Constraint Formalization for Automated Assessment of Enterprise Models

Stef Joosten, Ella Roubtsova and El Makki Haddouchi

Open Universiteit
www.ou.nl

Outline

- Motivation of Constraint Formalization for Automated Assessment of Enterprise Models
- ArchiChecker- How it works
- A Method for constraint preparation and automatic assessment of enterprise models
- A Case study - two enterprise models and ten policies
- Discussion the two next results inspired by the Constraint Formalization for Automated Assessment of Enterprise Models

An Enterprise Model is always a set of views or sub-models

- Enterprise modelling is a set of tools, methods and practices `for an aligned development of all parts of an enterprise, e.g. business, functional, organizational and technical aspects.
- So, an Enterprise Model is always a set of views or sub-models.
- Each view presents a specific part of the Enterprise.
- **Each view or sub-model may be created by a different team member.**

The need of constraint formalisation and checks for Enterprise models in ArchiMate

- The ArchiMate specification has been designed for consistent enterprise modeling. ArchiMate defines an **internal structure of an enterprise model as a sets of views, elements and relations.**
- The internal structure is filled in by elements and relations from views. Different views may use the same element.
- Tools give to each element a unique code and check some generic modelling conventions on views.
- However, each organization defines its own specific policies that may be interpreted as modelling conventions.

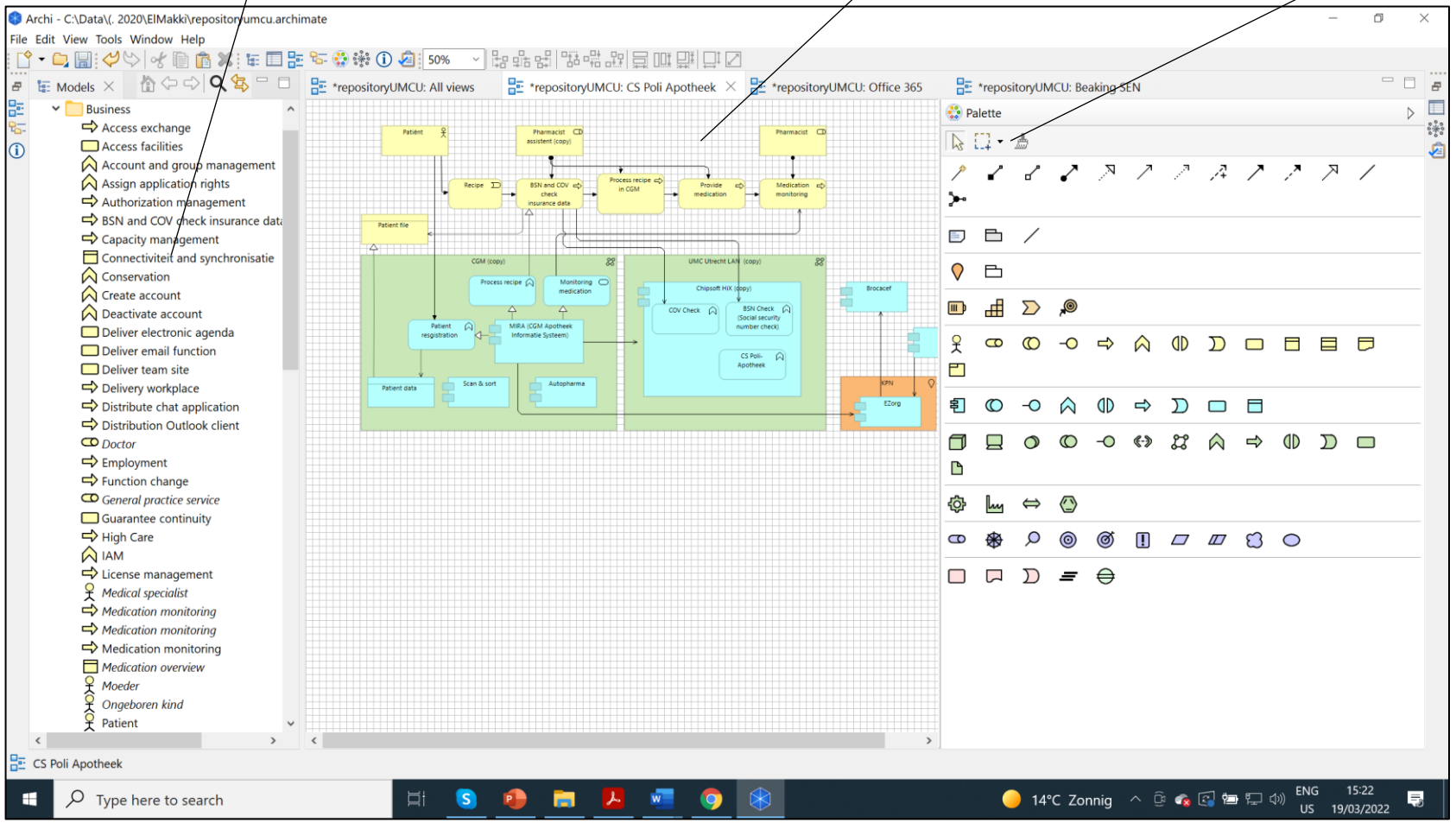
In order to guarantee a consistent team work on an enterprise model, the organization-specific policies have to be formalized for automated checks.

ArchiMate

An internal ArchiMate model

a view

the palette



In each single **view**, ArchiMate shows a set of **elements** (boxes) and **relations** (lines) has been picked from a palette: a limited set from elements of the metamodel.

Each enterprise model has an **internal ArchiMate model** being a collection of views, elements, and relationships.

Different views can use the same elements. An internal key identifies an element.

ArchiMate Metamodel fragment

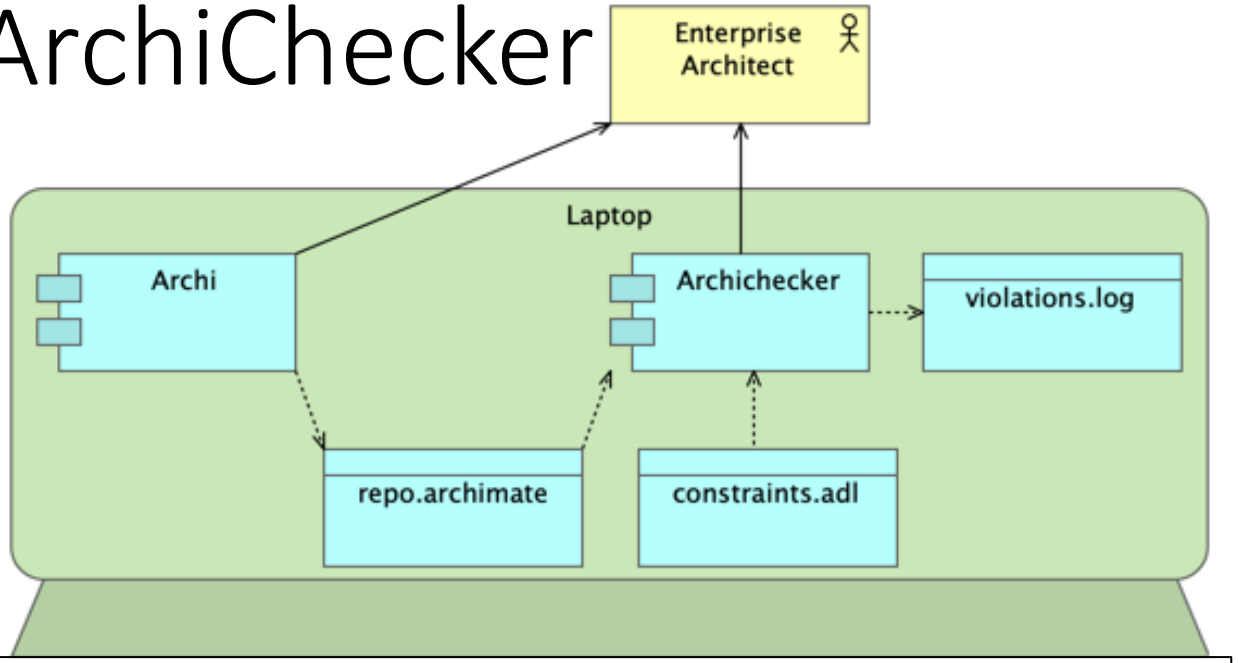
```
250
251    RELATION realization [ApplicationComponent*ApplicationFunction]
252
253
254    RELATION access [ApplicationFunction*DataObject]
255
256
257    RELATION assignment [BusinessRole*BusinessProcess]
258
259
260    RELATION serving [ApplicationComponent*ApplicationComponent]
261
262
263    RELATION assignment [ApplicationComponent*ApplicationFunction]
264
265
266    RELATION access [BusinessProcess*BusinessObject]
267
268
269    RELATION serving [BusinessProcess*ApplicationFunction]
270
```

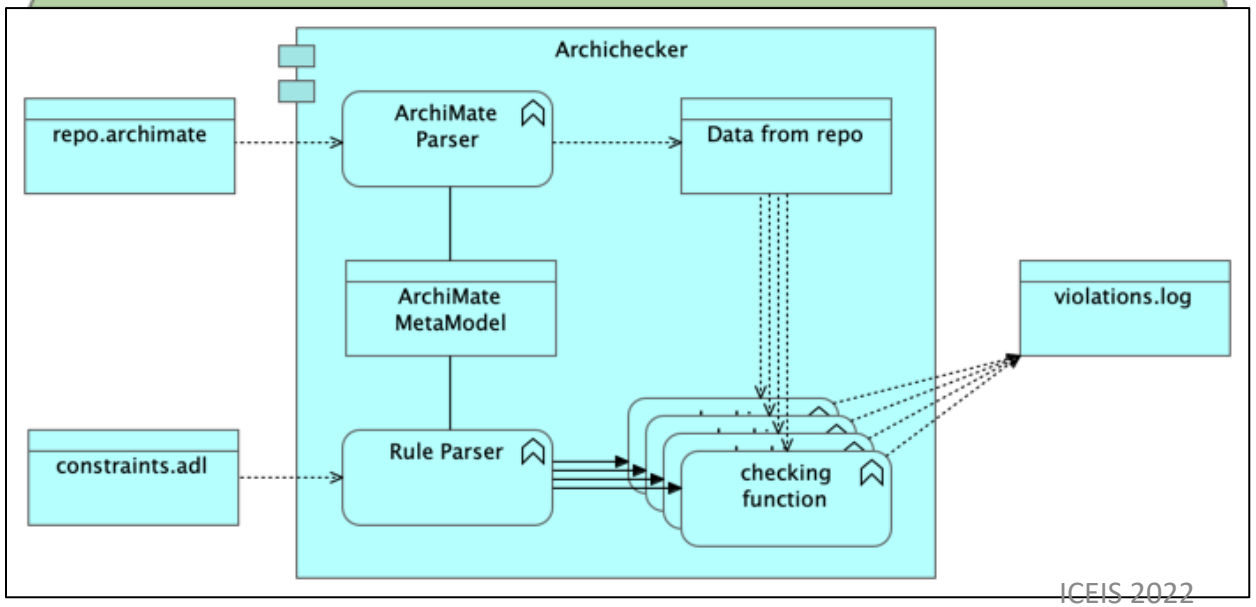
A fragment of an Enterprise Model populated with instances of elements

```
1082    POPULATION realization [Location*ApplicationComponent] CONTAINS [( "18df25ec-9db9-4aa7-9ef5-7948c3f37112" , "d9e0a493
1083    POPULATION triggering [BusinessEvent*BusinessProcess] CONTAINS [( "05df36e0-1dde-4f07-85d4-de4b0c9e7586" , "3f7dfb5e-
1084    POPULATION triggering [BusinessActor*ApplicationFunction] CONTAINS [( "5a48ce8e-d0fe-4e02-9bdb-2969da063200" , "1ba34
1085    POPULATION realization [BusinessFunction*BusinessService] CONTAINS [( "37d64852-3ed1-466e-99d7-22a64c36516d" , "77c76
1086                                                              ,( "823b1d78-3424-4ab7-b677-b4d9e4d4af5a" , "86768
1087                                                              ,( "fa69bf84-a264-4d38-bf96-04a0dfd34644" , "9b8ba
1088                                                              ,( "77bb5f5f-c55c-4bc4-987f-2fa63554dace" , "3f39d
1089    POPULATION serving [BusinessFunction*ApplicationFunction] CONTAINS [( "37d64852-3ed1-466e-99d7-22a64c36516d" , "ccfa9
1090                                                              ,( "823b1d78-3424-4ab7-b677-b4d9e4d4af5a" , "9011b
1091                                                              ,( "823b1d78-3424-4ab7-b677-b4d9e4d4af5a" , "13784
1092                                                              ,( "b7a14070-8fab-4343-ab67-46cb9f0ad9d8" , "4e38b
1093                                                              ,( "d8d6062f-a5a2-4c59-a0b7-85af380adba5" , "4e38b
1094                                                              ,( "c185778b-fff7-4f73-8627-c9d736787bc5" , "29a5f
1095    POPULATION realization [BusinessProcess*BusinessService] CONTAINS [( "50c0a317-98f8-4b7c-b17b-d87d24b5bd05" , "413dbf
1096                                                             ,( "c0132f25-03f8-40c0-945b-f1c0460c1216" , "e5ba09
1097                                                             ,( "cbaa6931-504a-4e6b-a616-09f80d7a1d38" , "adc163
1098                                                             ,( "cbaa6931-504a-4e6b-a616-09f80d7a1d38" , "594937
1099                                                             ,( "22548614-8da8-4734-8311-11922da051ea" , "448c70
1100                                                             ,( "22548614-8da8-4734-8311-11922da051ea" , "f0eb5c
1101                                                             ,( "22548614-8da8-4734-8311-11922da051ea" , "392101
1102    POPULATION triggering [BusinessProcess*BusinessFunction] CONTAINS [( "aa2d0e70-5d0f-4e43-bf80-574cefaa5382" , "d8d606
1103                                                             ,( "3384f350-ffa7-4798-bff4-410b66ca1044" , "b7a140
```

In this model filled with instances we need to find violations of a constraint.

ArchiChecker





Tool Archi stores an ArchiMate model in the form of an XML file with extension ".archimate": in the figure "repo.archimate".

Constraints are stored in constraints.adl.

Component ArchiChecker parses the ArchiMate model (repo.archimate) according to the ArchiMate Metamodel.
It parses constraints (constraints.adl) according to the same metamodel.

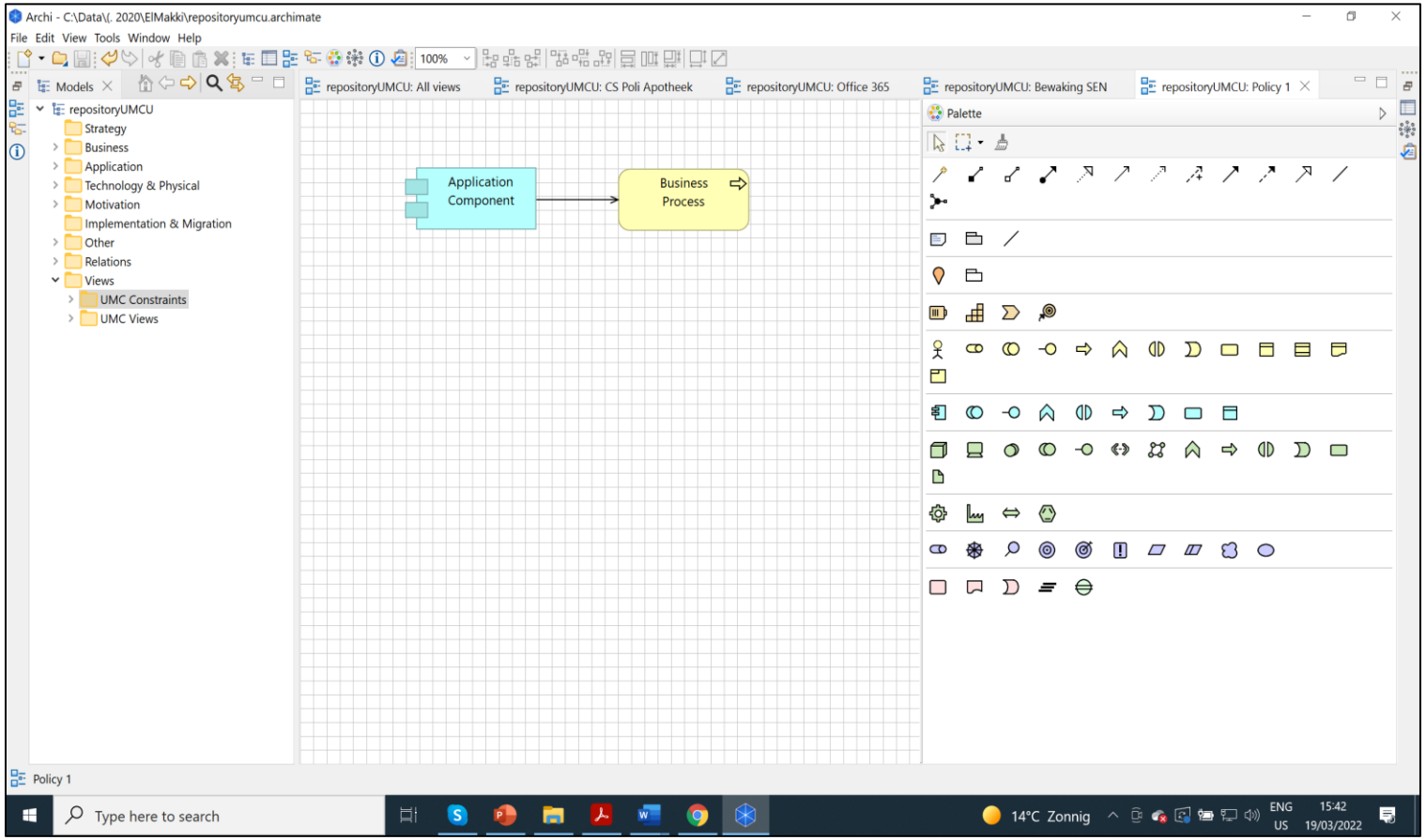
It compiles each constraint into a checking function that can compute violations with respect to that constraint.

ArchiChecker emits all violations to the log.

In literature: Constraints Modelling in ArchiMate

- Not Formalised - Graphical presentation in ArchiMate some constraints of access, security and privacy: Zhi et al.(2018), Mayer et al.(2019), Blanco-Lain´e et al.(2019)
- Formalization of constraints as
  - The ontological specification - Marosin et al.(2014)
  - The semantic web- Kharlamov et al.(2016)
  - The tool Archi (Beauvoir and Sarrodie, 2018) with a JavaScript-based scripting plug-in called jArchi (Beauvoir and Sarrodie, 2019).
  - The formalization of the metamodel of the ArchiMate language in Alloy (Babkin and Ponomarev, 2017) and the use of the MIT Alloy Analyzer.

We proposed to formalise constraints using the metamodel elements of ArchiMate itself



- A constraint can be visualised in ArchiMate.
- We propose to derive the metamodel elements and relations from the internal ArchiMate model for constraint formalization.
- **The enterprise model and the analysed constraint are in the same metamodel of ArchiMate.**
- The analysed constraint is than formalised as a rule in the Ampersand language to be used in our tool called ArchiChecker.

Policies in the Project Start Architecture Documents of a Medical Center

1. Only one information system is in use for each functionality.
2. Unambiguous and one-time recording of data (and multiple use).
3. Each business process should be realized by at least one application system.
4. A Business process has precisely one owner.
5. Healthcare providers and patients work with one shared file.
6. A data or data group uses one or more business objects.
7. The continuity of critical systems of the Medical Center is guaranteed.
8. Use of central applications is mandatory.
9. The core of information provision is an Enterprise Data Warehouse (EDW).
10. Every data and data type has someone responsible.

A method to formalise and analyse a policy

1. Reformulate the policy in natural language.
2. Visualize the policy in terms of ArchiMate element types, properties and relationships.
3. Formulate an expression of the policy using the Predicate Logic and Ampersand.
4. Run the ArchiChecker on the ArchiMate model to generate a list of policy violations.
5. Analyse the found violations and the given ArchiMate view to find possible reasons of violations and possible actions.

Policy 5 - Healthcare Providers and Patients Work with One Shared File

1. **Reformulate the policy in natural language.**

   For every patient, there must be one file called “Patient File”.

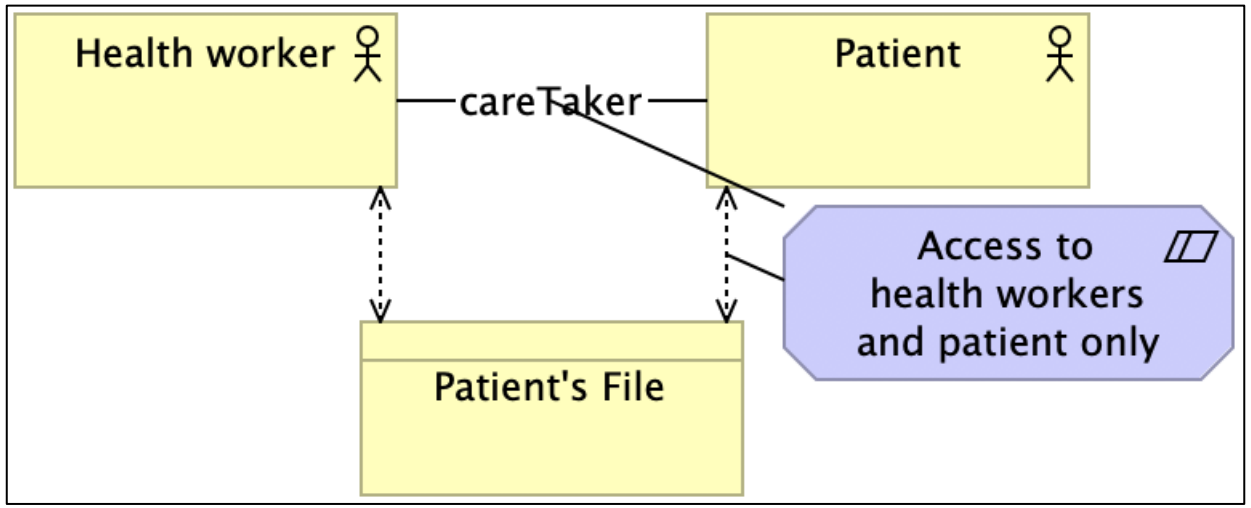
   Each patient must have access to his or her “Patient File”. .

   Each health worker directly involved in medical care for a patient must have access to that patient’s file.

   Others have no access to this file.

Policy 5 - Healthcare Providers and Patients Work with One Shared File

2. **Visualize the policy in ArchiMate.**



3. Formalize in Predicate Logic.
Let patient be a predicate on **BusinessActor**.
Let **Patient File** be a predicate on **BusinessObject**.
Let **careTaker** be a relation **BusinessActor** × **BusinessActor** that relates health workers and their own patients.

$$\forall a,b \in BusinessActor$$
$$\forall o \in BusinessObject :$$
$$a\ careTaker\ b \Rightarrow\ b\ access\ o \wedge a\ access\ o.$$

Policy 5 - Healthcare Providers and Patients Work with One Shared File

3. Formalize in Ampersand. We need a textual form of the produced feedback.

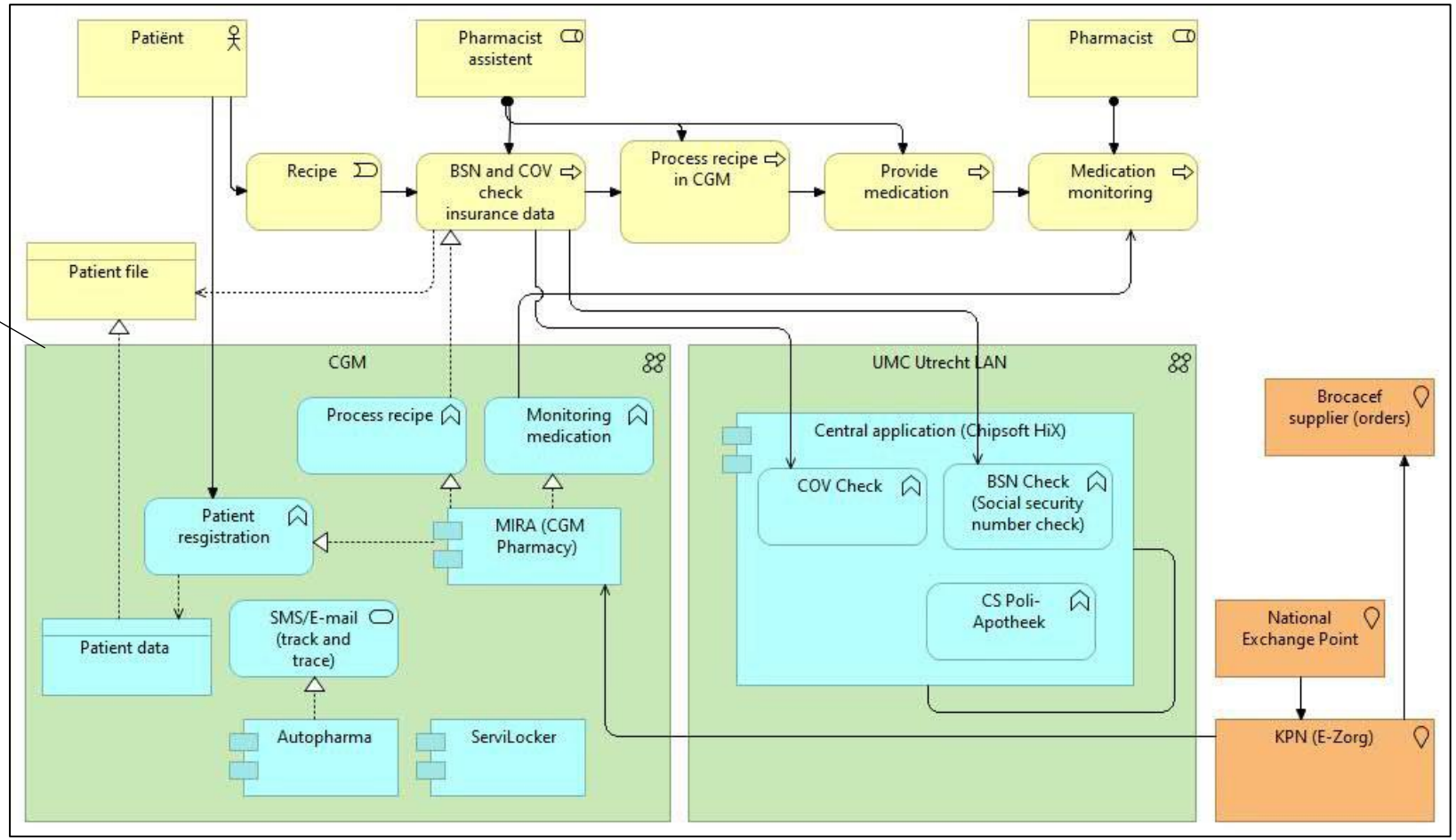
CLASSIFY PatientFile ISA BusinessObject
CLASSIFY Patient ISA BusinessActor
CLASSIFY Patient ISA Person

I[Patient] = name;"Patient";name˜
I[PatientFile] = name;"Patient's File";name˜
[Patient] |- access[Patient*PatientFile] ;
access[Patient*PatientFile]˜careTaker;
access = access[Patient*PatientFile]˜
RULE "MC 5": I [BusinessObject] |- access [BusinessObject*BusinessActor]; access [BusinessObject*BusinessActor]˜

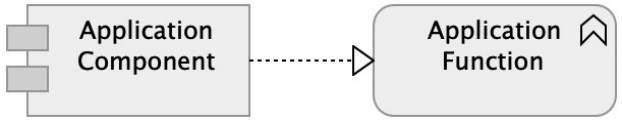
4. Run the ArchiChecker with an ArchiMate Model.
Tool: PatientFile (Business Object) 'Patient file ' is not accessed by a caretaker/Patient (Health worker).
5. Analyse violations**.**

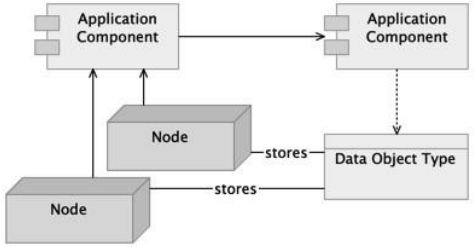
5. Analyze Violation: ’Patient file ’ is not accessed by a CareTaker

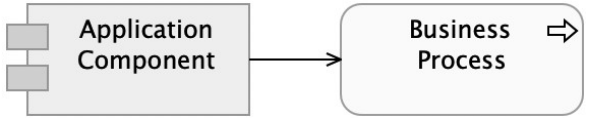
ArchiMate View ``CS Pharmacy’’ presenting communication of two information systems supporting medication supply and distribution: CGM Pharmacy and Chipsoft HiX

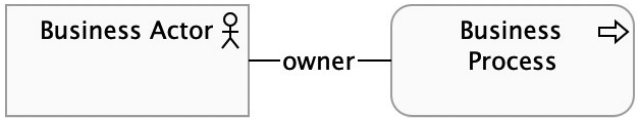


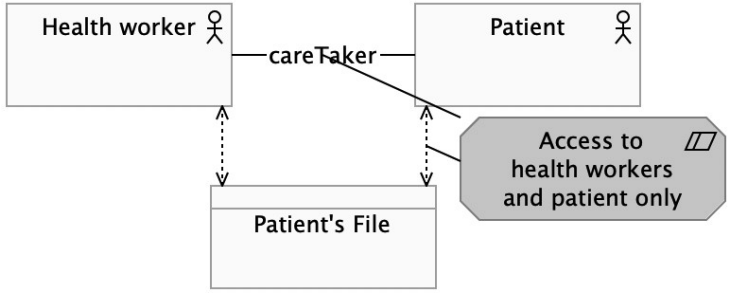
We have visualised and formalised 10 policies as constraints

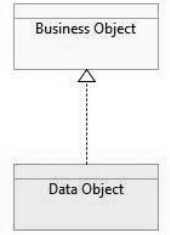












Most of constraints are reusable. Other may be reusable with renaming.

Results of the method testing

Policy	Violations in two models
1. Only one information system is in use for each functionality.	1
2. Unambiguous and one-time recording of data (and multiple use).	0
3. Each business process should be realized by at least one application system.	12
4. A Business process has precisely one owner.	18
5. Healthcare providers and patients work with one shared file.	1
6. A data or data group uses one or more business objects.	0
7. The continuity of critical systems of the Medical Center is guaranteed.	12
8. Use of central applications is mandatory.	12
9. The core of information provision is an Enterprise Data Warehouse (EDW).	1
10. Every data and data type has someone responsible.	7

Contribution : A Method for constraint formalisation om the ArchiMate metamodel and a Rule Engine: ArchiChecker

- A **method for formalising modelling conventions** as business rules **using the ArchiMate metamodel**.
- A **rule engine ArchiChecker** for verifying of modelling conventions on enterprise models in ArchiMate.
- The results of testing of the method and the rule engine.
- A set of reusable constraints found as a by-product of the case study.

Pros and Cons of the presented method and the rule engine

Pros

- The method produces reusable formalised and visualised constraints. The architects begin to understand and agree on the meaning of each policy.
- The method initiates formulating the goals of the enterprise and the goals of policies and increases the understanding of the analysed enterprise models. Conventions appear in modelling process, but not noted.

Cons

- The method is labour-consuming because of agreements that have to be made between architects;
- It demands accuracy in formalising;

Joosten, S. M. M., & Roubtsova, E. E. (2022, April). Constraint formalization for automated assessment of enterprise models. In The 24th International Conference on Enterprise Information Systems (ICEIS2022) (pp. 430-441). SCITEPRESS-Science and Technology Publications, Lda..

Semantic Relations of Sub-Models in an Enterprise Model

Ella Roubtsova and Sefanja Severin

*We proposed to use **sub-models as constraints** to each other and direct the design by relations of sub-models.*

- *The goal sub-model is a constraint for the business object sub-model, roles, applications and other concepts sub-models.*
- *The goal sub-model is a constraint for the business process sub-model.*
- *The business object sub-model is a constraint for the business process sub-model.*

A new method for modelling of a consistent Enterprise Model

1. Design the Goal sub-model. Refine the Goal sub-model to the level when the elements of Concepts (countable, comparable) and state changes can be identified.
2. Analyse of the Goal model (Lexical analysis and Refinement analysis).
3. Derive the Concept sub-model aligned with the Goal sub-model.
4. Derive the Business Process model aligned with the Goal and Concept sub-models.

Adaptation of ArchiMate for our method

- The **refinement relation** is expressed with the **realization relation.**
- The element "**event**" in ArchiMate is defined differently than in many other notations. An event in ArchiMate is actually a **state change** [ArchiMate Specification 3.0, sec.8.3.4].

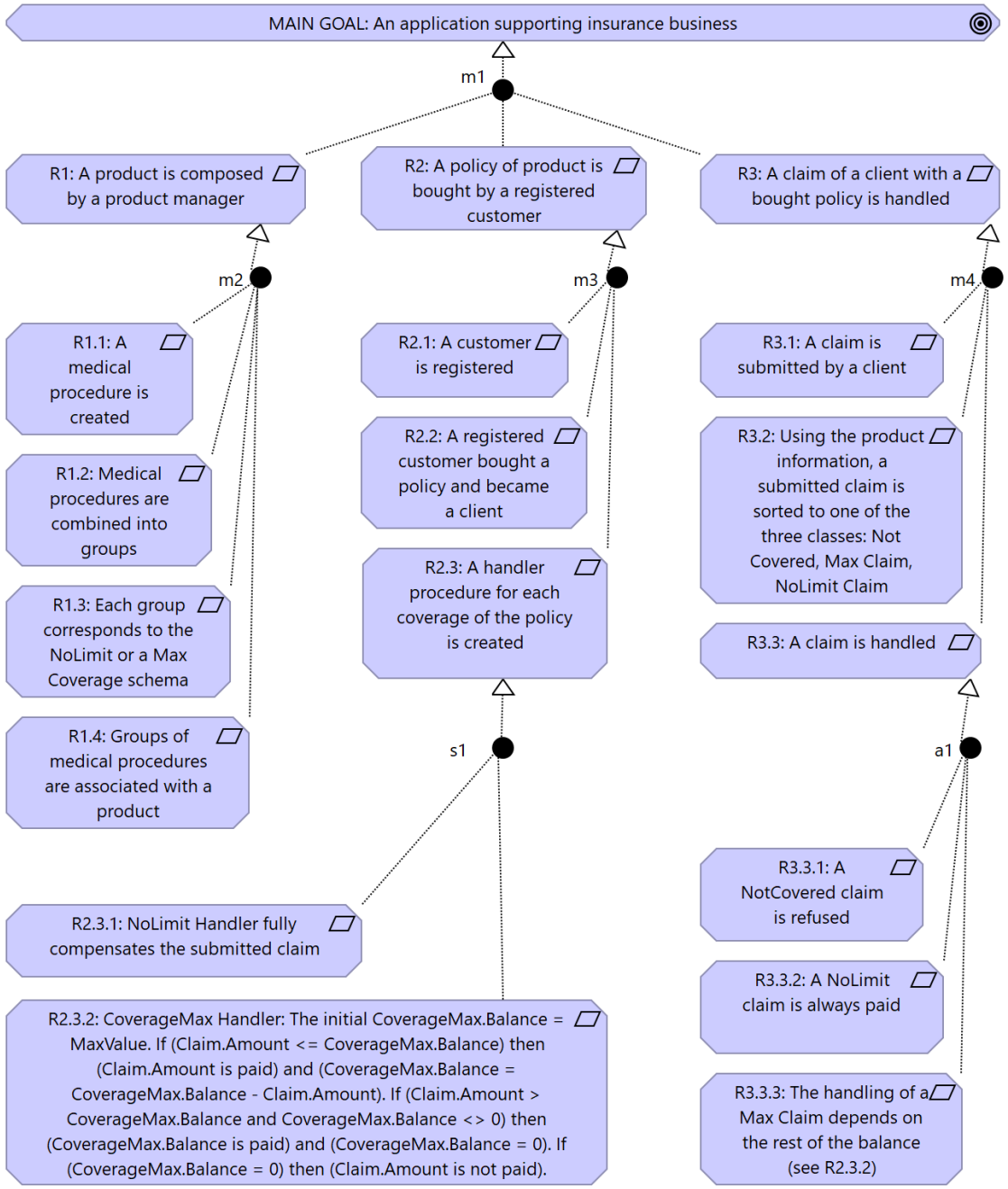
Lexical analysis of elements of the Goal Sub-Model to define *Concepts* and *Relations of Concepts*

Each element $g \in G$ is a sentence in natural language presenting a state or a partial state of the modelled system.

(1) Nouns can become objects (concepts), ***Product, Policy, Claim, etc.***

(2) Prepositions and verbs can become relations between objects (concepts) ***is-composed-by, is-handled, etc.***

(3) Plurals or keywords such as 'each', 'one of', etc. can become specialization relationships



Analysis of refinements to define *Relations of Process states*

A Goal sub–model = (G,Rm,Rs,Ra).

1) ***Rm*** *is a set of* ***milestone****-type refinements*

rmn = (gmn, (Gmn,Omn)),

Omn={(gmn1 , gmn2), …,(gmnk–1 , gmk)} |= gmn.

state-sequence-of (Business Object, states)

2) ***Rs*** is a set of **domain separation** refinement relations: ***rsn = (gsn, (Gsn,Usn)),***

Usn = (gsn1 ∪… ∪gsnk) |= gsn.

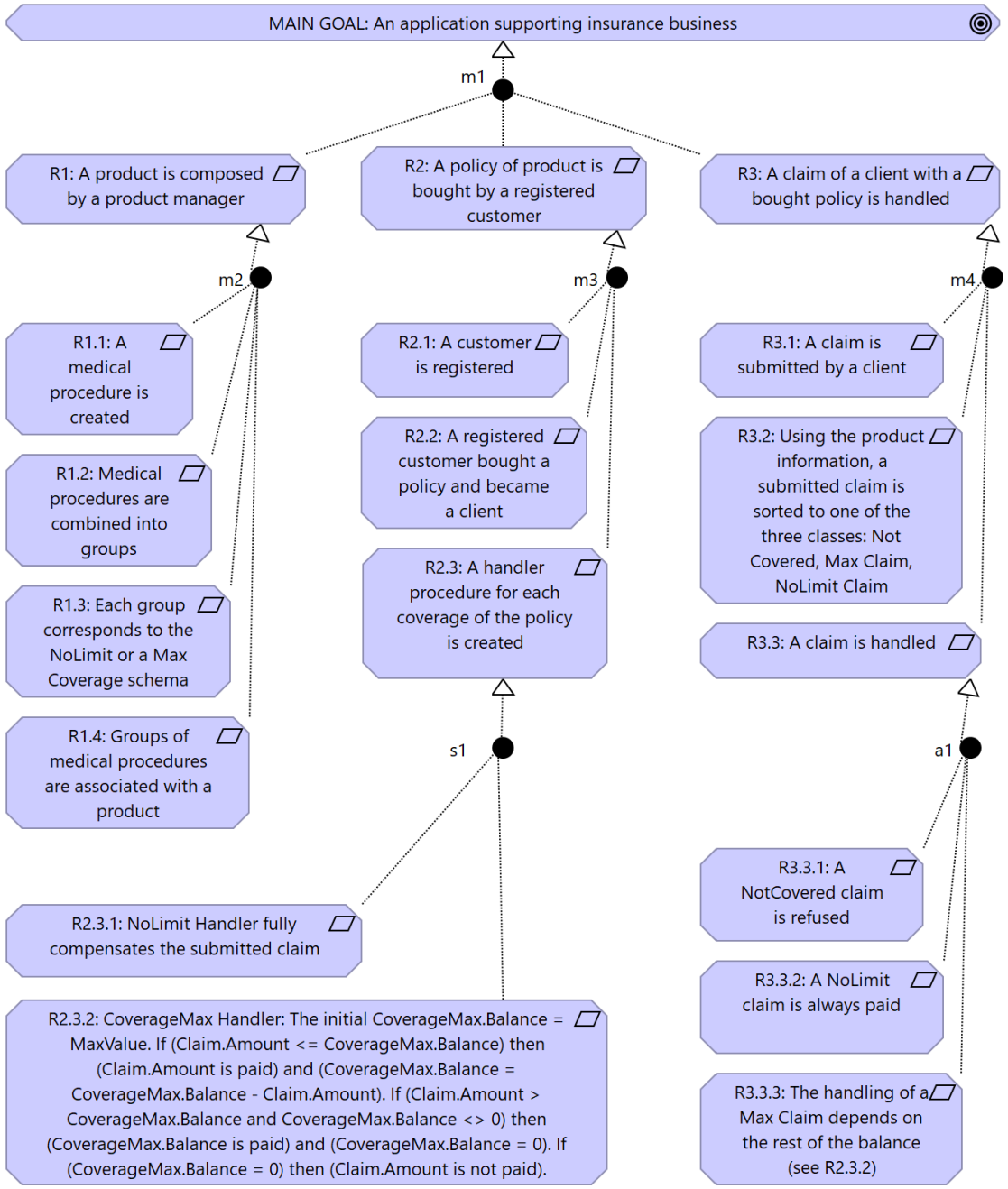
state-of(Business Object, state)

3) ***Ra*** is a set of **alternatives** ***ran = (gan, (Gan,Aan)),***

Gan = {gank , …, , gank}, n, k ∈N,

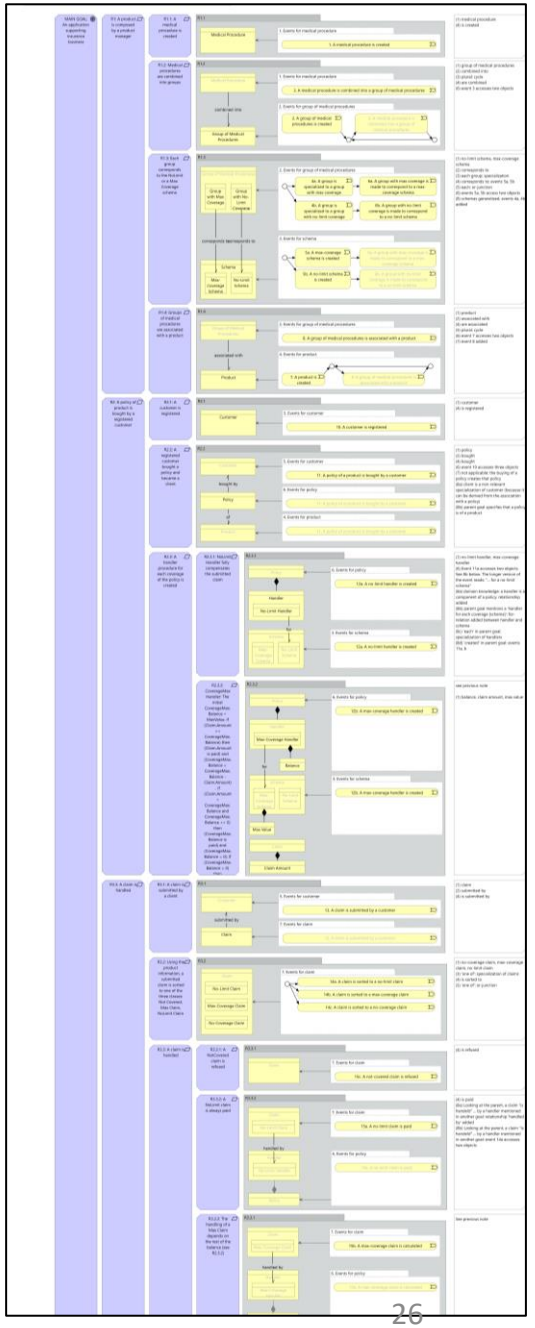
(gan1 ∩ …,∩gank) = ∅.

Aan = ∀i = 1, …k : (gani |= gan).

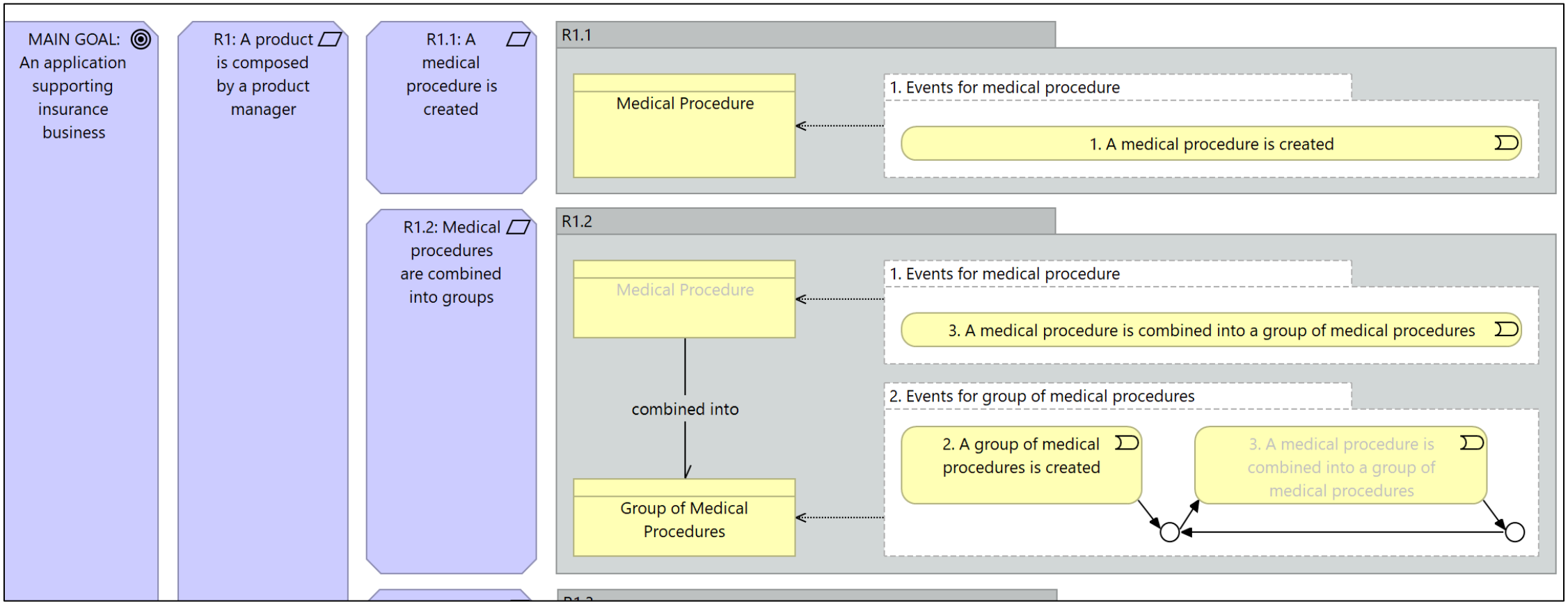


A developer view in Archi

The elements of the goal model are placed vertically and undergo the lexical analysis and depicted:
(1) Nouns can become objects (concepts),
(2) Prepositions and verbs can become relations between objects (concepts)
(3) Plurals or keywords such as 'each', 'one of', etc.
can become specialization relationships
(4) Plurals or keywords such as 'each', 'one of', etc. can become junctions
(5) The refinement relations of the goal model are used to capture sequences of events and alternatives in the process model.
(6) Events access all the objects that they mention



A fragment of the developer view in ArchiMate



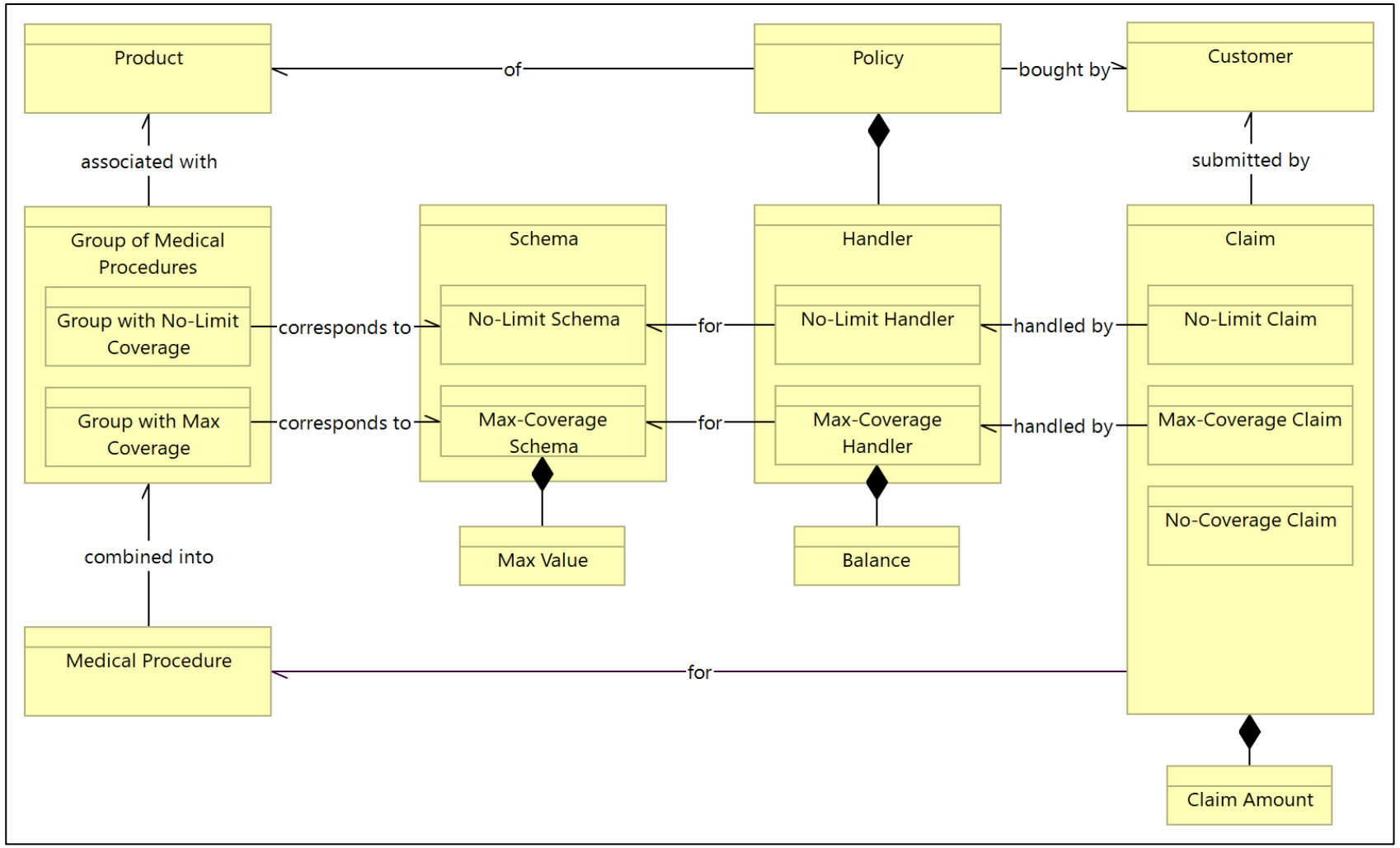
The grey concept has been already filled out and taken from the internal model.

Select business objects and derive the Concept sub-model constrained by the Goal model

Concept sub–model = (C,R)

Constraint:
Nouns(G) ⊆ C
Relations of Concepts(G) ⊆ R

Designers of the Concept sub-model often add some new concepts-attributes and their relations with other concepts. In our case: *Policy is composed by handlers. Also a Claim got an extra relation with a Medical Procedure.*



Select events and derive the business process model constrained by the Goal and Concept sub-models

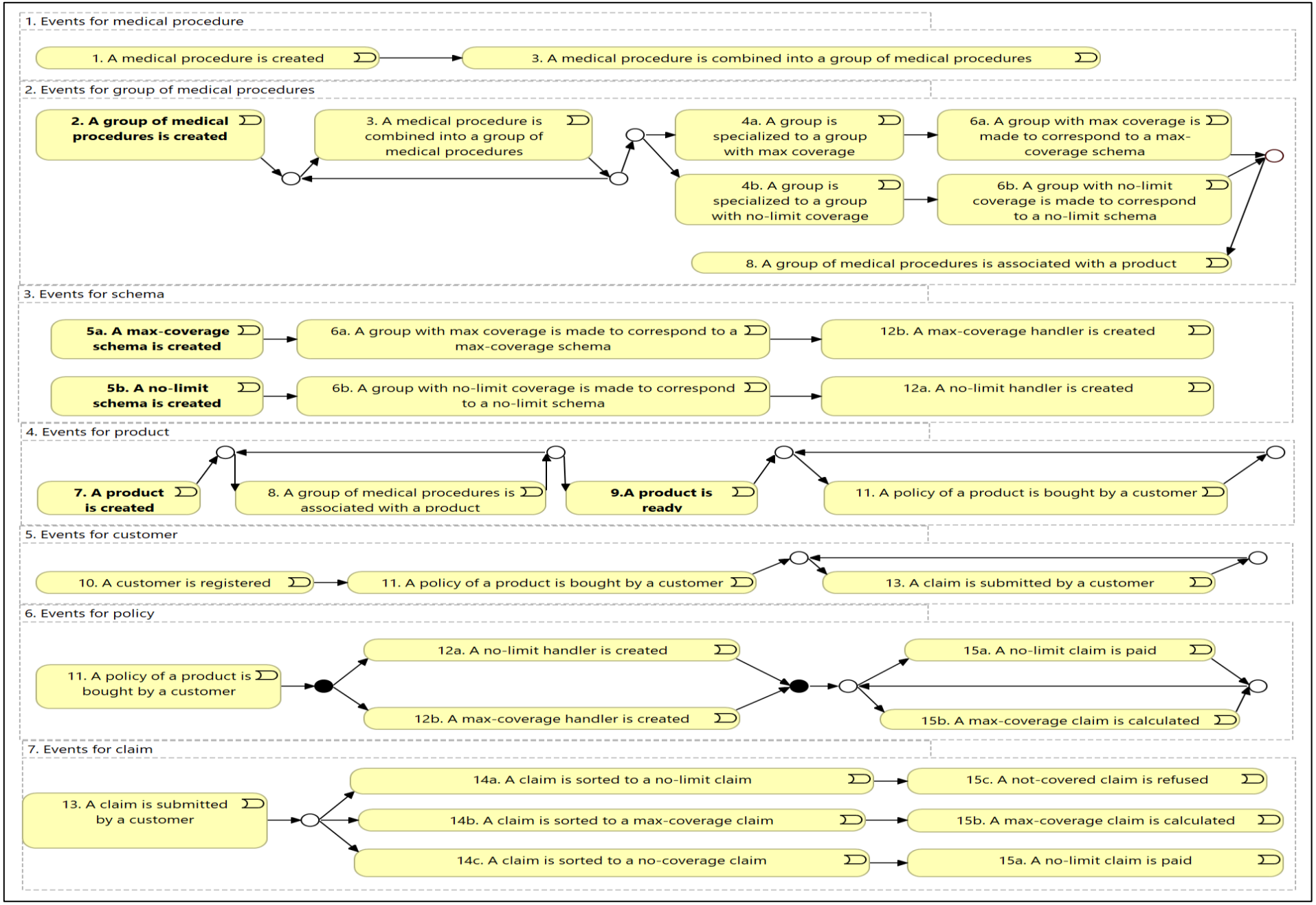
Business Process sub–model = {Behavior$_k$ | k = 1, ...,K, K ∈N}.

A behavior is a tuple ***Behavior = (S, T).***

Constraints:

- Each noun n ∈ Nouns(G), being a concept c ∈ C of the Concept sub-model, has a corresponding Behavior, except if the concept composes or specializes another one.
- Each goal g ∈ G of a Goal sub-model has a corresponding state s ∈ S in the Process sub-model, named after this goal.
- Each pair of goals of an milestone refinement the Goal sub-model corresponds to a transition of states in the Process sub-model.
- Each alternative refinement corresponds to an OR-split of states in the Process sub-model.
- Each sub-domain refinement corresponds to an AND-split of states in the Process sub-model

Business Process sub−model



Result of the Method Application

- The Goal sub-model is the leading constraint for the Concept and Business Process sub-models.
- The internal ArchiMate model is filled out with unique elements and relations.
- The internal model can be used to generate sub-models and views.

Roubtsova, E., & Severin, S. (2022). Semantic Relations of Sub-models in an Enterprise Model. In *International Symposium on Business Modeling and Software Design* (pp. 104-121). Springer, Cham.

Semantic Enterprise Modeling: Enhancing ArchiMate with Semantic Modeling Conventions

S. Joosten, E. Roubtsova

We propose to include constraints (as small structural views and corresponding rules) into the process of enterprise modelling.
The constraints are

- formulated during modelling of views,
- checked,
- used for model corrections and
- become model features

We are working of Semantic Enterprise Modelling right now.
I am happy to answer your questions.