# A Formal Approach to Architectural Descriptions – Refining the ISO Standard 42010

Sabine Buckl[1], Sascha Krell[2], and Christian M. Schweda[1]

[1] Lehrstuhl für Informatik 19
Technische Universität München
Boltzmannstraße 3, 85748 Garching, Germany
{sabine.buckl,schweda}@in.tum.de
[2] Professur für Wirtschaftsinformatik
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
sascha.krell@unibw.de

**Abstract.** Architectural descriptions representing and modeling the architecture of a system or parts thereof are typically used in the engineering disciplines to plan, develop, maintain, and manage complex systems. Primarily originating from construction engineering, the means of architecting and architectural descriptions have been successfully transferred to related disciplines like software engineering. While a rich and formal theory on conceptual modeling exists as well as frameworks on how to approach architectural descriptions, e.g. the ISO standard 42010, only few attempts have yet been made to integrate the prescriptions and guidelines from these sources into a formal architectural description framework. In this paper, we establish such a framework against the background provided by the ISO standard 42010 by formally defining the terms *concern*, *view*, *viewpoint*, and *architectural description*. Further, an outlook discusses potential application areas of the framework.

## 1 Motivation

Development, design, and maintenance of architectures have a long history in the engineering disciplines. Primary originating from construction engineering the objectives of architecture – to be strong or durable (firmitas), useful (utilitas), and beautiful (venustas) [1] – and their means have been applied to other disciplines to address challenges in related domains (cf. [2,3]). One of these disciplines is software engineering, in whose context the term architecture can be defined in accordance with the ISO Std. 42010 as "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" [4]. In the same sense, Rechtin coined the following proverb "every system has an architecture" in [5]. He thereby, emphasized on the fact that an architecture is an *intrinsic* property of a system that cannot be neglected. The architecture, however, is not necessarily documented, i.e. made explicit. This idea yields a delicate but central distinction of the two terms *architecture* and *architecture description* that will reverberate through the remainder of the article. During the software development process, for instance, a variety of different documentations, so called "views" or "models", of the system under consideration are created and used to facilitate the communication between the involved stakeholders, e.g. customers, software architects, and software engineers. Examples for such "views" are different diagrams corresponding to the different diagram types as proposed by the UML, the de-facto standard for software engineering [6]. In accordance with the ISO Std. 42010 the entirety of these views is referred to as *architectural description*, a "collection of products documenting the architecture" [4] of the system.

In the context of architectural descriptions, the way human beings, i.e. stakeholders, comprehend knowledge about what is perceived to exist, i.e. the system under consideration, plays an important role. The science dealing with such aspects is referred to as *epistemology* [7]. In [8] Becker and Niehaves present an epistemological reference framework for information systems research, which details on questions in which way a person can arrive at *true* cognition. In the context of architectural descriptions, especially the *ontological aspect* discussed in the framework is of interest. The ontological aspect is concerned with the object of cognition, i.e. the architecture of the system under consideration. Thereby three positions can be differentiated

**ontological realism,** according to which the system's architecture exists independently of human cognition,

**ontological idealism,** according to which the system's architecture is a construct depending on the consciousness of the observing person, and

**kantianism,** according to which the system's architecture consists of parts, which are dependent and independent of the observing person.

Abstaining from in-depth discussions on the ontological perspective, we resort ourselves to the position of ontological realism that well aligns to the understanding of Rechting [5].

Although the importance of architectural descriptions, e.g. in the context of software engineering is unquestioned, no general formal framework for such descriptions has yet been developed. For special purpose languages as e.g. the UML, formalization approaches have nevertheless been undertaken, see e.g. [9]. The absence of a general framework is especially interesting as rich and formal theories for conceptual modeling exists as well as frameworks on how to approach architectural descriptions (cf. [4]). Building on the framework of the ISO Std. 42010, Dijkman et al. present in [10] a basic formalization approach targeting consistency of viewpoints in the context of multi-viewpoint modeling. Doubtlessly their approach provides a valuable contribution in the field, but does not deliver a comprehensive formalization for the field of architecture description. This in contrast is the experienced research gap, that our paper addresses by introducing and discussing frameworks for conceptual modeling and for architectural descriptions in Section 2. These frameworks are used as foundation for the approach to architectural descriptions presented in Section 3, which formally defines the terms *concern*, *view*, *viewpoint*, and *architectural description*. Finally, Section 4 provides a critical reflection of the contribution of the paper and discusses future topics of research.

## 2 Describing Architectures

Postponing more elaborate considerations on the structure of an architectural description to Section 2.2, we restrict ourselves to an intuitive understanding of description as a purposeful abstraction of an architecture, i.e. as some sort of *conceptual model* thereof. This aligns with the definition of the latter term given by Mylopoulos, who claims in [11] that the activity of conceptual modeling, which is the activity of creating conceptual models, is "the activity of describing the physical and social world around us for purposes of communication and understanding". He further states, that conceptual models are developed for certain users or *stakeholders*, as we call them in accordance with the ISO Std. 42010 (cf. [4]). In the remainder of the paper, we will understand architectural descriptions as conceptual models of the architecture, more precisely as specifications of conceptual models of the architecture. We precedingly discuss further properties of conceptual models and on ways to describe these properties in a more abstract fashion in the following section.

### 2.1 Introduction to Conceptual Models

Guizzardi introduces in [12] a framework that can be applied to promote the understanding of the relationships between architectures, conceptual models thereof, and architectural descriptions. Figure 1 summarizes the framework, thereby also introducing additionally relevant concepts as *conceptualization* and *modeling language*.

Central concepts of the framework are on the one hand the *(mental) model* of the architecture and the corresponding *model specification*. The mental model thereby alludes to a stakeholder's understanding of what the architecture looks alike. The model specification is therein understood as the *representation* of a model using an appropriate modeling language. This introduces the concept of the representation that in accordance to Guizzardi [12] can be understood as function mapping from the space of model entities to a corresponding space of syntactically correct instantiations of modeling language constructs. Complementing the representation function, the framework provides the *interpretation function* that conversely maps from syntactically correct instantiations of the modeling language constructs to corresponding model entities. With these definitions at hand, it intuitively becomes clear that the purposefulness and appropriateness of an architectural description, i.e. a model specification, heavily depends on the backing modeling language's ability to express and represent the concepts' underlying the mental model. Staying in the field of conceptual modeling, Guizzardi derives in [12] a framework for evaluating the appropriateness of a

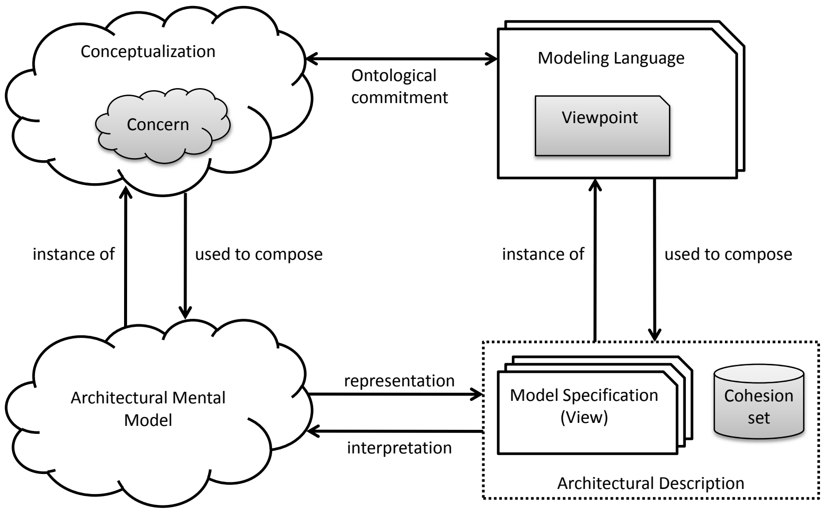

**Fig. 1.** Conceptual modeling on architectures

modeling language. This framework builds on a basic idea coined by Gurr in [13] according to which the representation function needs to be an *isomorphism* with the interpretation function as its inverse. Elaborating on the difficulties associated with proofing the isomorphism nature of a function mapping mental models to representations thereof, Guizzardi reformulates the demand to two requirements for the representation function and two complementing requirements targeting the interpretation function. These requirements read as follows:

**Lucidity.** A specification must be *lucid* with respect to its corresponding model, which means that every construct in the specification must represent at most one entity from the model. Thereby, overloaded representation constructs are forbidden. *(injective representation function)*

**Soundness.** A specification must be *sound* with respect to its corresponding model, which means that every construct in the specification must represent at least one entity from the model. Thereby, construct excess in the representation is avoided. *(surjective representation function)*

**Laconicity.** A specification must be *laconic* with respect to the corresponding model, which means that every entity in the model can be derived by interpretation from at most one construct in the specification. Thereby, construct redundancy is forbidden. *(injective interpretation function)*

**Completeness.** A specification must be *complete* with respect to the corresponding model, which means that every entity in the model must be derivable by interpretation from at least one construct in the specification. Thereby, it is ensured that the entire model can be derived from its specification. *(surjective interpretation function)*

In the light of above requirements, different definitions for the term *modeling language* are discussed in literature, where e.g. Guizzardi in [12] stays to a structural perspective on languages inspired by the idea of the two mappings. More advantageous for our subsequent considerations is a notion of language that provides more insights into the internals of modeling. In this vein, we adopt the definition advocated by Kühn in [14], where he names the following language constituents:

**Syntax.**[1] The syntax defines the structure of the language, i.e., the constructs that form the language body, and the grammar, i.e., the rules for combining the constructs to valid language expressions.

**Semantics.** The semantics provides the meaning for the language constructs, i.e., prescribes the interpretation of the constructs to corresponding model entities.

**Notation.**[2] The notation introduces user-perceivable concepts, as e.g. symbols, used to present language constructs to a language user.

In terms of Kühn [14], the representation function of Guizzardi [12] consists of semantic and notational aspects. Explicitly accounting for the latter concepts is beneficial for our subsequent consideration due to multiple reasons, of which the most prominent one is ”notational plurality”. This aspect is discussed by Kühn on a general level, but also emphasized in the context of different architecture-related disciplines, as e.g. EA management, for which Buckl et al. present in [15] a technique for decoupling notational aspects from semantic ones and show how such a technique can be implemented in a tool. With this in mind, one can justify to abstain from in-depth considerations on the notational aspect of languages.

### 2.2 ISO Std. 42010

The ISO Std. 42010 on architectural descriptions for software and systems engineering [4] establishes a terminological framework for discussions on architectural descriptions. As part of this framework, the relevant terms of architecture and architectural description are defined. A more in-depth discussion on the structure of an architectural description is provided along a *metamodel*[3] for such descriptions, of which Figure 2 presents the relevant part for this article. The concepts introduced thereby are defined in [4] as follows:

**Stakeholder.** A stakeholder is an individual, team, organization, or role having concerns with respect to the system.

**Concern.** A concern is an area of interest in a system pertaining to developmental, technological, business, operational, organizational, political, regulatory, social, or other influences important to one or more of its stakeholders.

**View.** A view is a representation of a system from the perspective of an identified set of architecture-related concerns. Thereby, the system is abstracted to *architectural models.*

[1] In other language specifications, the concept is alluded to as *abstract syntax.*

[2] In other language specifications, the concept is alluded to as *concrete syntax.*

[3] The standard actually uses the term ”conceptual model” and denotes ”metamodel” only as alternative term. To avoid confusions, the remainder of the article employs the latter term.

**Viewpoint.** A viewpoint contains conventions for the construction and interpretation of a view.

The notion of architectural model is further detailed in the standard, leading to the conclusion that such model should be identified with a model specification in terms of Guizzardi [12]. In addition, explanation for the distinction between view and model is provided, emphasizing on the possibility to reuse architectural models in different views. While such modularity might in fact be beneficial during applications of the framework, it does to us seem only of minor importance in the context of the subsequent formal approach to architectural descriptions. In this vein, we apply without loss of generality a simplification to the metamodel by uniquely identifying views and architectural models and subsuming these concepts under the term "view".

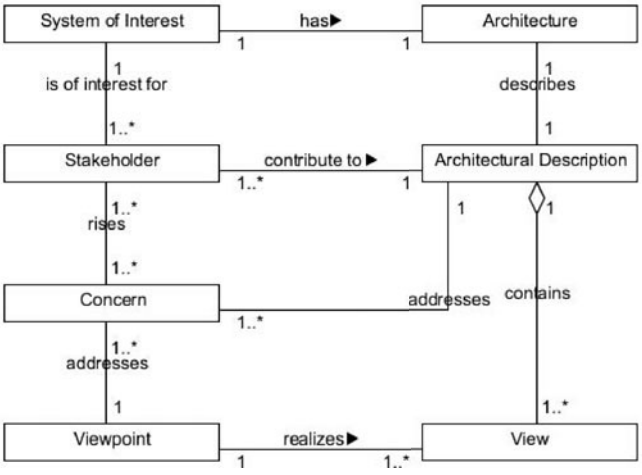

**Fig. 2.** Metamodel of the ISO Std. 42010 [4]

With this shortened version of the standard at hand, we briefly revisit the concepts introduced in Section 2.1. The stakeholder concept allows to make explicit the owner of a mental model that is represented as part of an architectural description. The conceptualization underlying the mental model as reflected in the corresponding modeling language is mirrored in the stakeholder's concerns. Put in other words, a concern employs an underlying conceptualization that again is tightly linked to the syntax and semantics of the modeling language, while notational aspects may not apply here. These notational aspects finally come into play during the construction of the actual views, which are developed and interpreted along the guidelines stated in the viewpoint. From this, one can sensibly derive that the modeling language constitutes a part of the viewpoint. The above considerations lay the basis for our subsequent elaborations on a formal approach to architectural descriptions.

## 3 A Formal Approach to Architectural Descriptions

The ISO Std. 42010 provides a valuable framework for understanding architectural descriptions embedded into their creation and utilization context, as well as their related stakeholders. This is achieved by a metamodel of concepts used for making the description's context explicit. Following the interpretation of architectural descriptions or more precisely their constituting views as *models* of the architecture in terms of conceptual modeling, a disadvantage of the informal conceptualization of the standard becomes apparent. In particular, the standard does not allow for reasoning on an architectural description's underlying representation and interpretation function, which recruits from corresponding functions in the constituting viewpoints.

Subsequently, a formalization of the concepts from the metamodel of the ISO Std. 42010 is provided, which can be used for profound formal considerations on the model nature of an architectural description. Preparing this formalization, we introduce two basic concepts as follows:

– the *set of all possible architectures* $\mathcal{A}$, which corresponds to the "set of possible worlds" as introduced by Guizzardi in [12], and
– the *set of all entities and relationships* $\mathcal{D}$. The entities and relationships make up the constituents of an architecture, i.e. exist in the corresponding "world".

With these basic notions in mind, our roadmap for this section at first reconciles the terms of *conceptualization* and *modeling language*, respectively. Building on a formalized understanding of these terms, we derive relationships on the corresponding structures, leading to the notion of an *ontological commitment* of a language in respect to a conceptualization (cf. [12]). Formal definitions for *concerns* and *viewpoints* are presented as extensions of *conceptualizations* and *modeling languages*, respectively. Alongside, formalized relationships are introduced to define the terms of *embedding*, *equivalence*, and *orthogonality* of concerns on the one hand, as well as the *appropriateness* of a viewpoint in respect to a concern on the other hand. Concluding our formalization approach, we elaborate on *consistency* requirements on architectural descriptions that employ multiple viewpoints. For the subsequent discussions, it also has to be noted that albeit the formality of the presented model, we do not assume that each stakeholder describes his or her conceptualizations, concerns, or viewpoints on such an abstract level.

### 3.1 Conceptualization and Modeling Language

Guizzardi introduces his definition of conceptualization in [12] as a set of structuring principles used (implicitly) during the creation of a mental model. In this vein, a conceptualization provides *concepts* that are used to classify real-world entities and relationships[4] to *classes* or *association types*. In line with this understanding, a conceptualization $z \in \mathcal{Z}$ is described as a tuple consisting of:

[4] Entities and relationships are subsequently subsumed as "instances".

- the set of admissible architectures $\mathcal{A}_z \subseteq \mathcal{A}$,
- the set of admissible instances $\mathcal{D}_z \subseteq \mathcal{D}$, and
- the set of concepts[5] $\mathcal{R}_z \subseteq \bigcup_{n \in \mathbb{N}} (\mathcal{A}_z \rightarrow \mathbb{P}(\mathcal{D}_z^n))$.

The above definition yields a *contextual* and *extensional* understanding of concepts, i.e., defines the concept not solely over its instances, but in relation to the architecture that they are contained in, and understands a concept in a set-theoretic fashion. This also explains, why the powerset notation is used: a concept $\langle a, \mathcal{D}_1 \rangle$ is a tuple of an architecture and the corresponding instances that together form the *extension* of the concept.

With this understanding of conceptualizations, we can further establish an *equivalence* relationship $\equiv \subseteq \mathcal{Z} \times \mathcal{Z}$. This relationship builds on the existence of a bijective function $\underline{d}_z : \mathcal{D}_{z_1} \rightarrow \mathcal{D}_{z_2}$. Based on this, the equivalence of two conceptualizations $z_1 = \langle \mathcal{A}_{z_1}, \mathcal{D}_{z_1}, \mathcal{R}_{z_1} \rangle$ and $z_2 = \langle \mathcal{A}_{z_2}, \mathcal{D}_{z_2}, \mathcal{R}_{z_2} \rangle$ can be defined as:

$$\begin{aligned} z_1 \equiv\ z_2 \Leftrightarrow\ & \mathcal{A}_{z_1} = \mathcal{A}_{z_2} \ \wedge \\ & \forall \ \langle a, \{d_1, ..., d_n\} \rangle \in \mathcal{R}_{z_1} : \langle a, \{\underline{d}_z(d_1), ..., \underline{d}_z(d_n)\} \rangle \in \mathcal{R}_{z_2}. \end{aligned}$$

Pursuing the path from a conceptualization towards a modeling language, a language $l \in \mathcal{L}$ is defined as tuple $l = \langle \mathcal{S}_l, i_l, n_l \rangle$ with three constituents: *(abstract) syntax* $\mathcal{S}_l$, a mapping function $i_l : \mathcal{S}_l \rightarrow \mathcal{D}_l \cup \mathcal{R}_l$ reflecting the *semantics*, and a *notation* mapping $n_l$. The latter mapping defines for each syntactical concept the corresponding symbolic representation. Resorting to the considerations from Section 2.1, we abstain from discussions on the notation function here.

Linking conceptualizations and modeling languages, we resort to the notion of *ontological commitment* as discussed by Guizzardi in [12]. In these terms, a modeling language commits itself to a conceptualization, iff it is suitable to represent the conceptualization's concepts. In the operationalization of the commitment relationship $\sim \subseteq \mathcal{L} \times \mathcal{Z}$, the suitability is formalized as the existence of a bijective function $m : \mathcal{D}_l \cup \mathcal{R}_l \rightarrow \mathcal{D}_z \cup \mathcal{R}_z$ such that

$$l \sim z \Leftrightarrow \exists \ L \subseteq image(i_l) : m(L) = \mathcal{D}_z \cup \mathcal{R}_z.$$

Refraining to the equivalence relationship between conceptualizations as defined above, an alternative definition of the ontological commitment relationship can be established between a modeling language and an equivalence class of conceptualizations $[z]_\equiv$. Based on this notion, a modeling language $l$ commits itself to a set of (equivalent) conceptualizations $[z]_\equiv$, iff a bijective function $m : \mathcal{D}_l \cup \mathcal{R}_l \rightarrow \mathcal{D}_z \cup \mathcal{R}_z$ exists such that

$$l \sim [z]_\equiv \Leftrightarrow \exists \ L \subseteq image(i_l) : m(L) = \mathcal{D}_z \cup \mathcal{R}_z,$$

where $z$ on the right hand side denotes an arbitrary element from the equivalence class $[z]_\equiv := \{z' \in \mathcal{Z} | z' \equiv z\}$.

[5] The symbol $\mathbb{P}$ represents the powerset of a given set.

## 3.2 Concern and Viewpoint

An architectural concern represents a stakeholder's "area of interest" in an architecture. In this vein, the concern is reflected by a distinct mental model of the architecture or more precisely a part thereof, complemented with an underlying conceptualization. The concern further denotes which parts of the overall architecture are mirrored in the corresponding "model", i.e. selects the relevant *instances.* Put in other words, a concern goes beyond a sole conceptualization but also brings along a *filter function* $f$ that determines which parts of the architecture are of interest. Formally, a concern $c \in \mathcal{C}$ can be defined as tuple $c = \langle \mathcal{A}_c, \mathcal{D}_c, \mathcal{R}_c, f_c \rangle$ with:

– the set of admissible architectures $\mathcal{A}_c \subseteq \mathcal{A}$,
– the set of admissible instances $\mathcal{D}_c \subseteq \mathcal{D}$,
– the set of concepts $\mathcal{R}_c \subseteq \bigcup_{n \in \mathbb{N}} (\mathcal{A}_c \rightarrow \mathbb{P}(\mathcal{D}_c^n))$ that is further consistent with
– the filter function $f_c : \mathcal{A}_c \rightarrow \mathbb{P}(\mathcal{D}_c)$.

The term "consistent" in this case means

$$\forall \langle a, \{d_1, ..., d_n\} \rangle \in \mathcal{R}_c : \{d_1, ...d_n\} \subseteq f_c(a).$$

At this point it is necessary to give some general remarks. We do not try to formalize the term mental model or tantamount architecture[6] itself. In our opinion this is a pointless task, because of the impossibility to discover the complete nature of a complex system. Any kind of identification, with a colored graph for example, would be a simplification of that system, which might be allowed for modeling purposes but not for a formalization approach, aiming at a complete and consistent specification. Neither we can give construction rules for the above introduced filter functions with the same argument. A certain filter function strongly relies on a specific theory about the modeled field, which cannot be anticipated in advance. The intension behind a concern has thus to be left to philosophical observations.

Figure 2 makes obvious that a viewpoint can relate to more than one concern. From our perspective such an aggregation of concerns can only be made, if the aggregated concerns can themselves be interpreted as *one* consistent concern again. Descriptive this means, all participating concerns must employ a compatible conceptualization, reflected in an assignment of equal concepts for the same instances, or must in contrast be completely disjoint, i.e. filter for other parts of the architecture under consideration. Otherwise, we would gain complex types, which could cause problems in interpreting them, i.e. there might be no correspondent entity in the real world domain for a given complex type. This would break the *soundness* requirement of Guizzardi (cf. Section 2.1) hence depriving an architectural description using this "inconsistent" viewpoint of its validity as representation of a mental model of the architecture. In these cases, it would be more beneficial to define a further viewpoint to avoid this situation.

[6] Epistemological, an architecture could be as well understood as the intension of a system of interest.

Approaching the notion of a "consistent aggregation" of concerns, we can establish a relationship among concerns expressing that one concern can be *embedded* into another one. Put in other words, embedding ($c_1 \sqsubseteq c_2$) means that the area of interest represented by one concern $c_1$ is completely covered by the area of interest of another concern $c_2$. The *embedding*-relationship ($\sqsubseteq \subseteq \mathcal{C} \times \mathcal{C}$) can formally be defined via an auxilliary set $\mathcal{D}'_{c_1} \subseteq \mathcal{D}_{c_2}$ of instances covered by $c_2$ and a bijective function $\underline{d}_c : \mathcal{D}'_{c_1} \to \mathcal{D}_{c_2}$ as follows[7]

$$\begin{aligned} c_1 \sqsubseteq c_2 \Leftrightarrow \mathcal{A}_{c_1} = \mathcal{A}_{c_2} \wedge & \\ & \forall \langle a, \{d_1, ..., d_n\}\rangle \in \mathcal{R}_{c_1} : \langle a, \{\underline{d}_c(d_1), ..., \underline{d}_c(d_n)\}\rangle \in \mathcal{R}_{c_2} \wedge \\ & \forall a \in \mathcal{A}_{c_2} : \{\underline{d}_c(d) \mid d \in f_{c_1}(a)\} \subseteq f_{c_2}(a). \end{aligned}$$

From the above *embedding*-relationship between concerns, we can consistently derive an *equivalence*-relationship as a mutual embedding. In formal terms, the equivalence of concerns $\equiv \subseteq \mathcal{C} \times \mathcal{C}$ can be defined as

$$c_1, c_2 \in \mathcal{C} : c_1 \equiv c_2 \Leftrightarrow c_1 \sqsubseteq c_2 \wedge c_2 \sqsubseteq c_1.$$

The equivalence of concerns thereby builds on the equivalence of the underlying conceptualizations $z(c_1)$ and $z(c_2)$, respectively. In more detail two concerns $c_1$ and $c_2$ are equivalent (regarding a bijective function $\underline{d}_c$) as follows:

$$c_1 \equiv c_2 \Leftrightarrow z(c_1) \equiv z(c_2) \wedge \forall a \in \mathcal{A}_{c_2} : \{\underline{d}_c(d) \mid d \in f_{c_1}(a)\} = f_{c_2}(a).$$

Following the logic of this observation it becomes apparent, that the "biggest" possible concern comprehends the interests of all stakeholders and consequently reflects the whole architecture. It is out of doubt that such a concern would be overly sized to be handled effectively. Nevertheless, the embedded-relation is capable of building the foundation for *abstract* concerns, which comprise the interests of different stakeholders.

We suggest two additional terms in the matter of further improvement for the understanding of compatibility of concerns. Both provide sufficient but not necessary conditions for compatibility.

At first, we are aiming at concerns, which do not "affect" each other, and can therefore be put together without depriving the description's soundness. To substantiate this depiction, we establish the notion of *orthogonality*: Two concerns $c_1, c_2 \in \mathcal{C}$ are orthogonal $c_1 \perp c_2$, if they cover different areas of interest, more precisely, if there is no bijective function $\underline{d}$ that maps the concerns' instances ($\mathcal{D}_{c_1}$ and $\mathcal{D}_{c_2}$) consistently for all commonly admissible architectures. Preparing a formalization of orthogonality, we introduce the auxiliary function $A : \mathcal{C} \times \mathcal{C} \to \mathbb{P}(\mathcal{A})$ of architectures that are admissible in respect to two concerns, as follows

$$A(c_1, c_2) := dom(f_{c_1}) \cap dom(f_{c_2}).$$

[7] In the definition we apply the bijection $\underline{d}_c$ to a set of instances. Thereby, we want to denote the set that results from element-wise application of the bijection.

With this shorthanded notation, the orthogonality of concerns can be defined as[8]

$$c_1 \perp c_2 \Leftrightarrow \nexists\, \underline{d} : \mathcal{D}_{c_1} \rightarrow \mathcal{D}_{c_2} : \forall\, a \in A(c_1, c_2) : f_{c_1}(a) \cap \underline{d}(f_{c_2}(a)) = \emptyset,$$

where $\underline{d}$ is a bijective function.

The second sufficient prerequisite for compatibility of concerns is *conceptualization compatibility*. To allow for a concise definition of the latter relation, we introduce another auxiliary function $R_{\underline{d}} : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{A} \times \bigcup_n \mathbb{P}(\mathcal{D}^n)$ as follows[9]

$$R_{\underline{d}}(c_1, c_2) := \{\langle a, \delta \rangle | a \in A(c_1, c_2) \wedge \delta \subseteq \bigcup_n (f_{c_1}(a) \cap \underline{d}(f_{c_2}(a)))^n\}.$$

For two given concerns $c_1, c_2$, the value of $R_{\underline{d}}(c_1, c_2)$ denotes the possible concepts as tuples $\langle a, \delta \rangle$ over common instances and corresponding architectures that are admissible for both concerns. The value thereby depends on the selection of the bijective function $\underline{d} : \mathcal{D}_{c_1} \rightarrow \mathcal{D}_{c_2}$ in the sense of the former definition. For the context of a specific architecture $a \in A(c_1, c_2)$ Figure 3 illustrates the meaning of $R_{\underline{d}}(c_1, c_2)$.

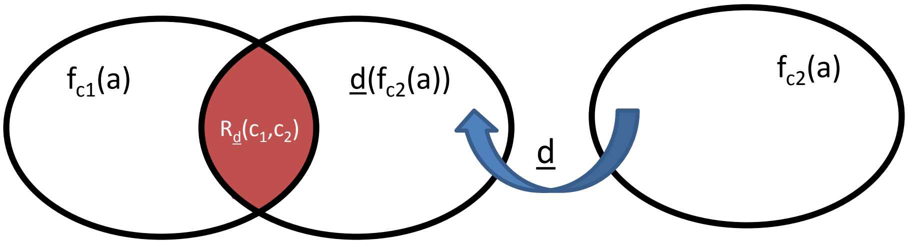

**Fig. 3.** Illustrating $R_{\underline{d}}(c_1, c_2)$ in the context of an architecture $a$

This allows for a concise definition of a *conceptualization compatibility*-relationship via a bijective function $\underline{d} : \mathcal{D}_{c_1} \rightarrow \mathcal{D}_{c_2}$ as

$$\begin{aligned} c_1 \sim c_2 \Leftrightarrow\ & \forall\, a \in A(c_1, c_2) \\ & \forall\, \delta_1 \in \{\{d_1, ...d_n\} \mid \langle a, \{d_1, ...d_n\}\rangle \in \mathcal{R}_{c_1} \cap R_{\underline{d}}(c_1, c_2)\} \\ & \forall\, \delta_2 \in \{\{\underline{d}(d_1), ...\underline{d}(d_n)\} \mid \langle a, \{d_1, ...d_n\}\rangle \in \mathcal{R}_{c_2} \cap R_{\underline{d}}(c_1, c_2)\} : \\ & \delta_1 \subseteq \delta_2 \vee \delta_2 \subseteq \delta_1 \vee \delta_1 \cap \delta_2 = \emptyset. \end{aligned}$$

For subsequent considerations about the term *viewpoint*, we assume to have a single concern, that might aggregate the interests of a set of stakeholders in the sense of the prior relations.

Refraining our considerations from Section 2.1, a viewpoint can be understood as an extension of a modeling language in the same manner, as a concern extends an underlying conceptualization. In this vein, we define a viewpoint $v \in \mathcal{V}$ as a tuple $v = \langle \mathcal{S}_v, i_v, n_v, f_v \rangle$ with:

[8] Footnote 7 also applies in this definition.

[9] Footnote 7 also applies in this definition.

- an abstract syntax denoted by a set $\mathcal{S}_v$ of the instances of $\mathcal{D}_v$ that the viewpoint employs,
- a semantics denoted by a mapping function $i_v : \mathcal{S}_v \rightarrow \mathcal{D}_v \cup \mathcal{R}_v$,
- a notation denoted by a function[10] $n_v$, and
- a filter, determining the relevant part of an architecture, mirrored in a function
  $f_v : \mathcal{A}_v \rightarrow \mathbb{P}(\mathcal{D}_v)$.

Rounding up the definition from above, we operationalize the relationship between a viewpoint and the (aggregated) concern that it is meant to address. Constructing this *addresses*-relationship, we can rely on the definition of the ontological commitment that relates conceptualizations and modeling languages, of which concerns and viewpoints are extensions. Therefore, it is justifiable to start with the generalized notion of commitment, building on the notion of equivalence classes of conceptualizations. Based on this, we establish an *addresses*-relationship as $\sim \subseteq \mathcal{V} \times [\mathcal{C}]_{\equiv}$ via a bijective function $m : \mathcal{D}_v \cup \mathcal{R}_v \rightarrow \mathcal{D}_c \cup \mathcal{R}_c$, as:

$$v \sim [c]_{\equiv} \Leftrightarrow \mathcal{A}_c = dom(f_v) \wedge \\ \forall\, a \in \mathcal{A}_c : \{m(d) \mid d \in f_v(a)\} = f_c(a).$$

## 3.3 View and Architectural Description

Inline with the prefabricates of the preceding section, we define a view $m_v$ corresponding to a viewpoint $v$ as the application of $v$ to an architecture $a \in \mathcal{A}$. We understand a view as a function $m_v : \mathcal{A}_v \rightarrow \mathbb{P}(\mathcal{S}_v)$, that satisfies the following expression

$$\forall\, a \in \mathcal{A}_v : \{i_v(s) \mid s \in m_v(a)\} \cap \mathcal{D}_v = f_v(a).$$

The function $m_v$ evaluates to the elements of the viewpoint's syntax that are needed to represent the relevant architectural instances in a *sound* and *lucid* way (cf. Section 2.1). This formalization of view hence complies with the term *specification model* introduced by Guizzardi in [12].

A considerable advantage of a multi-viewpoint description of an architecture can only be achieved, if all views can be tied together in a "sensible" way. Our proposal establishes this correlation by connecting the sets $\mathcal{D}_v$ and $\mathcal{S}_v$ of all implemented viewpoints $v$ into one *cohesion set* at architectural description level. This set allocates all instances to their underlying concepts defined in $\mathcal{R}_v$ in the different viewpoints as well as the abstract syntax as context in which these elements are embedded.

With $\mathcal{V}_{ad}$ we denominate the set of all viewpoints realized within an architectural description *ad*. The cohesion set $\mathcal{CS}_{ad}$ of an architectural description is defined as:

$$\mathcal{CS}_{ad} := \langle \mathcal{E}^{sem}_{\mathcal{V}_{AD}}, \mathcal{E}^{syn}_{\mathcal{V}_{AD}} \rangle$$

[10] Resorting to the argumentation at the modeling language, we abstain from detailing the notation function.

with

$$\mathcal{E}^{sem}_{\mathcal{V}_{ad}} := \{\langle s, \{i_v(s)\}\rangle \mid \exists\, v \in \mathcal{V}_{ad} : s \in \mathcal{S}_v \wedge i_v(s) \in \mathcal{R}_v\}$$
$$\mathcal{E}^{syn}_{\mathcal{V}_{ad}} := \{\langle s, \{i_v(s)\}\rangle \mid \exists\, v \in \mathcal{V}_{ad} : s \in \mathcal{S}_v \wedge i_v(s) \in \mathcal{D}_v\}.$$

Finally, we have all ingredients at hand to present the essential characterization for an architectural description $ad(a)$ of an architecture $a$ as the unity of all its views $m_v$ and the cohesion set $\mathcal{CS}_{ad}$, in short

$$ad(a) := \langle \bigcup_{v \in \mathcal{V}_{ad}} m_v(a),\ \mathcal{CS}_{ad}\rangle.$$

In this manner, we are constructing an architectural description strictly bottom up. For a real world case this might be insufficient, if all instances to be modeled are familiar. In this case, it would be possible to create a taxonomy (or even ontology) of instances to be modeled in advance and purport them as presetting to the architects. The conceptualization would then be known from the outset. Since this condition cannot be achieved under all circumstances, we neglected this proceeding. The implicit assumed matching process over all components in our procedure might seem fairly complex, but it assures to detect all structural relationships and semantical embodiments. Based on the cohesion set constituents, as introduced above, we can establish a mechanism to check for validity of the various semantic representations over the different views as well as the embodiment of instances into the different structural environments.

## 4 Conclusion and Outlook

In this paper, we provided a formal approach for describing architectures of systems. This approach builds on basic principles of conceptual modeling, as outlined by Guizzardi in [12] and by Kühn in [14]. It further accounts for the multi-perspectivity of architectural descriptions especially of complex systems, as discussed in the ISO Std. 40201 [4]. Bringing these perspectives together in a formal way seems to us valuable to allow for well-founded discussions on the nature of architectural descriptions in many application fields as e.g. software engineering or enterprise architecture (EA) management. In the former field, other formal approaches relevant to multi-perspective modeling exist, such as the "precise UML" approach initially outlined by Breu et al. in [9] and further developed in subsequent publications. This approach necessarily deals with multi-perspectivity as incorporated in the UML, but can resort itself to less general considerations than our paper, as the approach builds on the concrete language and viewpoints as defined by the modeling technique. In the light of an increasing interest in *domain specific languages* (DSLs) for software engineering, our general considerations, e.g. on compatibility and embedding of concerns, may be helpful to provide formal underpinnings usable during the development of DSLs. The field of EA management presents itself as an even more interesting field of application for our approach, as it up to this point lacks a widely-agreed technique for describing EAs. Some researchers even doubt that such a technique fitting the needs of all companies can be found [16,17]. In this vein, our formal approach may be helpful for the advancement of the field. One may think of the construction of application- or enterprise-specific architectural description techniques and languages, whose internal consistency can be assessed in a more formal way based on the contribution of this paper.

Staying to the application field of EA management, we outline a possibly interesting direction for future research. Buckl et al. proposed in [16] a pattern-based approach to EA management that encompasses a method for constructing appropriate viewpoints for different stakeholders' given concerns in respect to the EA. As part of this approach syntax and semantics corresponding to a concern's underlying conceptualization are made explicit as object-oriented metamodels and textual descriptions, respectively. While this departs from the abstract and intentional nature of the concern, it in contrast allows for more intricate considerations on compatibility and embeddings of concerns based on this paper's considerations. Put in other words, an experienced modeler can check for the compatibility of two object-oriented metamodels in an almost algorithmic manner, while the complementing semantic descriptions ensure that no accidental synonyms invalidate the compatibility check. This formal technique for compatibility checking becomes even more interesting with later versions of the pattern-based approach, which are documented in an online catalog of patterns, of which the V-patterns describe corresponding architectural viewpoints (see [18]). In the course of the refinement of the approach, further V-pattern have been added and new concerns have been described. Complementing this increase in material, also relationships between the patterns have been established (cf. [19]) to reflect that different viewpoints build on each other, i.e., that one viewpoint is "embedded" into another. While up to this point, these whole-part-relationships are derived in an informal and subjective manner, the formal architectural description model presented in this paper may allow for a revision of the relationships. In a long term perspective, an ordering relationship on the concerns and related viewpoints could be derived. With the help of this relationship, a using company could easily understand how their currently selected concerns could be generalized to more comprehensive ones, and could determine, which additional information had to be collected to support this generalization. Thereby, the formal approach towards architectural descriptions would help in the development of a tool support for pattern-based EA management methods.

## References

1. Pollio, V.: Vitruvii De Architectura Libri Decem – Zehn Bücher über Architektur – Übersetzt und mit Anmerkungen versehen von Curt Fensterbusch, 5th edn. Primus-Verlag, Darmstadt (1996)
2. Freestone, R.: Urban Planning in a Changing World: The Twentieth Century Experience (Studies in History, Planning, and the Environment). Spon Press, London (2000)
3. Reussner, R., Hasselbring, W.: Handbuch der Software-Architektur. Dpunkt Verlag, Heidelberg (2006)

4. International Organization for Standardization: Iso/iec 42010:2007 systems and software engineering – recommended practice for architectural description of software-intensive systems (2007)
5. Rechtin, E.: Systems Architecting of Organizations – Why Eagles can't swim. CRC Press LLC, New York (1999)
6. Object Management Group (OMG): Uml 2.2 superstructure specification (formal/2009-02-02) (2009), `http://www.uml.org` (cited 2010-02-25)
7. Burrel, G., Morgan, G.: Sociological Paradigms and Organizational Analysis. Heinemann Educational Publishers, London (1979)
8. Becker, J., Niehaves, B.: Epistemological perspectives on is research: a framework for analysing and systematizing epistemological assumptions. Information Systems Journal 17, 197–214 (2007)
9. Breu, R., Hinkel, U., Hofmann, C., Klein, C., Paech, B., Rumpe, B., Thurner, V.: Towards a formalization of the unified modeling language. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 344–366. Springer, Heidelberg (1997)
10. Dijkman, R.M., Quartel, D.A., van Sinderen, M.J.: Consistency in multi-viewpoint design of enterprise information systems. Information and Software Technology 50(7-8), 737–752 (2008)
11. Mylopoulos, J.: Conceptual modeling and Telos, pp. 49–68. Wiley, New York (1992)
12. Guizzardi, G.: Ontological foundations for structural conceptual models. PhD thesis, CTIT, Centre for Telematics and Information Technology, Enschede, The Netherlands (2005)
13. Gurr, C.A.: On the isomorphism, or lack of it, of representations. In: Visual language theory, New York, NY, USA, pp. 293–305. Springer, Heidelberg (1998)
14. Kühn, H.: Methodenintegration im Business Engineering. PhD thesis, Universität Wien (2004)
15. Buckl, S., Ernst, A.M., Lankes, J., Matthes, F., Schweda, C., Wittenburg, A.: Generating visualizations of enterprise architectures using model transformation (extended version). Enterprise Modelling and Information Systems Architectures – An International Journal 2(2), 3–13 (2007)
16. Buckl, S., Ernst, A.M., Lankes, J., Schneider, K., Schweda, C.M.: A pattern based approach for constructing enterprise architecture management information models. In: Wirtschaftsinformatik 2007, Karlsruhe, Germany, pp. 145–162. Universitätsverlag Karlsruhe (2007)
17. Kurpjuweit, S., Aier, S.: Ein allgemeiner Ansatz zur Ableitung von Abhängigkeitsanalysen auf Unternehmensarchitekturmodellen. In: 9. Internationale Tagung Wirtschaftsinformatik (WI 2007), Wien, Austria, pp. 129–138. Österreichische Computer Gesellschaft (2009)
18. Chair for Informatics 19 (sebis), Technische Universität München: Eam pattern catalog wiki (2009), `http://eampc-wiki.systemcartography.info` (cited 2010-02-25)
19. Ernst, A.M.: A Pattern-Based Approach to Enterprise Architecture Management. PhD thesis, Technische Universität München, München, Germany (2010) (in submission)