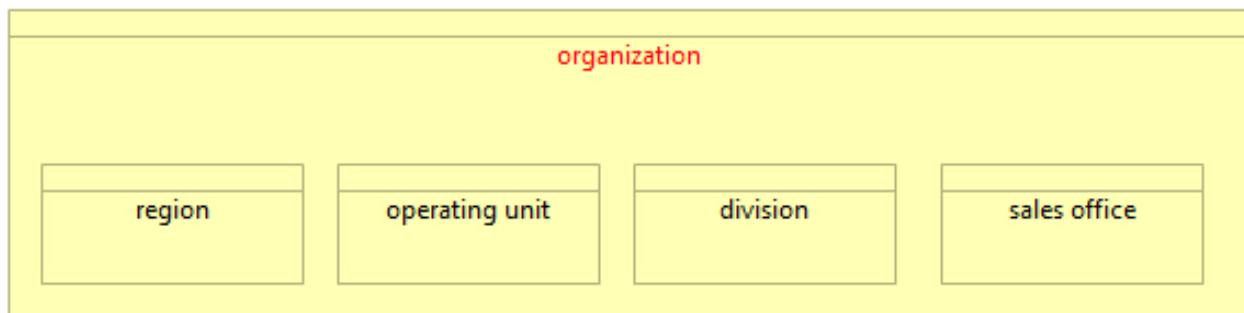




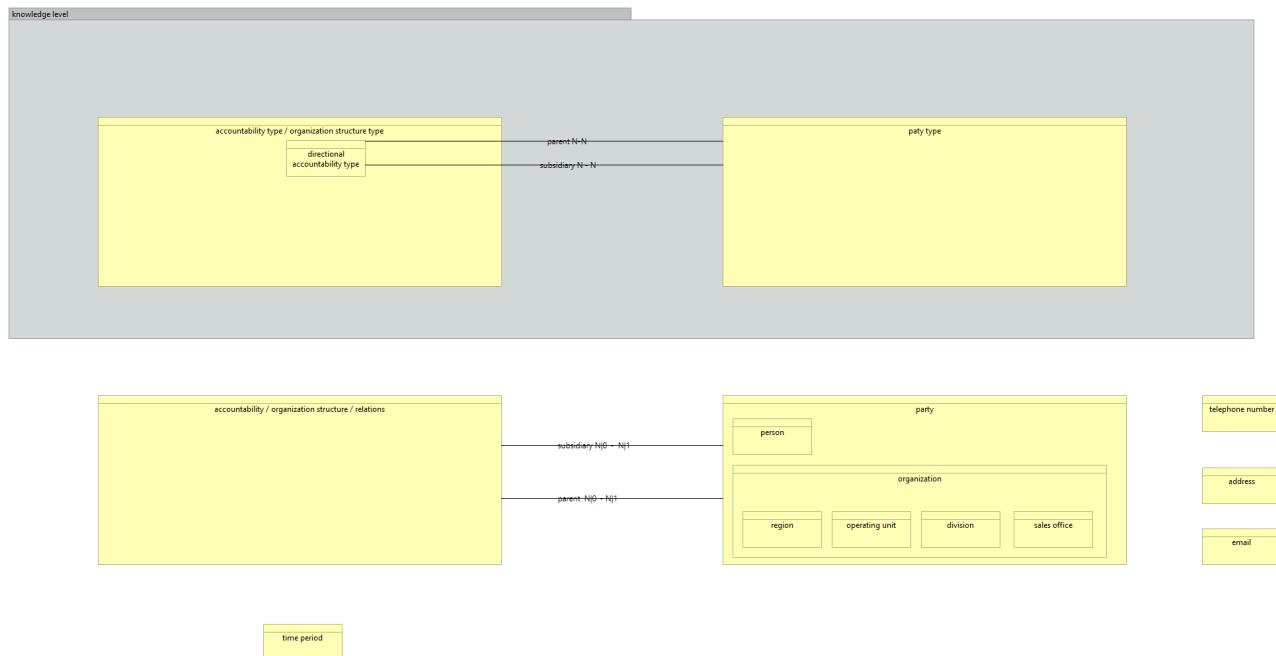
ORGANIZATION HIERARCHIES



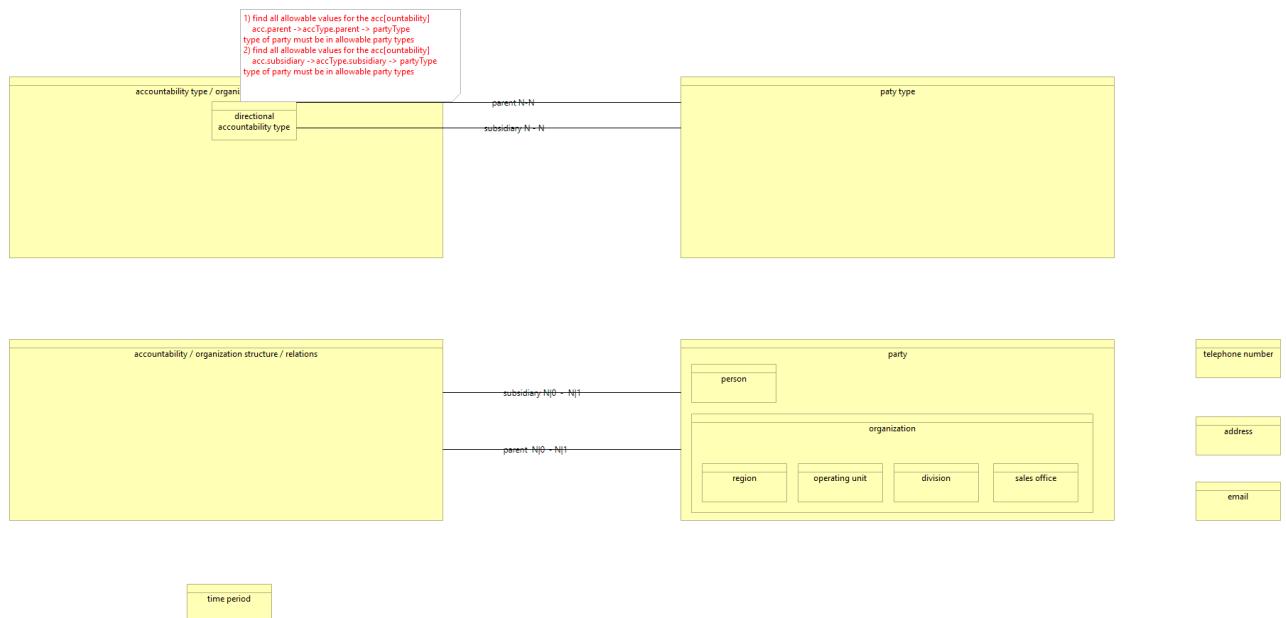
ORGANIZATION STRUCTURE



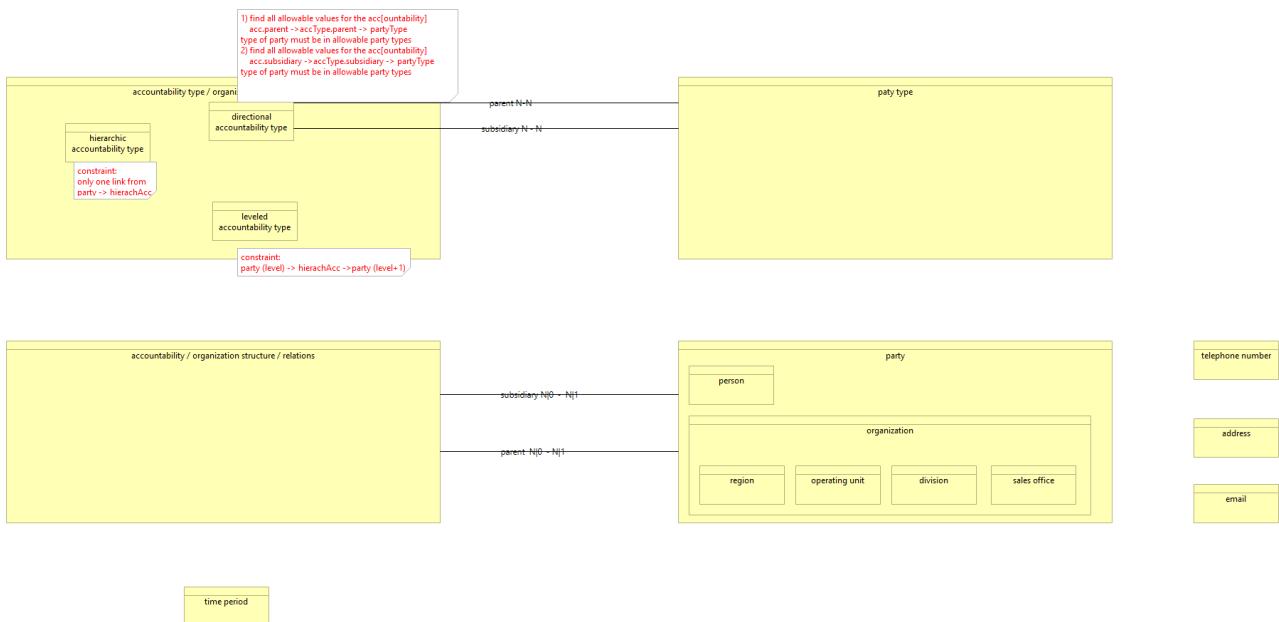
ACCOUNTABILITY KNOWLEDGE LEVEL



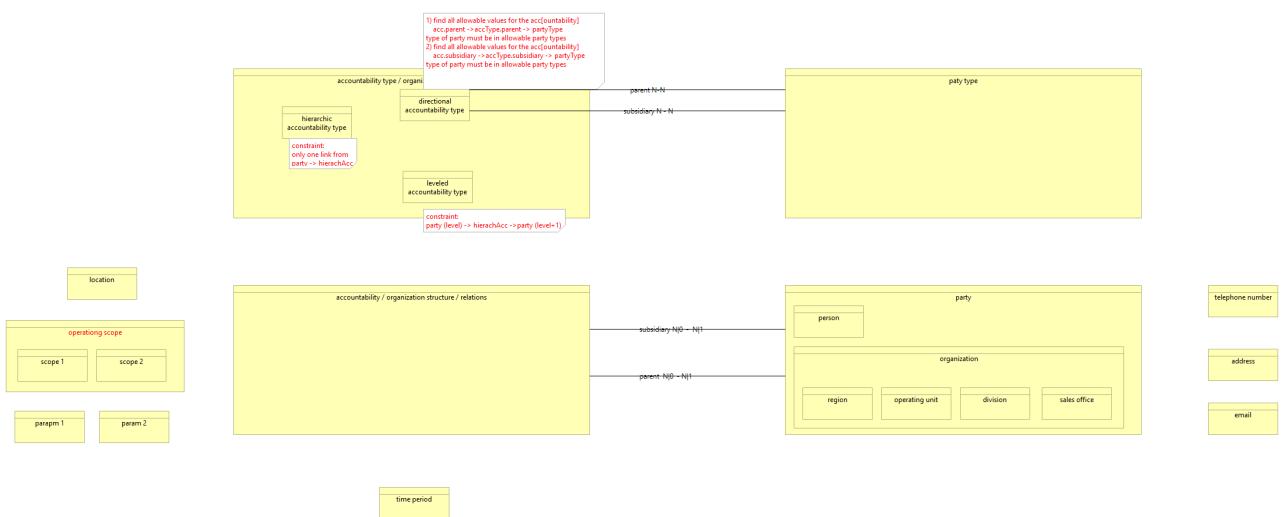
PARTY TYPE GENERALIZATIONS



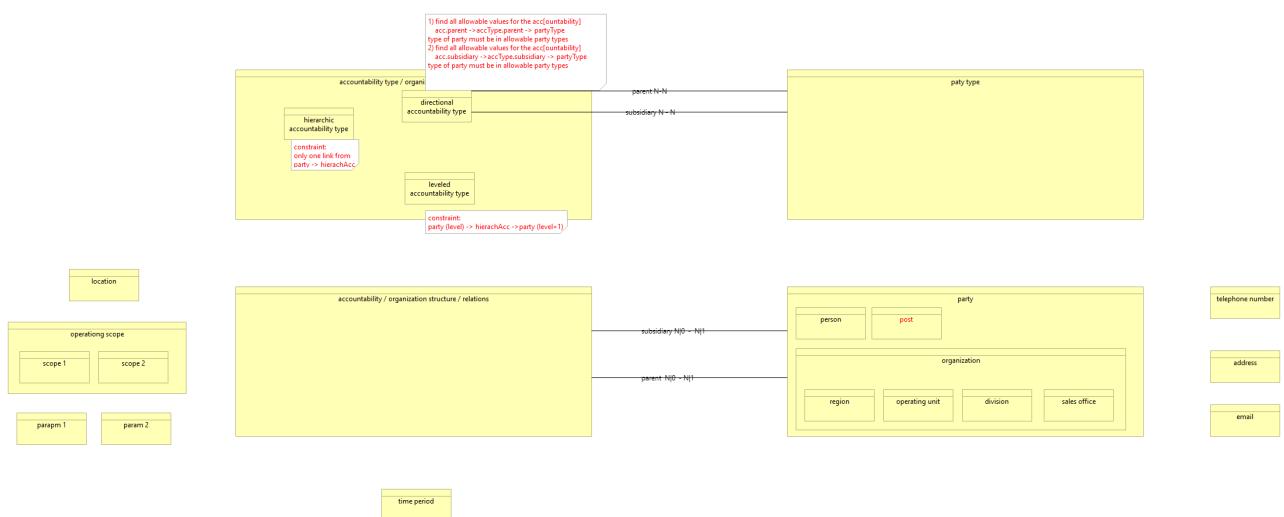
HIERARCHIC ACCOUNTABILITY



OPERATING SCOPES



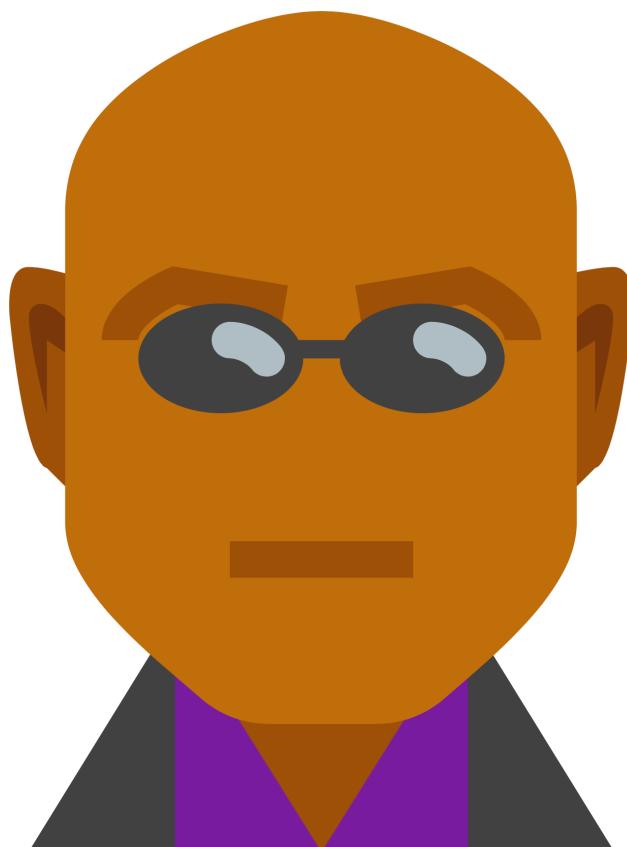
POST



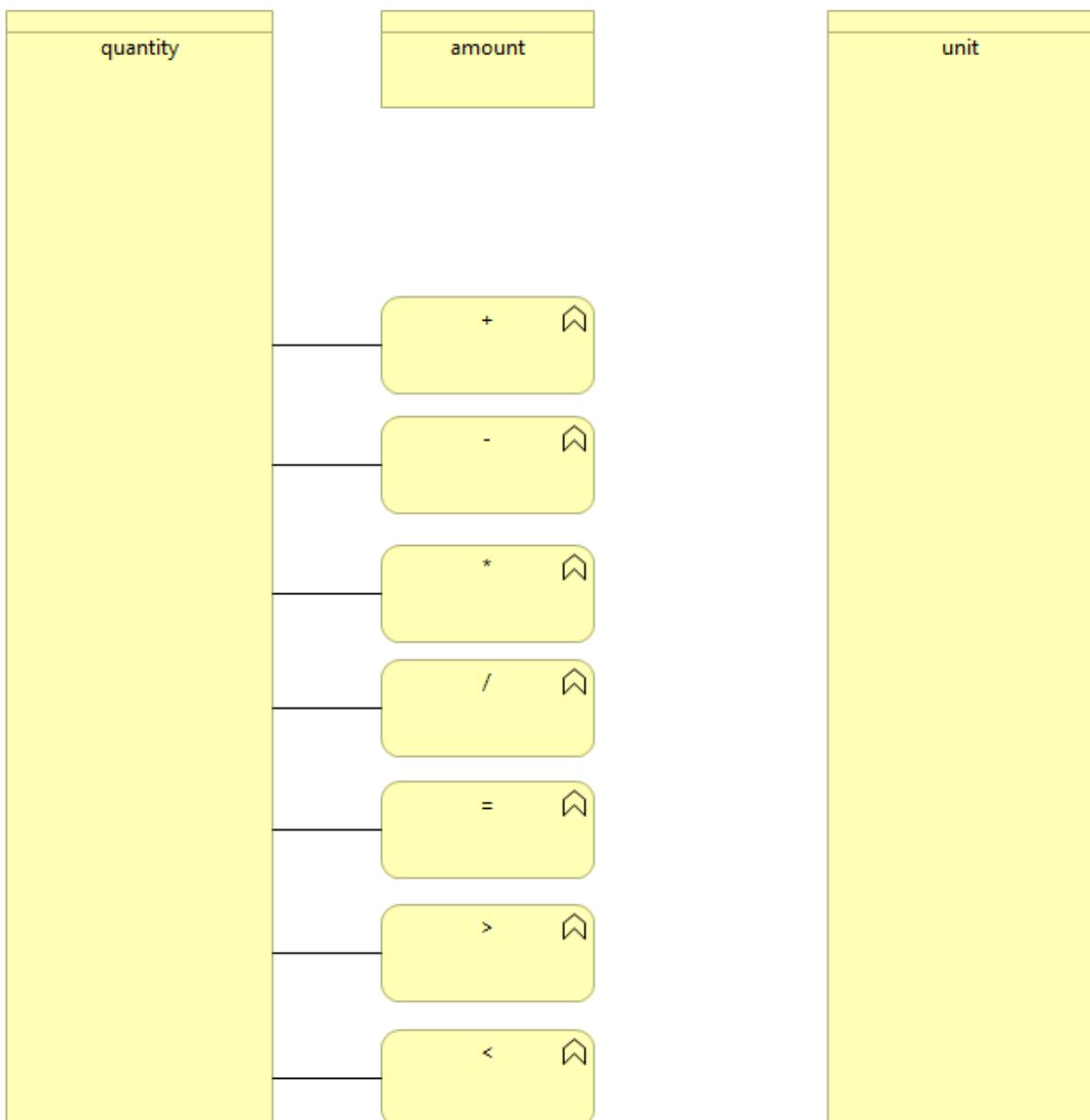
OBSERVATIONS AND MEASUREMENTS

ANALYSIS PATTERNS

Martin Fowler



QUANTITY

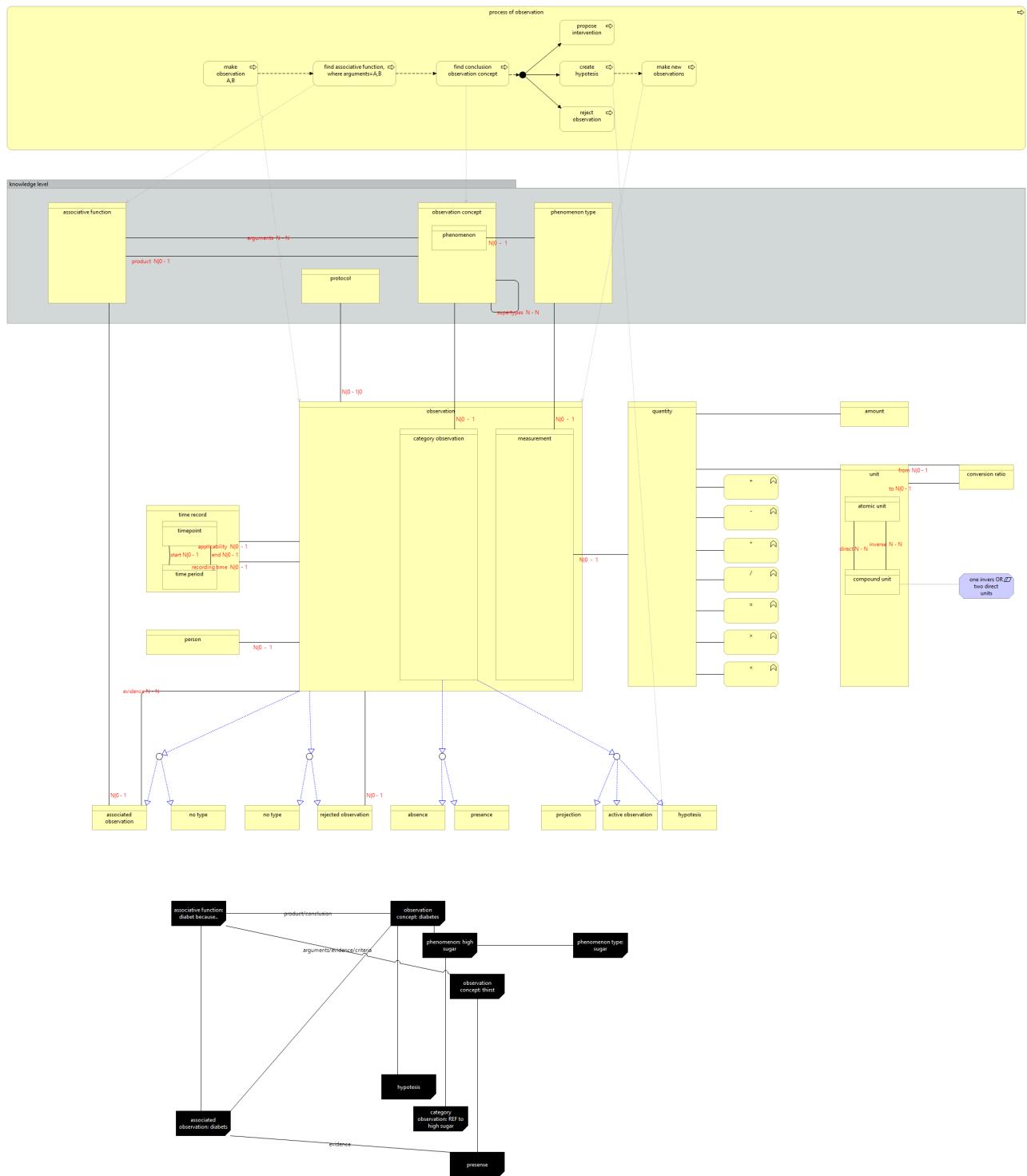


CONVERSION RATIO

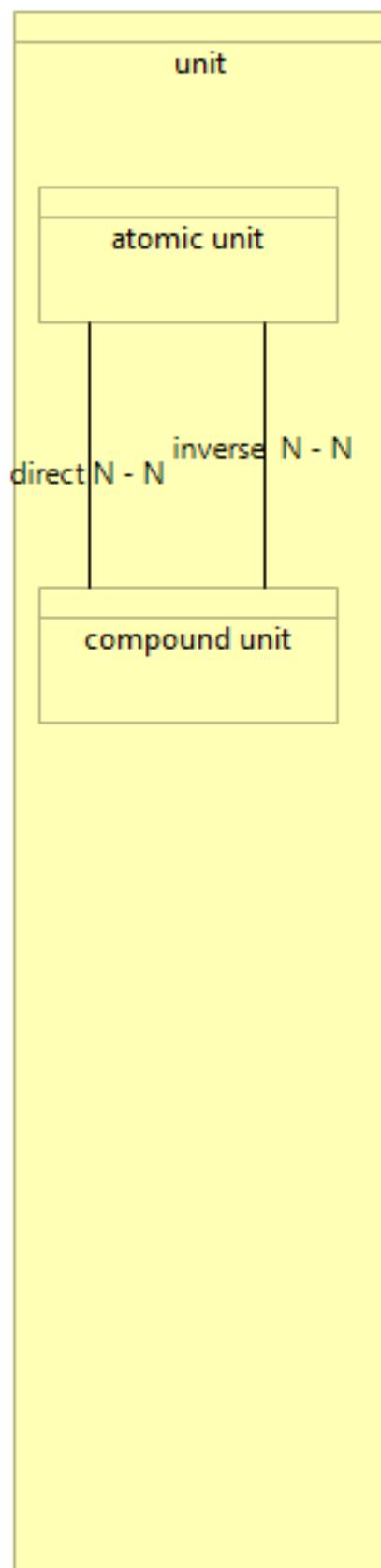
unit

conversion ratio

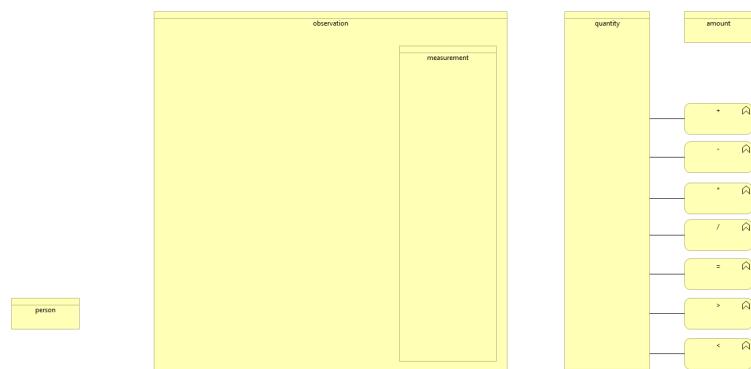
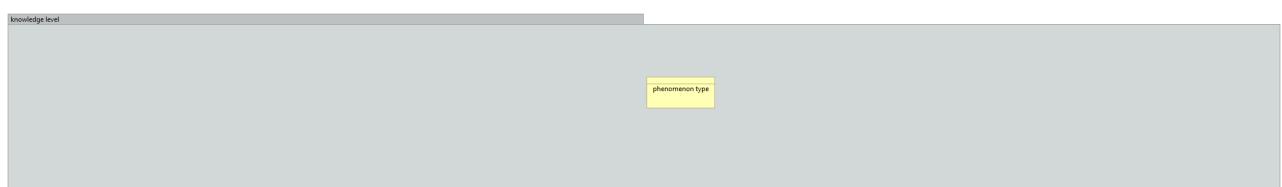
OBSERVATIONS AND MEASUREMENTS



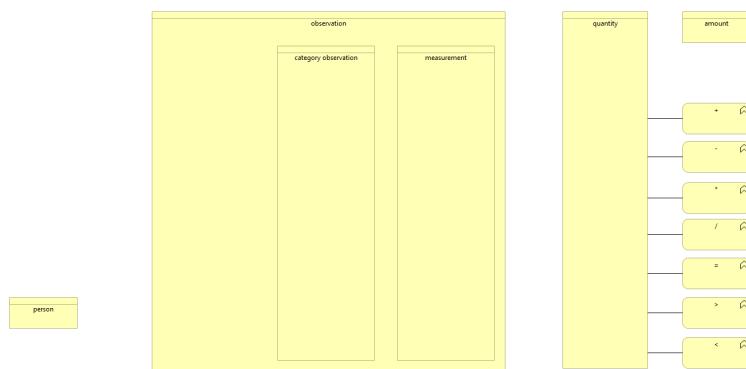
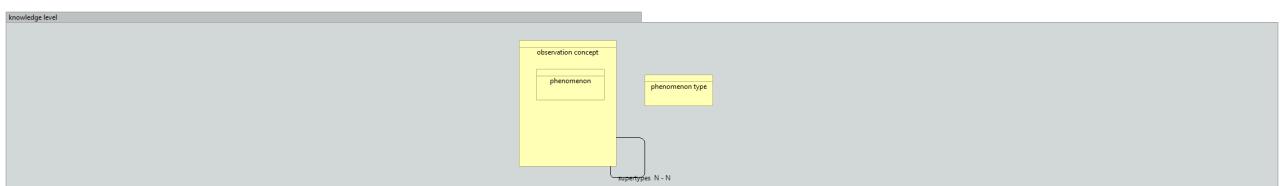
COMPOUND UNITS



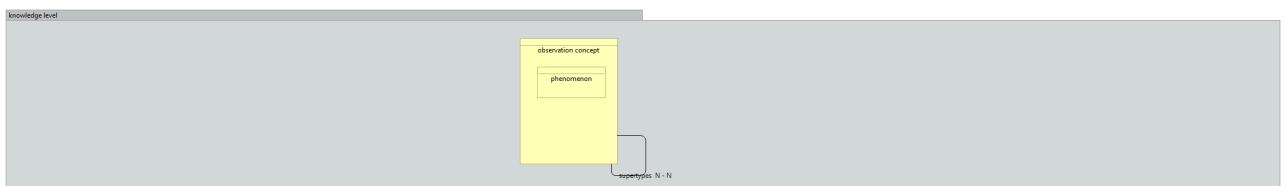
MEASUREMENT



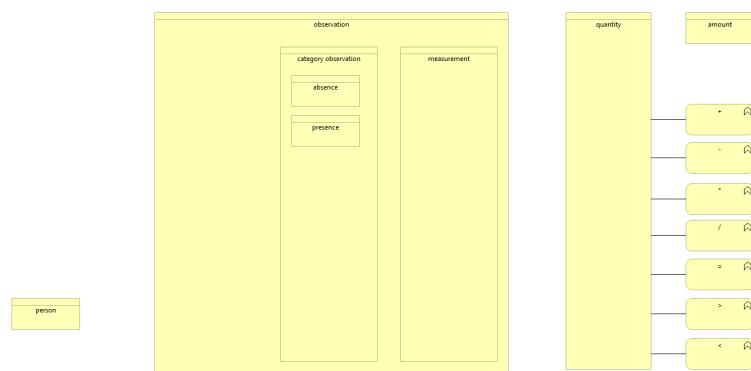
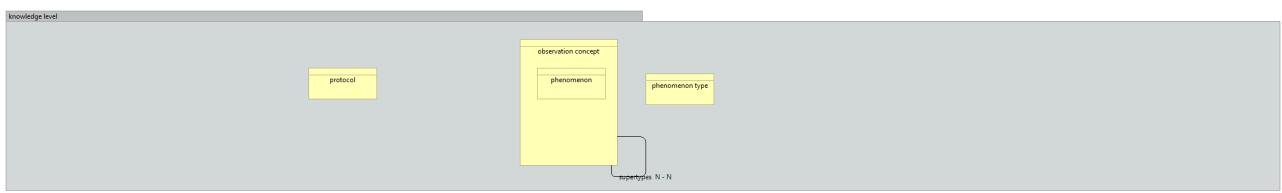
OBSERVATION



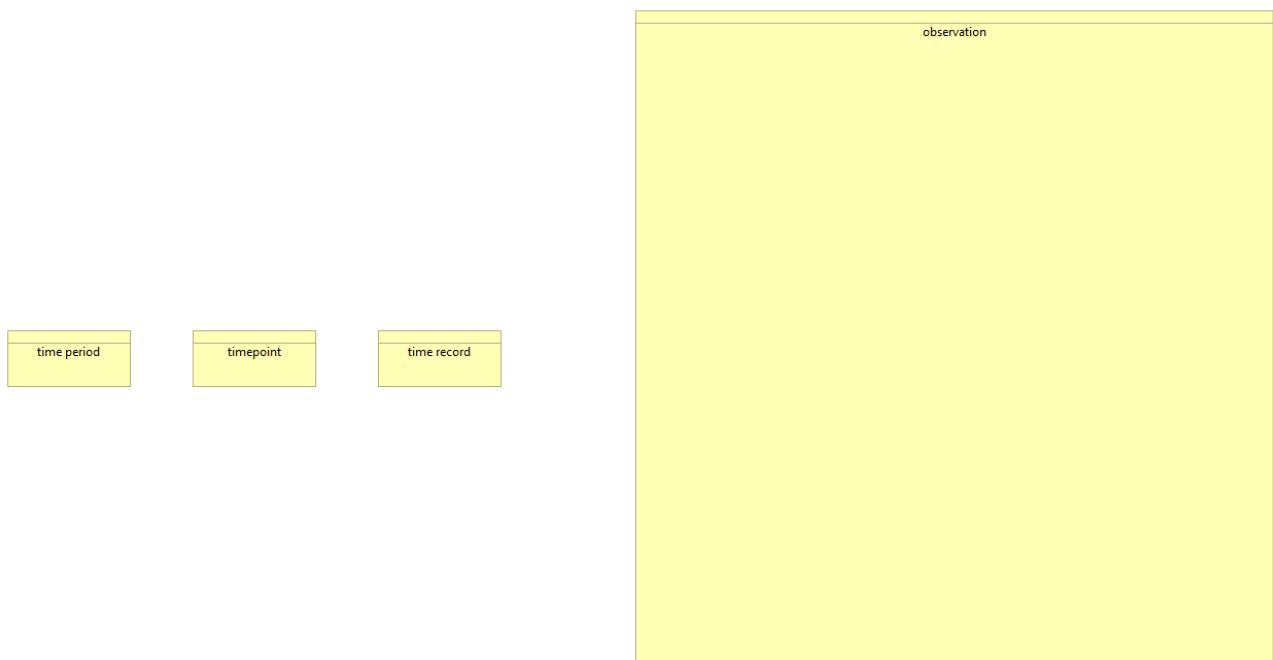
SUBTYPING OBSERVATION CONCEPTS



PROTOCOL



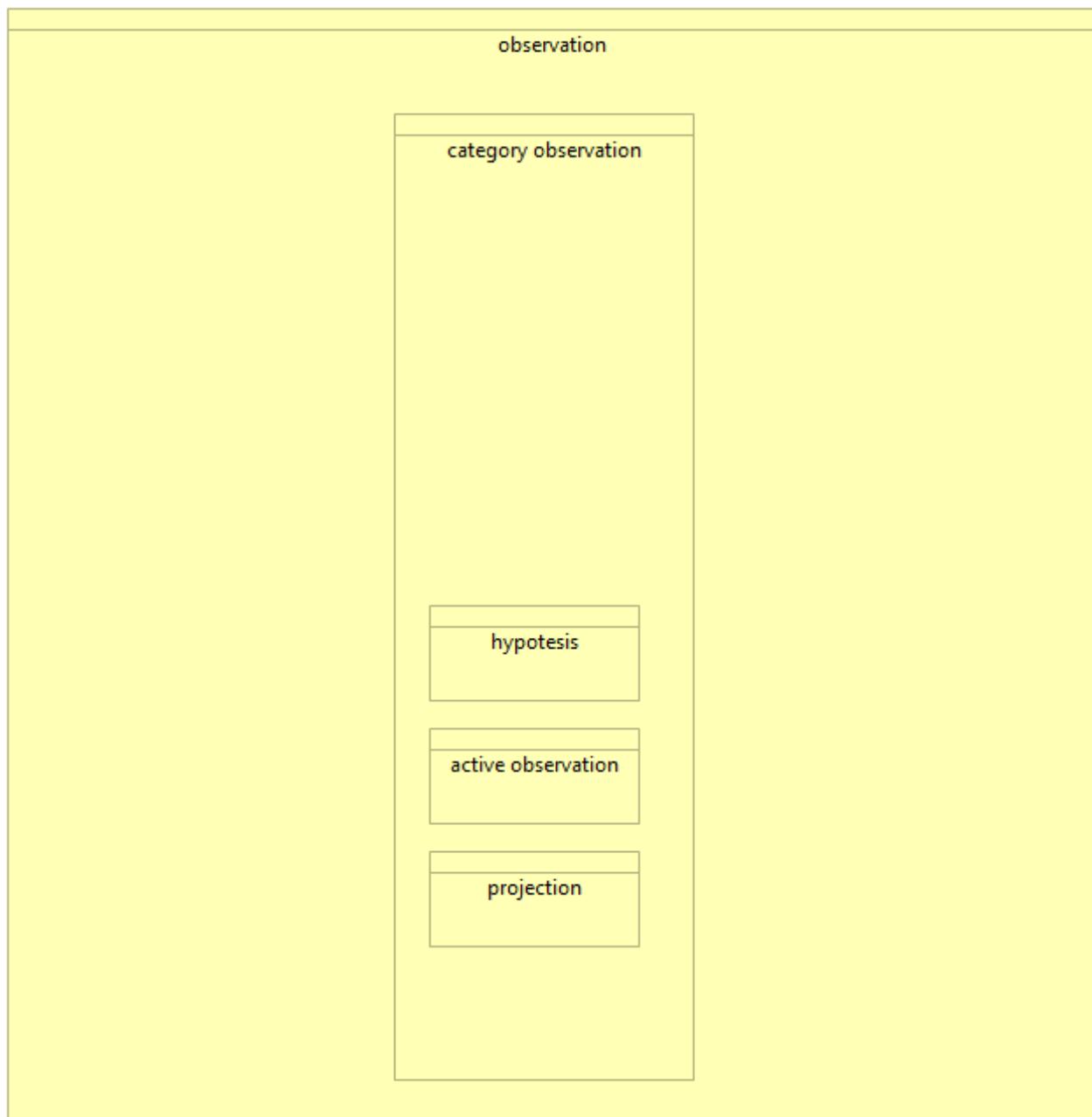
DUAL TIME RECORD



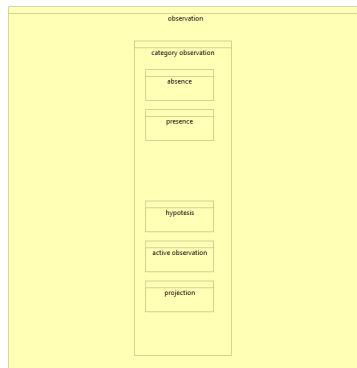
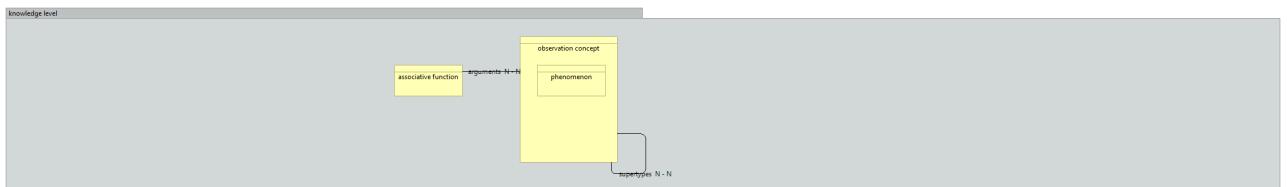
REJECTED OBSERVATION

observation

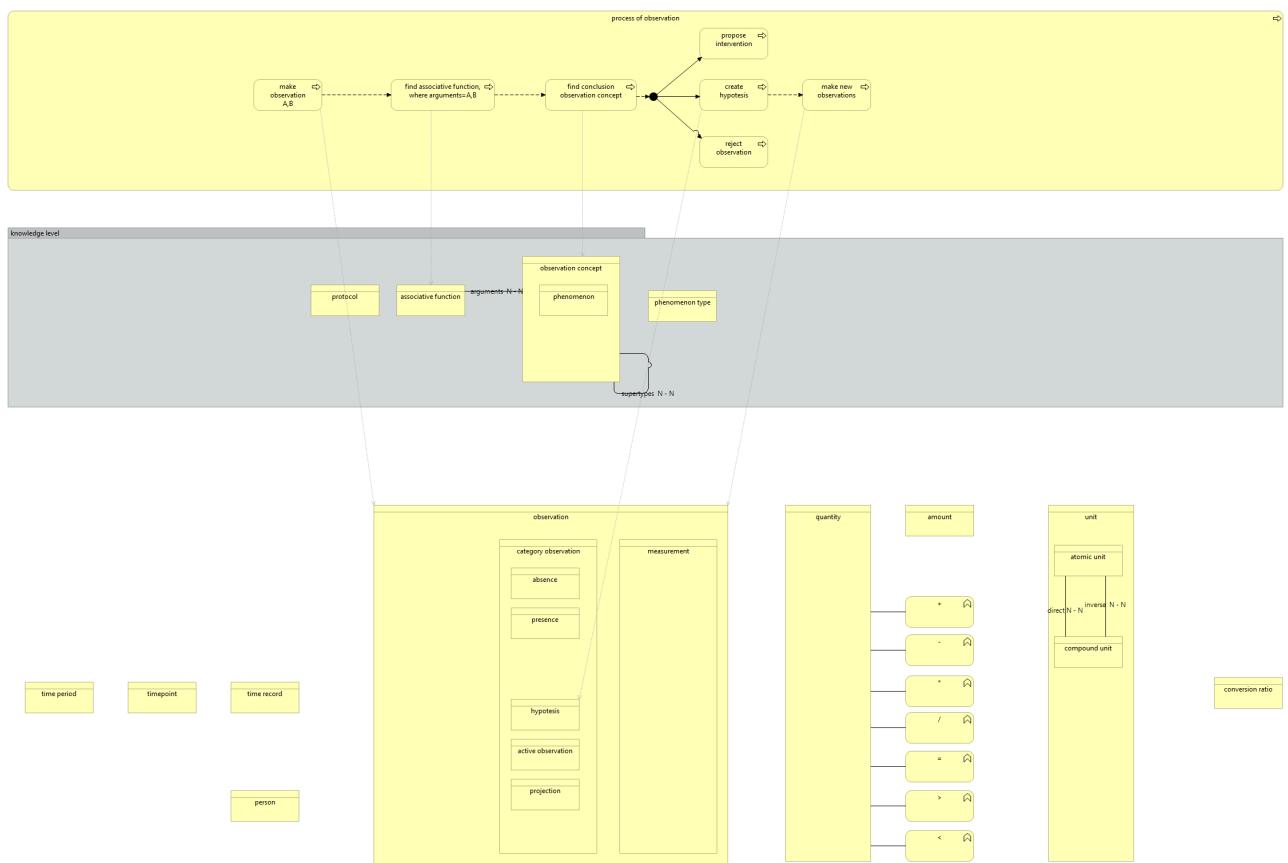
ACTIVE OBSERVATION, HYPOTHESIS, AND PROJECTION



ASSOCIATED OBSERVATION



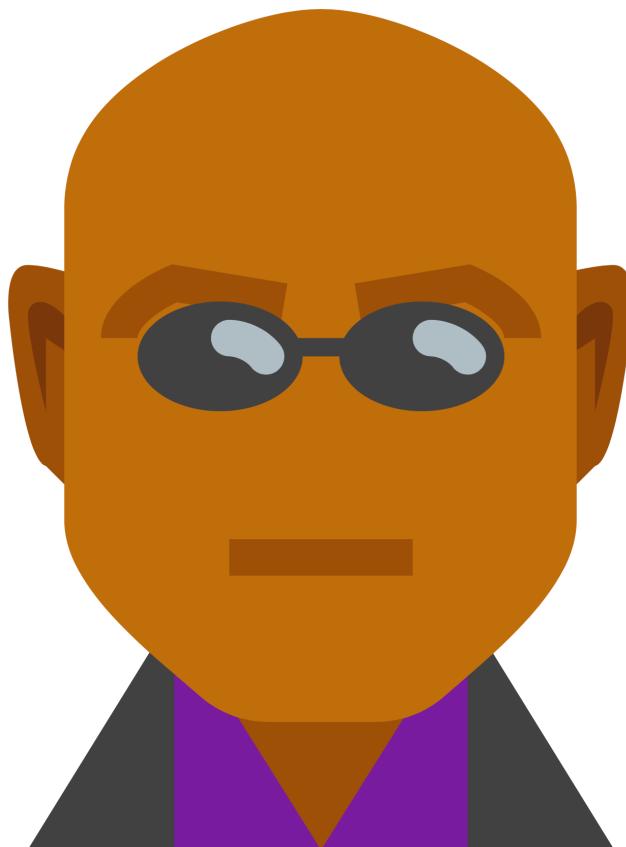
PROCESS OF OBSERVATION



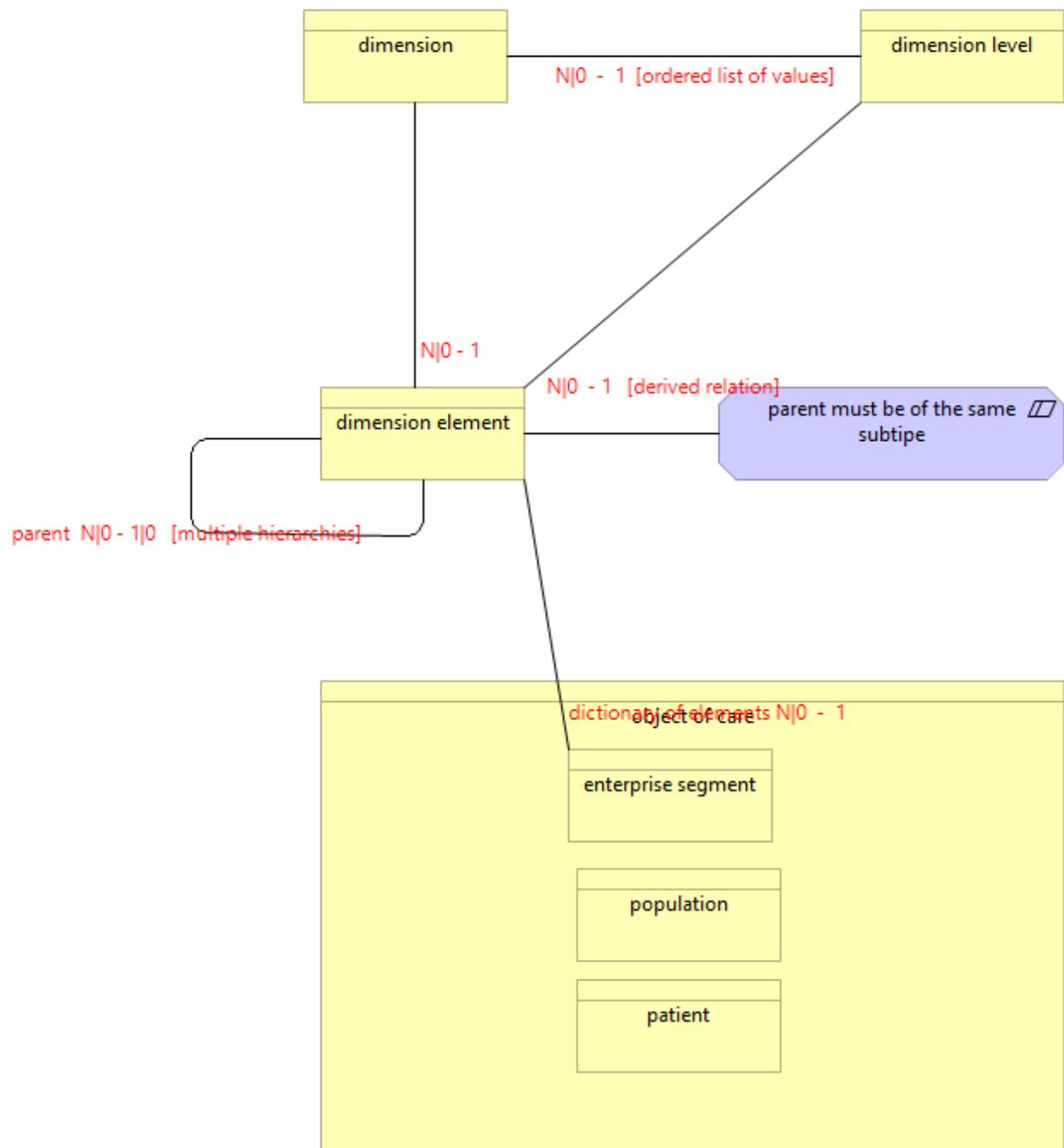
OBSERVATIONS FOR CORPORATE FINANCE

ANALYSIS PATTERNS

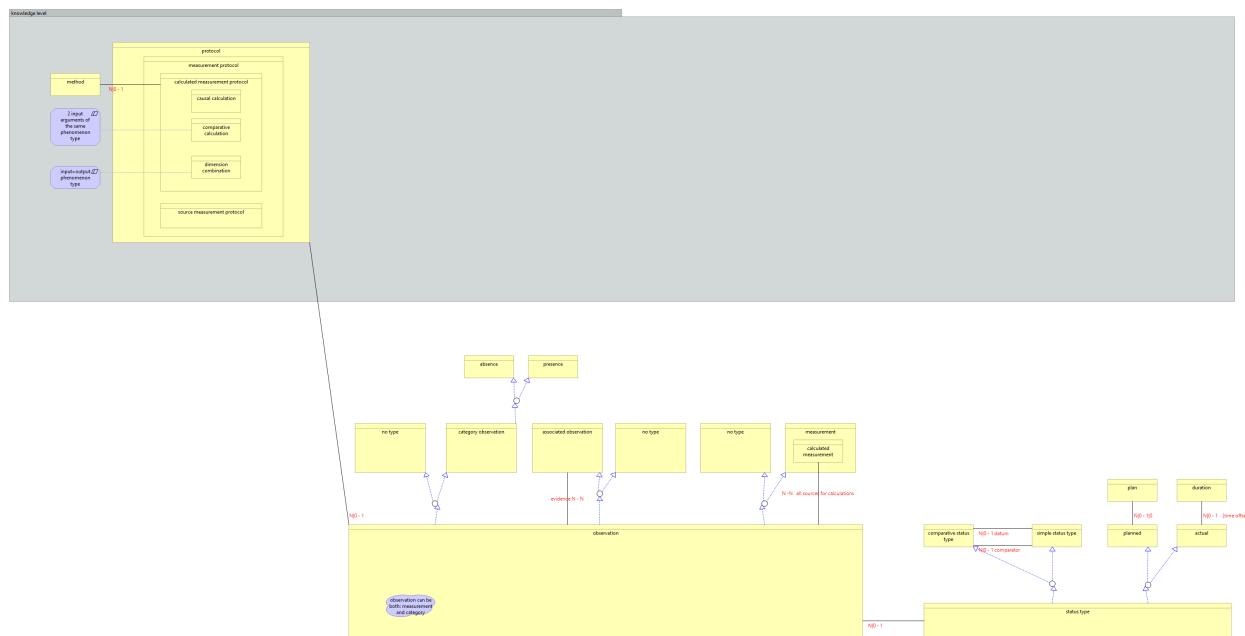
Martin Fowler



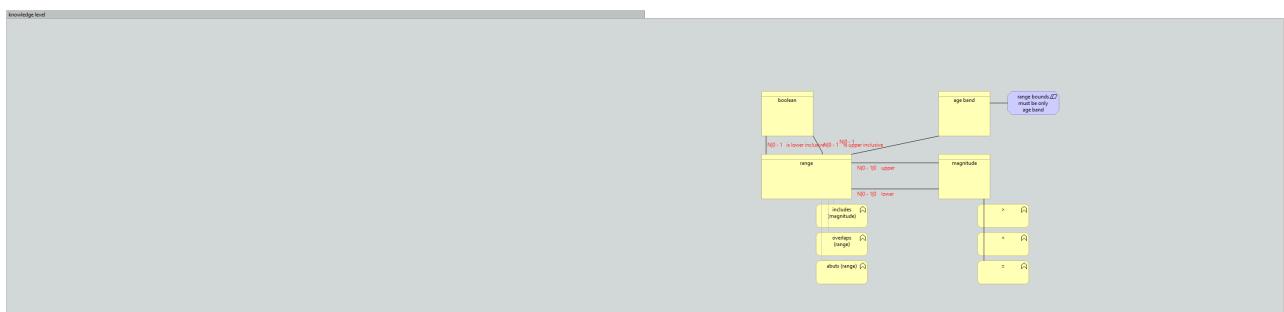
ENTERPRISE SEGMENT



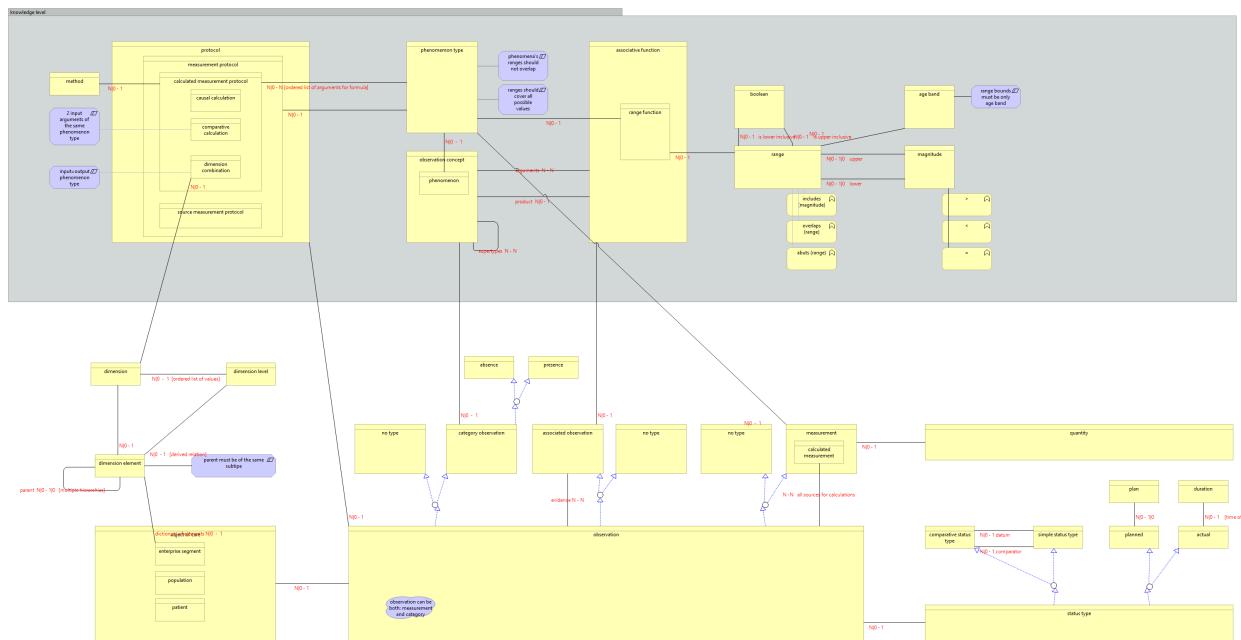
MEASUREMENT PROTOCOL

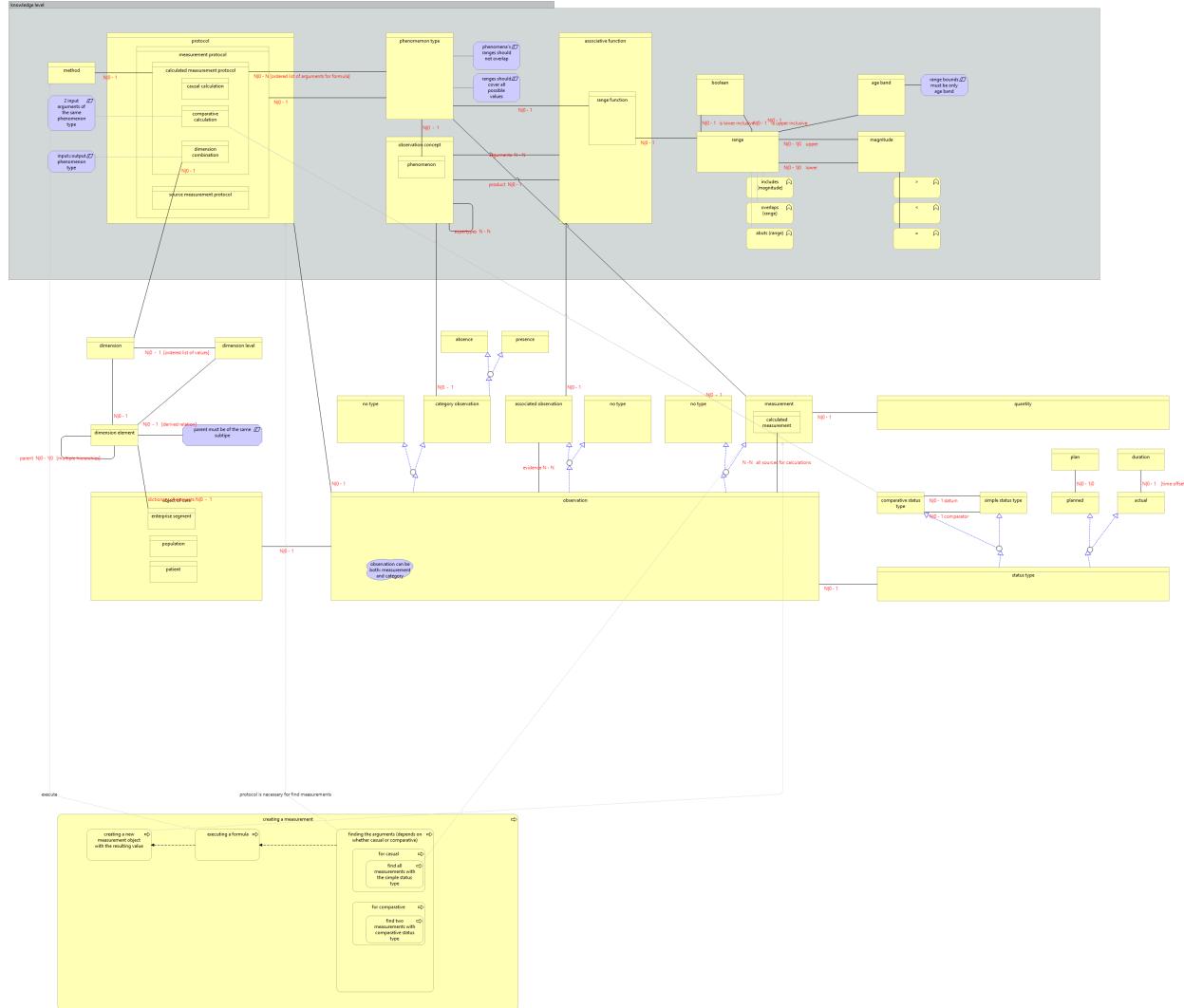


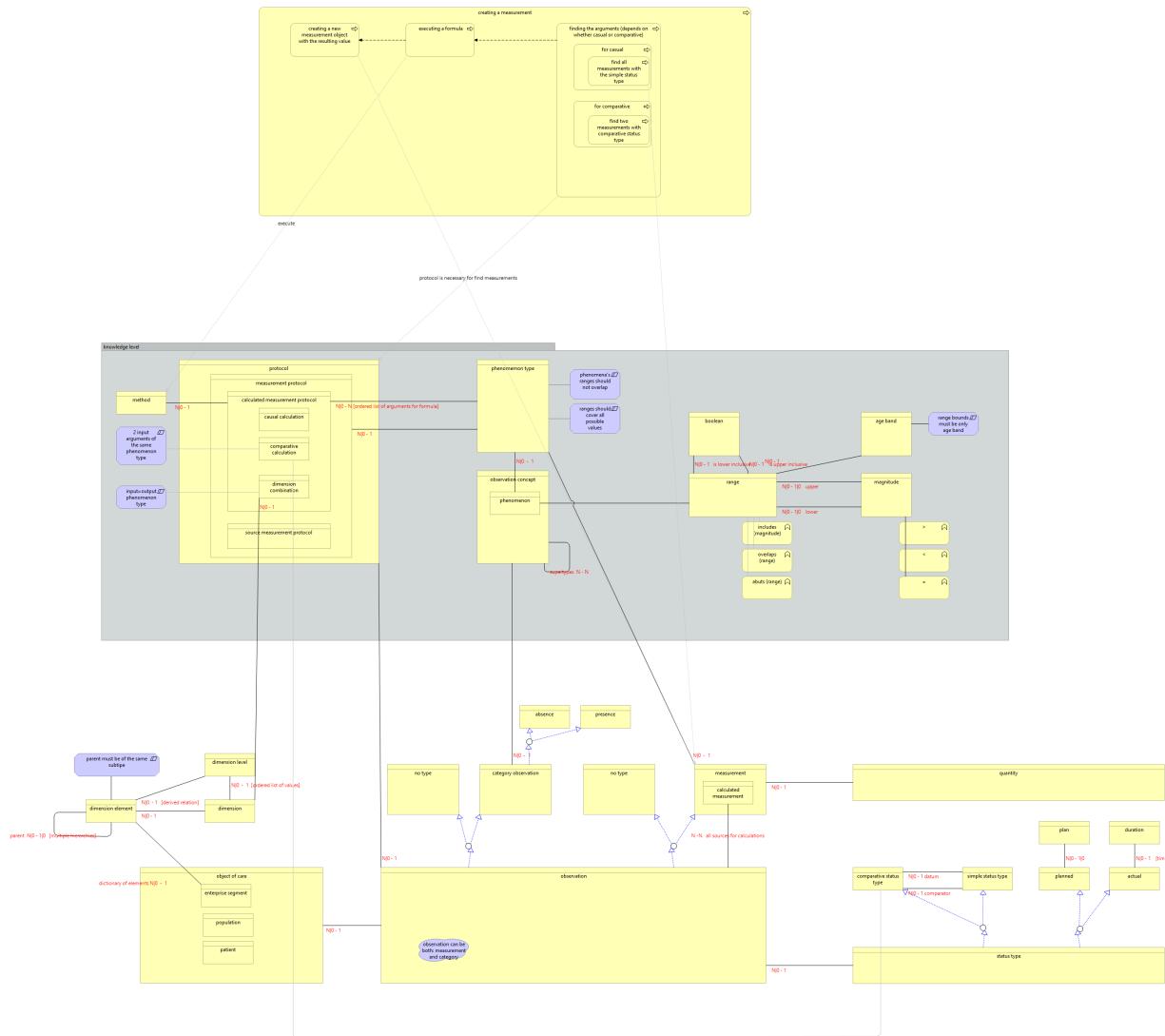
RANGE



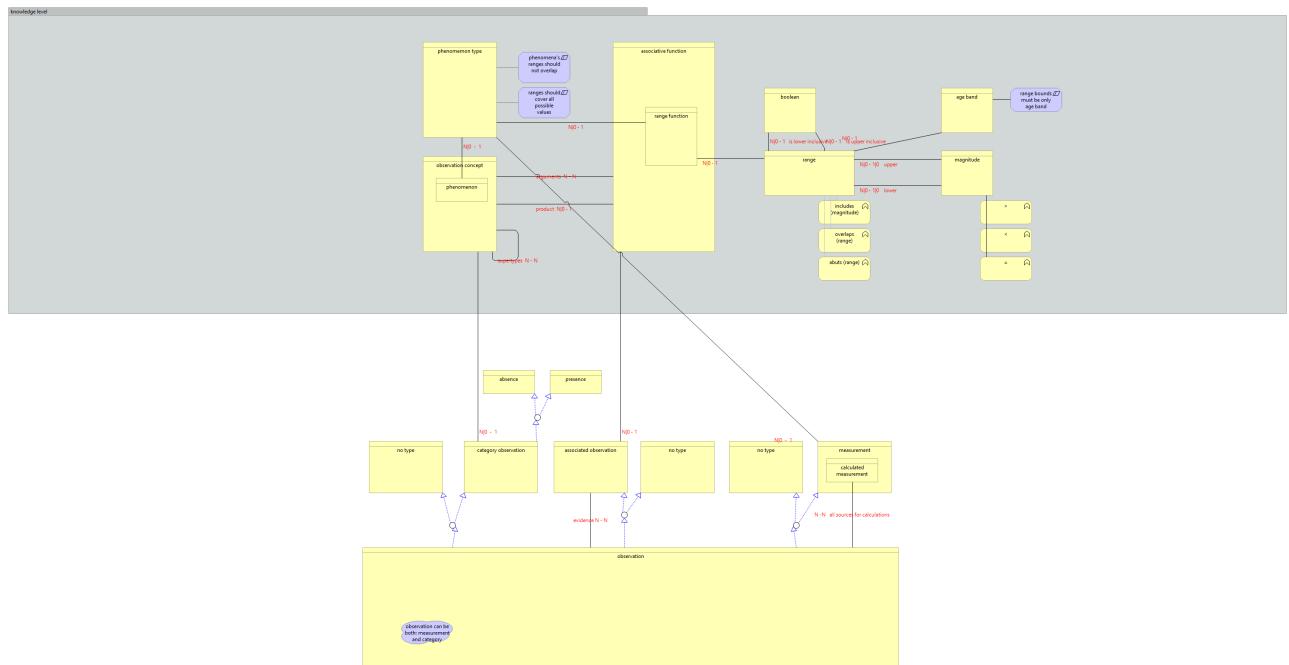
OBSERVATIONS FOR CORPORATE FINANCE







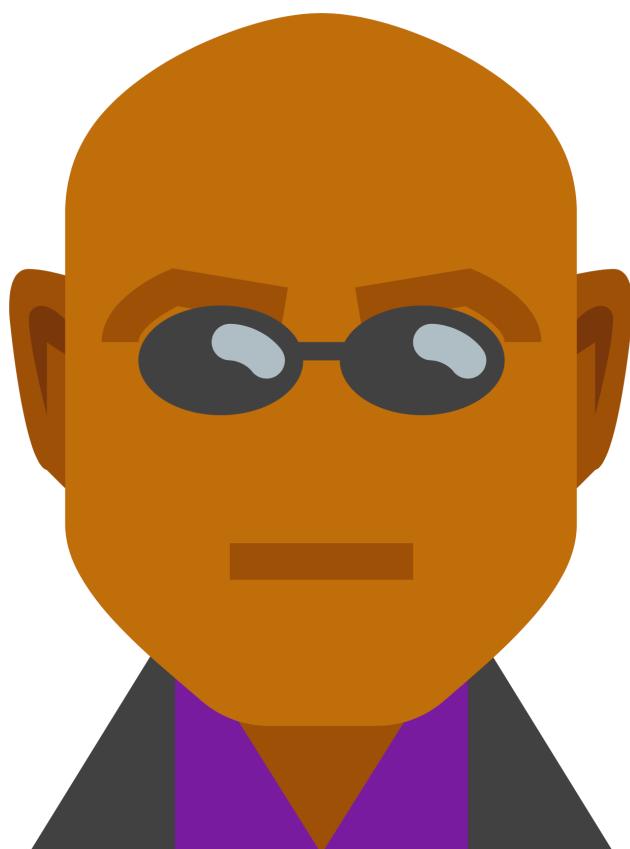
PHENOMENON WITH RANGE



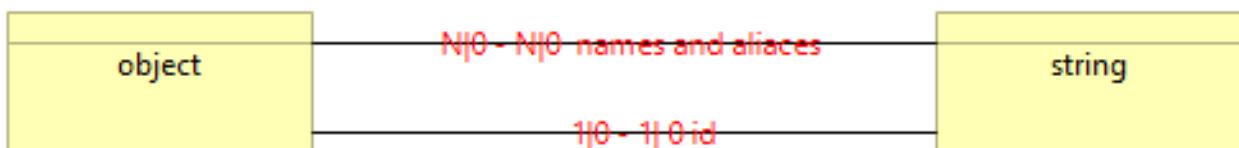
REFERRING TO OBJECTS

ANALYSIS PATTERNS

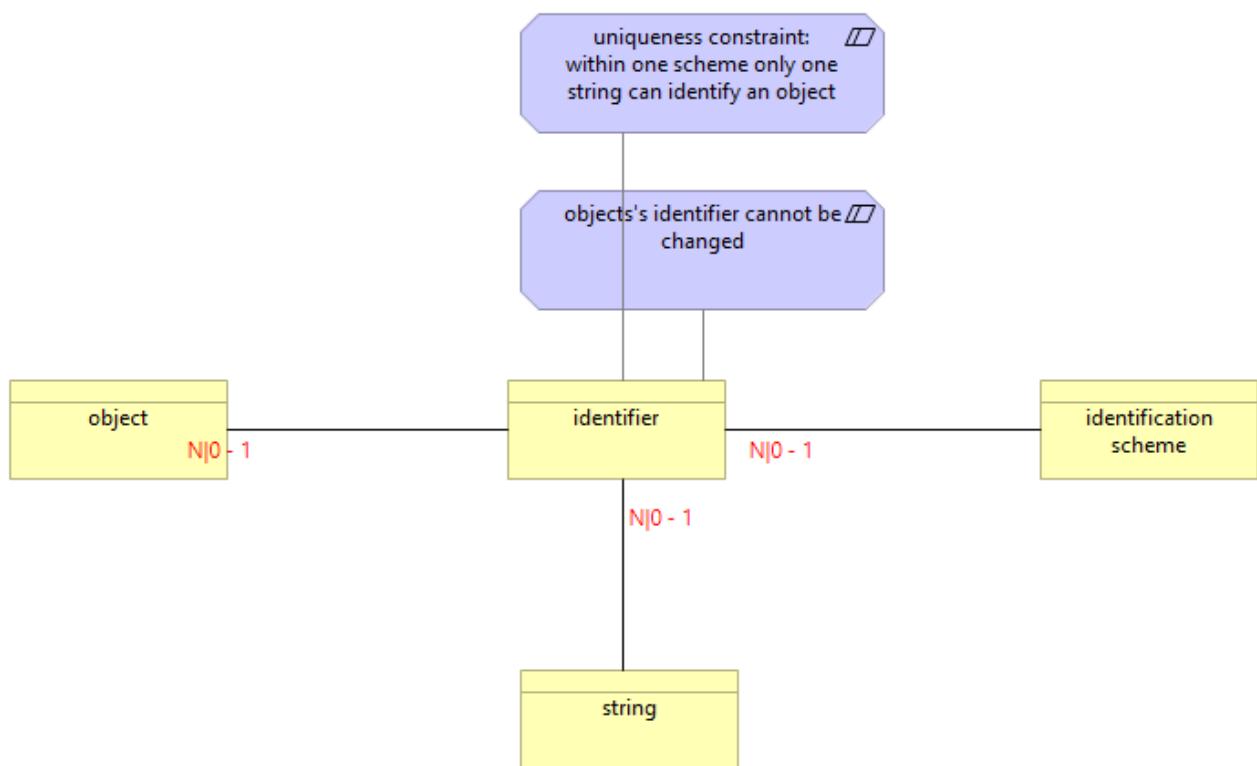
Martin Fowler



NAME



IDENTIFICATION SCHEME

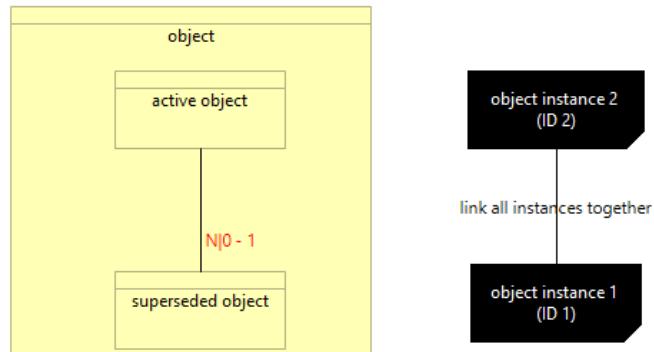


OBJECT MERGE

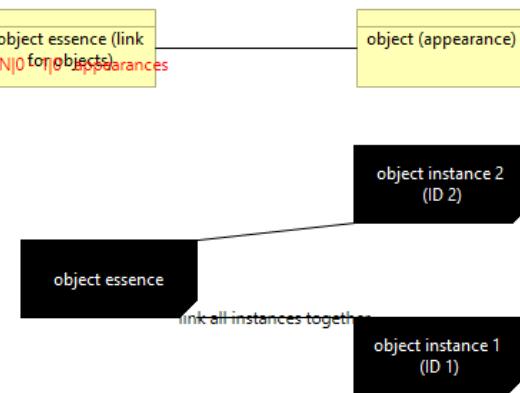
object merge strategy:
copy and replace



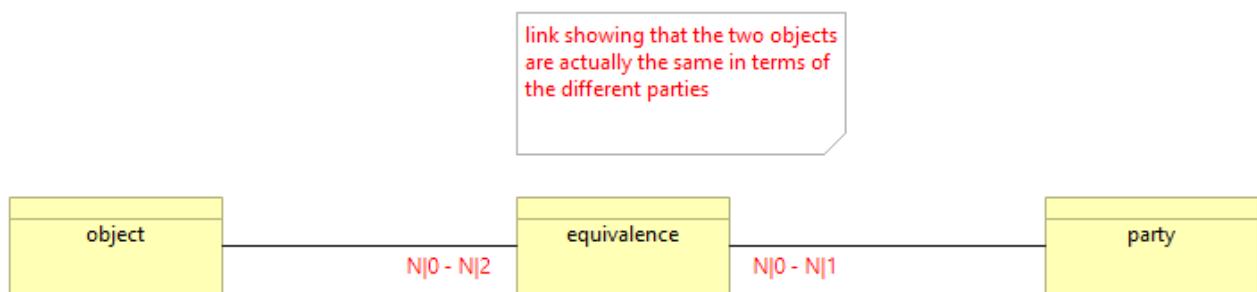
object merge strategy:
superseeded



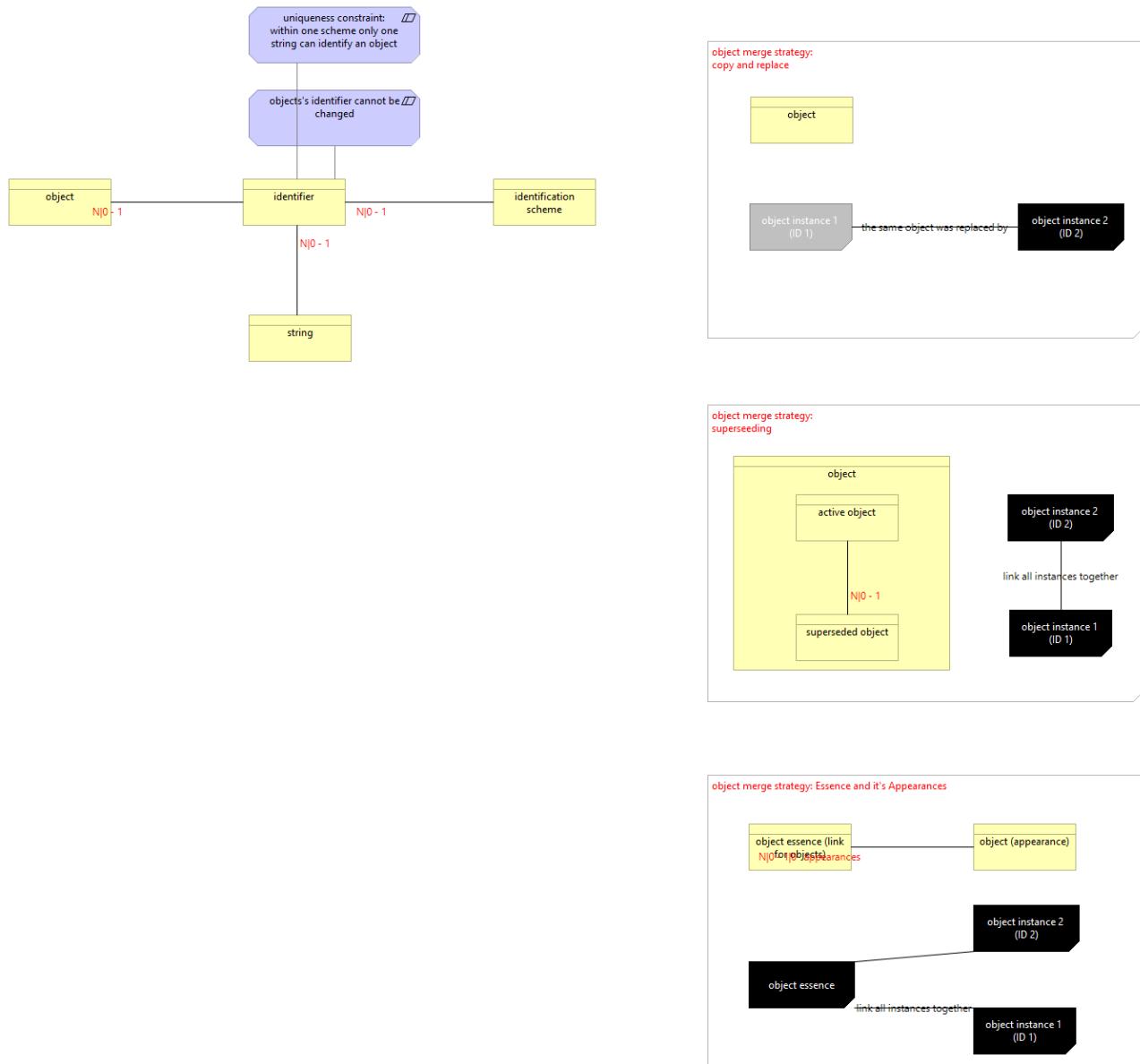
object merge strategy: Essence and its Appearances



OBJECT EQUIVALENCE



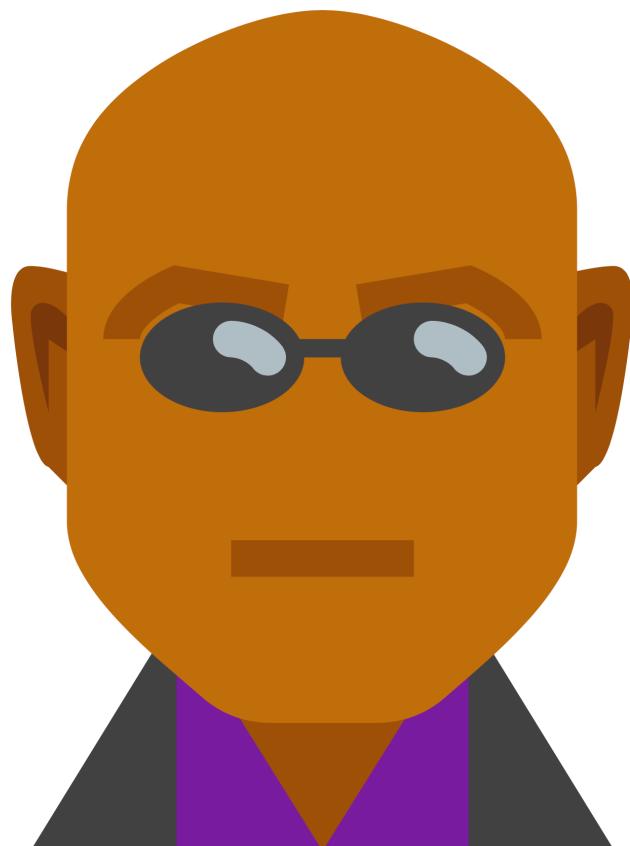
REFERRING TO OBJECTS



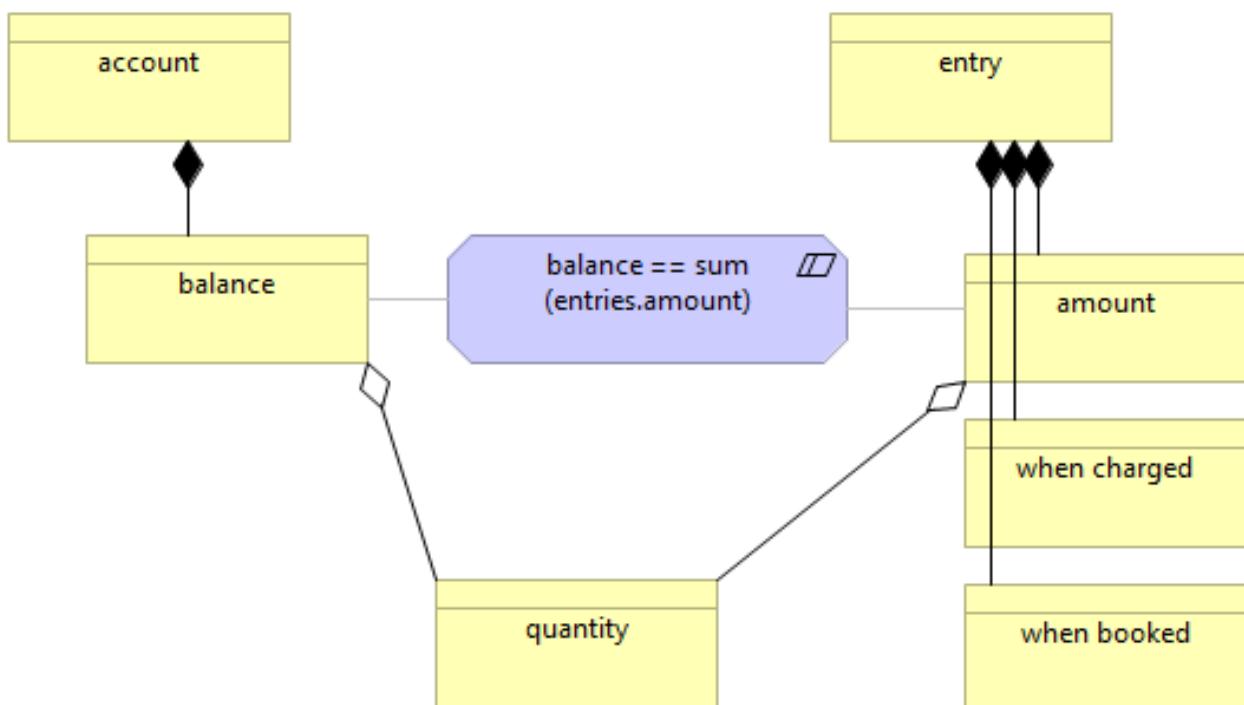
INVENTORY AND ACCOUNTING

ANALYSIS PATTERNS

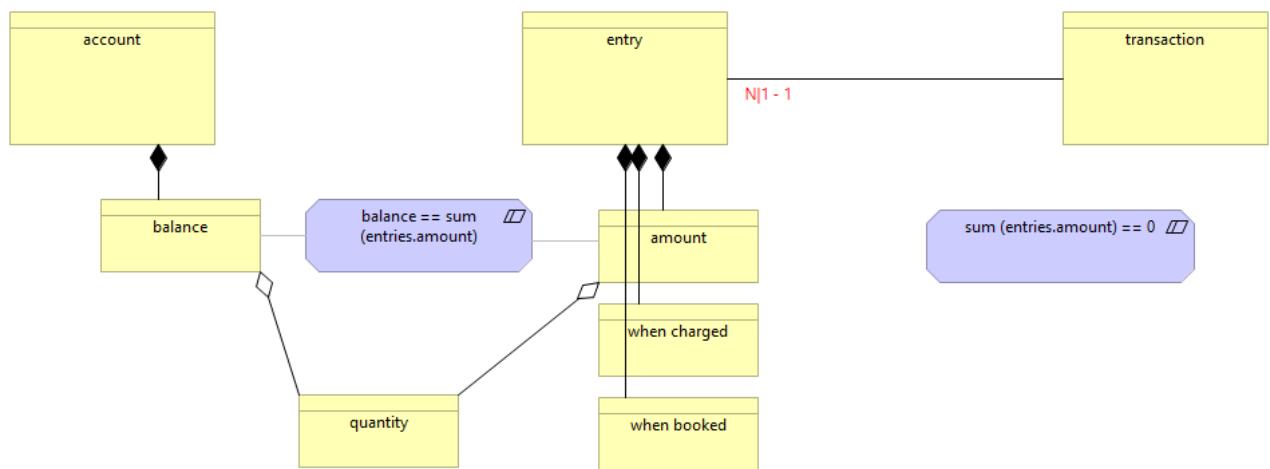
Martin Fowler



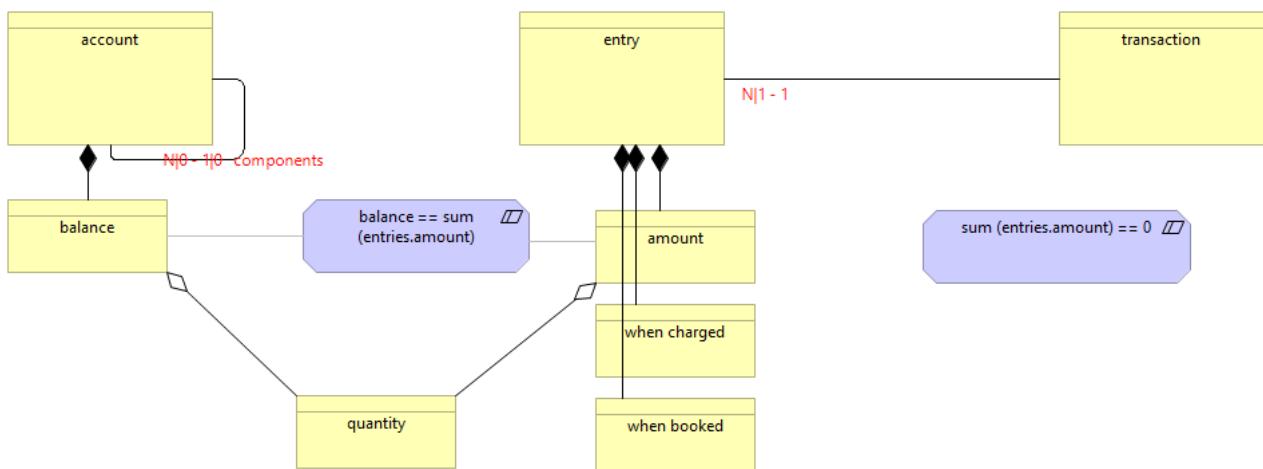
ACCOUNT



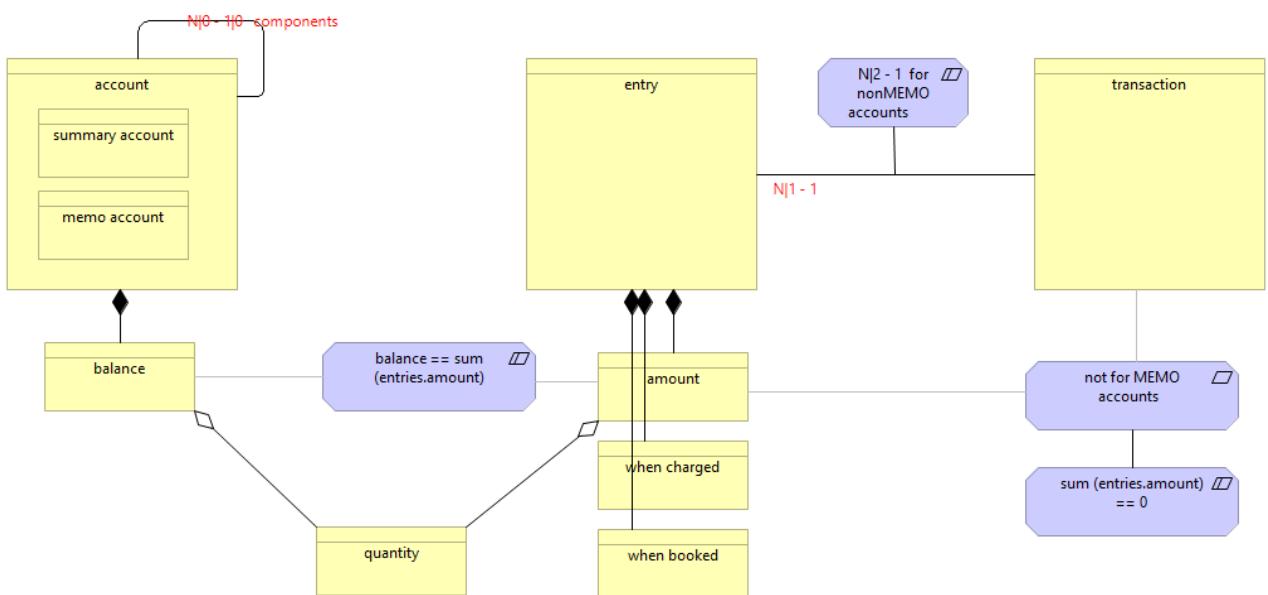
TRANSACTIONS



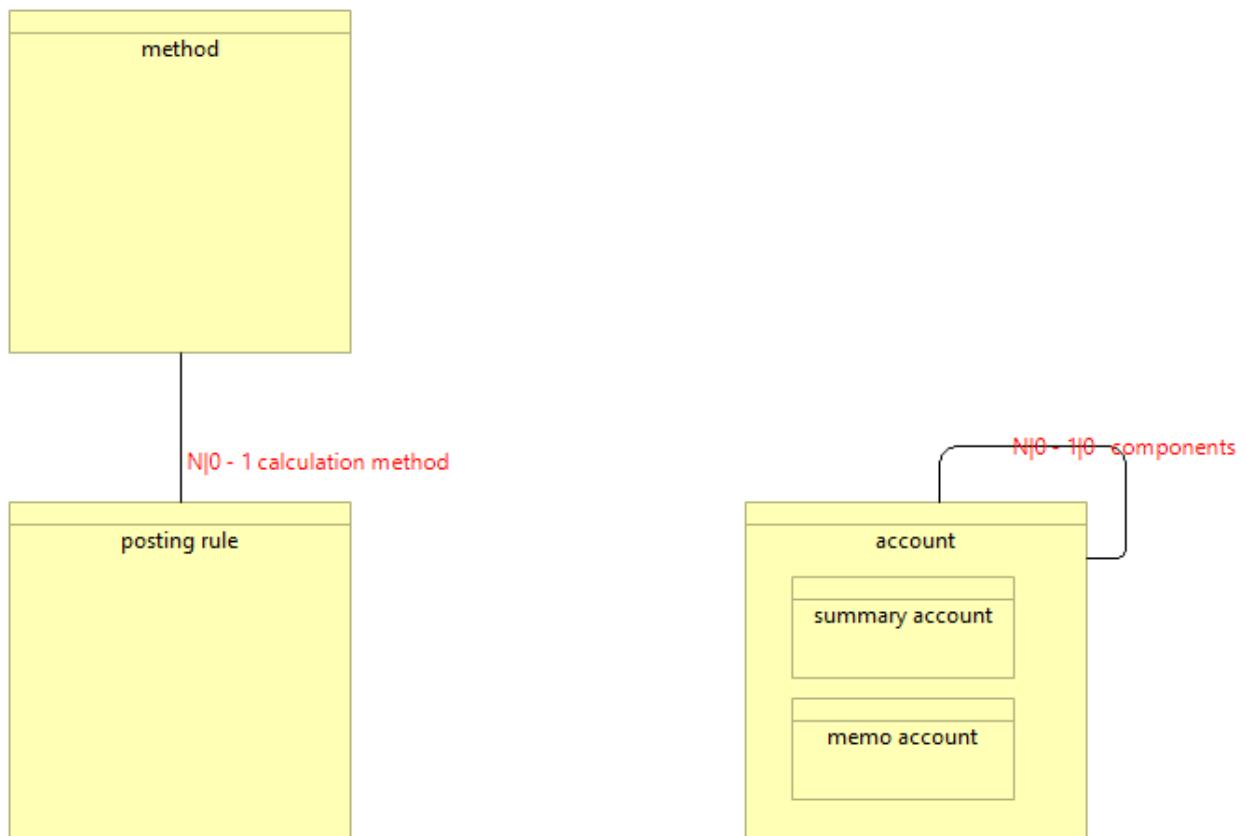
SUMMARY ACCOUNT



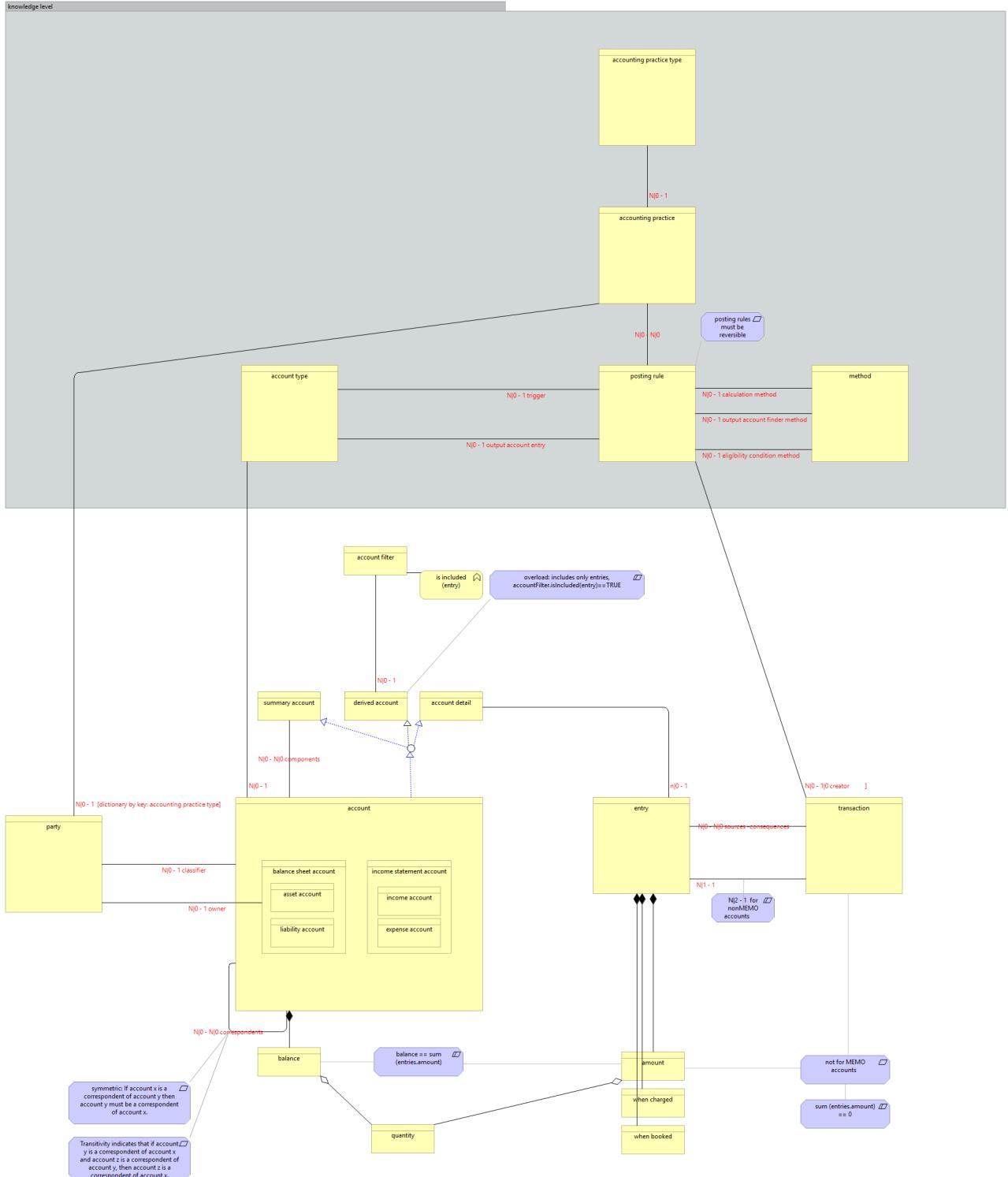
MEMO ACCOUNT



POSTING RULES



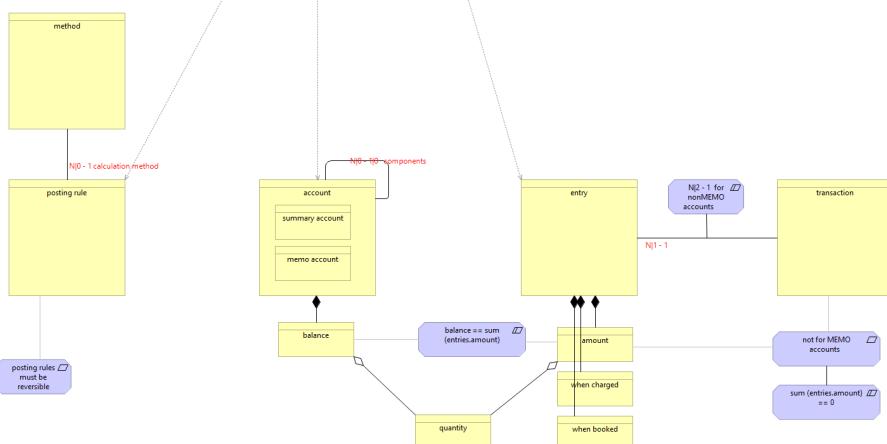
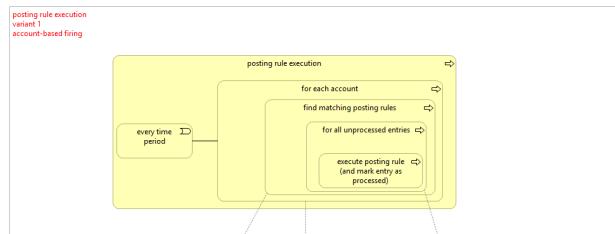
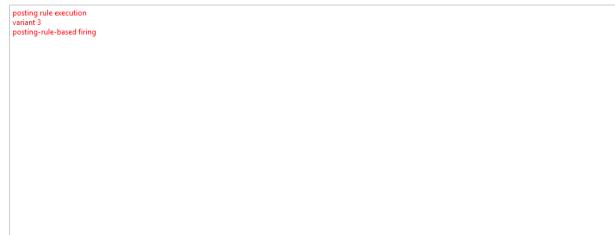
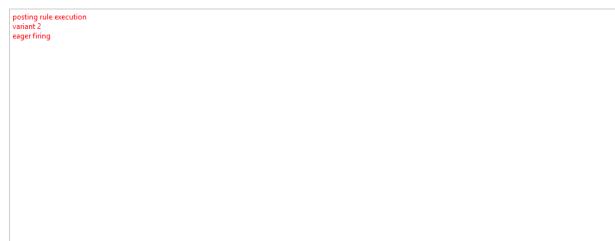
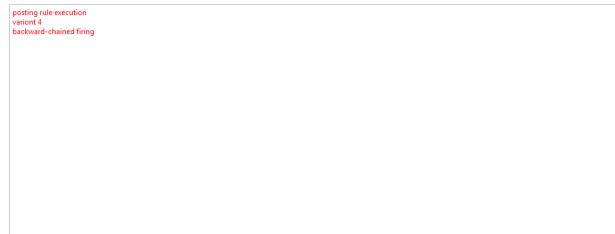
INVENTORY AND ACCOUNTING



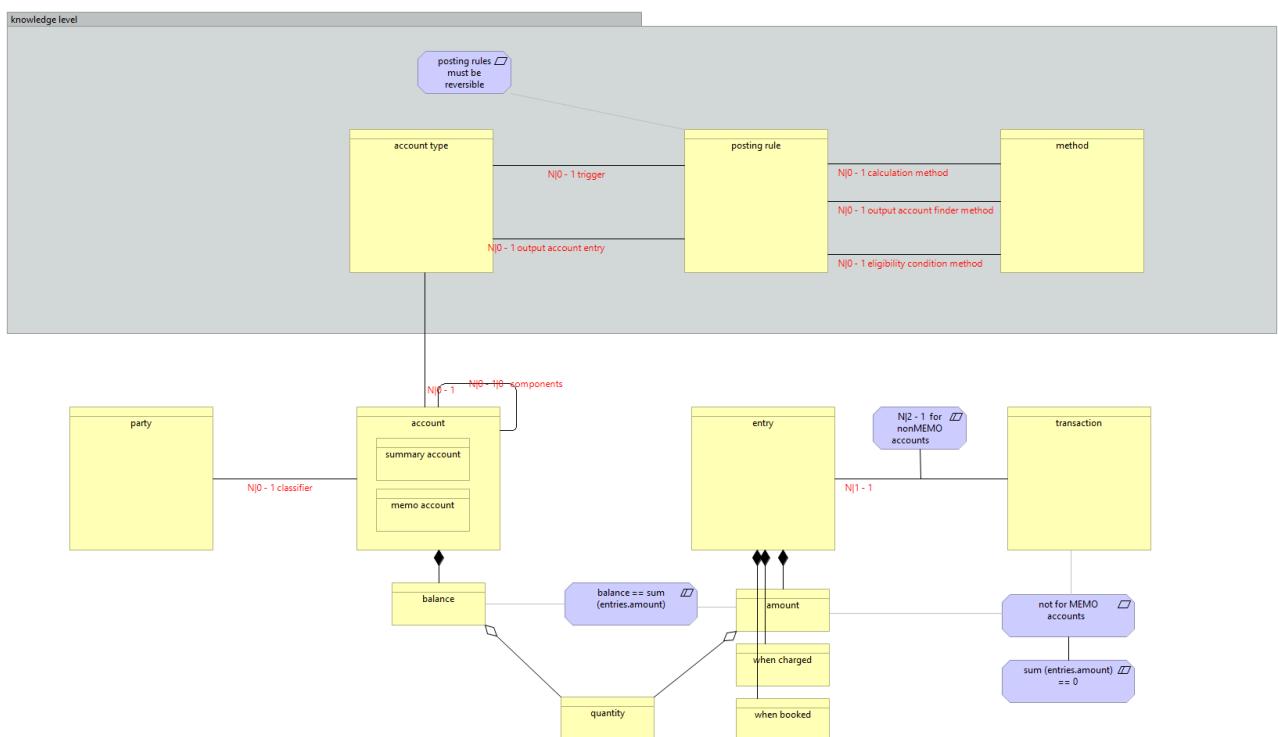
INDIVIDUAL INSTANCE METHOD



POSTING RULE EXECUTION



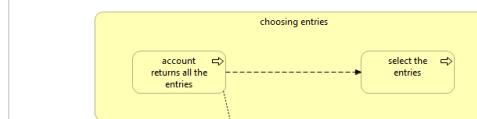
POSTING RULES FOR MANY ACCOUNTS



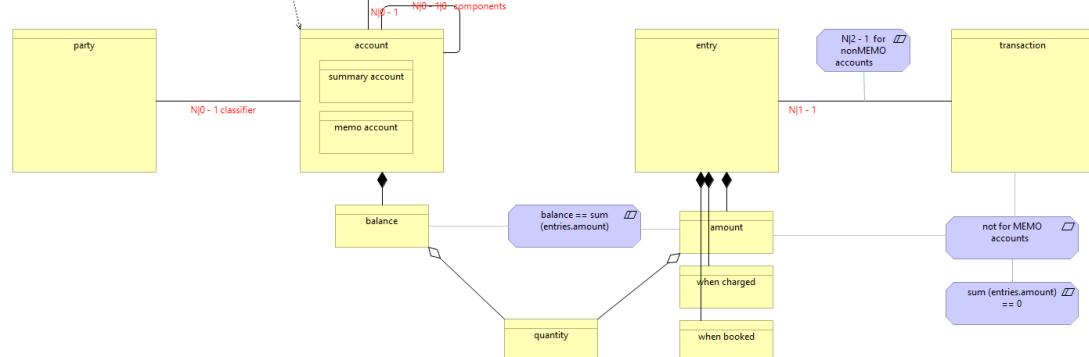
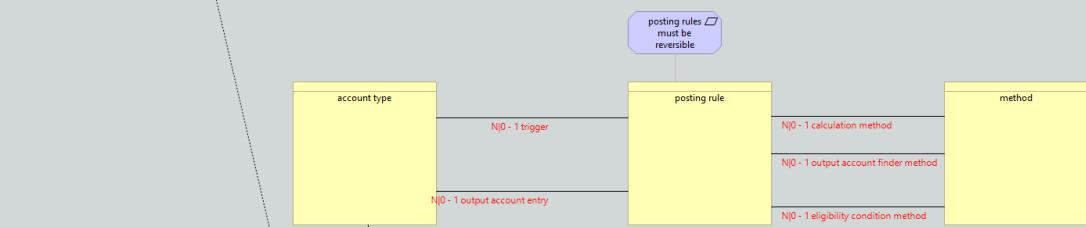
CHOOSING ENTRIES

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

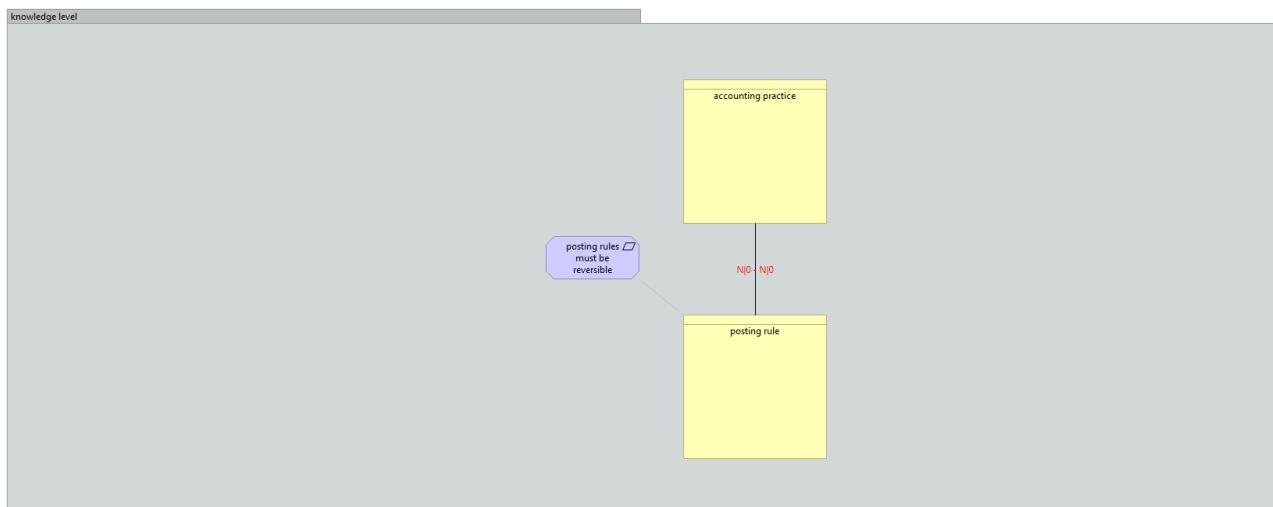
selection of entries for the application of the posing rules:
variant 1
The account returns all the entries, and the client selects the entries it needs.



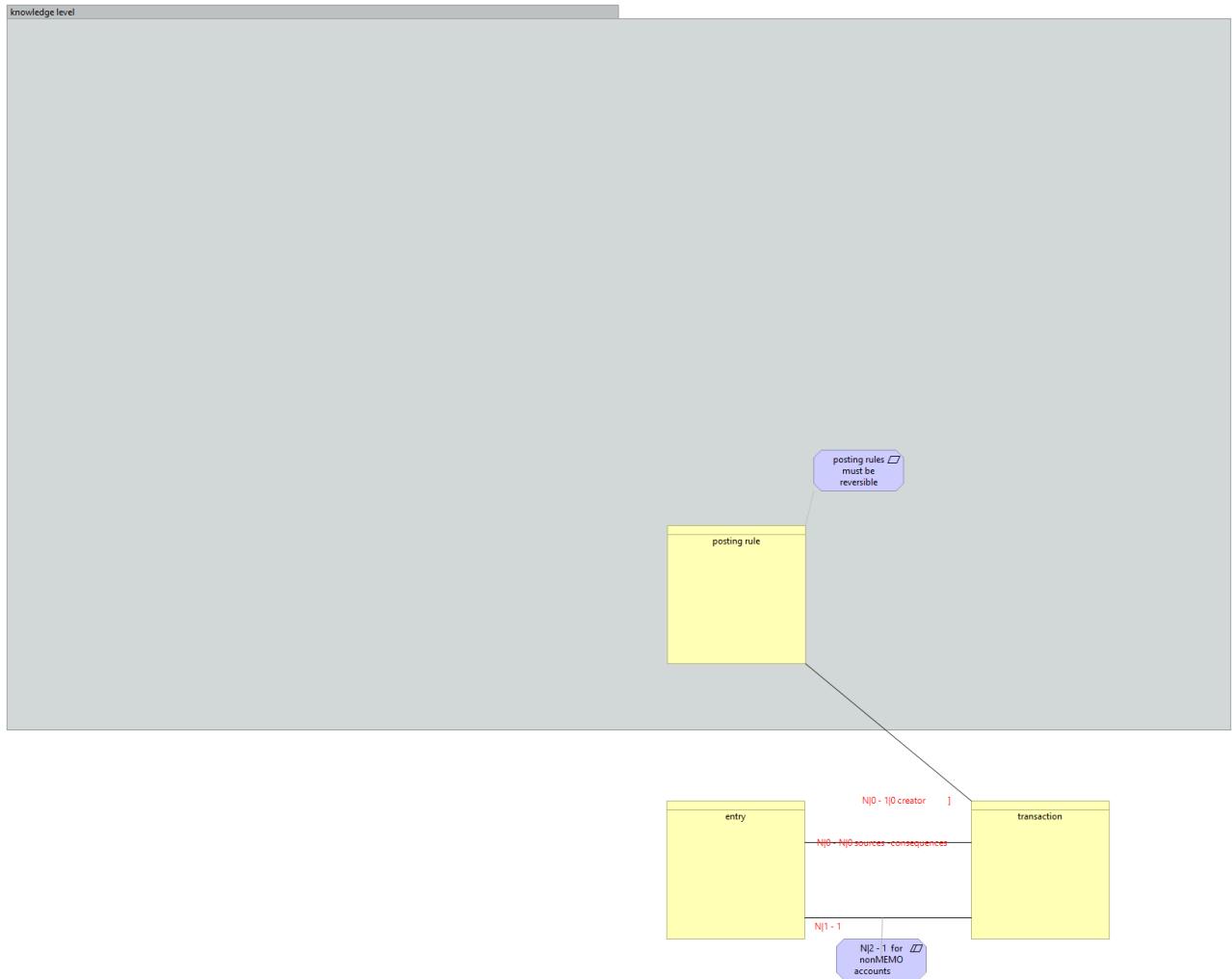
knowledge level



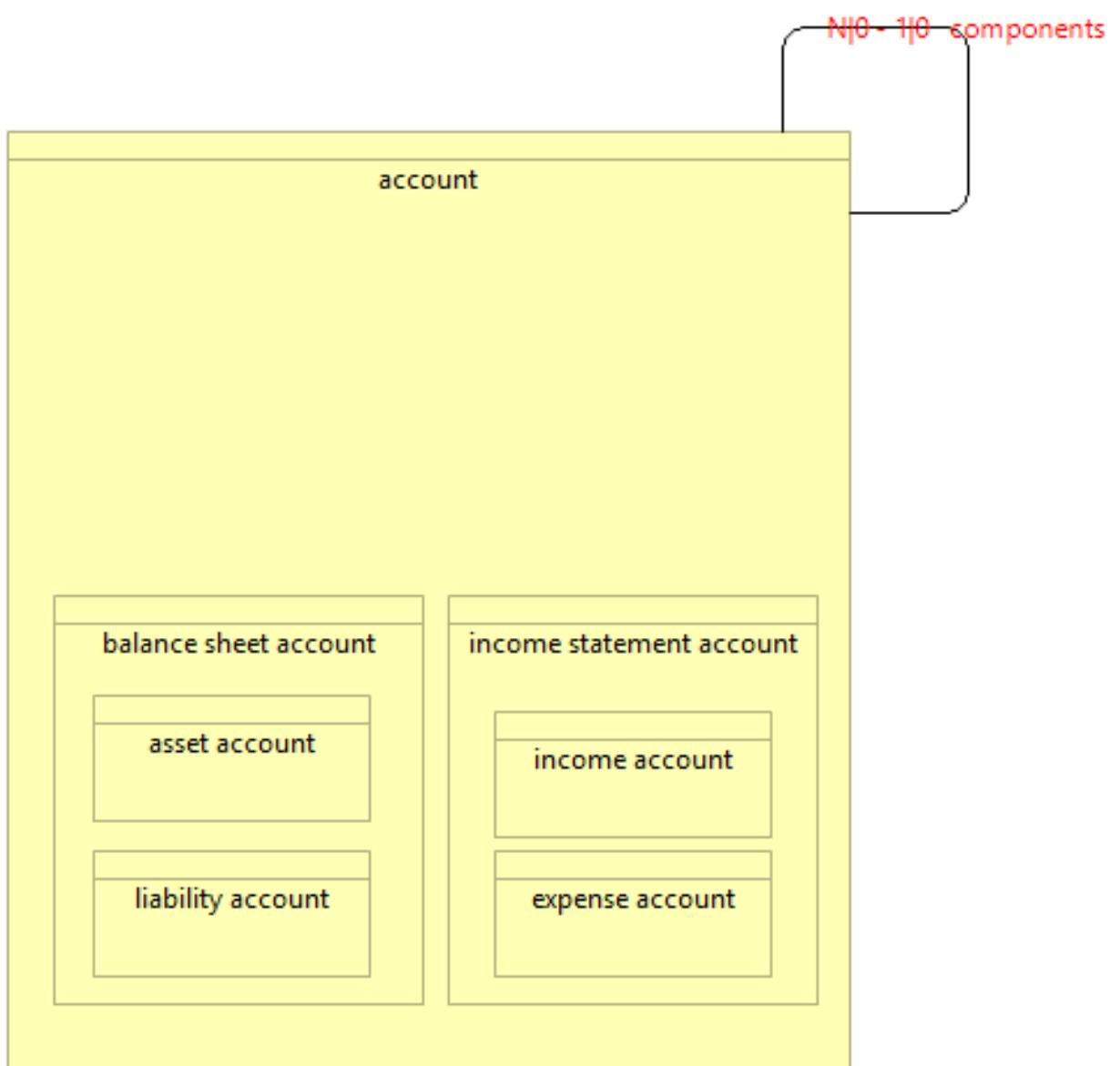
ACCOUNTING PRACTICE



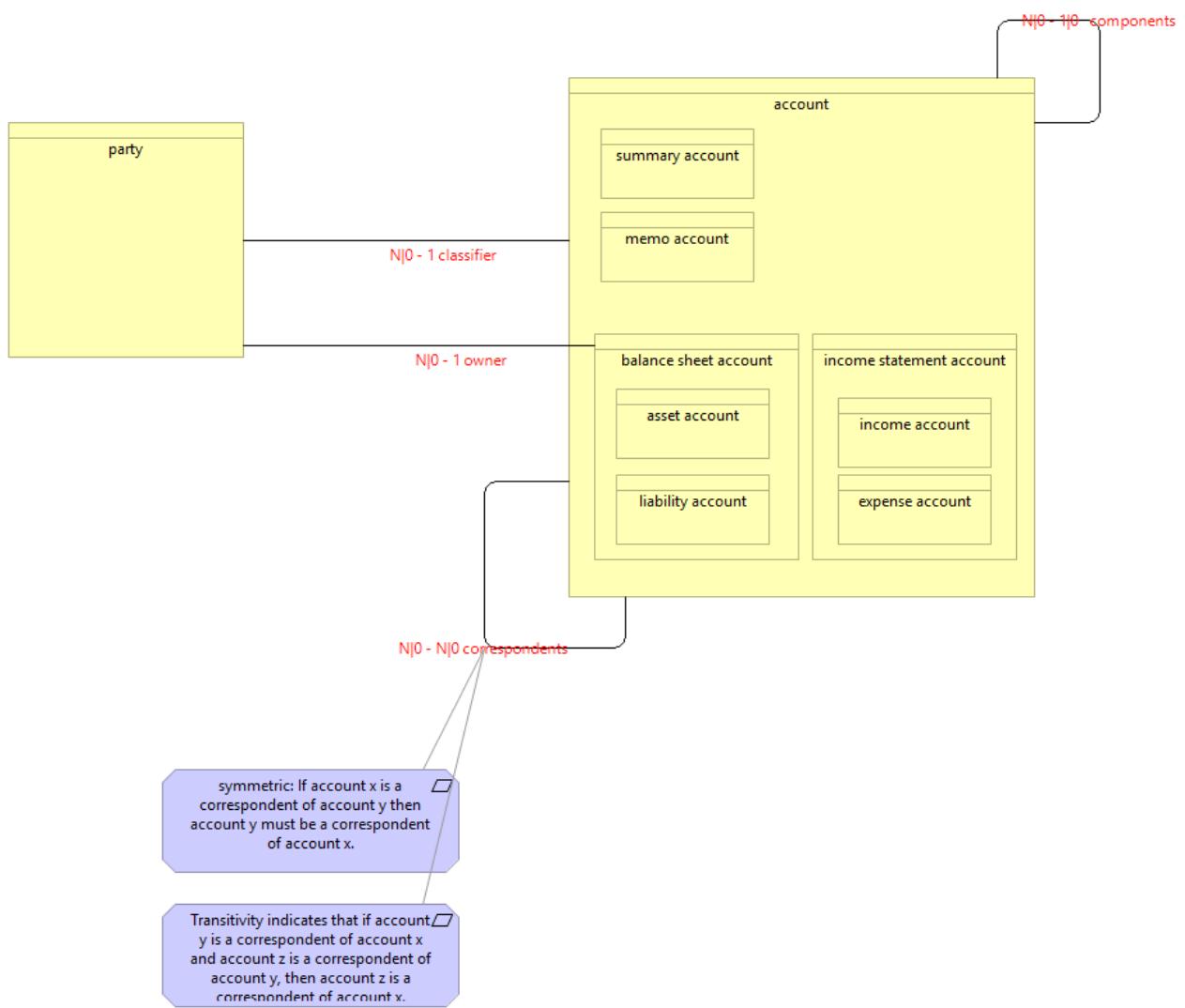
SOURCES OF AN ENTRY



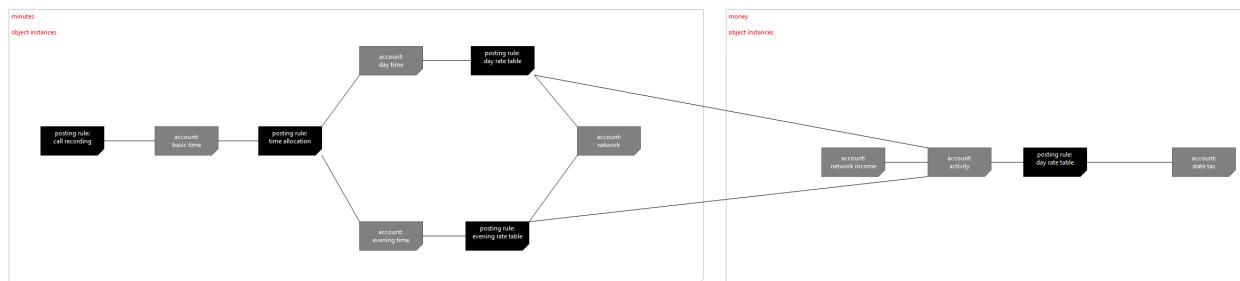
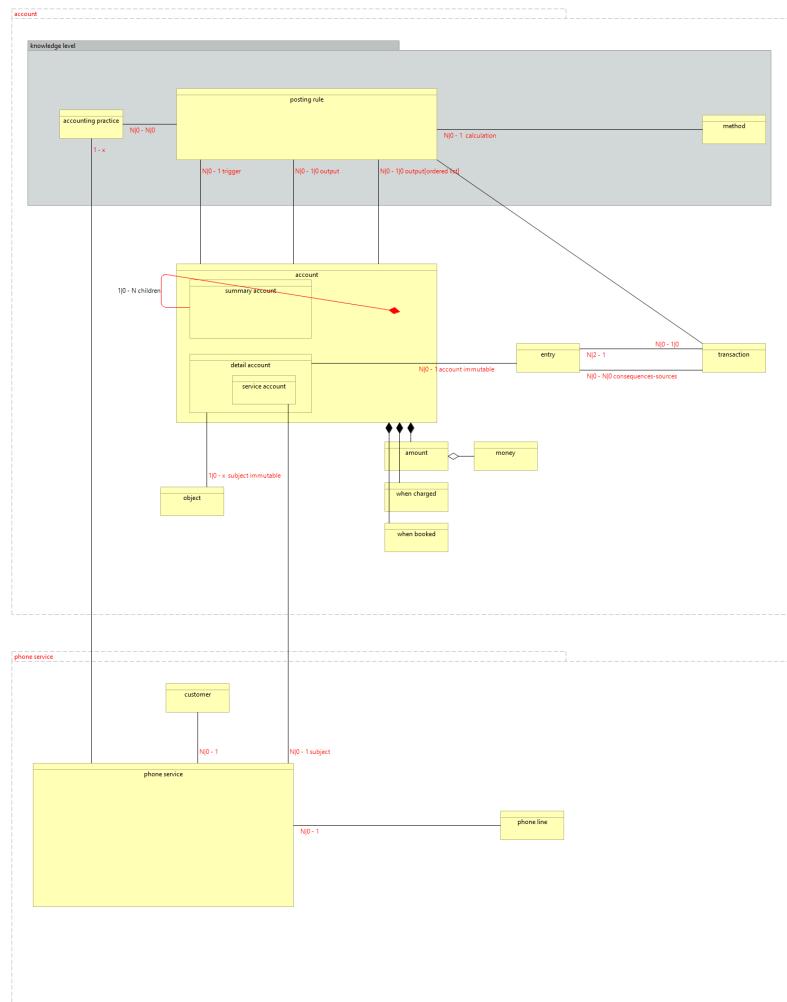
BALANCE SHEET AND INCOME STATEMENT



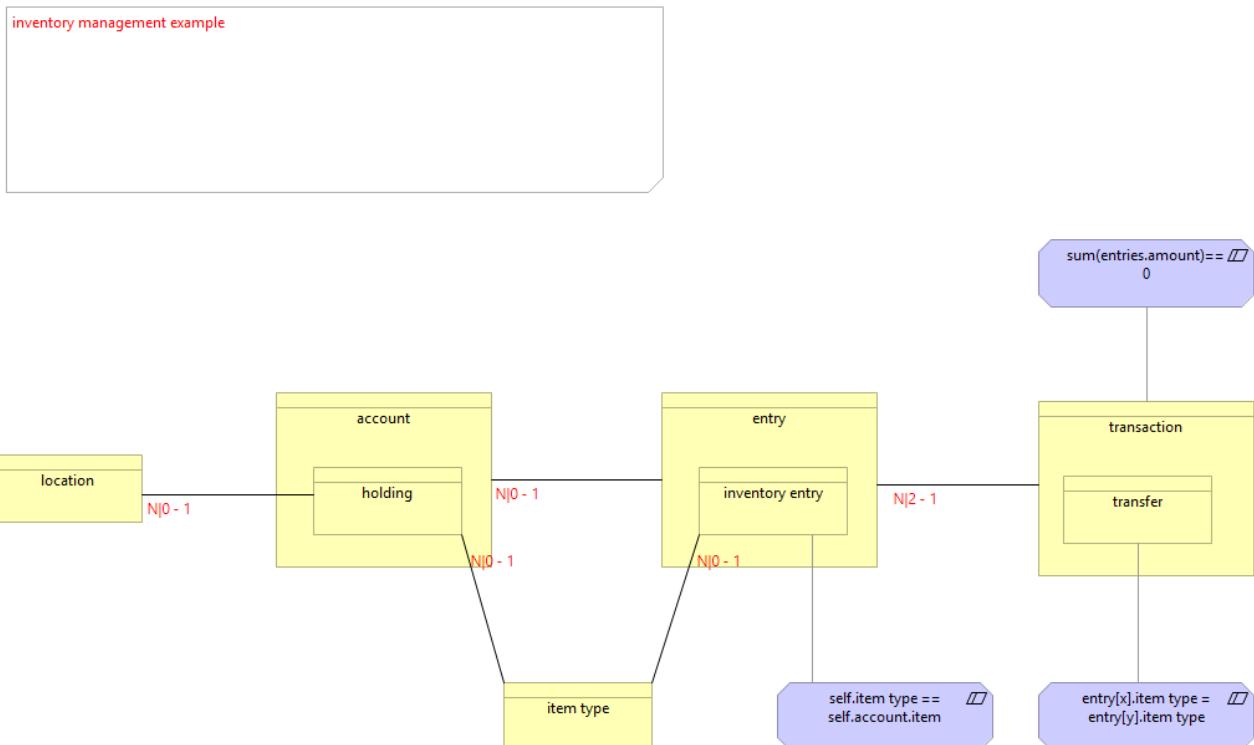
CORRESPONDING ACCOUNT



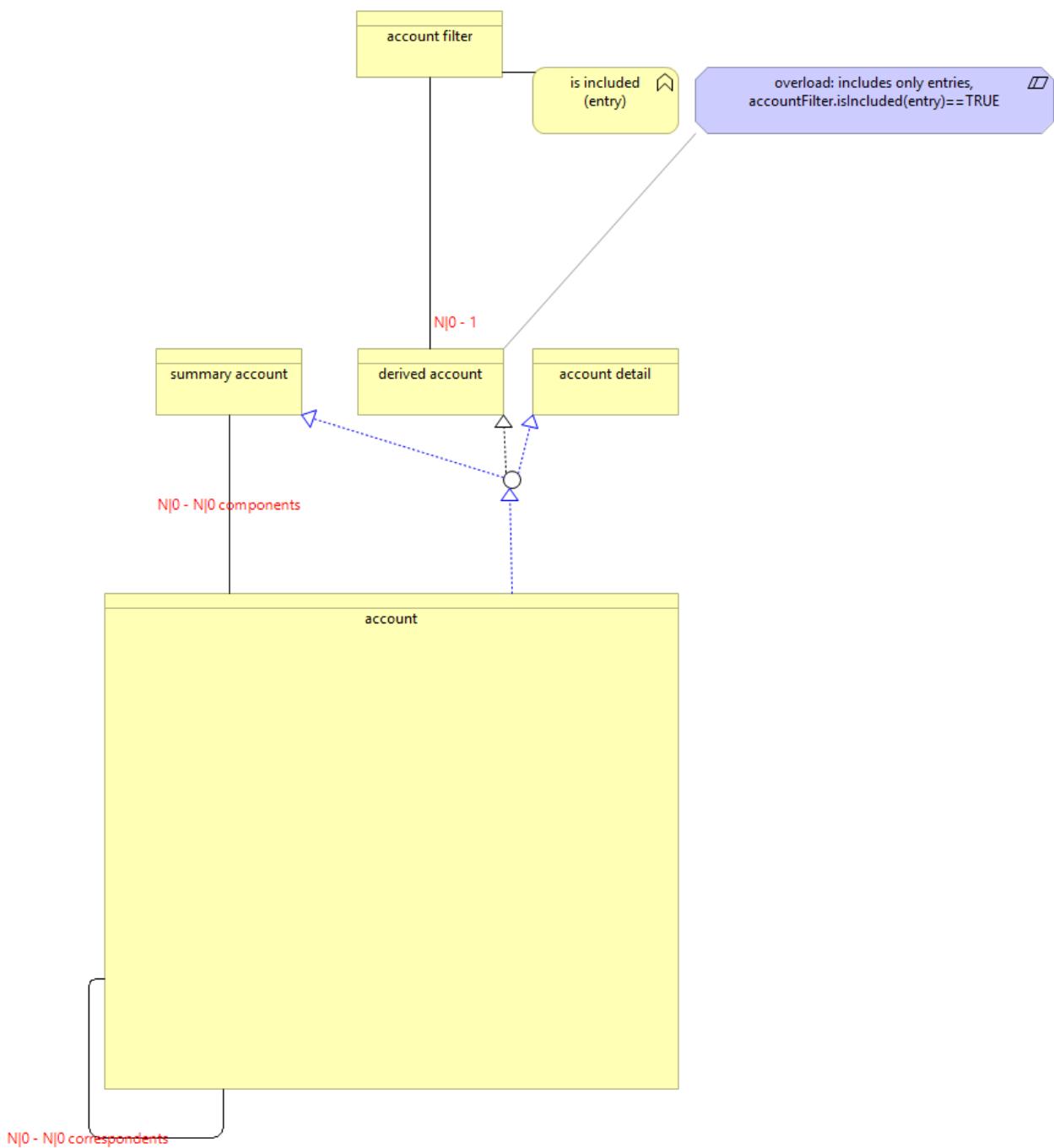
SPECIALIZED ACCOUNT MODEL (BILLING EXAMPLE)



SPECIALIZED ACCOUNT MODEL (INVENTORY EXAMPLE)



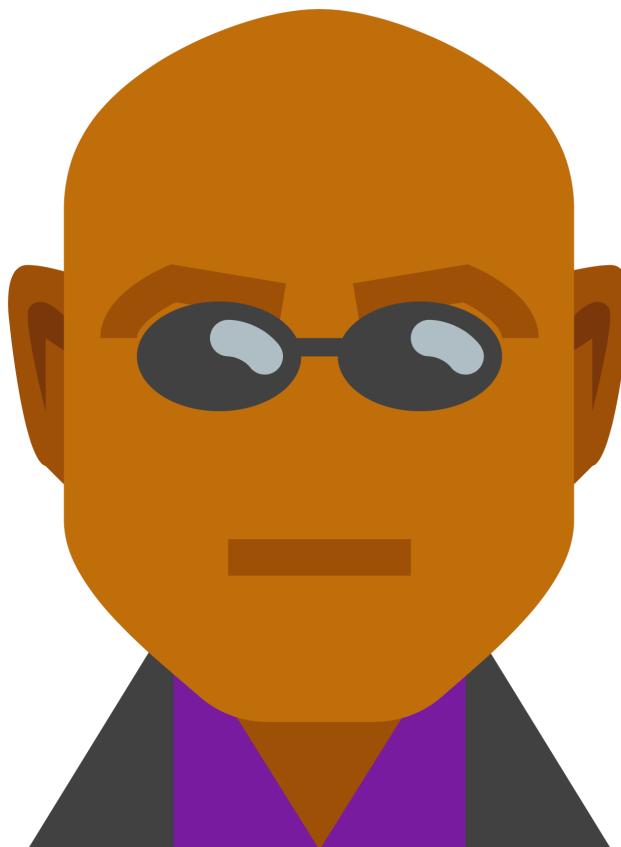
BOOKING ENTRIES TO MULTIPLE ACCOUNTS



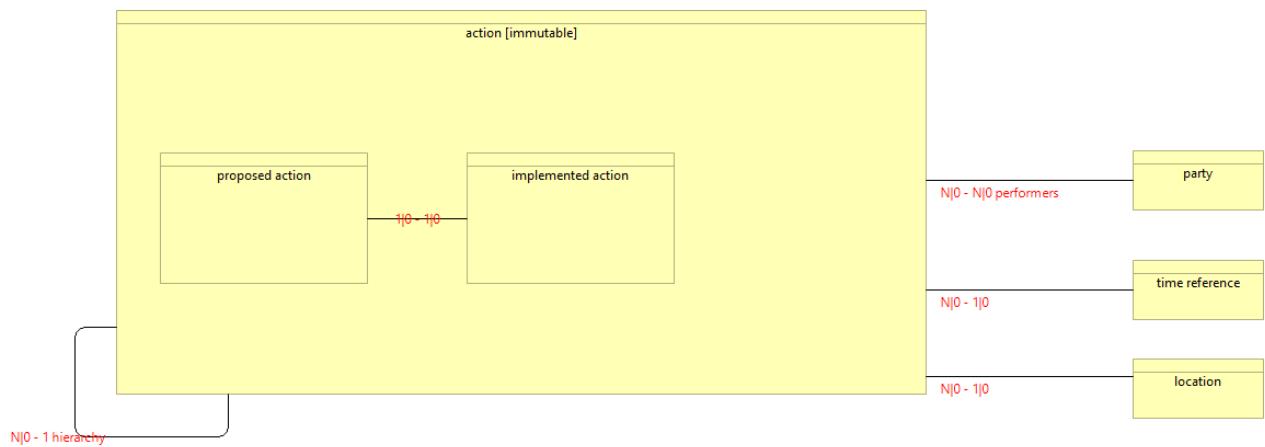
PLANNING

ANALYSIS PATTERNS

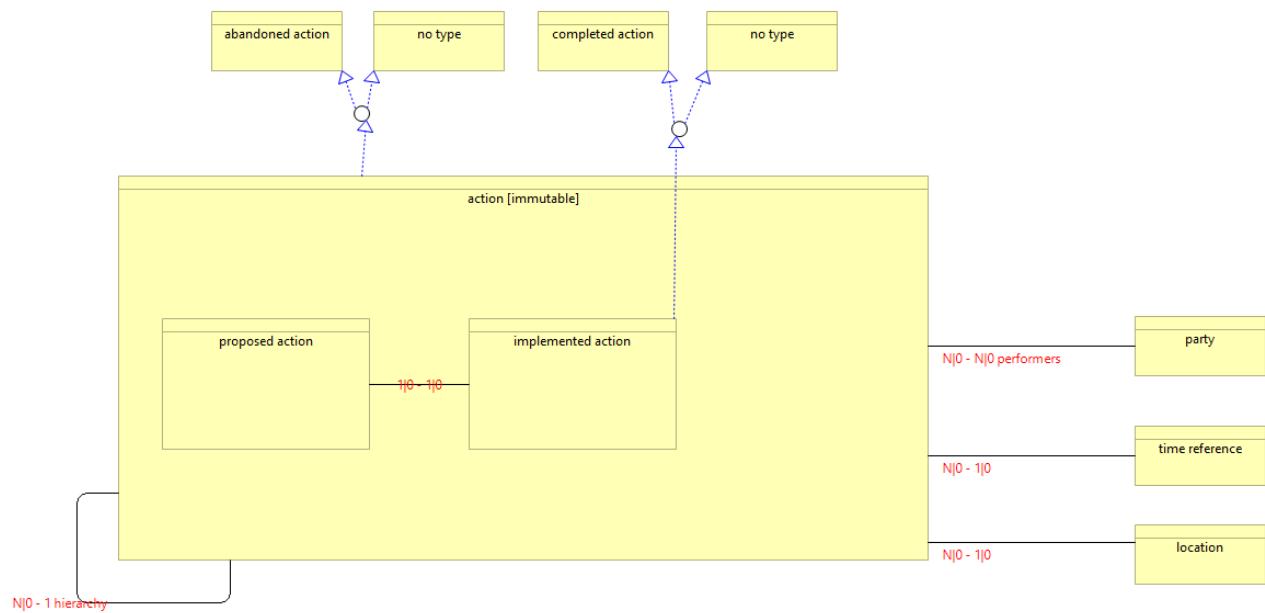
Martin Fowler



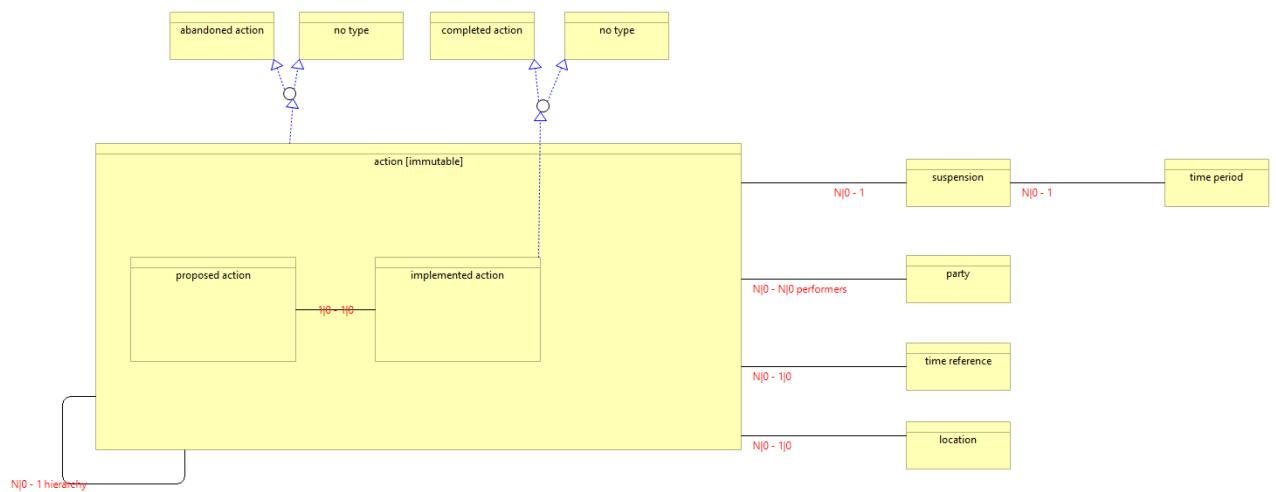
PROPOSED AND IMPLEMENTED ACTION



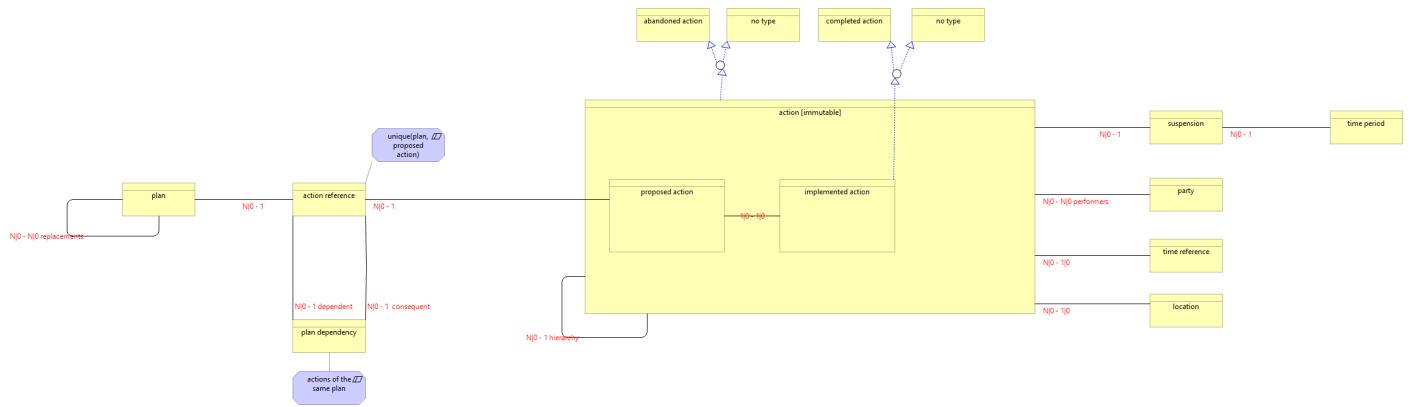
COMPLETED AND ABANDONED ACTIONS



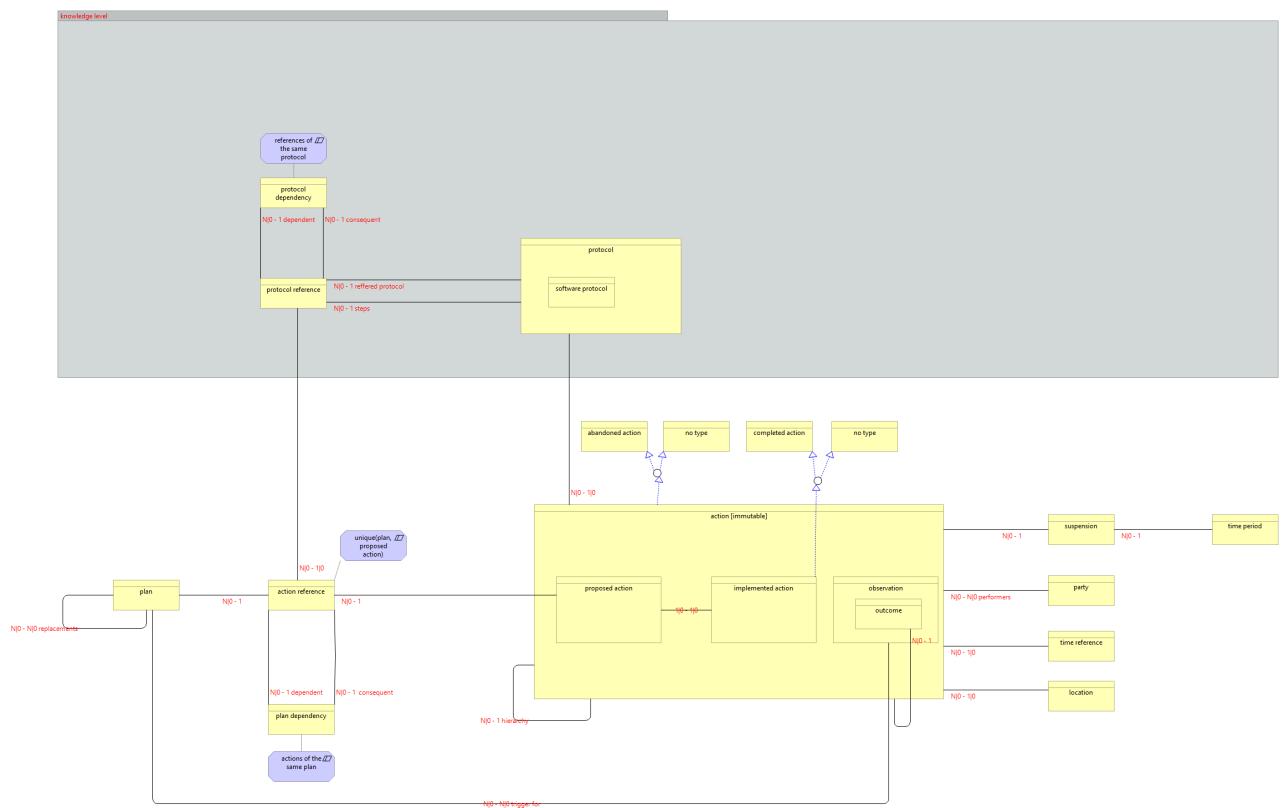
SUSPENSION



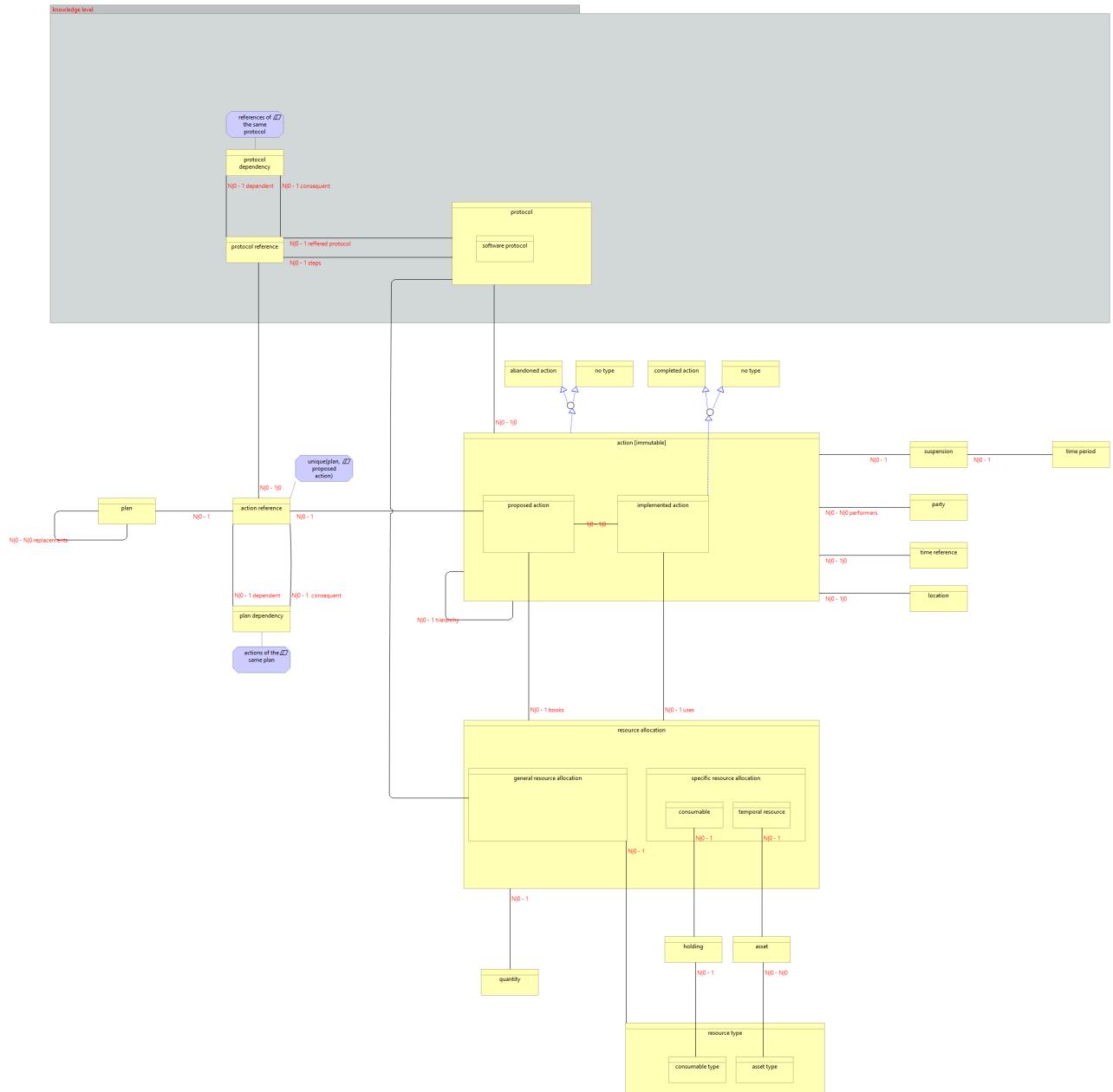
PLAN



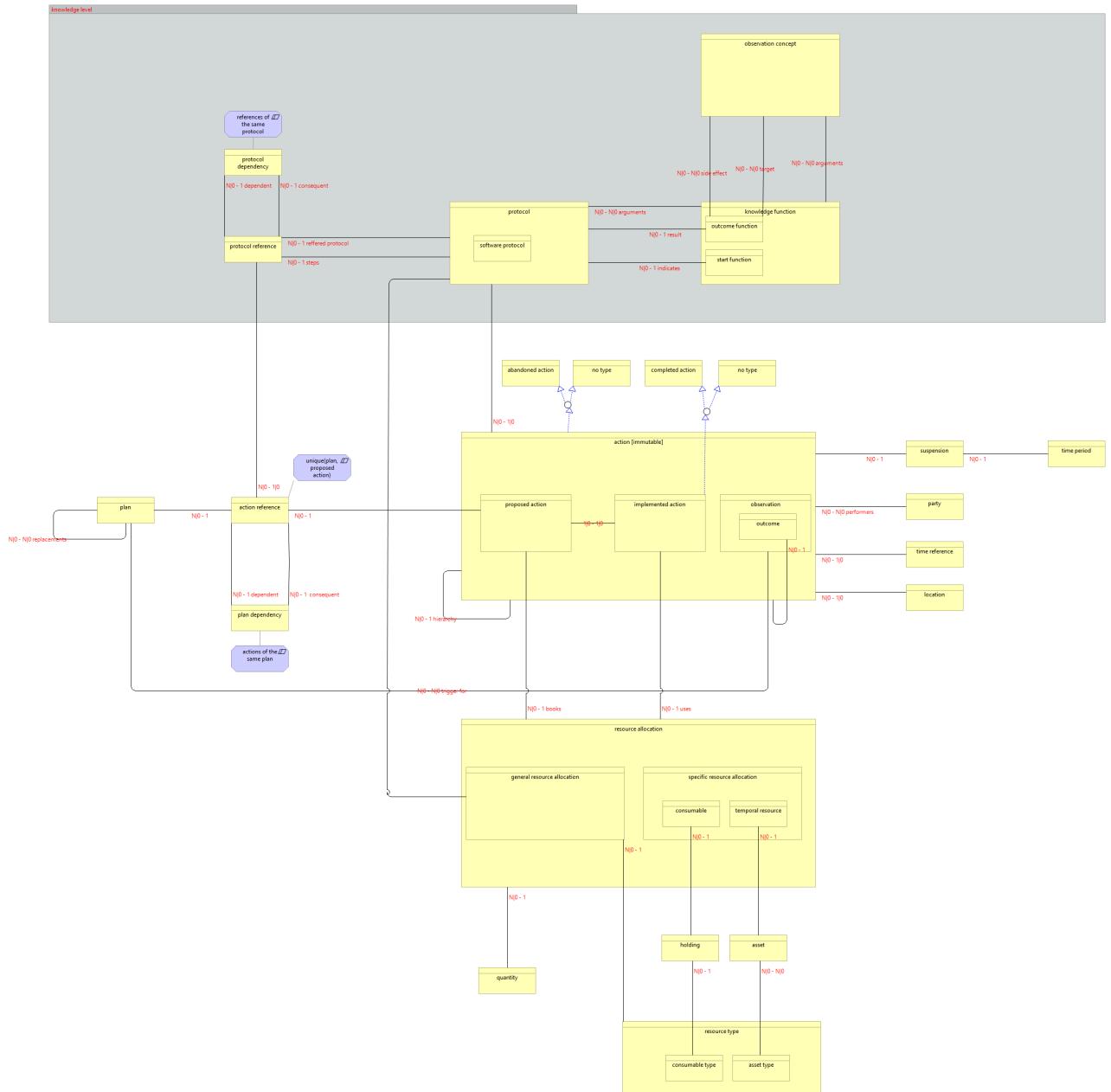
PROTOCOL



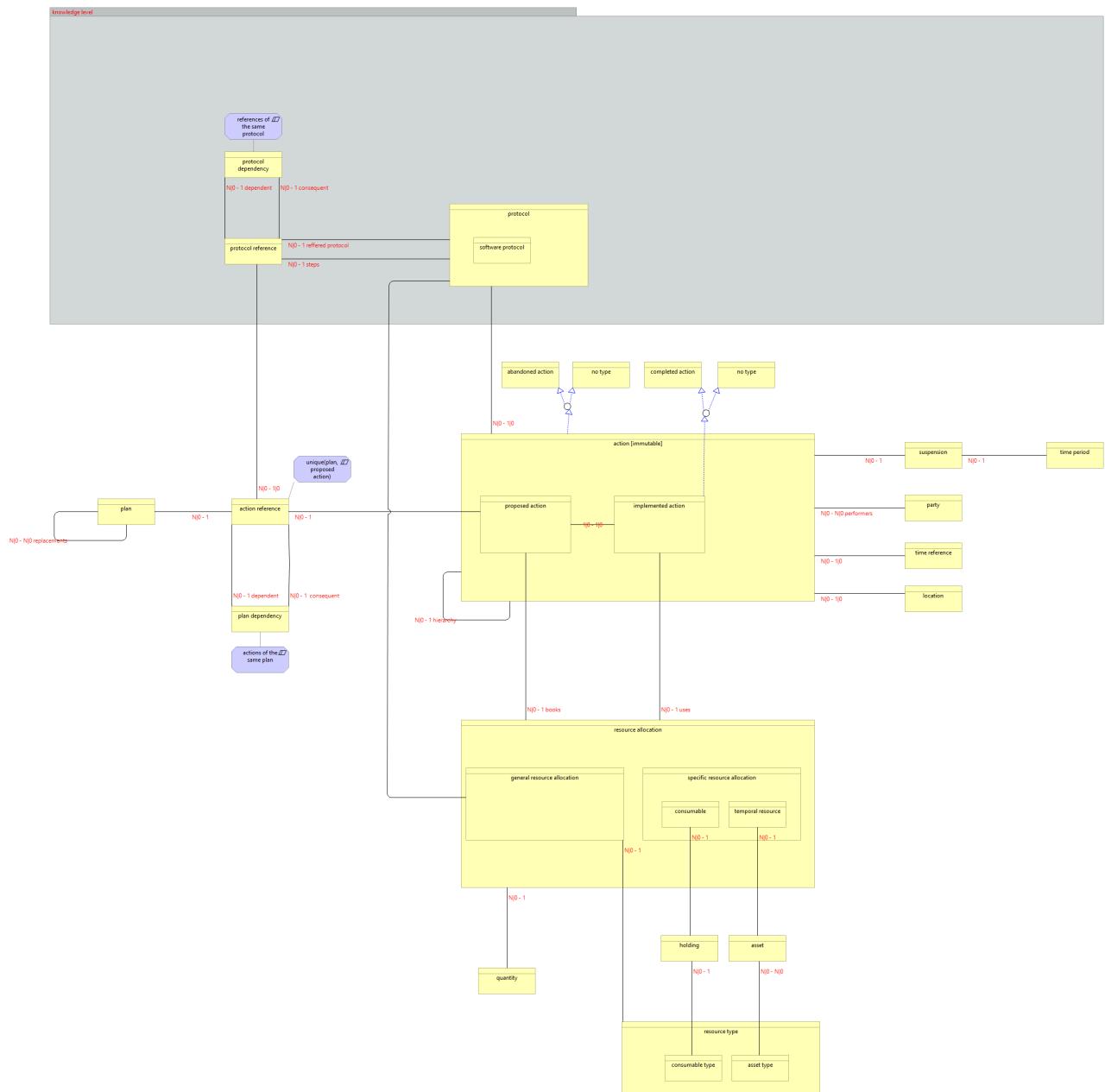
RESOURCE ALLOCATION



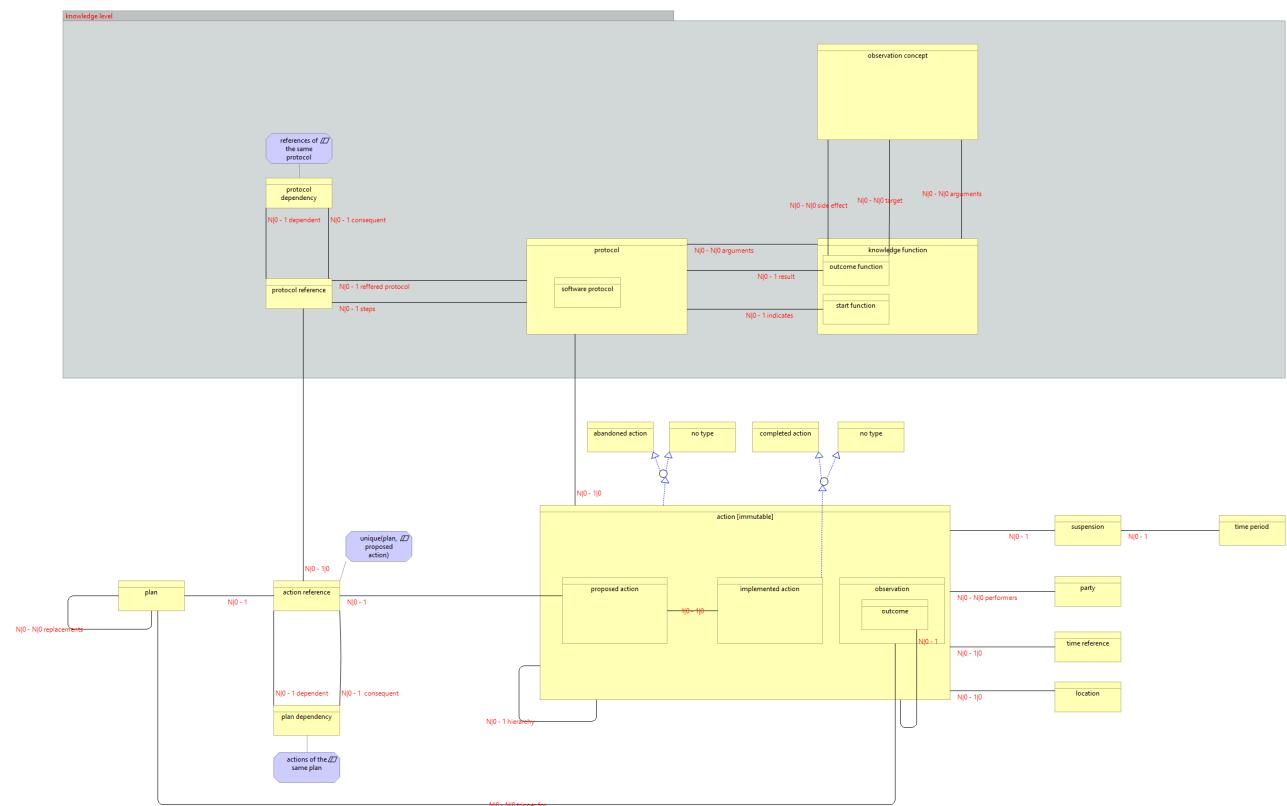
PLANNING



PLANNING (NO OUTCOME)



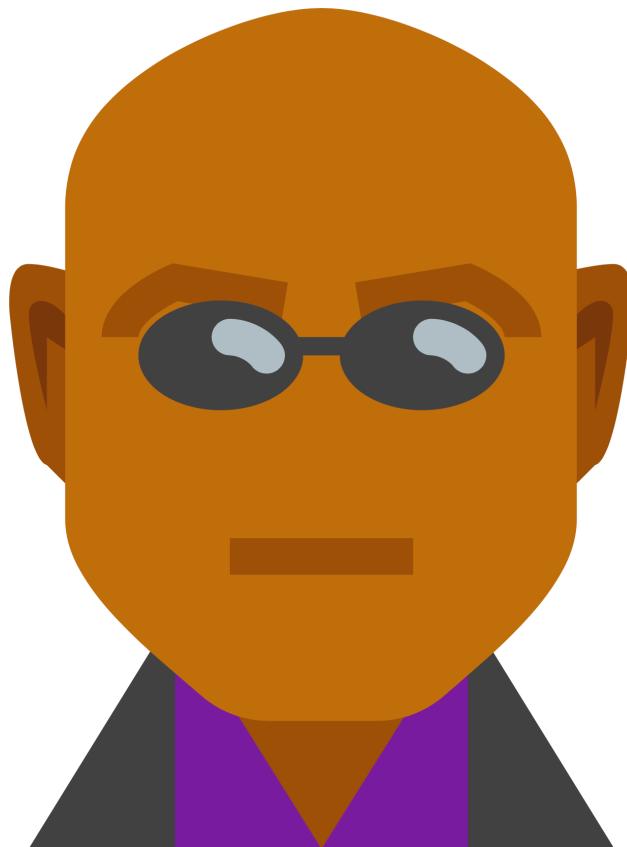
OUTCOME AND START FUNCTIONS



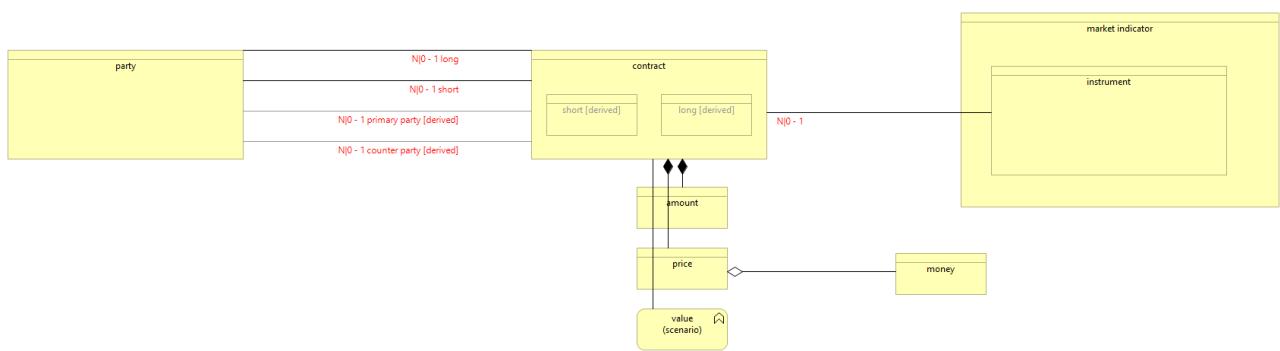
TRADING

ANALYSIS PATTERNS

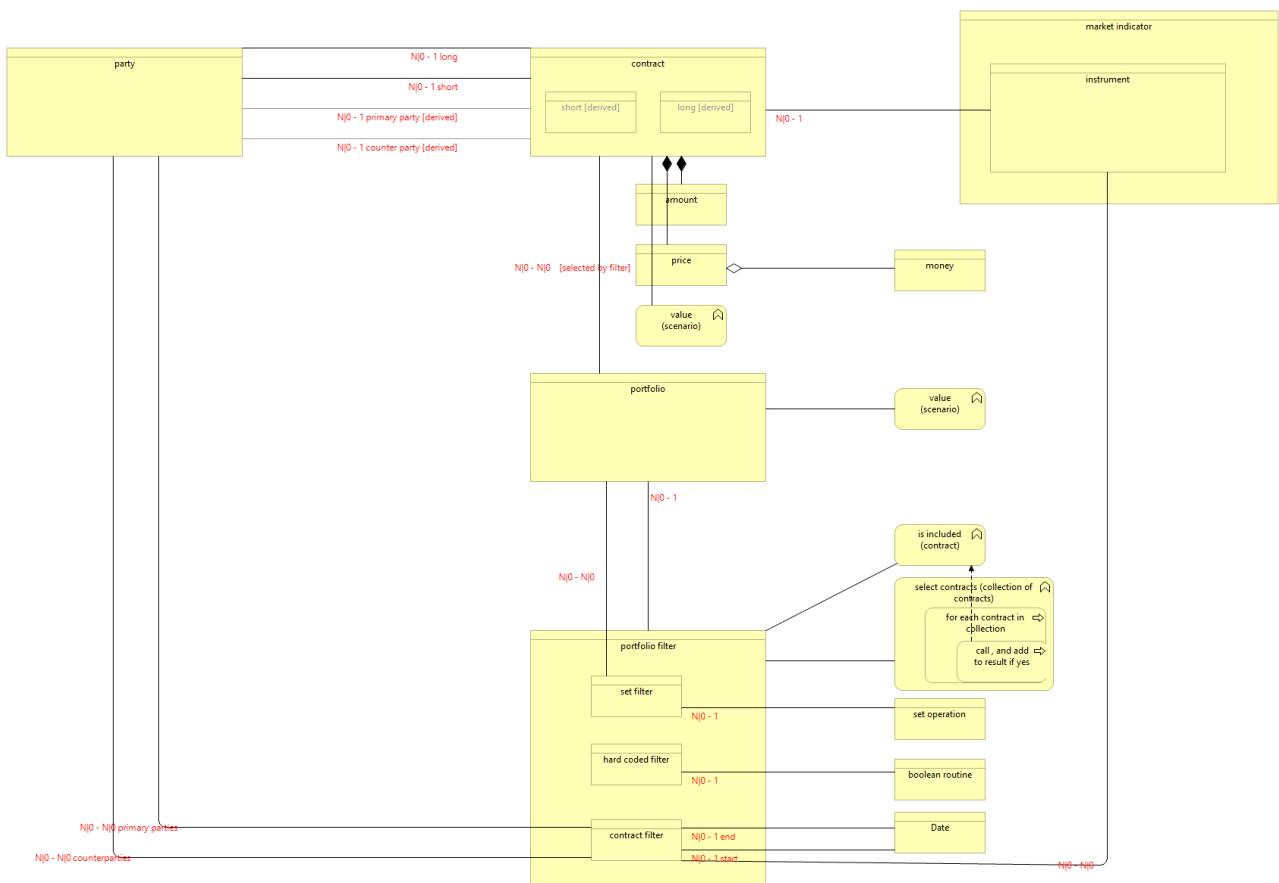
Martin Fowler



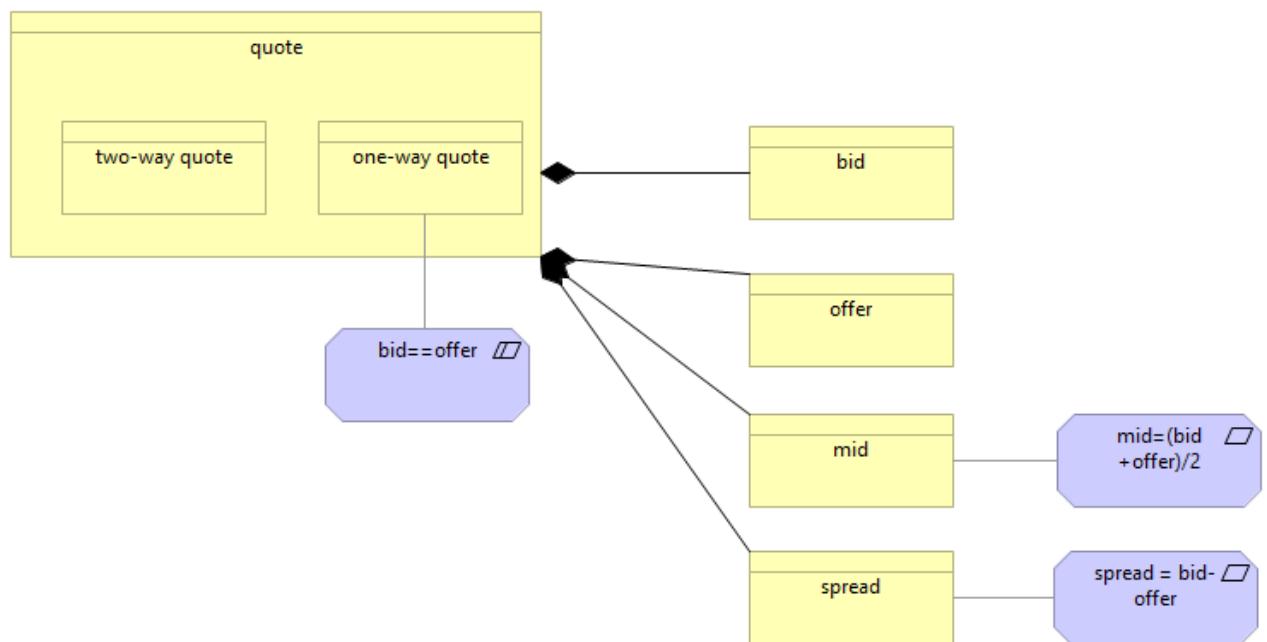
CONTRACT



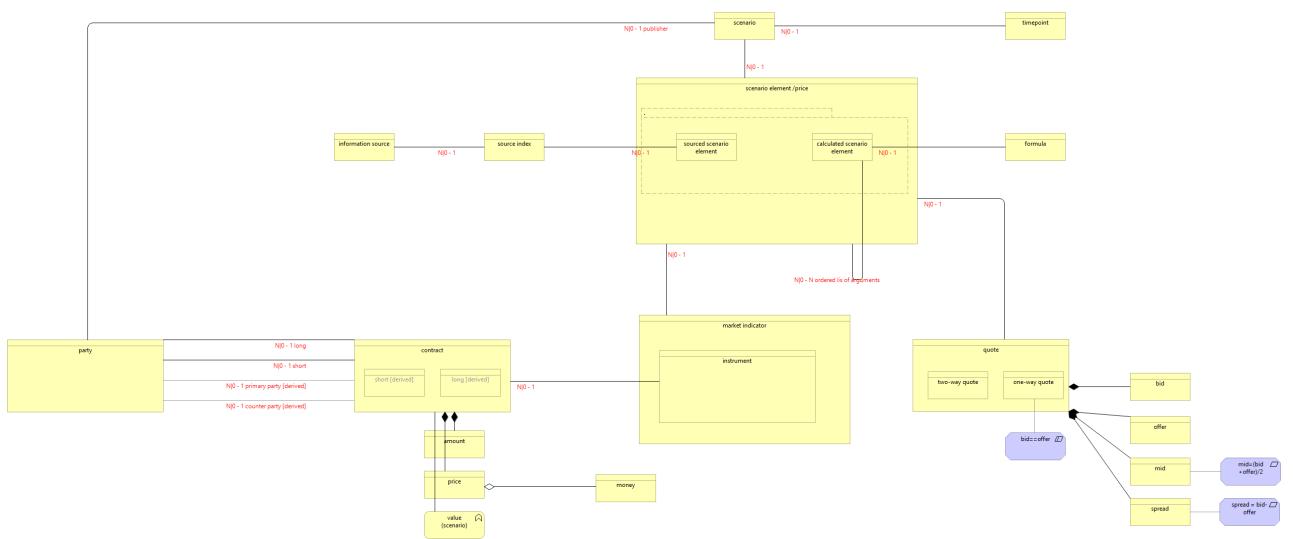
PORTFOLIO



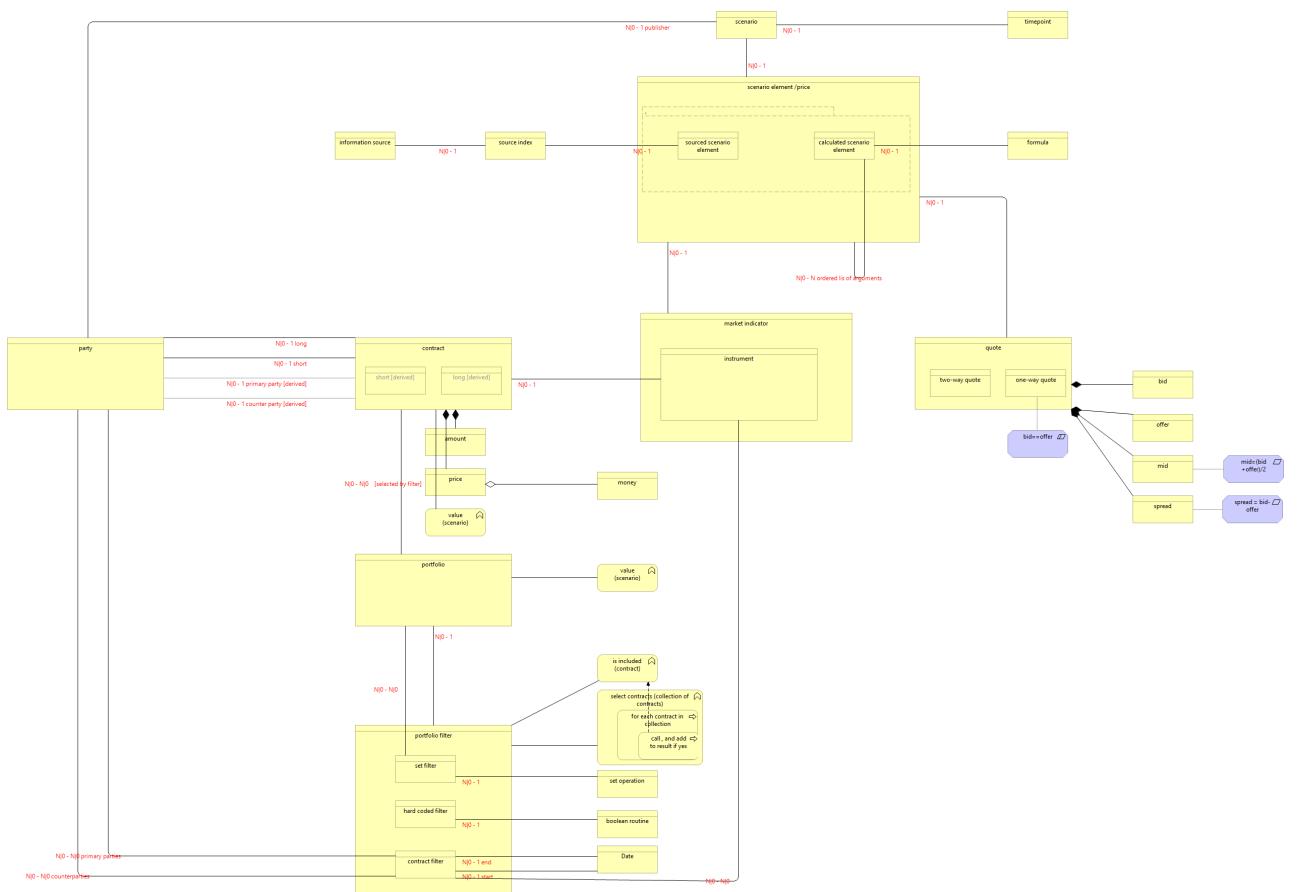
QUOTE



SCENARIO



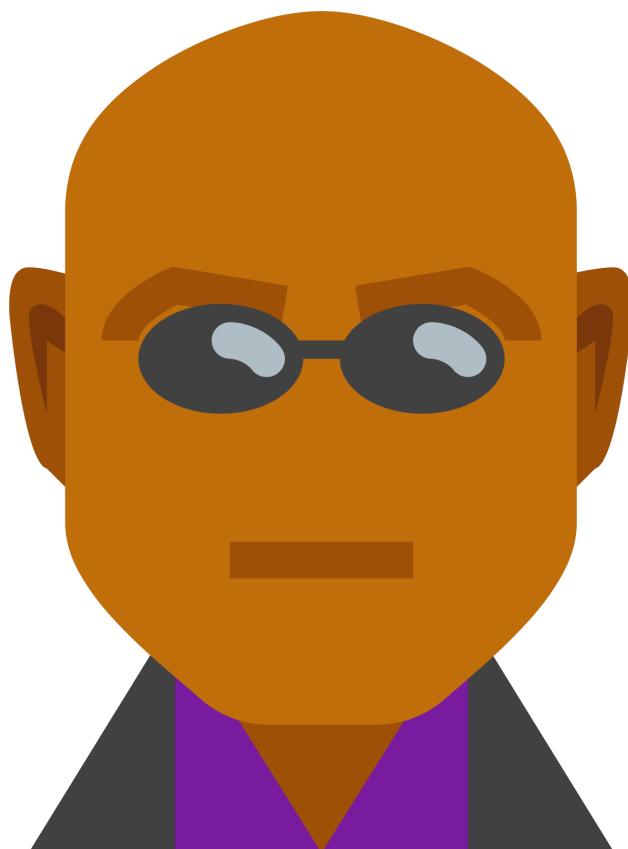
TRADING



DERIVATIVE CONTRACTS

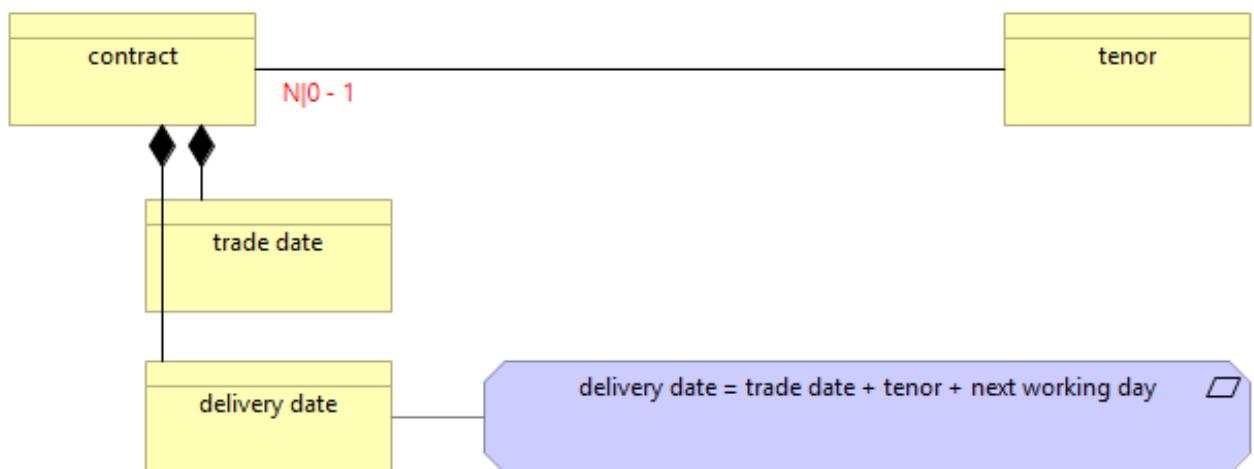
ANALYSIS PATTERNS

Martin Fowler



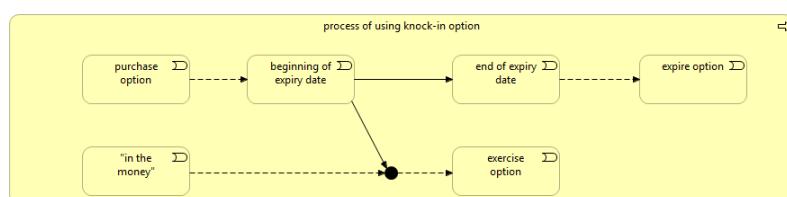
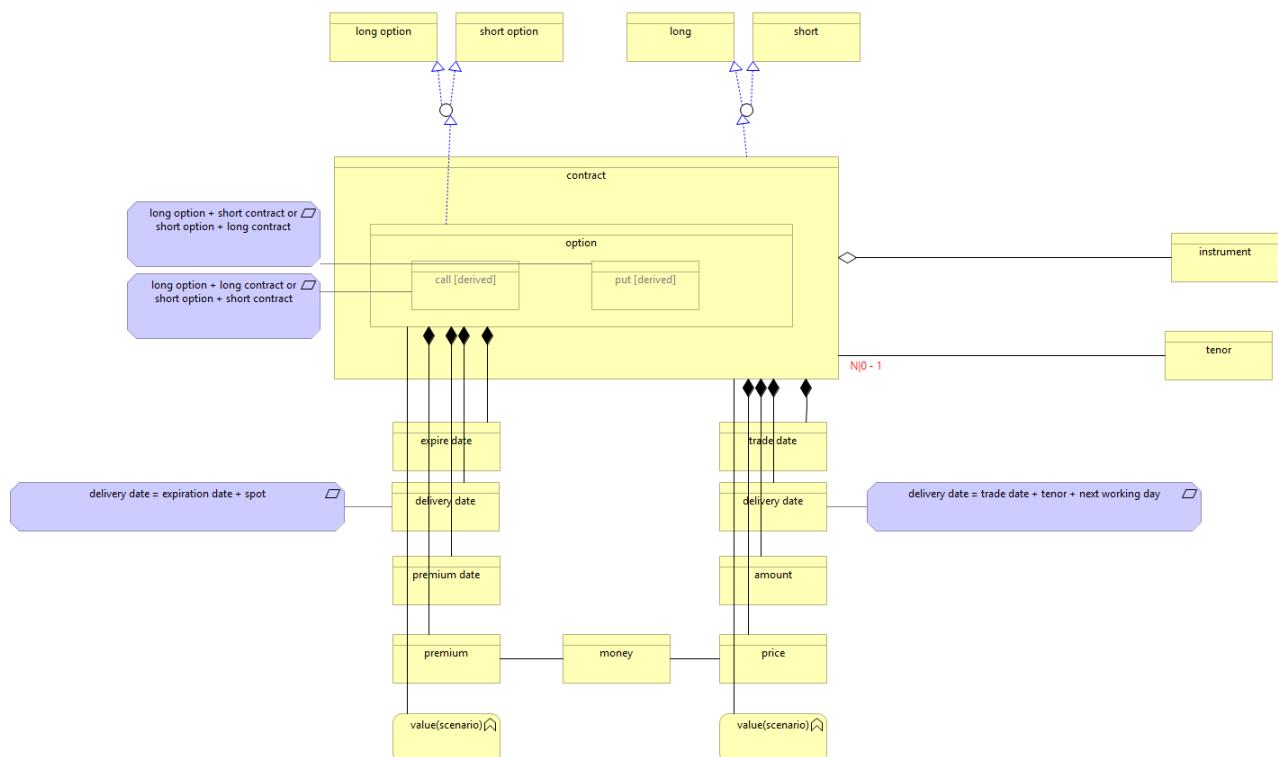
FORWARD CONTRACTS

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



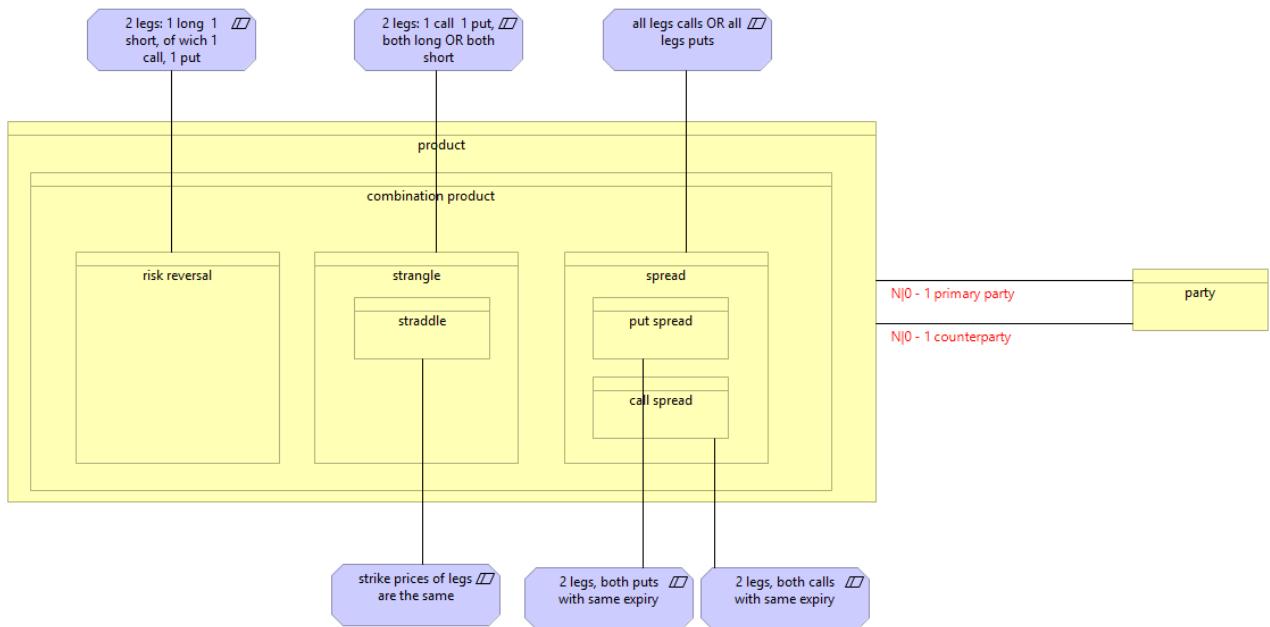
OPTIONS

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



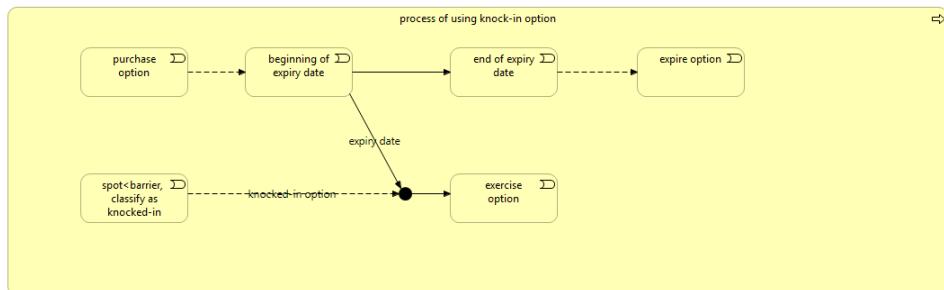
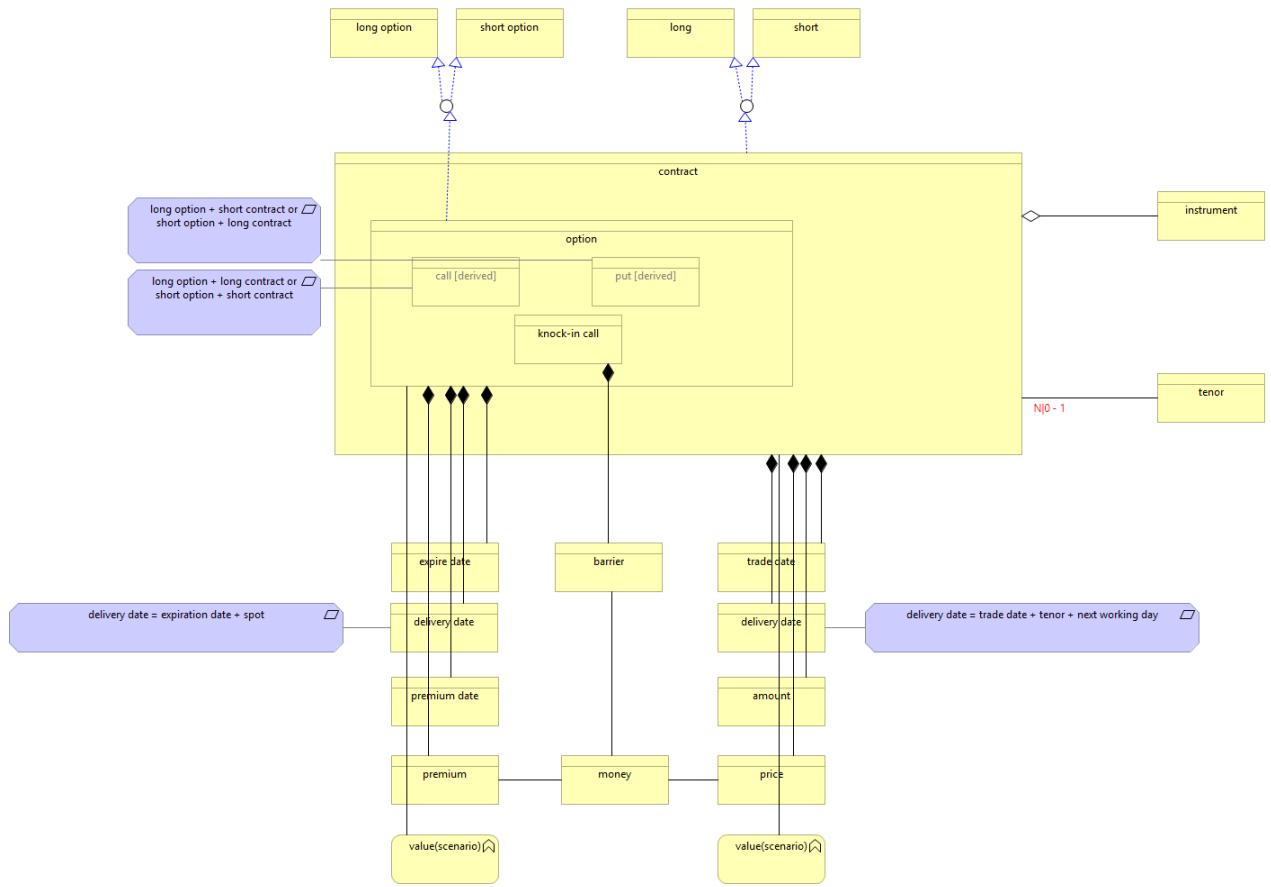
PRODUCT

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



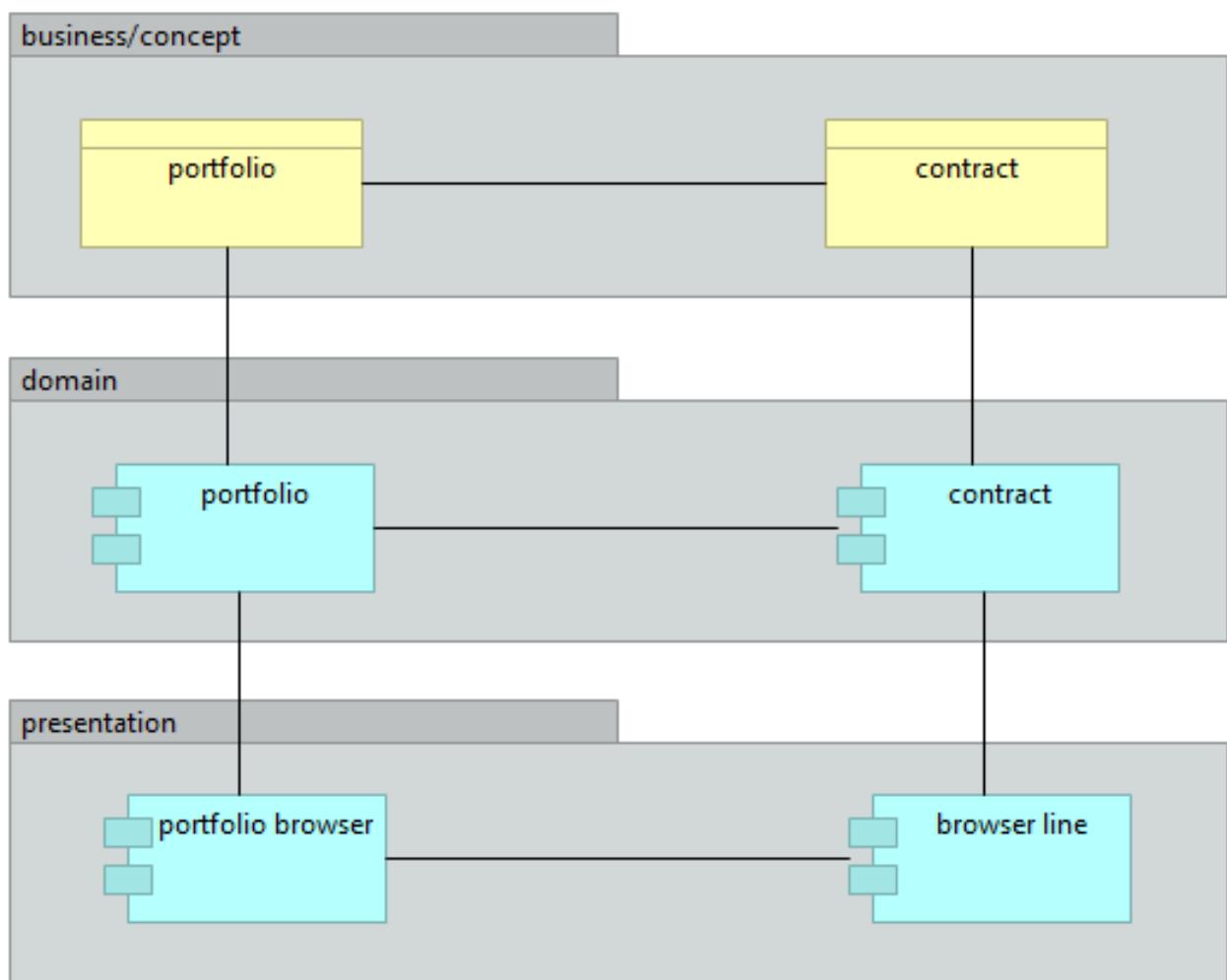
SUBTYPE STATE MACHINES

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

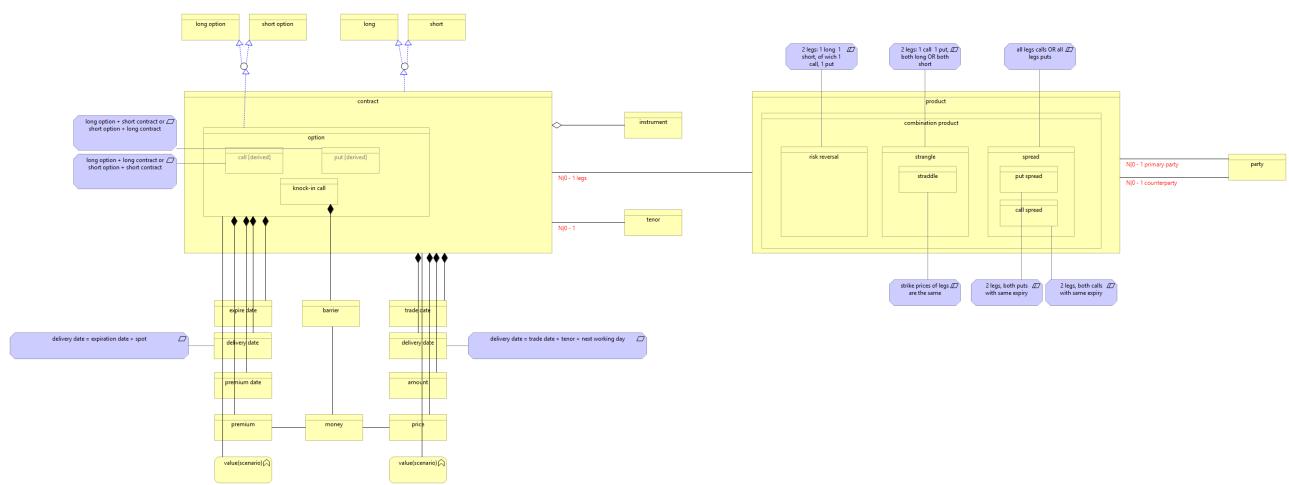


PARALLEL APPLICATION AND DOMAIN HIERARCHIES

example, a report containing a
list of contracts



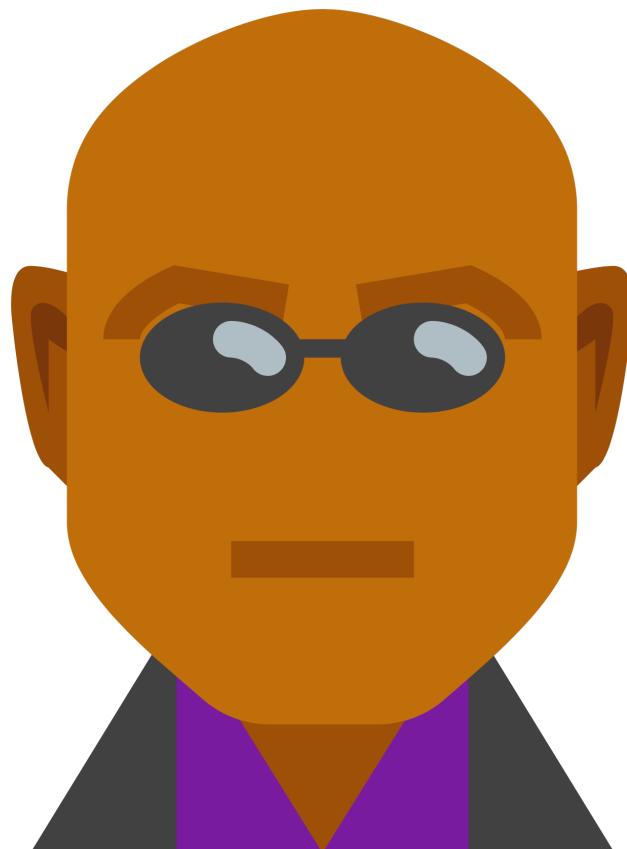
DERIVATIVE CONTRACTS



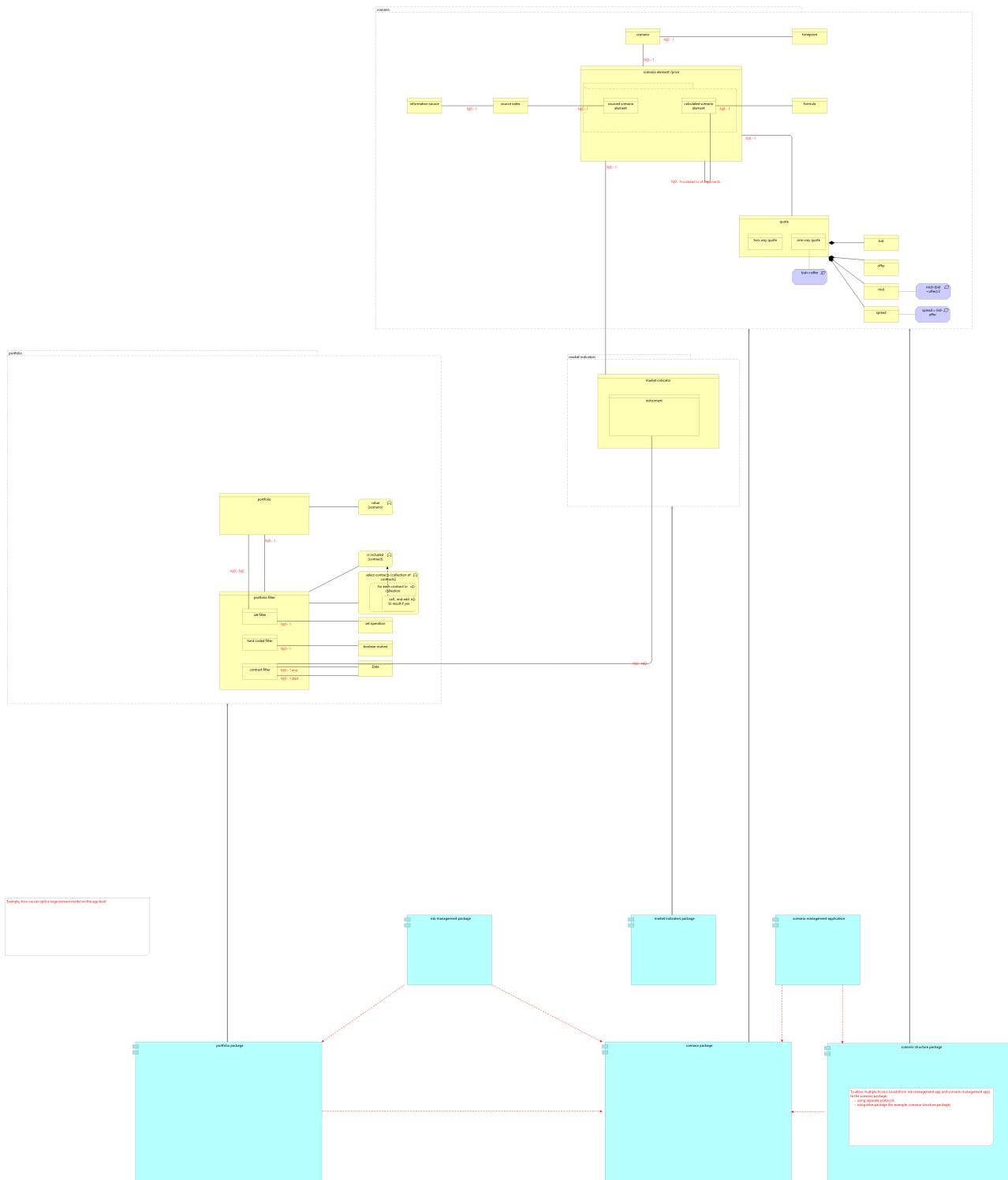
TRADING PACKAGES

ANALYSIS PATTERNS

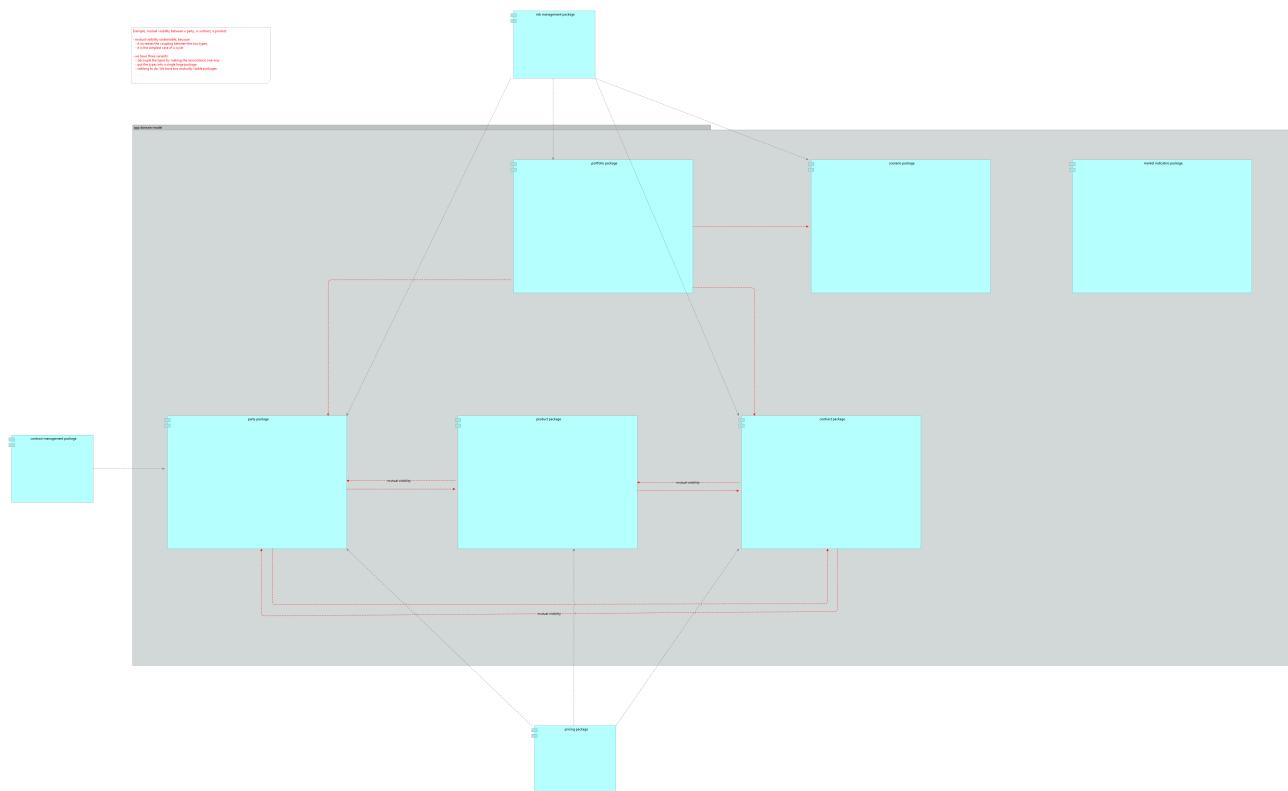
Martin Fowler



MULTIPLE ACCESS LEVELS TO A PACKAGE



MUTUAL VISIBILITY

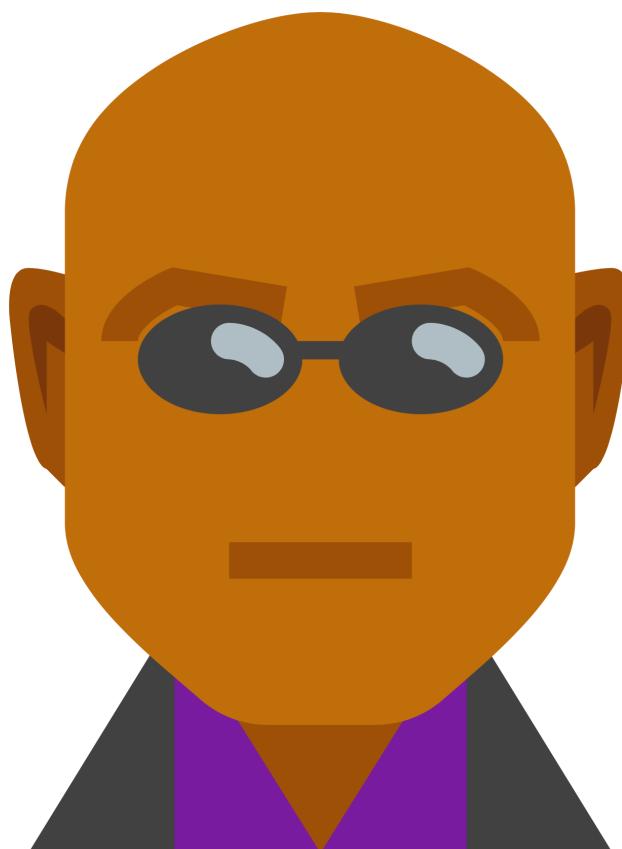


TRADING PACKAGES

LAYERED ARCHITECTURE

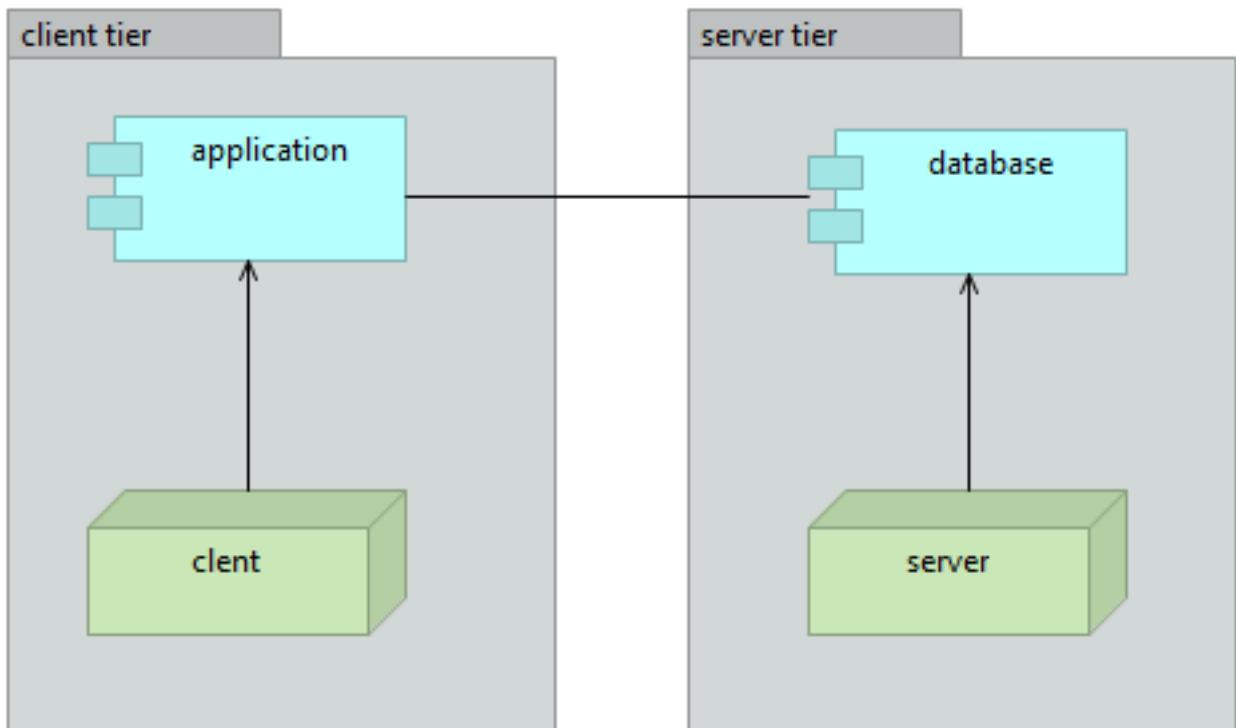
ANALYSIS PATTERNS

Martin Fowler

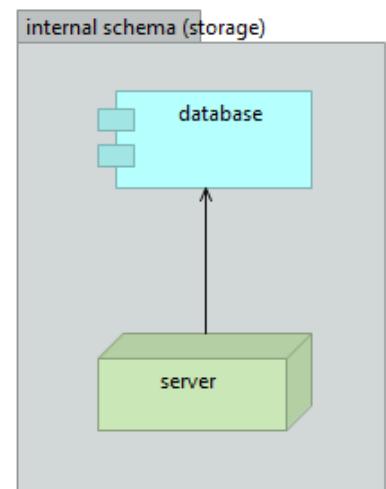
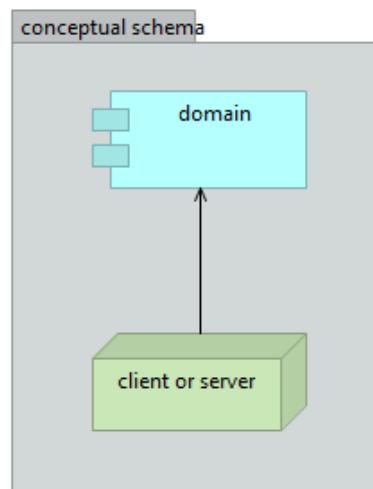
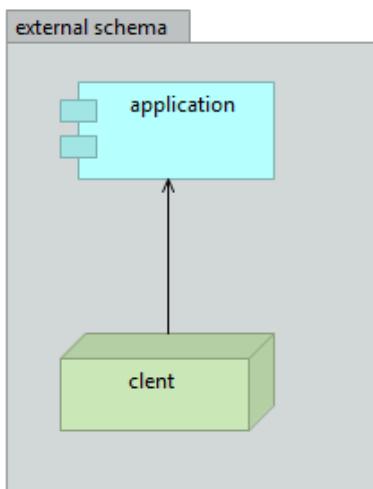


TWO-TIER ARCHITECTURE

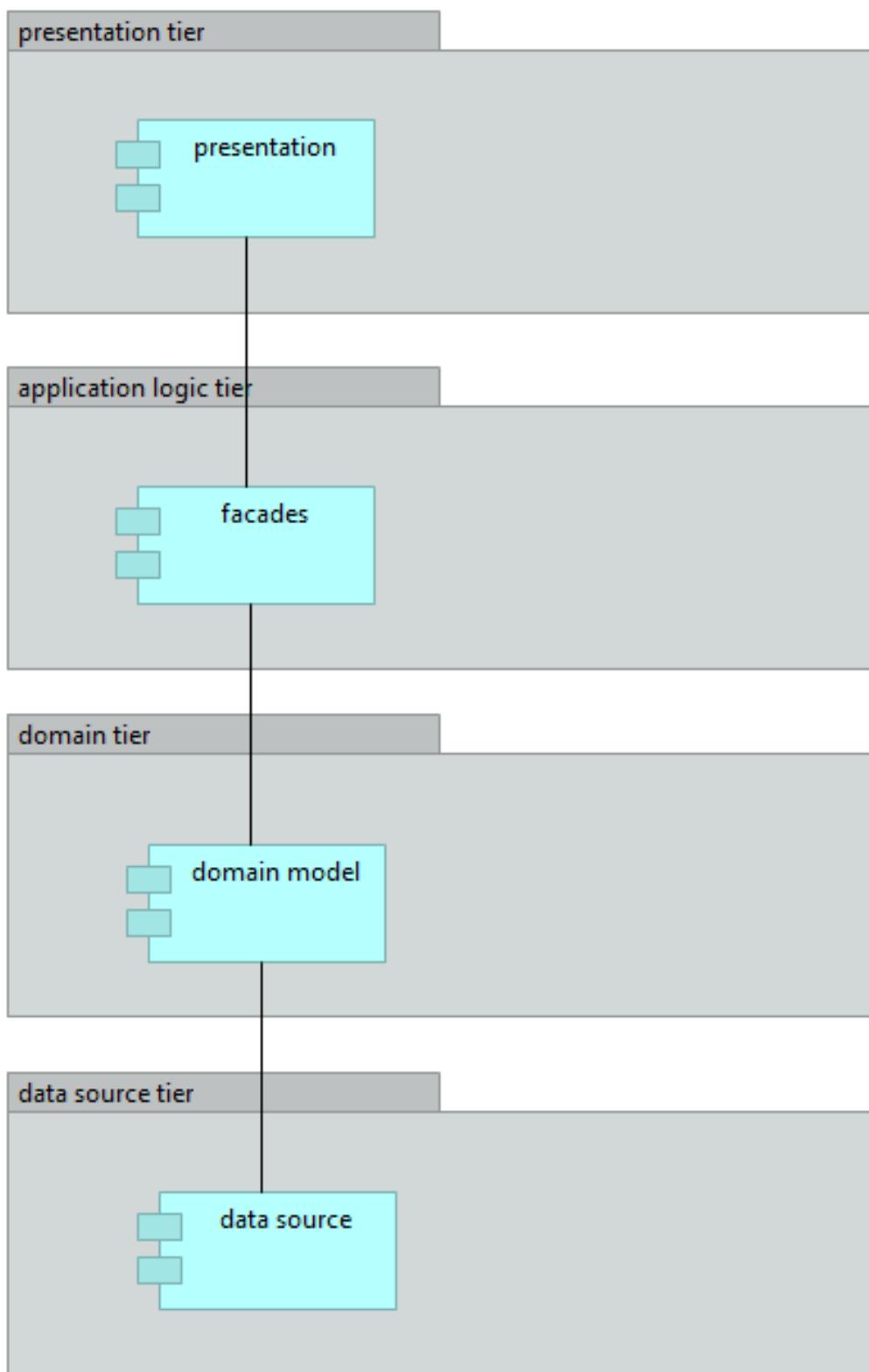
two-tier architecture



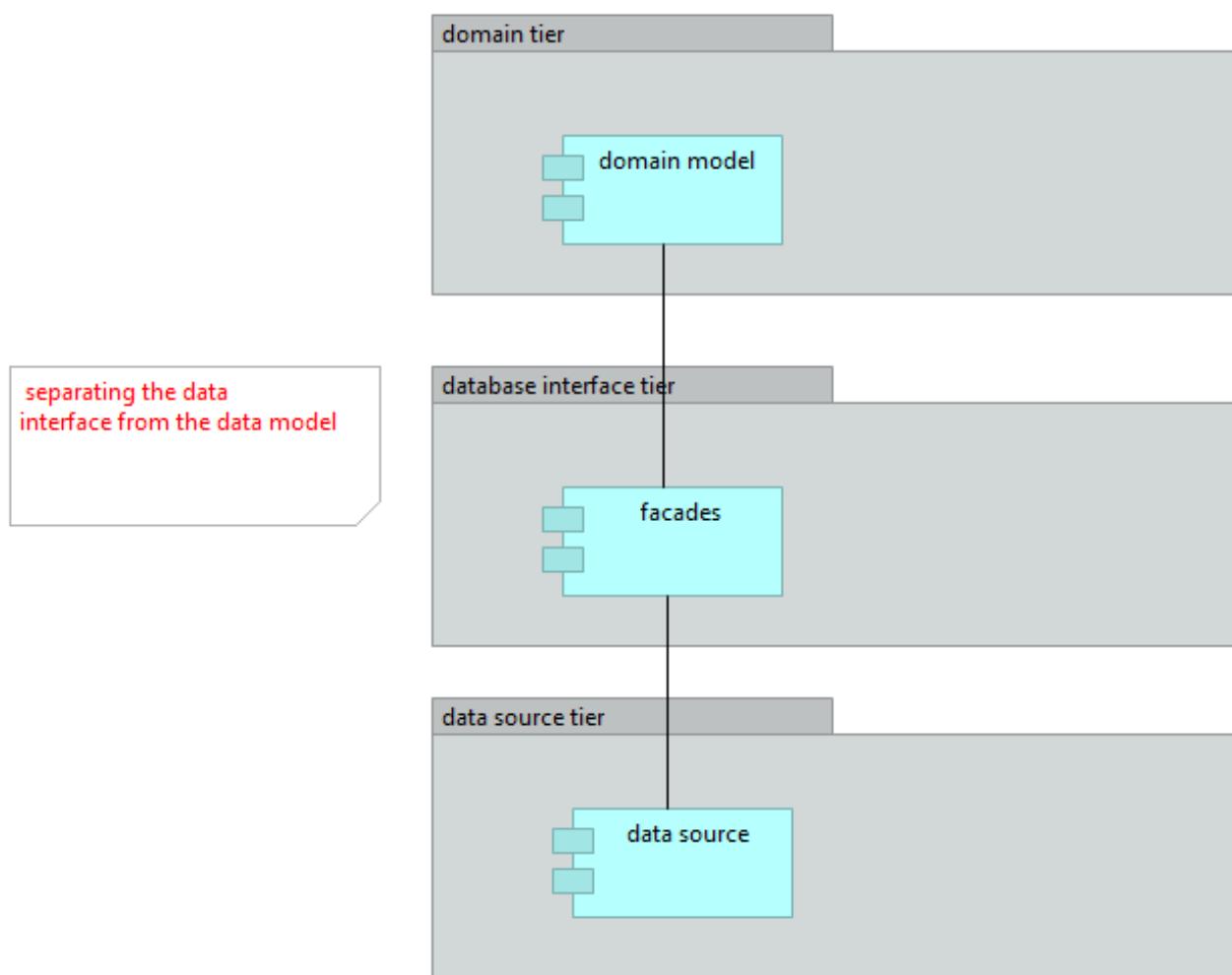
THREE-TIER ARCHITECTURE



PRESENTATION AND APPLICATION LOGIC



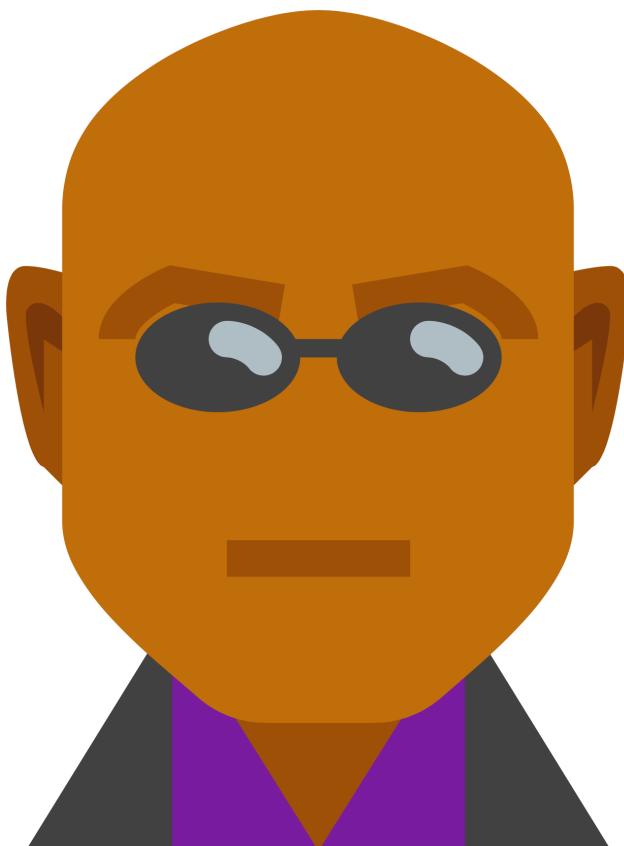
DATABASE INTERACTION



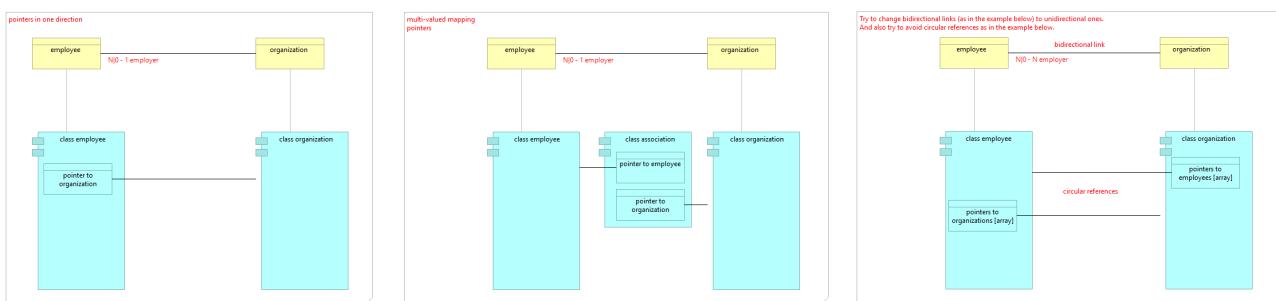
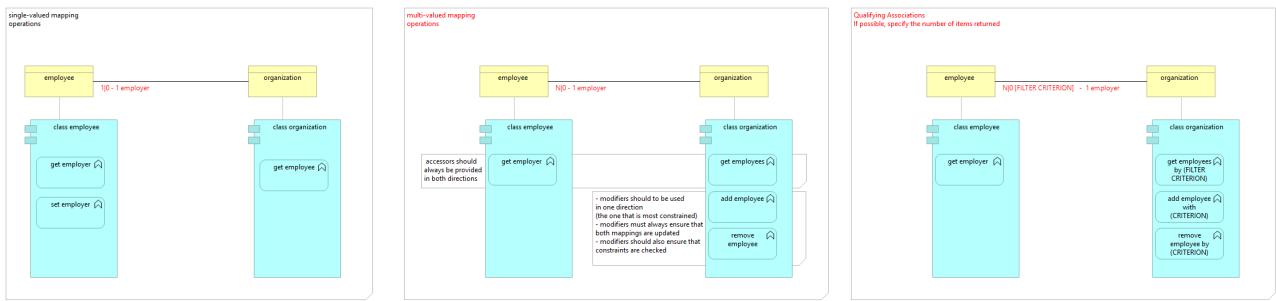
TYPE MODEL DESIGN

ANALYSIS PATTERNS

Martin Fowler

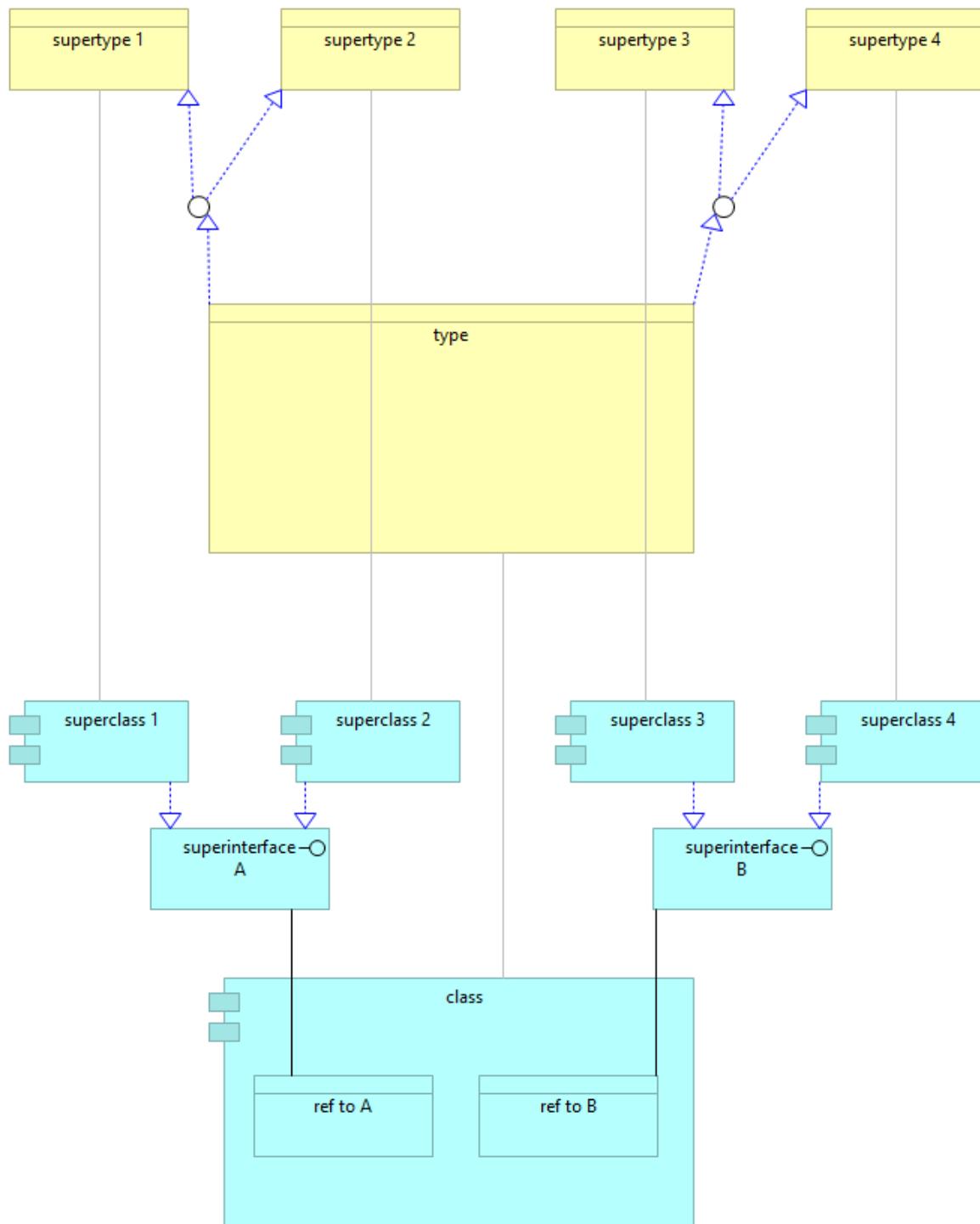


IMPLEMENTING ASSOCIATIONS

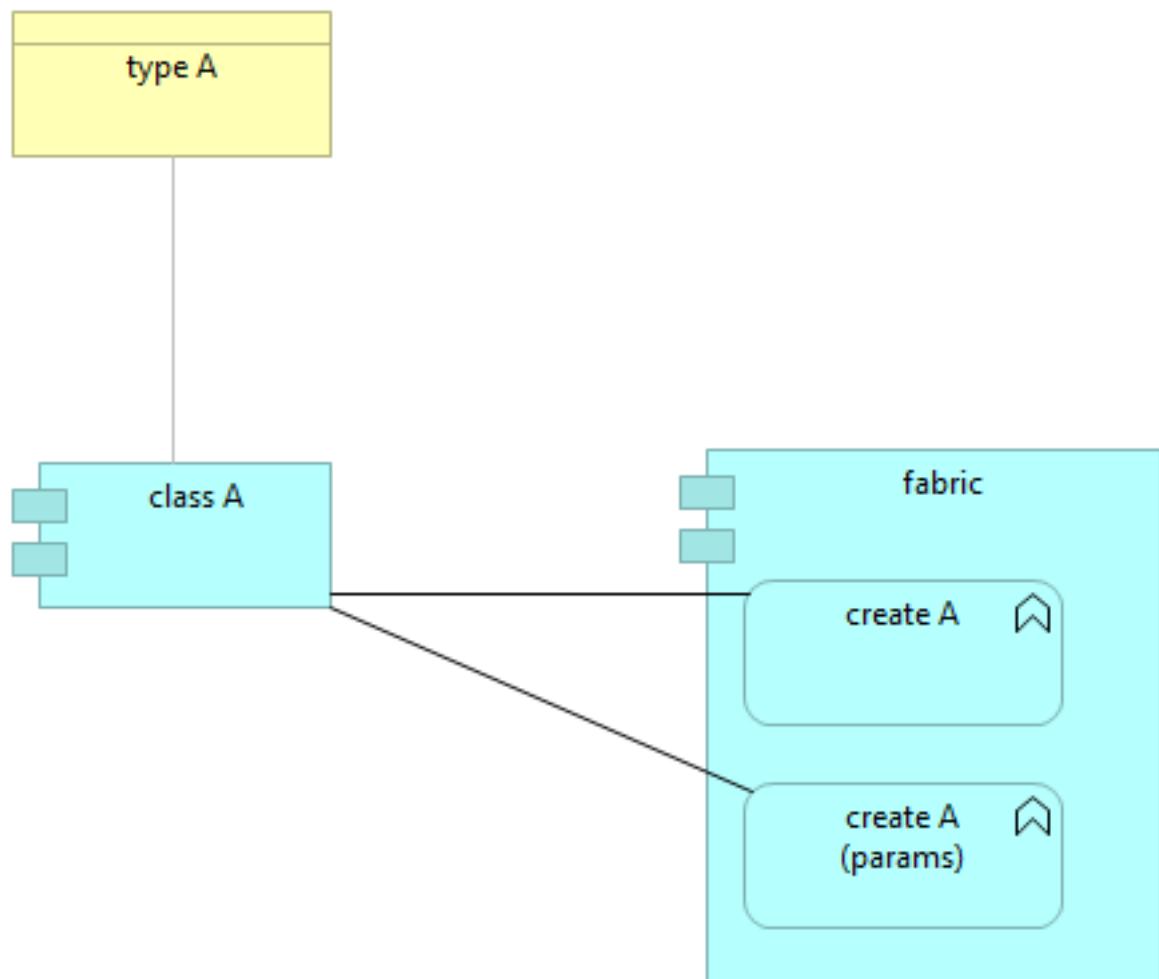


IMPLEMENTING GENERALIZATION

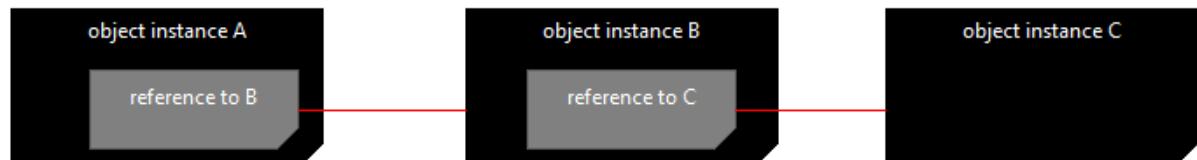
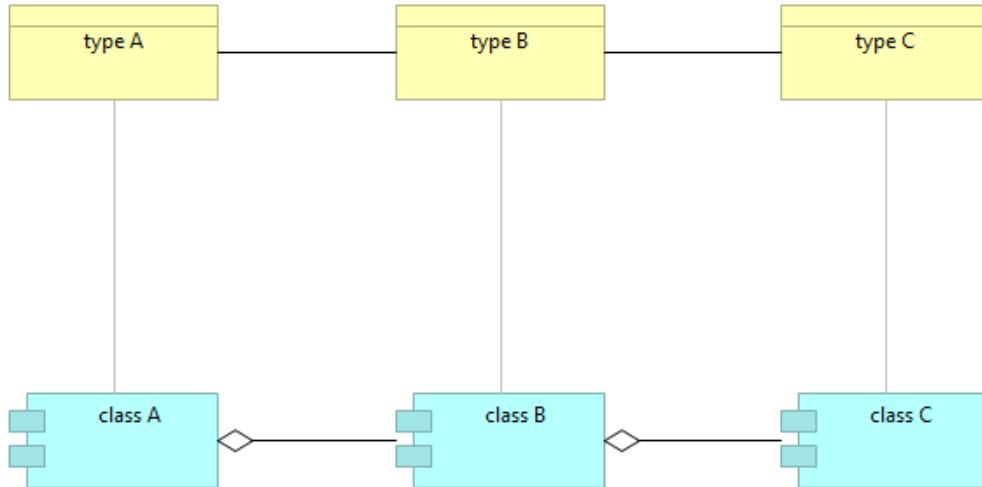
- example of implementation of multiple inheritance
- for example, the composition is applicable instead of inheritance



OBJECT CREATION



OBJECT DESTRUCTION

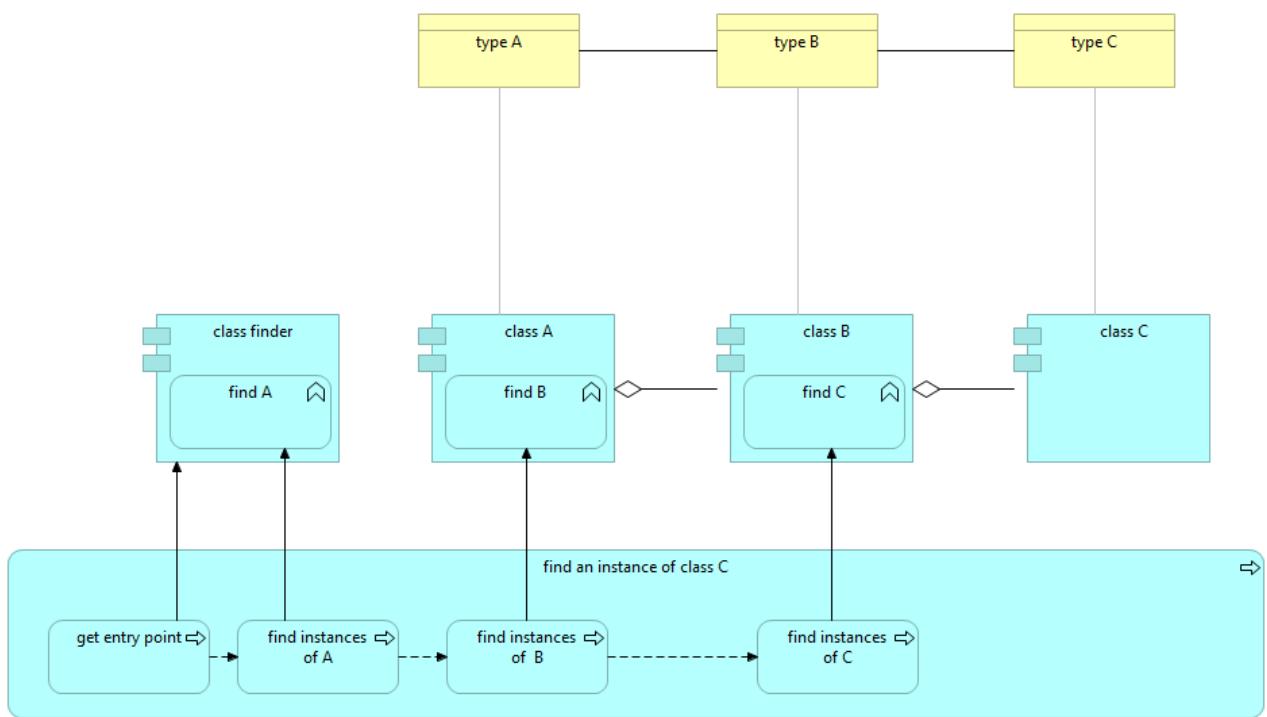


for example, delete object B

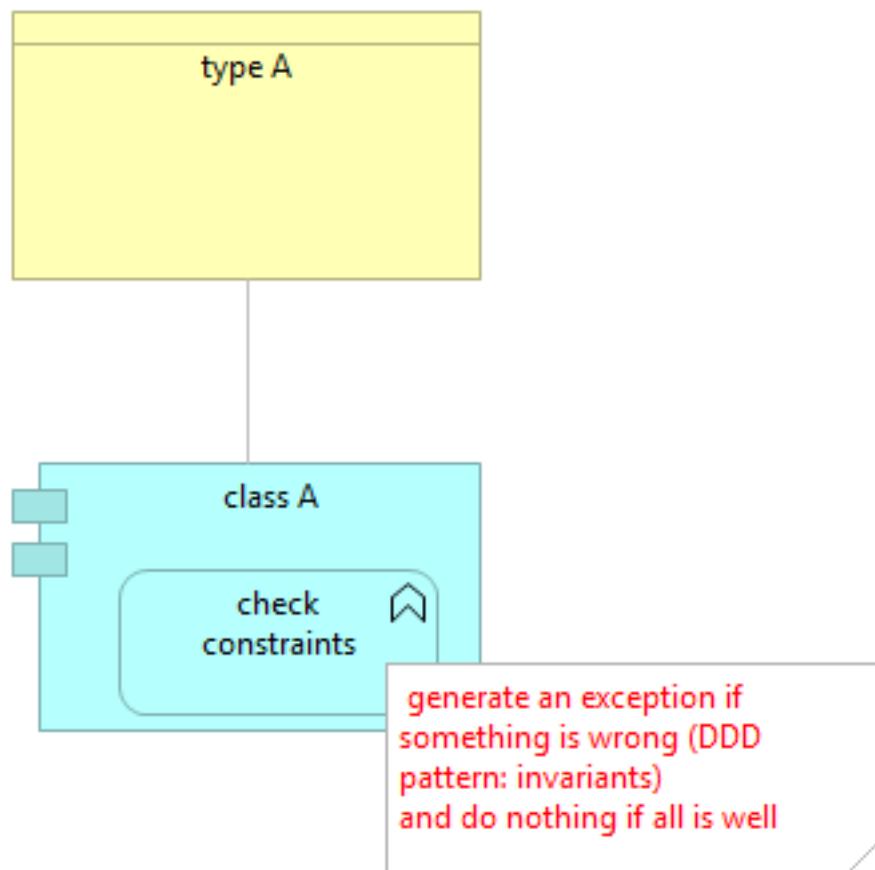
- all constraints are met
- no dangling references

object instance A

ENTRY POINT.



IMPLEMENTING CONSTRAINTS

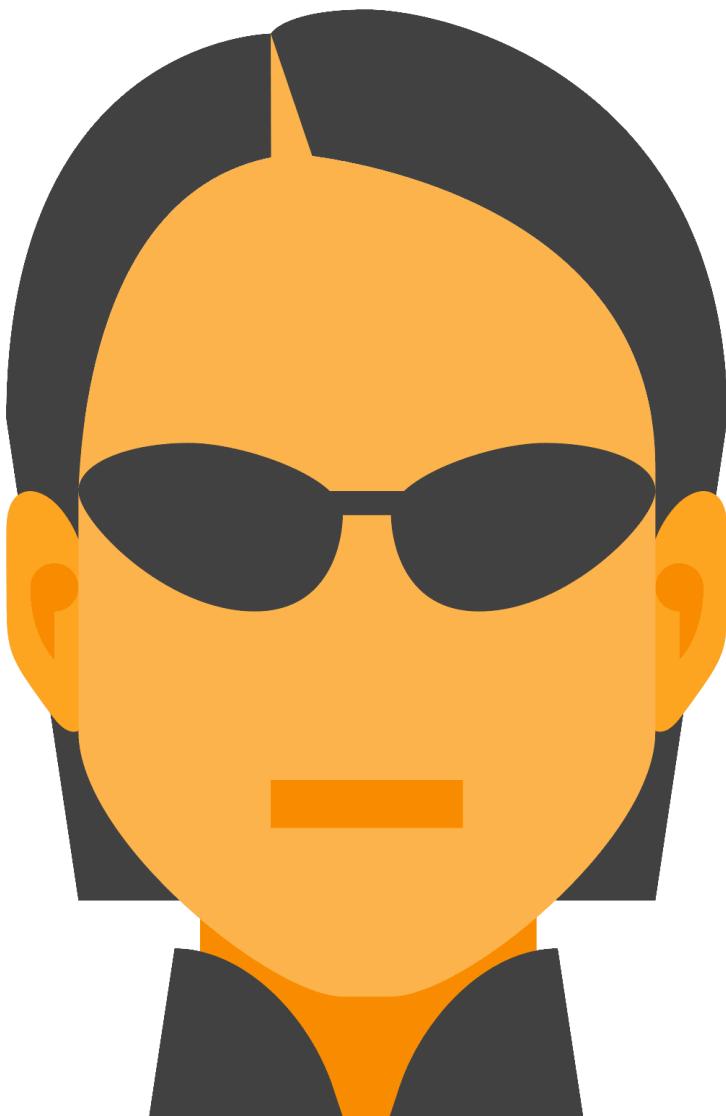


04

Domain Driven Design

Tackling Complexity in the Heart

ERIC EVANS



MODEL AND STRUCTURAL ELEMENTS

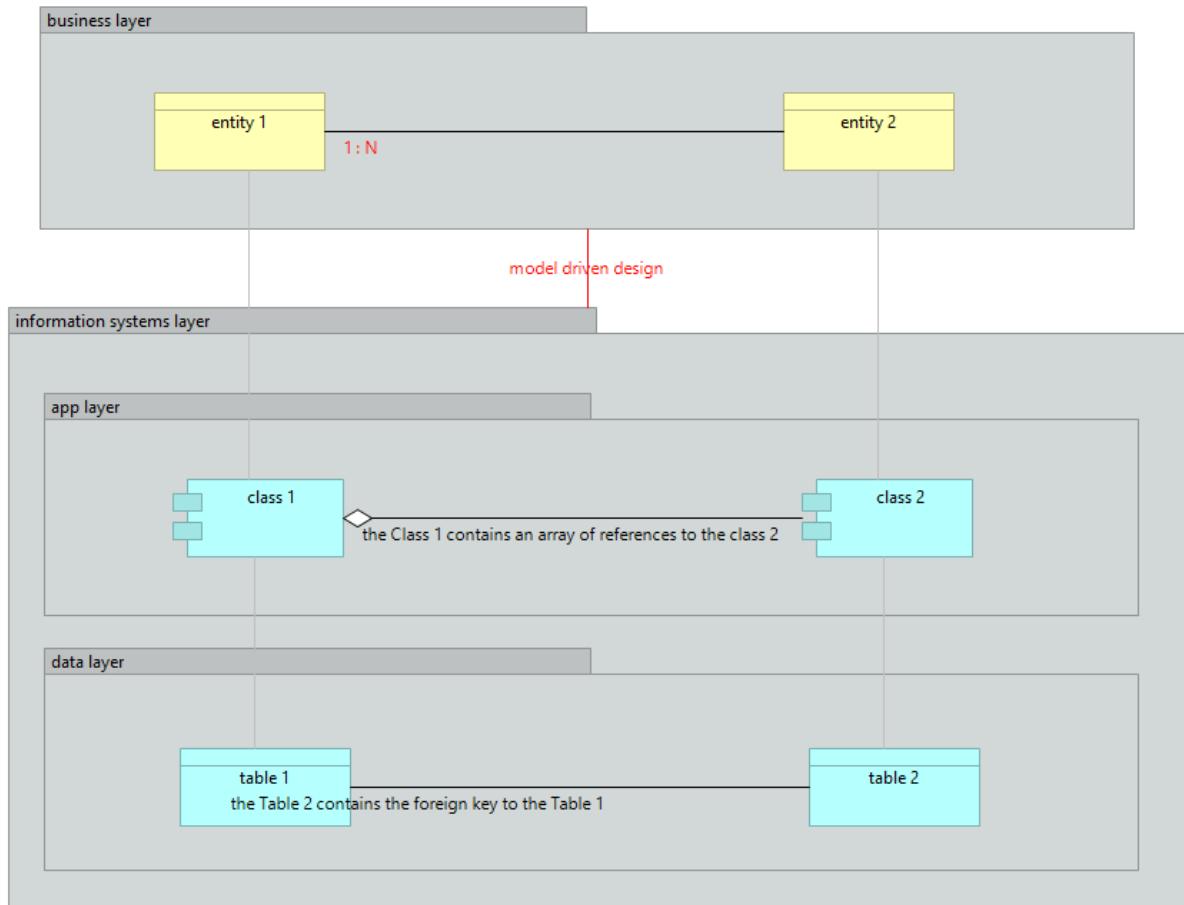
DOMAIN DRIVEN DESIGN

Eric Evans



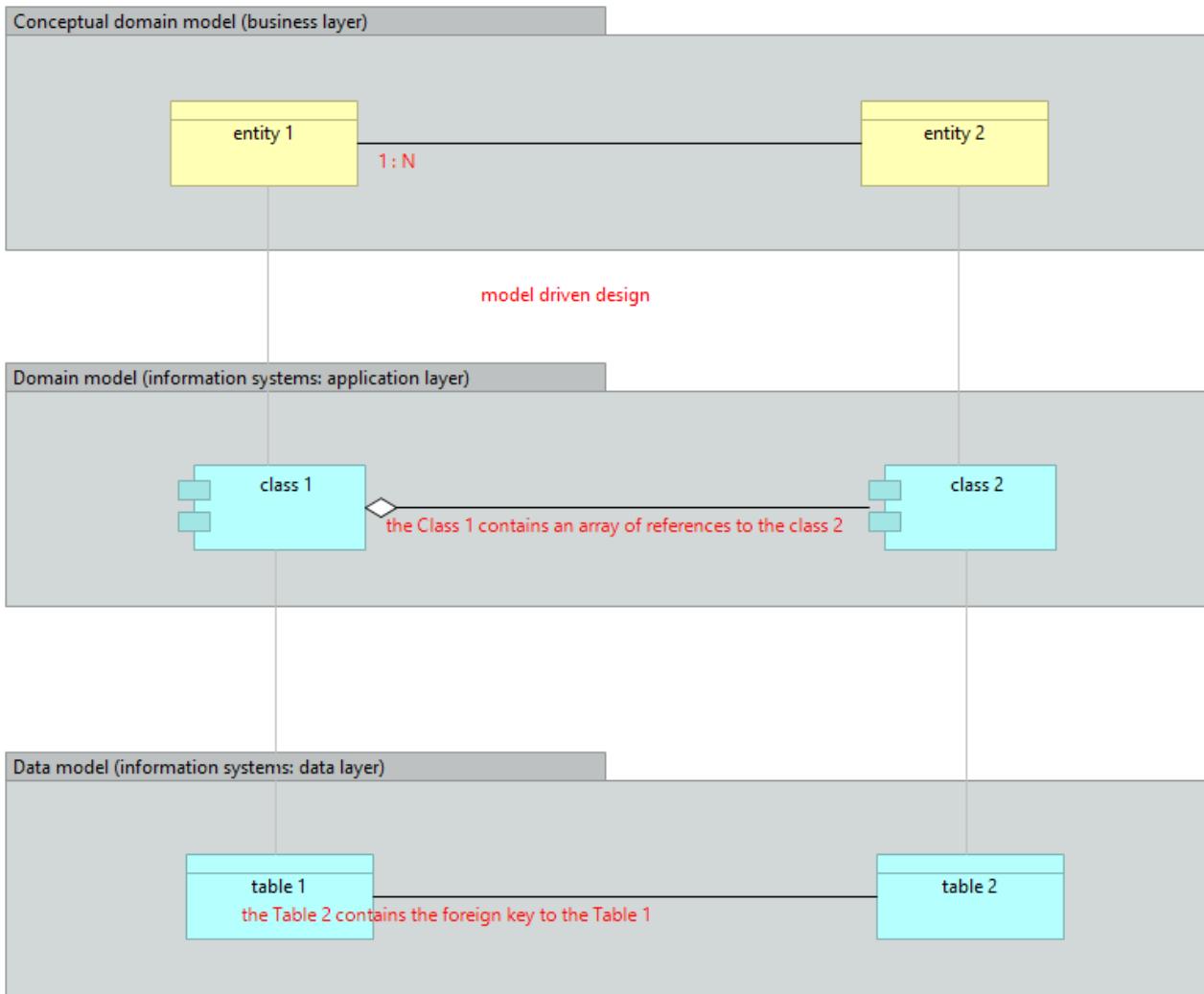
MODEL-DRIVEN DESIGN

layers - TOGAF's architecture domains



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





Single responsibility principle from the SOLID model

Conceptual domain model (business layer)

Business Actor 1 ♀

Business Actor 2 ♂

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

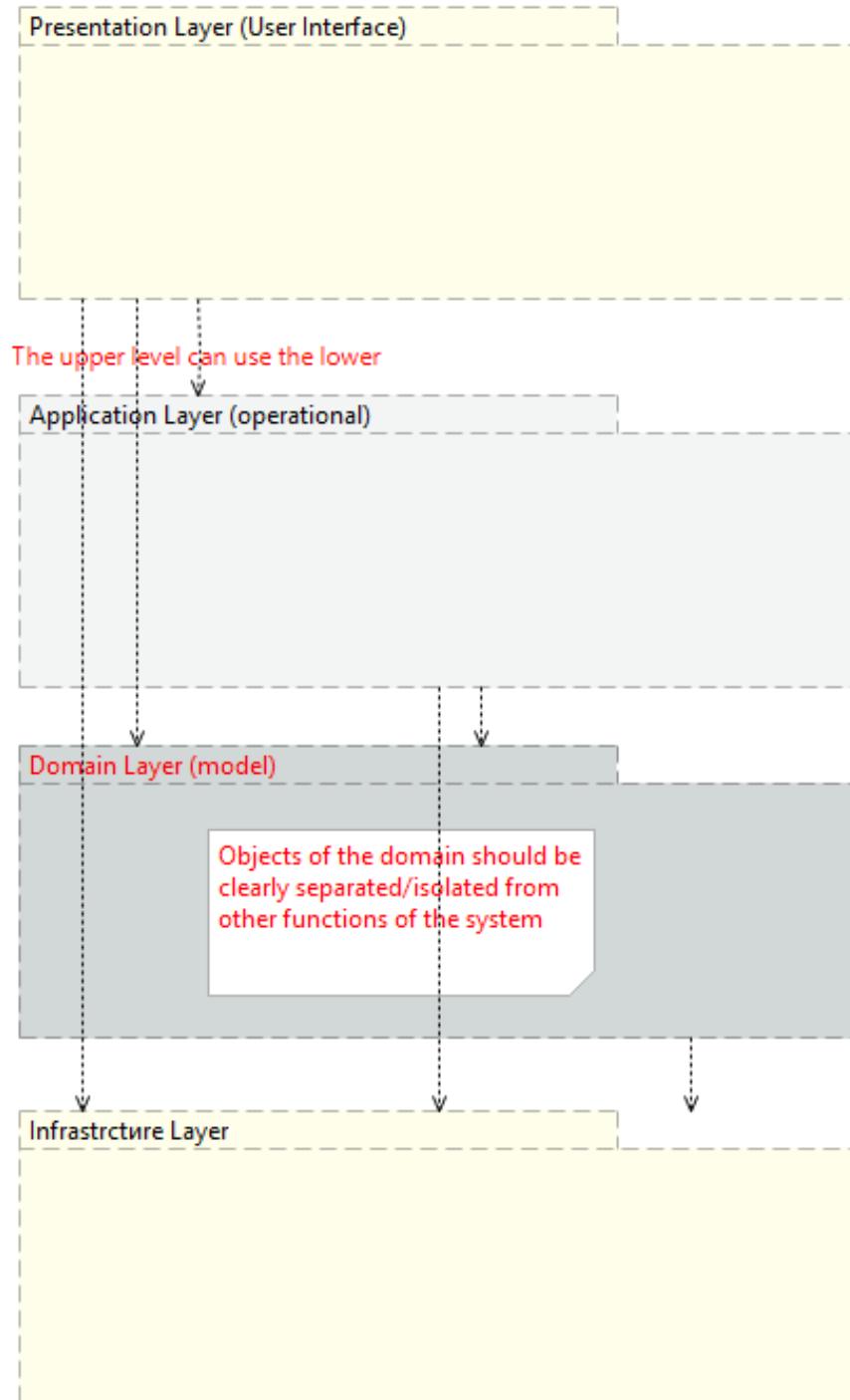
Domain model (information systems: application layer)

class 1

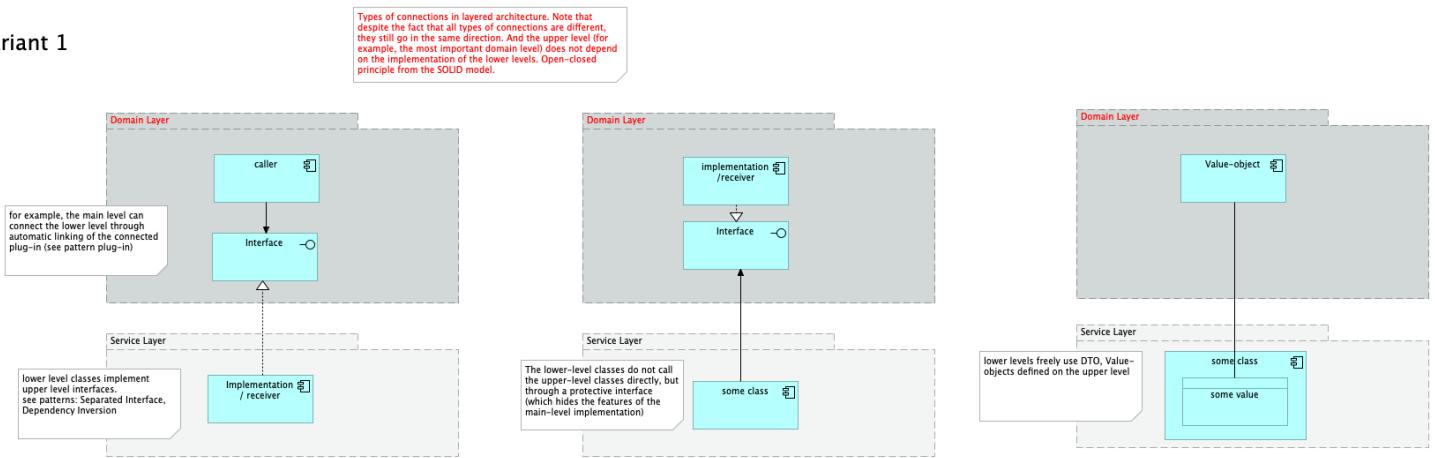
class 2

LAYERED ARCHITECTURE (ASYMMETRIC)

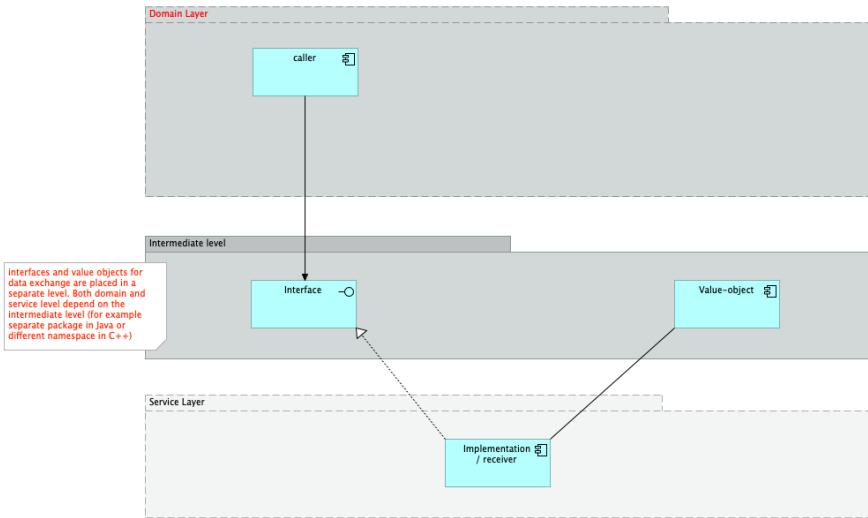
layered architecture
Only domain layer is required



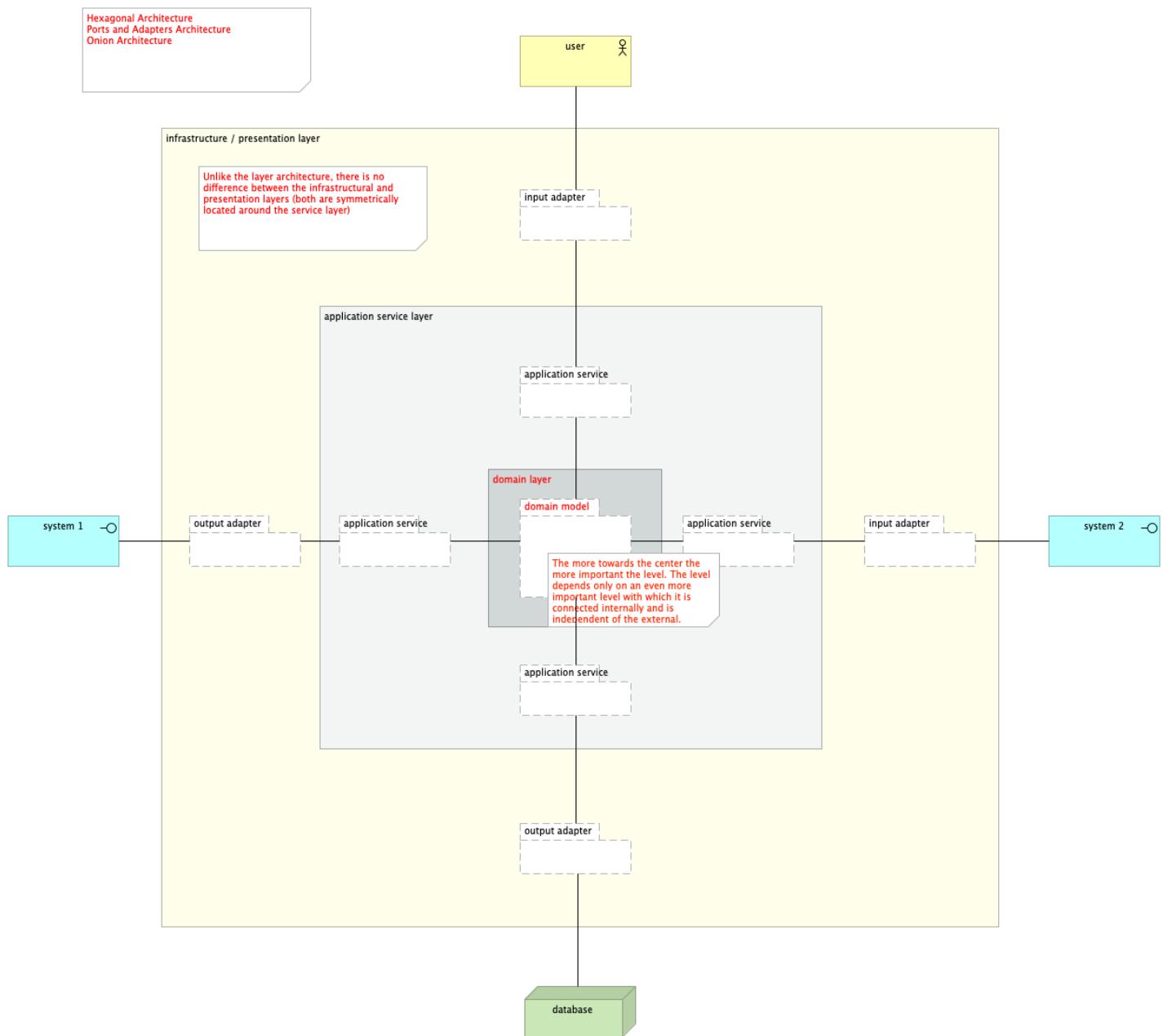
variant 1



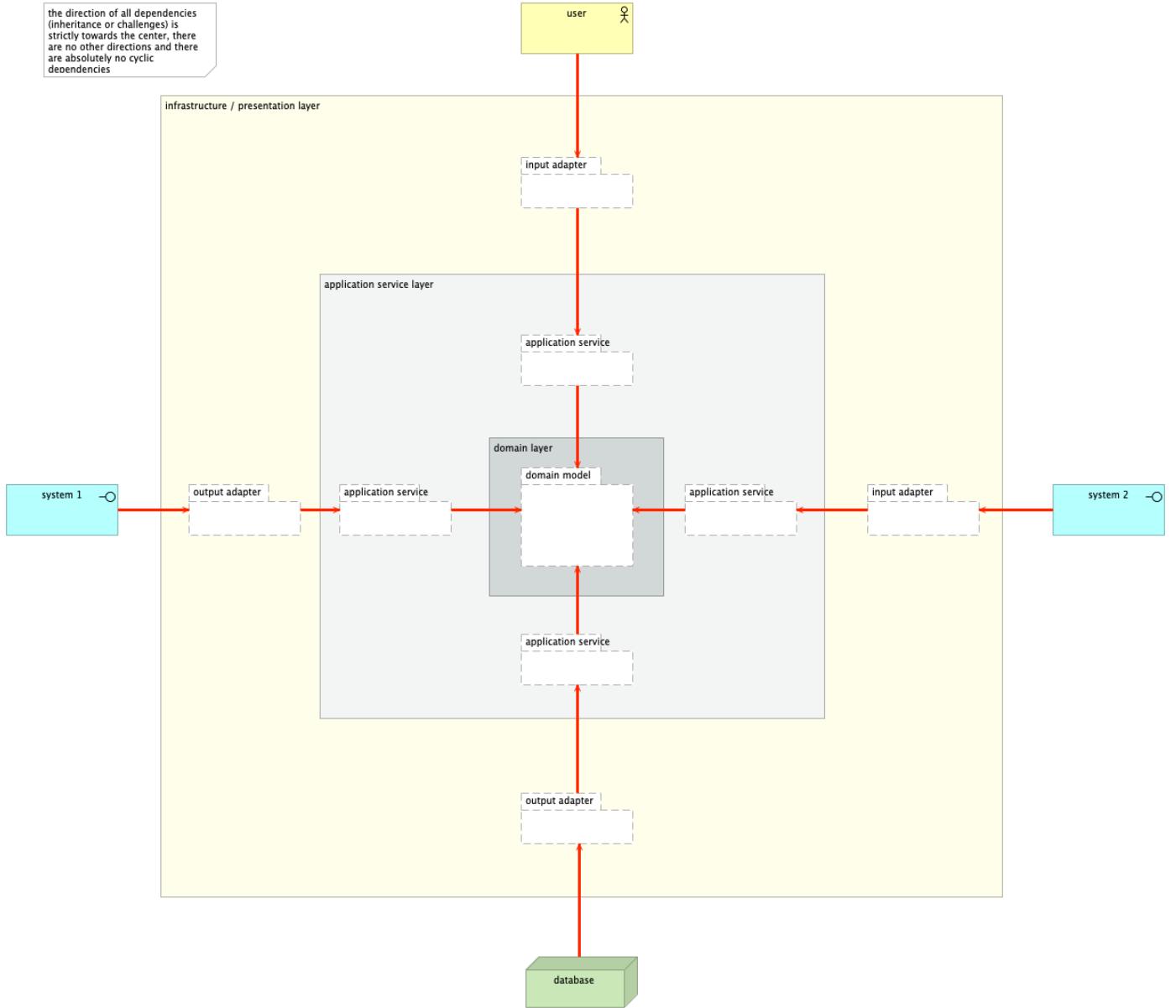
variant 2

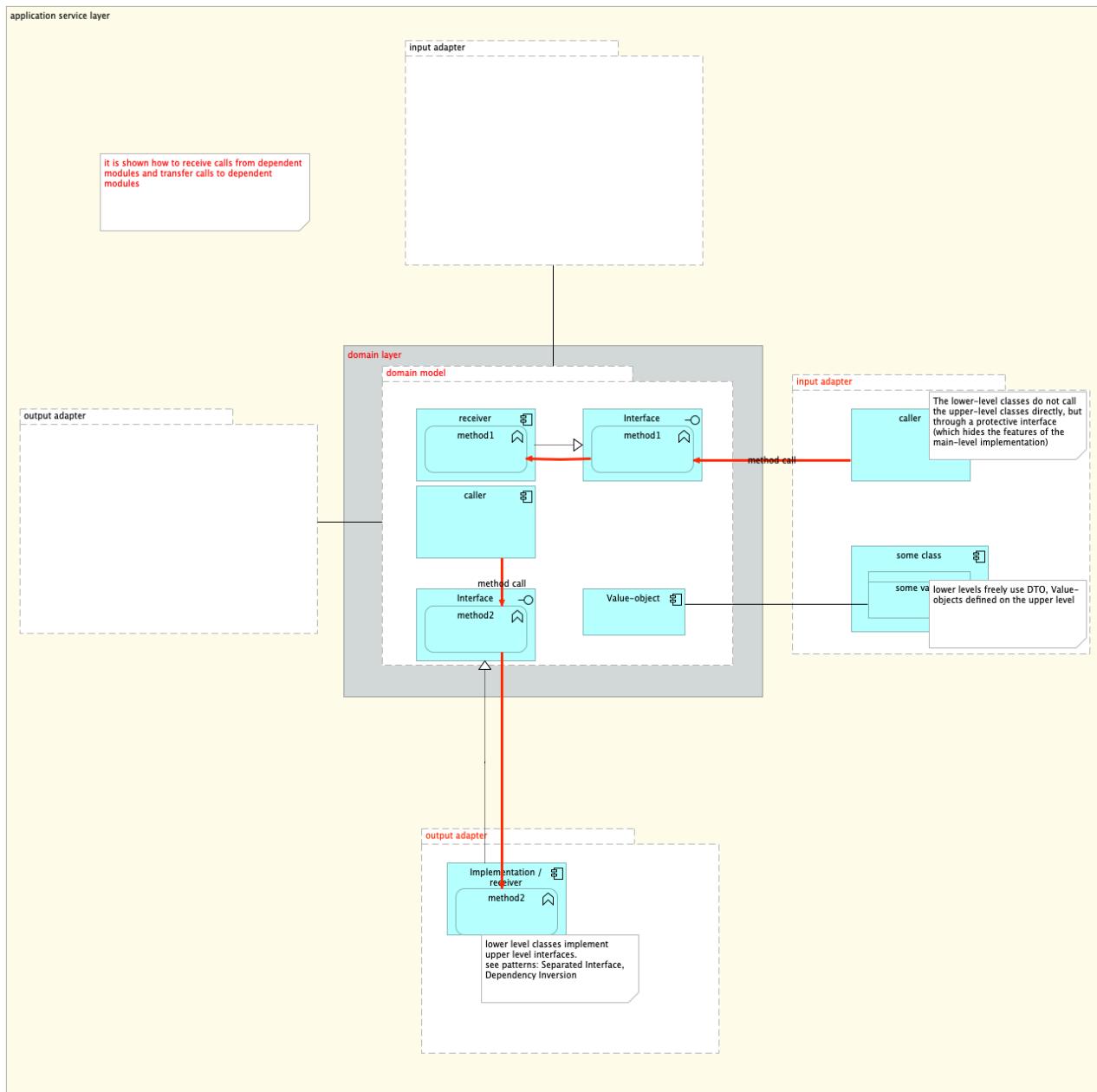


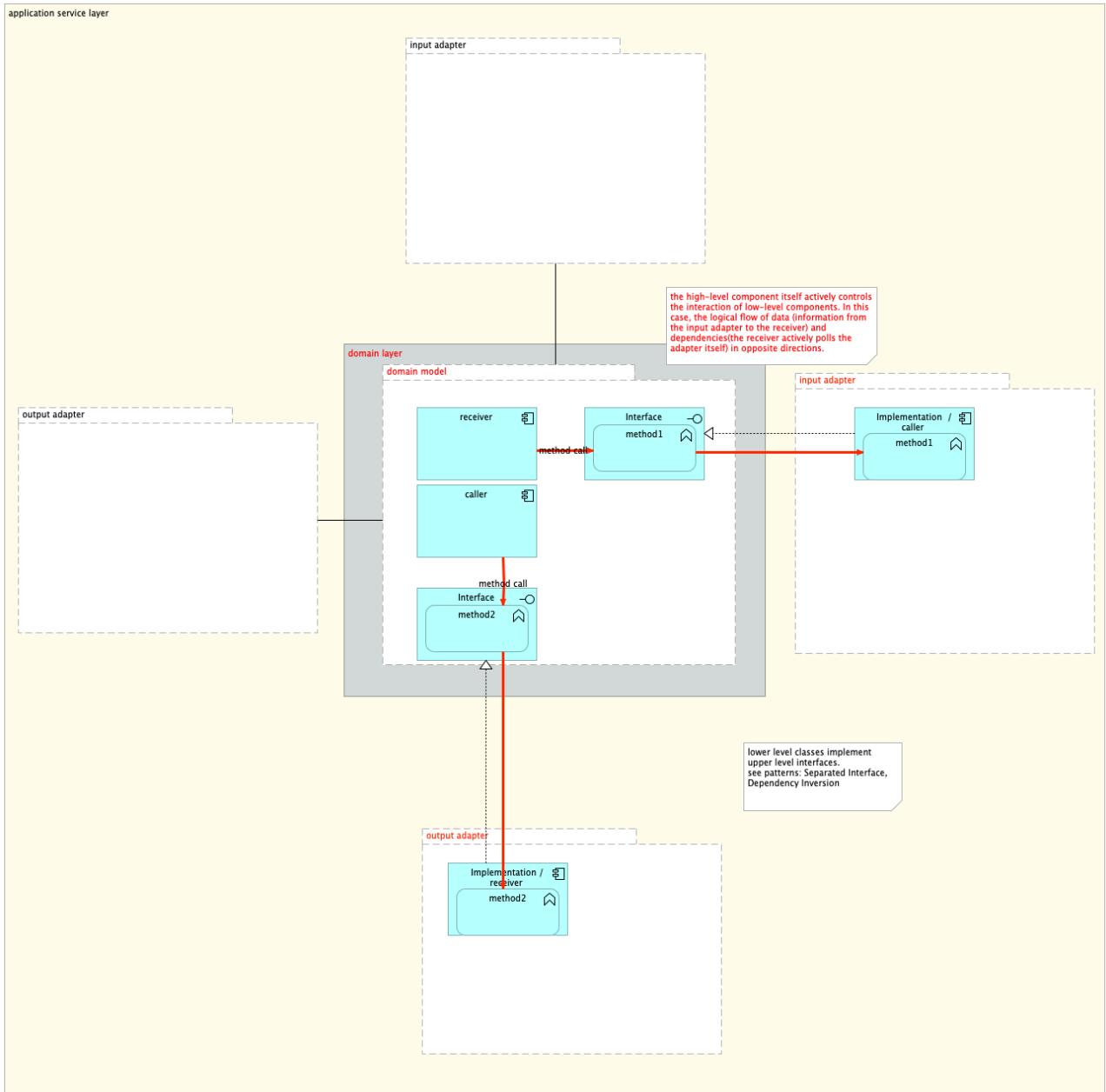
HEXAGONAL ARCHITECTURE (SYMMETRIC)



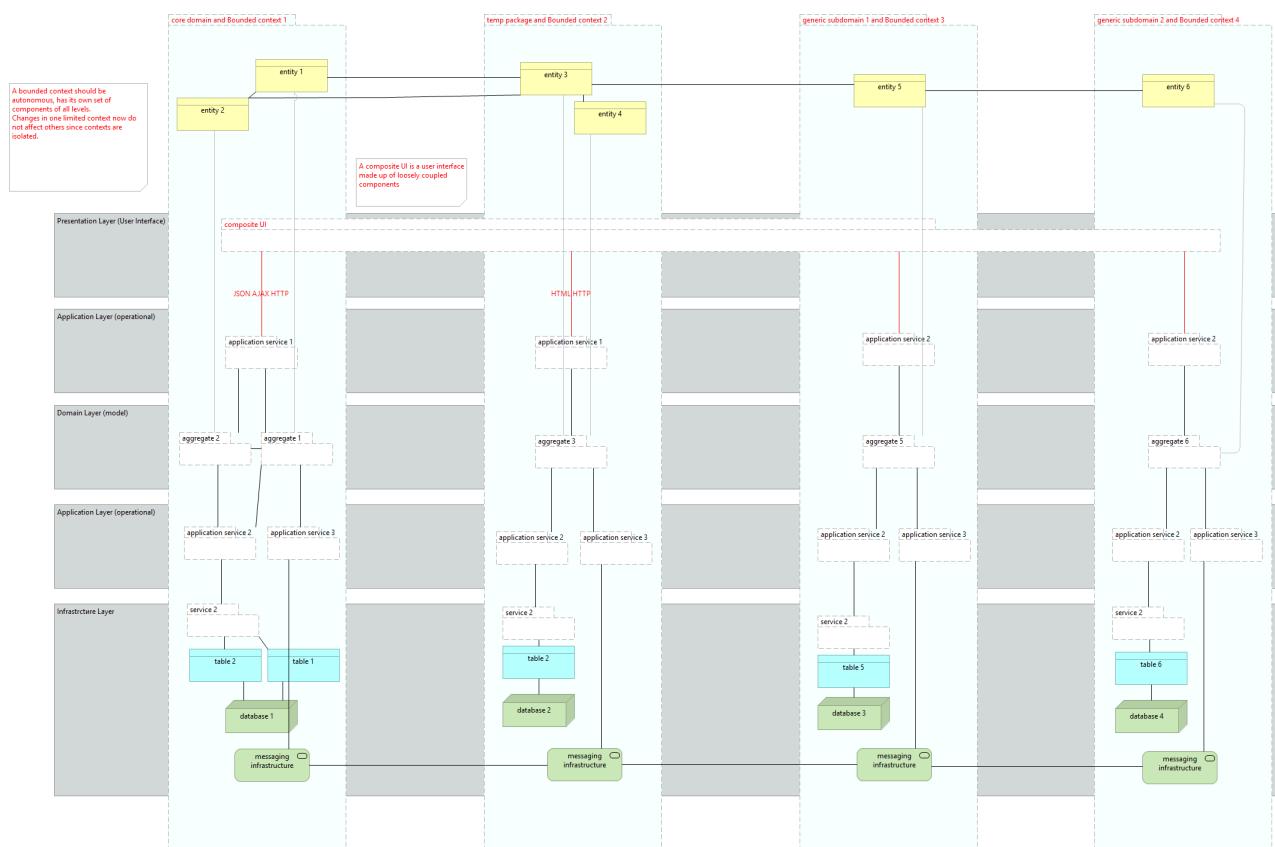
the direction of all dependencies (inheritance or challenges) is strictly towards the center, there are no other directions and there are absolutely no cyclic dependencies



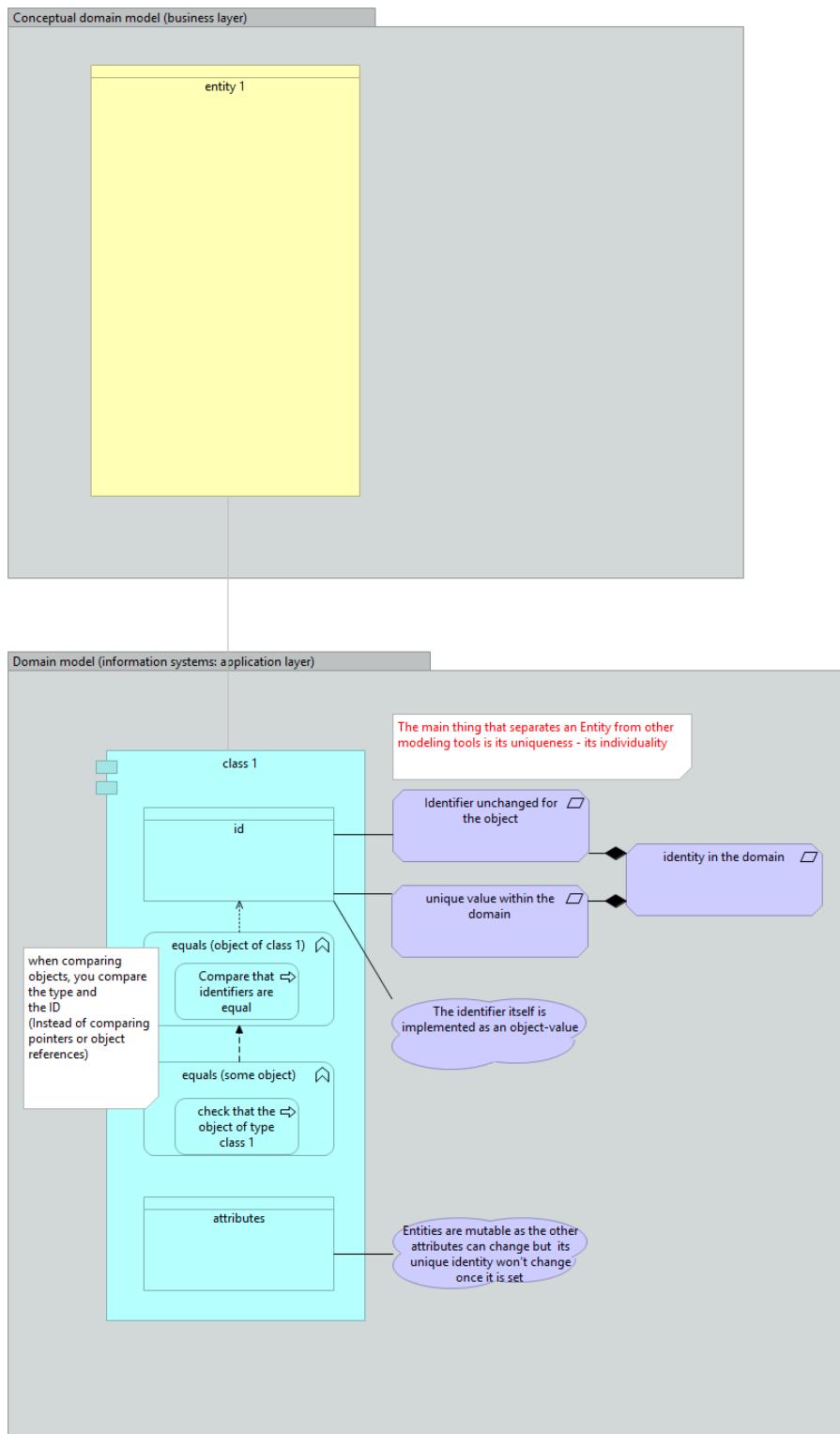


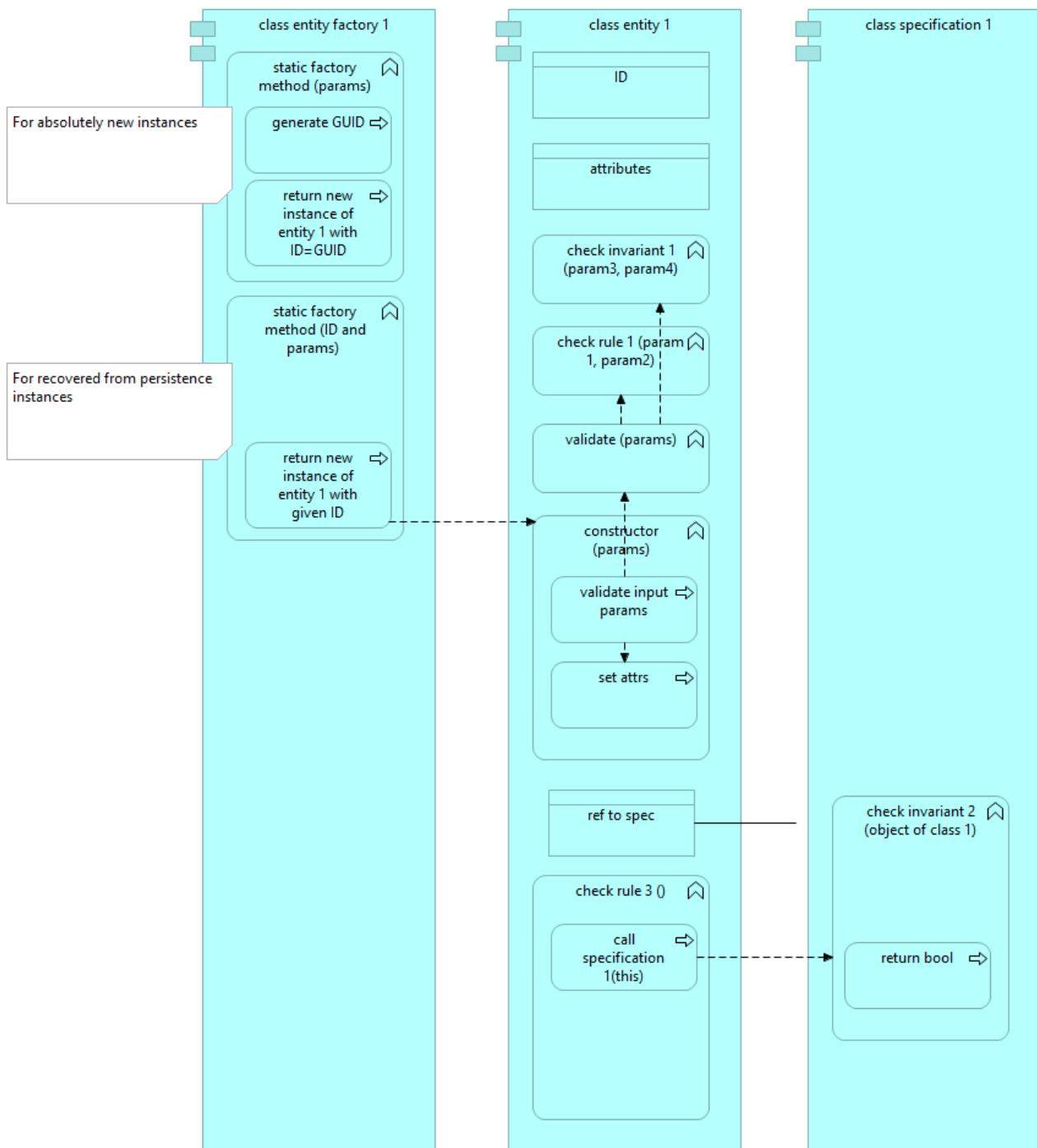


COMPOSITE UI

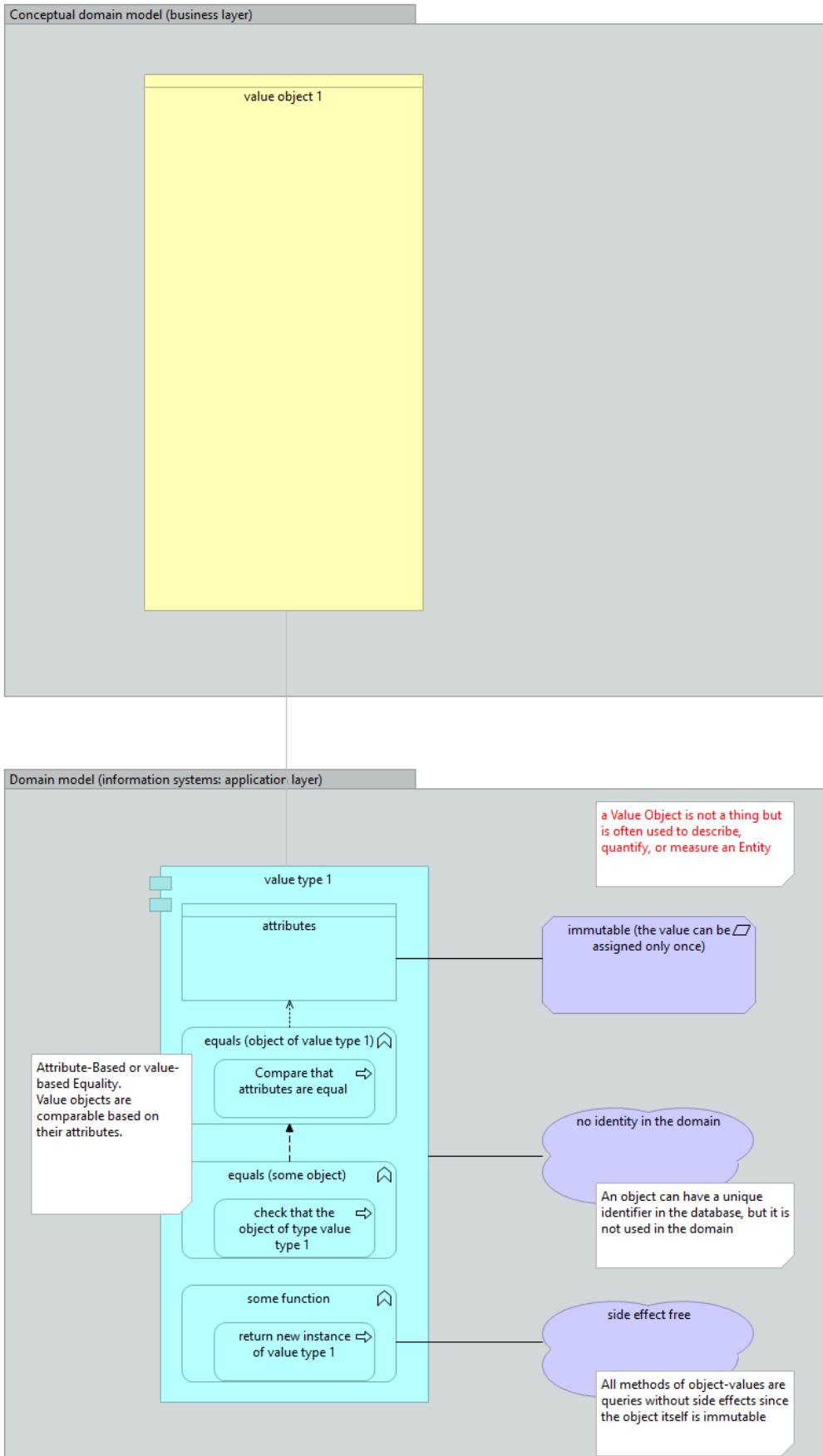


ENTITIES

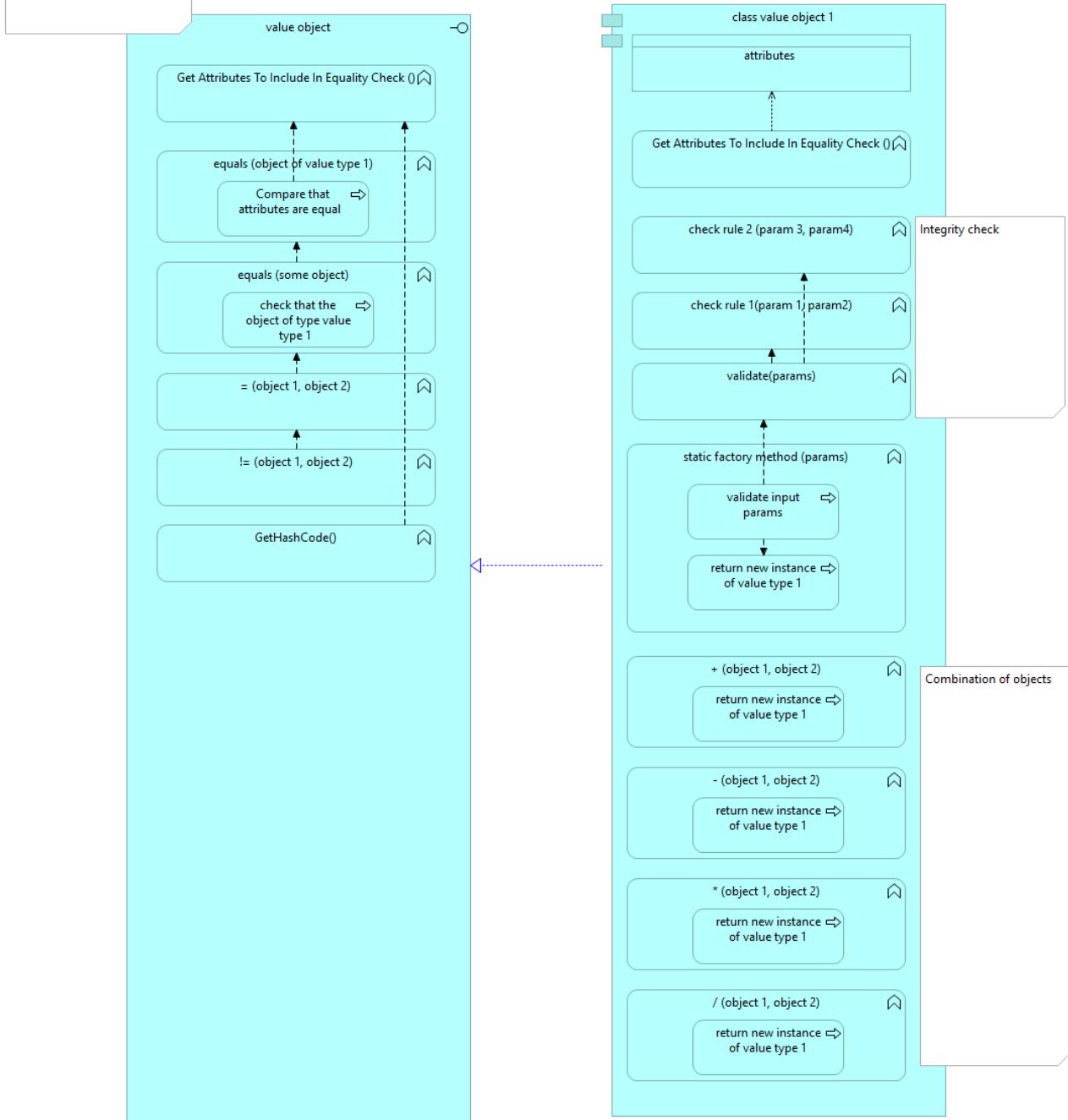




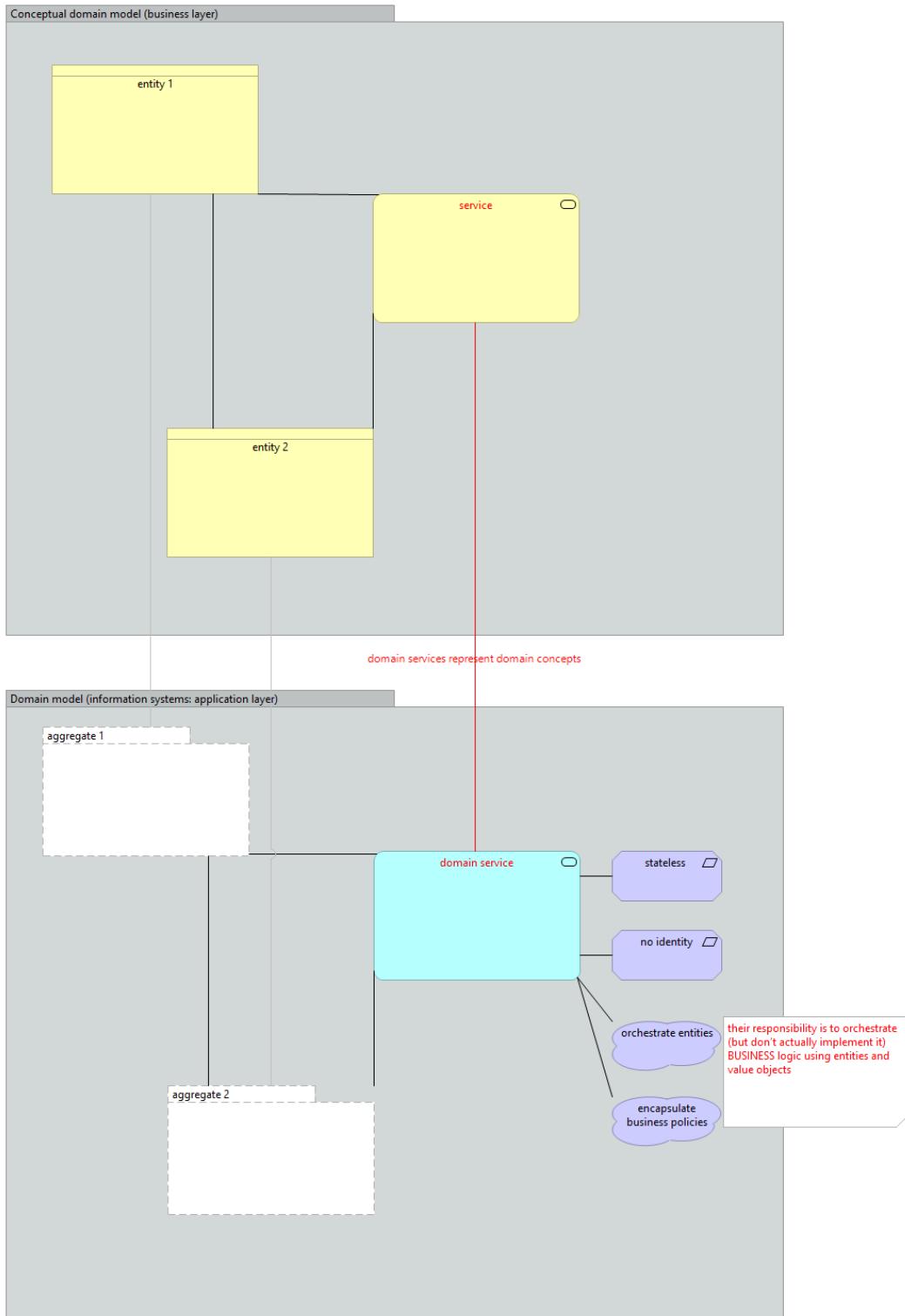
VALUE-OBJECTS



See example: Fowler's Money

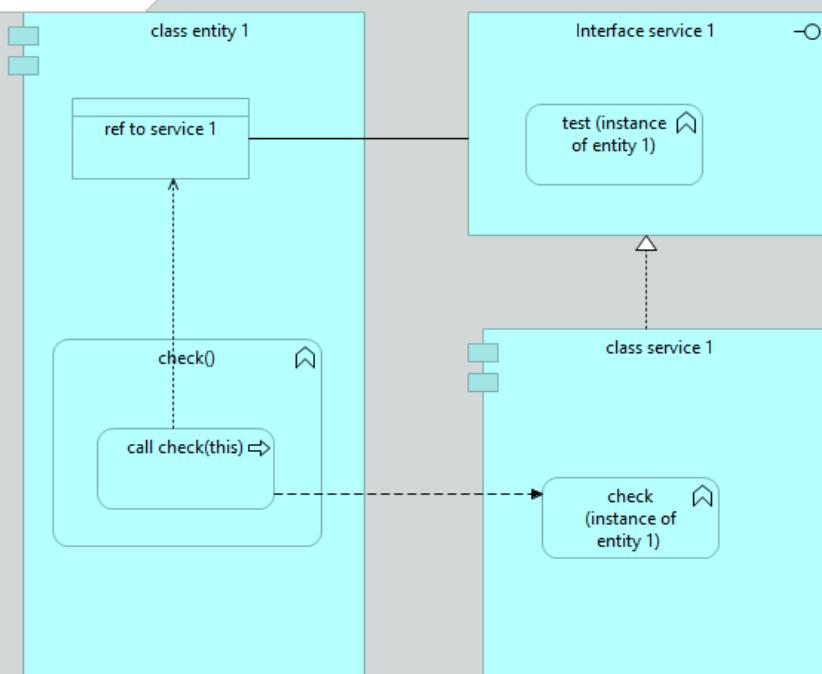


DOMAIN SERVICES

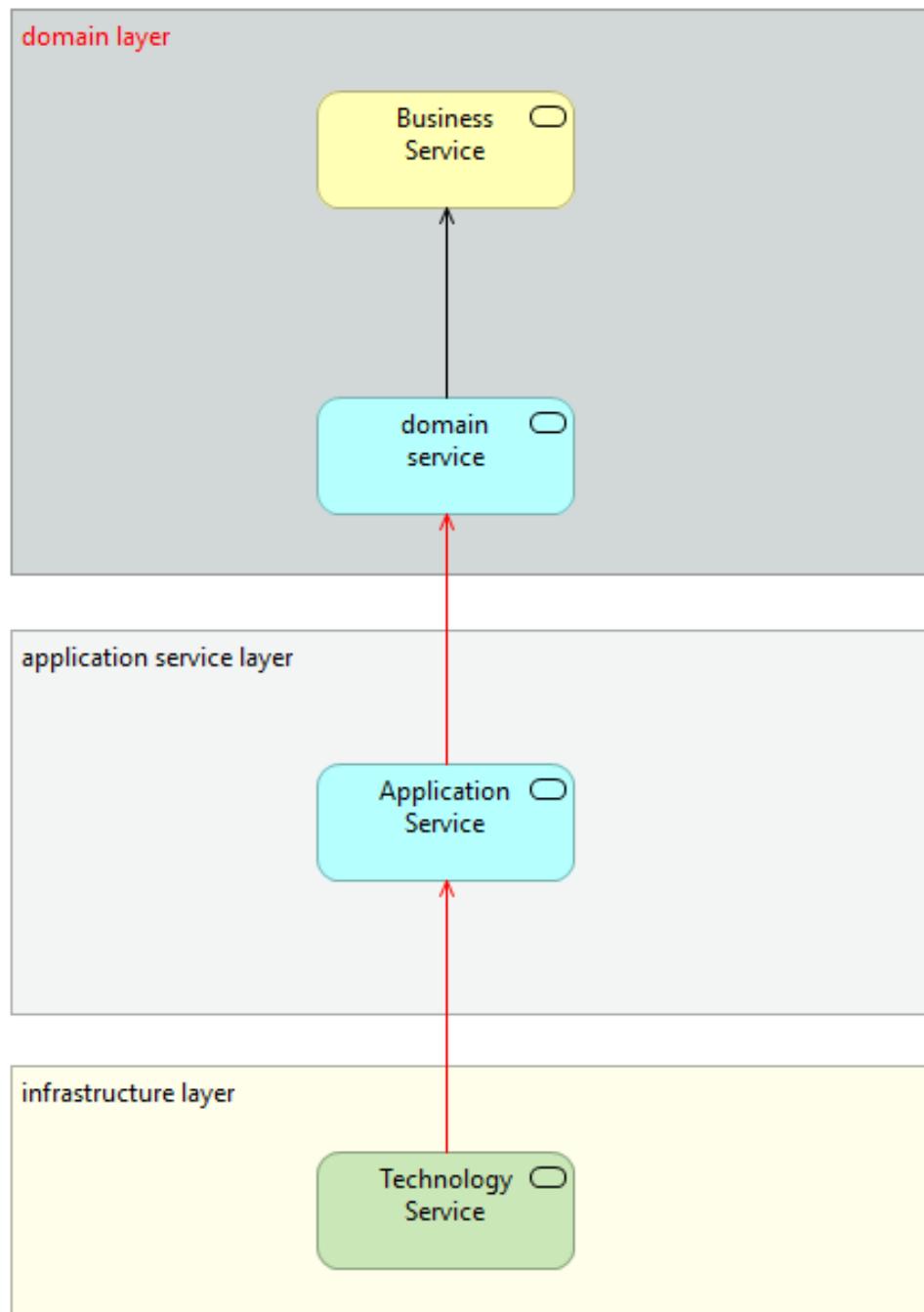


Domain model (information systems: application layer)

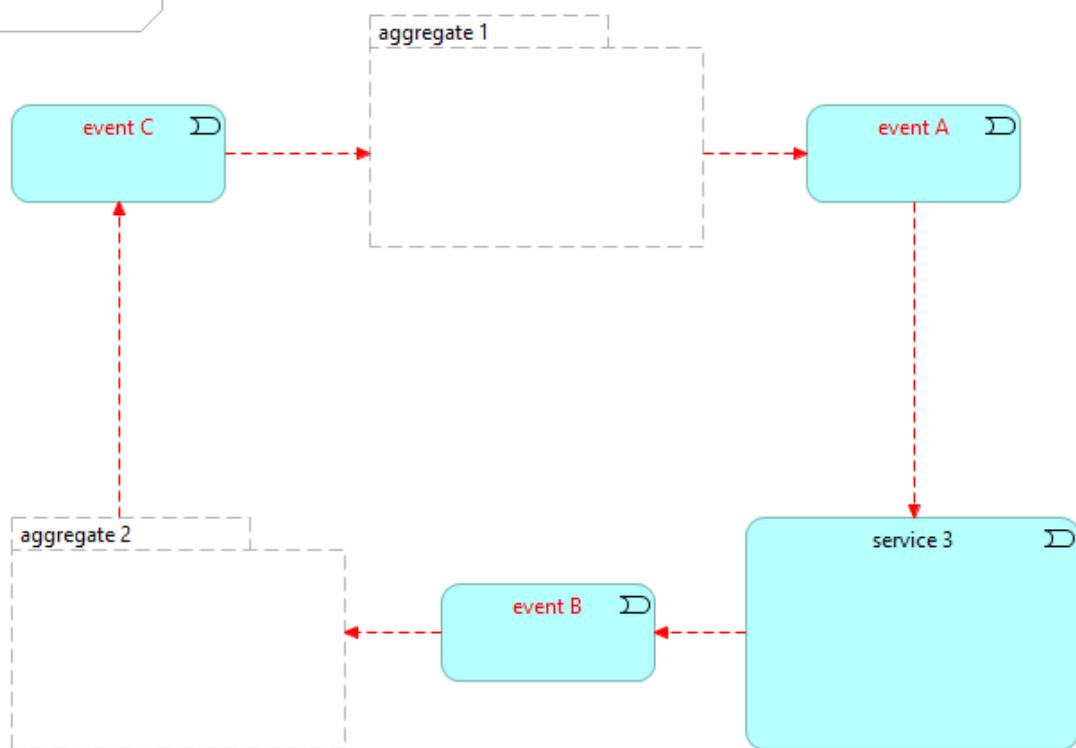
Example of communicating services with aggregates via direct service call

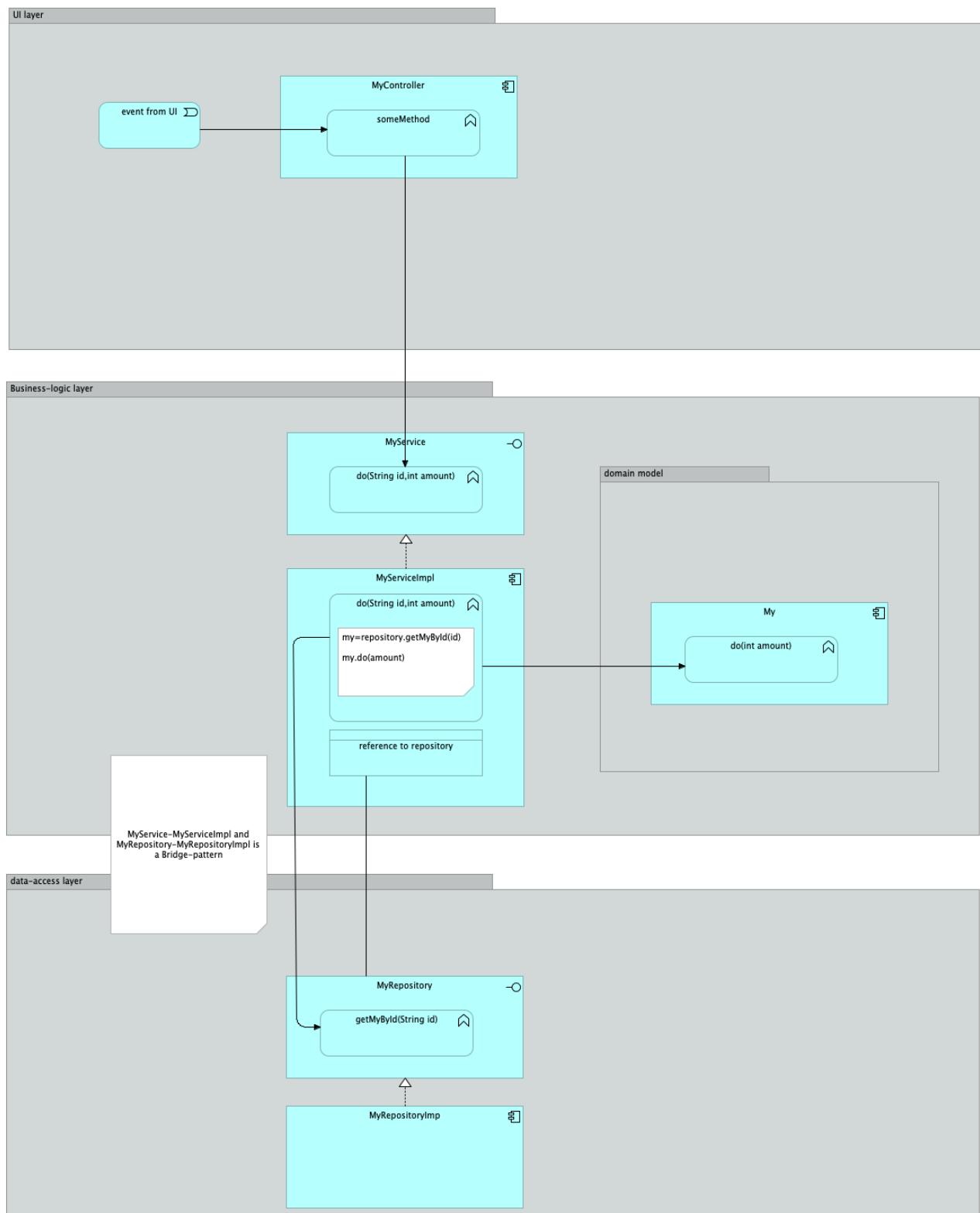


Services support each other

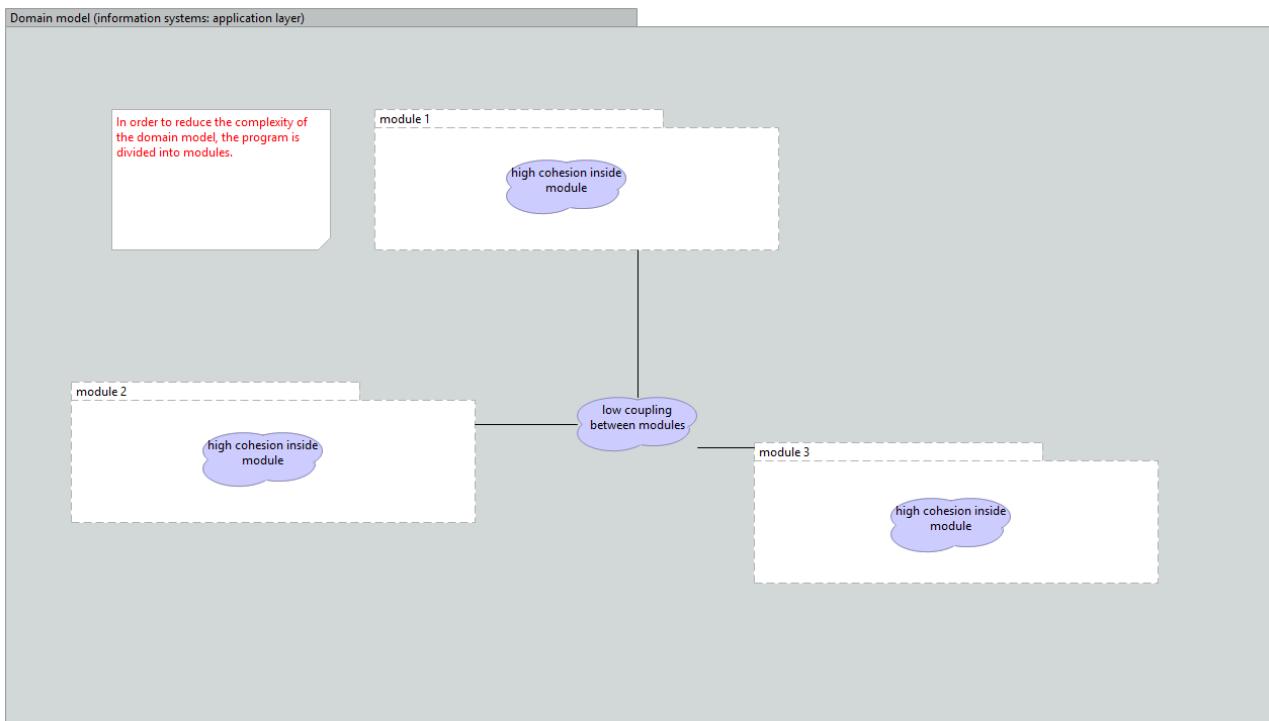


Example of communicating services with aggregates through events

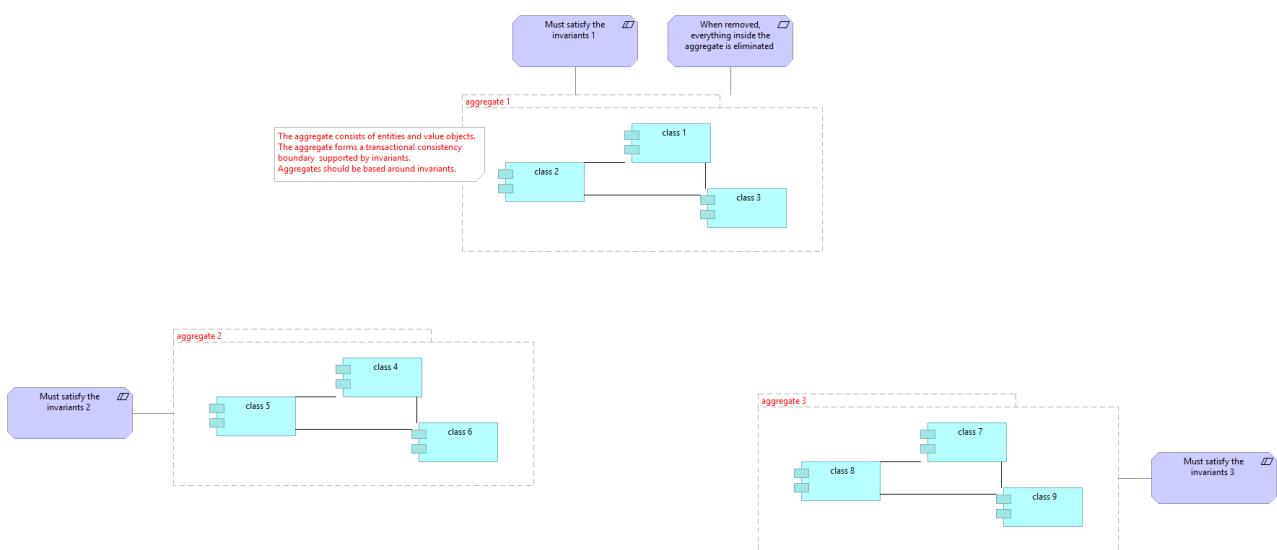




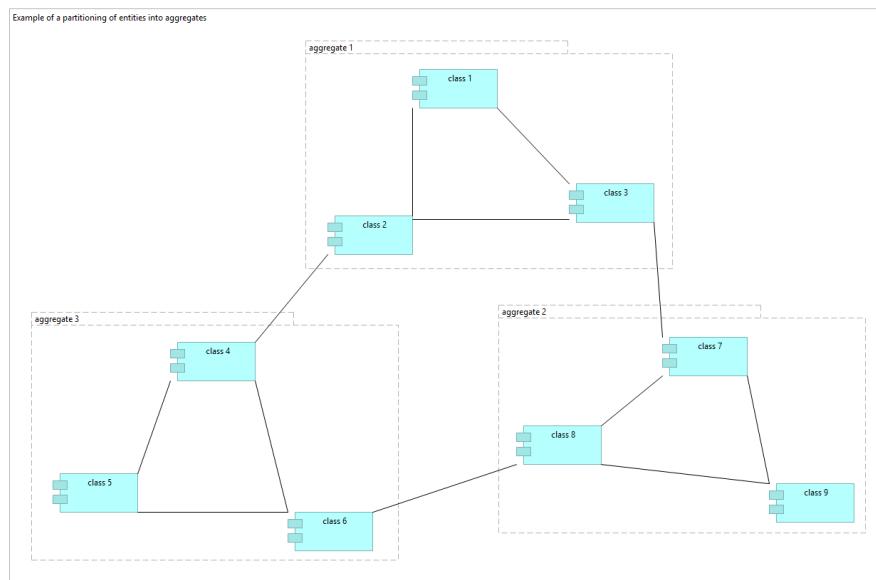
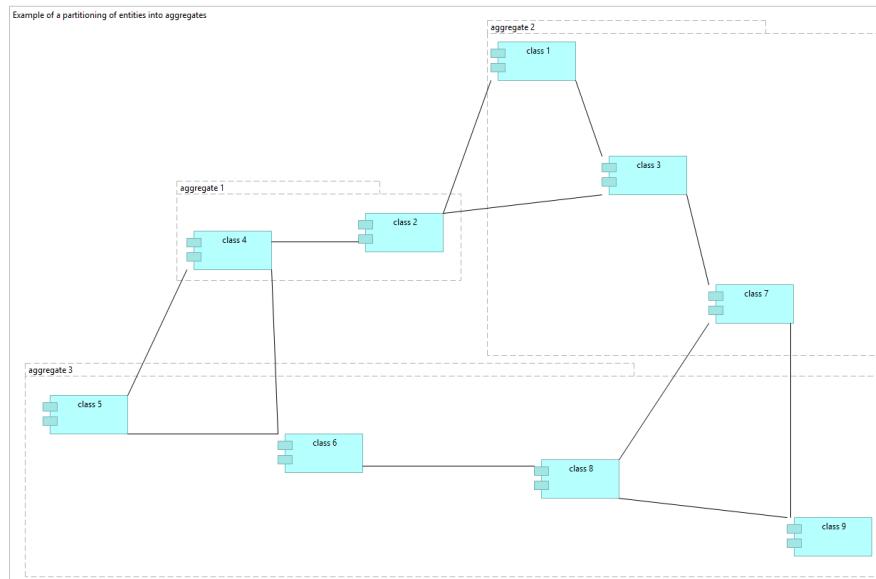
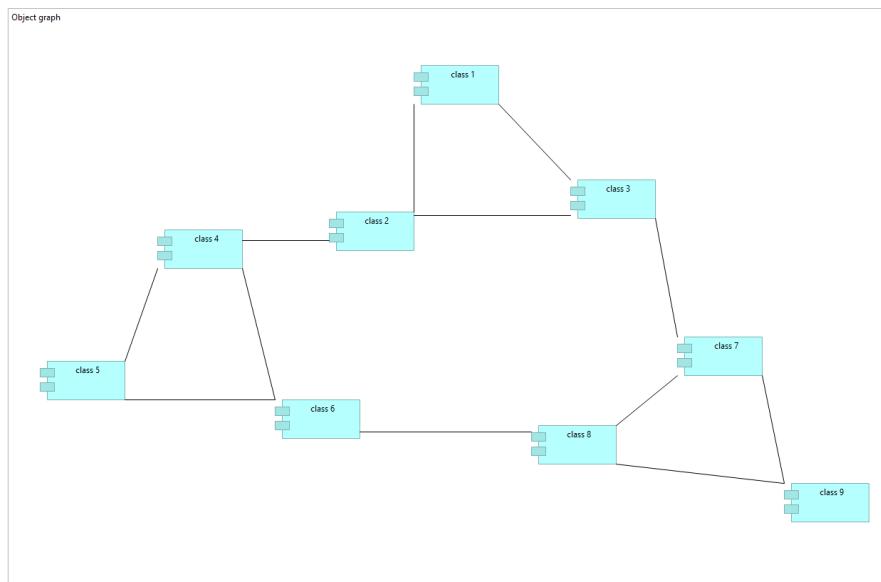
MODULES



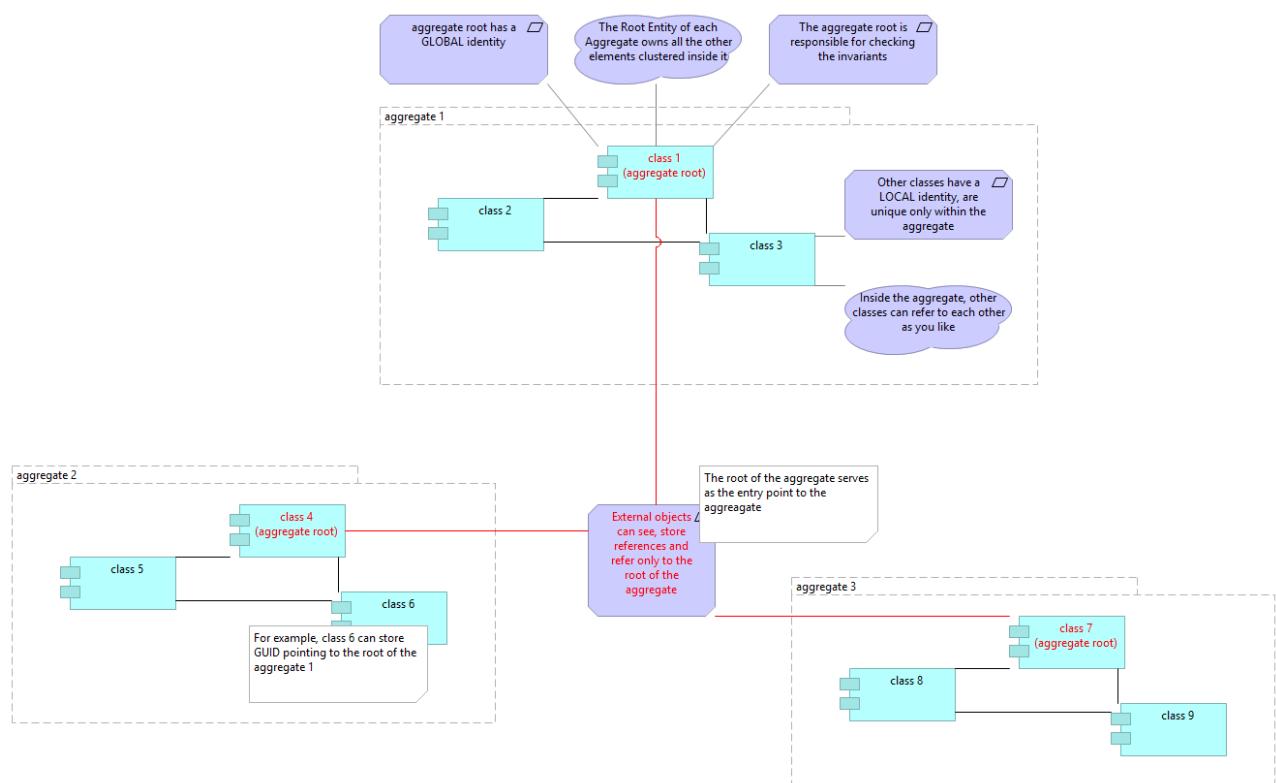
AGGREGATES



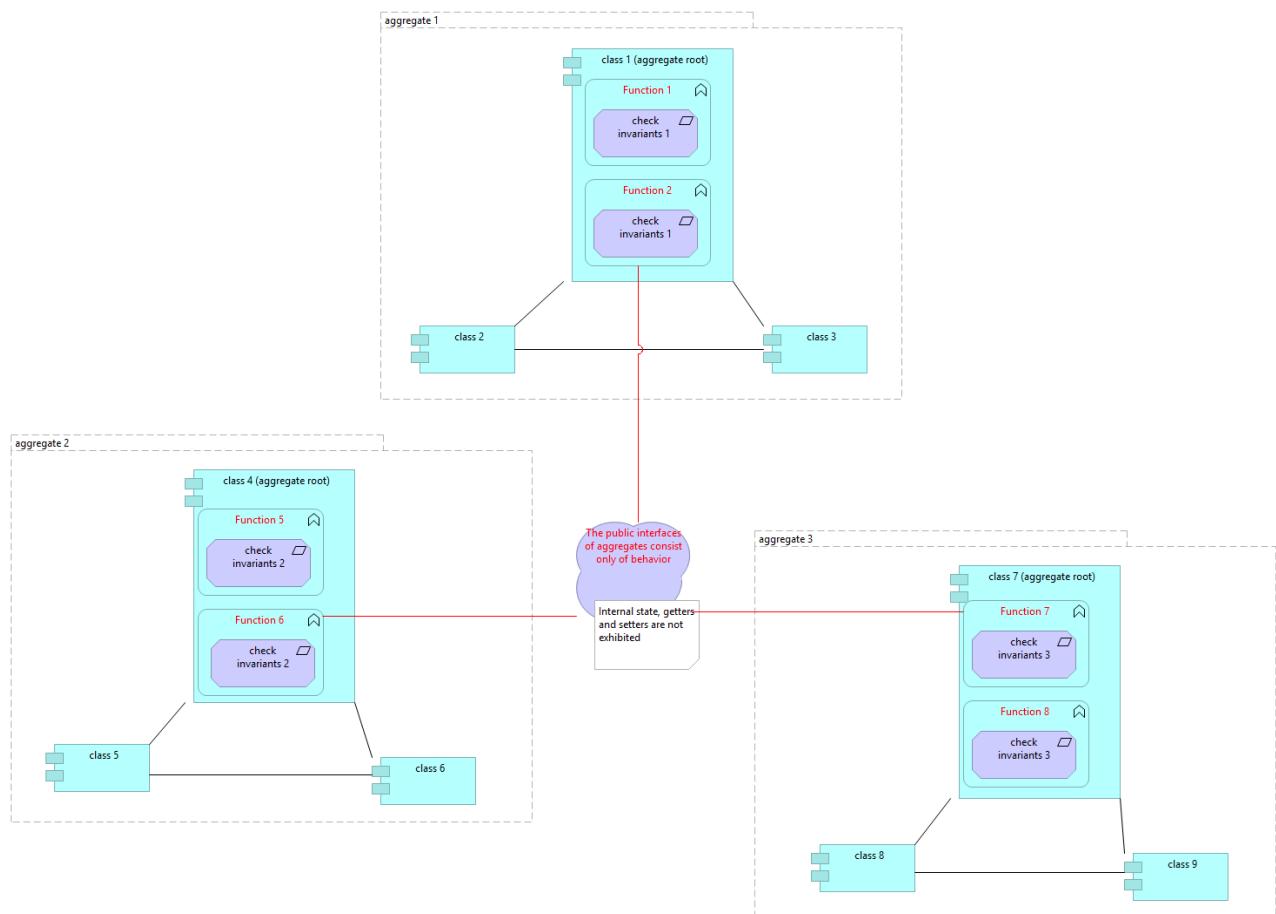
In the course of modeling, think about different ways of splitting the model into aggregates



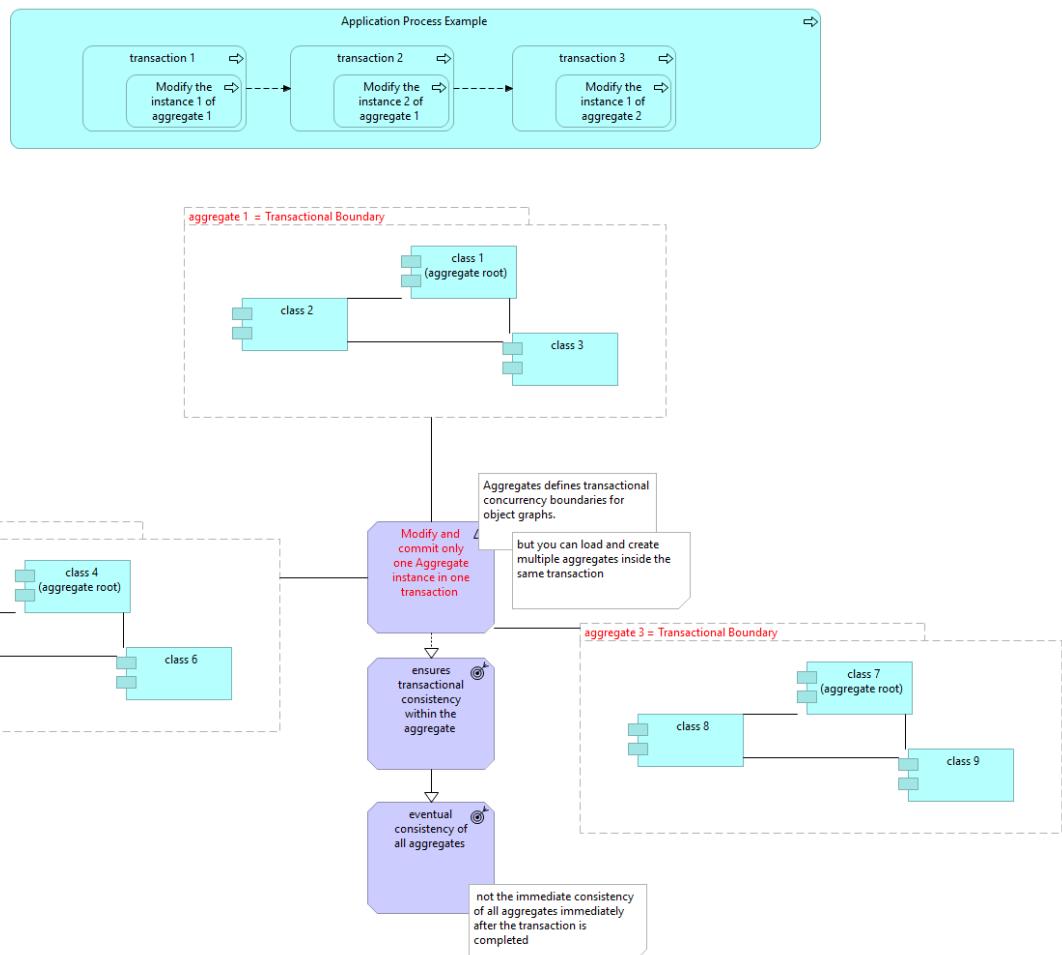
AGGREGATE ROOT



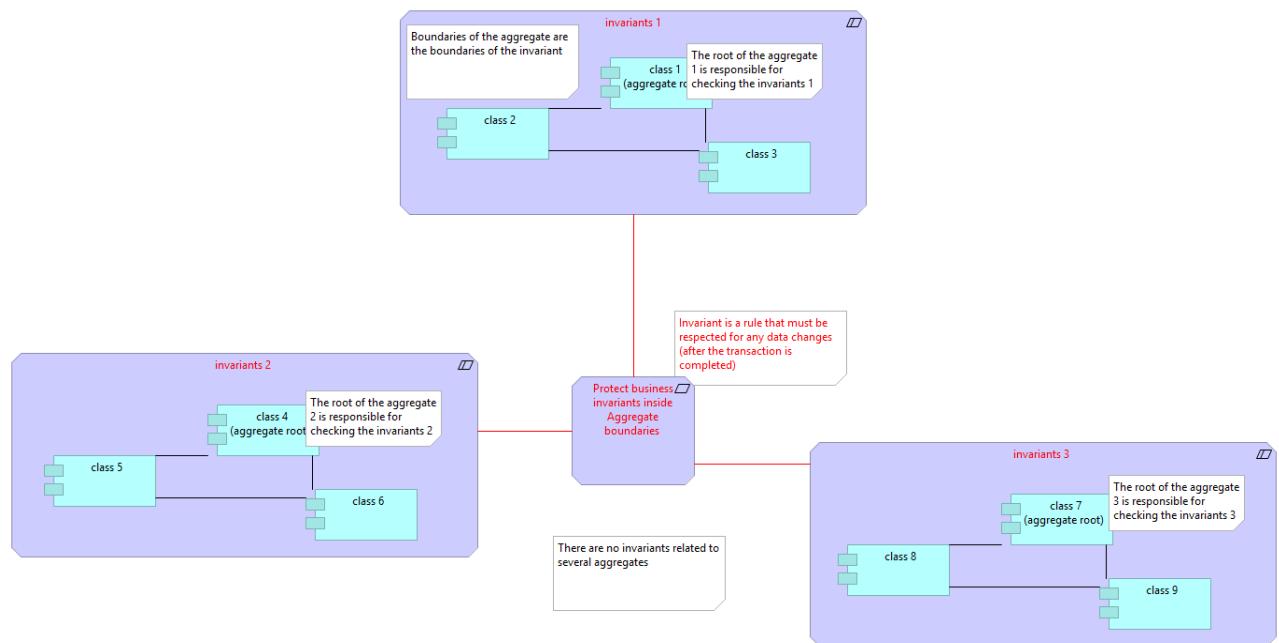
BEHAVIOR-FOCUSED AGGREGATE ROOT



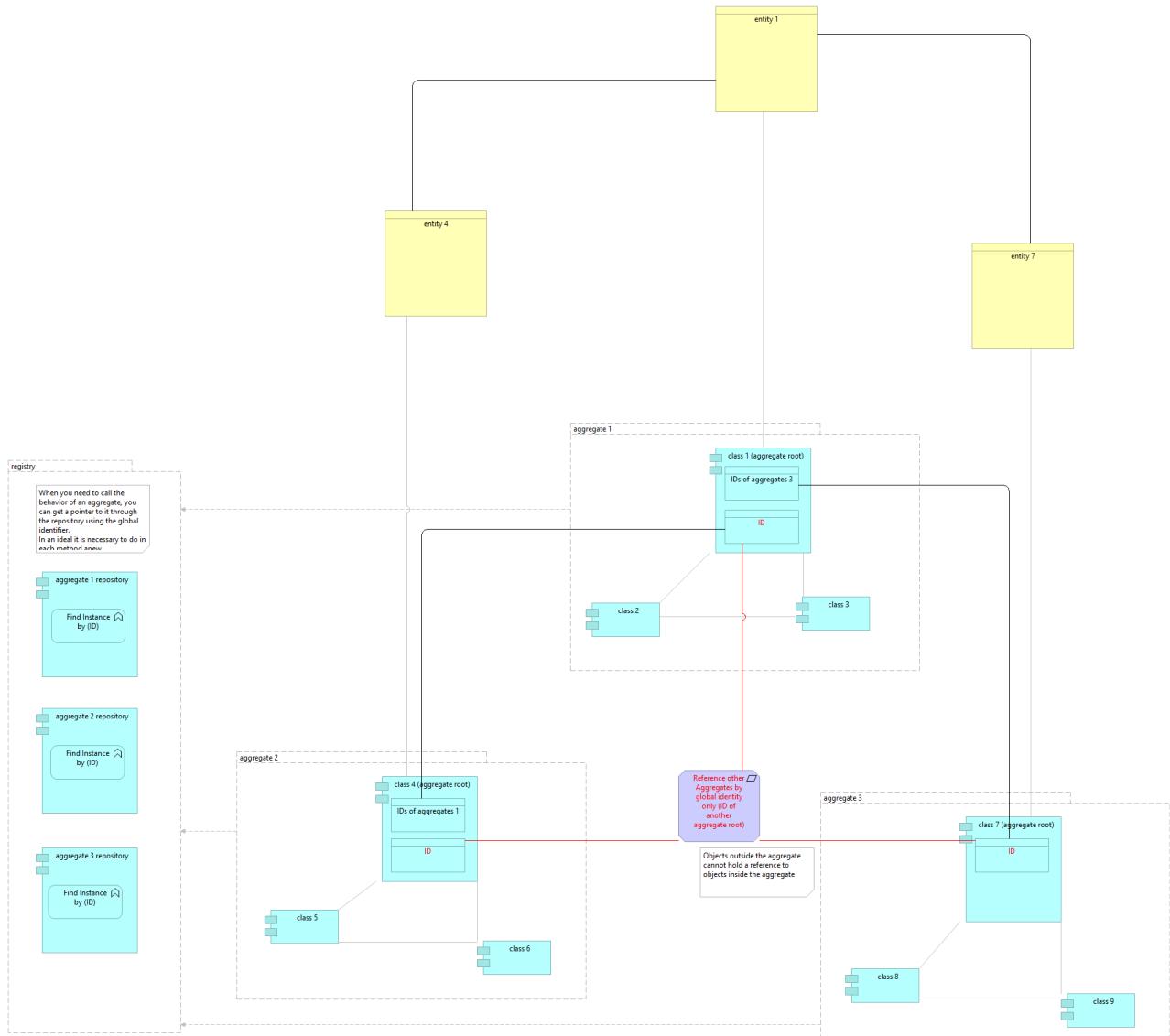
MODIFY AND COMMIT ONLY ONE AGGREGATE INSTANCE IN ONE TRANSACTION



PROTECT BUSINESS INVARIANTS INSIDE AGGREGATE BOUNDARIES



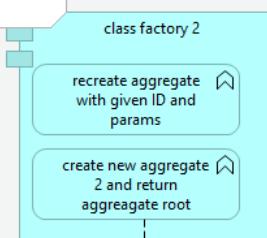
REFERENCE OTHER AGGREGATES BY IDENTITY ONLY



FACTORIES

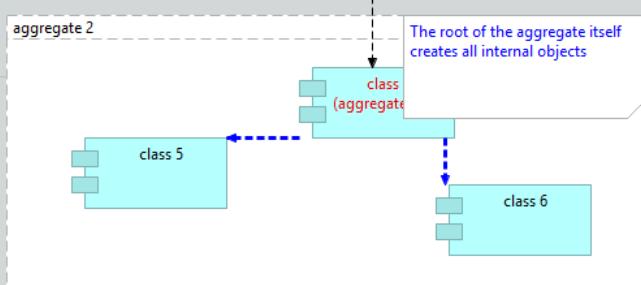
application service layer

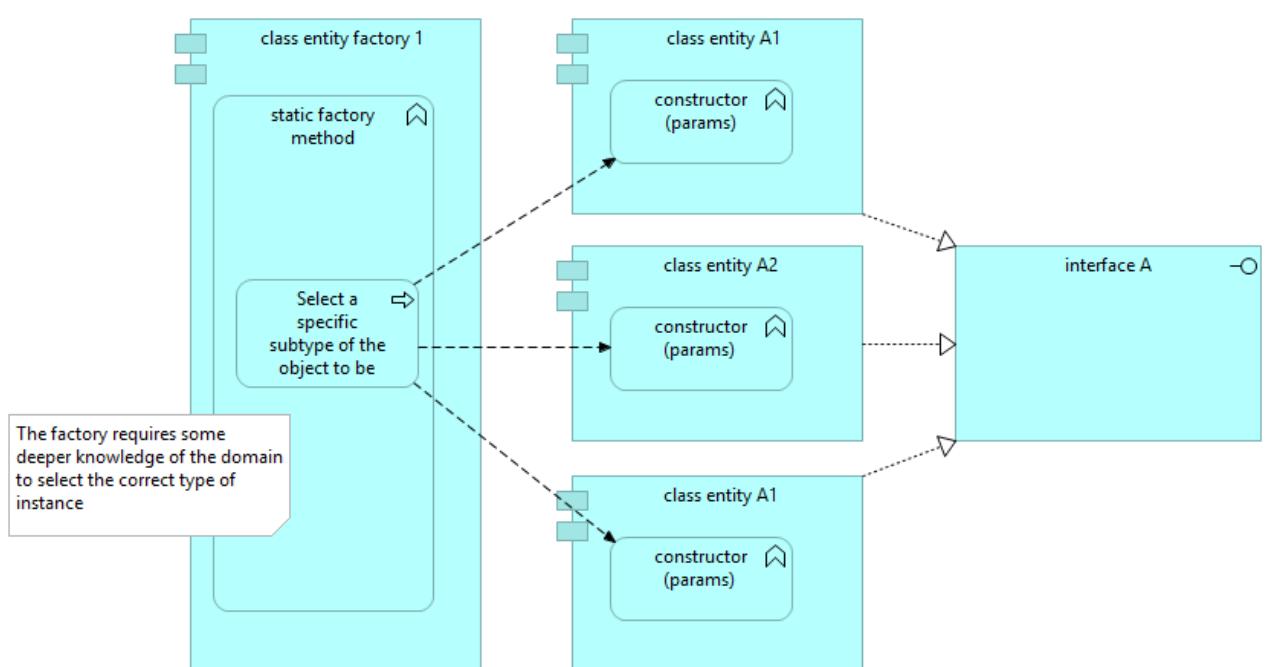
Use factory to create aggregates, complex entities and value objects.
Use factory to re-create domain objects from persistent storage.
The factory creates an aggregate entirely, with the satisfaction of all invariants
GoF pattern: factory



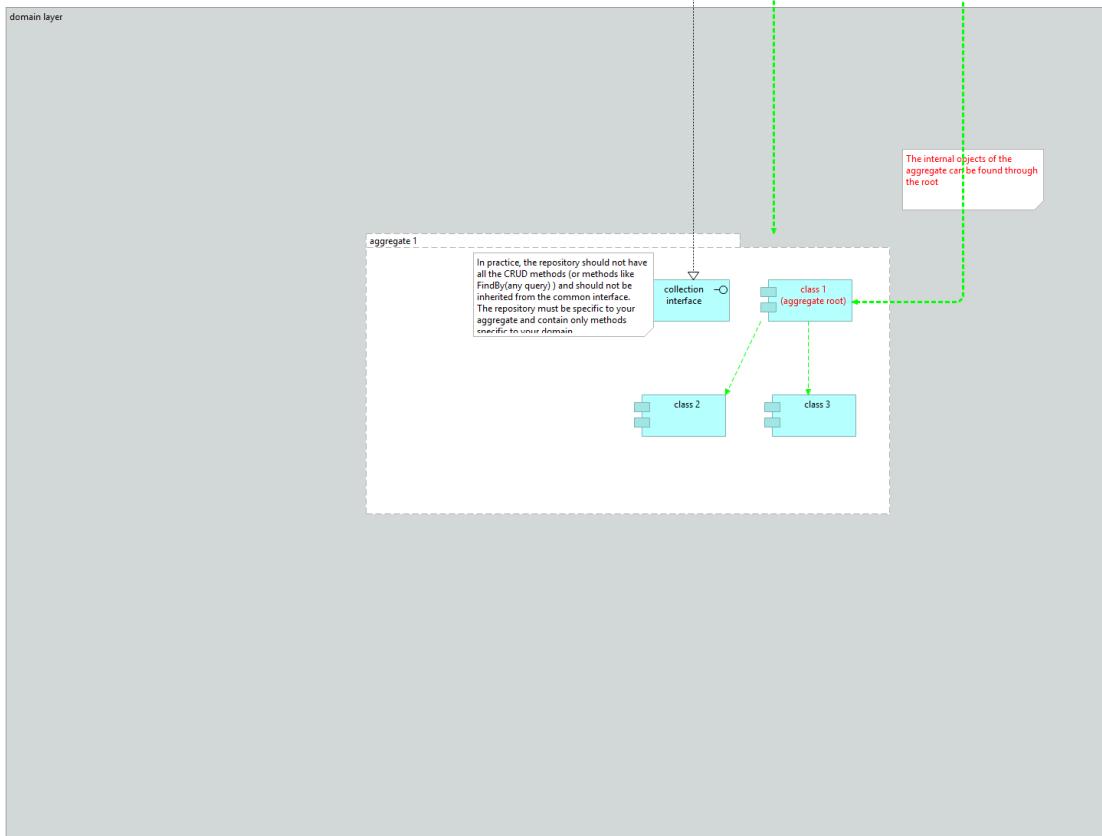
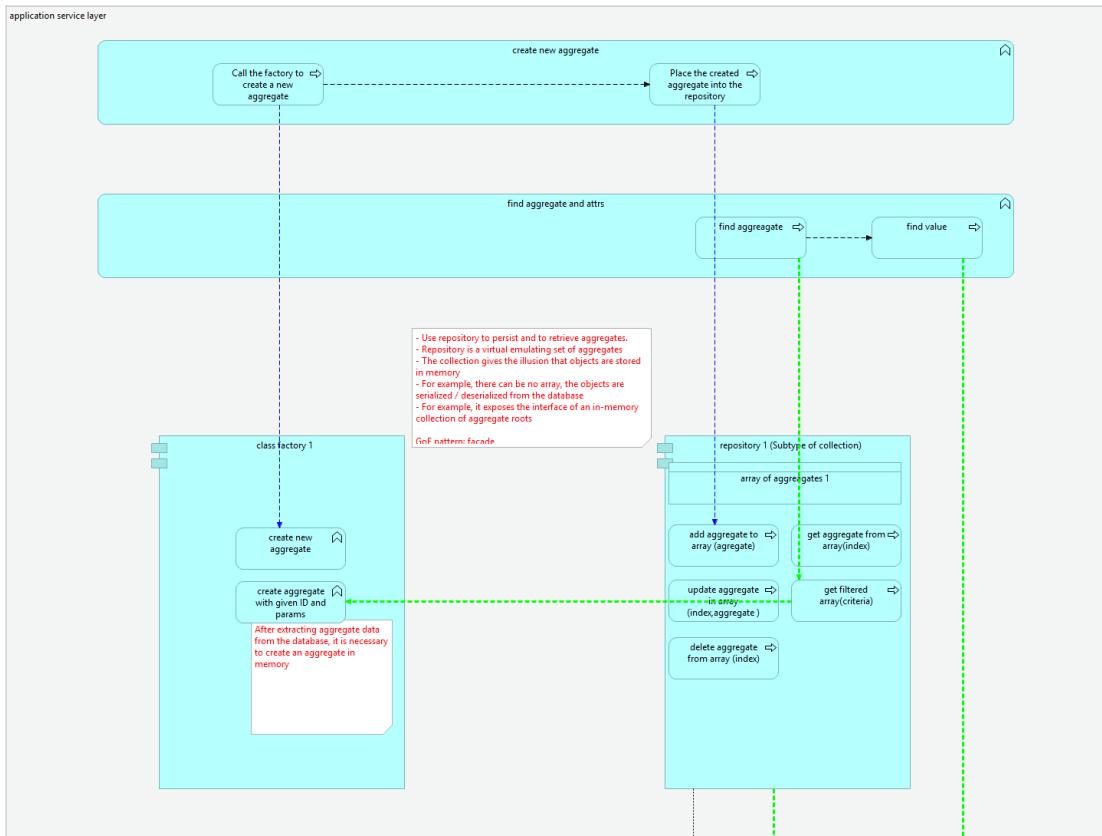
domain layer

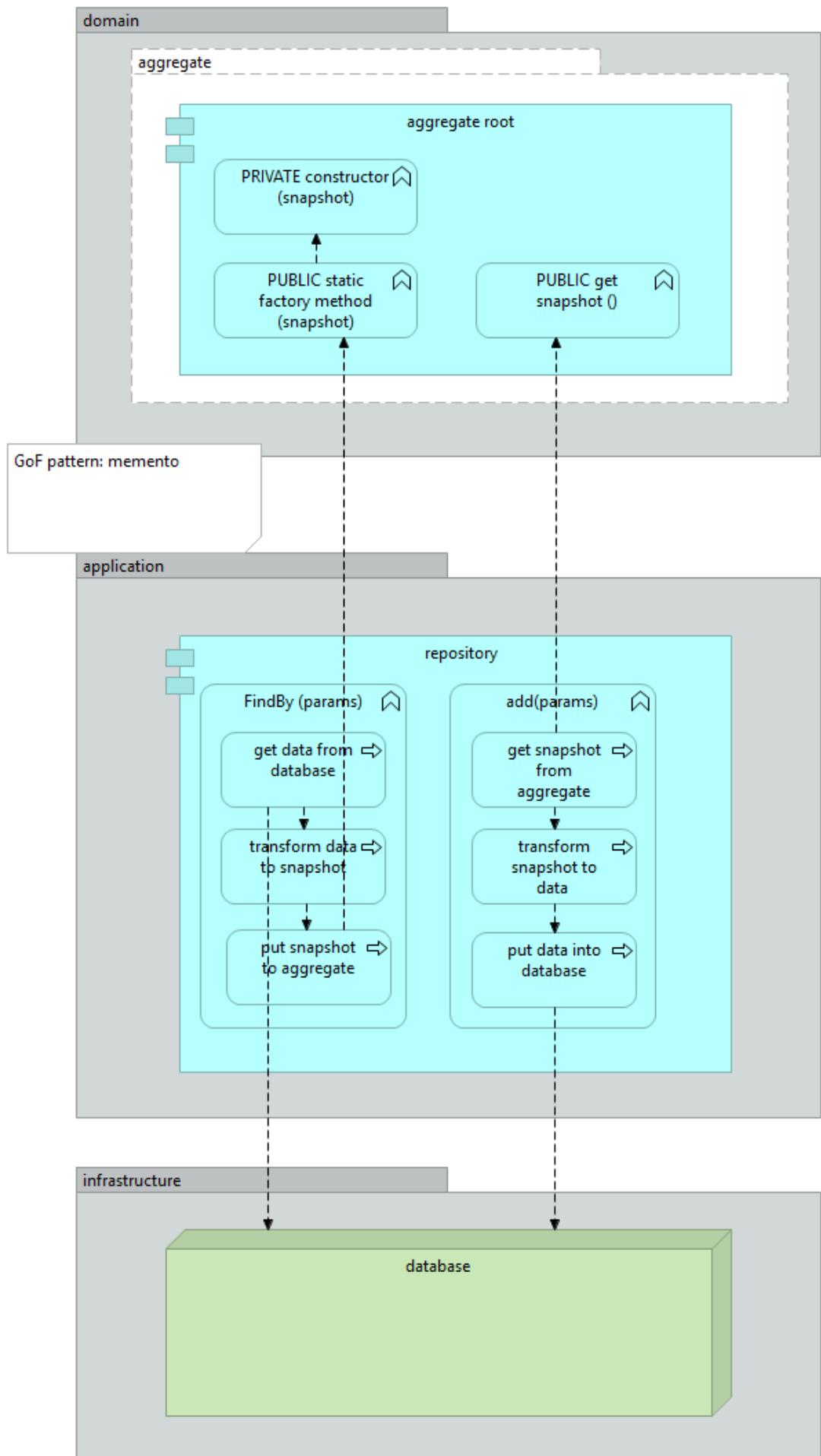
Must satisfy the invariants 2 after creation

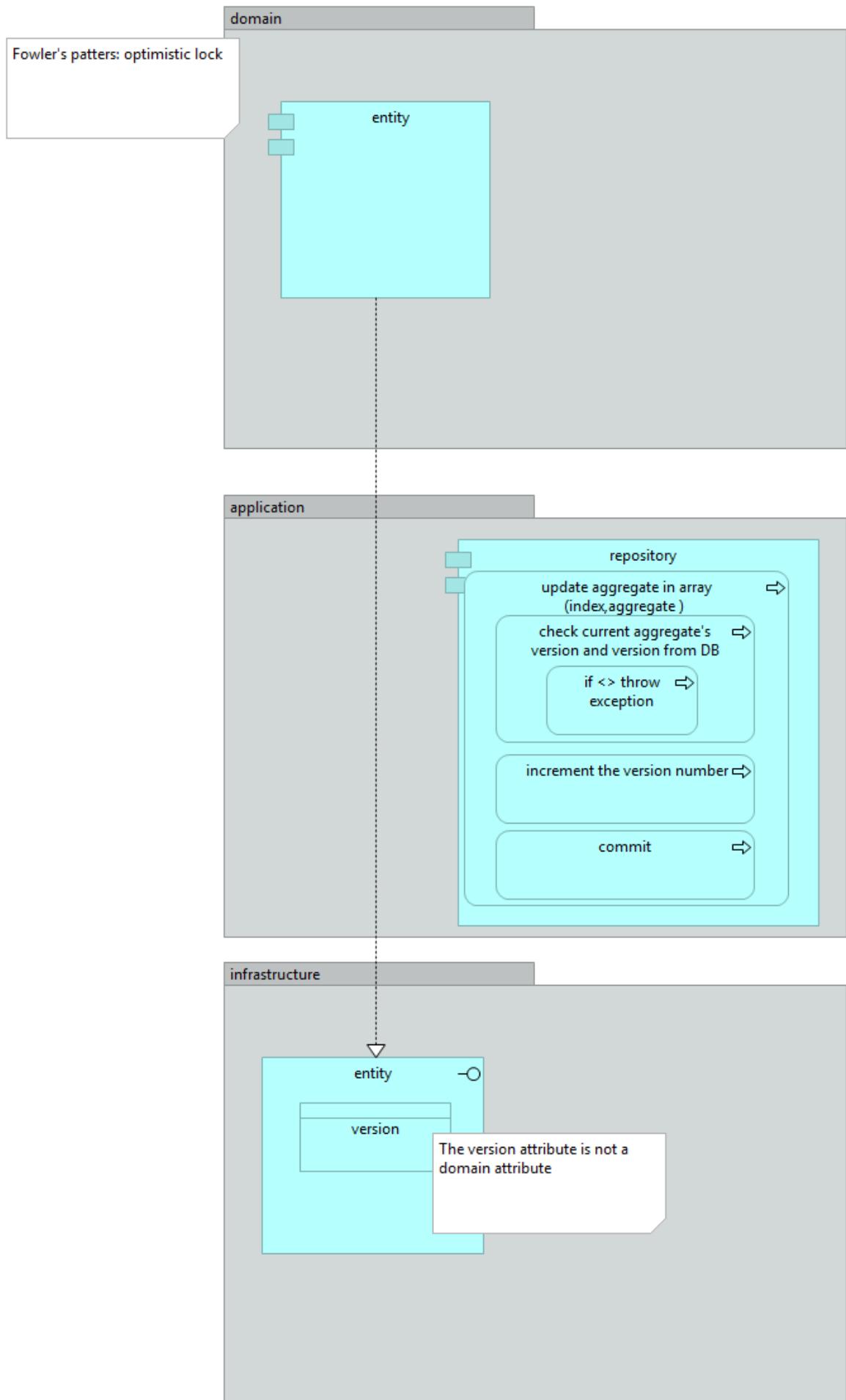


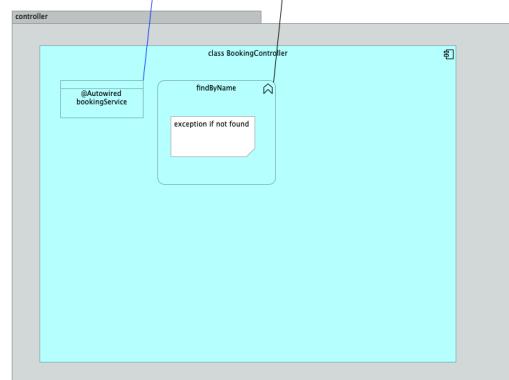
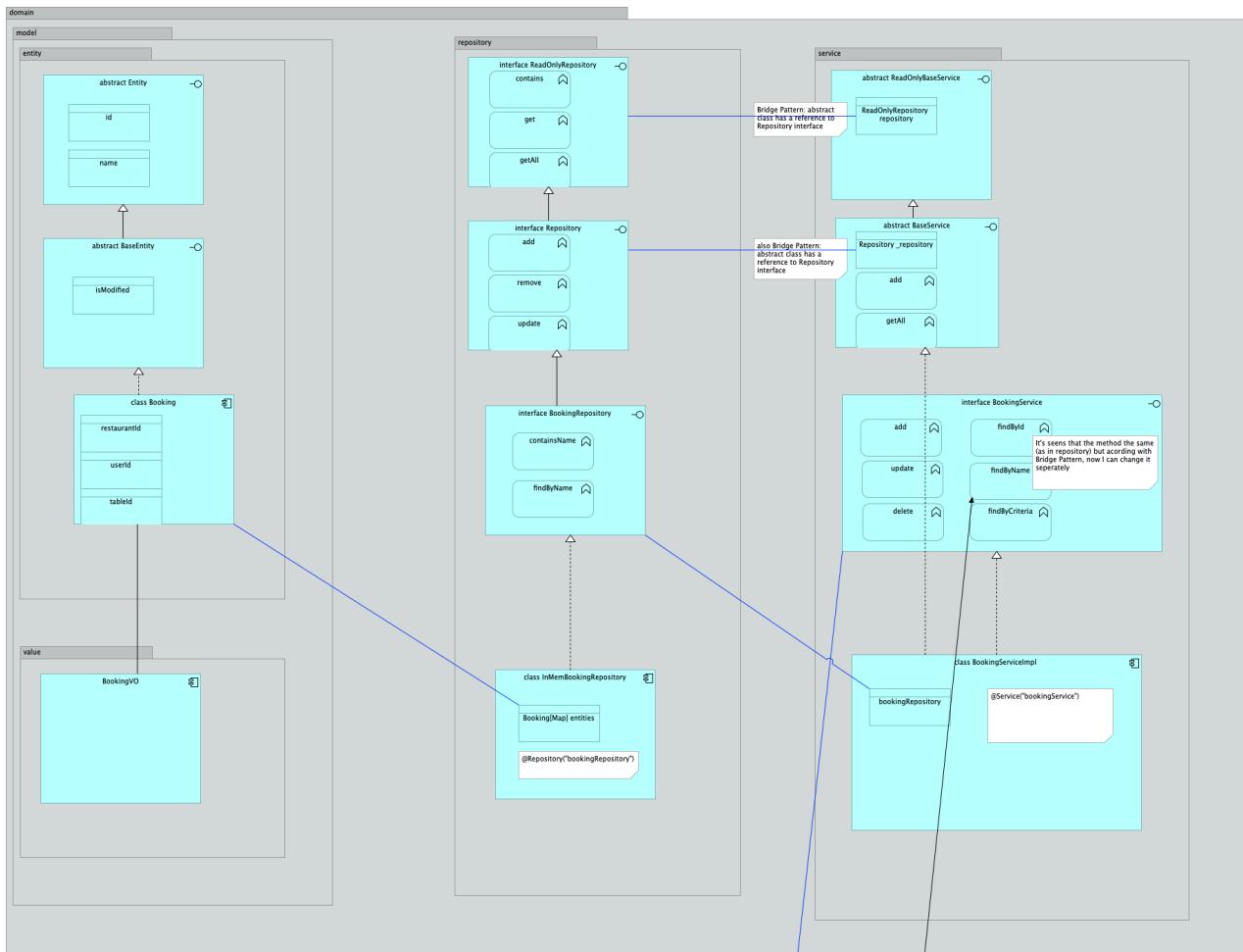


REPOSITORIES









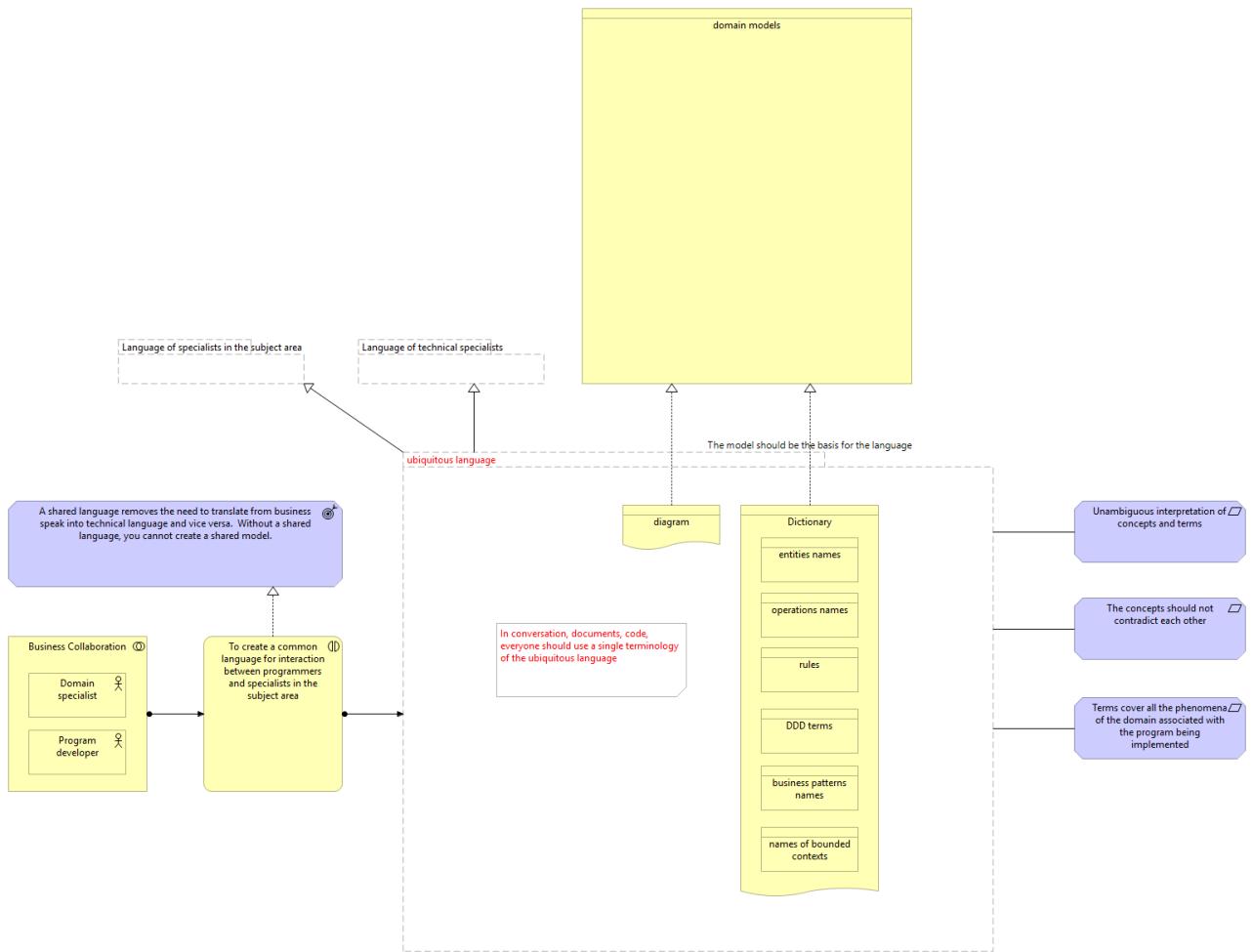
SUPPLE DESIGN

DOMAIN DRIVEN DESIGN

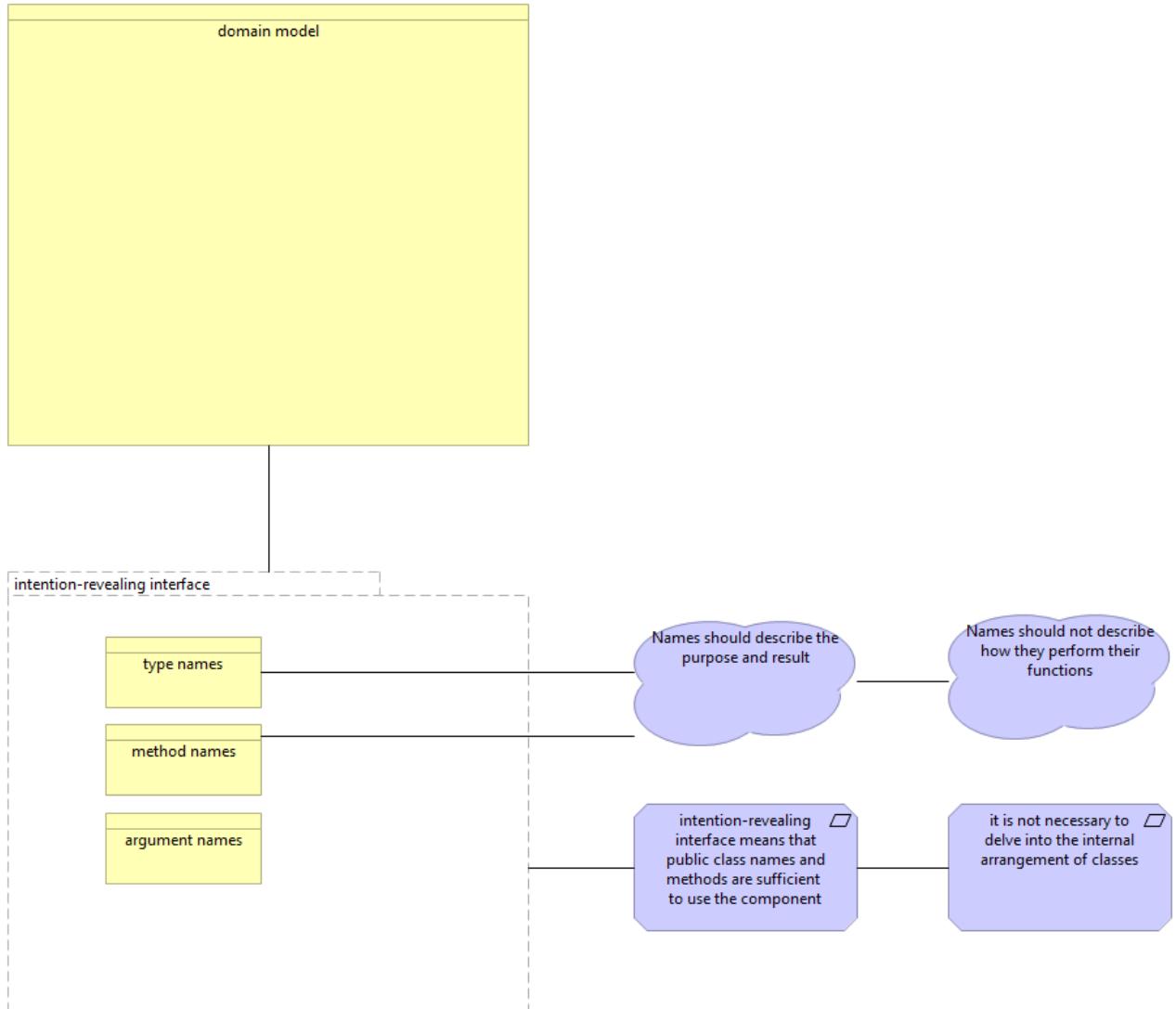
Eric Evans



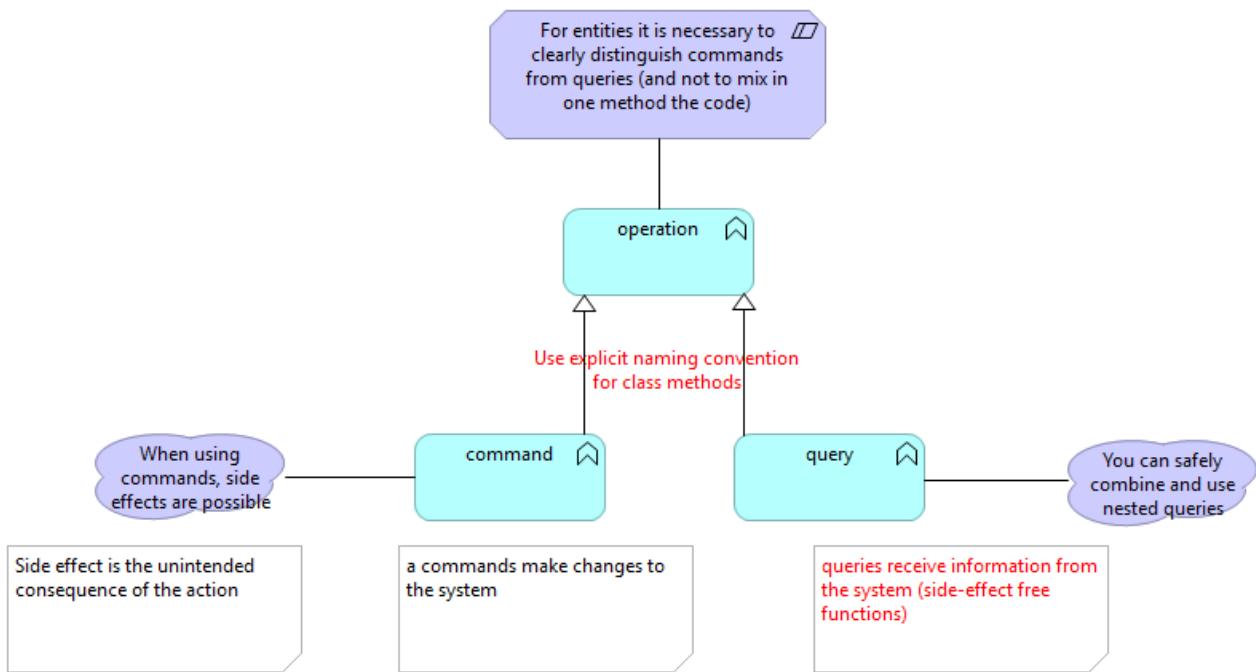
UBIQUITOUS LANGUAGE



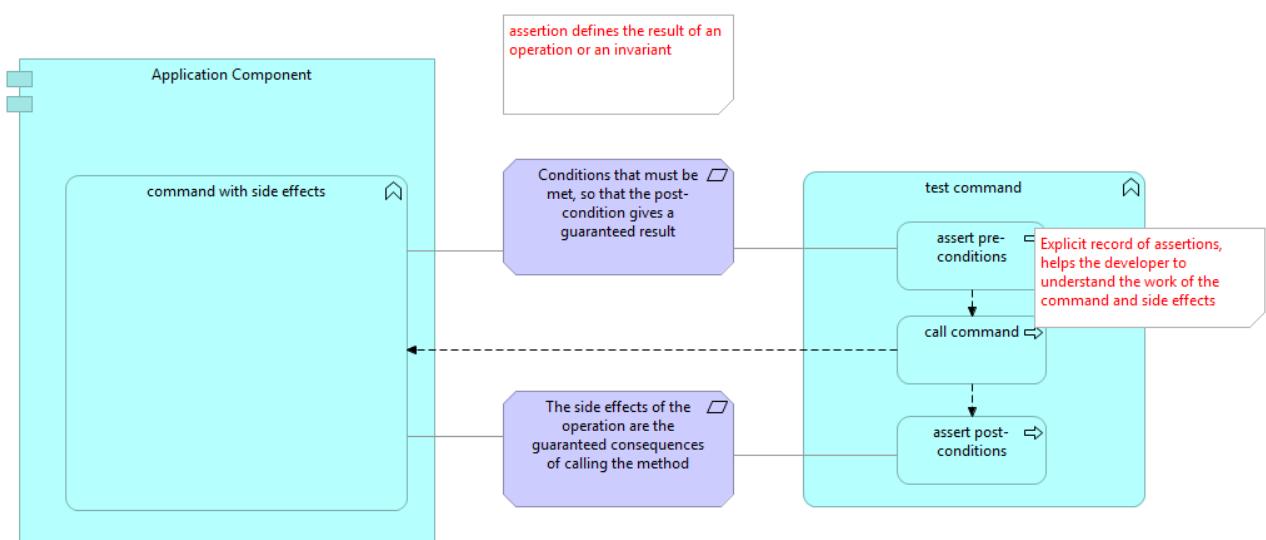
INTENTION-REVEALING INTERFACES



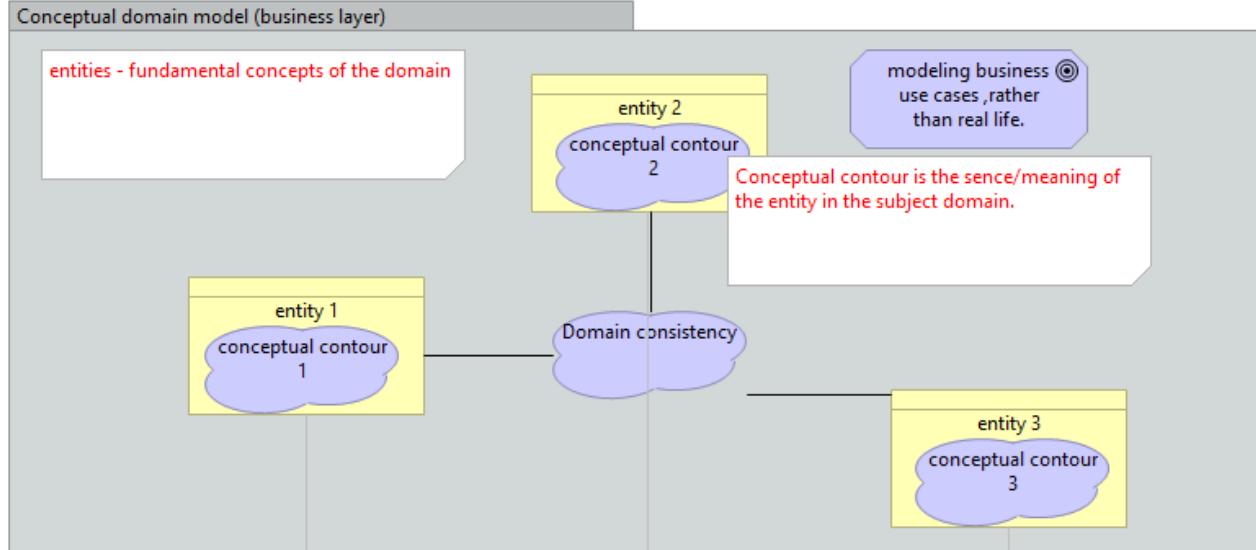
SIDE-EFFECT FREE FUNCTIONS



ASSERTIONS

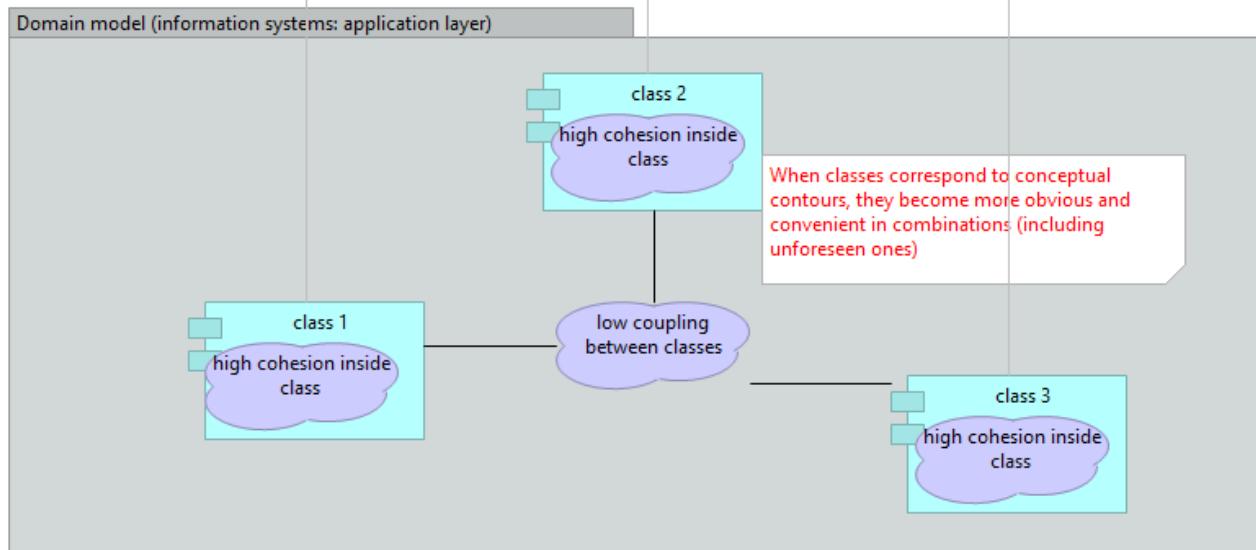


CONCEPTUAL CONTOURS

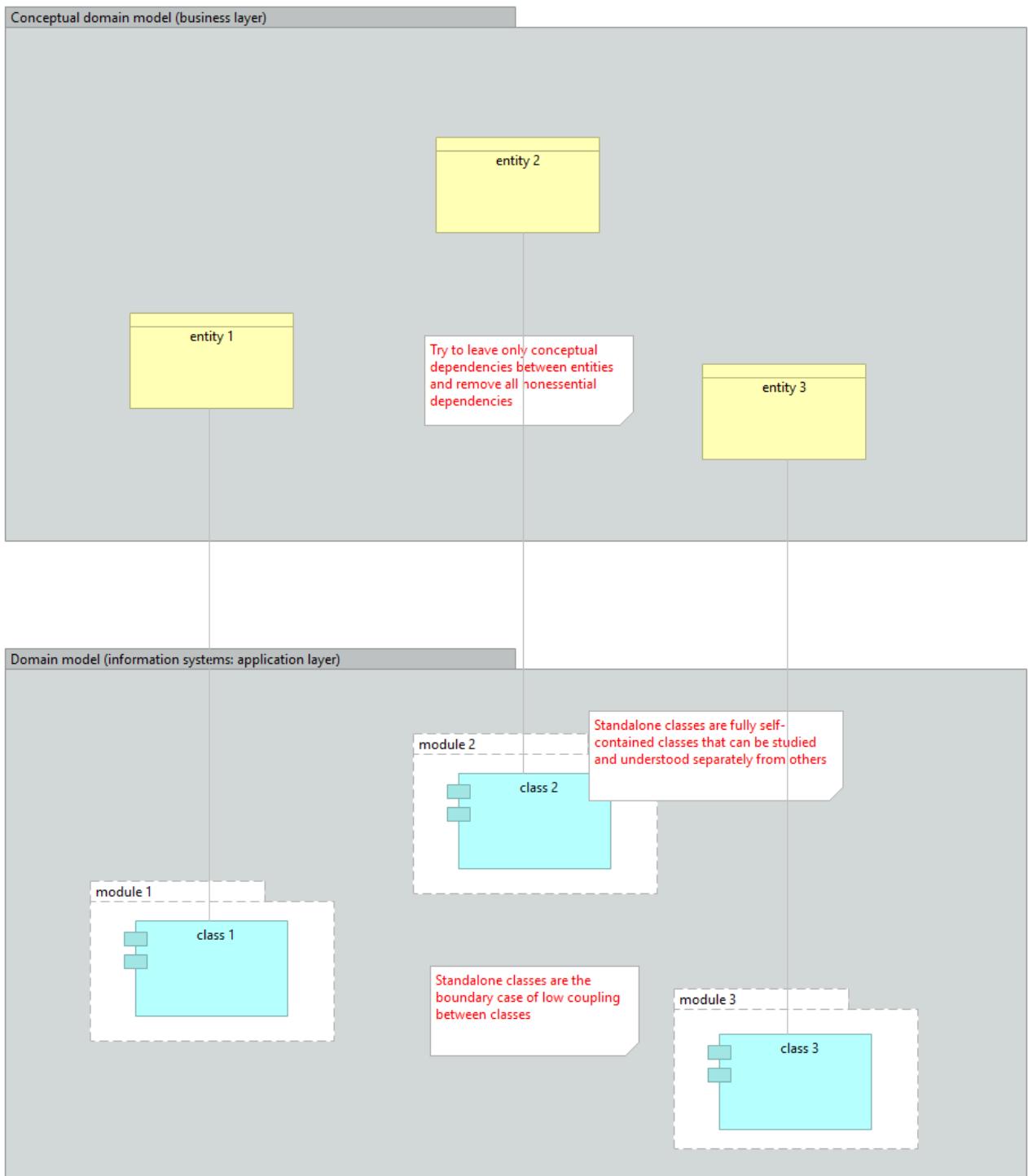


Classes are a reflection of the conceptual/semantic contours of the domain

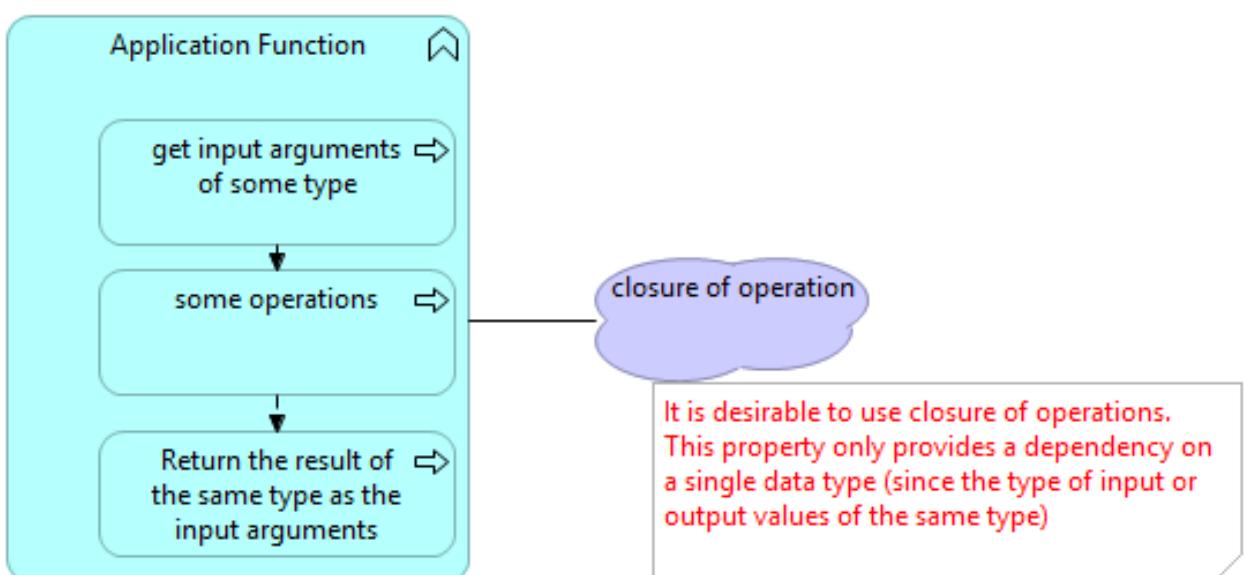
All entities of the domain and classes must correspond to each other and have meaning



STANDALONE CLASSES



CLOSURE OF OPERATIONS



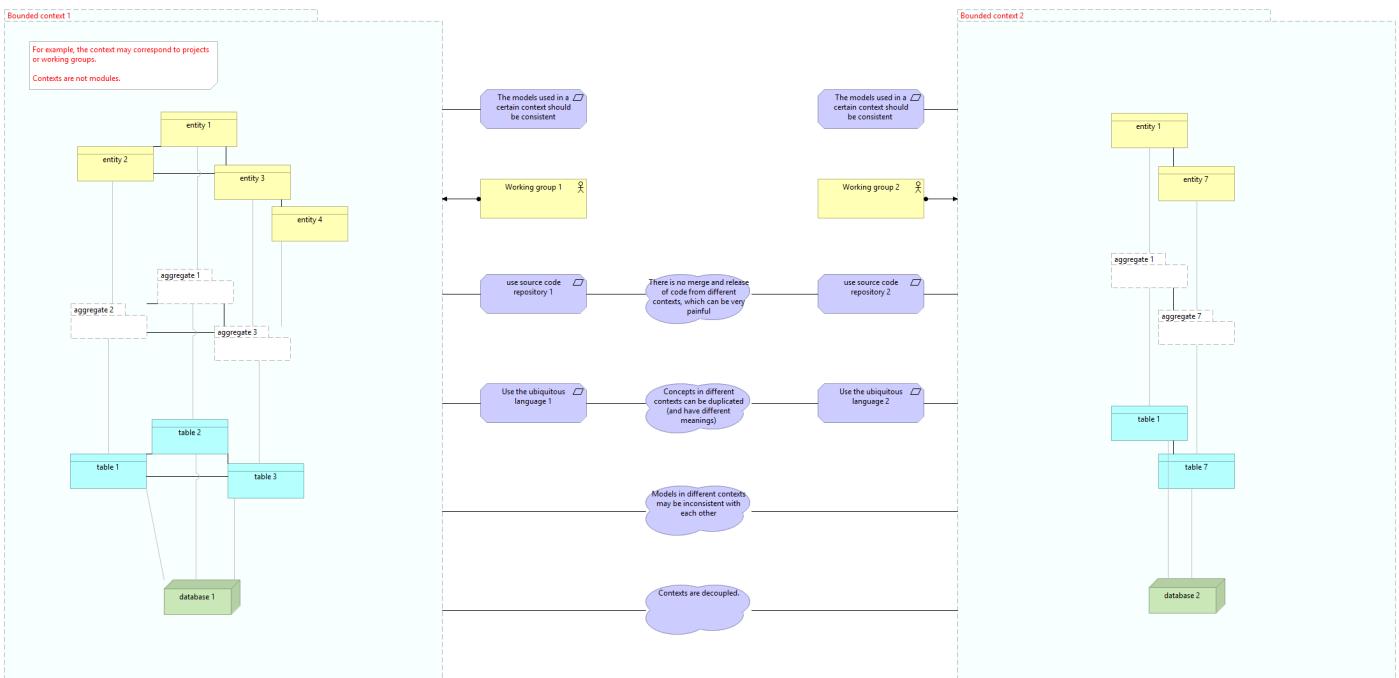
MODEL INTEGRITY AND CONTEXT

DOMAIN DRIVEN DESIGN

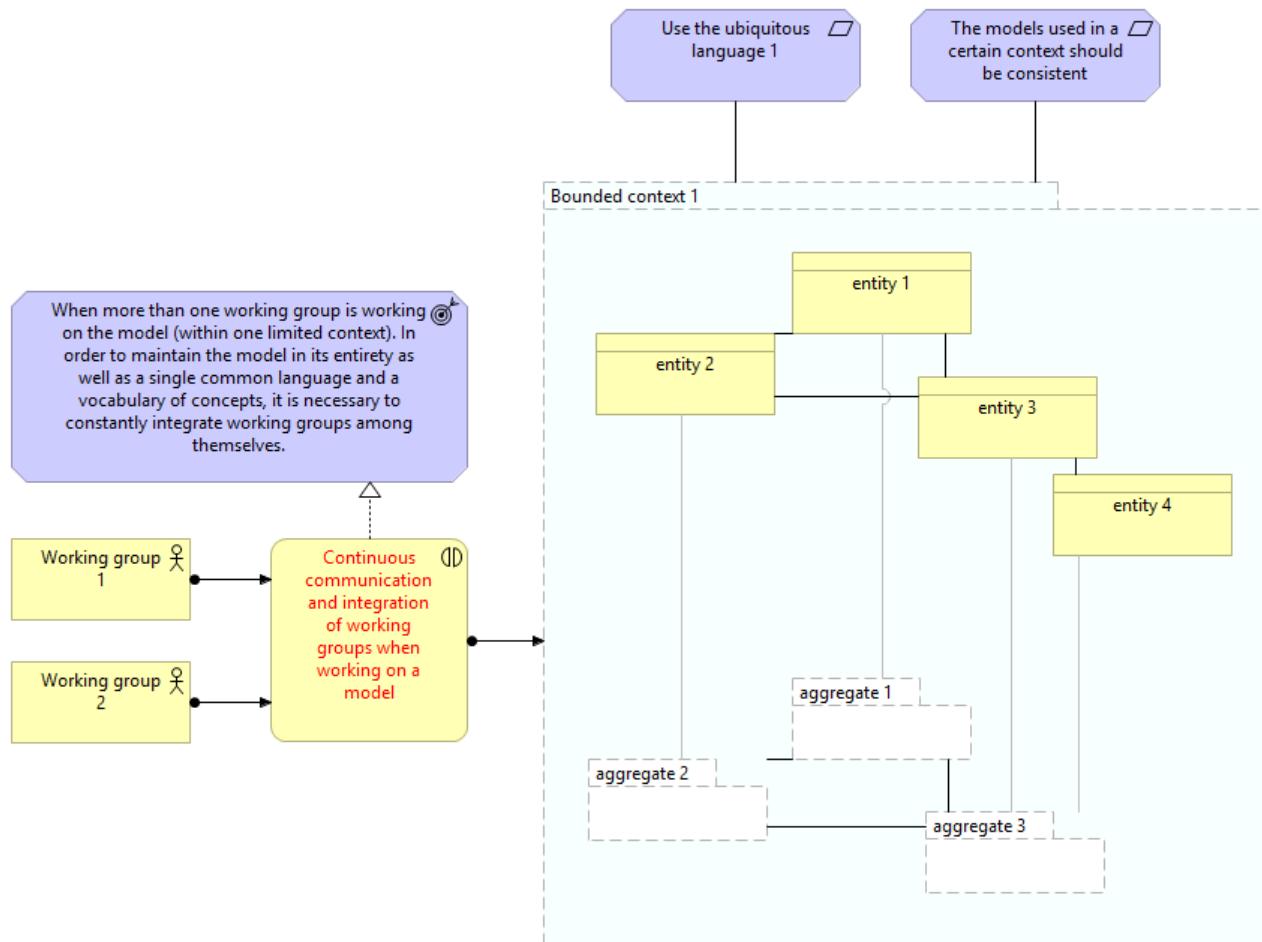
Eric Evans



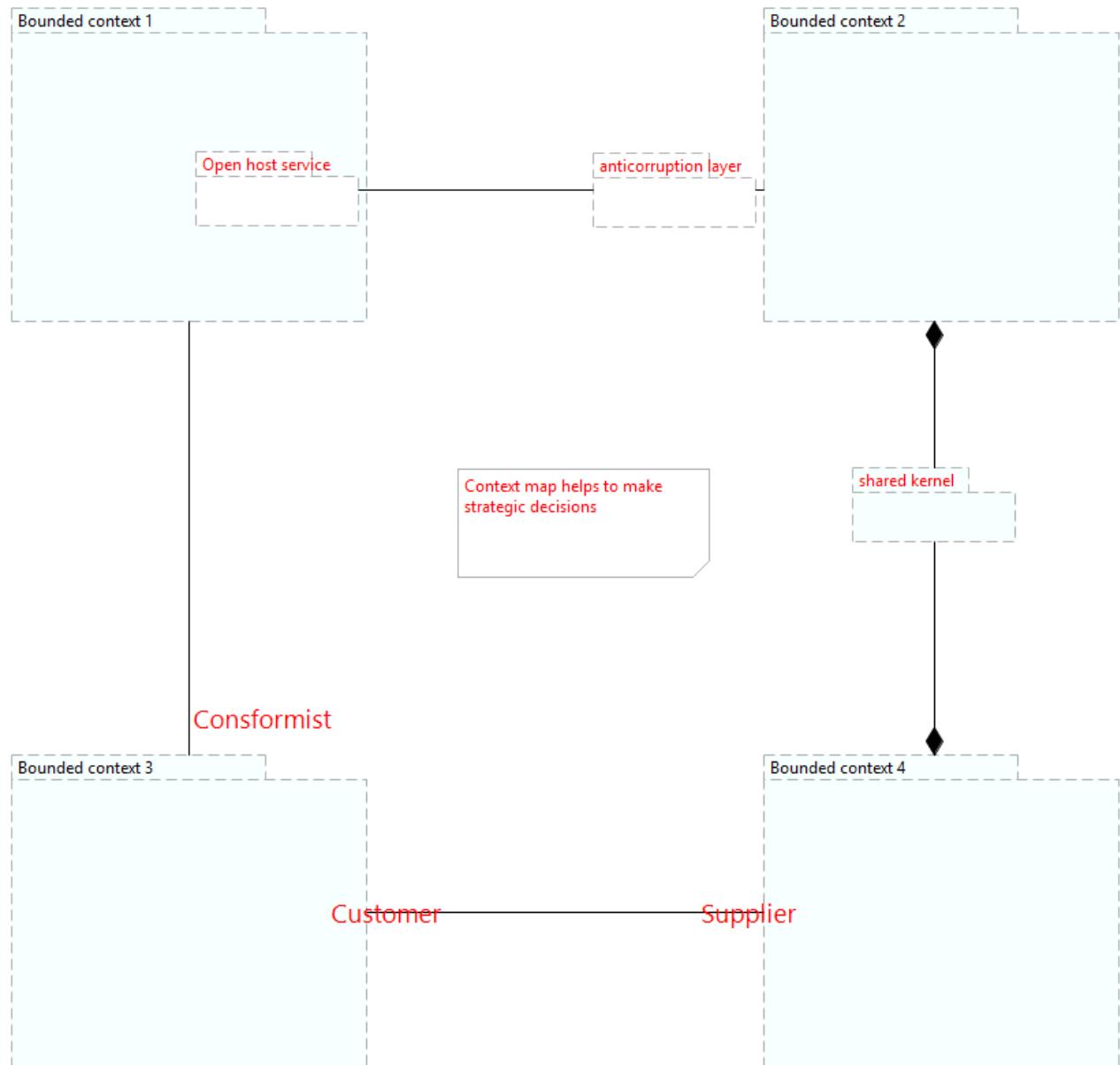
BOUNDED CONTEXT



CONTINUOUS INTEGRATION

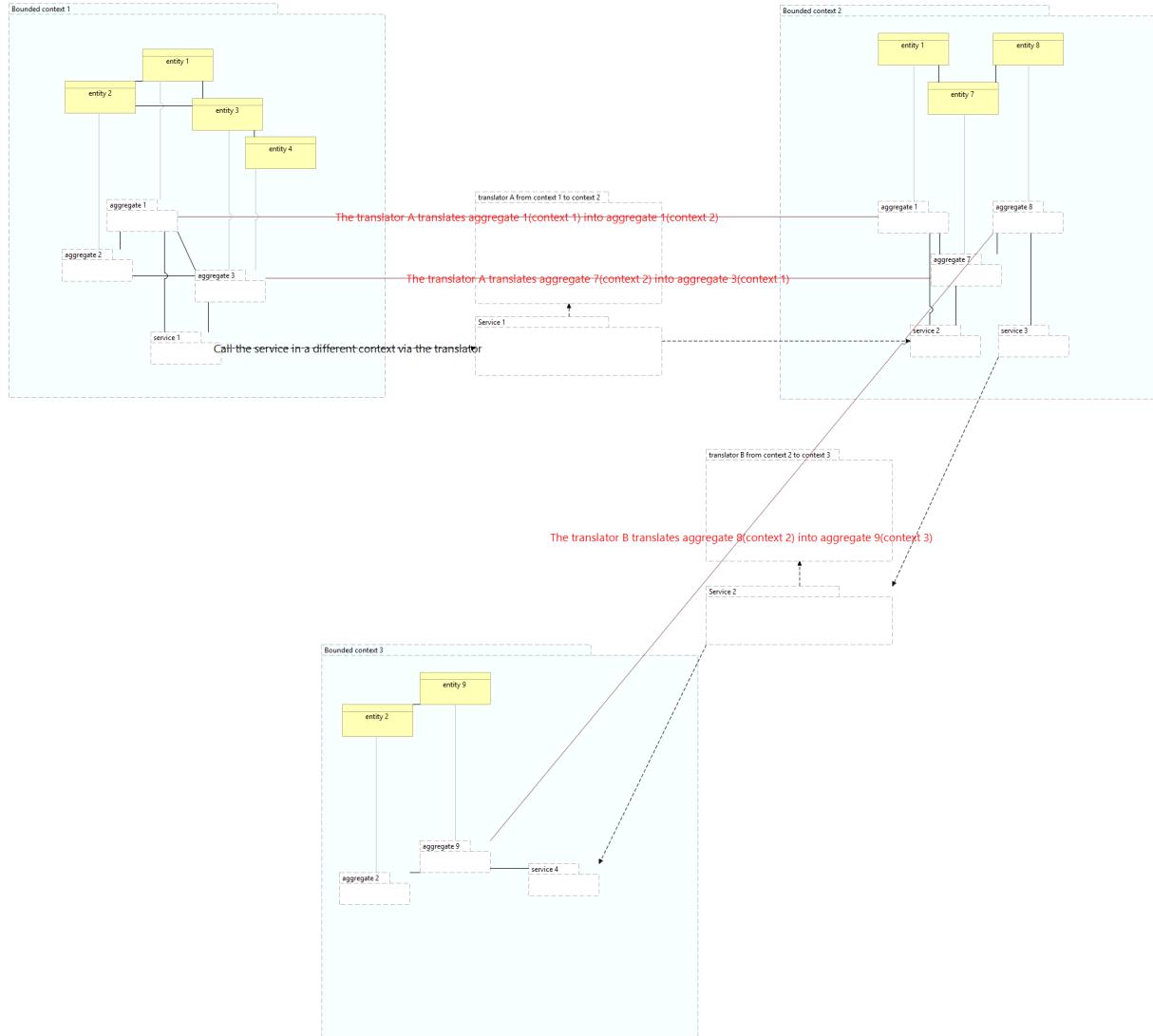


STRATEGIC CONTEXT MAP

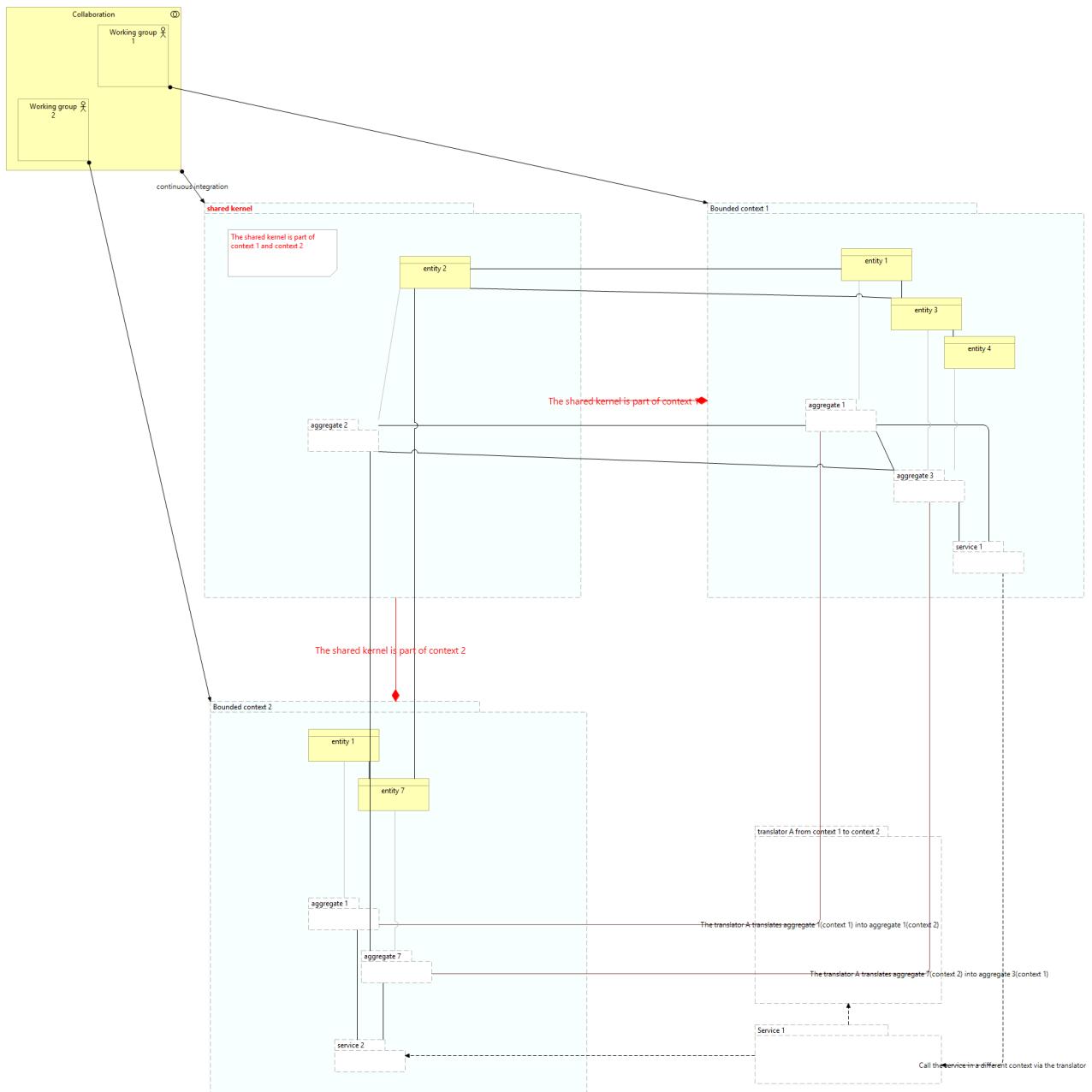


CONTEXTUAL MAP

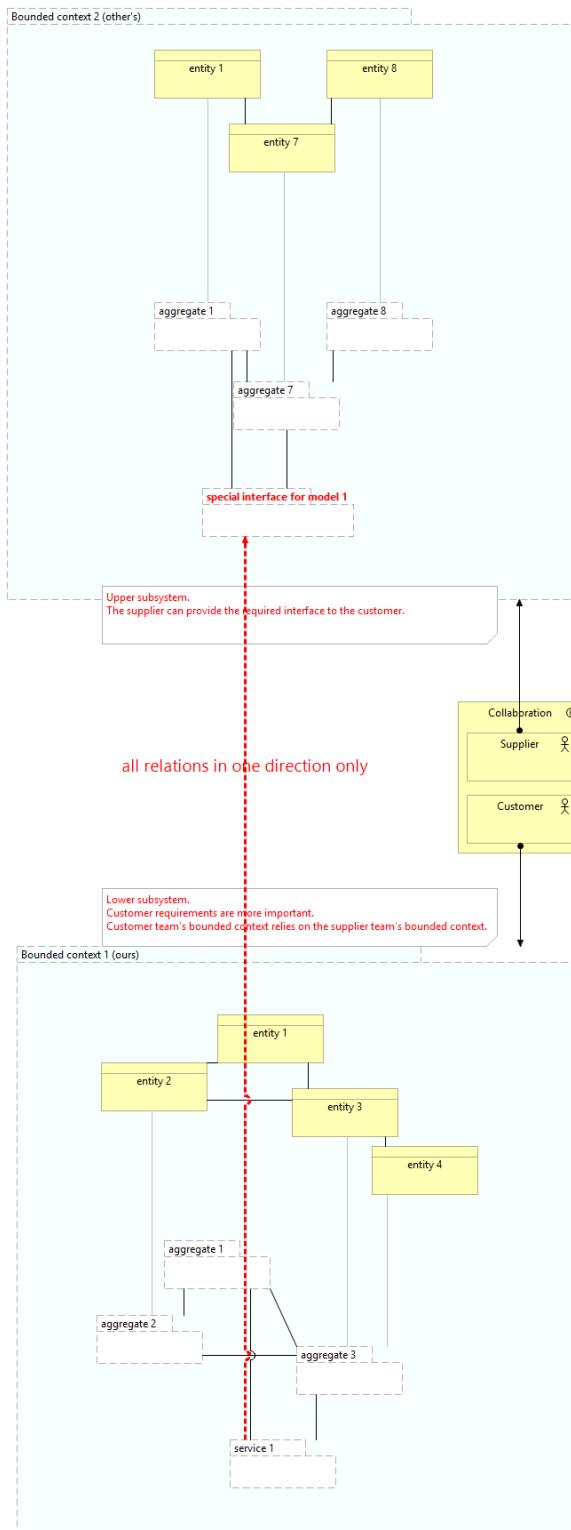
A single map of all contexts.
Integration of all bounded contexts.



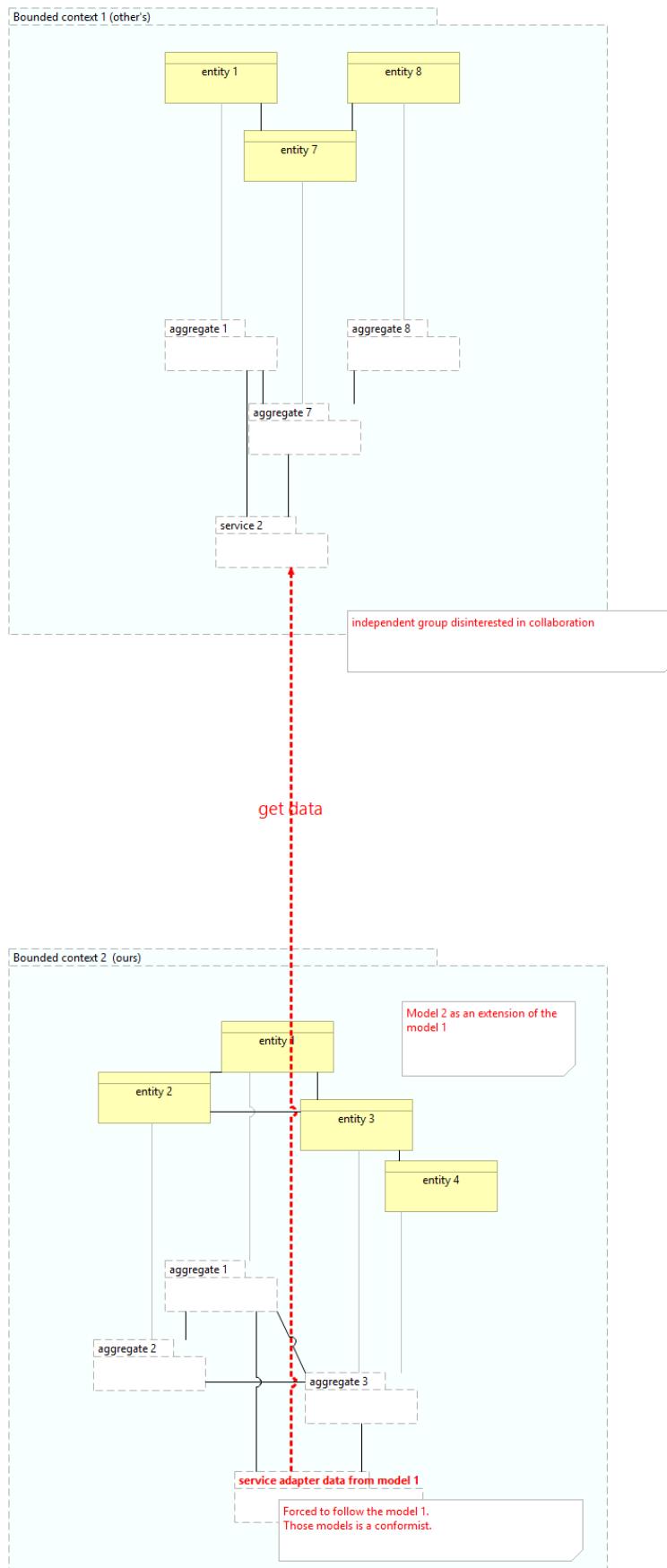
SHARED KERNEL



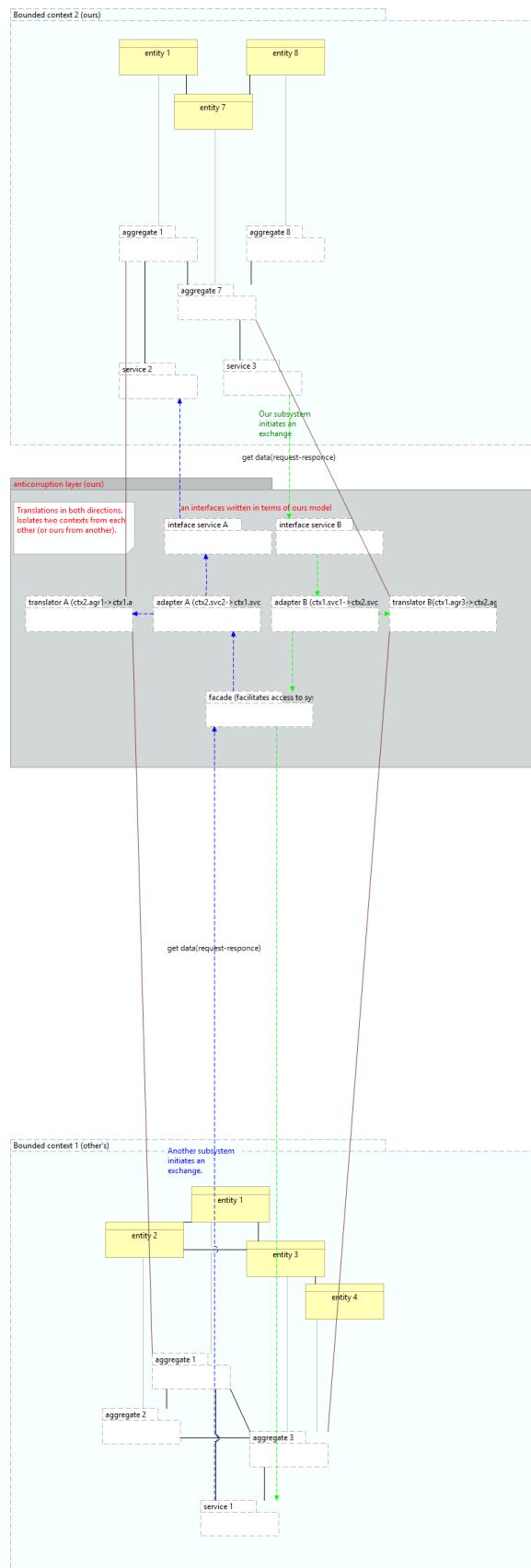
CUSTOMER-SUPPLIER TEAMS



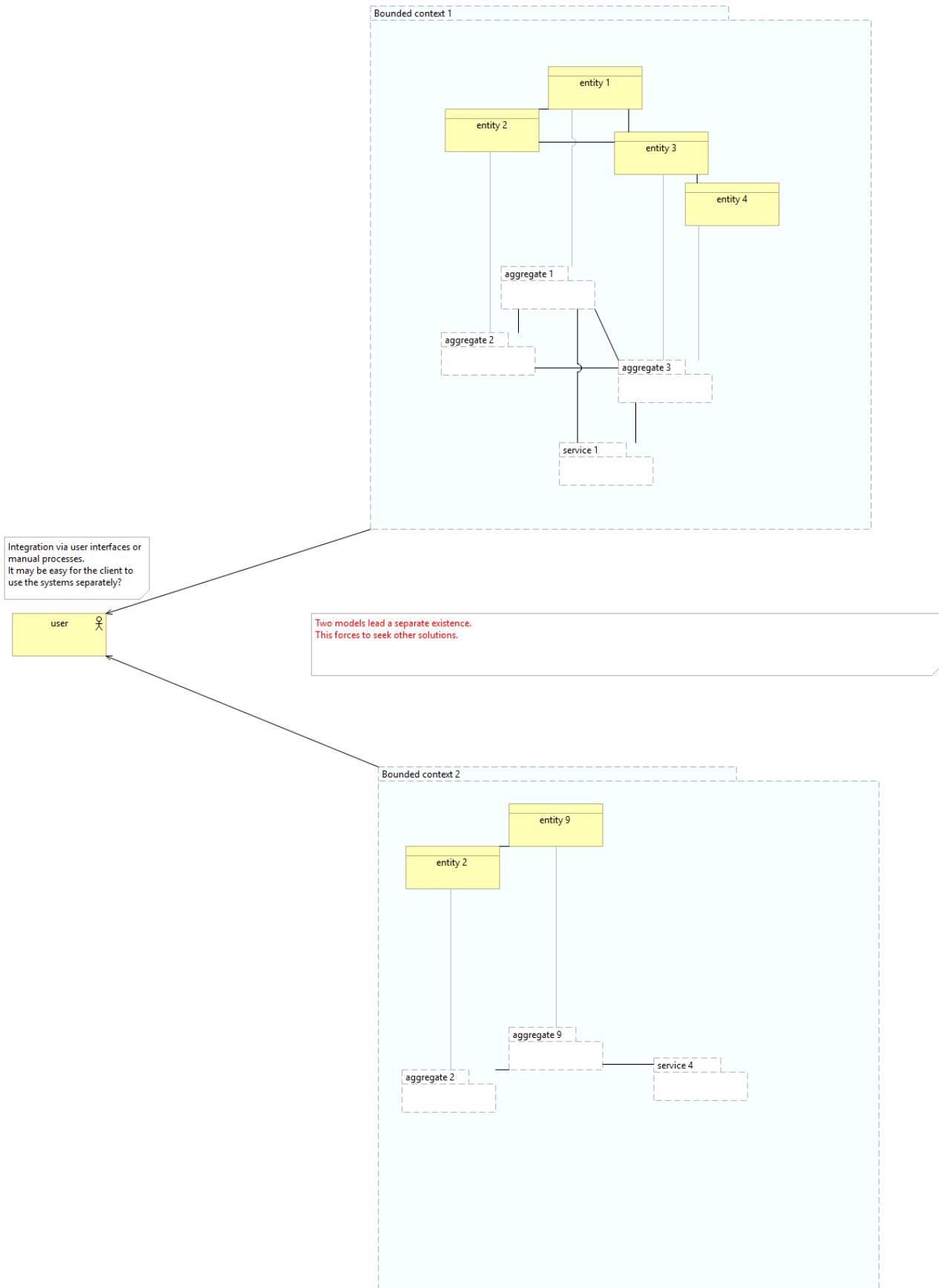
CONFORMIST



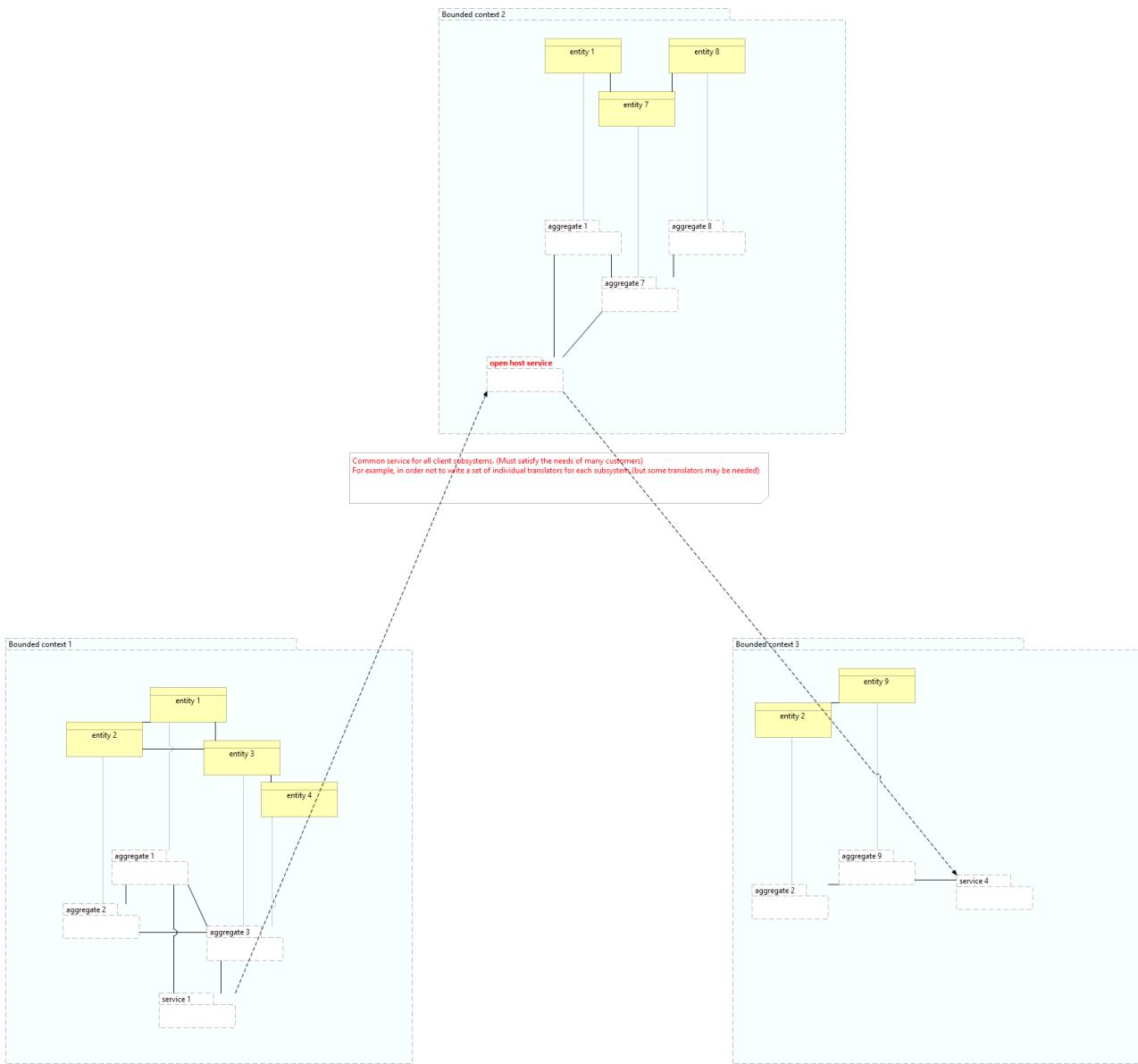
ANTICORRUPTION LAYER



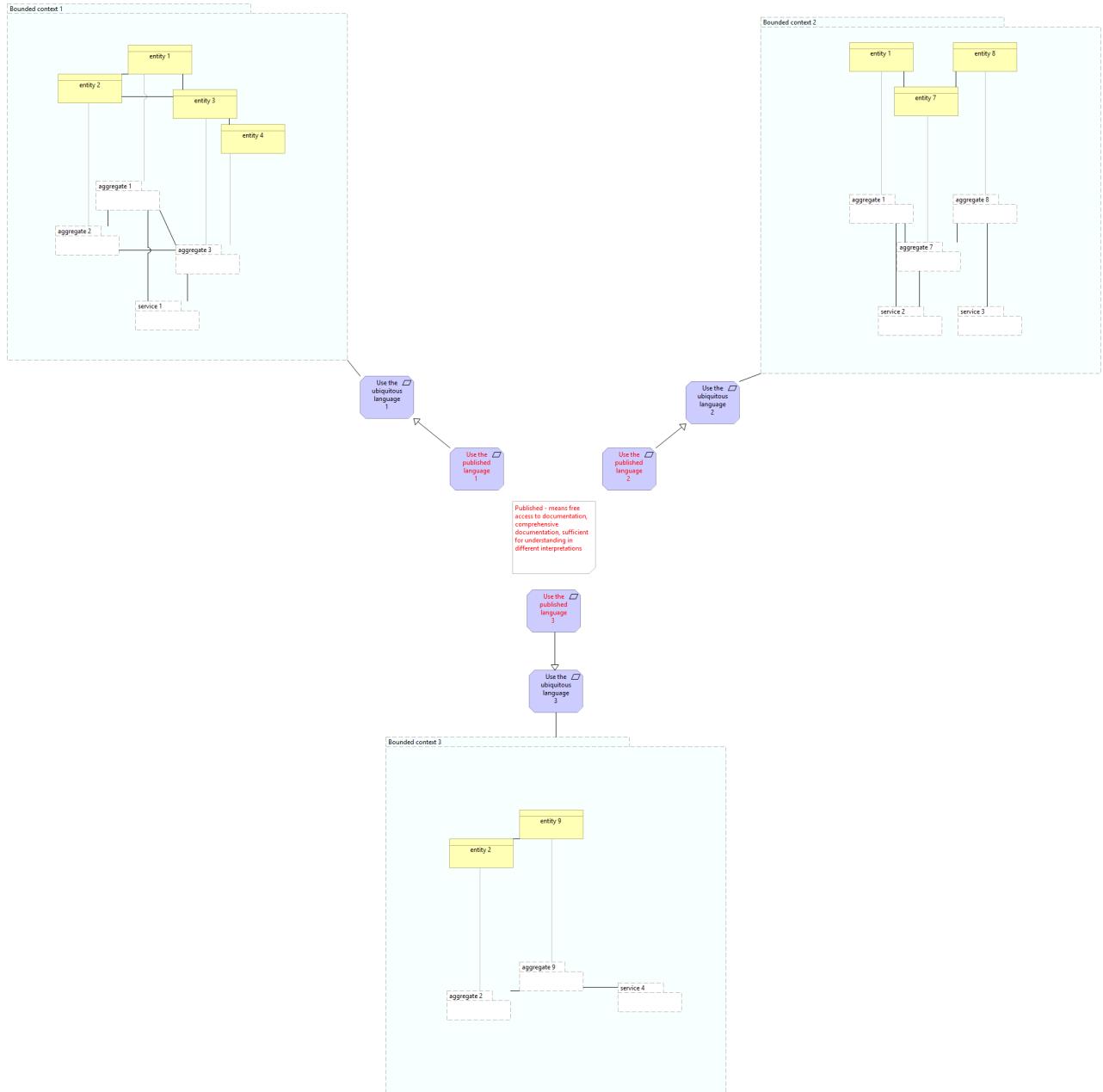
SEPARATE WAYS



OPEN HOST SERVICE



PUBLISHED LANGUAGE



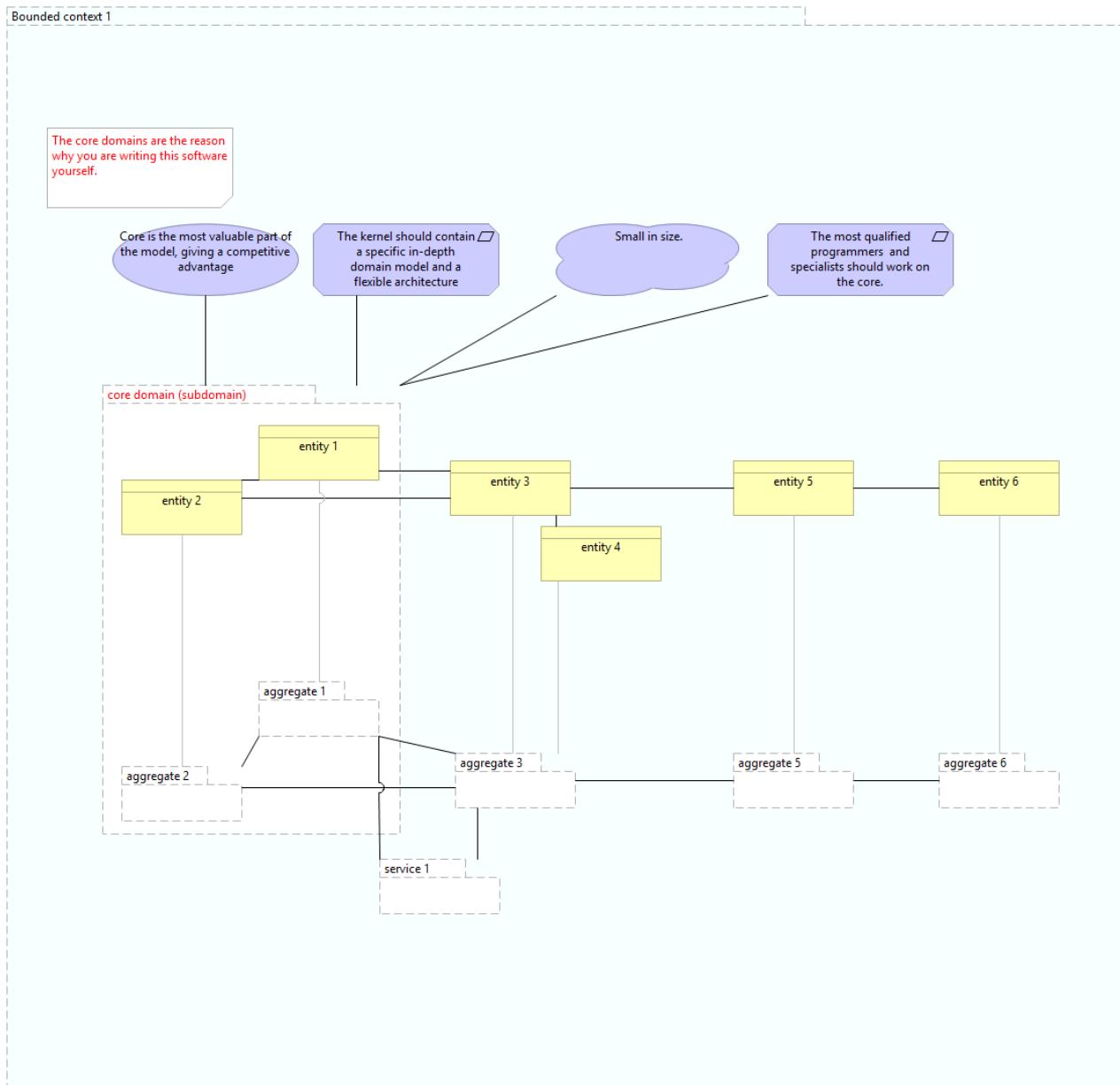
DISTILLATION

DOMAIN DRIVEN DESIGN

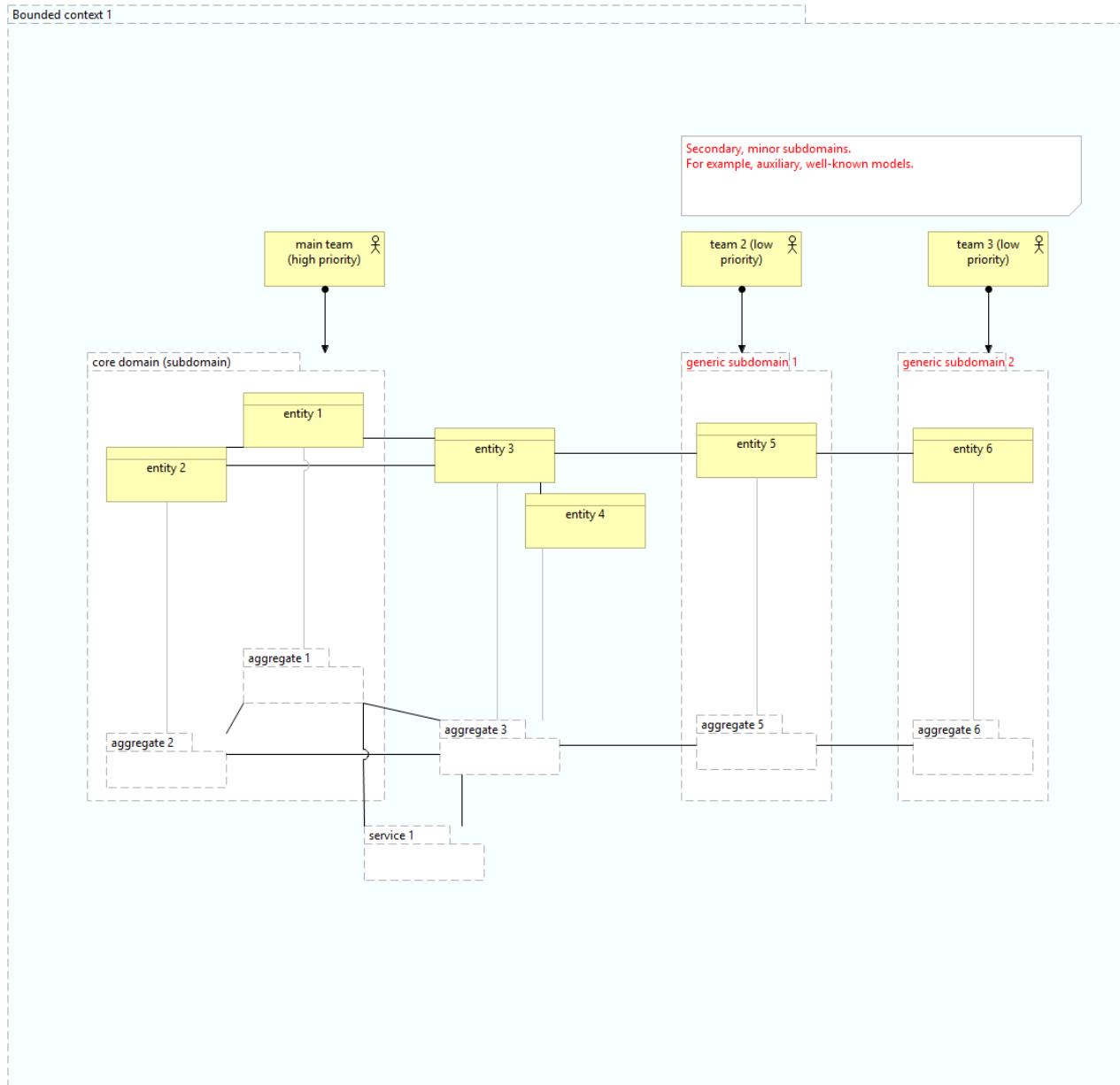
Eric Evans



CORE DOMAIN



GENERIC SUBDOMAINS



DOMAIN VISION STATEMENT

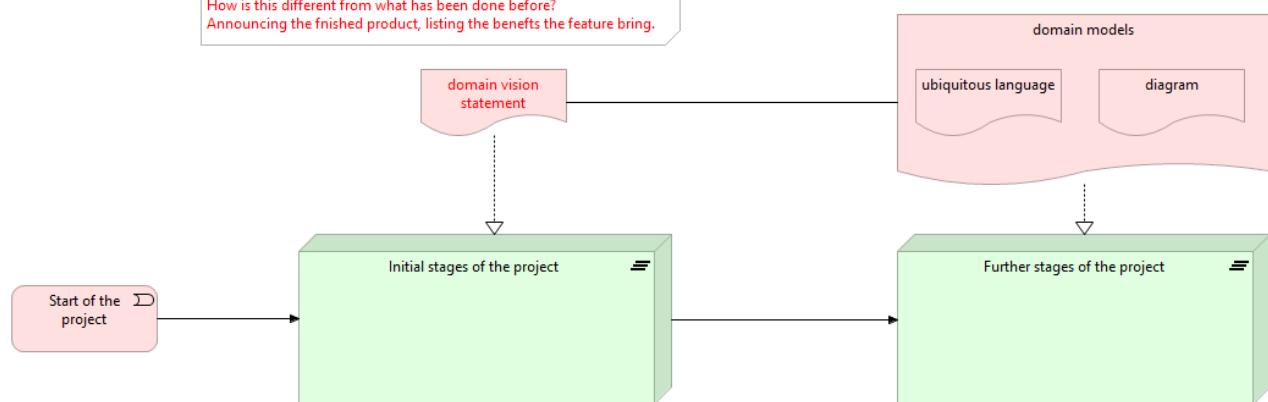
This document is a guideline that sets the direction before the development of the model.

Defines the core domain in the most general terms.

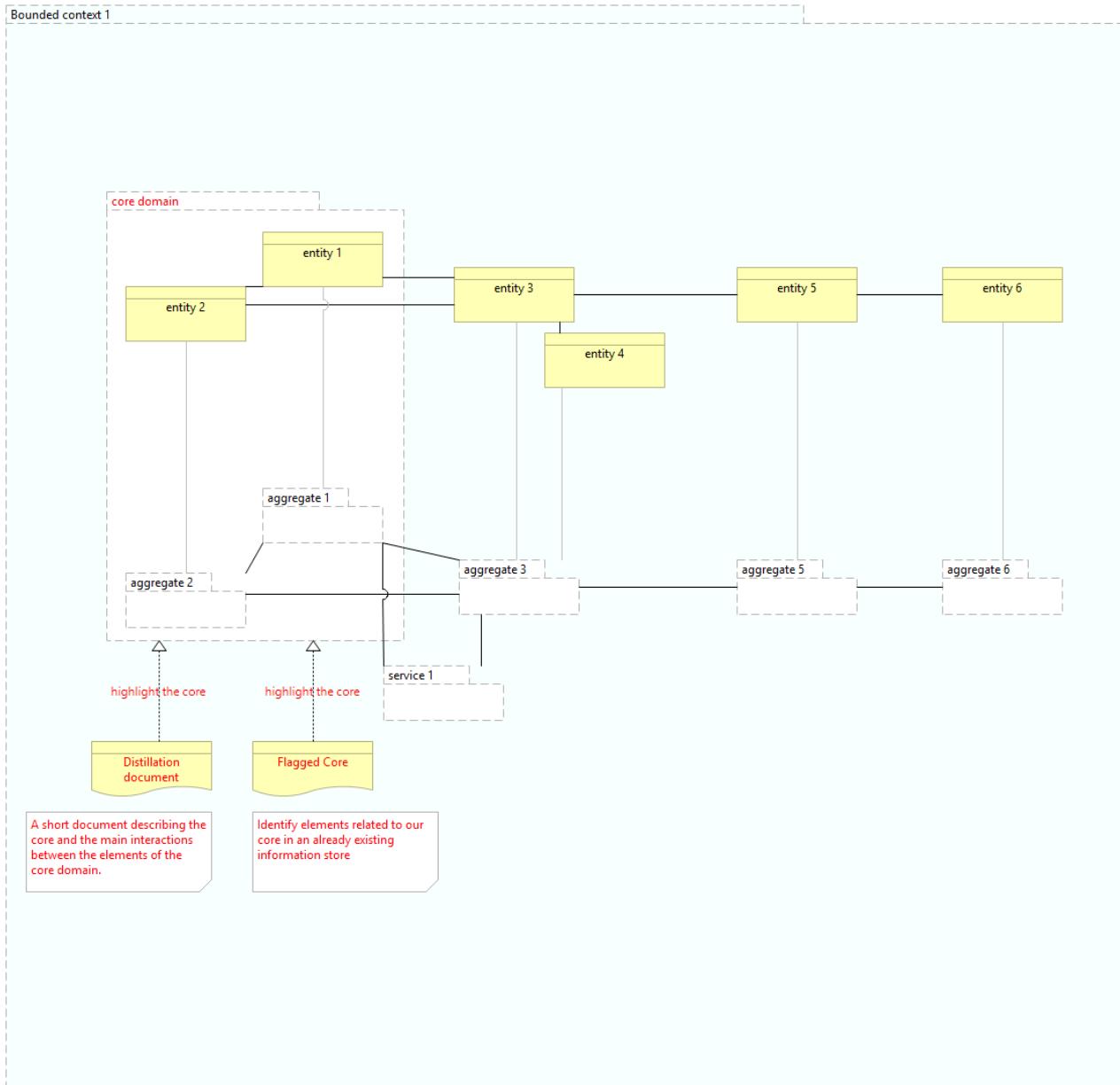
Draw out the reason you are opting to build rather than buy a product.

How is this different from what has been done before?

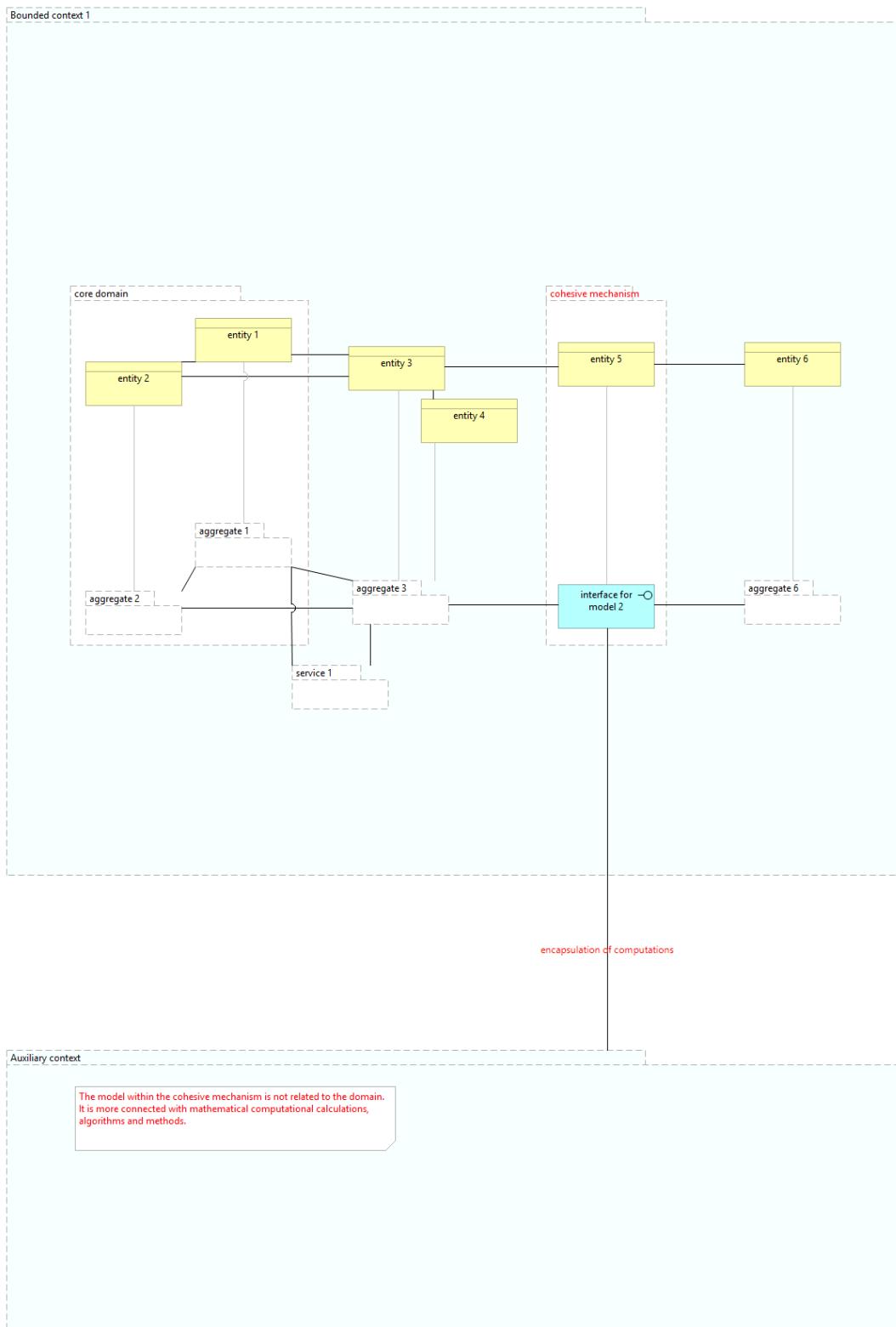
Announcing the finished product, listing the benefits the feature bring.



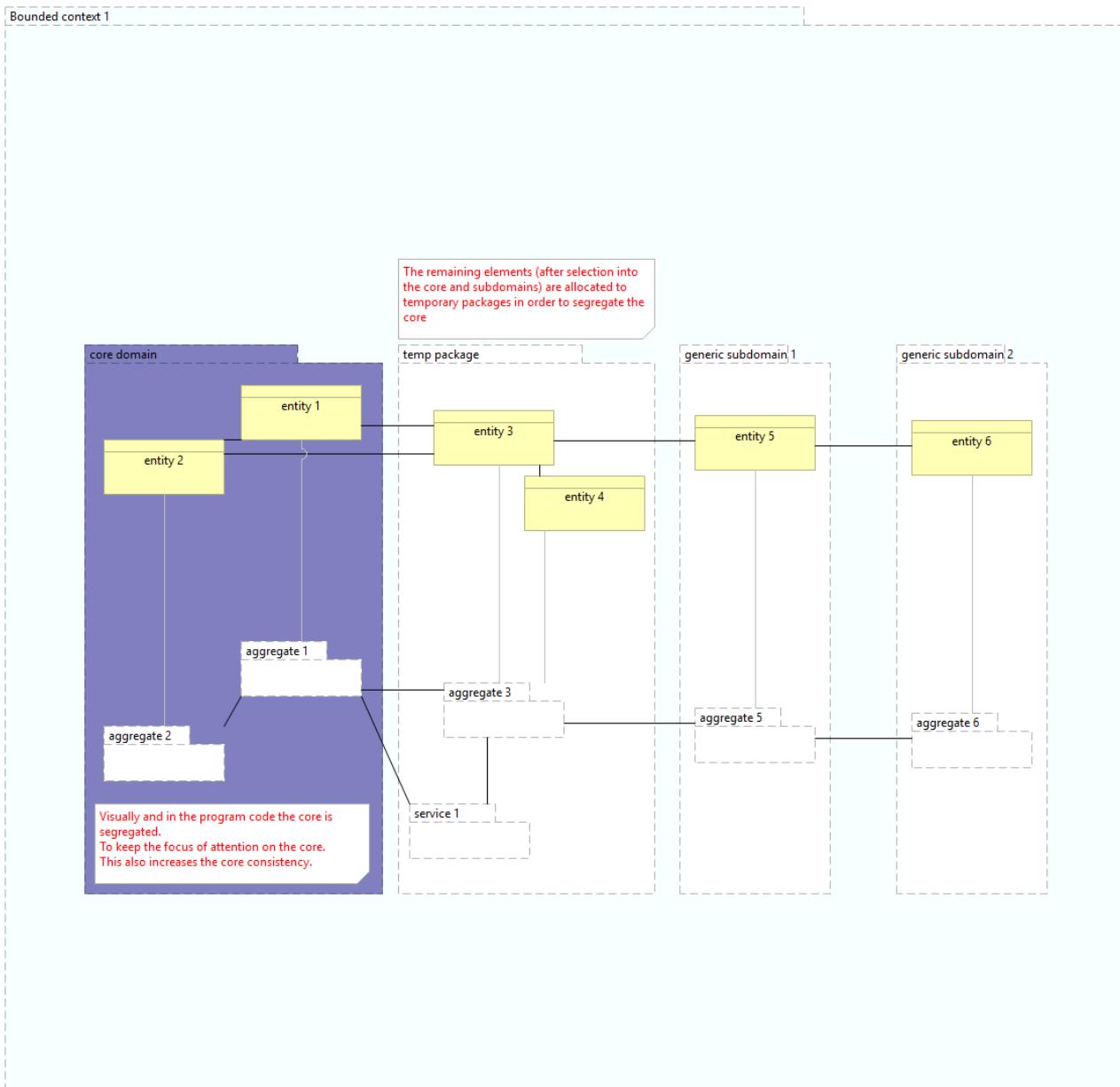
HIGHLIGHTED CORE



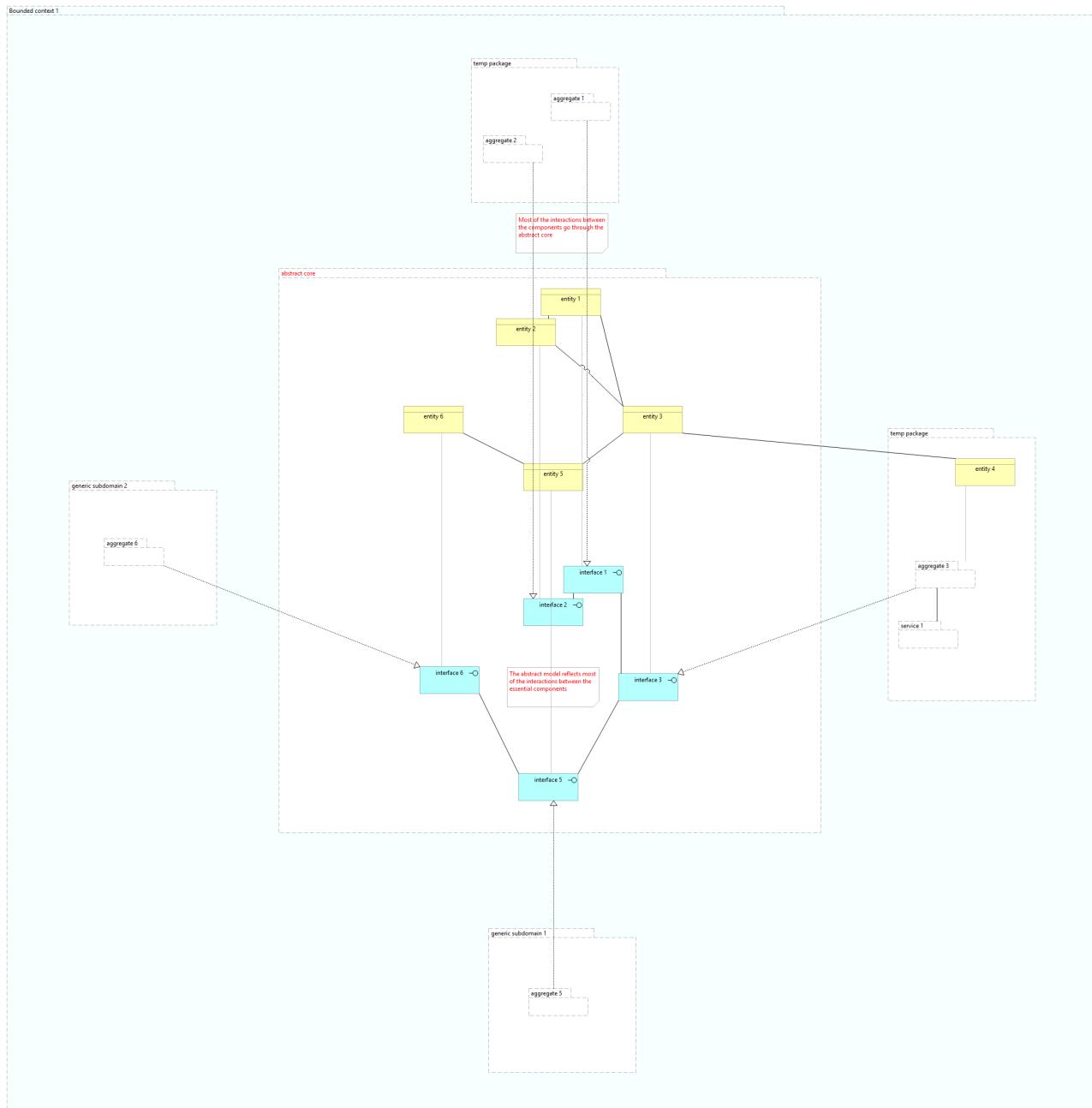
COHESIVE MECHANISMS



SEGREGATED CORE



ABSTRACT CORE



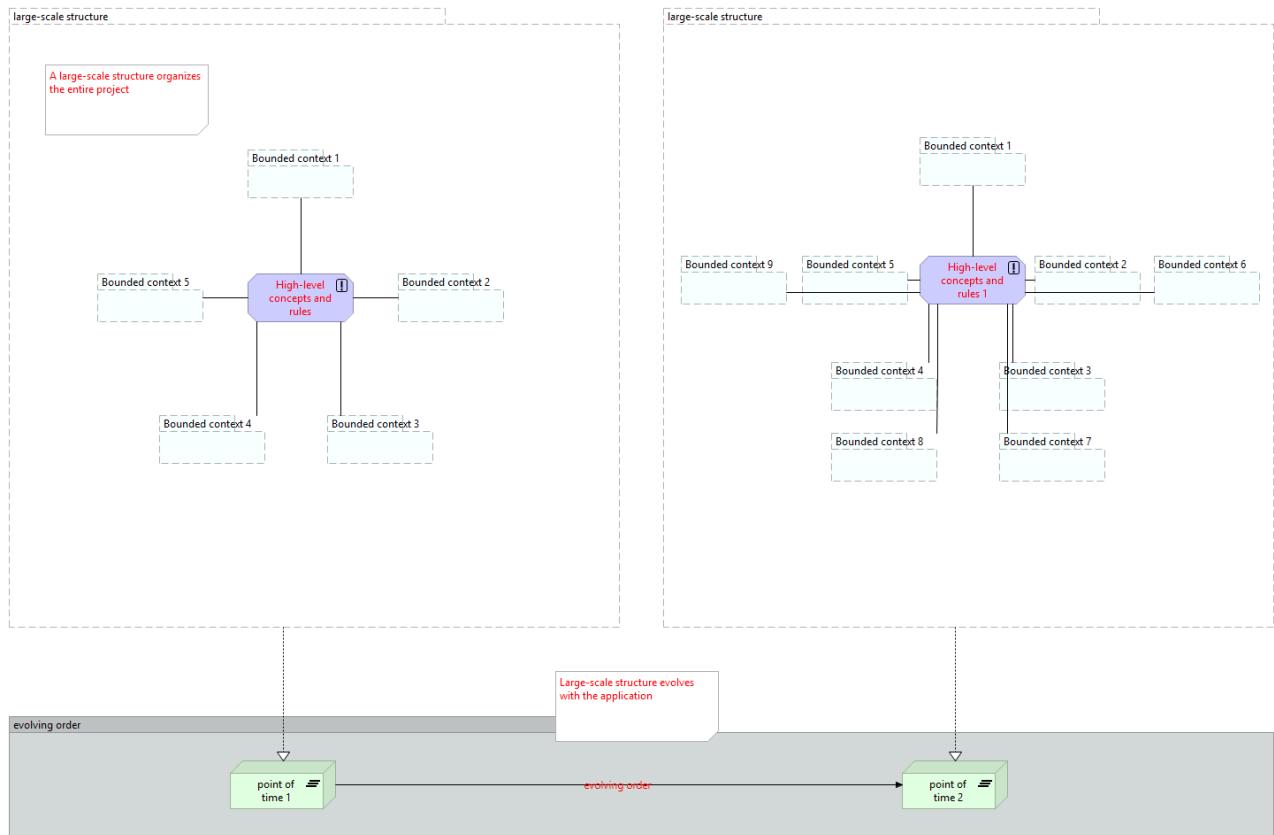
LARGE-SCALE STRUCTURE

DOMAIN DRIVEN DESIGN

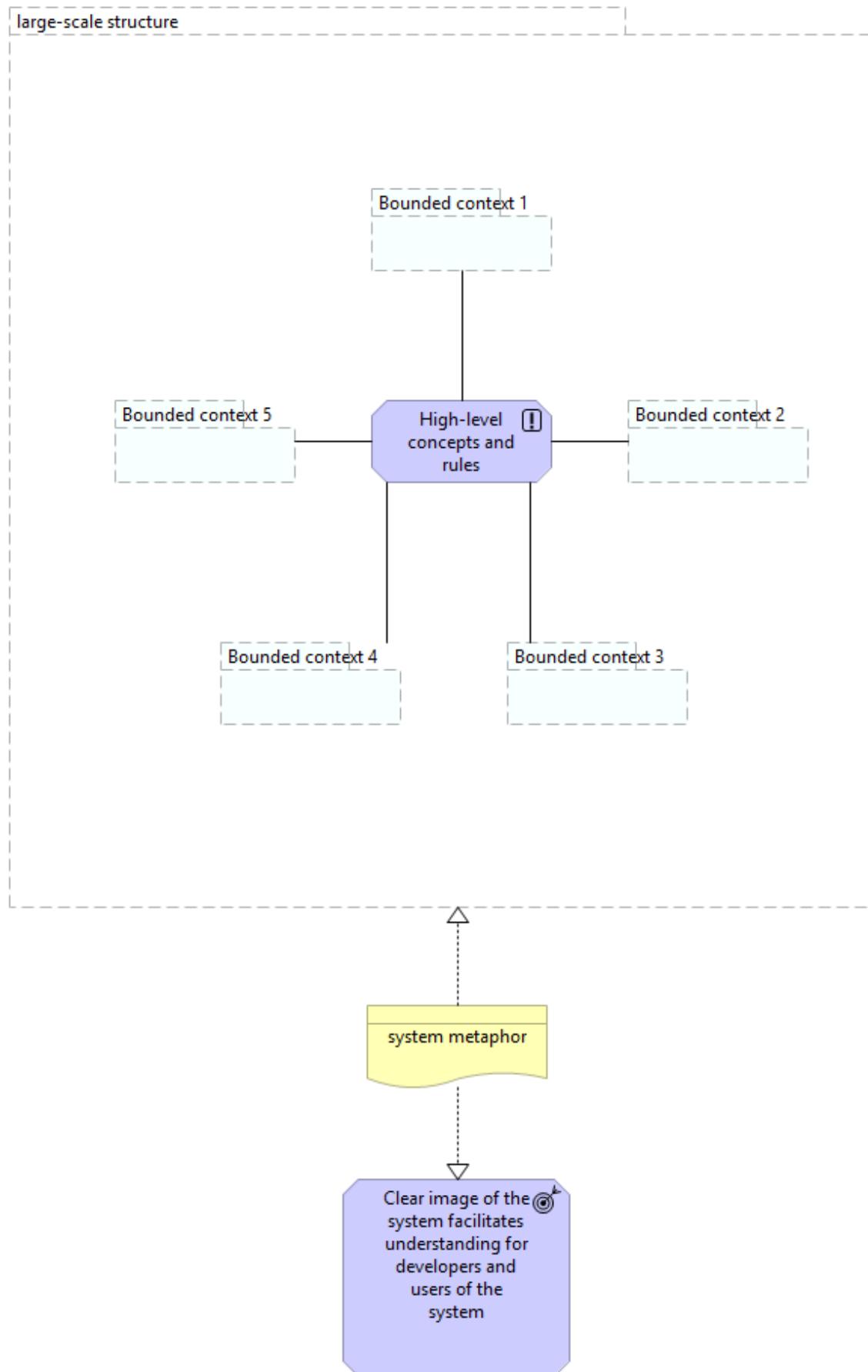
Eric Evans



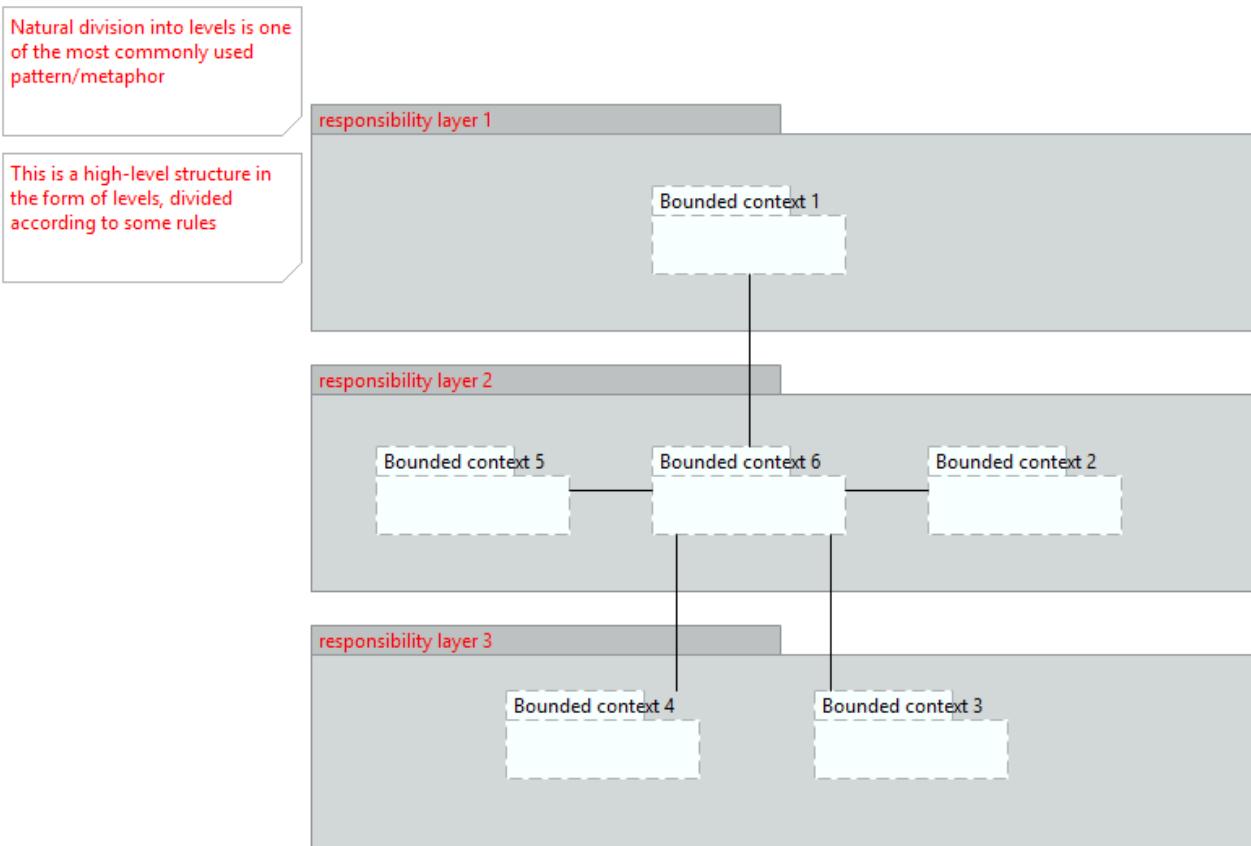
EVOLVING ORDER



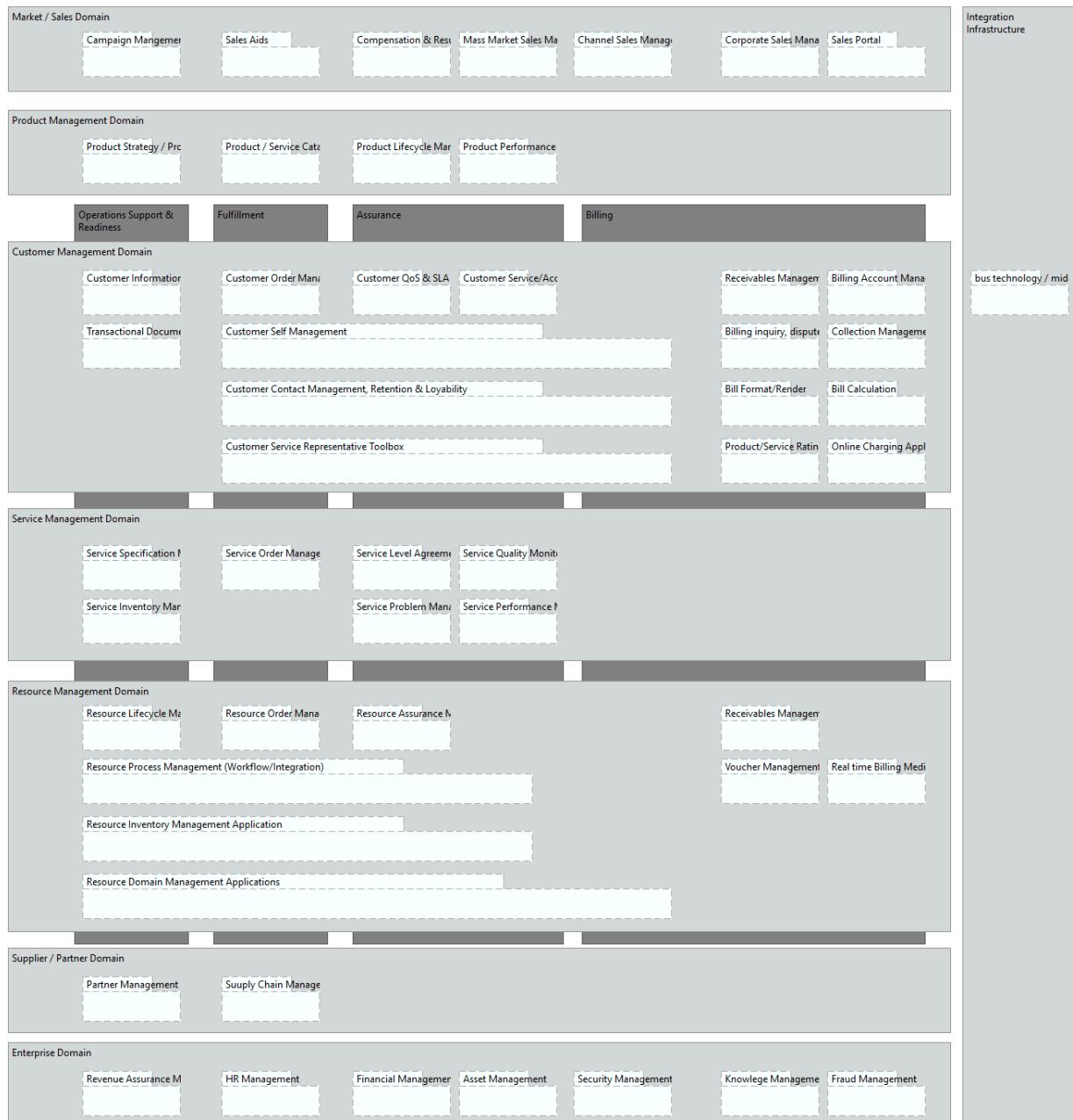
SYSTEM METAPHOR



RESPONSIBILITY LAYERS



Layers based on NGOSS
Framework Telecom Application
Map



Layers based on Porter's Value Chain

Firm infrastructure

Bounded context 1

Human Resources Management

Bounded context 5

Bounded context 6

Bounded context 2

Technological Development

Bounded context 4

Bounded context 3

Procurement

Bounded context 5

Bounded context 6

Inbound Logistics

Bounded context 7

Operations

Bounded context 8

Outbound Logistics

Bounded context 9

Marketing & Sales

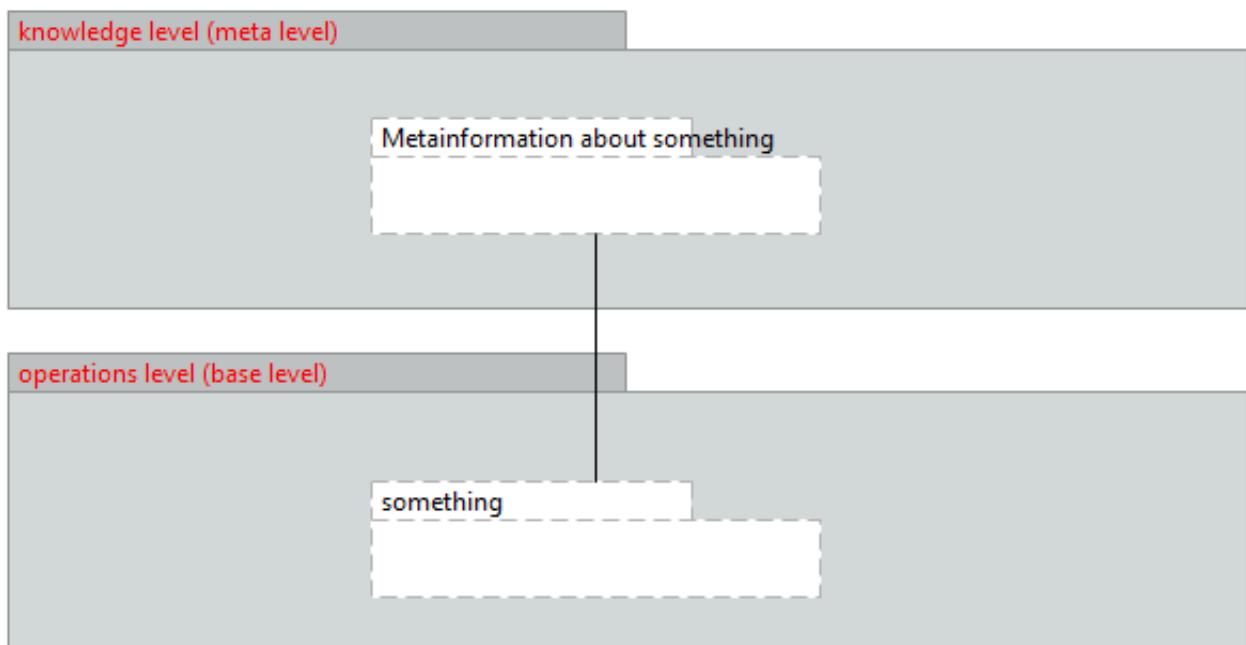
Bounded context 10

Service

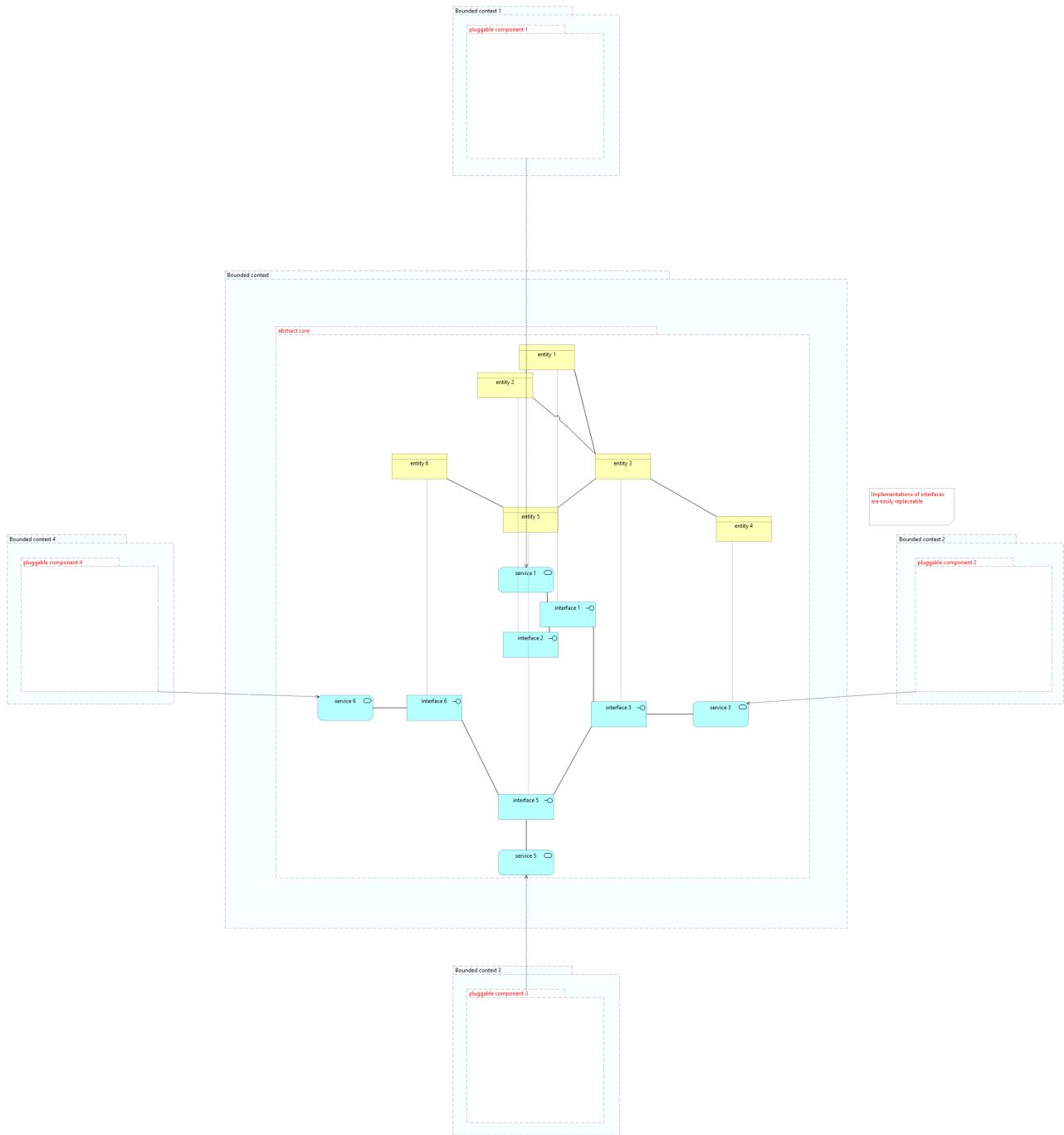
Bounded context 11



KNOWLEDGE LEVEL



PLUGGABLE COMPONENT FRAMEWORK



ADDITIONAL PATTERNS

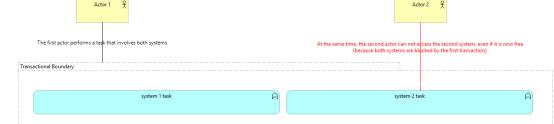
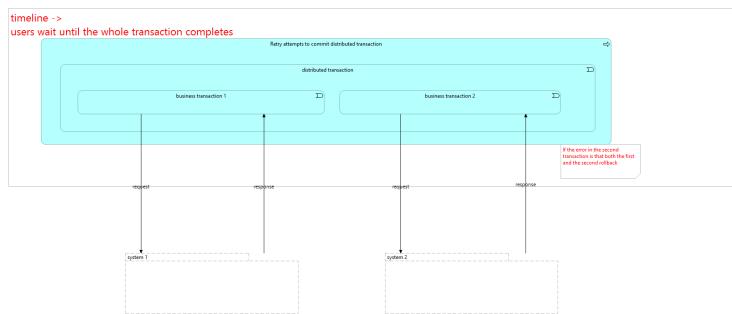
DOMAIN DRIVEN DESIGN

Eric Evans

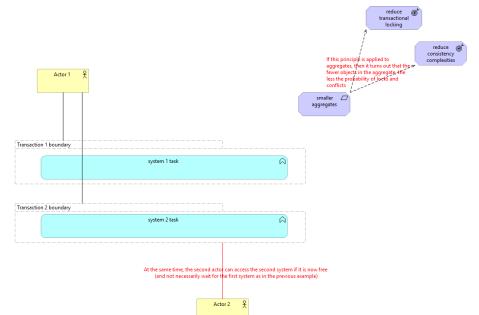
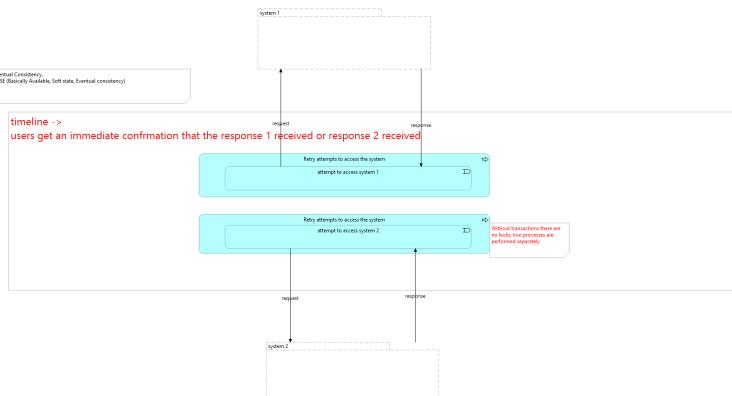


TYPES OF CONSISTENCY

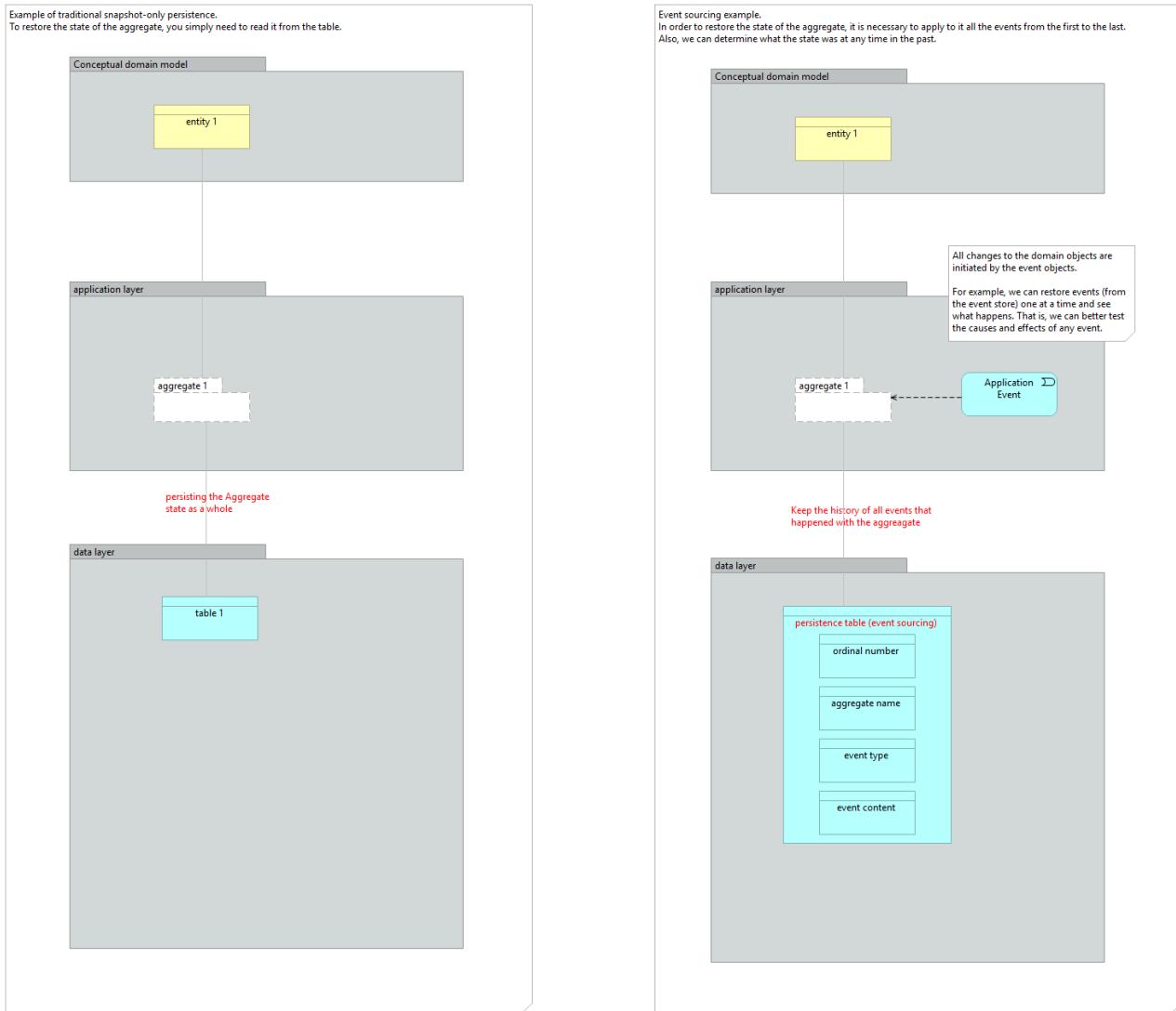
Transactional consistency:
Ensures that the system is in a consistent state.
ACID (Atomicity, Consistency, Isolation, Durability)
The more systems a distributed transaction, the greater the risk of potential locks and conflicts.



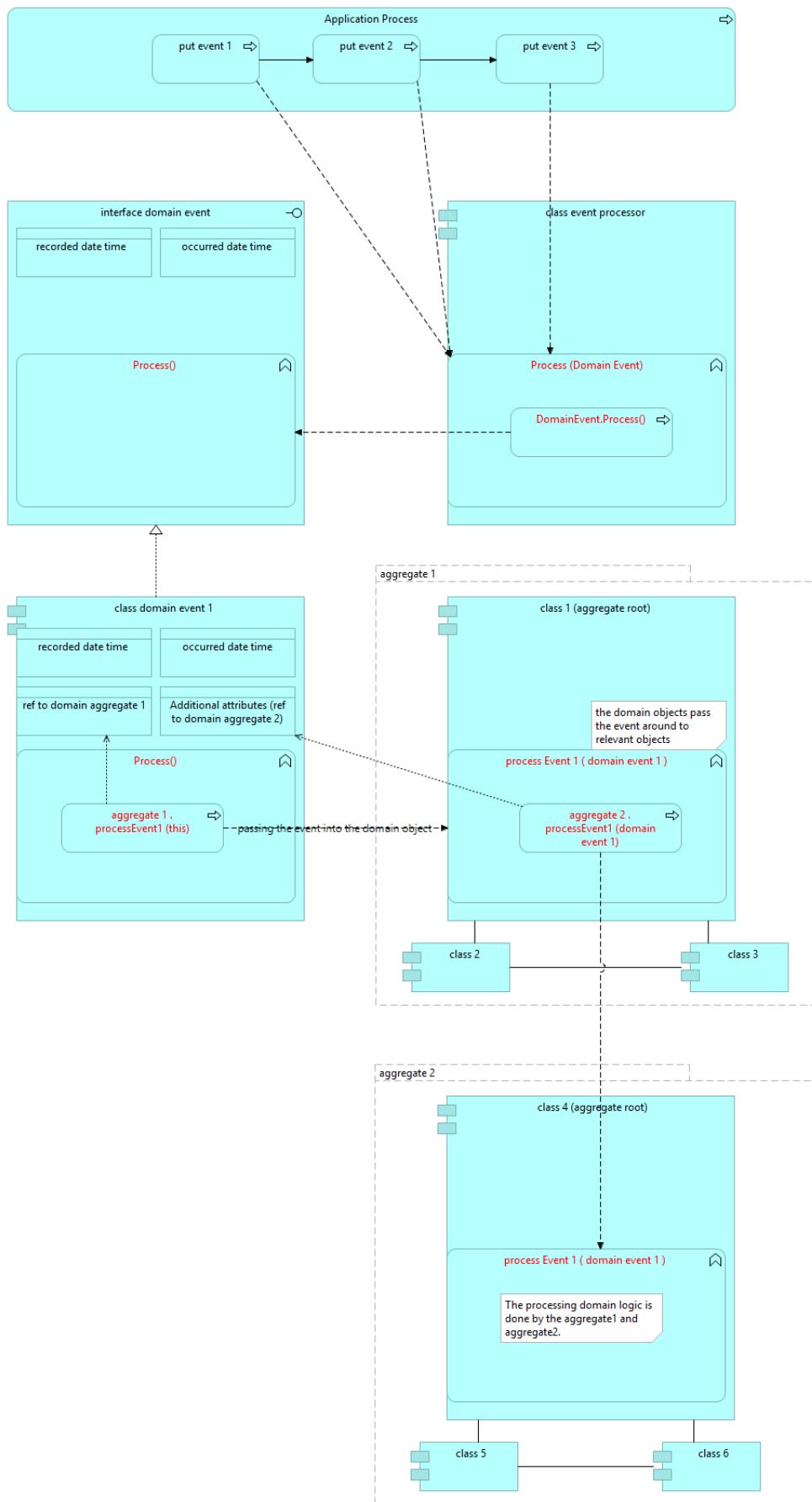
Eventual Consistency:
BASE (Basically Available, Soft state, Eventual consistency)



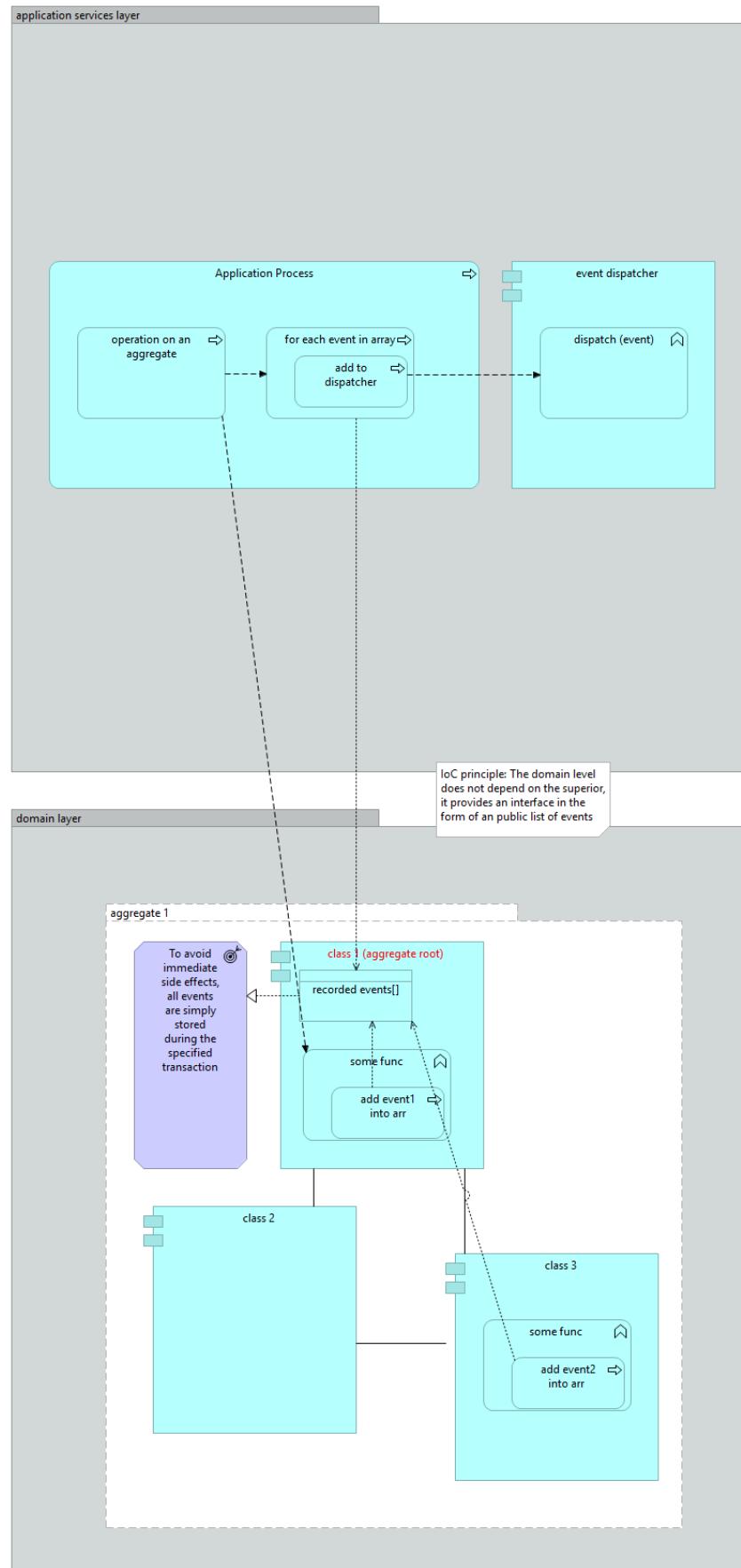
EVENT SOURCING



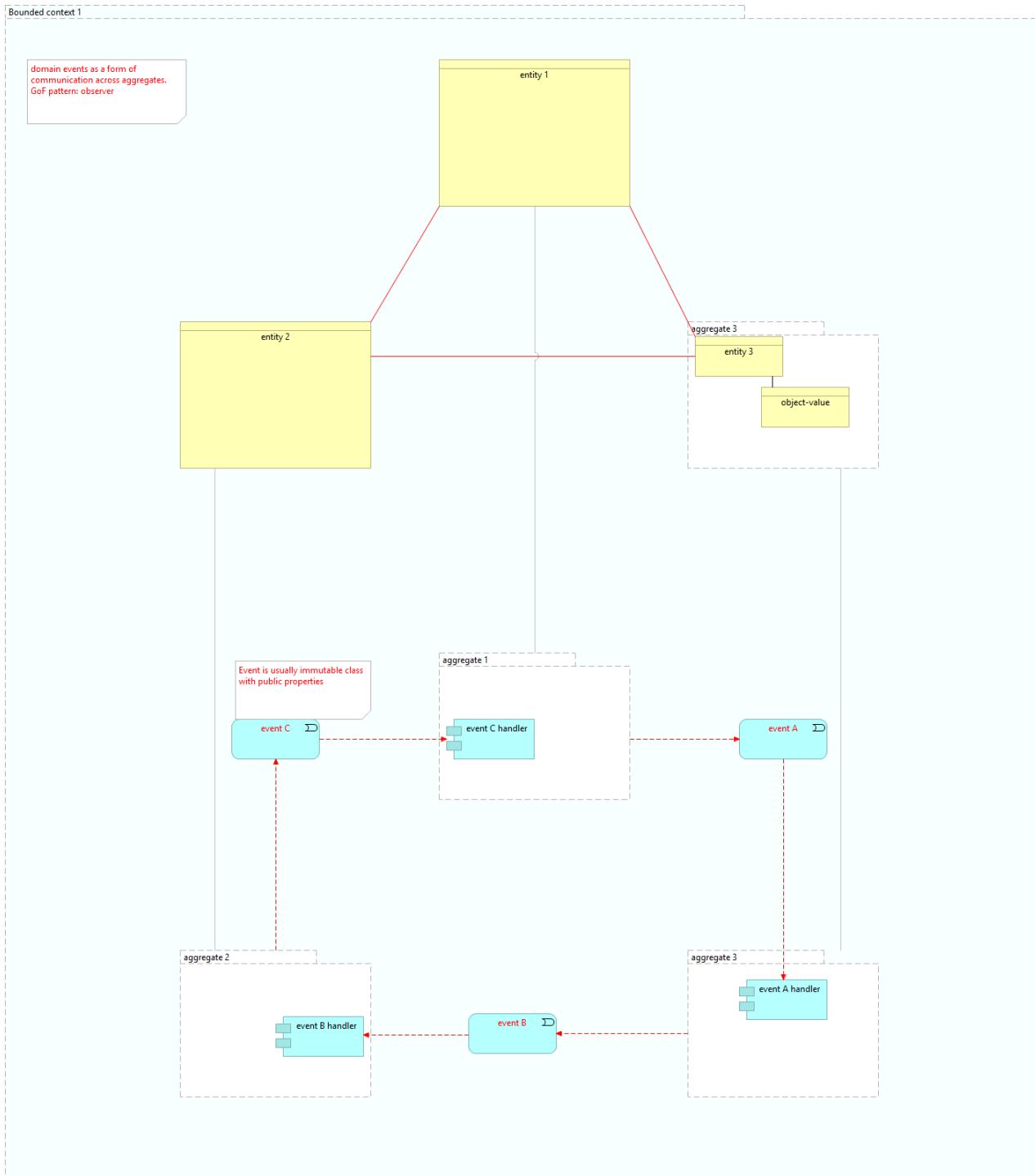
EVENT PROCESSOR



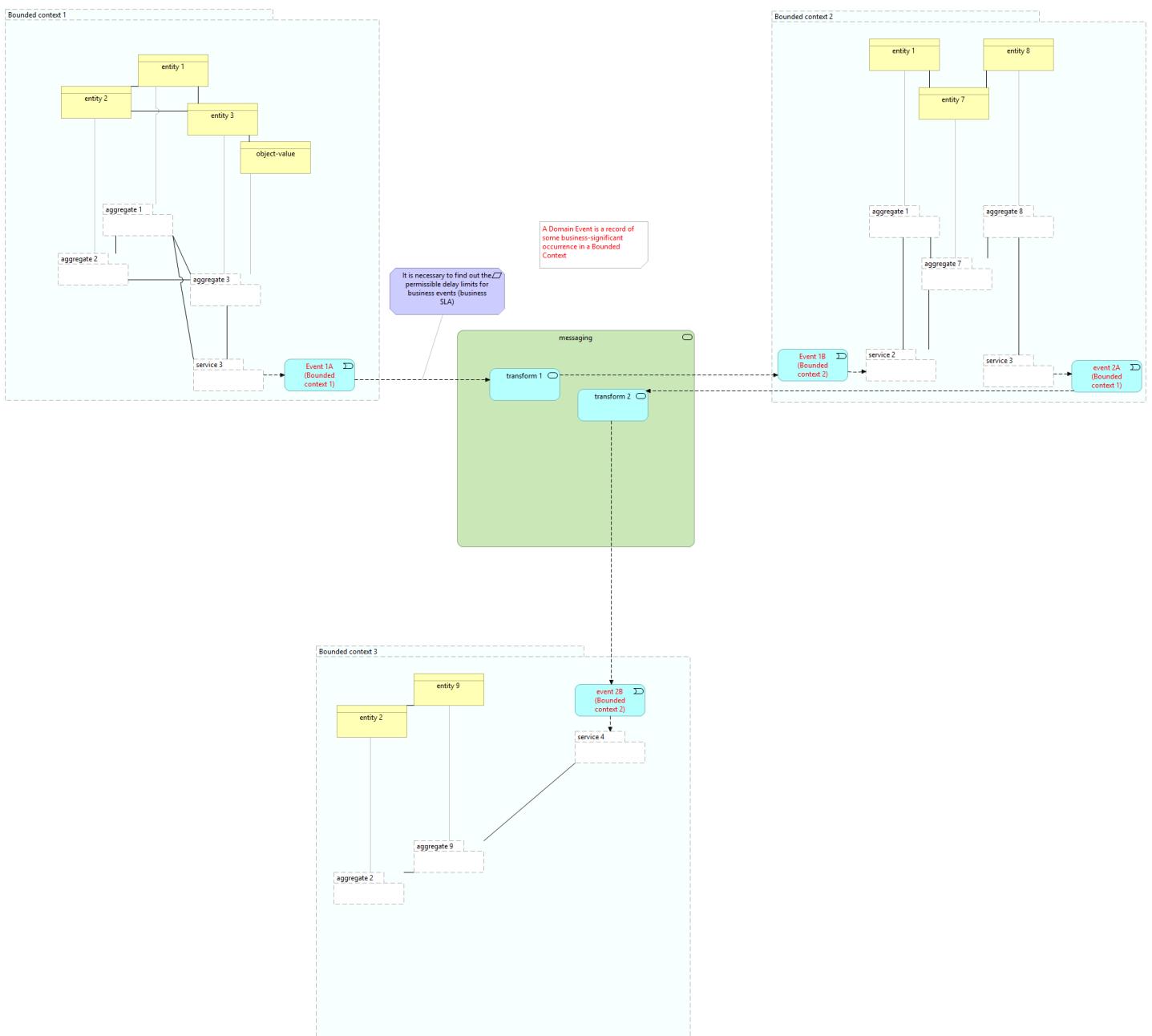
EVENT DISPATCHER



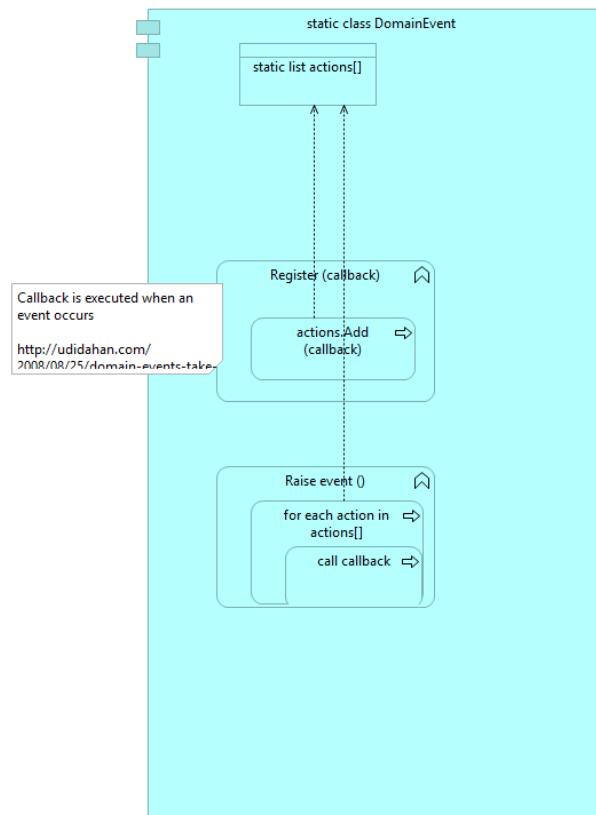
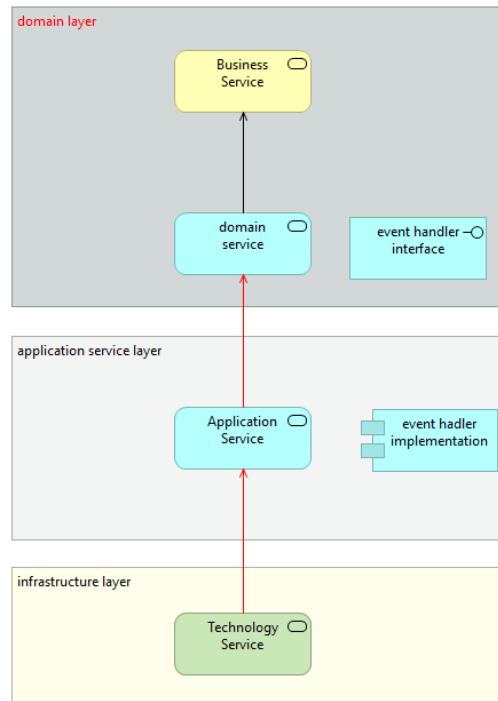
INTERNAL DOMAIN EVENTS



EXTERNAL DOMAIN EVENTS, TRANSFER BETWEEN CONTEXTS

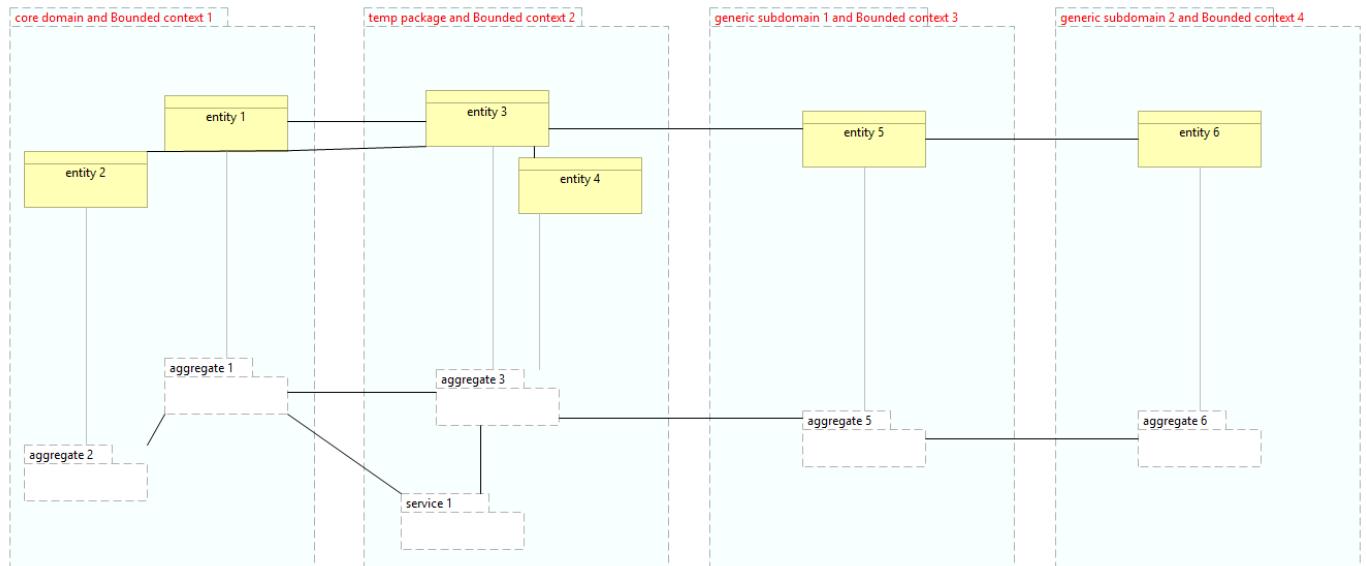


STATIC DOMAIN EVENTS CLASS

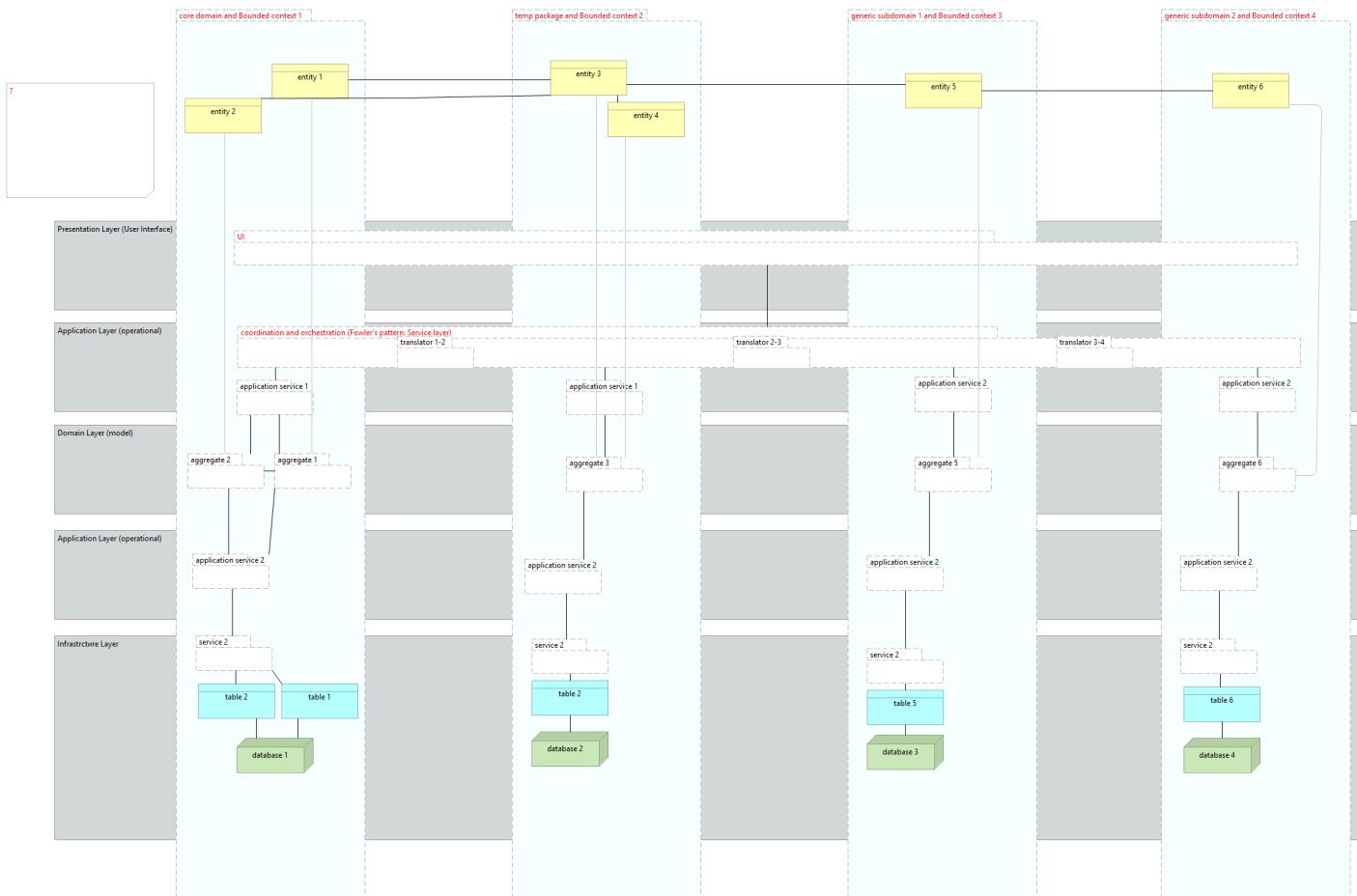


ONE SUBDOMAIN PER BOUNDED CONTEXT

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

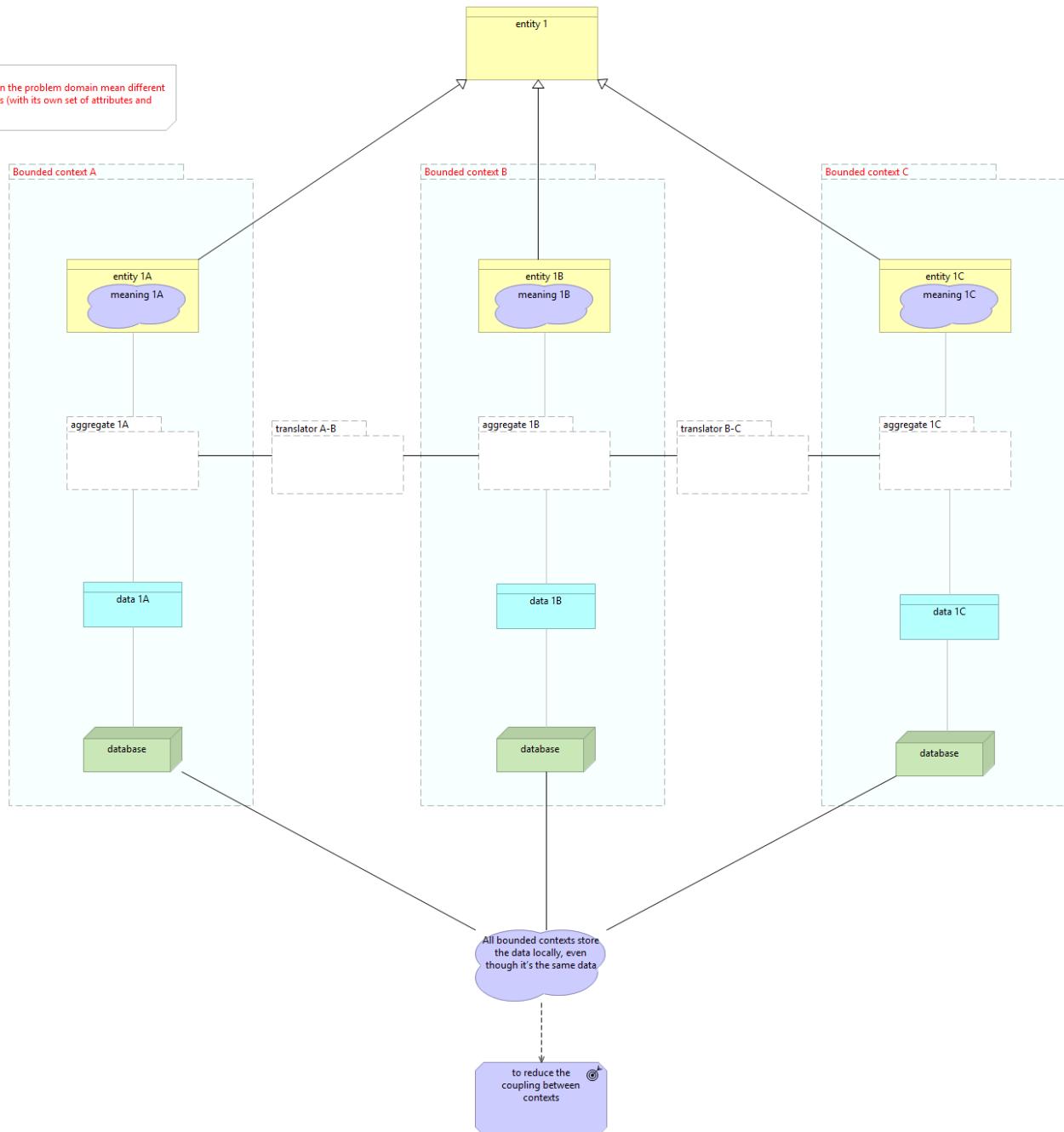


THE APPLICATION LAYER COORDINATES THE WORK BETWEEN CONTEXTS

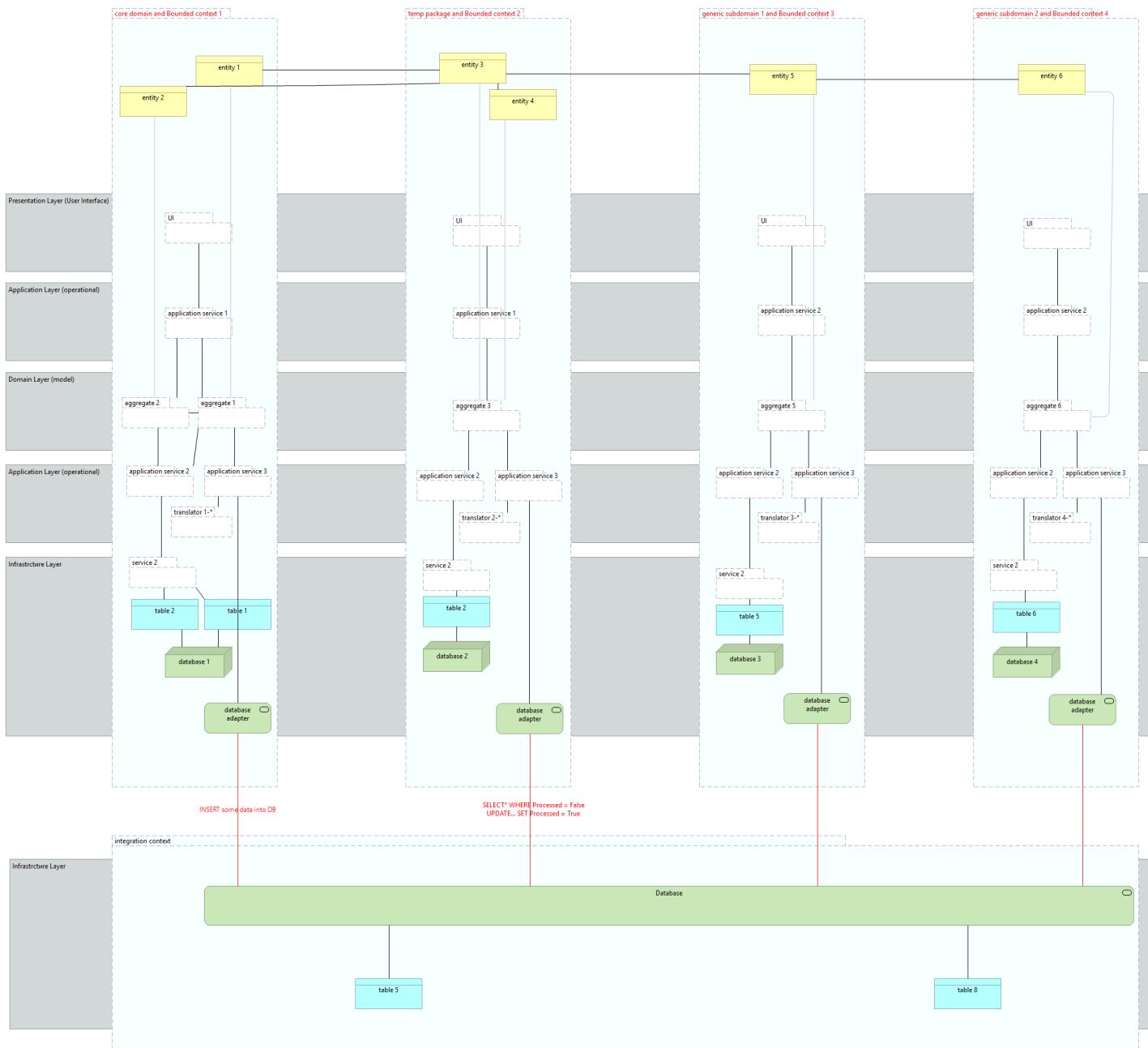


THE SAME PHYSICAL ENTITY IN DIFFERENT CONTEXTS

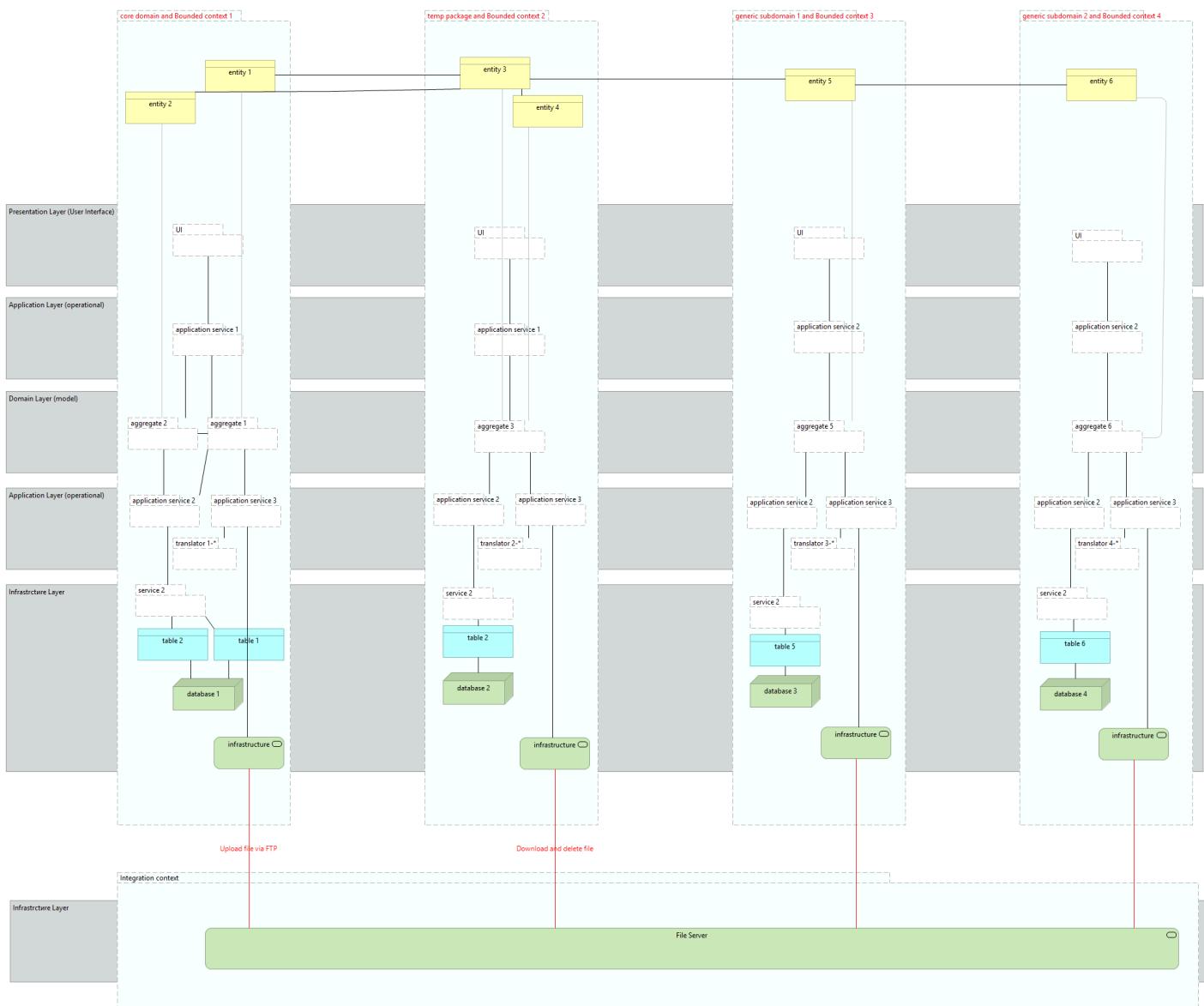
Example
The same physical entity in the problem domain mean different things in different contexts (with its own set of attributes and aspects of behavior).



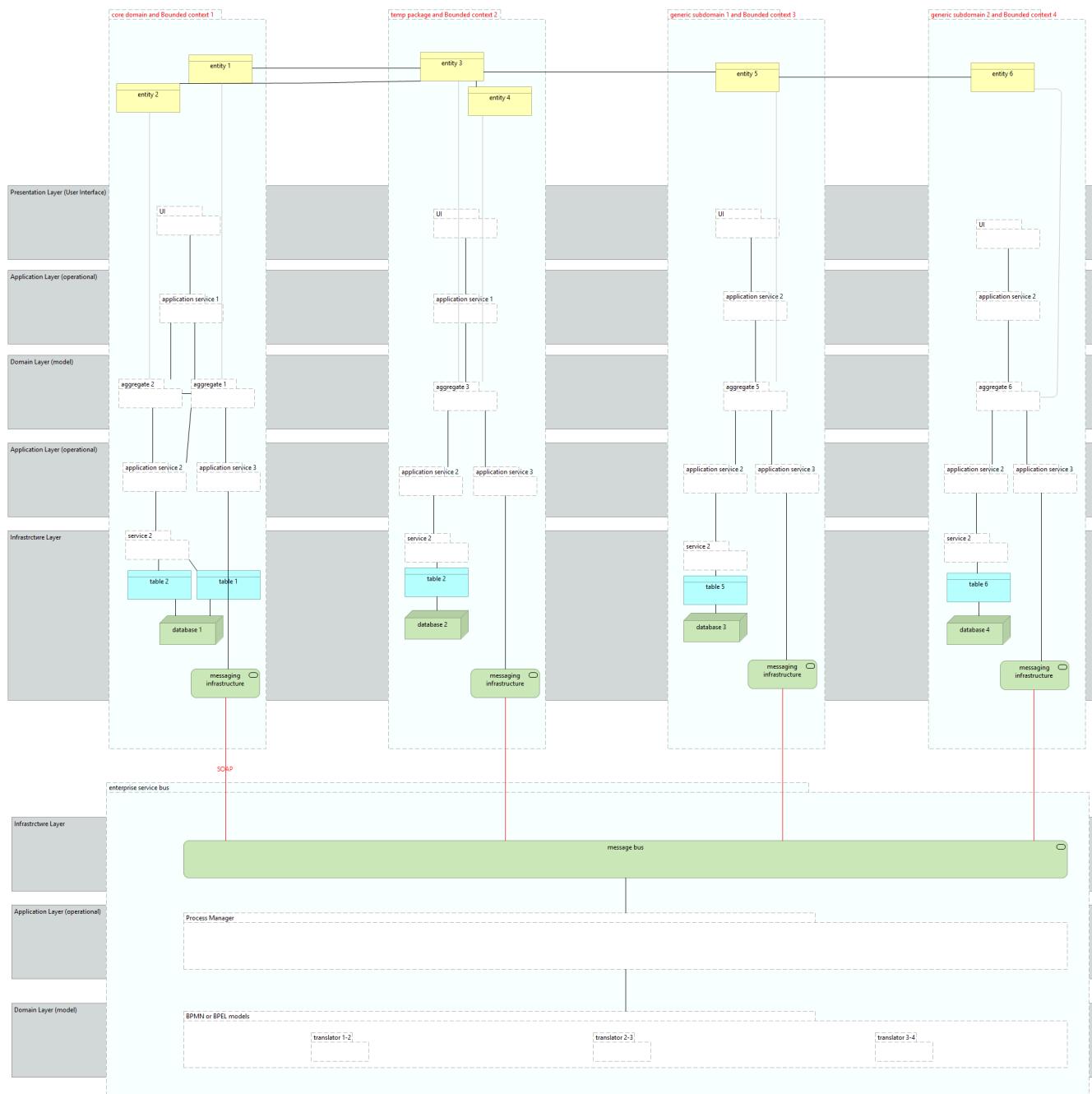
INTEGRATION OF BOUDED CONTEXTS THROUGH DATABASE



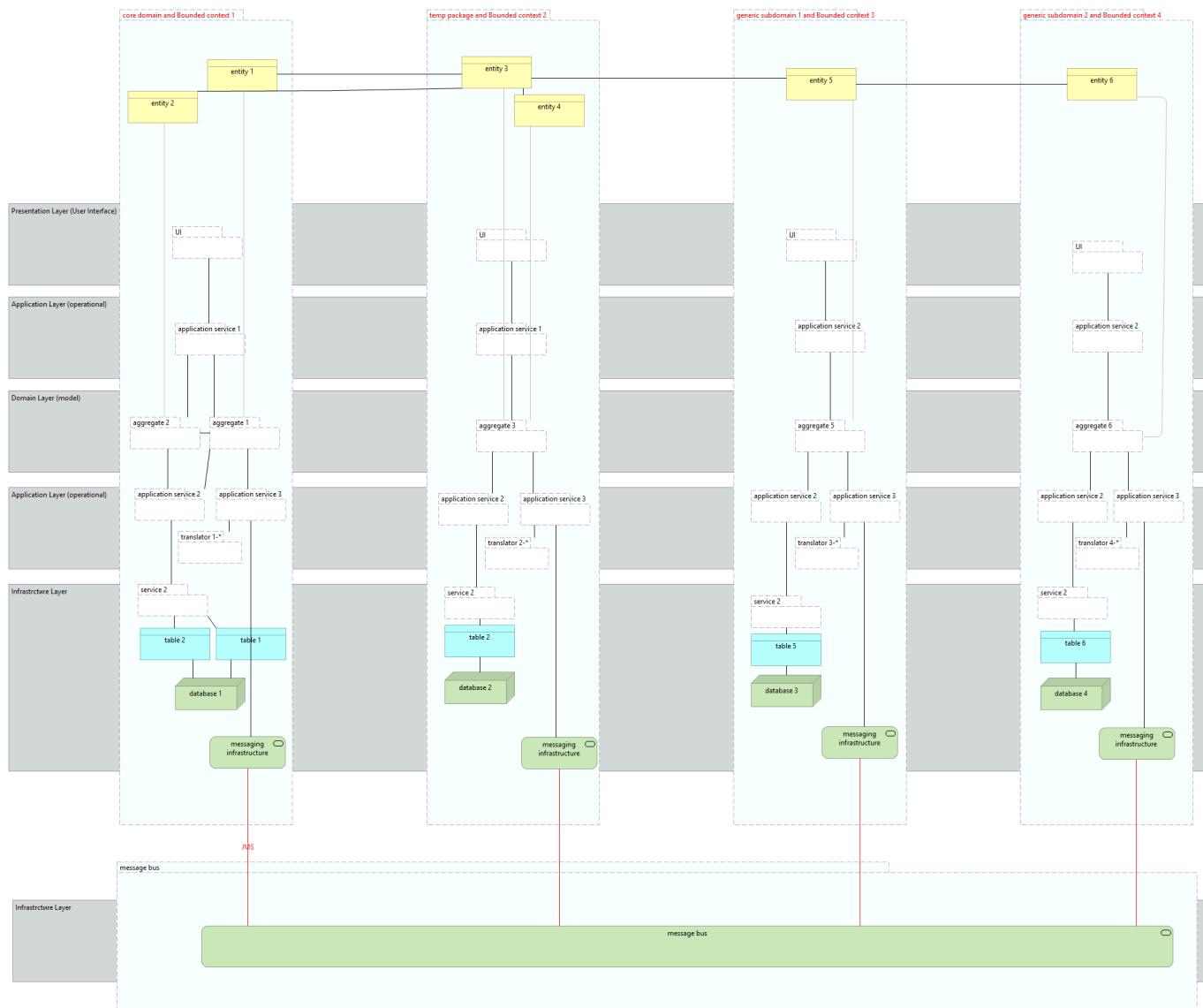
INTEGRATION OF BOUDED CONTEXTS THROUGH FLAT FILES



INTEGRATION OF BOUDED CONTEXTS THROUGH ENTERPRISE SERVICE BUS



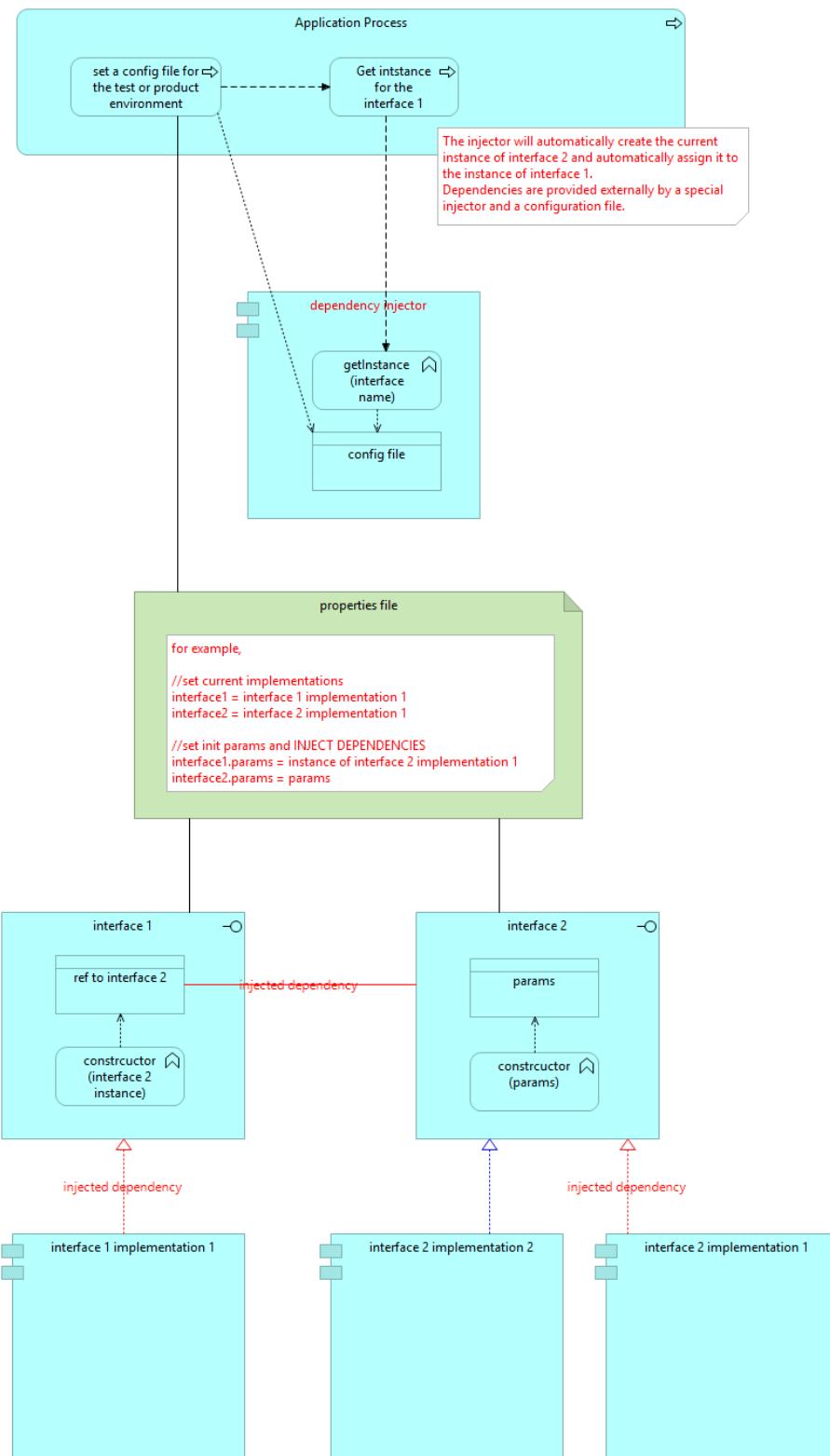
INTEGRATION OF BOUDED CONTEXTS THROUGH MESSAGE QUEUE



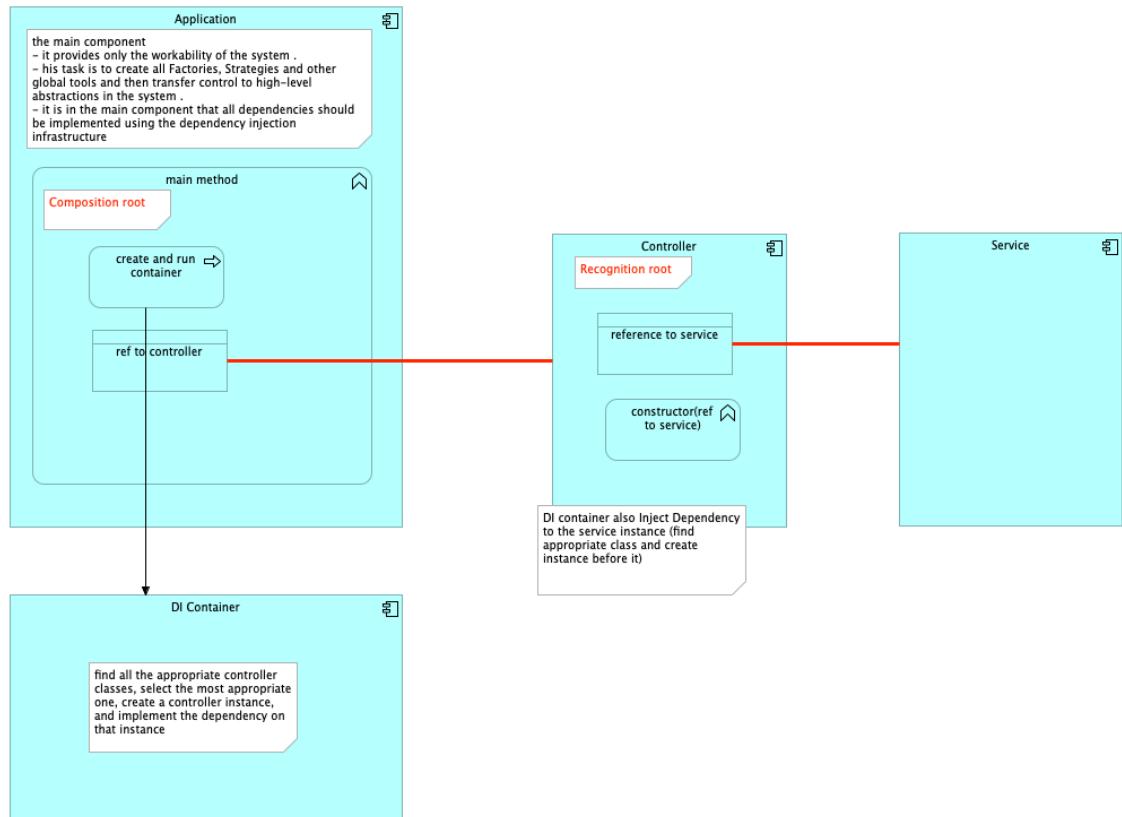
DEPENDENCY INJECTION

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

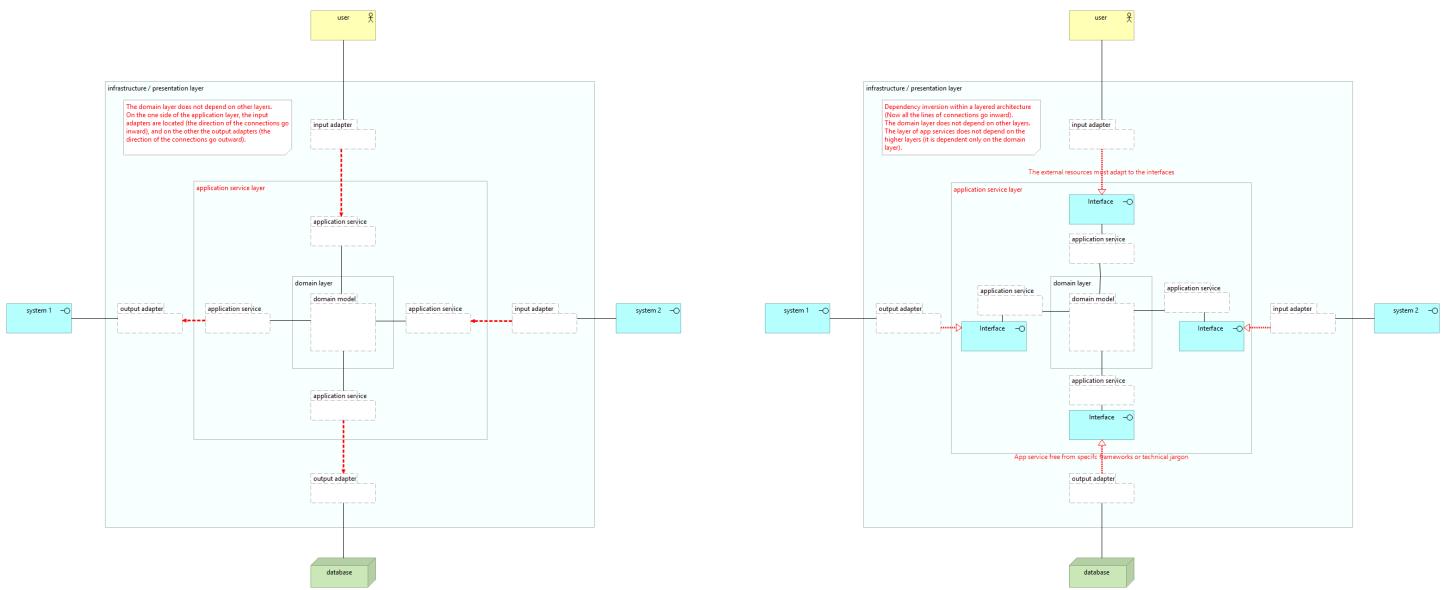
- inject the implementation into the application class
- removing the dependency from the application class to the plugin implementation



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



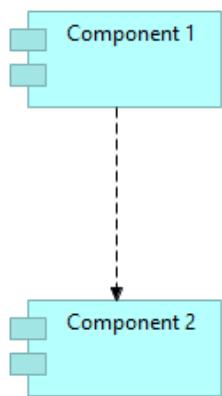
DEPENDENCY INVERSION



INVERSION OF CONTROL

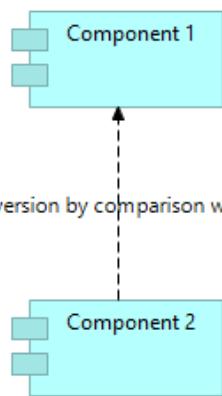
IoC
Inversion of control is about who initiates messages.

Variant 1



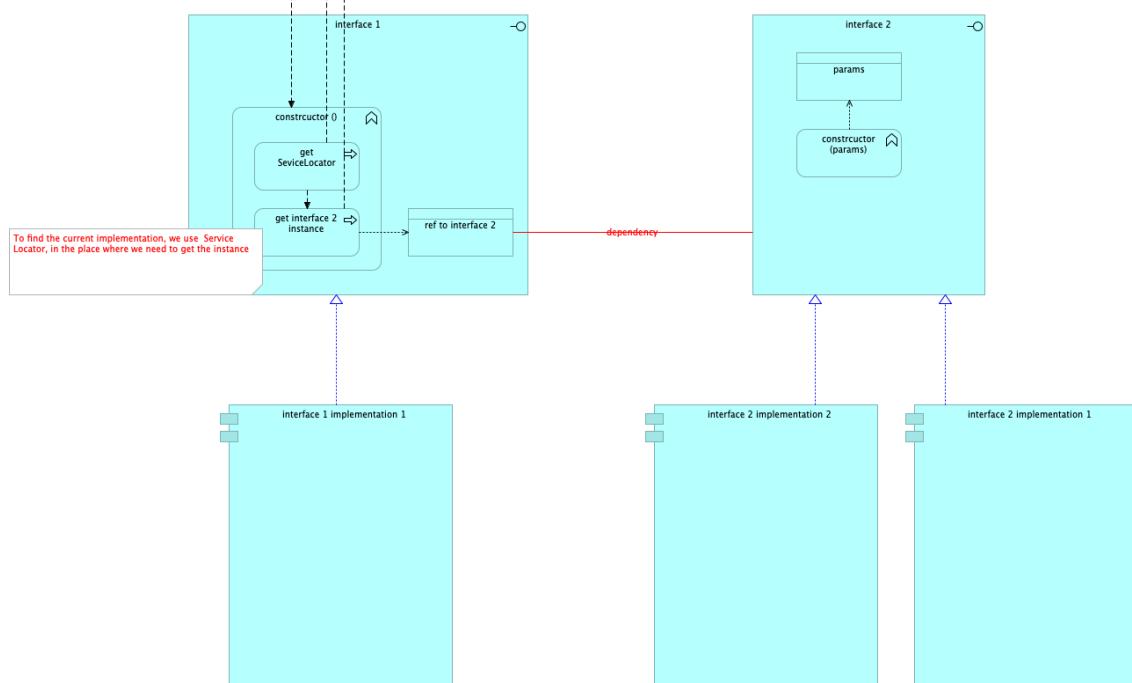
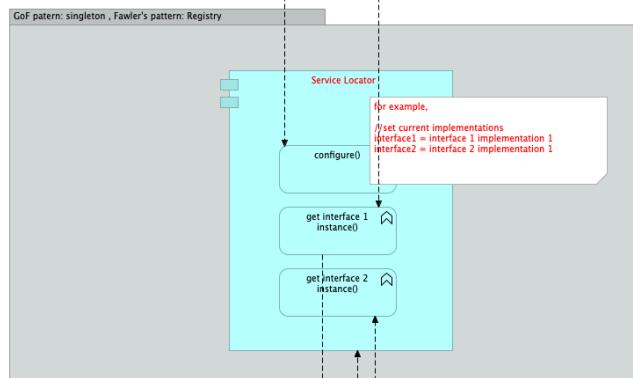
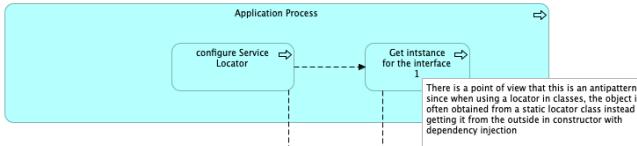
Variant 2

Here the control inversion by comparison with the previous case



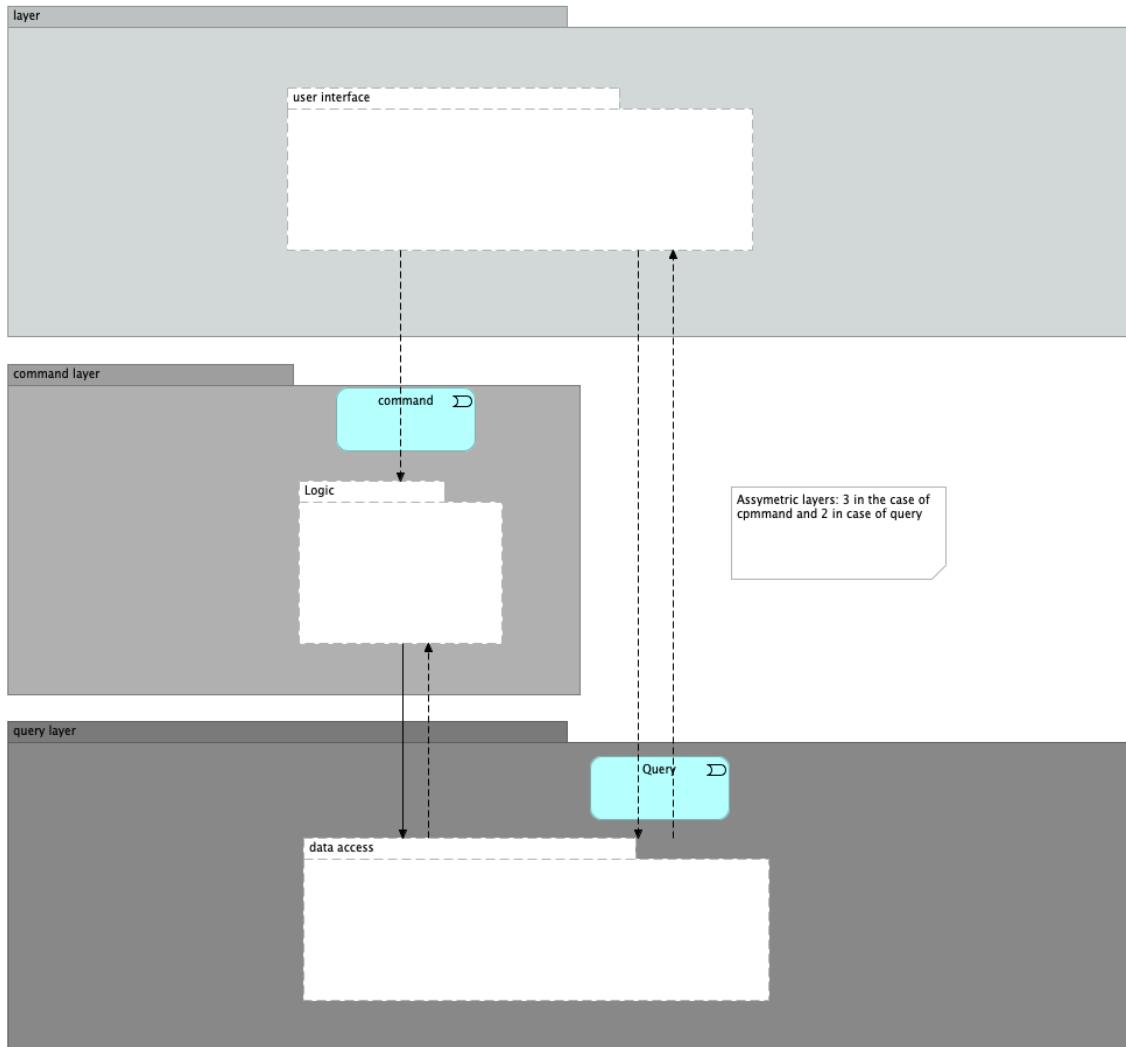
SERVICE LOCATOR

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



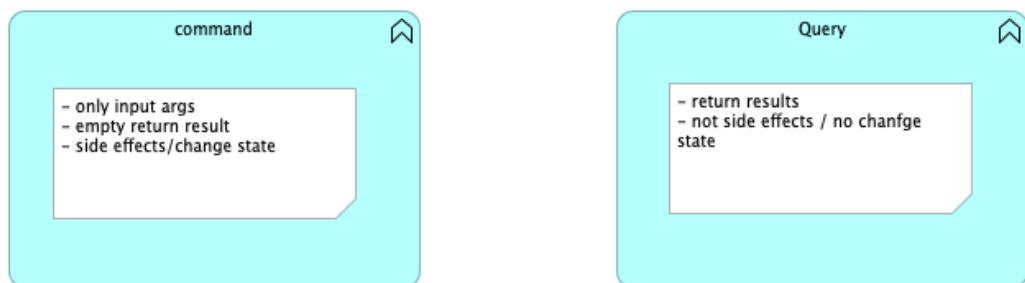
CQRS

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

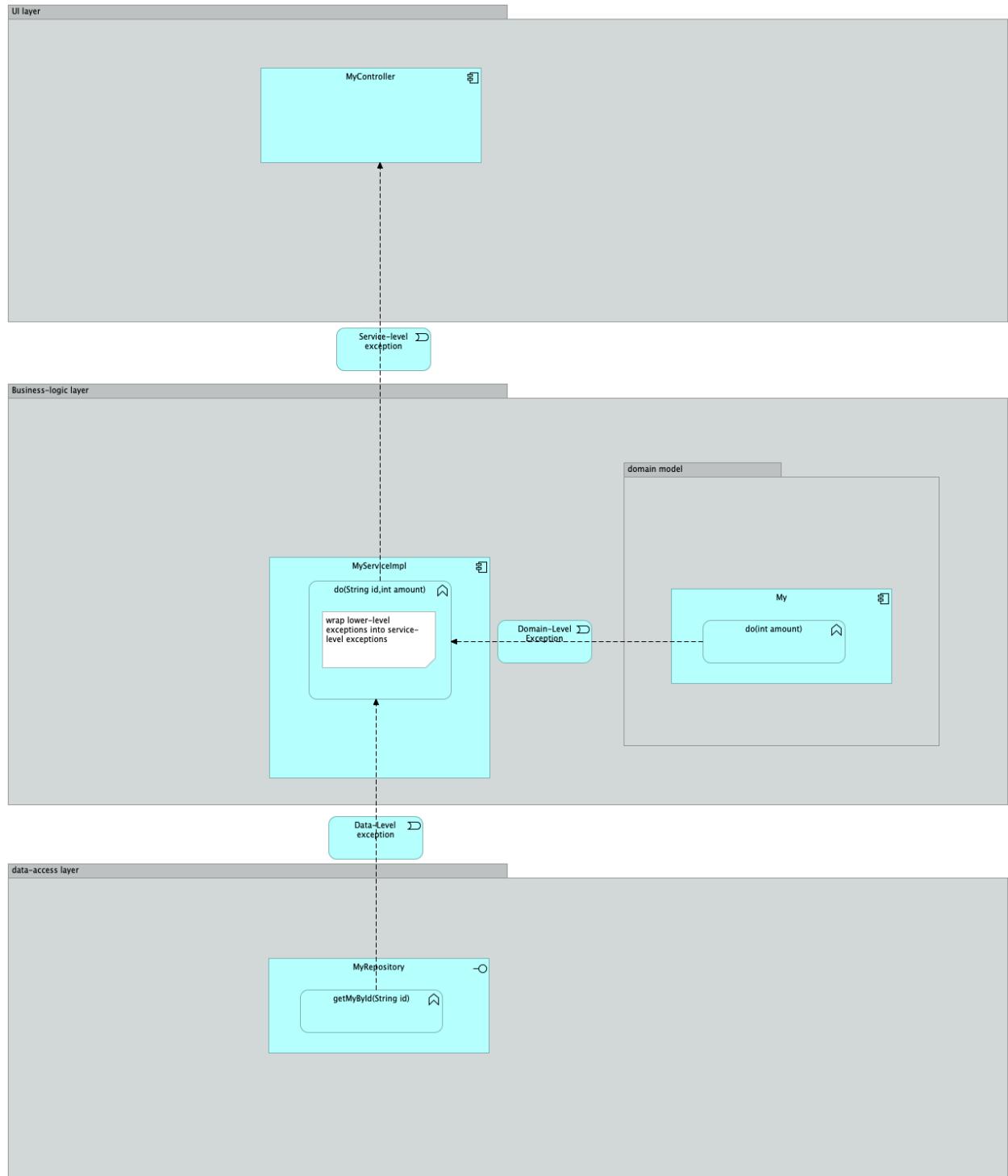


CQS

CQS Command/Query Separation

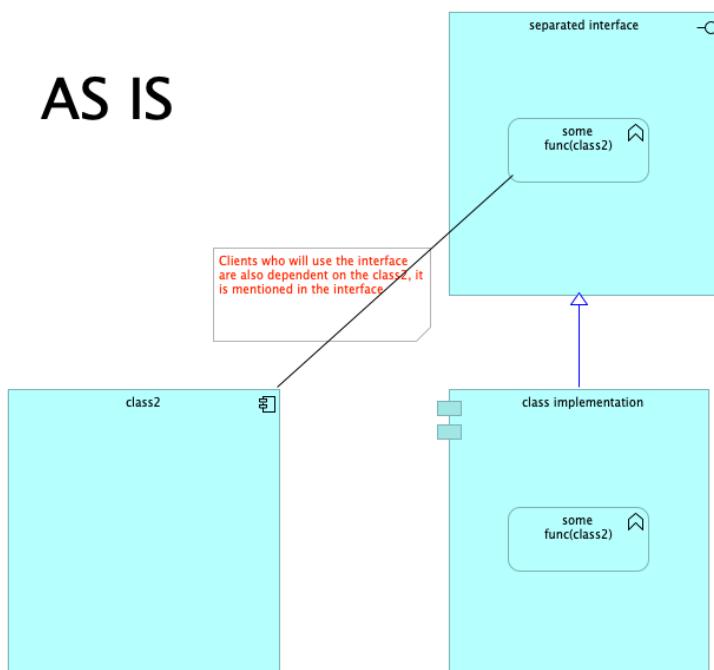


WRAP LOW-LEVEL EXCEPTIONS

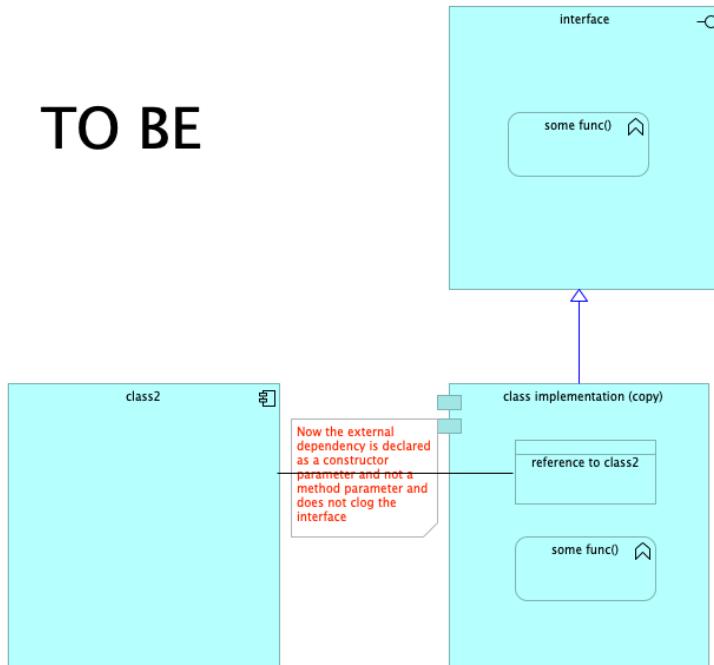


EXTRACT DEPENDENCY FROM INTERFACE TO CONSTRUCTOR

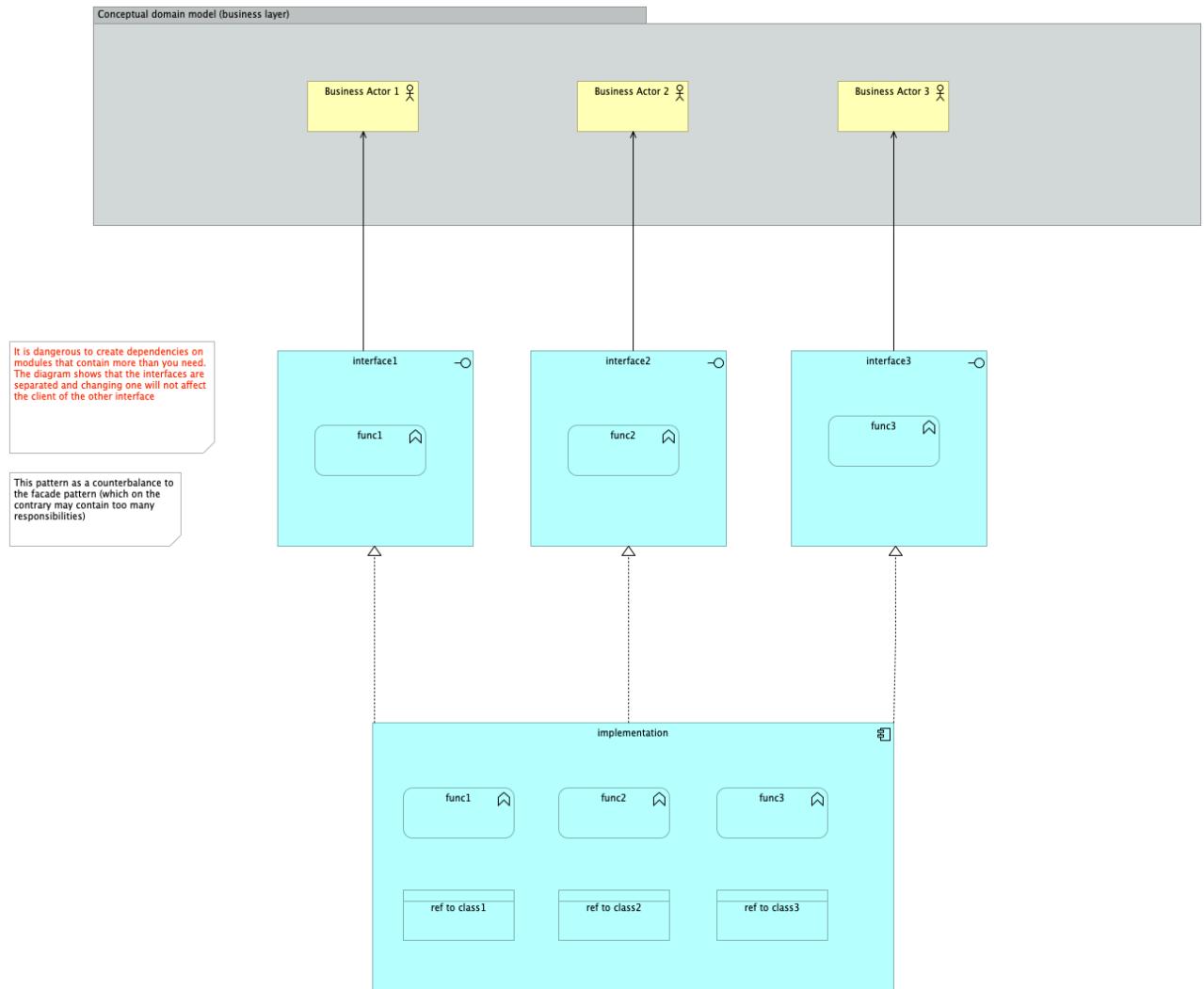
AS IS



TO BE



INTERFACE SEGREGATION

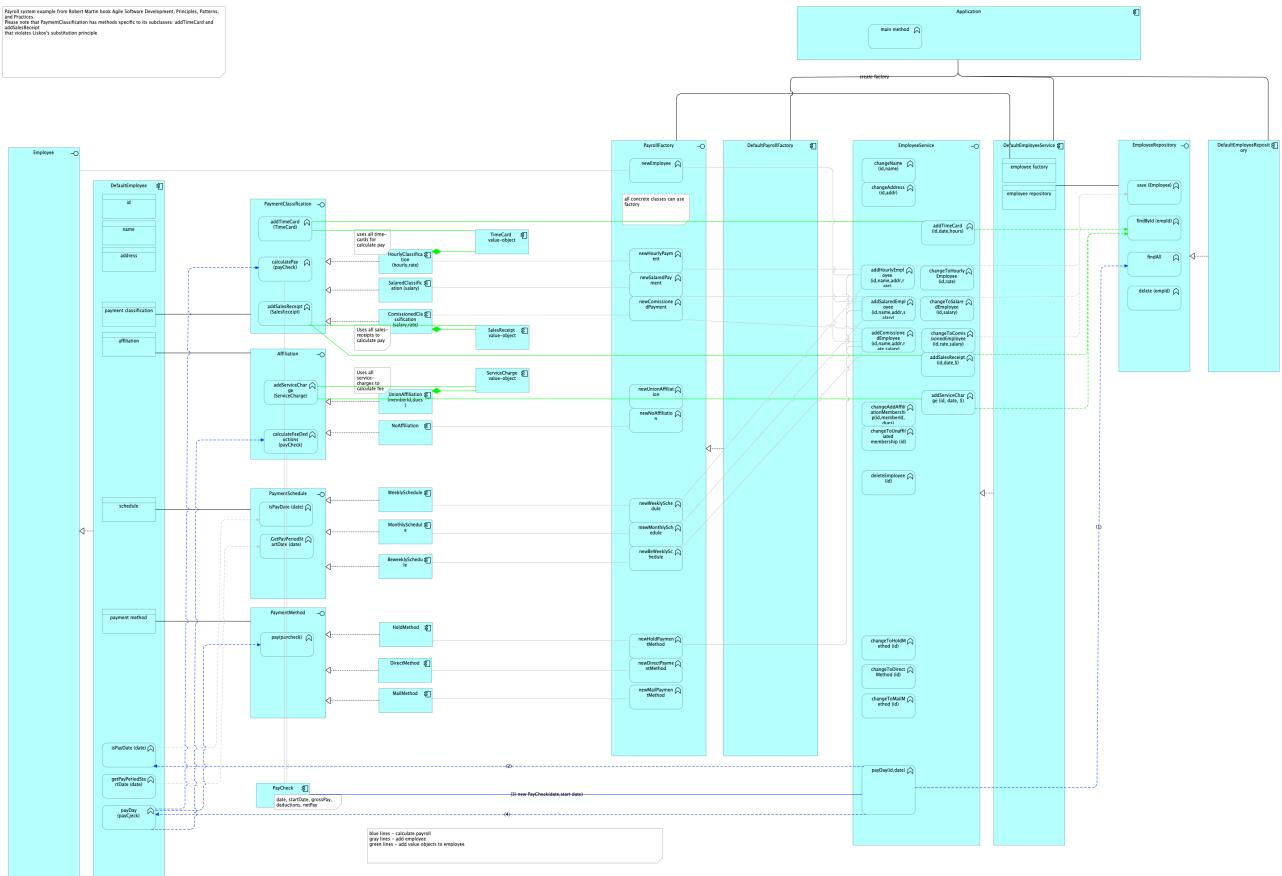


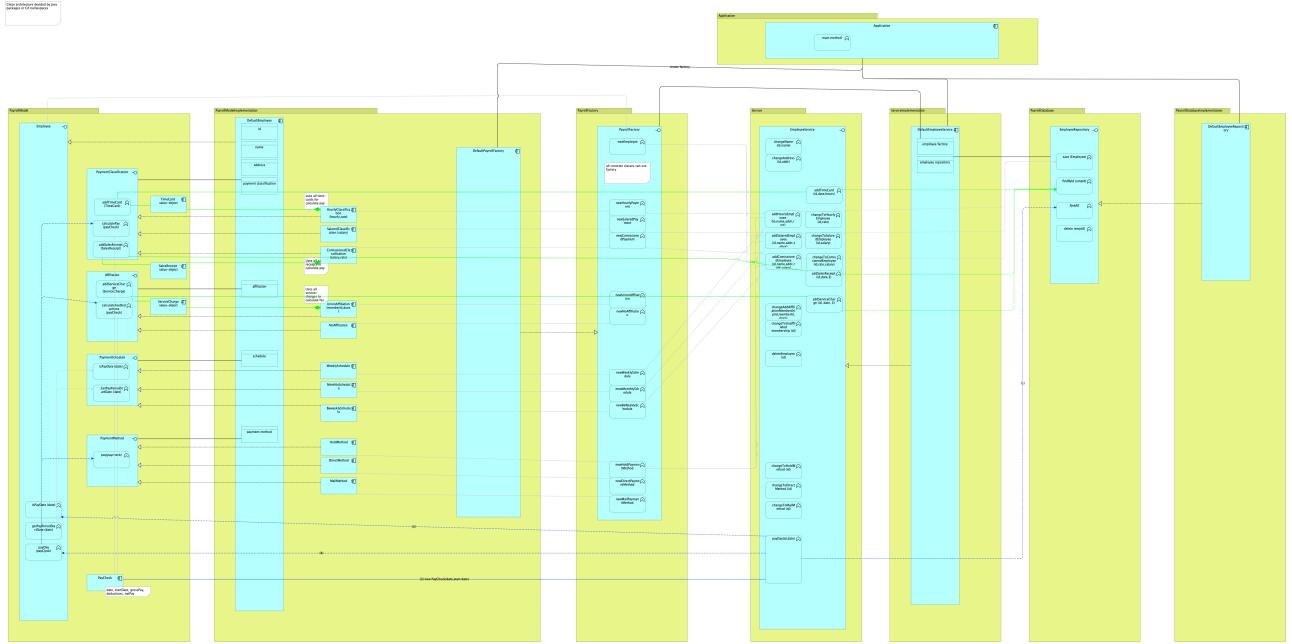
CLEAN ARCHITECTURE

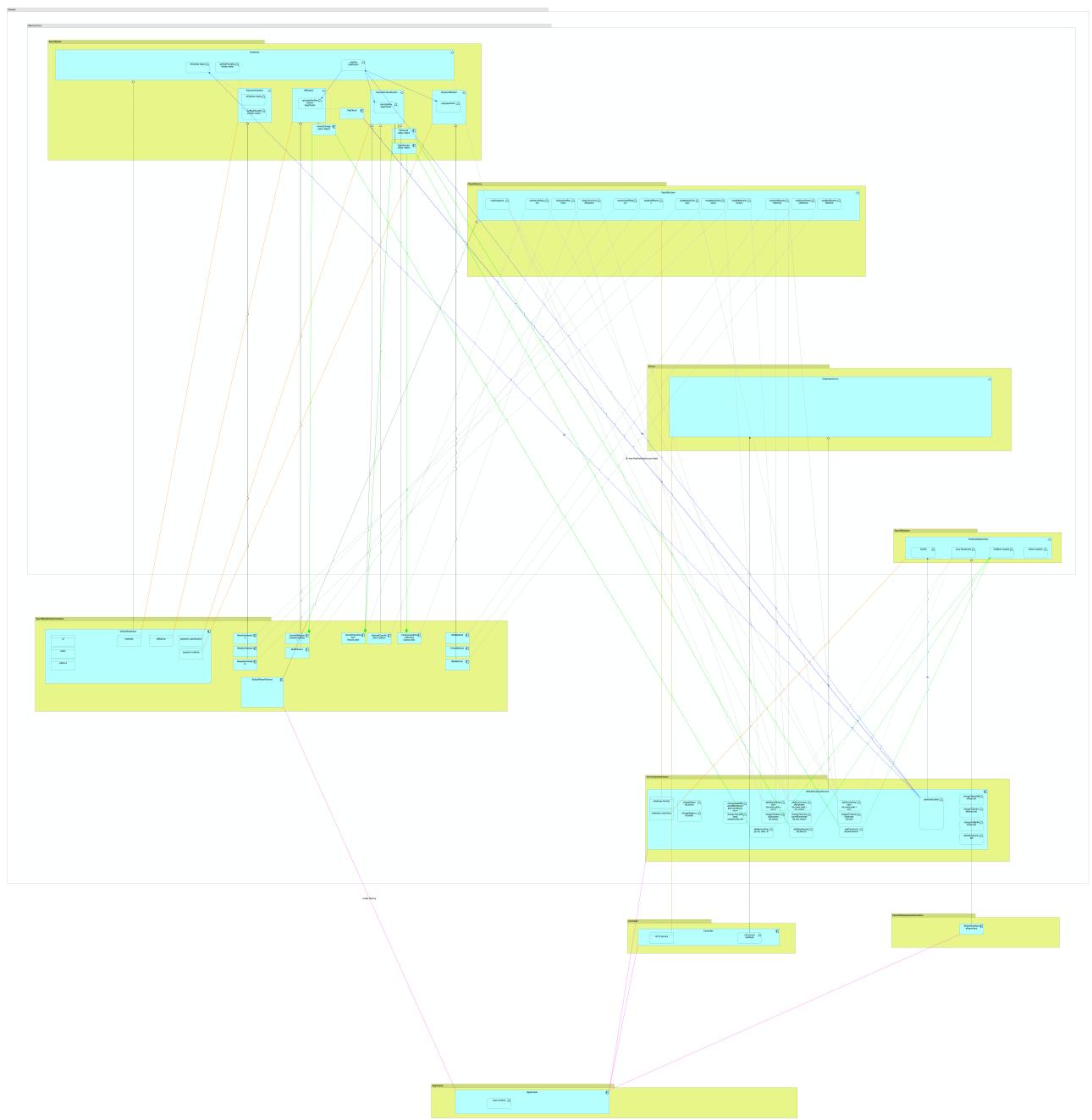
Payroll system example from Robert Martin book Agile Software Development: Principles, Patterns, and Practices.

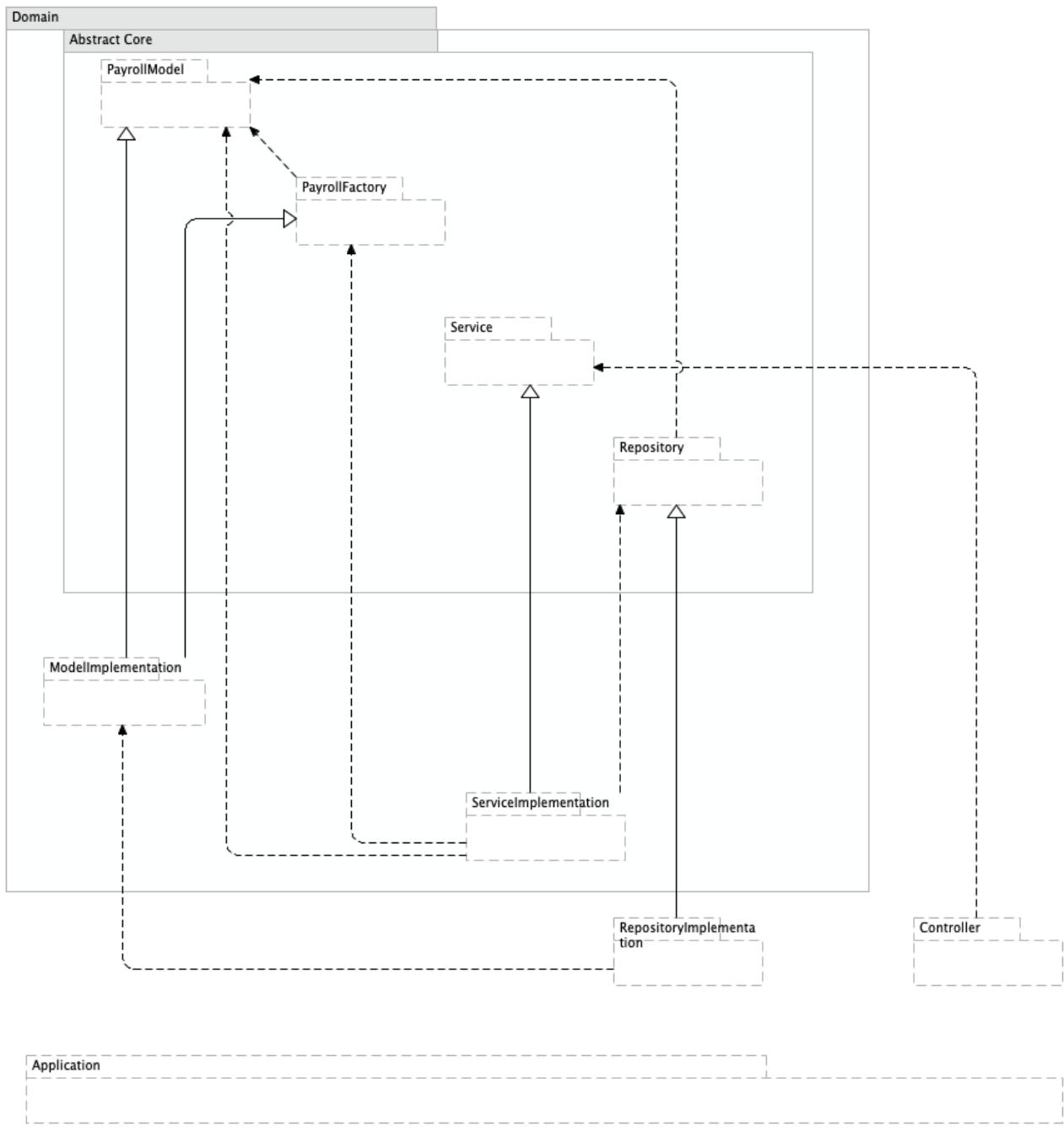
Principle: PaymentClassification has methods specific to its subclasses: addTimeCard and addBankStatement.

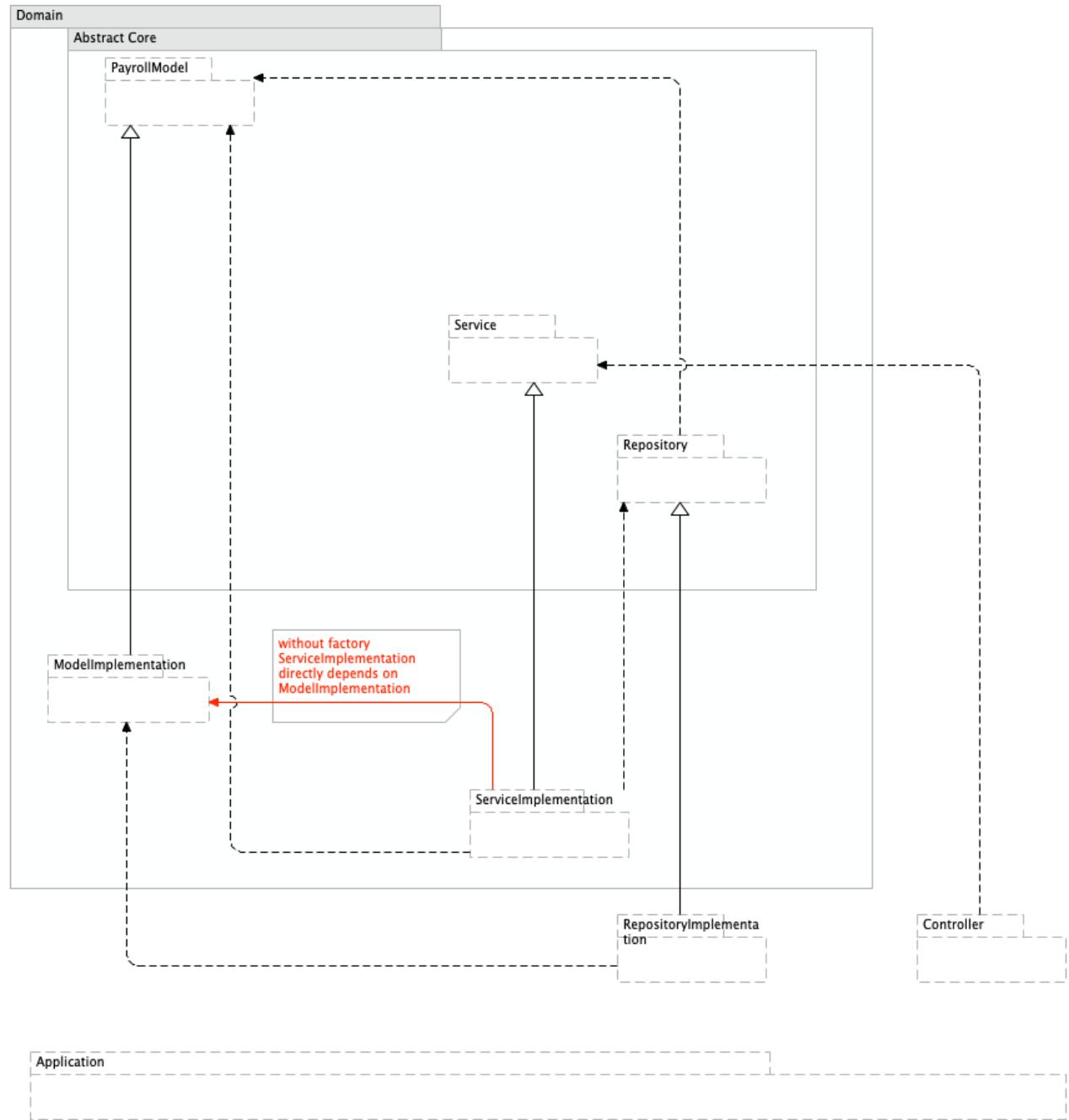
Violation: Liskov's Substitution principle

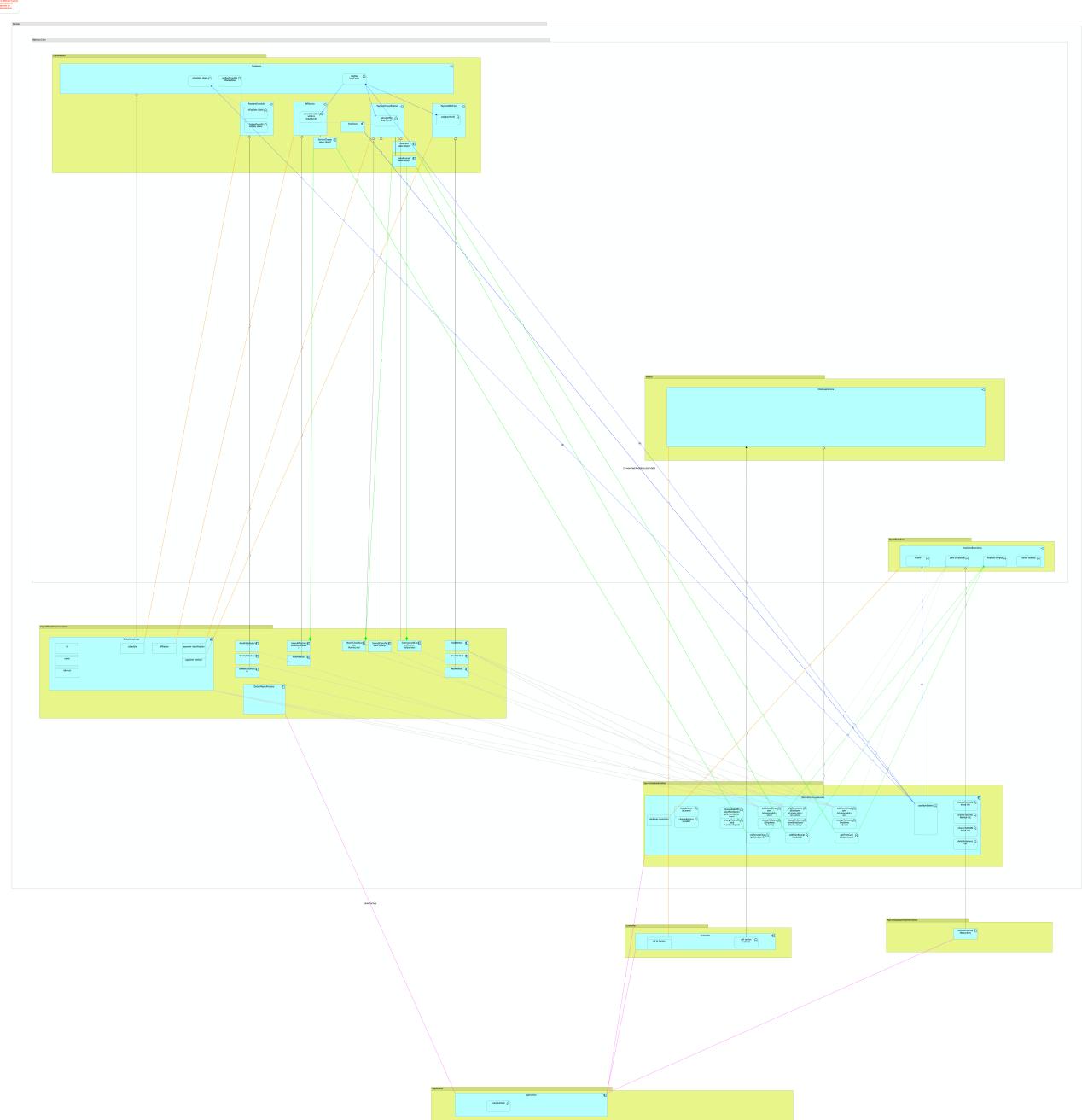




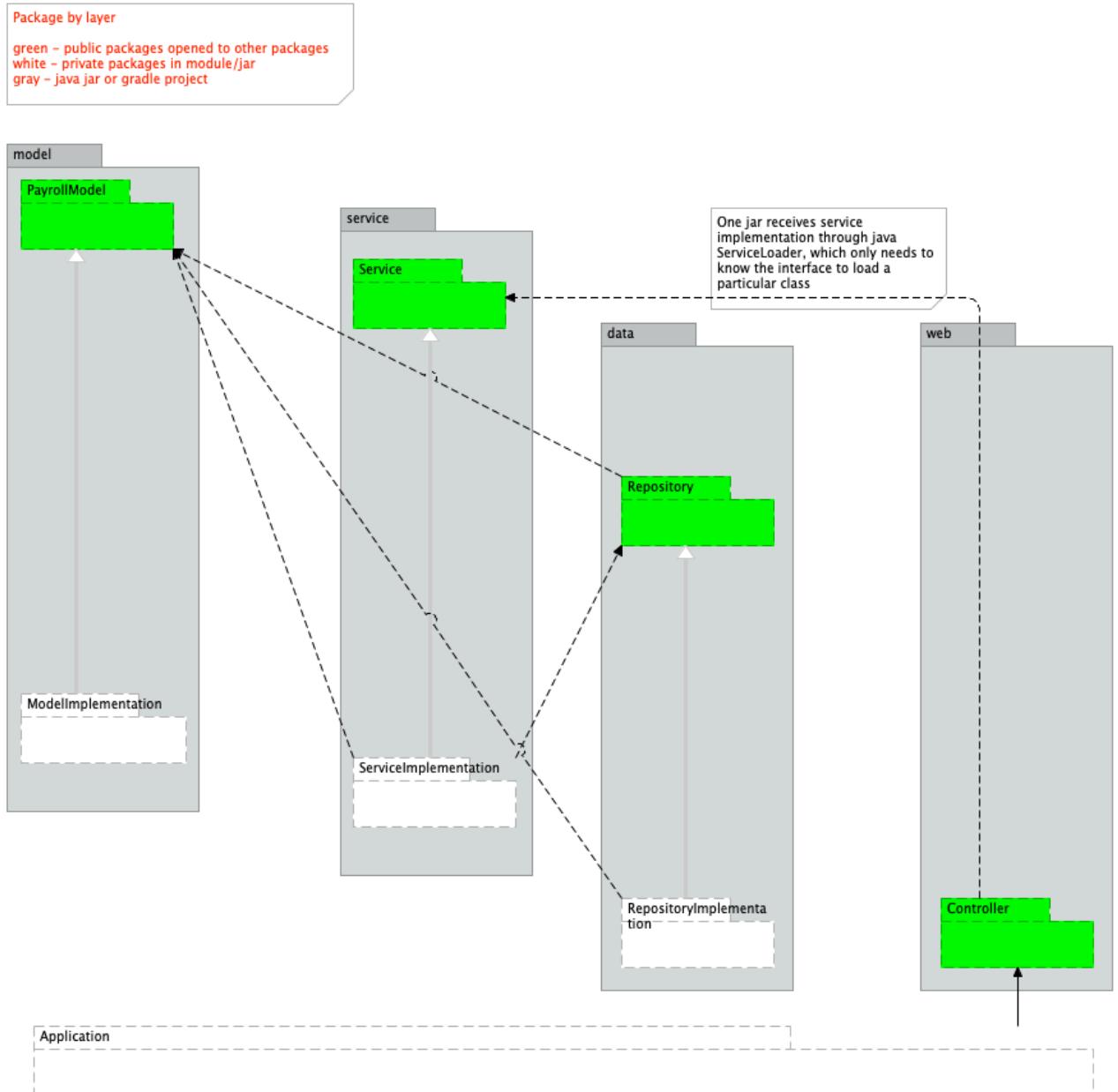




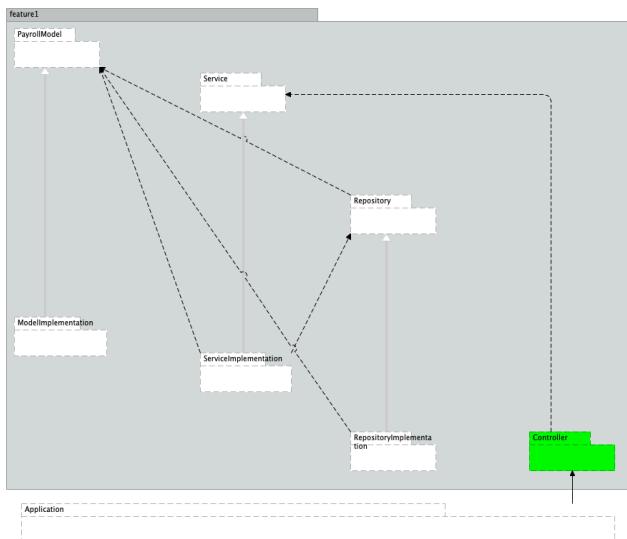




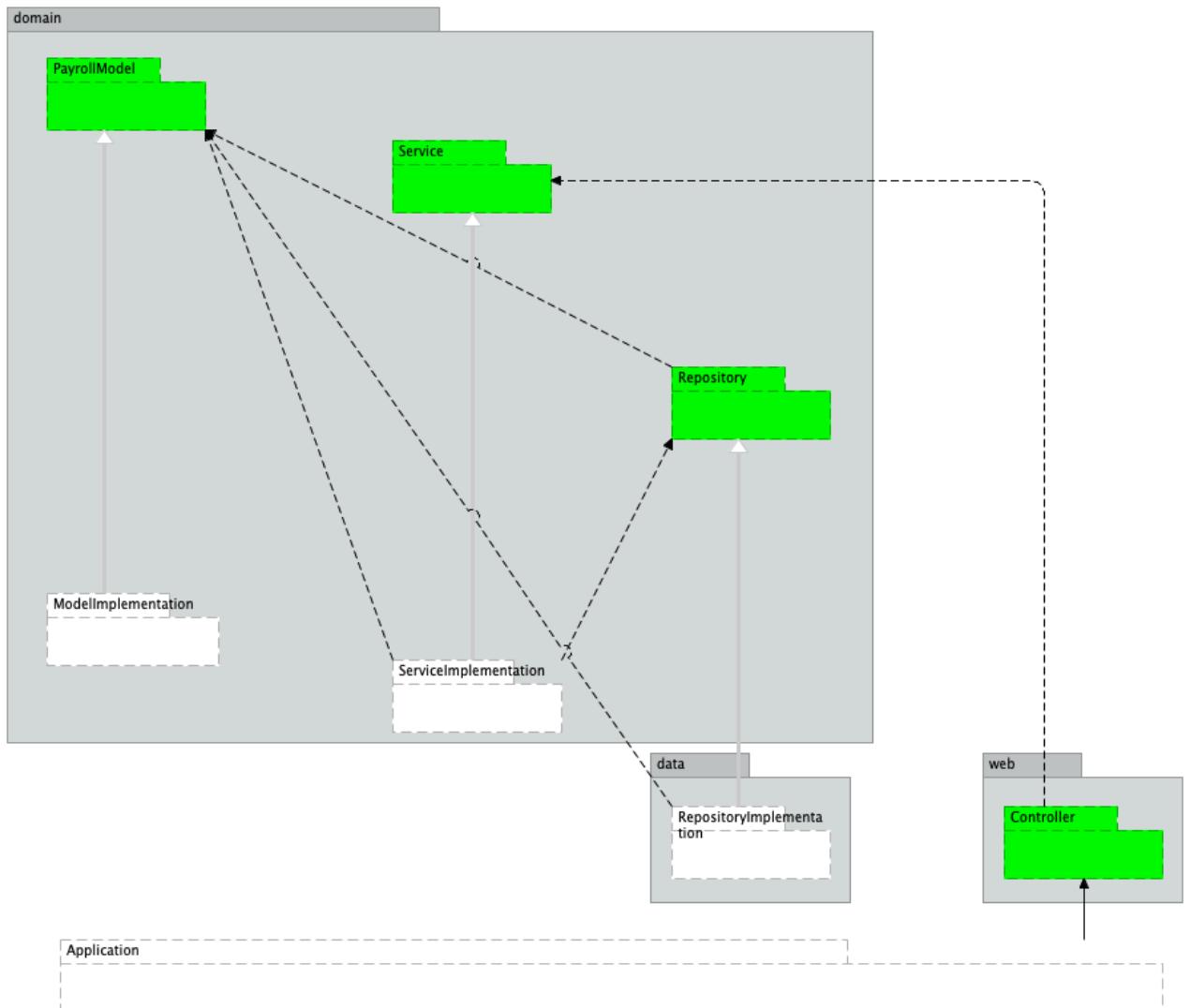
ARCHITECTURAL STYLES



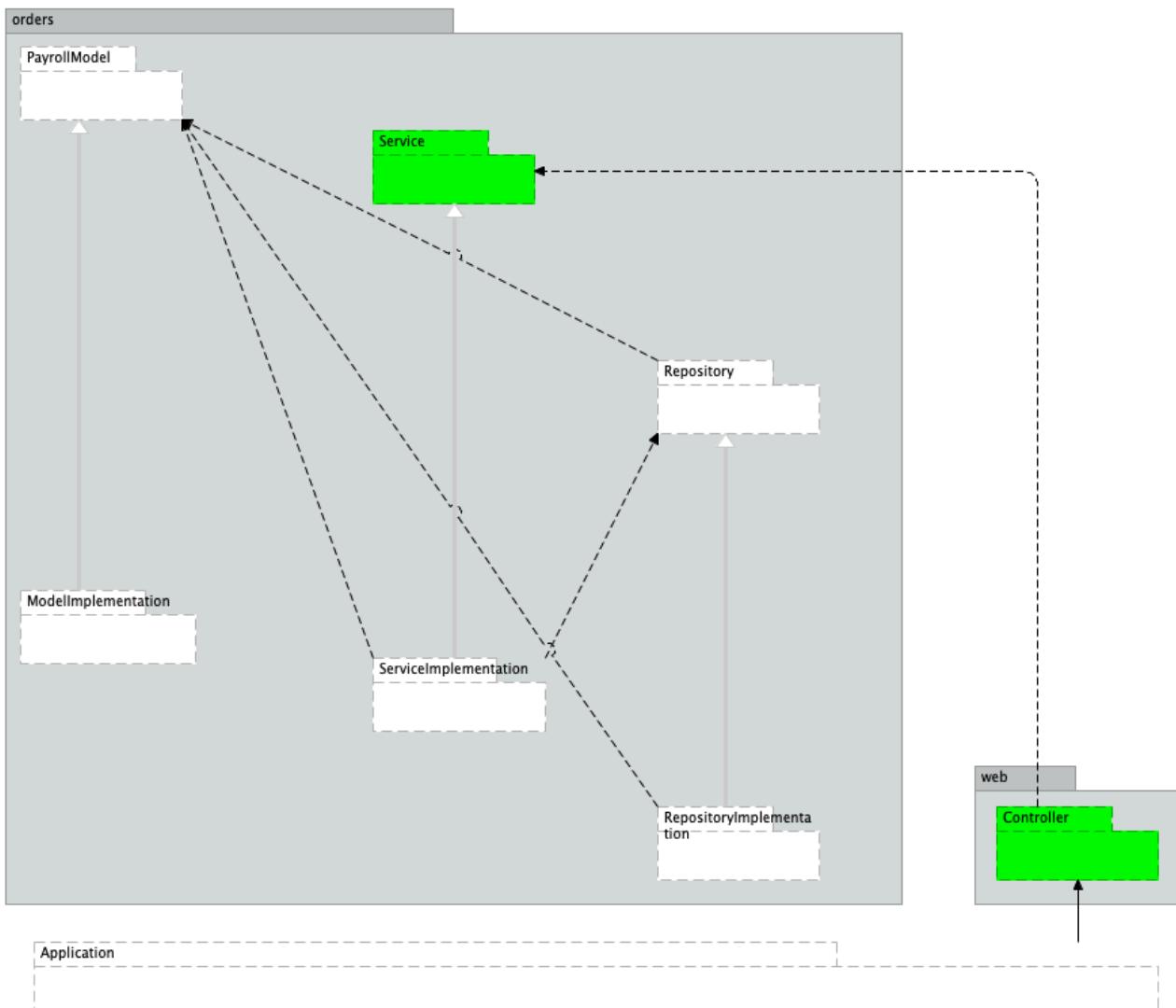
Package by feature
green - public packages in module/jar
white - private packages in module/jar



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



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





SOME ARCHITECTURAL DIAGRAMS
ARE CONSTRUCTED USING
STRUCTURE101 STUDIO,

structure101



SOME ARCHITECTURAL DIAGRAMS
ARE CONSTRUCTED USING
JARCHITECT, COURTESY OF

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



KrispWilmer