Semantic Relations of Sub-models in an Enterprise Model

Ella Roubtsova(✉) and Sefanja Severin

Open University, Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands
ella.roubtsova@ou.nl

Abstract. Enterprise modeling is a set of tools, methods and practices for an aligned development of business, functional, organizational and technical aspects of an enterprise. Therefore, an enterprise model is always a set of sub-models of different semantics. In order to form a consistent enterprise model, its sub-models should be aligned to each other. The practice of modeling shows the difficulties in design of an aligned set of sub-models of an enterprise model. In this paper we present a review of enterprise modeling approaches aiming to find the reasons of difficulties. Our review shows that enterprise modeling approaches not sufficiently use the semantic relations of sub-models for building an enterprise model. This paper identifies and formalizes the semantic relations of sub-models and suggests to use them as constraints directing the design of aligned sub-models. The constraints imposed by sub-models of the enterprise model to each other are illustrated with a case study in ArchiMate.

Keywords: Enterprise modeling · Sub-models · Goal sub-model · Concept sub-model · Process sub-model · Semantic relations of sub-models · Model consistency · Sub-models alignment · ArchiMate

1 Introduction

Enterprise modeling 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" [23].

Enterprise modeling (EM) approaches present these different parts of an enterprise as sub-models. There are sub-models that use the same semantics and notation of concepts and relations (boxes and arrows). Such sub-models separate sub-domains of the modeled enterprise (resources, technical components, actors, business concepts). There are sub-models that present dynamics with a process semantics. They use processes and flow relations, states, events and triggering relations. Other sub-models present the motivation of the modeled enterprise and use goals and requirements as elements and their refinement relations.

The sub-models in a consistent enterprise model should be related or aligned to each other [8]. The practice shows how difficult it is to align sub-models into a consistent enterprise model. Many authors, for example, [12,17], emphasize

B. Shishkov (Ed.): BMSD 2022, LNBIP 453, pp. 104–121, 2022.
https://doi.org/10.1007/978-3-031-11510-3_7

the critical alignment of enterprise sub-models to the goal sub-model. Kaisler at al. [8] identify the critical problems of alignment, such as using in sub-models the non-matching levels of abstraction, limited tool support for business process alignment with other sub-models and for managing the integrated enterprise life cycle. All these problems cannot be solved without methodological support for designing aligned sub-models.

Section 2 of this paper presents the result of our review of enterprise modeling approaches that shows some history of accumulating and aligning a set of sub-models in an enterprise model. By evaluating enterprise modeling approaches, we have recognized that the relations of sub-models can be defined only at some level of detail and these relations may direct the modeling to a consistent enterprise model. We define an aligned or consistent[1] enterprise model as follows:

An aligned or consistent enterprise model is a set of sub-models of different semantics, where sub-models restrict or constrain each other and the restrictions and constraints are defined in terms of semantics of sub-models.

Section 3 of this paper shows a case study of enterprise modeling, directed with semantic relations of sub-models.

Section 4 generalizes the semantic relations of sub-models in an enterprise model.

Section 5 concludes the paper and presents ideas for future work.

2 Related Work. Attempts to Address Consistency of an Enterprise Model

The notion of consistency of an enterprise model, meaning aligning of all its sub-models to each other, faces difficulties caused by the need to include a sub-model representing requirements for the model and by the different semantics of sub-models. Let us show how it was recognized and handled historically.

The 4+1 approach has been designed "for describing the architecture of software-intensive systems" [10]. The *+1* in the *4+1* approach is an attempt to include a sub-model of requirements, into a *4+1* enterprise model and make all sub-models aligned with this sub-model and with each other.

The *+1* sub-model is a set of scenarios presented in the UML Use Case Diagram. "Scenarios are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype" [10]. Each scenario corresponds to a set of requirements, often combined as a set of milestones or presenting separation of domains. Speaking about the order of designing of sub-models, we see that the *+1* sub-model should be the first, however, the *+1* is never seen as a complete presentation of requirements. The architectural elements identified in the *+1* are depicted in other sub-models presented using *4* semantics: (1) Class Diagrams, (2) State Machines or Protocol State Machines, (3) Sequence or Communication or Activity Diagrams, (4) Component and Package Diagrams [5,9,16]. The relations of sub-models are defined as rules on *4* semantics. The *+1* sub-model is

[1] Both terms are used, even by the same group of authors, for example [8].

excluded. Egyed [3] reports that there is "a division between those who compare design models directly and those who transform design models into some intermediate, usually formal, representation to compare there". The comparison of sub-models directly requires consistency rules, for example, "Name of a message must match an operation in receiver's class"; "Calling direction of a message must match an association". There is an analyzer for "instant error feedback that profiles consistency rules" [3]. The analyzer needs an external information to collect semantic rules for consistency checks of sub-models.

The need of a sub-model representing requirements in an enterprise model has initiated the attempts to combine goal models with UML models. *The KAOS approach* [19] has introduced a practical goal modeling with refinement of goals and its application for system modeling. However, Letier et al. [11] have identified a semantic difference between the goal models and the UML behavior semantics (State Machines, Sequence or Communication or Activity Diagrams). The goal models define quiescent states that should be achieved by the system. A quiescent state means that the system cannot change this state by itself; the system preserves this state indefinitely long time until an external event. In a UML behavior semantics, the changes of the system state may be caused by events taken from the bags or queues of earlier arrived events [21].

In parallel, the enterprise modeling approaches have tried to include the goal sub-model into an enterprise model and choose the behavior semantics that can be aligned with the semantics of goal models.

The 4EM approach has defined an Enterprise Model as a tuple of six sub-models.

- Goal sub-model uses the semantics of [19]. An element (box) is a goal, business rule or requirement. A relation (arrow) is a refinement relation of a goal to a sub-goal or a requirement, a business rule to requirement.
- Concept Model, Actors and Resource Model and Technical Components Model types use the semantics of object models. An element (box) presents a business object or a physical resource. A relation (arrow) can be a binary, an operation or a specialization or an aggregation relation [23, p. 112].
- Business Process Model uses a process semantics. There are two types of elements (boxes): information sets and processes. A relation(arrow) between a process and an information set or between an information (material) set and a process presents a control flow. *AND* and *OR* join and split connectors are used to present alternatives and cycles [23, p. 121].

The relations of sub-models in an enterprise model are abstractly described in [23, p. 78] with a diagram. For example, a Goal Model "uses" and "is related" to the Concept Model, "motivates" and "requires" Business Process Model and Technical Components Model [23, p. 78]. The restrictions that sub-models impose on each other in an enterprise model are expressed as "a number of consistency rules" [23, p. 211]. For example, the rule: *"Every information set or material set in a Business Process Model must be expressed using concepts of the Concept Model,"* relates concepts in the Concept Model and information sets in the Business Process Model.

To design a consistent enterprise model, the 4EM approach suggests the modeler to (1) integrate the sub-models, so that "the inter-model links should establish a clear line of reasoning" [23, p. 211]; (2) visually check the consistency rules; (3) identify inconsistencies and make iterations to improve the model. The practice of application of the 4EM shows that the integration can be applied in small cases and usually results in a model having boxes and arrows with different meaning, what makes the model difficult to observe and understand. The examples can be seen in [23, p. 213, 214]. The visual checks of consistency rules are very useful for small models, however, the consistency rules just partially define the semantic relations of sub-models and do not direct the design of sub-models.

The ExtREME approach [20,22] presents an enterprise model as a set of a Goal, a Concept, and an executable Protocol sub-models. ExtREME exploits the similarity of semantics of a goal, being a quiescent state of the modeled system, in a Goal sub-model and a quiescent state in an executable Protocol sub-model [13].

The relations of sub-models are used to direct the design of the enterprise model from a Goal sub-model to a Protocol sub-model. The refinement patterns of the Goal sub-model identify the states of life cycles of concepts.

A Protocol sub-model is a set of *protocol machines*. Each protocol machine presents a life cycle of a concept. The elements of a protocol machine are the following: a local structure (to present states of the life cycle of a concept) and a set of transition relations. A transition from state "a" to another state "b" is labeled with an event "e", that can happen in state "a".

An event "e" is a happening in environment. An event is presented with a data structure. The data of an event and the local structure of the protocol machine are used to check if the event can happen and to update the local structure of the protocol machine after the transition.

The instances of life cycles of concepts (instances of protocol machines) are synchronized using the CSP-parallel composition [13,14]: if an event is recognized by several instances protocol machines, it can happen only if all these instances are in the state where this event can happen. A business process is a set of sequences of synchronized executions of instances of protocol machines. The consistency of sub-models in ExTREME is achieved by executing the protocol sub-model and testing all requirements presented in the goal sub-model.

The ArchiMate approach [24] has been designed as a foundation for a consistent enterprise model. ArchiMate provides a "structure or a storage for an internal model of an enterprise", that includes sets of elements and relations. If an internal model of an enterprise model has been filled in with unique (non-duplicating) elements and relations, then this model can be used as a source for designing views being consistent with the internal model.

The practice of enterprise modeling shows that it is difficult to fill the internal model in with the unique (not duplicating) elements and relations. The reasons of the difficulties are caused by the team modeling and human factors: (1) modelers draw different sub-models and even their parts (views) as teams and the internal model is filled in from these drawings; (2) the sub-models and the order of

their building, and the naming of elements and relations are chosen by modelers (often different team members). Modelers can make typos, duplicate names, miss elements or relations.

The structure of an internal ArchiMate model of an enterprise model is the following:

- 'Two main types of elements: "structure elements" and "behavior elements", " inspired by natural language, where a sentence has a subject (active structure), a verb (behavior), and an object (passive structure) [24, sec. 4].
- The element "event" in ArchiMate is defined differently than in many other notations [4,6,14]. An event in ArchiMate is actually a state: "*A business event represents an organizational state change*" [24, sec. 8.3.4].
- There are motivation elements. "A motivation element represents the context of or a reason behind the architecture of an enterprise" [24, sec. 4]. Elements "Goal" and "Requirement" are among motivation elements.
- There is a set of relations: access, composition, flow, aggregation, assignment, influence, association, realization, specialization, triggering, serving [24, sec. 5.6]. There is a table [24, B.5] that specifies what kinds of relations are allowed for what kinds of elements. The semantics of most relations corresponds to the Concept sub-model semantics. Some relations are applicable in Process sub-models.
- The refinement relation used in Goal sub-models does not exist in ArchiMate.

Providing an internal model structure for an enterprise model, the ArchiMate does not provide any systematic way or method for collecting elements and relations for the internal model. The internal model is often filled in with elements and relations from sub-models or their partial views drawn by modelers on the spur of the moment. As a result, the internal model may contain double copies of elements, not-related elements, elements that are not related to goals and all typos and mistakes that a modeler can make, trying to capture a case description. Although there is a tool support to mark suspicious model elements [1], but the decision about any model correction is delegated to the modeler and the correction is often postponed and forgotten.

Our review of enterprise modeling approaches shows that the semantic relations of sub-models of an enterprise model are recognized in all approaches, but they are not used to direct the design of sub-models. The semantic relations of sub-models, expressed in 4EM, KAOS and ExtREME, need generalization to direct the design of sub-models aligned to each other in a consistent enterprise model.

3 Semantic Analysis of Sub-models

In order to identify the semantic relations of sub-models, we use a combination of research methods, namely, a case study and a semantic analysis of sub-models within the case study.

Our case study is an executable enterprise model of an insurance business taken from [22]. This enterprise model includes sub-models that represent all three semantics: goals, concepts and behaviors. The sub-models have been already made consistent in ExREME by executing and testing techniques.

By redrawing of sub-models of this enterprise model in ArchiMate, we use the internal ArchiMate model, i.e. elements and relations (Sect. 2). The initial elements are (1) goals and requirements. The initial relations of goals and requirements are the goal refinement patterns. We also use the ArchiMate elements of two categories: (2) objects (concepts) and (3) events.

In majority of enterprise modeling notations, the terms "concept" is used to present enterprise objects at different levels of abstraction.[2] Therefore, we use the term "concept" instead of "object".

Concepts (objects) can be identified by the lexical and semantic analysis of goals and requirements. A concept can be a business object, a role, an technical component named as a noun in a sentence presenting a goal. A relation of concepts is a named pair of concepts. A name of relation is identified as a verb or a preposition in a sentence presenting a goal.

An event (a behavior element) in ArchiMate is defined differently from other notations. "*A business event represents an organizational state change*" [24, sec. 8.3.4]. Because there is no any data structure associated with an event in ArchiMate, a name of an event is a goal-sentence representing a state in a life cycle of a concept. Two events can be related with a triggering relation. "The triggering relationship is used to model the temporal or causal precedence of behavior elements in a process." [24, sec. 5.3.1].

A sequence of events is identified using a milestone refinement pattern of a goal. An alternative refinement pattern of a goal corresponds to an alternative ArchiMate events. A cycle of events can be identified by lexical analysis of a goal, when it expresses that the life cycle of a concept needs a set of instances of other concepts.

In terms of these elements and relations of the ArchiMate internal model, we define the sub-models and the semantic relations of sub-models.

3.1 Relations of Concepts and Relations of Process States Identified in the Goal Sub-model

In goal-oriented approaches, the refinement relation is used between goals, subgoals, requirements and constraints. The ArchiMate does not specify the refinement relation. The same way as [18], we use the realization relation to present refinement. The interpretation of a realization relationship is that the whole or part of the source element realizes the whole of the target element [24, sec. 5.1.5.]. The realization relation can express the sufficient condition of refinement relation. We have modified the Archi-tool and made realization relation allowed for all pairs of motivation elements.

[2] In ArchiMate, both elements and relations are concepts [24, sec. 2.8].

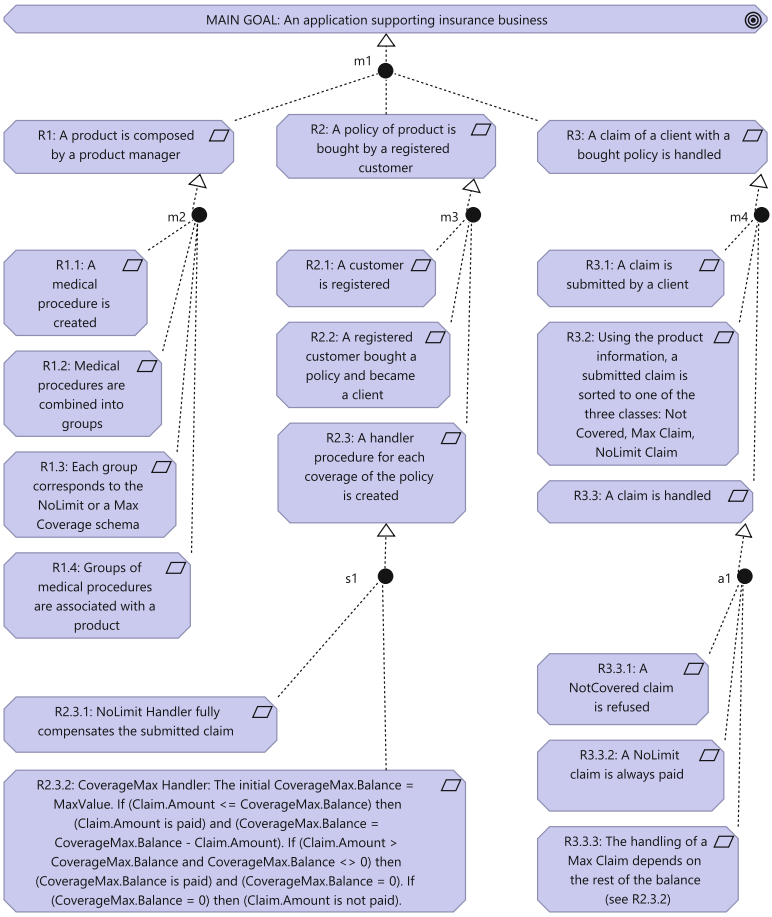

Fig. 1. Goal sub-model

The Goal sub-model of the case is presented in Fig. 1. Figures 2, 3, 4, 5 present the Relations of Concepts and Relations of Process States (ArchiMate events) identified in the Goal sub-model.

In Fig. 2, the reader can see that *"MAIN GOAL: Application supporting insurance business."* is refined to a tuple R1, R2 and R3. The refinement uses the mile-stone pattern, which means that R1, R2, R3 is a sequence of states of the concepts from the Main Goal: "Application" and "Insurance business". Each of these concepts represents the entire model. We have chosen the "Application" point of view, i.e. we focus on the data structure and relations and do not model actors.

R1, R2 and R3 are formulated as sentences in natural language. For example, *"R1. A product is composed by a product manager."* Using lexical analysis of this sentence, we can find a *is-composed(Product)* applied to any object of type

Requirements, Refinement	**Nouns**	**Verbs, prepositions**	**Relations of Concepts**	**Relations of Process States (ArchiMate Events)**
MAIN GOAL: An application supporting insurance business". **Refinement milestones-of** (MAIN GOAL, (R1,R2,R3))	Insurance business, Application			***state-sequence-of "***An application supporting insurance business". *named in requirements R1,R2,R3*
R1: A product is composed by a product manager	Product Product Manager (role)	is-composed-by	***is-composed-by***(Product, Product Manager)	**state-of**(Product,(Product is composed**))** **added by the modeler is-created (**Product)
R2: A policy of product is bought by a registered customer	Policy Product Customer	of is-bought-by	***of(***Policy, Product) ***is-bought-by***(Policy, Registered Customer)	***state-of*** *(Policy, (Policy is bought by a Registered Customer))*
R3: A claim of a client with a bought policy is handled	Claim	is-handled		***state-of*** *(Claim, (Claim is handled))*

Fig. 2. Relations from the refinement of the Main Goal

Requirements Refinement	**Nouns**	**Verbs**	**Relations of Concepts**	**Relations of Process States**
Refinement R1 **milestones-of**(R1, (R1.1, R1.2. R1.3, R1.4))				State sequences defined by synchronized business objects named in requirements.
R1.1: A medical procedure is created	Medical Procedure	is-created		***state-of***(Medical Procedure, (A Medical Procedure is created))
R1.2: Medical procedures are combined into groups	Group of Medical Procedures	is-combined	*Is-combined*(Medical Procedure, Group of Medical Procedures)	***state-sequence-of***(Medical Procedure, (A Medical Procedure is created; A Medical Procedure is combined with a Group of Medical procedures)).
R1.3: Each group corresponds to the NoLimit or a Max Coverage schema	NoLimit Schema MaxCoverage Schema Group with NoLimit Coverage Group with Max Coverage	corresponds -to	*correspond-to*(Group of Medical Procedures, NoLimit Schema) corresponds -to (Group of Medical Procedures, MaxCoverage Schema) corresponds-to(Group with NoLimit Coverage, NoLimit Schema) corresponds-to(Group with Max Coverage, MaxCoverage Schema)	***state-sequence-of*** (Group of Medical Procedures, (A Group of medical procedures is created; ***state-alternative(***Group of Medical Procedures, (A group corresponds to the NoLimit Schema; A group corresponds to the Max Coverage Schema); A Medical Procedure is combined with a Group of Medical procedures)). ***state-of***(NoLimit Schema, (NoLimit Schema is created)); ***state-of(***MaxCoverage Schema**,** **(**MaxCoverage Schema is created)); States "is created" for the Schemas and the Group are added by a modeler to enable specified events.
R1.4: Groups of medical procedures are associated with a product		are-associated	is-associated-with(Group of Medical Procedures, Product)	***state-sequence-of***(Product, (A Product is created; A Group of medical procedures is associated with a Product; A product is composed)); ***state-of(***Group of Medical Procedures, (A Group of medical procedures is associated with a Product)); State "A Product is created" is added by a modeler to enable specified events**.**

Fig. 3. Relations identified in the Goal sub-model by refinement of R1.

Requirements, Refinement	**Nouns**	**Verbs, prepositions**	**Relations of Concepts**	**Relations of Process States**
Refinement R2 **milestones-of** (R2, (R2.1, R2.2, R2.3))				State sequences defined by synchronized business objects named in requirements
R2.1: A customer is registered	Customer	is-registered		***state-of***(Customer, (A Customer is registered))
R2.2: A registered customer bought a policy and became a client		of is-bought	of(Policy, Product) is-bought-by(Product, Customer)	***state-of***(Policy,(A policy of a product is bought by a customer))
R2.3: A handler procedure for each coverage of the policy is created	Handler Claim NoLimit Handler CoverageMax Handler	 is-created is-created	is-created-for(NoLimit Handler, NoLimit Coverage Schema) is-created-for (CoverageMax Handler, Max Coverage Schema)	***state-of*** ((Policy, (NoLimit Handler is created; AND CoverageMax Handler is created))
R2.3.1 Figure 1	NoLimit Handler	is-created	is-created	
R2.3.2 Figure 1	***added by the modeler*** ***CoverageMax Balance*** ***Max Value*** ***Claim Amount***	*is-composed-of*	***added by the modeler:****is-composed-of(CoverageMax Handler,CoverageMax Balance);* *is-composed-of(MaxCoverage Schema,Max Value);* *is-composed-of(Claim,Claim Amount)*	

Fig. 4. Relations from the refinement of R2

Requirements, refinement	**Nouns**	**Verbs**	**Relations of Concepts**	**Relations of Process States**
Refinement R3: milestones-of(R3, (R3.1,R3.2, R3.3))				State sequences defined by synchronized business objects named in requirements
R3.1: A claim is submitted by a client	A Customer is a Client	is-submitted	is-submitted-by(Claim, Customer)	**state-of** (Claim, (A Claim is submitted by a customer)).
R3.2: Using the product information, a submitted claim is sorted to one of the three classes: Not Covered, Max Claim, NoLimit Claim	NotCovered Claim Max Claim NoLimit Claim	is-sorted	*specializes*(NotCovered Claim, Claim) *specializes*(Max Claim, Claim) *specializes*(NoLimit Claim, Claim) *one-of(Claim, Medical Procedure)* – ***added***	***state-alternative-of***(Claim, (Is sorted to NotCovered Claim OR Is sorted to NoLimit Claim OR Is sorted to MaxClaim))
R3.3: A claim is handled		is-handled		***state-of****(Claim, (A Claim of a Customer with the bought policy is handled))* ***state-sequence-of****(Claim, (*A Claim is submitted by a Customer; (Is sorted to NotCovered Claim OR Is sorted to NoLimit Claim OR Is sorted to MaxClaim)), A Claim of the registered customer with the bought policy is handled*)*
Refinement R3.3 **Alternative-of**(R3.3, (R3.3.1, R3.3.2, R3.3.3))				State alternatives defined by R3.3.1, R3.3.2, R3.3.3.
R3.3.1: A NotCovered claim is refused	*NoCoverage Handler – added for symmetry*	is-refused-by	is-refused-by(No Coverage Handler)	***state-alternative-of***(Claim, (is-refused))
R3.3.2: A NoLimit claim is always paid		is-paid-by	is-paid-by(NoLimit Handler)	***state-alternative-of***(Claim, (is-paid))
R3.3.3: The handling of a Max Claim depends on the rest of the balance		Is-calculated-by	is-calculated-by(CoverageMax Handler)	***state-alternative-of***(Claim, (is-calculated-using-MaxClaim-and-Balance))

Fig. 5. Relations from the refinement of R3

Product, so it defines a life cycle of a *Product*. Only one state of this life cycle is seen in R1: *state-of(Product,(Product is composed)*. The modeler adds the state *is-created(Product)*, because a product should exist to be composed. Analogically, R2 defines the life cycle of an object *Policy* and R3 defines the life cycle of an object *Claim* (Fig. 2).

Figure 3 shows the concepts identified by lexical analysis of R1.1, R1.2, R1.3, R1.4. Requirement *"R1.3. Each group corresponds to the NoLimit or MaxCoverage Schema"* defines a specialization relation of the object *Group of medical procedures* to *Group with Nolimit Coverage* and *Group with Max Coverage* and objects *NoLimit Schema* and *MaxCoverage Schema*. R1.3 also defines a state-alternative of a *Group of Medical Procedures*.

Figure 4 presents Relations of Concepts and Relations of Process States used for a *Policy* life cycle are shown in Fig. 4. A *Policy* object is created when it is bought.

Relations of Concepts and Relations of Process States used for a *Claim* life cycle are shown in Fig. 5. The alternative states of the business object *Claim* are caused by sorting each instance of object *Claim*. A Claim state *is-handled* may be one of the following states: *is-refused*, *is-paid* or *is-calculated-using-MaxClaim-and-Balance*.

Internal model of ArchiMate. "Elements" is an existing structure of the internal model presenting an enterprise model in ArchiMate. It can be filled in with "Nouns"(Objects). "Relations of Concepts" is an existing structure of the internal model, presenting an enterprise model, in ArchiMate. The elements of this structure are the results of lexical analysis of the Goal sub-model (Figs. 2, 3, 4, 5).

Because a goal represents a state of the modeled system and the ArchiMate defines an event as a state change, we use the names of goals to fill in events in the internal ArchiMate model. "Relations of Process States" (Figures 2, 3, 4, 5) are now filled in the "Relations of Concepts" in ArchiMate, but we visually check that each relation of process states is a triggering relation between a pair of ArchiMate events.

3.2 A Concept Sub-model Using Relations of Concepts

Using the "Relations of Concepts" identified in the Goal sub-model we build the Concept sub-model. Building of Concept sub-models is well guided by ArchiMate [24, sec. 4], so we have depicted the Concept sub-model aligned with the Goal sub-model in Fig. 6.

The constraints imposed by sub-models on each other is the basis of our method. Ideally, the set of "Relations of Concepts" of the Concept sub-model is equal to the set of "Relations of Concepts" identified from the Goal sub-model. The first version of the Concept model can be generated from the internal ArchiMate model.

In practice of enterprise modeling, there are two possible deviations from this constraint.

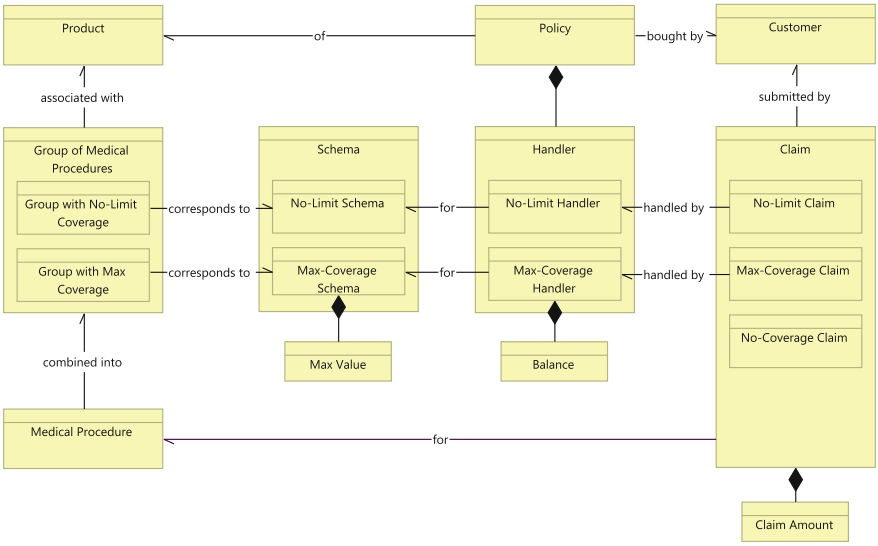

Fig. 6. Concept sub-model

1. A Concept sub-model may present a subset of "Relations of Concepts" identified from the Goal sub-model. It is a partial Concept sub-model called a view. Views are often used in enterprise modeling. A set of views may eventually cover all "Relations of Concepts" found in the Goal sub-model. Views can be generated from "Relations of Concepts" identified from the Goal sub-model.
2. A concept sub-model may contain extra relations added by designers. These extra relations have not been presented in the Goal sub-model.

Our case study illustrates the second deviation. Our Concept sub-model contains extra relations a *Policy* is composed by *handlers*. Also a *Claim* has an extra relation with a Medical Procedure.

So, in general, the set of "Relations of Concepts" identified from the Goal sub-model is a subset or an equal set of "Relations of Concepts" used in a Concept sub-model (or in a set of partial Concept sub-models).

3.3 A Process Sub-model Using Relations of Process States

Figure 7 depicts a Process sub-model built on "Relations of Process States" identified the Goal sub-model. We have already mentioned, that an ArchiMate event is a state change, so we present states using ArchiMate events. Each sentence presenting goals (requirements) is transformed to an ArchiMate event.

Most concepts *Medical Procedure*,*Group of Medical Procedures*, *Product* are business objects and have their life cycles shown in Fig. 7.

We use only one of possible semantics for the Process Model, namely Protocol Modeling. Events with the same name in different life cycles of this model mean

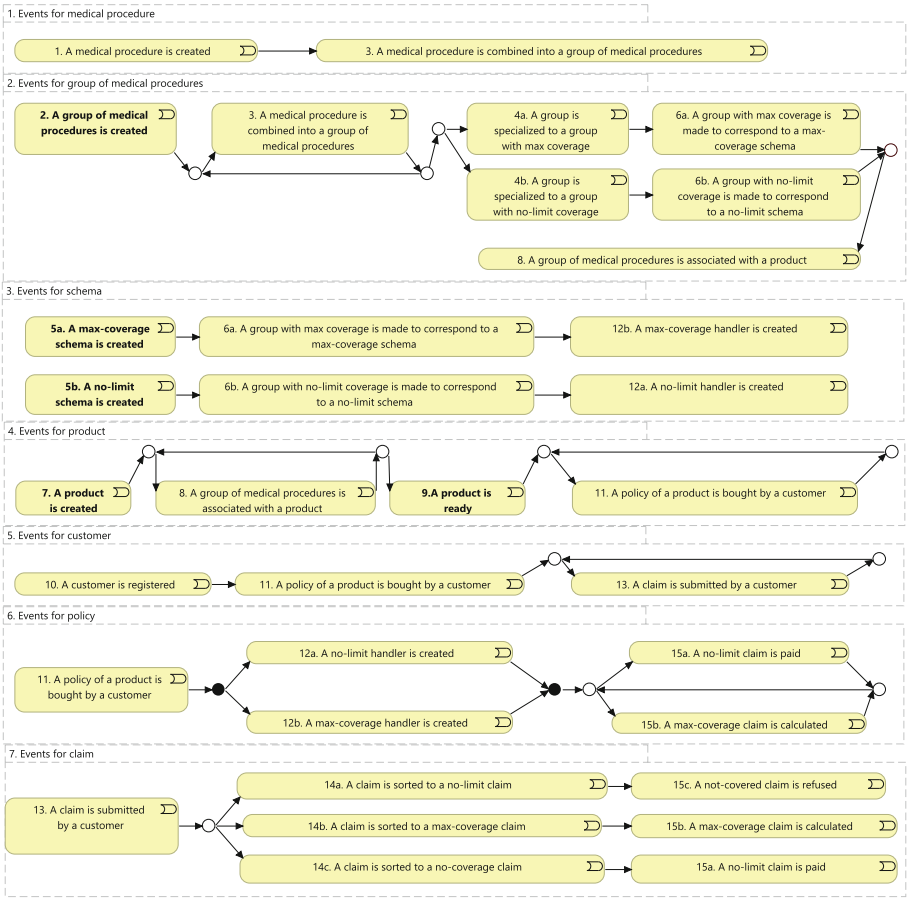

Fig. 7. Business Process sub-model

the CSP-parallel composition [14]. The CSP-parallel composition restricts the events allowed in each system state as it is explained in Sect. 2: if an event is recognized by several instances of protocol machines, it can happen only if all these instances are in the state where this event can happen.

A business process is a set of sequences of synchronized execution of instances of protocol machines. We have numbered the events in Fig. 7 to work the reader through one of the possible sequences of the process presented in Fig. 7.

- 1. A medical procedure is created.
- 2. A group of medical procedures is created.
- 3. A medical procedure is combined with a group of medical procedures.
- 4a. A group is specialized to a group with max coverage.
- 5a. A max-coverage schema is created. (This event is added by the modeler. It is missing in the Goal sub-model.)

- 6a. A group with max coverage is made to correspond to a max-coverage schema.
- 7. A product is created.
- 8. A group of medical procedures is associated with a product.
- 9. A product is ready. (This event is added by the modeler. It is missing in the Goal sub-model.)
- 10. A customer is registered.
- 11. A policy of a product is bought by a customer.
- 12a. A no-limit handler is created. AND 12b. A max-coverage handler is created.
- 13. A claim is submitted by a customer.
- 14b. A claim is sorted to a max-coverage claim.
- 15b. A max-coverage claim is calculated (The Balance of the Policy is updated).

Figure 7 shows that there are synchronous events. For example, *"A Group of medical procedures is associated with a Product"* for a *Group* and a *Product*; *"A Medical Procedure is combined with a Group of Medical procedures"* for a *Medical Procedure* and a *Group*.

The life cycle of a *Group of Medical Procedures* has alternative states *4a. A group is specialized to a group with max coverage.* and *4b. A group is specialized to a group with no-limit coverage.*

If a life cycle of an business object needs a set of instances of another business object, a cycle is designed. For example, a set of instances of "Medical Procedure" can be added to each "Group of Medical procedures".

Some process states, that are not presented in the Goal sub-model, have been added in the Business Process sub-model. For example, *2. A group of medical procedures is created*, *5a. A max-coverage schema is created*, *5b. A no-limit schema is created* and 9. *A product is ready.* This means that the set "Relations of Process States" identified in the Goal sub-model is a subset of "Relations of Process States" used in the Business Process sub-model.

4 Generalization of Semantic Relations Between Sub-models of an Enterprise Model

In the presented case study, we used the semantic relations of sub-models of an enterprise model to organize a modeling process that results in aligned set of sub-models. In this section, we formalize the semantic relations of sub-models in one enterprise model.

4.1 Analysis of a Goal Sub-model

A $Goal\ sub-model = (G, R_m, R_s, R_a)$.

The elements $g \in G$ in Fig. 1 are indexed to show how they are refined into sub-goals or how they refine a parent goal. To show how a parent goal is refined, we use the following notation. Let g_{tn} be a goal under refinement tn:

$$\{g_{tn} | t \in \{m, s, a\} \wedge n \in \mathbf{N}\}.$$

where $t \in \{m, s, a\}$ denote the type of refinement, being milestone (m), sub-domain(s) and alternative(a) refinement respectively. And where n is used to distinguish refinements of the same type (such as refinements $m1, m2, m3$ and $m4$ in the goal sub-model in Fig. 1).

To show how a sub-goal refines its parent goal, we use the following notation:

$$G_{tn} = \{g_{tn_k} \in G |\ t \in \{m, s, a\} \wedge\ n \in \mathbf{N}\ \wedge\ k \in \mathbf{N}\}.$$

The set of sub-goals $G_{tn} = \{g_{tn_1}, ...g_{tn_k}\}$ refines the parent goal g_{tn}.

- G is a finite set of goals, sub-goals and requirements. Each element $g \in G$ is a sentence in the natural language presenting a state or a partial state of the modeled system.
- R_m is a set of milestone-type refinement relations.
  An element $r_{mn} \in R_m$ refines a goal g_{mn} to a finite set of goals, that are ordered, namely, form a sequence to complete the goal g_{mn}:

$$r_{mn} = (g_{mn}, (G_{mn}, O_{mn})),$$

$$G_{mn} = \{g_{mn_k}\},\ n, k \in \mathbf{N},$$

$$O_{r_{mn}} = \{(g_{mn_1}, g_{mn_2}), ..., (g_{mn_{k-1}}, g_{m_k})\} \models g_{mn}.$$

- R_s is a set of sub-domain AND refinement relations. An element $r_{sn} \in R_s$ refines a goal g_{sn} to a set of goals, union of which means the completion of g_{sn}:

$$r_{sn} = (g_{sn}, (G_{sn}, U_{sn})),$$

$$G_{sn} = \{g_{sn_1}, ..., g_{sn_k}\},\ n, k \in \mathbf{N},\ (g_{sn_1} \cap ... \cap g_{sn_k}) = \emptyset.$$

$$U_{sn} = (g_{sn_1} \cup ... \cup g_{sn_k}) \models g_{sn}.$$

- R_a is a set of alternatives, i.e. OR refinement relations. An element $r_{an} \in R_a$ refines a goal g_{an} to a set of goals, appearance of one of which means the completion of g_{an}:

$$r_{an} = (g_{an}, (G_{an}, A_{an})),$$

$$G_{an} = \{g_{an_k}, ..., , g_{an_k}\},\ n, k \in \mathbf{N},\ (g_{an_1} \cap ... \cap g_{an_k}) = \emptyset.$$

$$A_{an} = \forall\ i = 1, ...k :\ (g_{an_i} \models g_{an}).$$

4.2 Goal Sub-model and Concept Sub-model

A $Concept\ sub - model = (C, R)$ is a tuple

- C a finite set of concepts; $c_i \in C,\ i = 1, ..., n,\ n \in N$.
- R is finite set of relations; $r \in R,\ r = (c_i, c_j),\ c_i, c_j \in C$.

The lexical analysis of goals G expressed in natural language in a goal model is aimed to identify the constraints imposed by the Goal sub-model on the Concept sub-model in one enterprise model:

- $Nouns(G)$ is a finite set of nouns (noun phrases), $n \in Nouns(G)$, forming the sentences presenting goals, sub-goals and requirements.
- $Verbs(G)$ is a finite set of verbs (verb phrases, prepositions), $v \in Verbs(G)$, used in the sentences presenting goals, sub-goals and requirements.
- $Relations\ of\ Concepts(G) = \{(v, n_i, n_j) \mid v \in Verbs(G),\ n_i, n_j \in Nouns(G)\}$, is a finite set of triples (v, n_i, n_j) found in the sentences presenting goals, sub-goals and requirements; where v is a verb, n_i, n_j are nouns.

The $Nouns(G)$ and $Relations\ of\ Concepts(G)$ are subsets of concepts and relations of the Concept sub-model. It is because the designers of the Concept sub-model often add some new concepts-attributes and their relations with other concepts.

Figure 6 shows a Concept sub-model that respects the constraints imposed by the Goal sub-model in Fig. 1.

4.3 Process, Goal and Concept Sub-models

A $Process\ sub-model$ is a set of behaviors.
$Process\ sub-model = \{Behaviour_k \mid k = 1, ..., K,\ K \in N\}$.
A behavior is a tuple $Behaviour = (S, T)$, where

- S is a finite set of states, $s_i, s_j \in S;\ i, j \in N$.
- T is a finite set of transitions between states: $t_h \in T\ h \in N,\ t_h = (s_i, s_j) \in T$. Transitions are relations of process states.

The constraints imposed by the Goal and Concept sub-models on the Process sub-model in one enterprise model are the following.

- Each noun $n \in Nouns(G)$, being a concept $c \in C$ of the Concept sub-model, has a corresponding $Behavior$, except if the concept composes or specializes another concept.
- Each goal $g \in G$ of a Goal sub-model has a corresponding state $s \in S$ in the Process sub-model.
- The transitions (being Relations of Process States) are identified by analysis of each milestone refinement of the Goal sub-model. Each pair of goals of an milestone refinement r_{mn} in the Goal sub-model:

$$O_{r_{mn}} = \{(g_{mn_1}, g_{mn_2}), ...(g_{mn_{k-1}}, g_{m_k})\} \models g_{mn},$$

  corresponds to a transition of states in the Process sub-model. The states are named after the goals in the set G_{mn}.
- Each alternative refinement r_{an} in the Goal sub-model:

$$A_{an} = \forall i = 1, ...k :\ (g_{an_i} \models g_{an}),$$

  corresponds to an OR-split of states in the Process sub-model. The states are named after the goals in the set G_{an}.

– Each sub-domain refinement r_{sn} in the Goal sub-model:

$$U_{sn} = (g_{sn_1} \cup ... \cup g_{sn_k}) \models g_{sn},$$

corresponds to an AND-split of states in the Process sub-model. The states are named after the goals in the set G_{sn}.

Figure 7 presents a Process sub-model that respects constraints imposed by the Goal and Concept sub-models (Figs. 1, 6).

In this work, we have defined the semantic relations or constraints that are imposed by sub-models in an enterprise model. The semantic relations cover three modeling semantics used for sub-models. The goal modeling semantics is the leading semantics. The concept modeling and the process modeling semantics are applied within constraints imposed by the goal sub-model. The Concept and Process sub-models are complete if they present the elements (*concepts*, *states*) and relations (*relations-of-concepts*, *relations-of-process-states*) identified in the Goal sub-model.

If a sub-model is depicted as a set of partial sub-models (views), then the semantic relations, defined in this paper, can be applied to the union of elements and relations of views of the sub-model. For the partial sub-models, that do not form a complete sub-model, the partial constraints can be derived from the semantic relations of sub-models, however, this needs more investigation.

5 Conclusion and Future Work

Enterprise modeling approaches have difficulties in producing enterprise models with aligned sub-models. Even the ArchiMate [24] approach, which has been designed for consistent enterprise modeling, experiences these difficulties. The ArchiMate provides a storage for an internal model of each enterprise model. This internal model is to be used for generating sub-models. However, the filling the internal model in with elements and relations is done by drawing sub-models of different semantics. The drawing and naming of elements and relations is made by a team of enterprise architects. Team members are humans and may duplicate elements of models using different names, forget the details of the semantics of sub-models and the semantic relations of sub-models. This results in unaligned sub-models of an enterprise model.

We have reviewed several enterprise modeling approaches and found out that the semantic relations of sub-models are recognized, but not used to direct the design of aligned sub-models.

In order to analyze and formalize the semantic relations of sub-models in an enterprise model, we have analyzed a case study depicted in ArchiMate. We have identified the structures and formulated semantic constraints imposed by sub-models to each other. All these structures can be identified in the Goal sub-model of an enterprise model and can be used for design of a Concept and a Process sub-models. The identified structures and constraints can be used in any enterprise modeling approach.

Currently, we apply the found structures for the design of aligned sub-models directed by the semantic relations via the integration of sub-models. We are experimenting in one of the ArchiMate tools, namely, Archi [2]. In the future work, building on our former results [7,15], we plan to define constraints imposed by sub-models for tool extensions to enable automated checks and directed design of aligned sub-models.

References

1. Beauvoir, P., Sarrodie, J.: Archi-the free archimate modelling tool. User Guide, The Open Group (2018)
2. Beauvoir, P., Sarrodie, J.B.: Archi-Open Source Archimate Modelling (2019)
3. Egyed, A.: Instant consistency checking for the UML. In: Proceedings of the 28th International Conference on Software Engineering, pp. 381–390 (2006)
4. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM **21**(8), 666–677 (1978)
5. Hui, L.M., Leung, C.W., Fan, C.K., Wong, T.N.: Modelling agent-based systems with UML. In: Proceedings of the Fifth Asia-Pacific Industrial Engineering and Management Systems Conference (2004)
6. Jackson, M.: System Development. Prentice-Hall, Englewood Cliffs (1983)
7. Joosten, S., Roubtsova, E., Haddouchi, E.M.: Constraint formalization for automated assessment of enterprise models. In: International Conference on Enterprise Information Systems (ICEIS), vol. 2, pp. 430–441 (2022)
8. Kaisler, S., Armour, F., Valivullah, M.: Enterprise architecting: critical problems. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, p. 224b (2005)
9. Kontio, M.: Architectural Manifesto: Designing Software Architectures. Part 5. Introducing the 4+ 1 View Model. IBM developerWorks (2005)
10. Kruchten, P.B.: The 4+ 1 view model of architecture. IEEE Softw. **12**(6), 42–50 (1995)
11. Letier, E., Kramer, J., Magee, J., Uchitel, S.: Deriving event-based transition systems from goal-oriented requirements models. Autom. Softw. Eng. **15**(2), 175–206 (2008)
12. Marosin, D., van Zee, M., Ghanavati, S.: Formalizing and modeling Enterprise Architecture (EA) principles with Goal-Oriented Requirements Language (GRL). In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 205–220. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_13
13. McNeile, A., Roubtsova, E.: CSP parallel composition of aspect models. In: Proceedings of the 2008 AOSD Workshop on Aspect-Oriented Modeling, pp. 13–18 (2008)
14. McNeile, A., Simons, N.: Protocol modelling: a modelling approach that supports reusable behavioural abstractions. Softw. Syst. Model. **5**(1), 91–107 (2006)
15. Michels, G., Joosten, S., van der Woude, J., Joosten, S.: Ampersand. In: de Swart, H. (ed.) RAMICS 2011. LNCS, vol. 6663, pp. 280–293. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21070-9_21
16. OMG: Unified Modeling Language (UML, formal) 01 March 2015. https://www.omg.org/spec/UML/2.5/

17. Pereira, C.M., Sousa, P.: Enterprise architecture: business and IT alignment. In: Proceedings of the 2005 ACM Symposium on Applied Computing, pp. 1344–1345 (2005)
18. Quartel, D., Engelsman, W., Jonkers, H., Van Sinderen, M.: A goal-oriented requirements modelling language for enterprise architecture. In: 2009 IEEE International Enterprise Distributed Object Computing Conference, pp. 3–13. IEEE (2009)
19. Respect-IT: A KAOS Tutorial, V1.0 (2007)
20. Roubtsova, E.: EXTREME: EXecuTable requirements engineering, management, and evolution. In: Progressions and Innovations in Model-Driven Software Engineering, pp. 65–89. IGI Global (2013)
21. Roubtsova, E.: Advances in behavior modeling. In: Advances in Computers, vol. 97, pp. 49–109. Elsevier (2015)
22. Roubtsova, E.: Interactive Modeling and Simulation in Business System Design. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-15102-1
23. Sandkuhl, K., Stirna, J., Persson, A., Wißotzki, M.: Enterprise Modeling. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43725-4
24. The Open Group: ArchiMate 3.1 Specification (2012–2021). https://pubs.opengroup.org/architecture/archimate3-doc/