

Constraint Formalization for Automated Assessment of Enterprise Models

Stef Joosten, Ella Roubtsova and El Makki Haddouchi

Open Universiteit
www.ou.nl



Outline

- Motivation
- 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 and Future work

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.

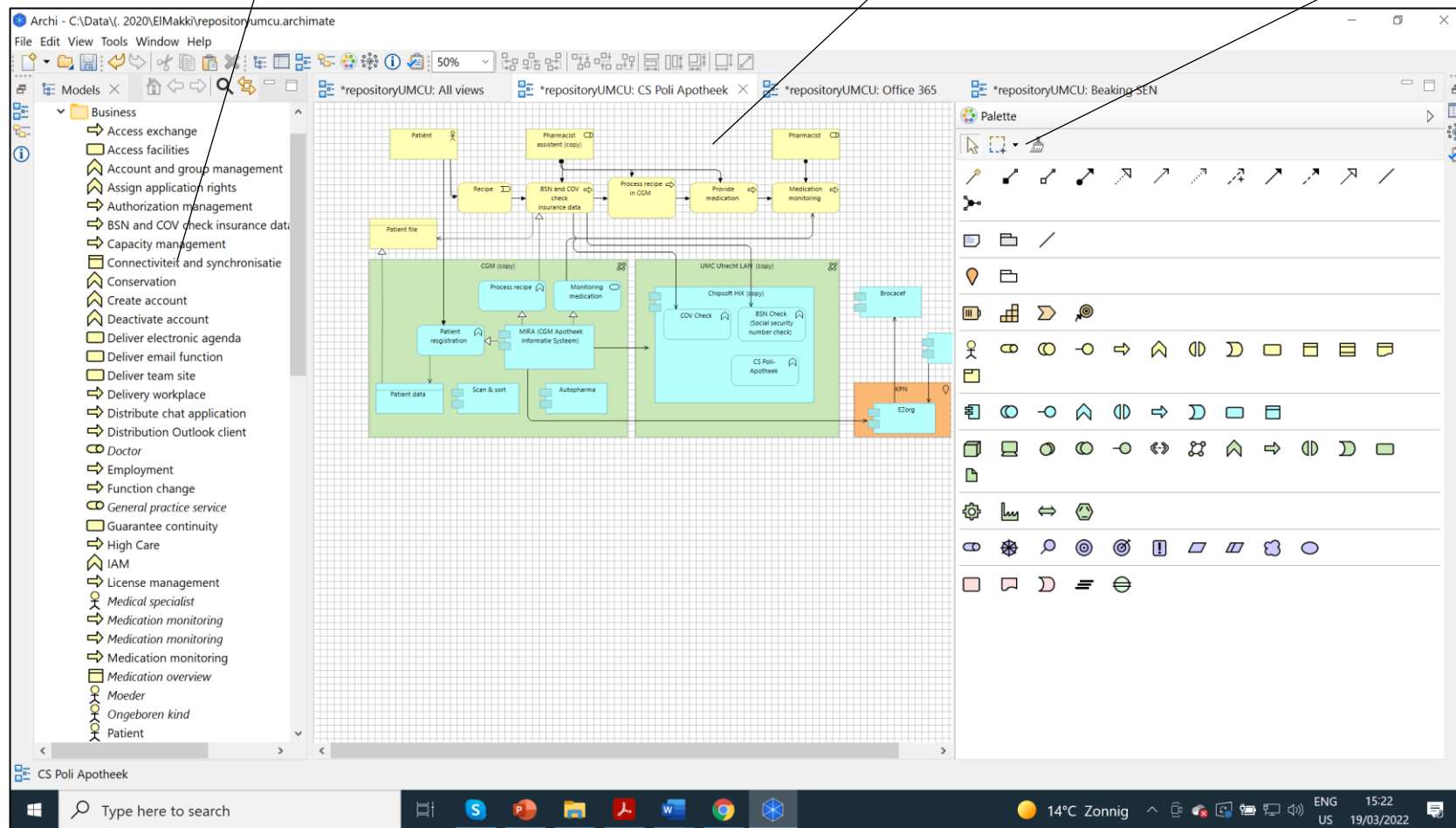
In literature:

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.

ArchiMate

An internal ArchiMate model

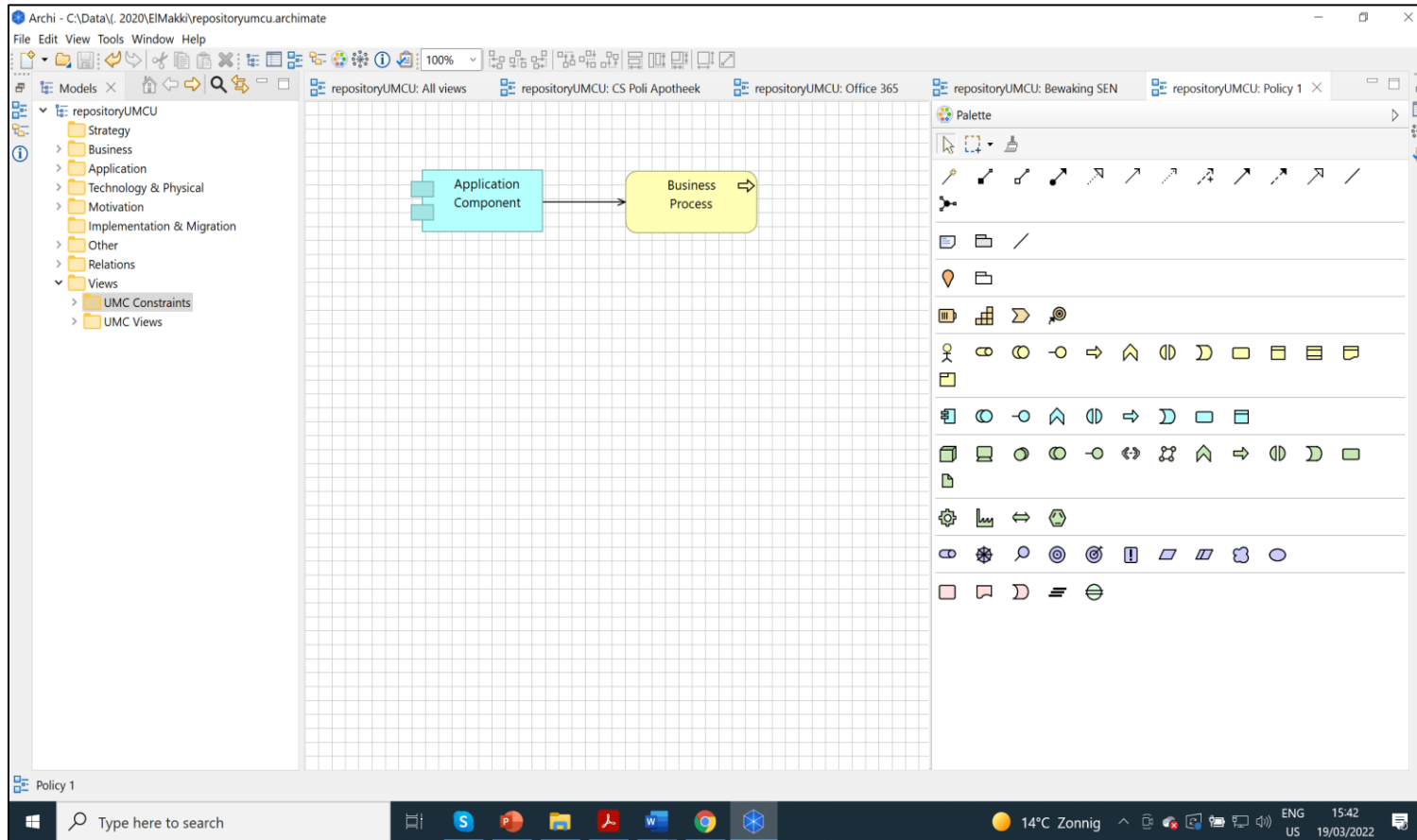


In each single **view**, ArchiMate shows a set of **elements** (boxes) and **relations** (lines) has been picked from a palette: a limited set from elements of the metamodel.

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

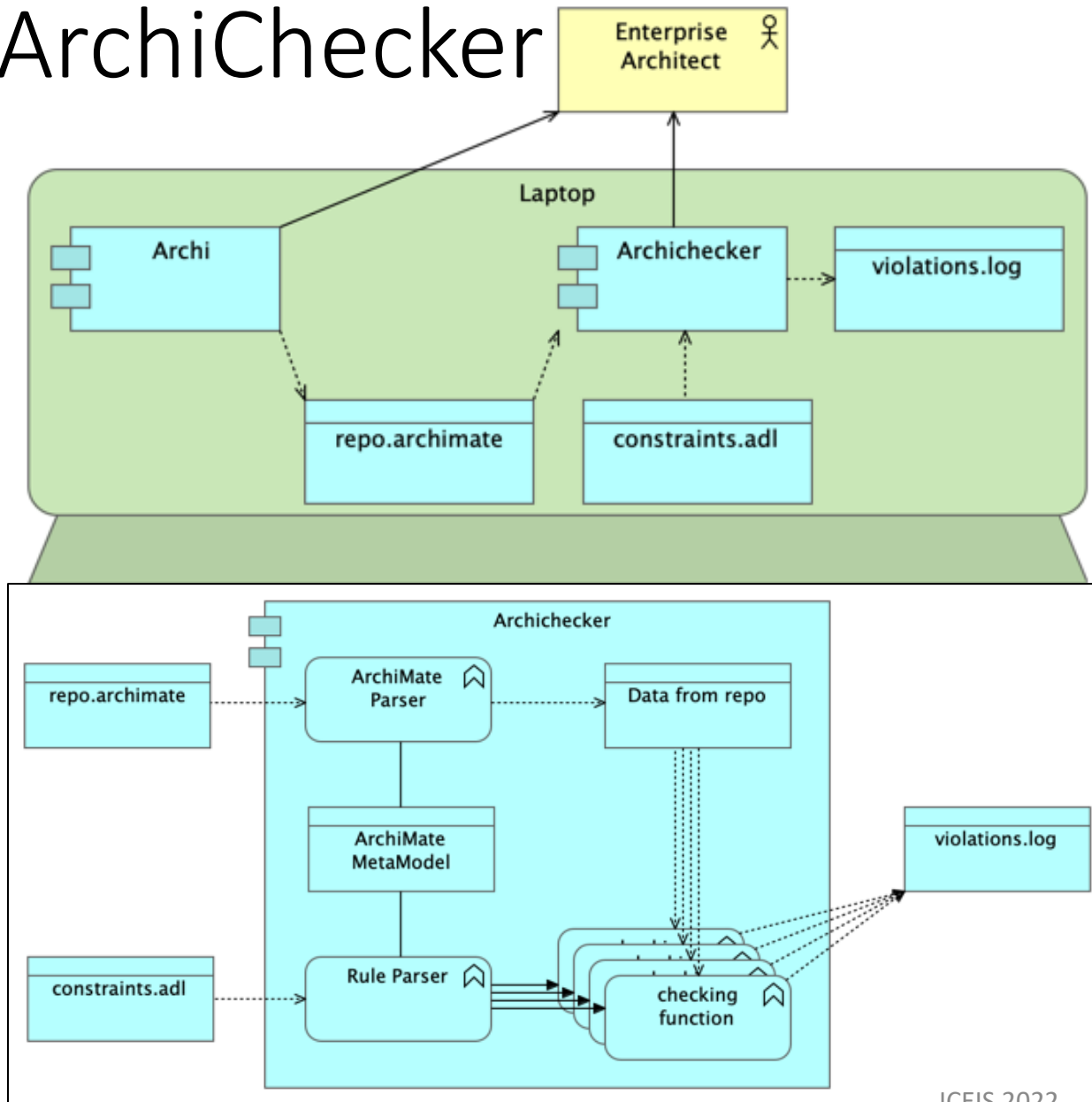
Different views can use the same elements. An internal key identifies an element.

We propose to formalise constraints using the metamodel elements of ArchiMate



- 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 then formalised as a rule in the Ampersand language to be used in our tool called ArchiChecker.

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.

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
```


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”.

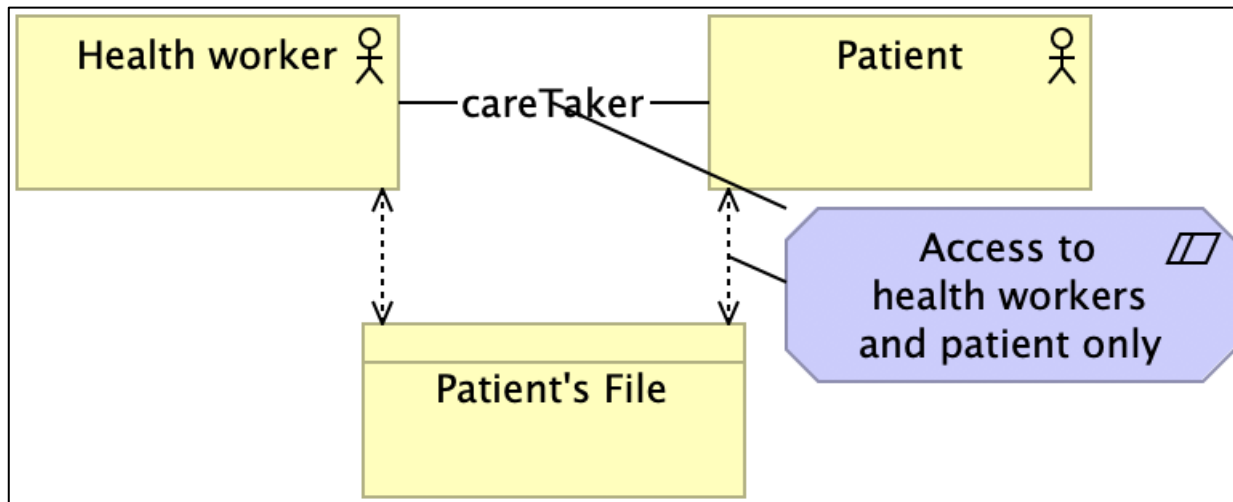
Each patient must have access to his or her “Patient File”.

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.

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

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]~

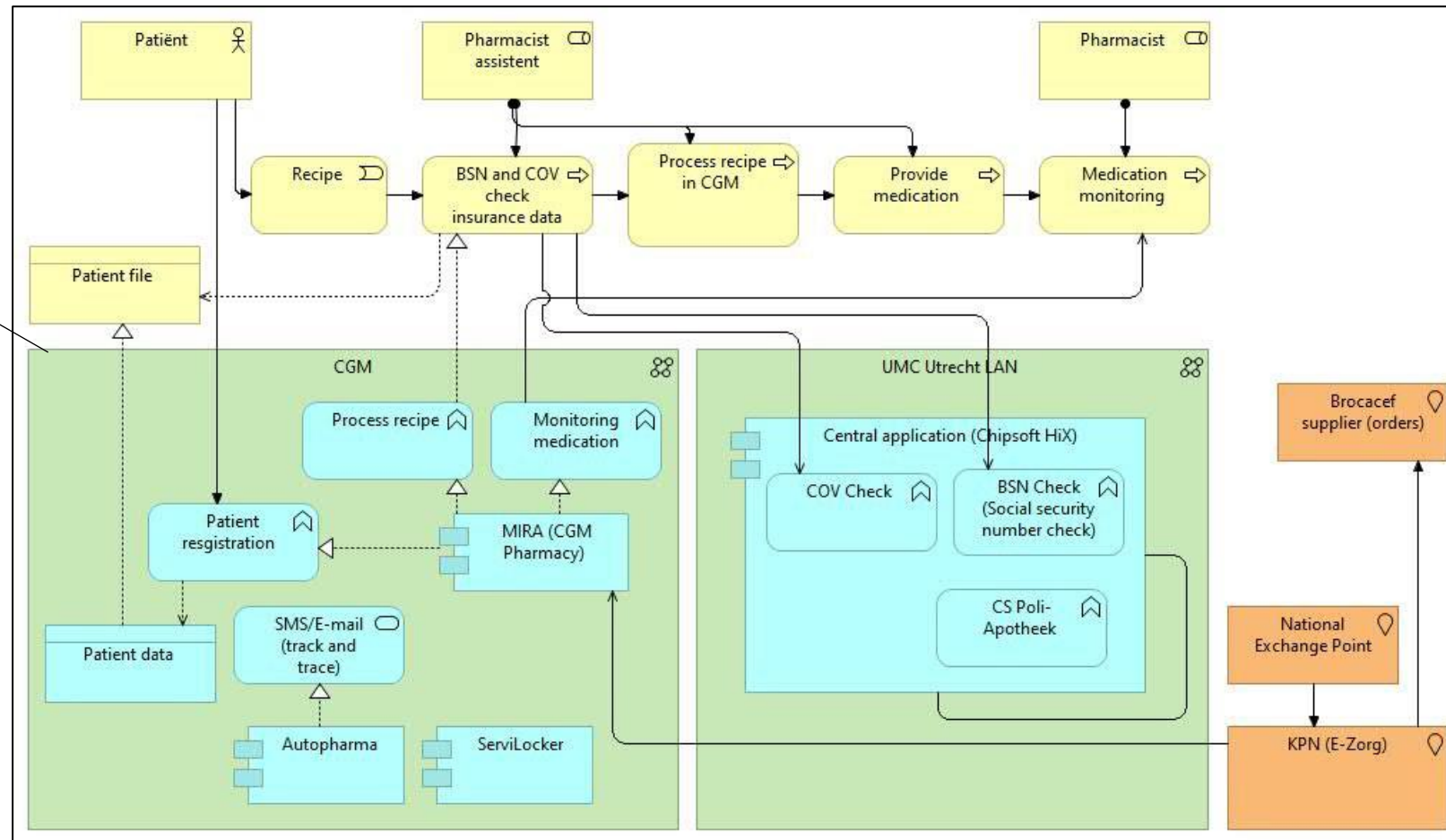
4. Run the ArchiChecker with an ArchiMate Model.

Tool: PatientFile (Business Object) 'Patient file' is not accessed by a caretaker/Patient (Health worker).

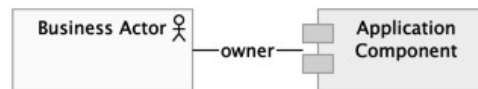
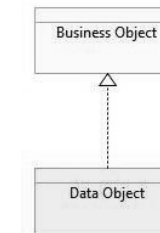
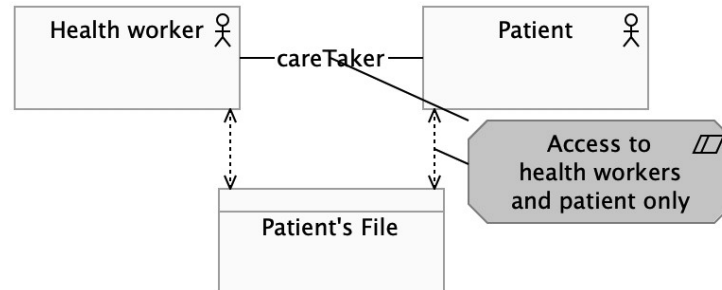
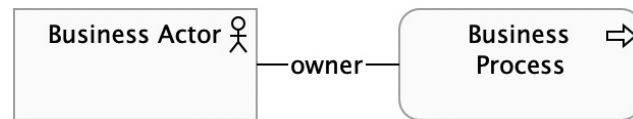
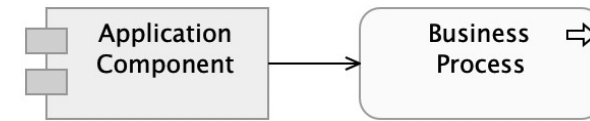
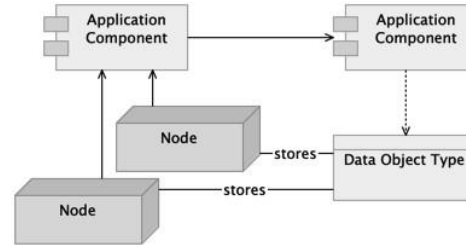
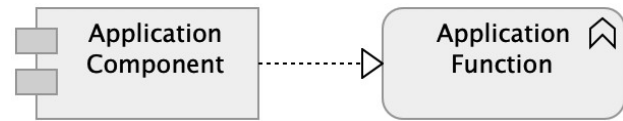
5. Analyse violations.

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 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

Cons

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

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.

In Future work

The method and the rule engine
**can become a part of an approach for consistent modelling in
ArchiMate**
that includes goal models refined to requirements and **associated
with policies and constraints** within each enterprise model.

Thank you for your attention and I look forward to your questions.