DOI 10.1007/s10958-016-2814-1
Journal of Mathematical Sciences, Vol. 214, No. 6, May, 2016

CATEGORY-THEORETIC APPROACH TO SOFTWARE SYSTEMS DESIGN

S. P. Kovalyov UDC 519.68

Abstract. Category theory is applied to the problem of representing heterogeneous software engineering technologies in a unified form suitable for their integration and coordination within the common software systems engineering cycle. Special attention is paid to modern technologies such as model-driven engineering and aspect-oriented programming. Universal category-theoretic semantic models of these technologies are constructed. A novel method of separation of concerns by explicating the aspectual structure of formal models of programs is proposed. We construct and analyze formal technologies (architecture schools) for designing technologies that comprise a mathematical basis for model-driven engineering.

Introduction

Automation of technical products design and manufacturing processes is duly considered the major way to improve the quality of products and efficiency of the labor of engineers. In the materials industry, automatic machines and 3D-printers routinely build products as instructed by digital files generated by CAD (computer-aided design) tools. Modern tools are able to reduce engineering to creation, analysis, and transformation of computer models, viz. simplified formal representations of products reflecting their coarse-grained structure and basic properties. The main role is traditionally played by the geometric and multiphysical models that allow virtual simulation of product behavior under diverse conditions, of assembling composite products from parts of various technological procedures at the design stage.

Virtual simulation and automatic manufacturing can be very useful in software development as well. Tools for modeling programs, analysis of models, and source code generation (CASE-tools, computer aided software engineering) have been actively developed for over half a century. However, the effect of their implementation is much smaller than from the effect of CAD-tools [34]. This is largely due to a special intangible nature of software that unlike material products has no geometric or physical characteristics. Software products are characterized by a wide set of specific indicators, often nonnumerical, difficult to formalize, having different meaning depending on the subject domain of concerns intended to be resolved by the product. Examples of such indicators are flexibility, stability, and correctness [19]. A large number of diverse indicators lead to the emergence of many diverse program modeling languages and techniques, impeding the creation of truly general-purpose CASE-tools. There exist object-oriented, structured, scenario-based, automata, algebraic, logical, denotational, and many other kinds of program models.

A novel approach to automate software engineering processes called MDE (model-driven engineering) was proposed in the mid-2000's in order to efficiently support such diversity of models in the software systems engineering cycle [33]. This approach offers to facilitate modeling by creating various formalized DSLs (domain-specific languages). Languages are constructed by specialization at the level of metamodels from general-purpose language such as the object-oriented modeling language UML (Unified Modeling Language) [15], real-time languages [5], logical inference languages [37], and so on. Instruments are provided to create automatic tools for creating, analyzing, assembling, and transforming models, including the source code generation. A large set of instruments and tools have been created and published in the open source Eclipse Modeling Project [9].

However, MDE technologies appear to be poorly scalable: labor costs associated with their use too quickly increase with the size and structural complexity of software systems [23]. Particular difficulties

Translated from Fundamentalnaya i Prikladnaya Matematika, Vol. 19, No. 3, pp. 111–170, 2014.

during systems development are associated with the coordination of consistent modifications in numerous interrelated models responsible for different parts or aspects of the system and written in different languages [20]. The consistency conditions are detected by tracing stakeholder concerns (requirements), viz. tracking their implementation in diverse models and programs. The more completely and detailedly tracing is performed, the easier it is to locate modifications in response to newly occurring requirements and to avoid side effects that can undermine compliance with other requirements. Tracing is known as one of the most expensive operations in software engineering [2]. The major problem is that software developers are not accustomed to consider the ease of tracing a mandatory quality criterion of results they produce [21]. Commonly used languages, technologies, and styles of software design and programming allow leaving the purpose of fragments of models and programs "implicit," freeing authors from having to meticulously record the traceability links. A known exception is AOP (aspect-oriented programming) [22], which supports automatic tracing of auxiliary programmatic concerns such as logging system operation, security, and so on. Pieces of code that implement such concerns are scattered across modular architecture units and tangled with software implementation of the primary tasks that are localized in the modules. AOP technologies allow shaping such crosscutting concerns as so-called aspects, viz. special independent program units that are automatically embedded into the modules at explicitly specified points providing traceability "by construction."

Of great interest is application of AOP in MDE technologies because during the development of models involving an aspect-oriented approach, detailed traceability links emerge [16]. However, such application is hindered by lack of an AOP conceptual basis whose implementations are tied to particular program composition and modeling technologies [36]. It is unclear how to extend an arbitrary software systems design technology of MDE flavor by efficient techniques to create models of aspects and embed them into models of modules.

A more general relevant problem is that of representing various software engineering technologies in unified form convenient for integrating and coordinating them within the common design cycle of complex heterogeneous systems. The choice of mathematical tools for modeling and analysis of software engineering processes with a sufficient level of rigor plays an important role in solving this problem. The traditional approach to the mathematical modeling by means of differential equations and minimizable functionals developed in physics and other sciences does not help there due to an absence of suitable analogues for variational principles, conservation laws, statistical regularities and the like [34]. An alternative approach exists based on the observation that the history of assembling from certain primitive components is available (or easily retrievable) for the majority of systems. If mathematical models of the components and system development activities are known, then it is possible to calculate integral characteristics of the system over formal analogues of assembly drawings or "megamodels," viz. directed graphs (diagrams) with nodes labeled by designations of the components and edges labeled by designations of activities. Here one needs to generate and process graphs that are too large even to be depicted as a whole and so could be described only by structural constraints. Powerful means for constructing and analyzing graphs of this type have been developed within the framework of category theory, viz. a part of higher algebra that "starts with the observation that many properties of mathematical systems can be unified and simplified by a presentation with diagrams" [29, p. 1]. System units (components, subsystems, systems, etc.) are represented by objects of suitable categories, activities are represented by morphisms, and complex technological procedures are represented by diagrammatic constructions [7]. For structural coordination of diverse procedures defined in different categories of such kind, suitable functors between these categories are introduced.

For a comprehensive study of software systems engineering processes by means of category theory a special general construction of a formal design technology (architecture school) was proposed [11]. In the present paper, in the course of advance of the results of [26], constructions of this kind are treated as objects of a category whose morphisms represent developing and transforming technologies including the creation of domain-specific languages and modeling tools. In particular, enhancing a technology by traceability and AOP techniques is formally described as its transformation that consists in equipping models with additional structure that represents labeling by concerns at the level of integration interfaces. If the labeling can be "lifted" to the level of models themselves (explicated), then aspects that constitute the model can be extracted from it as modular units by means of tracing, thus producing full separation of concerns. Such an approach to the formalization of MDE and AOP is illustrated by examples from a variety of software systems design technologies. Of course, many other formal approaches can be found in the literature (see [8] and others) but they are presented in terms of specific theoretical computer science formalisms (model checking, lambda calculus, etc.) and, therefore, can be used only in specific technologies.

The paper is organized as follows. In Sec. 1, the notion of a formal technology is introduced. Section 2 is devoted to tracing concerns and transforming modular technologies into aspect-oriented ones. In Sec. 3, constructions of aspect weaving and explication of aspectual structure are described. In Sec. 4, we introduce and analyze formal technologies for design of formal technologies. In Sec. 5 the design of domain-specific technologies with ample support for tracing is considered. We conclude with a summary of results and directions for future research.

1. Category-Theoretic Description of Software Engineering

Definitions of category-theoretic concepts used in this paper can be found in [1,29]. Specifically, in [1] objects and morphisms of a category C are concisely called C-objects and C-morphisms, respectively, and a functor $\Delta\colon X \to C$ is called a C-diagram with schema X (X is assumed to be a small category). A diagram is considered as a graph of category X with nodes labeled by C-objects and edges labeled by C-morphisms. The largest discrete subdiagram in Δ obtainable by removing all nonidentity X-morphisms is denoted as $|\Delta|$ and its schema is denoted as $|X|$. All C-diagrams comprise the category $\mathbf{D}C$ (a covariant "supercomma" category [29], viz. a Grothendieck flattening construction for the functor $C^{-}\colon X \mapsto C^{X}$) in which a morphism of a diagram $\Delta\colon X \to C$ to $\Xi\colon Y \to C$ is a pair $\langle \varepsilon, \mathrm{fd}\rangle$ consisting of a functor $\mathrm{fd}\colon X \to Y$ and a natural transformation $\varepsilon\colon \Delta \to \Xi \circ \mathrm{fd}$. Every functor $\mathrm{fun}\colon C \to D$ induces a functor

$$\begin{aligned} \mathrm{fun} \circ -\colon & \mathbf{D}C \to \mathbf{D}D \\ & \colon \Delta \mapsto \mathrm{fun} \circ \Delta,\ \langle \varepsilon, \mathrm{fd}\rangle \mapsto \langle \mathrm{fun}(\varepsilon), \mathrm{fd}\rangle, \end{aligned}$$

whence there exists an endofunctor

$$\begin{aligned} \mathbf{D}\colon & \mathbf{CAT} \to \mathbf{CAT} \\ & \colon C \mapsto \mathbf{D}C,\ \mathrm{fun} \mapsto (\mathrm{fun} \circ -), \end{aligned}$$

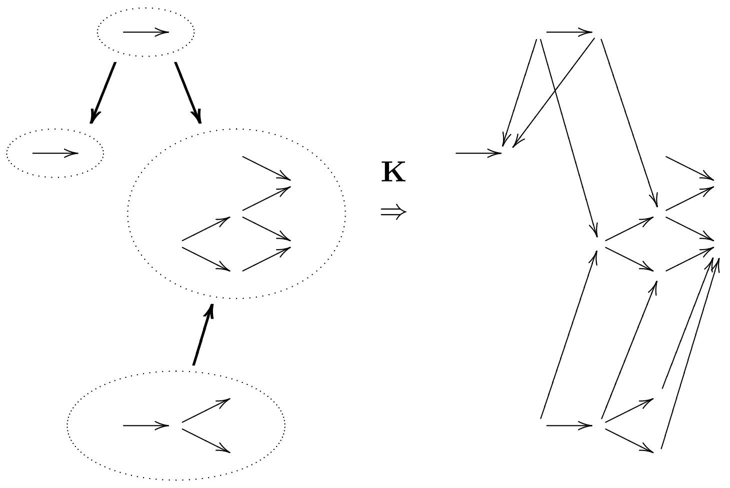
where **CAT** denotes the category of all categories and all functors.

Let $\boldsymbol{1}$ be a terminal **CAT**-object which is a category consisting of a single object 0 and a single morphism 1_0. A full embedding $\ulcorner - \urcorner\colon C \hookrightarrow \mathbf{D}C$ exists, which maps each C-object S to the diagram $\ulcorner S \urcorner\colon \boldsymbol{1} \to C\colon 0 \mapsto S$, which is called a dot. A $\mathbf{D}C$-morphism with a dot as a codomain is called a cocone. A colimit of a diagram Δ is a cocone denoted as $\mathrm{colim}\,\Delta\colon \Delta \to \ulcorner S \urcorner$ which is universal in the sense that for every C-object T and cocone $\delta\colon \Delta \to \ulcorner T \urcorner$ there exists a unique C-morphism $w\colon S \to T$ (colimit arrow) such that $\delta = \ulcorner w \urcorner \circ \mathrm{colim}\,\Delta$. A colimit is unique up to an isomorphism provided that it exists: a cocone δ is a colimit of a diagram Δ if and only if there exists a C-isomorphism i such that $\delta = \ulcorner i \urcorner \circ \mathrm{colim}\,\Delta$.

Let Cdia be a class of C-diagrams each of which has a colimit. We can consider it as a full subcategory of $\mathbf{D}C$ and define the colimit functor colim that acts from it to C by mapping each diagram from Cdia to a vertex of its colimit and each $\mathbf{D}C$-morphism $\theta\colon \Delta \to \Xi$ for some $\Delta, \Xi \in$ Cdia to a colimit arrow $\mathrm{colim}(\theta)$ that satisfies the condition $\mathrm{colim}\,\Xi \circ \theta = \ulcorner \mathrm{colim}(\theta) \urcorner \circ \mathrm{colim}\,\Delta$.

$$\begin{tikzcd} \Delta \arrow[r, "\mathrm{colim}\,\Delta"] \arrow[d, "\theta"'] & \ulcorner \mathrm{colim}(\Delta) \urcorner \arrow[d, dotted, "\ulcorner \mathrm{colim}(\theta) \urcorner"] \\ \Xi \arrow[r, "\mathrm{colim}\,\Xi"] & \ulcorner \mathrm{colim}(\Xi) \urcorner \end{tikzcd}$$

Colimits and other diagrammatic structures specified in the category $\mathbf{D}C$ are routinely sketched as C-diagrams for illustrative purposes. For example, a cocone $\nu\colon \Delta \to \ulcorner P \urcorner$ can be sketched by "adding" an extra node P and arrows directed to it from every node of a sketch of a base diagram Δ. This technique can be formalized and generalized as follows. First, a cocone is considered as an obvious $\mathbf{D}C$-diagram with schema $\mathit{2}$ (recall that $\mathit{2}$ denotes the category that consists of two objects 0, 1, their identity morphisms, and a single nonidentity morphism $0 \to 1$). Next, every $\mathbf{D}C$-diagram can be turned into a C-diagram by the canonical natural drawing functor $\mathbf{K}\colon \mathbf{DD}C \to \mathbf{D}C$. The functor $\mathbf{K}$ acts on an arbitrary diagram $\Gamma\colon Z \to \mathbf{D}C$ by replacing each node A of a graph of schema Z by a graph of the C-diagram $\Gamma(A)$ and each edge $f\colon A \to B$ by a collection of edges, one per node I of a graph of the diagram $\Gamma(A)$, directed from I to the node $\mathrm{fd}(I)$ of a graph of the diagram $\Gamma(B)$ and labeled by the C-morphism ε_I, where $\langle \varepsilon, \mathrm{fd} \rangle = \Gamma(f)$. A rigorous definition of the drawing functor is given in [17] and a triple $\langle \mathbf{D}, \ulcorner - \urcorner, \mathbf{K} \rangle$ is proved there to define a monad in $\mathbf{CAT}$ which has certain properties similar to powerset monad $\langle 2^{-}, \{-\}, \cup \rangle$ in the category $\mathbf{Set}$ of all sets and all maps.



For example, the result of "gluing" a cocone $\nu\colon \Delta \to \ulcorner P \urcorner$ to a C-diagram Ξ that contains a node P can be drawn as a C-diagram $\mathbf{K}\Pi$, where Π is a $\mathbf{D}C$-diagram with schema $\mathit{2}$ that represents a $\mathbf{D}C$-morphism $\pi \circ \nu\colon \Delta \to \Xi$ with $\pi\colon \ulcorner P \urcorner \hookrightarrow \Xi$ being an embedding. Note that if the diagram Ξ has a colimit, then gluing a cocone leaves it intact (more rigorously: there exists a $\mathbf{D}C$-morphism $\rho\colon \mathbf{K}\Pi \to \Xi$ such that $\operatorname{colim} \mathbf{K}\Pi = (\operatorname{colim} \Xi) \circ \rho$ and $\rho \circ \chi = 1_{\Xi}$, where $\chi\colon \Xi \hookrightarrow \mathbf{K}\Pi$ is a canonical subdiagram embedding). We will also employ a diagrammatic structure called a singular patch and defined as a drawing of a $\mathbf{D}C$-diagram that consists of a pair of arrows with common beginning $\varphi\colon \Phi \leftarrow \ulcorner \mathbf{1} \urcorner \to \Theta :\theta$, where $\mathbf{1}$ denotes a terminal C-object. A colimit of a patch has the same vertex as a colimit of the C-diagram $s\colon \operatorname{colim}(\Phi) \leftarrow \mathbf{1} \to \operatorname{colim}(\Theta) :t$, where $\ulcorner s \urcorner = (\operatorname{colim} \Phi) \circ \varphi$ and $\ulcorner t \urcorner = (\operatorname{colim} \Theta) \circ \theta$ (provided that all these colimits exist).

We will consider how various functors act upon colimits. According to [1, Sec. 13], a functor $\mathrm{fun}\colon C \to D$:

- preserves colimits of a diagram Δ if for every colimit $\delta\colon \Delta \to \ulcorner S \urcorner$ cocone $\mathrm{fun} \circ \delta\colon \mathrm{fun} \circ \Delta \to \ulcorner \mathrm{fun}(S) \urcorner$ is a colimit of the diagram $\mathrm{fun} \circ \Delta$;
- lifts colimits of a diagram Δ if for every colimit $\xi\colon \mathrm{fun} \circ \Delta \to \ulcorner T \urcorner$ there exists a colimit $\delta\colon \Delta \to \ulcorner S \urcorner$ such that $\mathrm{fun} \circ \delta = \xi$.

According to [26], a functor that preserves and lifts colimits of a diagram Δ is said to *determine* them provided that Δ has a colimit. For example, a projection functor from a product of categories to each component determines colimits of all diagrams that have them. The same is true for an injection functor from each component to a sum of categories.

The category-theoretic approach to formalizing software engineering is steadily evolving, starting from early 1970's (see, e.g., [13]). Categories that are considered primarily have formal models of programs as objects and formal descriptions of various activities of their integration as morphisms. Composition of morphisms represents execution of multi-step activities (processes), and identical morphisms represent "doing nothing." We will denote the category of this kind by c-DESC. We do not apply any restrictions on it *a priori*, since there is no reason to exclude the possibility that any category describes some (maybe not yet invented) software engineering technology.

Assembling systems is described by colimits of configurations, or formal "megamodels" [20] which are c-DESC-diagrams that consist of components and their integration interrelations. For example, consider *connection* of a component P with a system S, viz. a technique of assembling by adding an extra component G called "glue" or connector [3] capable of being integrated both with a component and a system. Connection is used in software design to integrate systems by means of a middleware which is employed as a connector. A megamodel of the connection is represented by a pair of morphisms $f\colon P \leftarrow G \rightarrow S :g$ denoted as Δ. A cocone over Δ is a commutative square which is determined by a vertex object V and a pair of edge morphisms $p\colon P \rightarrow V \leftarrow S :s$ that satisfy the condition $p \circ f = s \circ g$. For a cocone to be a colimit (and to be called a pushout) the following universal condition shall hold: for every object T and a pair of morphisms $u\colon P \rightarrow T \leftarrow S :v$ that determine a cocone $\delta\colon \Delta \rightarrow \ulcorner T \urcorner$ (i.e., satisfy the condition $u \circ f = v \circ g$) there exists a unique morphism $w\colon V \rightarrow T$ such that $w \circ p = u$ and $w \circ s = v$. This is a specific instance of the condition $\delta = \ulcorner w \urcorner \circ \operatorname{colim} \Delta$ that holds by the definition of a colimit. It induces a triangular **D**c-DESC-diagram whose drawing is usually visualized as follows:

$$\begin{tikzcd}
G \arrow[r, "g"] \arrow[d, "f"'] & S \arrow[d, dashed, "s"] \arrow[ddr, "v"] & \\
P \arrow[r, dashed, "p"] \arrow[drr, "u"'] & V \arrow[dr, dotted, "w"] & \\
 & & T.
\end{tikzcd}$$

Clearly the object V represents the system assembled from S and P by connection via G (without any extra components) provided that the pushout exists. Otherwise we gather that the glue G is unable to connect the component P with S with activities f and g.

Concepts of an interface and a refinement need to be formalized in order to conduct comprehensive studies of software systems engineering processes by means of category theory [11]. An interface is a part of a component that is "seen" by a system during integration, such as a declaration of a web service specified in the WSDL language. A refinement is a development step of an individual component, such as an implementation of the web service's specification by means of a programming language. Formal models of interfaces comprise a category denoted SIG (since they are often referred to as signatures in programming) and interface extraction is formalized as a functor $\mathrm{sig}\colon \text{c-DESC} \rightarrow \mathrm{SIG}$. Clearly the functor sig shall not be injective on objects since different models can have the same interface. However, sig-images of two different activities of integrating the same component into the same system must be different since otherwise interfaces happen to describe integrational capabilities of components with inappropriately low precision. It follows that the functor sig shall be faithful, viz. injective on each set $\mathrm{Mor}(A, B)$, $A, B \in \mathrm{Ob}\,\text{c-DESC}$. In addition, interfaces shall be implementable, i.e., a discrete interface implementation functor $\mathrm{sig}^*\colon \mathrm{SIG} \rightarrow \text{c-DESC}$ shall exist, such as $\mathrm{sig} \circ \mathrm{sig}^* = 1_{\mathrm{SIG}}$, and there exists a bijection $\mathrm{sig}\colon \mathrm{Mor}(\mathrm{sig}^*(I), S) \cong \mathrm{Mor}\big(I, \mathrm{sig}(S)\big)$ for each $I \in \mathrm{Ob}\,\mathrm{SIG}$, $S \in \mathrm{Ob}\,\text{c-DESC}$ (category theory provides a concise wording of this requirement in terms of adjunction of functors). For example, discrete implementation of a WSDL-declaration of a web-service consists of stubs (empty procedures) so it can be automatically generated by CASE-tools and used for rapid building debug versions of applications.

Natural correlation between interfaces and configurations exists. Two c-DESC-diagrams that have the same **D**sig-image shall both either belong to the class Conf or not. This requirement states a kind of logical noncontradiction law for interfaces: valid configurations shall be properly recognizable at interface level. In addition, interface extraction shall be natural with respect to systems assembling in the "strong" sense that a colimit of **D**sig-image of every configuration shall be **D**sig-image of a colimit of the configuration (i.e., the functor sig shall lift colimits of configurations). We will see later (Proposition 2) that this property of the functor sig along with other properties imply naturality in a "weak" sense more customary in category theory: the functor sig preserves colimits of configurations (in other words, **D**sig-image of a colimit of every configuration is a colimit of **D**sig-image of the configuration). In summary, the functor sig determines colimits of all configurations.

As far as refinements are concerned, they are described by morphisms in an appropriate category denoted as r-DESC, which generally differs from c-DESC although it has the same class of objects and contains all their isomorphisms from c-DESC. Refinements are required to be natural with respect to systems assembling in the sense that every set of refinements of objects of every configuration shall induce a refinement of a system assembled from it as a whole. This requirement can be formalized by means of natural transformations consisting of r-DESC-morphisms since every discrete c-DESC-diagram is also an r-DESC-diagram. An arbitrary set of refinements of objects of a diagram $\Delta\colon X \to \text{c-DESC}$ is precisely a natural transformation (an $\text{r-DESC}^{|X|}$-morphism) of the discrete diagram $|\Delta|$ to discrete diagram Σ that consists of all refinements results. If $\Delta \in \text{Conf}$, then diagram Σ shall be contained in some configuration whose colimit vertex can be obtained from $\text{colim}(\Delta)$ by a refinement.

Considerations above lead to the following complex structure suitable for formally exploring software systems engineering processes from many different viewpoints [11].

Definition 1. Let c-DESC be a category whose objects are formal models of programs and morphisms are formal descriptions of activities of integration of software systems. A *formal design technology* (architecture school) over c-DESC is a quadruple $\langle \text{c-DESC}, \text{Conf}, \text{sig}, \text{r-DESC}\rangle$, where:

- Conf is a class of c-DESC-diagrams, which are called valid configurations of systems;
- sig is a functor from c-DESC to a category denoted as SIG, whose objects are called interfaces or signatures and morphisms are called formal descriptions of activities of integration of interfaces;
- r-DESC is a category, whose objects are formal models of programs and morphisms are called formal descriptions of refinements

such that the following conditions hold.

(i) Every diagram from the class Conf has a colimit.
(ii) The functor sig is faithful.
(iii) The functor sig has a left adjoint denoted as sig^* with an identity as the adjunction unit.
(iv) The functor sig lifts colimits of all diagrams from the class Conf.
(v) For any c-DESC-diagrams Δ and Ξ the conditions $\text{sig} \circ \Delta = \text{sig} \circ \Xi$ and $\Delta \in \text{Conf}$ imply that $\Xi \in \text{Conf}$.
(vi) $\text{Ob}\,\text{r-DESC} = \text{Ob}\,\text{c-DESC}$.
(vii) The subcategory of c-DESC that consists of all c-DESC-objects and all isomorphisms is a subcategory of r-DESC.
(viii) For every diagram $\Delta\colon X \to \text{c-DESC} \in \text{Conf}$ and every natural transformation of the kind $\varphi\colon |\Delta| \to \Sigma \in \text{Mor}\,\text{r-DESC}^{|X|}$ there exists a diagram $\Delta \oplus \varphi \in \text{Conf}$ containing Σ as a subdiagram and an r-DESC-morphism $r\colon \text{colim}(\Delta) \to \text{colim}(\Delta \oplus \varphi)$.

We derive the following two extra structures.

Definition 2. A triple $\langle \text{c-DESC}, \text{Conf}, \text{sig}\rangle$ that satisfies the conditions (i)–(v) above is called a *formal specification technology*, and a pair $\langle \text{c-DESC}, \text{Conf}\rangle$ that satisfies the condition (i) is called a *formal configuration technology.*

Now we are going to prove that interface extraction commutes with systems assembling, as stated above. We will employ a minor technical statement, which we prove first.

Proposition 1. *An arbitrary functor* fun *determines colimits of a diagram* Δ *if and only if it lifts colimits and the diagram* $\text{fun} \circ \Delta$ *has a colimit.*

Proof. The "if" part is obvious, so we prove the "only if" part. Let δ be an arbitrary colimit of the diagram Δ, and let ξ be an arbitrary colimit of the diagram $\mathrm{fun} \circ \Delta$. By assumption, diagram Δ has a colimit δ' with $\mathrm{fun} \circ \delta' = \xi$. We have that $\delta = \ulcorner i \urcorner \circ \delta'$ for some isomorphism i, hence cocone $\mathrm{fun} \circ \delta = \ulcorner \mathrm{fun}(i) \urcorner \circ \xi$ is a colimit of the diagram $\mathrm{fun} \circ \Delta$. We see that the functor fun preserves colimits of the diagram Δ; hence it determines them. □

Proposition 2. *In every formal specification or design technology, the functor* sig *preserves colimits of all configurations.*

Proof. Let η and ε be a unit and a counit of the adjunction $\mathrm{sig}^* \dashv \mathrm{sig}$ given by condition (iii) of Definition 1. Recall that η is a natural transformation of functor 1_{SIG} to $\mathrm{sig} \circ \mathrm{sig}^* = 1_{\mathrm{SIG}}$ that consists of identity SIG-morphisms while ε is a natural transformation of the functor $\mathrm{sig}^* \circ \mathrm{sig}$ to $1_{\text{c-DESC}}$. According to the standard definition of a counit via a unit [29, p. 82], the condition $\mathrm{sig}(\varepsilon_S) \circ \eta_{\mathrm{sig}(S)} = 1_{\mathrm{sig}(S)}$ holds for every $S \in \mathrm{Ob}\,\text{c-DESC}$, so $\mathrm{sig}(\varepsilon_S) = 1_{\mathrm{sig}(S)}$. Since every faithful functor reflects monomorphisms [1, Proposition 7.37(2)], condition (ii) of Definition 1 implies that ε_S is a monomorphism. (At the same time, ε_S is an epimorphism since every faithful functor reflects epimorphisms as well [1, Proposition 7.44].)

Choose an arbitrary diagram $\Delta \in \mathrm{Conf}$ and let $\Delta^* = \mathrm{sig}^* \circ \mathrm{sig} \circ \Delta$. Since $\mathrm{sig} \circ \Delta^* = \mathrm{sig} \circ \Delta$, condition (v) of Definition 1 implies $\Delta^* \in \mathrm{Conf}$. Hence, by condition (i) of Definition 1, there exists a colimit of the kind $\sigma\colon \Delta^* \to \ulcorner S \urcorner$. Consider cocone $\sigma^* = \mathrm{sig}^* \circ \mathrm{sig} \circ \sigma\colon \Delta^* \to \ulcorner S^* \urcorner$ where $S^* = \mathrm{sig}^*\big(\mathrm{sig}(S)\big)$. Let $u\colon S \to S^*$ be a colimit arrow that satisfies the condition $\sigma^* = \ulcorner u \urcorner \circ \sigma$. Since $\sigma = \ulcorner \varepsilon_S \urcorner \circ \sigma^*$, we have $\varepsilon_S \circ u = 1_S$, so ε_S as a right-invertible monomorphism is an isomorphism [1, Proposition 7.36]. Hence σ^* is a colimit of the diagram Δ^*. Since the functor sig^* induces full embedding of category SIG into c-DESC, cocone $\mathrm{sig} \circ \sigma^* = \mathrm{sig} \circ \sigma$ is a colimit of the SIG-diagram $\mathrm{sig} \circ \Delta^* = \mathrm{sig} \circ \Delta$. So the diagram $\mathrm{sig} \circ \Delta$ has a colimit. Hence, by condition (iv) of Definition 1 and Proposition 1, the functor sig preserves colimits of diagram Δ. □

This result allows determining all diagrams capable to be used as configurations. A SIG-diagram Θ is called sig-*preconfiguration* if it has a colimit and the functor sig lifts colimits of all diagrams from class $\mathbf{D}\mathrm{sig}^{-1}(\{\Theta\}) = \{\Delta \mid \mathrm{sig} \circ \Delta = \Theta\}$. It is easy to see that the class Conf can always be represented as $\mathbf{D}\mathrm{sig}^{-1}(\mathrm{IC})$ for some class IC of sig-preconfigurations [26, Proposition 6].

As stated in the Introduction, formal technologies are themselves considered as objects of categories whose constructions represent procedures of developing complex technologies. Engineering of technologies will be considered in detail in Sec. 4. As an introduction to it a notion of morphism for formal technologies is defined following the ideas of [26, Sec. 6].

Definition 3. A *morphism of a formal design technology*

$$\langle \text{c-DESC}_1,\ \mathrm{Conf}_1,\ \mathrm{sig}_1\colon \text{c-DESC}_1 \to \mathrm{SIG}_1,\ \text{r-DESC}_1 \rangle$$

to

$$\langle \text{c-DESC}_2,\ \mathrm{Conf}_2,\ \mathrm{sig}_2\colon \text{c-DESC}_2 \to \mathrm{SIG}_2,\ \text{r-DESC}_2 \rangle$$

is a triple of functors

$$\langle \mathrm{cm}\colon \text{c-DESC}_1 \to \text{c-DESC}_2,\ \mathrm{sm}\colon \mathrm{SIG}_1 \to \mathrm{SIG}_2,\ \mathrm{rm}\colon \text{r-DESC}_1 \to \text{r-DESC}_2 \rangle$$

that satisfies the following conditions:

(i) $\mathrm{cm} \circ \mathrm{Conf}_1 \subseteq \mathrm{Conf}_2$;
(ii) cm preserves colimits of all diagrams from the class Conf_1;
(iii) $\mathrm{sig}_2 \circ \mathrm{cm} = \mathrm{sm} \circ \mathrm{sig}_1$;
(iv) $\mathrm{rm}(i) = \mathrm{cm}(i)$ for each $i \in \mathrm{Iso}\,\text{c-DESC}_1$.

A pair of functors $\langle \mathrm{cm}, \mathrm{sm} \rangle$ that satisfies conditions (i)–(iii) is called a *morphism of a formal specification technology* $\langle \text{c-DESC}_1, \mathrm{Conf}_1, \mathrm{sig}_1 \rangle$ to $\langle \text{c-DESC}_2, \mathrm{Conf}_2, \mathrm{sig}_2 \rangle$. A functor cm that satisfies conditions (i)–(ii) is called a *morphism of a formal configuration technology* $\langle \text{c-DESC}_1, \mathrm{Conf}_1 \rangle$ to $\langle \text{c-DESC}_2, \mathrm{Conf}_2 \rangle$.

If all constituents of a morphism of technologies are embeddings of subcategories, then the domain of the morphism is called a *subtechnology* of its codomain.

$$\begin{tikzcd}
\langle \text{c-DESC}_1, \arrow[d, "\text{cm}"'] & \text{Conf}_1, \arrow[d, phantom, "\mathbf{D}\text{cm} \mid\!\cap"] & \text{c-DESC}_1 \arrow[r, "\text{sig}_1"] \arrow[d, "\text{cm}"'] & \text{SIG}_1, \arrow[d, "\text{sm}"] & \text{r-DESC}_1\rangle \arrow[d, "\text{rm}"] \\
\langle \text{c-DESC}_2, & \text{Conf}_2, & \text{c-DESC}_2 \arrow[r, "\text{sig}_2"] & \text{SIG}_2, & \text{r-DESC}_2\rangle
\end{tikzcd}$$

Requirements to functors that comprise a morphism of technologies guarantee naturality of morphism action with respect to systems assembling (conditions (i)–(ii)), extraction of interfaces (condition (iii)), and refinements (condition (iv)). Naturality with respect to componentwise refinement of configurations (condition (viii) of Definition 1) holds in the following sense. For every diagram $\Delta\colon X \to \text{c-DESC} \in \text{Conf}_1$ and every natural transformation of the kind $\varphi\colon |\Delta| \to \Sigma \in \operatorname{Mor}\text{r-DESC}_1^{|X|}$, family $\text{rm}(\varphi)$ refines all objects of diagram $\text{cm}\circ\Delta \in \text{Conf}_2$, and diagram $\text{cm}\circ(\Delta\oplus\varphi)$ can be considered as an ultimate refinement result (i.e., taken for $(\text{cm}\circ\Delta)\oplus\text{rm}(\varphi)$) with the r-DESC_2-morphism $\text{rm}(r)$ as an appropriate refinement of its colimit vertex. Indeed, we have $\text{rm}(\varphi)\colon |\text{cm}\circ\Delta| \to \text{cm}\circ\Sigma$. Diagram $\text{cm}\circ(\Delta\oplus\varphi)$ contains subdiagram $\text{cm}\circ\Sigma$ and, since the functor cm preserves colimits of diagrams Δ and $\Delta\oplus\varphi$, we have $\text{rm}(r)\colon\ \text{colim}(\text{cm}\circ\Delta) \to \text{colim}\big(\text{cm}\circ(\Delta\oplus\varphi)\big)$.

Consider several examples of formal technologies. An arbitrary category C generates the trivial formal design technology $\mathbf{triv}(C) = \langle C, \varnothing, 1_C, (\operatorname{Ob}C, \operatorname{Iso}C)\rangle$ so that mapping **triv** can be obviously extended to a functor from **CAT** to category **ARCH** comprised of all formal design technologies and all their morphisms. Among nontrivial formal technologies we are interested in technologies "over" category **Set** that emerge as a result of formalizing many familiar software program modeling methods. Here sets with some structure (algebraic systems, graphs, etc.) are employed as models and maps compatible with the structure are employed as descriptions of activities of integration. sig is a canonical functor that sends a model to its underlying set by "forgetting" the structure, and left adjoint to it creates "minimal" (discrete) structure on a set. Antifunctional relations are often employed as refinements that "expand" underlying set elements to subsets. For example, consider the formal technology for discrete event modeling [24] defined as

$$\text{SM} = \langle \mathbf{Pos}, \text{CPos}, |-|, \text{r-Pos}\rangle$$

where:

- **Pos** is a category of all partially ordered sets and all monotonic maps (a model here is a scenario, viz. a set of events partially ordered by causal relations);
- CPos is the smallest class of **Pos**-diagrams that contains all dots and is closed under gluing of cocones, formation of coproducts, and singular patches (complex scenarios can be assembled by means of pairwise synchronization of constituents in at most one event in order to prevent violation of causality);
- $|-|\colon \mathbf{Pos} \to \mathbf{Set}\colon S \mapsto |S|$ is a canonical functor that "forgets" order (external observer of a scenario "sees" a set of events comprising it but does not see how they are causally ordered; in particular, every procedure of scenario integration can be specified by mapping events);
- r-Pos is a category with all partially ordered sets as objects in which a morphism of X to Y is a total antifunctional relation $R \subseteq X \times Y$ that satisfies the condition

  $$\forall\, x, x' \in X\ \forall\, y, y' \in Y\ \ \big((xRy \wedge x'Ry' \wedge x \neq x') \Rightarrow (x \leq x' \Leftrightarrow y \leq y')\big)$$

  (refinement of a scenario consists in expanding events to subscenarios with full inheritance of order).

Left adjoint to the functor of extracting interfaces $|-|\colon \mathbf{Pos} \to \mathbf{Set}$ is the discrete ordering functor $\text{dord}\colon \mathbf{Set} \to \mathbf{Pos}\colon\ I \mapsto \langle I, =\rangle$. The technology SM can be used as an example to show how "rigid" is a formal construction of an interface: there exists a unique (up to an equivalence of categories) nontrivial functor of extracting scenario interfaces that satisfies conditions (ii)–(iii) of Definition 1. Indeed, let psig be a functor with domain **Pos** that satisfies said conditions, let ω be a counit of adjunction $\mathrm{psig}^* \dashv \mathrm{psig}$. The pair $\Big(\mathrm{psig}^*\big(\mathrm{psig}(\mathrm{Ob}\,\mathbf{Pos})\big), \mathrm{psig}^*\big(\mathrm{psig}(\mathrm{Mor}\,\mathbf{Pos})\big)\Big)$ is a full subcategory of **Pos**, denoted as PPos, which is isomorphic to a codomain of the functor psig. Since ω consists of monotonic bijections, PPos contains a discretely ordered set of arbitrary cardinality. If PPos also contains a set X equipped with an order different from equality, then we will show that ω consists of isomorphisms. Indeed, there exist elements $a, b \in X$ such that $a < b$ (i.e., $a \neq b$ and $a \leq b$). Choose arbitrary **Pos**-object Y and elements $p, q \in \mathrm{psig}^*\big(\mathrm{psig}(Y)\big)$ that satisfy the condition $\omega_Y(p) \leq \omega_Y(q)$. Define map $f\colon X \to Y$ as follows:

$$f(x) = \begin{cases} \omega_Y(p), & x \leq a, \\ \omega_Y(q) & \text{otherwise.} \end{cases}$$

Clearly, the map f is monotonic, so by the definition of adjunction there exists a monotonic map $f^*\colon X \to \mathrm{psig}^*\big(\mathrm{psig}(Y)\big)$ with $\omega_Y \circ f^* = f$, so $p = f^*(a) \leq f^*(b) = q$. Since p and q are chosen arbitrarily, ω_Y is an isomorphism. Consequently, the category PPos either consists of discretely ordered sets and is equivalent to the category **Set** or is equivalent to the category **Pos**.

As a more advanced example consider the formal technology of automated development of algebraic specifications and ontologies SPECWARE [35]. In it specifications are written in classic multi-sorted first-order language (extended by certain higher-order constructions). A signature of each employed language is a finite set of sort symbols and multi-sorted functional symbol ranks (types). A signature morphism is a map of signatures $\phi\colon \sigma \to \sigma'$ consistent with ranks in the sense that for each rank $f\colon S_1, \ldots, S_k \to S_0 \in \sigma$ we have $\phi(f)\colon \phi(S_1), \ldots, \phi(S_k) \to \phi(S_0)$. All finite multi-sorted signatures and all signature morphisms comprise a subcategory of **Set** denoted as MSS.

Signature morphisms are "lifted" to the level of specifications as follows. Let SLS_σ be the set of all specifications, viz. finite sets of statements of language of signature σ. Denote by SLS a category with a class of objects $\bigcup\limits_{\sigma \in \mathrm{Ob}\,\mathrm{MSS}} \mathrm{SLS}_\sigma$ and $\mathrm{Mor}(T, T') = \{\phi\colon \sigma(T) \to \sigma(T') \mid T' \vdash \phi T\}$ for all SLS-objects T and T'. Standard map composition law turns it into a category since the conditions $T' \vdash \phi T$ and $T'' \vdash \vartheta T'$ for arbitrary specifications T, T', and T'' and arbitrary signature morphisms $\phi\colon \sigma(T) \to \sigma(T')$ and $\vartheta\colon \sigma(T') \to \sigma(T'')$ imply $T'' \vdash \vartheta\phi T$. The map that sends a specification to its language signature induces a functor $\sigma(-)\colon \mathrm{SLS} \to \mathrm{MSS}$. This functor is faithful and has a left adjoint that sends each signature to an empty specification of this signature with an identity being the adjunction unit.

The category SLS is finitely complete. Indeed, the category MSS considered as a subcategory of **Set** is closed under formation of colimits of finite diagrams since it is easy to see that a pushout constructed in **Set** over every pair of MSS-morphisms is contained in MSS. In turn, a colimit of a finite diagram $\Delta\colon X \to \mathrm{SLS}$ is de-facto computed in the category MSS: its vertex is $\ulcorner \bigcup\limits_{I \in \mathrm{Ob}\,X} \varphi_I \Delta(I) \urcorner$, where φ is a family of edges of a colimit of the MSS-diagram $\sigma \circ \Delta$. In other words, the functor $\sigma(-)$ lifts colimits of all finite SLS-diagrams whose class will be denoted as FinSLS.

A refinement of an SLS-object T to T' is a pair of SLS-morphisms with the same end of the kind $\phi\colon T \to X \leftarrow T' :\!\phi'$ provided that ϕ' determines a definitional extension of the theory T', i.e., morphism ϕ' is injective and X contains axioms that unambiguously define each symbol from $\sigma(X) \setminus \phi'\big(\sigma(T')\big)$ by means of symbols from $\phi'\big(\sigma(T')\big)$. Roughly speaking, a refinement exceeds signature morphism by its capability to remove derived symbols from its codomain. An identity refinement is a pair of identity SLS-morphisms. Composition of refinements is computed via pushouts: if a pair $\vartheta\colon T' \to Y \leftarrow T'' :\!\vartheta'$ is a refinement as well, then their composition is pair $\vartheta''\phi\colon T \to Z \leftarrow T'' :\!\phi''\vartheta'$ where $\vartheta''\colon X \to Z \leftarrow Y :\!\phi''$ are edges of pushout $\vartheta''\phi' = \phi''\vartheta$. It is proved in [35] that the class of all definitional extensions is closed under compositions and formation of pushouts. Hence all SLS-objects and all their refinements comprise a category, denoted as r-SLS. It includes SLS as a (proper) subcategory since every SLS-morphism $\phi\colon T \to T'$ induces refinement $\phi\colon T \to T' \leftarrow T' :\!1_{T'}$. For every finite SLS-diagram Δ and every family φ of refinements of its objects the procedure of constructing a diagram of the kind $\Delta \oplus \varphi$ from condition (viii) of Definition 1 and a refinement $r\colon \operatorname{colim}(\Delta) \to \operatorname{colim}(\Delta \oplus \varphi)$ is described in [35].

In summary, there exists a formal design technology

$$\mathrm{SPECWARE} = \langle \mathrm{SLS},\ \mathrm{FinSLS},\ \sigma(-)\colon \mathrm{SLS} \to \mathrm{MSS},\ \text{r-SLS}\rangle.$$

2. Traceability and Aspect-Oriented Approach

We build a category-theoretic semantic model of traceability and AOP by means of constructions in formal design technologies. Fix an arbitrary formal design technology

$$\mathbf{AR} = \langle \text{c-DESC}, \mathrm{Conf}, \mathrm{sig}, \text{r-DESC}\rangle$$

and let ε be a counit of the adjunction $\mathrm{sig}^* \dashv \mathrm{sig}$. It is well known that traceability is easily compromised by refinement (the textbook example is implementing an algebraic specification by means of an algorithmic programming language). During the integration, in contrast, at least partial tracing is usually provided (here the example is a sum of indexed family of sets in the category **Set** constructed by supplying elements of each component with its index as a “label”). Therefore, for an arbitrary transformation, viz. an r-DESC-morphism $r\colon S \to T$, the following necessary condition for ability to trace its result to the source along it emerges [26]: reversing its direction, viz. category-theoretic dualization, shall turn the refinement into an activity of integration, viz. a c-DESC-morphism $r^{\mathrm{op}}\colon T \to S$.

Refinements and activities of integration shall be traced jointly as refinement is interspersed with systems assembling during the software engineering process. Such joint tracing is easiest to perform in the case where the trace r^{op} is right-invertible [24]. Indeed, the existence of a c-DESC-morphism $s\colon S \to T$ with $r^{\mathrm{op}} \circ s = 1_S$ is equivalent to the following condition. For every c-DESC-morphism $p\colon X \to S$ that describes integration of a component X into the system S there exists an activity of integration X into T compatible with tracing of refinement r in the sense that composition of trace r^{op} with this activity produces p. Such activity is $s \circ p$, since $r^{\mathrm{op}} \circ (s \circ p) = p$.

$$\begin{tikzcd} T \\ S \arrow[u, "r"'] \end{tikzcd} \qquad \begin{tikzcd} & T \arrow[d, "r^{\mathrm{op}}"] \\ X \arrow[ur, dashed, "s\circ p"] \arrow[r, "p"] & S \end{tikzcd}$$

In turn, for each c-DESC-morphism $q\colon T \to Y$, the morphism $q \circ s$ describes an activity of integration of S into Y compatible with tracing in a slightly different sense: choice $q = r^{\mathrm{op}}$ turns it into an identity morphism. Note that s is a regular monomorphism which is the category-theoretic analog of embedding the refinement source into the result without invading its structure. In practice its construction can be very labor-intensive, but it is not always needed since the primary subject for tracing along systems assembling processes are integrational requirements asserted to model interfaces. In this case, it is enough to require the SIG-morphism $\mathrm{sig}(r^{\mathrm{op}})$, which represents the trace at interface level and is called a labeling, to be right-invertible rather than the trace r^{op} itself [24]. Implementation of inverting the labeling is not usually too costly, since interfaces are designed so that their integration is “easier” than integration of models. A particular case of value-based requirements traceability [10] is obtained as we reduce costs associated with traceability by limiting the class of requirements being traced in accordance with their importance.

According to condition (vii) of Definition 1, a trivial example of traceable refinement is a c-DESC-isomorphism (recall that morphism dual to an isomorphism is identified with an inverse to it and is an isomorphism as well [1]). Use the following technique to reveal nontrivial traceable refinements. Consider a collection of all common subcategories of c-DESC and $\text{r-DESC}^{\mathrm{op}}$ that contains all c-DESC-isomorphisms. It is a complete lower semilattice under inclusion. Denote by cr-DESC intersection of all its maximal elements.

Definition 4. A cr-DESC-morphism t is called a (sig-)*trace* if the SIG-morphism $\mathrm{sig}(t)$ is a retraction (i.e., has a right inverse). A sig-image of a trace is called a (sig-)*labeling*. An r-DESC-morphism dual to a trace is called a *traceable refinement*. A refinement is called *invertible* if it is traceable and dual to a c-DESC-retraction. A formal technology **AR** is said to *support tracing* if all refinements in it are invertible.

Denote by tr-DESC a pair that consists of the class of all c-DESC-objects and the class of all traces.

Proposition 3. tr-DESC *is a subcategory of* c-DESC *that possesses the following properties.*

(i) $\mathrm{Iso}\,\text{c-DESC} \subseteq \mathrm{Mor}\,\text{tr-DESC} \subseteq \mathrm{Epi}\,\text{c-DESC}$.
(ii) $\mathrm{Iso}\,\mathrm{SIG} \subseteq \mathrm{sig}(\mathrm{Mor}\,\text{tr-DESC}) \subseteq \mathrm{Retract}\,\mathrm{SIG}$.

Proof. Every retraction is an epimorphism [1, Proposition 7.42] and every faithful functor reflects epimorphisms [1, Proposition 7.44]. □

Traceable refinements "nicely" act on configurations. Consider arbitrary diagrams $\Delta\colon X \to \text{c-DESC} \in \mathrm{Conf}$ and $\Sigma\colon |X| \to \text{c-DESC}$ linked by a **D**c-DESC-morphism $\langle \tau, jx\rangle\colon \Sigma \to \Delta$, where τ is a family of c-DESC-morphisms and $jx\colon |X| \hookrightarrow X$ is an embedding of categories. Denote by $\tau \rightrightarrows \Delta$ the drawing of the **D**c-DESC-diagram with schema *2* that represents this morphism and call it a push of diagram Δ by family τ [26, Sec. 5]. Push does not change a colimit of diagram Δ since it can be reduced to a multiple gluing of cocones of the kind $\ulcorner t \urcorner$, where t runs over family τ. Therefore, traceable refinements of components are noninvasive with respect to systems assembling: if family τ consists of traces and the push is a configuration, then condition (viii) of Definition 1 can be satisfied by taking the push for $\Delta \oplus \tau^{\mathrm{op}}$. This fact motivates introducing the following concept.

Definition 5. A formal technology **AR** is called *counified* if $\text{r-DESC} \subseteq \text{c-DESC}^{\mathrm{op}}$ and the class Conf is closed under pushes by $\text{r-DESC}^{\mathrm{op}}$-morphisms.

In a formal technology over **Set** labelings are surjective maps since by axiom of choice every **Set**-epimorphism is a retraction. Action of a refinement $t^{\mathrm{op}}\colon X \to Y$ dual to a surjection $t\colon |Y| \to |X|$ can be considered as expansion of all elements of set $|X|$ to their preimages under t that comprise partitioning of set $|Y|$ with (partial) transition of a structure of object X to them. Element $x \in |X|$ can be considered as a designator of a concern that is implemented via expansion according to intuitive understanding of refinement. An example is a refinement of scenarios in a discrete event modeling technology SM that partitions its result to subsets that are well-ordered in the sense that for every $x, y, z, u \in Y$ such that $t(x) = t(y) \neq t(z) = t(u)$ the condition $x \leq z$ implies $y \leq u$ (see [24]). The class of all scenario labelings consists of all surjective maps of sets. The technology SM supports tracing and is counified. Moreover, it will be shown in Sec. 5 that the choice of refinements in this technology is optimal with respect to balance between breadth of available transformations of models and the cost of tracing them.

The most direct and cost-effective method to provide full traceability consists in "storing" traces of refinements with models of programs generated from concerns by them [2]. Take into account that for software engineering in general and for AOP in particular of specific interest is the impact of refinements on integrational capabilities of models. So it is enough to equip models with actions of refinements at interface level, viz. labelings. (In Sec. 3 we will see that capability to "lift" refinements to level of models is a canonical condition of separation of concerns, viz. shaping each constituent concern as a modular architecture unit.) Integration as well as refinement of such enriched models shall be performed consistently on two levels: modular bases and aspect structures. As shown in [12, Sec. 7], there is a specific category-theoretic construction intended for natural attaching of actions to objects, viz. a comma category [29, Sec. II.6]. Consider a comma category $\mathrm{sig} \downarrow \mathrm{SIG}$. Its objects are all pairs of the kind $\langle A, l\colon \mathrm{sig}(A) \to L\rangle$, where A is a c-DESC-object and l is a SIG-morphism. A morphism of an object $\langle A_1, l_1\colon \mathrm{sig}(A_1) \to L_1\rangle$ to $\langle A_2, l_2\colon \mathrm{sig}(A_2) \to L_2\rangle$ is a pair $\langle f\colon A_1 \to A_2, b\colon L_1 \to L_2\rangle$ with $b \circ l_1 = l_2 \circ \mathrm{sig}(f)$.

$$\begin{tikzcd}
\langle A_1, \arrow[d, "f"'] & \mathrm{sig}(A_1) \arrow[r, "l_1"] \arrow[d, "\mathrm{sig}(f)"'] & L_1\rangle \arrow[d, "b"] \\
\langle A_2, & \mathrm{sig}(A_2) \arrow[r, "l_2"] & L_2\rangle
\end{tikzcd}$$

Definition 6. An *aspect-oriented model* (AO-model) is a (sig $\downarrow$ SIG)-object $\langle A, l\colon \mathrm{sig}(A) \to L\rangle$ such that l is a sig-labeling. The c-DESC-object A is called a (modular) *base* of the AO-model, the SIG-morphism l is called its (aspectual) *labeling*, the SIG-object L is called its *aspectual structure*.

Denote by AO the full subcategory of sig $\downarrow$ SIG whose class of objects consists of all AO-models. We explain this construction by example of a formal technology over **Set**. Recall that here a labeling l is a surjective map that sends each element of set $|A|$ to an element of set L that designates a concern for which it is intended. In particular, it is natural to call an AO-model an aspect if L is a singleton set in it. Essentially (up to isomorphism) a labeling is an equivalence relation on set $|A|$ whose equivalence classes represent particular aspects. Any equivalence relation turns A into a valid AO-model so aspectual structure in general has no interference with modular structure. An AO-morphism is precisely a c-DESC-morphism that preserves this equivalence relation. Since a canonical forgetful functor from the category **Equ** of all sets equipped with equivalence relation and all their homomorphisms to **Set** is topological [1, Sec. 21], a category of labeled models is in many familiar cases equivalent to a topological category over c-DESC. In these cases it "inherits" systems assembling techniques from original unlabeled models.

For example, in the discrete event modeling technology SM an AO-model is precisely a partially ordered multiset (pomset) of aspects. All labeled scenarios comprise a category equivalent to a topological category over **Pos**. A similar approach to scenario modeling was proposed as long ago as in the 1980's [31]; however the nature of labels and methods of their synthesis remained unclear because they have not been examined in the context of AOP. Various classes of labeled scenarios are mastered by process engineering technologies that offer different specialized notations: graphical, hypertext, Petri nets, etc. In addition, one of the classical semantical models of AOP known as trace semantics is based upon labeled scenarios [8].

We return to considering arbitrary formal technologies. We will employ an arrow category of the kind $C^{\mathbf{2}}$ induced by an arbitrary category C. Recall that the class of all $C^{\mathbf{2}}$-objects consists of all C-morphisms (diagrams of the kind $\mathbf{2} \to C$) and a morphism from f to g is a pair of C-morphisms $\langle u, v\rangle$ with $v \circ f = g \circ u$. There exist functors $\mathrm{dom}, \mathrm{codom}\colon C^{\mathbf{2}} \to C$ that send a C-morphism to its domain and codomain, respectively.

Let LAB be a full subcategory of $\mathrm{SIG}^{\mathbf{2}}$ whose class of objects consists of all sig-labelings and let $\mathrm{il}\colon \mathrm{LAB} \hookrightarrow \mathrm{SIG}^{\mathbf{2}}$ be an embedding. We will use the following "forgetful" functors induced by the structure of the category sig $\downarrow$ SIG:

$$\begin{aligned}
\mathrm{mod}\colon\ & \mathrm{AO} \to \text{c-DESC}\\
:\ & \langle A, l\rangle \mapsto A,\ \langle f, b\rangle \mapsto f,\\
\mathrm{asp}\colon\ & \mathrm{AO} \to \mathrm{LAB}\\
:\ & \langle A, l\rangle \mapsto l,\ \langle f, b\rangle \mapsto \langle \mathrm{sig}(f), b\rangle,\\
\mathrm{int} = \mathrm{sig} \circ \mathrm{mod} = \mathrm{dom} \circ \mathrm{il} \circ \mathrm{asp}\colon\ & \mathrm{AO} \to \mathrm{SIG}\\
:\ & \langle A, l\rangle \mapsto \mathrm{sig}(A),\ \langle f, b\rangle \mapsto \mathrm{sig}(f),\\
\mathrm{str} = \mathrm{codom} \circ \mathrm{il} \circ \mathrm{asp}\colon\ & \mathrm{AO} \to \mathrm{SIG}\\
:\ & \langle A, l\rangle \mapsto \mathrm{codom}\, l,\ \langle f, b\rangle \mapsto b.
\end{aligned}$$

The functors 1_{AO}, mod, asp, and int will be called *fundamental*. They allow building AO as a universal construction in the category **CAT** (cf. [29, pp. 47–48]): it is easy to see that the equality $\mathrm{sig} \circ \mathrm{mod} = (\mathrm{dom} \circ \mathrm{il}) \circ \mathrm{asp}$ that defines the functor int represents a pullback of a pair of functors $\mathrm{sig}\colon \text{c-DESC} \to \mathrm{SIG} \leftarrow \mathrm{LAB} :\! \mathrm{dom} \circ \mathrm{il}$ with AO as a vertex. It follows that enriching models of programs by aspect labeling can be performed uniquely (up to isomorphism).

$$\begin{tikzcd}
\mathrm{AO} \arrow[r, dashed, "\mathrm{asp}"] \arrow[d, dashed, "\mathrm{mod}"'] & \mathrm{LAB} \arrow[d, "\mathrm{dom} \circ \mathrm{il}"] \\
\mathrm{c\text{-}DESC} \arrow[r, "\mathrm{sig}"] & \mathrm{SIG}
\end{tikzcd}$$

Fundamental functors play an important role in formalizing aspect-oriented design: later we will prove (see Theorem 1) that they allow extracting different interfaces from AO-models in the sense of Definition 1, i.e., generate formal design technologies. Different interfaces are needed to elaborate design decisions of different kinds: the "modular" interface extracted by the functor mod plays a key role at modularizing aspects (see Sec. 3), the aspect interface extracted by the functor asp plays a key role at constructing labeling, the original interface extracted by the functor int plays a key role at specifying integrational requirements to models without detailing their modular or aspectual structure. Other kinds of interfaces may exist in general that "detail" original interfaces from the category SIG, i.e., are naturally mapped to them.

At the same time, as we see later (see Proposition 4), the aspectual structure extracted by the functor str cannot be employed as an interface of nontrivial AO-models. The functor str formally expresses a "quintessence" of an aspect-oriented extension of system design technologies which is irreducible to concepts of modular approach. Remarkably the functor str is surjective, so every aspectual structure is implemented in an appropriate AO-model (Corollary 1.4). It is shown in [25] that the formal definition of key concepts of aspect and weaving requires only this functor in addition to construction of the category AO. Recall that assembling of a program from aspects is called weaving since it exceeds the bounds of traditional modular linking. In Sec. 3, we will formally describe weaving as a universal construction in the category AO, in particular, as a colimit of a connection diagram of a certain specific kind.

We state rules to construct configurations and refinements of AO-models from modular "material." Refinements are naturally obtained as dual to traces (untraceable refinements of a technology **AR** are ignored completely). Let

$$\mathrm{tr\text{-}AO} = \big(\mathrm{Ob}\,\mathrm{AO}, \mathrm{mod}^{-1}(\mathrm{Mor}\,\mathrm{tr\text{-}DESC})\big).$$

By Proposition 3, tr-AO is a category.

For configurations of AO-models it is reasonable to employ AO-diagrams extending modular configuration whose colimits are consistently computed at levels of modular bases and aspectual structures. In addition, the counifiedness property is desired in order to guarantee non-invasive action of componentwise refinements of systems.

Definition 7. An AO-diagram Δ is called *aspectually determined* if functor $\langle \mathrm{mod}, \mathrm{str}\rangle : \mathrm{AO} \to \mathrm{c\text{-}DESC} \times \mathrm{SIG}$ determines its colimits. Let ai be an arbitrary functor from AO to an arbitrary category INT. A class of INT-diagrams AIDia is called *aspectually closed* (with respect to ai) if the class of AO-diagrams $\mathbf{D}\mathrm{ai}^{-1}(\mathrm{AIDia})$ consists of aspectually determined diagrams, is contained in class $\mathbf{D}\mathrm{mod}^{-1}(\mathrm{Conf})$, and is closed under pushes by tr-AO-morphisms.

Denote by $\mathrm{AOInt}_{\mathrm{ai}}$ the union of all aspectually closed classes of INT-diagram. Clearly, it is an aspectually closed class itself.

Definition 8. An arbitrary functor ai with domain AO is said to *generate an aspect-oriented formal technology* (AO-technology) over **AR** if a quadruple

$$\mathrm{AO}_{\mathrm{ai}}(\mathbf{AR}) = \langle \mathrm{AO}, \mathbf{D}\mathrm{ai}^{-1}(\mathrm{AOInt}_{\mathrm{ai}}), \mathrm{ai}, \mathrm{tr\text{-}AO}^{\mathrm{op}}\rangle$$

is a formal design technology and there exists a functor si such that $\mathrm{si} \circ \mathrm{ai} = \mathrm{int}$.

Theorem 1. *Every fundamental functor generates an AO-technology over* **AR**.

Proof. Conditions (i), (v)–(viii) of Definition 1 hold for the quadruple $\mathrm{AO}_{\mathrm{ai}}(\mathbf{AR})$ upon any choice of the functor ai with domain AO. It remains to verify conditions (ii)–(iv) for functors mod and asp since for the functor 1_{AO} they hold obviously and for the functor int they hold due to stability with respect to the composition of functors.

(ii). The functor $\mathrm{dom} \circ il$ is faithful since for two arbitrary LAB-morphisms $\langle f, b\rangle, \langle f, b'\rangle \colon l \to k$ the condition $b' \circ l = k \circ f = b \circ l$ holds, hence $b' = b$ given that l is right-invertible. Therefore, the functor mod, being an edge parallel to $\mathrm{dom} \circ \mathrm{il}$ in the pullback in **CAT**, is faithful. Similarly, the functor asp parallel to the faithful functor sig is faithful.

(iii). The discrete labeling functor

$$\begin{aligned}\mathrm{mod}^* &\colon \text{c-DESC} \to \mathrm{AO}\\ &\colon A \mapsto \langle A, 1_{\mathrm{sig}(A)}\rangle,\ f \mapsto \langle f, \mathrm{sig}(f)\rangle\end{aligned}$$

is left adjoint to mod with an identity as the adjunction unit. The functor

$$\begin{aligned}\mathrm{asp}^* &\colon \mathrm{LAB} \to \mathrm{AO}\\ &\colon l \mapsto \langle \mathrm{sig}^*(\mathrm{dom}\, l), l\rangle,\ \langle p, q\rangle \mapsto \langle \mathrm{sig}^*(p), q\rangle\end{aligned}$$

is left adjoint to asp with an identity as the adjunction unit.

(iv). By Definition 7, the functor mod lifts colimits of configurations. To see that the functor asp lifts them as well, consider an arbitrary diagram $\Delta \in \mathbf{D}\mathrm{asp}^{-1}(\mathrm{AOInt}_{\mathrm{asp}})$ and a colimit $\sigma \colon \mathrm{asp} \circ \Delta \to \ulcorner k \urcorner$. Choose an arbitrary colimit $\delta \colon \Delta \to \langle A, l\rangle$. By Definition 7, $\mathrm{str} \circ \delta$ is a colimit of diagram $\mathrm{str} \circ \Delta$ and $\mathrm{mod} \circ \delta$ is a colimit of the diagram $\mathrm{mod} \circ \Delta$, so $\mathrm{int} \circ \delta$ is a colimit of diagram $\mathrm{int} \circ \Delta$ since the functor sig preserves colimits of configurations by Proposition 2. It can be verified directly that l is a colimit vertex of a diagram $\mathrm{il} \circ \mathrm{asp} \circ \Delta$ in the arrow category $\mathrm{SIG}^{\mathbf{2}}$, so there exists a colimit $\mathrm{asp} \circ \delta \colon \mathrm{asp} \circ \Delta \to \ulcorner l \urcorner$ since l is sig-labeling. Since colimit is unique up to isomorphism, there exists a LAB-isomorphism i such that $\sigma = \ulcorner i \urcorner \circ (\mathrm{asp} \circ \delta)$, so a cocone $\mathrm{dom} \circ \mathrm{il} \circ \sigma = \ulcorner \mathrm{dom}\big(\mathrm{il}(i)\big) \urcorner \circ (\mathrm{int} \circ \delta)$ is a colimit of the diagram $\mathrm{int} \circ \Delta$. Since the functor sig lifts colimits of configurations, there exists a colimit $\theta \colon \mathrm{mod} \circ \Delta \to \ulcorner B \urcorner$ with $\mathrm{sig} \circ \theta = \mathrm{dom} \circ \mathrm{il} \circ \sigma$. Hence $\langle \theta, \mathrm{codom} \circ \mathrm{il} \circ \sigma\rangle \colon \Delta \to \ulcorner \langle B, k\rangle \urcorner$ is a colimit. □

Corollary 1.1. *An arbitrary functor* $\mathrm{ai} \colon \mathrm{AO} \to \mathrm{INT}$ *generates an AO-technology over* **AR** *if and only if it satisfies the following conditions.*

(i) *The functor* ai *has a left adjoint with an identity as the adjunction unit.*
(ii) $\mathrm{int} \circ \mathrm{ai}^* \circ \mathrm{ai} = \mathrm{int}$ *where* ai^* *is left adjoint from the condition* (i).
(iii) *Every aspectually closed class of* INT*-diagrams consists of* ai*-preconfigurations.*

Proof. The necessity follows from Definition 8 taking into account that $\mathrm{si} = \mathrm{si} \circ (\mathrm{ai} \circ \mathrm{ai}^*) = \mathrm{int} \circ \mathrm{ai}^*$ for every functor si that satisfies the condition $\mathrm{si} \circ \mathrm{ai} = \mathrm{int}$. The sufficiency follows from the fact that conditions (i), (v)–(viii) of Definition 1 hold upon any choice of the functor ai (cf. the proof of Theorem 1) and condition (ii) is guaranteed since the functor int is faithful and the functor ai is the first component of its factorization (cf. [1, Proposition 3.30(2)]). □

Corollary 1.2. *Every configuration in the technology* $\mathrm{AO}_{\mathrm{int}}(\mathbf{AR})$ *is a configuration in every AO-technology over* **AR**. *In turn, every configuration in every AO-technology over* **AR** *is a configuration in* $\mathrm{AO}_{1_{\mathrm{AO}}}(\mathbf{AR})$.

Proof. Let $\mathrm{ai} \colon \mathrm{AO} \to \mathrm{INT}$ be an arbitrary functor that generates an AO-technology over **AR**, and let $\mathrm{si} \colon \mathrm{INT} \to \mathrm{SIG}$ be a functor that satisfies the condition $\mathrm{si} \circ \mathrm{ai} = \mathrm{int}$. Choose an arbitrary AO-diagram $\Delta \in \mathbf{D}\mathrm{int}^{-1}(\mathrm{AOInt}_{\mathrm{int}})$ and let $\Theta = \mathrm{int} \circ \Delta$ and $\Xi = \mathrm{ai} \circ \Delta$. Every AO-diagram $\Delta' \in \mathbf{D}\mathrm{ai}^{-1}(\{\Xi\})$ is aspectually determined and belongs to the class $\mathbf{D}\mathrm{mod}^{-1}(\mathrm{Conf})$ since $\mathrm{int} \circ \Delta' = \mathrm{si} \circ \Xi = \Theta \in \mathrm{AOInt}_{\mathrm{int}}$. Denote by $\mathrm{ai} \rightrightarrows \Omega$ a class of $\mathbf{D}$ai-images of all pushes of an AO-diagram Ω by tr-AO-morphisms. Every AO-diagram $\Delta'' \in \mathbf{D}\mathrm{ai}^{-1}(\mathrm{ai} \rightrightarrows \Delta')$ is aspectually determined and belongs to $\mathbf{D}\mathrm{mod}^{-1}(\mathrm{Conf})$ as well since

$$\mathrm{int} \circ \Delta'' \in \mathrm{si} \circ (\mathrm{ai} \rightrightarrows \Delta') \subseteq \mathrm{int} \rightrightarrows \Delta' \subseteq \mathrm{AOInt}_{\mathrm{int}}.$$

By induction we define a sequence of classes of INT-diagrams PDia as

$$\mathrm{PDia}_0 = \{\Xi\}, \quad \mathrm{PDia}_{n+1} = \bigcup_{\Omega \in \mathbf{D}\mathrm{ai}^{-1}(\mathrm{PDia}_n)} \mathrm{ai} \rightrightarrows \Omega.$$

The class $\bigcup_{n\geq 0} \mathrm{PDia}_n$ is aspectually closed and contains Ξ. Hence $\Delta \in \mathbf{D}\mathrm{ai}^{-1}(\mathrm{AOInt}_{\mathrm{ai}})$. The first statement is proved.

The second statement follows directly from the fact that class $\mathbf{D}\mathrm{ai}^{-1}(\mathrm{AOInt}_{\mathrm{ai}})$ is aspectually closed with respect to functor 1_{AO}. □

Corollary 1.3. *The class* Mor tr-AO *consists of all traces of the technology* $\mathrm{AO}_{\mathrm{int}}(\mathbf{AR})$*.*

Corollary 1.4. *The functor* str *is right-invertible.*

Proof. The functor $1_{-}\colon \mathrm{SIG} \hookrightarrow \mathrm{LAB}\colon A \mapsto 1_A$, $f \mapsto \langle f, f\rangle$ is a right inverse to the functor $\mathrm{codom} \circ \mathrm{il}$ and the functor asp is right-invertible by Theorem 1. □

There exist design technologies in which aspect-oriented extension introduces no essentially new constructs. We will show formally that they are characterized by an absence of traceable refinements capable of generating a nontrivial aspectual structure: every labeling in them turns out to be an isomorphism. This criterion is equivalent to a number of others such as capability of the functor str to extract interfaces from AO-models.

Definition 9. A formal design technology **AR** is called *aspectually trivial* if the functor mod is an equivalence of categories AO and c-DESC.

Proposition 4. *The following conditions are equivalent for each formal technology* **AR**.

(i) *The formal technology* **AR** *is aspectually trivial.*
(ii) *Every labeling is an isomorphism.*
(iii) *The functor* int *is naturally isomorphic to the functor* str*.*
(iv) *The functor* str *is faithful.*

Proof. We will employ properties of equivalence of categories stated in [29, Sec. IV.4]. Denote by μ the counit of the adjunction $\mathrm{mod}^* \dashv \mathrm{mod}$. Fix an arbitrary sig-labeling $l\colon X \to L$.

(i) $\Longrightarrow$ (ii). By paragraph (iii) of the proof of Theorem 1, we have $l = \mathrm{str}(\mu_{\mathrm{asp}^*(l)})$. By assumption, μ consists of isomorphisms. Hence l is an isomorphism.

(ii) $\Longrightarrow$ (i). By the same paragraph (iii) of the proof of Theorem 1, the assumption implies that μ consists of isomorphisms.

(ii) $\Longrightarrow$ (iii). By assumption, a natural transformation of functor int to str that consists of labelings of all AO-models is a natural isomorphism.

(iii) $\Longrightarrow$ (iv). Every functor which is naturally isomorphic to a faithful functor is faithful itself.

(iv) $\Longrightarrow$ (ii). Consider the AO-morphism $l^* = \mathrm{asp}^*(\langle l, 1_L\rangle)$. Since $\mathrm{str}(l^*) = 1_L$, the assumption implies that $l^* \in \mathrm{Mono}\,\mathrm{AO}$ (every faithful functor reflects monomorphisms [20, Proposition 7.37(2)]). Hence $l = \mathrm{int}(l^*) \in \mathrm{Mono}\,\mathrm{SIG}$ (every functor that has a left adjoint preserves all colimits [29, Sec. V.5] in particular monomorphisms). Since l is right invertible, we conclude that $l \in \mathrm{Iso}\,\mathrm{SIG}$. □

For example, it is easy to verify that condition (ii) of Proposition 4 holds for the technology SPECWARE, so it is aspectually trivial: aspect-oriented techniques traditionally are not used in an algebraic approach to software design.

3. Weaving, Separation of Concerns, and Explication of Aspectual Structure

We are going to describe a megamodel of aspect weaving procedure as a universal construction in an AO-technology [25]. In classical AOP, weaving consists in plugging the aspect program called an advice into the base program at specified places called join points [4]. Every time when execution of the base program reaches a join point the advice is called. Therefore, an advice is usually a block of source code guarded by the condition that identifies the join point, and the beginning of the block serves as an entry point to the advice. Thus the weaving tool accepts two specifications as input:

- the description of join points in the base program known as pointcut and frequently specified in the form of a regular expression over syntactic structures of the programming language on which the base program is written;
- the description of entry points to the advice in join points.

In the first step of weaving, a sufficient number of copies of the advice, one for each join point, are (virtually) created with a correspondent entry point marked in each copy. In the second step, these points are “glued” together avoiding invasion into the aspectual structure of both the base and the advice. For formal description of the weaving specification, an extra model C called a connector [30] is employed in addition to models of the base B and of the advice W. The connector is integrated with the base at join points and with the advice at entry points generating a pair of AO-morphisms $j\colon B \leftarrow C \rightarrow W :e$. (Here the distinction of weaving from modular linking vividly manifests itself since linking is formalized by one-step activity of the kind $l\colon M \rightarrow S$, where M denotes a module and S denotes a system.) It is easy to see by example of a discrete event modeling AO-technology that the first step of weaving is represented as a product $C \times W$ and the second step is represented as a pushout (connection) of a pair of morphisms $j\colon B \leftarrow C \rightarrow C \times W :\langle 1_C, e\rangle$. These constructions shall be natural with respect to extracting modular base and aspectual structure so the requirement similar to aspectual determinedness is asserted (cf. Definition 7): the functor $\langle \mathrm{mod}, \mathrm{str}\rangle\colon \mathrm{AO} \rightarrow \text{c-DESC} \times \mathrm{SIG}$ shall determine both the product and the pushout. If this requirement is satisfied, then the weaving result is a vertex of the pushout denoted by $j \bowtie e$.

$$\begin{tikzcd}
C \arrow[r, "\langle 1_C, e\rangle"] \arrow[d, "j"'] & C \times W \arrow[d, dashed] \\
B \arrow[r, dashed] & j \bowtie e
\end{tikzcd}$$

Clearly, the weaving result is unique up to isomorphism provided that it exists. Moreover, since the AO-morphism $\langle 1_C, e\rangle\colon C \rightarrow C \times W$ is left-invertible (we have $\pi_C \circ \langle 1_C, e\rangle = 1_C$, where $\pi_C\colon C \times W \rightarrow C$ is a projection), the pushout edge parallel to it is left-invertible as well (cf. dual statement in [1, Proposition 11.18]), so the base is noninvasively embedded into the weaving result. One more property of weaving easily verifiable over the diagram above is independence of the weaving result on the order of weaving several mutually independent advices to the same base.

AOP technologies offer a variety of weaving tools including method calls dispatchers, converters of executable byte-code, source code pre-compilers. However it is possible to avoid employing them, making it easier to build complex programs, if the plugged aspects can be modularized, viz. shaped as modular architecture units: classes (elements of object-oriented decomposition), tables in the database, and so on. Modularization of all aspects comprising an AO-model is the most straightforward method to label them known as a separation of concerns that is one of the classical software engineering problems.

Modularizable AO-models are distinguished among others by behaving like modular units when being integrated with modules. Capabilities to integrate modules into an AO-model are determined by its modular base. In turn, capabilities to integrate an AO-model into modules are determined by its aspectual structure since aspects that constitute the AO-model act as its elementary units while integrating into a module. Formally modularizable models comprise a full subcategory of AO (denoted as m-AO) such that there exists an aspect-oriented extension (AO-extension) of a modular technology, viz. an embedding $\mathrm{am}\colon \text{c-DESC} \hookrightarrow \text{m-AO}$ that fully reproduces integrational capabilities of modules in the following sense. On the one hand, all ways of integrating a module $M \in \mathrm{Ob}\,\text{c-DESC}$ into an AO-model $A \in \mathrm{Ob}\,\text{m-AO}$ are determined by the set of morphisms $\mathrm{Mor}(\mathrm{am}(M), A)$. Hence functor mod that extracts modular interfaces (more rigorously, its restriction to m-AO denoted as am_*) shall establish a bijection between set $\mathrm{Mor}(\mathrm{am}(M), A)$ and set $\mathrm{Mor}\big(M, \mathrm{mod}(A)\big)$. On the other hand, all ways of integrating AO-model A

into module M are determined by the set of morphisms $\mathrm{Mor}\big(A, \mathrm{am}(M)\big)$. Hence a functor $\mathrm{am}^*\colon$ m-AO $\to$ c-DESC shall exist that modularizes aspectual structure, trivially acts on modules ($\mathrm{am}^* \circ \mathrm{am} = 1_{\text{c-DESC}}$), and establishes a bijection between sets $\mathrm{Mor}\big(A, \mathrm{am}(M)\big)$ and $\mathrm{Mor}(\mathrm{am}^*(A), M)$. The capability to treat c-DESC-object $\mathrm{am}^*(A)$ as a "lift" of the aspectual structure of the AO-model A to the modular level is justified by the following additional naturality requirement: the functor $\mathrm{sig} \circ \mathrm{am}^*$ that extracts an interface from a modularized aspectual structure shall coincide with restriction of the functor str on m-AO that reveals the original aspectual structure at the level of interfaces.

For example, every formal design technology has an AO-extension defined as an isomorphism that acts as the functor mod^* between c-DESC and a full subcategory of AO with class of objects $\{\langle A, 1_{\mathrm{sig}(A)}\rangle \mid A \in \mathrm{Ob}\,\text{c-DESC}\}$. In technologies over **Set** it generates discretely labeled models that are the most "aspect-unoriented" in the sense that in them each concern labels only one element of the underlying set so no scattering happens at all. We will call such an AO-extension trivial and show that every AO-extension acts essentially (up to a natural isomorphism) the same as it: modules (i.e., c-DESC-objects) are always sent to AO-models whose history of refining from concerns is lost (trivial). Thus, an AO-extension is essentially uniquely determined by its codomain, viz. the class of all modularizable AO-models.

In the language of category theory, requirements to an AO-extension are concisely stated in terms of adjunction of functors.

Definition 10. A functor $\mathrm{am}\colon$ c-DESC $\to$ m-AO, where m-AO is a full subcategory of AO, is called an *aspect-oriented extension* (AO-extension) of a formal technology if it has the following adjoint functors:

- right adjoint am_* with an identity as the adjunction unit such that $\mathrm{am}_*(f) = \mathrm{mod}(f)$ for each m-AO-morphism f;
- left adjoint am^* with an identity as the adjunction counit such that $\mathrm{sig}\big(\mathrm{am}^*(f)\big) = \mathrm{str}(f)$ for each m-AO-morphism f.

Under these conditions, m-AO-objects are called (am-)*modularizable* AO-models.

Fix an arbitrary AO-extension $\mathrm{am}\colon$ c-DESC $\to$ m-AO, and let $\mathrm{mao}\colon$ m-AO $\hookrightarrow$ AO be a full embedding.

Proposition 5. *AO-extension* am *is a full left-invertible embedding:* $\mathrm{am}_* \circ \mathrm{am} = \mathrm{am}^* \circ \mathrm{am} = 1_{\text{c-DESC}}$.

Proposition 6. *The functor* $\mathrm{mao} \circ \mathrm{am}$ *is naturally isomorphic to functor* mod^*.

Proof. By definition, for each $X, Y \in \mathrm{am}(\mathrm{Ob}\,\text{c-DESC}) \subseteq \mathrm{Ob}\,\text{m-}AO$ and each AO-morphism $f\colon X \to Y$, the condition $f = \mathrm{am}\big(\mathrm{am}_*(f)\big) = \mathrm{am}\big(\mathrm{mod}(f)\big)$ holds. Hence, by Proposition 5,

$$\mathrm{str}(f) = \mathrm{sig}\big(\mathrm{am}^*(f)\big) = \mathrm{sig}\Big(\mathrm{am}^*\Big(\mathrm{am}\big(\mathrm{mod}(f)\big)\Big)\Big) = \mathrm{int}(f).$$

Choose an arbitrary SIG-object I and let $T = \mathrm{am}\big(\mathrm{sig}^*(I)\big)$, $l = \mathrm{asp}(T)$. We have

$$\mathrm{mod}(T) = \mathrm{am}_*\Big(\mathrm{am}\big(\mathrm{sig}^*(I)\big)\Big) = \mathrm{sig}^*(I)$$

and $\mathrm{str}(T) = \mathrm{int}(T) = I$. Hence $\mathrm{dom}\, l = \mathrm{codom}\, l = I$. For an arbitrary SIG-morphism m such that $l \circ m = 1_I$ it can be verified directly that an AO-morphism $\langle \mathrm{sig}^*(m \circ l), 1_I\rangle\colon T \to T$ exists, so $1_I = \mathrm{sig}\big(\mathrm{sig}^*(m \circ l)\big) = m \circ l$. Hence $l \in \mathrm{Iso}\,\mathrm{SIG}$.

For an arbitrary c-DESC-object P let $k = \mathrm{asp}\big(\mathrm{am}(P)\big)$, $P^* = \mathrm{sig}^*\big(\mathrm{sig}(P)\big)$, $i = \mathrm{asp}\big(\mathrm{am}(P^*)\big)$. As proved right above, $i \in \mathrm{Iso}\,\mathrm{SIG}$. It can be verified directly that an AO-morphism $\langle \varepsilon_P, k \circ i^{-1}\rangle\colon \mathrm{am}(P^*) \to \mathrm{am}(P)$ exists, so $k \circ i^{-1} = \mathrm{sig}(\varepsilon_P) = 1_{\mathrm{sig}(P)}$, whence $k = i$. (Recall that ε denotes the counit of the adjunction $\mathrm{sig}^* \dashv \mathrm{sig}$.) Consequently, family of AO-isomorphisms $\big\langle 1_P, \mathrm{asp}\big(\mathrm{am}(P)\big)\big\rangle\colon \langle P, 1_{\mathrm{sig}(P)}\rangle \to \mathrm{am}(P)$, $P \in \mathrm{Ob}\,\text{c-DESC}$, constitutes a natural isomorphism of the functor mod^* to $\mathrm{mao} \circ \mathrm{am}$. □

Thus, an AO-extension is consistent with respect to assembling of systems (by Proposition 6 it preserves colimits of all c-DESC-diagrams), with respect to extracting interfaces (by Proposition 5 we have $\operatorname{int}\big(\operatorname{am}(-)\big) = \operatorname{sig}\Big(\operatorname{am}_*\big(\operatorname{am}(-)\big)\Big) = \operatorname{sig}(-)\big)$, and with respect to tracing (by Proposition 5 it sends a trace of every traceable refinement to a tr-AO-morphism).

Now we show that a category of modularizable AO-models can be employed as the basic category of models for a subtechnology in $\mathrm{AO}_{\mathrm{mod}}(\mathbf{AR})$. Let $\text{mtr-AO} = \text{m-AO} \cap \text{tr-AO}$. Denote by DAOExt the class of all c-DESC-diagrams $\Theta \in \mathrm{AOInt}_{\mathrm{mod}}$ (see Definition 7) such that every AO-diagram from class $\operatorname{mao} \circ \mathbf{Dam}_*^{-1}(\{\Theta\})$ has a colimit that belongs to the class $\operatorname{mao} \circ \mathbf{Dam}_*^{-1}(\{\theta\})$ for each colimit θ of diagram Θ.

Definition 11. A *modularizable AO-technology* over an arbitrary formal design technology **AR** generated by an AO-extension am is a quadruple

$$\text{m-AO}_{\mathrm{am}}(\mathbf{AR}) = \langle \text{m-AO}, \mathbf{Dam}_*^{-1}(\mathrm{DAOExt}), \mathrm{am}_*, \text{mtr-AO}^{\mathrm{op}} \rangle.$$

Proposition 7. *For each formal technology* **AR** *and each AO-extension* am *the quadruple* $\text{m-AO}_{\mathrm{am}}(\mathbf{AR})$ *is the counified formal design technology largest among all subtechnologies of* $\mathrm{AO}_{\mathrm{mod}}(\mathbf{AR})$ *that have* m-AO *for a category of models and* am_* *for a functor of extracting interfaces. The technology* $\text{m-AO}_{\mathrm{am}}(\mathbf{AR})$ *coincides with* $\mathrm{AO}_{\mathrm{mod}}(\mathbf{AR})$ *if and only if* $\text{m-AO} = \mathrm{AO}$.

Proof. Condition (ii) of Definition 1 is implied by Theorem 1, condition (iii) is implied by Definition 10. Conditions (i), (iv)–(vii) hold by construction. Condition (viii) along with counifiedness follows from the fact that class DAOExt is aspectually closed (in the sense of Definition 7). Triple $\langle \operatorname{mao}, 1_{\text{c-DESC}}, \operatorname{mao}(-^{\mathrm{op}})^{\mathrm{op}} \rangle$ determines a morphism of the formal design technology $\text{m-AO}_{\mathrm{am}}(\mathbf{AR})$ to $\mathrm{AO}_{\mathrm{mod}}(\mathbf{AR})$. In every subtechnology whose embedding into $\mathrm{AO}_{\mathrm{mod}}(\mathbf{AR})$ is determined by such a triple all configurations are contained in class $\mathbf{Dam}_*^{-1}(\mathrm{DAOExt})$ and all refinements are contained in the class $\operatorname{Mor} \text{mtr-AO}^{\mathrm{op}}$. □

For formal description of modularization of aspects that constitute AO-models, an important role is played by the unit of the adjunction $\mathrm{am}^* \dashv \mathrm{am}$, which will be denoted as χ. By definition, for each m-AO-object S m-AO-morphism $\chi_S\colon S \to \operatorname{am}\big(\operatorname{am}^*(S)\big)$ is a preimage of the c-DESC-morphism $1_{\mathrm{am}^*(S)}$ under bijection

$$\mathrm{am}^*\colon \operatorname{Mor}\Big(S, \operatorname{am}\big(\operatorname{am}^*(S)\big)\Big) \cong \operatorname{Mor}\big(\operatorname{am}^*(S), \operatorname{am}^*(S)\big).$$

Hence

$$\operatorname{str}(\chi_S) = \operatorname{sig}\big(\operatorname{am}^*(\chi_S)\big) = 1_{\operatorname{sig}(\operatorname{am}^*(S))} = 1_{\operatorname{str}(S)},$$

by Definition 10, so the action of the morphism χ_S is nontrivial only at the level of modular bases. Since the c-DESC-object $\mathrm{am}^*(S)$ represents the aspectual structure of the AO-model S on the modular level, the c-DESC-morphism $\mathrm{am}_*(\chi_S)\colon \mathrm{am}_*(S) \to \mathrm{am}^*(S)$ can be considered as a canonical method of integrating the modular base of the model to its modularized aspectual structure. This morphism is a regular c-DESC-epimorphism (see Proposition 11 below), so it determines a factorization of the modular base by aspects. Therefore, separate aspects can be extracted from the model by means of modular design via tracing concerns along this morphism (here the universal category-theoretic construction of pushout is employed which generalizes construction of full preimage in set theory [25]). Family $\mathrm{am}_*(\chi_S)$, $S \in \operatorname{Ob} \text{m-AO}$, comprises a natural transformation of the functor am_* that extracts modular bases from modularizable AO-models to the functor am^* that extracts their aspectual structures.

Definition 12. An (am-)*modularization* (of aspectual structure) of an arbitrary m-AO-object S is a c-DESC-morphism $\mathrm{am}_*(\chi_S)$, where χ is the unit of the adjunction $\mathrm{am}^* \dashv \mathrm{am}$.

A natural (although not unique) method of modularizing the aspectual structure of an AO-model consists in reconstructing a traceable refinement that induced its labeling. Indeed, if for an AO-model $\langle A, l \rangle$ there exists an appropriate traceable refinement $s\colon X \to A$ of some c-DESC-object X to A that satisfies the condition $\operatorname{sig}(s^{\mathrm{op}}) = l$, then X can be considered as a modular unit consisting of all concerns of the AO-model and s^{op} can be considered as “canonical” modularization of its aspectual structure.

Reconstruction of a refinement can be considered full-fledged if every activity of integrating the model into a system is projected onto a modular level in the form of a pair of activities of integrating modular bases and aspectual structures which are consistent with respect to reconstructed refinements. Refinements of AO-models can often be similarly reconstructed, viz. projected to modular level in the form of a consistent pair of refinements of modular bases and aspectual structures.

In formal technologies over **Set**, reconstruction of a refinement of an AO-model $\langle A, l\rangle$ is possible in the case where the labeling is to a certain extent consistent with respect to the structure of the object A. Specifically, reconstruction consists in creating enough structure on the set $l(|A|)$ to turn the map l into a trace. If it is possible to create such structure, then the AO-model is usually completely separated on modularizable aspects: their family looks like $\{\langle l^{-1}(x), l_x \colon y \mapsto x\rangle \mid x \in l(|A|)\}$. For example, a refinement of a labeled scenario is (obviously) reconstructed from a labeling if and only if the latter partitions the partially ordered set A on a well-ordered family of preimages of elements. Obvious examples are: every nonempty discretely labeled scenario, every aspect, and every scenario in which all elements of every order connection component have the same label. Since the technology SM support tracing, reconstruction is always full-fledged (see Proposition 9 below). Every refinement of labeled scenarios which permit reconstruction of refinements from their labelings is reconstructible itself.

Actions of modular level that generate labeling, integration, and refinement of AO-models will be called explications. Condition of full-fledgedness of explication is formalized as appropriate universal requirement.

Definition 13. An *explication* (of the aspectual structure) of an AO-model $\langle A, l\rangle$ is a traceable refinement s of some r-DESC-object to A such that $\mathrm{sig}(s^{\mathrm{op}}) = l$. An *explication of the action* of an AO-morphism $f \colon S \to R$ (along explications s and r of AO-models S and R, respectively) is a c-DESC-morphism q such that $q \circ s^{\mathrm{op}} = r^{\mathrm{op}} \circ \mathrm{mod}(f)$. An explication s of an AO-model S is called *universal* if every AO-morphism with a domain S is explicable along s and every explication of its codomain. The *aspectual core* of a formal technology **AR** is the full subcategory c-AO of AO that consists of all objects that have a universal explication. An *explication of a refinement* of AO-models g is a refinement of some r-DESC-objects that has a trace that explicates the action of the AO-morphism g^{op}.

$$\begin{tikzcd}
S \arrow[d, "f"'] & & \mathrm{mod}(S) \arrow[r, "s^{\mathrm{op}}"] \arrow[d, "\mathrm{mod}(f)"'] & {} \arrow[d, dotted, "q"] \\
R & \Longrightarrow & \mathrm{mod}(S) \arrow[r, "r^{\mathrm{op}}"] & {}
\end{tikzcd}$$

Proposition 8. *The following conditions are equivalent for an arbitrary AO-model $S = \langle A, l \colon \mathrm{sig}(A) \to L\rangle$ and its arbitrary explication $s \colon P \to A$.*

(i) *The explication s is universal.*
(ii) *The trace s^{op} is a regular* c-DESC*-epimorphism.*
(iii) *The counit equality $\varepsilon_P \circ \mathrm{sig}^*(l) = s^{\mathrm{op}} \circ \varepsilon_A$ determines a pushout in* c-DESC.

Proof. (i) $\Longrightarrow$ (iii). Choose an arbitrary pair of c-DESC-morphisms $y \colon \mathrm{sig}^*(L) \to X \leftarrow A : a$ with $y \circ \mathrm{sig}^*(l) = a \circ \varepsilon_A$. Applying the functor sig to this equality we obtain $\mathrm{sig}(y) \circ l = \mathrm{sig}(a)$, so there exists an AO-morphism $\langle a, \mathrm{sig}(y)\rangle \colon S \to \langle X, 1_{\mathrm{sig}(X)}\rangle$. Let $q \colon P \to X$ be its explication along s and 1_X, so $q \circ s^{\mathrm{op}} = a$. We have

$$(q \circ \varepsilon_P) \circ \mathrm{sig}^*(l) = q \circ s^{\mathrm{op}} \circ \varepsilon_A = a \circ \varepsilon_A = y \circ \mathrm{sig}^*(l),$$

so $q \circ \varepsilon_P = y$ since the morphism l has a right inverse. Since ε_P is an epimorphism (see beginning of the proof of Proposition 2), we conclude that q is a colimit arrow of a diagram $\mathrm{sig}^*(l) \colon \mathrm{sig}^*(L) \leftarrow \mathrm{sig}^*\big(\mathrm{sig}(A)\big) \to A : \varepsilon_A$ whose colimit is determined by the counit equality.

$$\begin{tikzcd}
\operatorname{sig}^*\big(\operatorname{sig}(A)\big) \arrow[r, "\varepsilon_A"] \arrow[d, "\operatorname{sig}^*(l)"'] & A \arrow[d, dashed, "s^{\mathrm{op}}"'] \arrow[rdd, "a"] & \\
\operatorname{sig}^*(L) \arrow[r, dashed, "\varepsilon_P"] \arrow[rrd, "y"'] & P \arrow[rd, dotted, "q"] & \\
 & & X
\end{tikzcd}$$

(iii) $\Longrightarrow$ (ii). An edge of a pushout parallel to a regular epimorphism is a regular epimorphism itself (see the dual statement in [1, Proposition 11.18(2)]). Since all retractions are regular epimorphisms [1, Proposition 7.75(1)], the edge of the colimit s^{op} parallel to retraction $\operatorname{sig}^*(l)$ is a regular epimorphism.

(ii) $\Longrightarrow$ (i). Assume that s^{op} is a coequalizer of a pair of c-DESC-morphisms $u, v\colon Q \rightrightarrows A$. Choose an arbitrary AO-model R with an explication r and an arbitrary AO-morphism $\langle h, b\rangle\colon S \to R$, assuming that $p = r^{\mathrm{op}} \circ h$ so $\operatorname{sig}(p) = b \circ l$. We have

$$\operatorname{sig}(p \circ u) = b \circ l \circ \operatorname{sig}(u) = b \circ \operatorname{sig}(s^{\mathrm{op}} \circ u) = b \circ \operatorname{sig}(s^{\mathrm{op}} \circ v) = \operatorname{sig}(p \circ v).$$

Hence $p \circ u = p \circ v$ since the functor sig is faithful. By the definition of a coequalizer, there exists a c-DESC-morphism q such that $q \circ s^{\mathrm{op}} = p$. By Definition 13, this morphism is an explication of the action of the morphism $\langle h, b\rangle$. □

Proposition 9. *All* r-DESC-*objects and universal explications of all aspectual core objects comprise a subcategory in* r-DESC *that contains all invertible refinements.*

Proof. Recall that all retractions are regular epimorphisms, so every invertible refinement s satisfies condition (ii) of Proposition 8. In particular, the identity morphism 1_A is a universal explication of AO-model $\langle A, 1_{\operatorname{sig}(A)}\rangle$. In addition, if $s\colon P \to A$ and $u\colon Q \to P$ are universal explications (of AO-models $\langle A, \operatorname{sig}(s^{\mathrm{op}})\rangle$ and $\langle P, \operatorname{sig}(u^{\mathrm{op}})\rangle$, respectively), then $v = s \circ u$ is a universal explication (of AO-model $\langle A, \operatorname{sig}(v^{\mathrm{op}})\rangle$), by implication (iii) $\Longrightarrow$ (i) of Proposition 8 and has the capability to compose pushouts (see dual statement in [1, Proposition 11.10(1)]): since two squares that comprise the diagram below are pushouts, the frame rectangle is a pushout as well.

$$\begin{tikzcd}
\operatorname{sig}^*\big(\operatorname{sig}(A)\big) \arrow[r, "\varepsilon_A"] \arrow[d, "\operatorname{sig}^*\big(\operatorname{sig}(s^{\mathrm{op}})\big)"'] & A \arrow[d, dashed, "s^{\mathrm{op}}"] \\
\operatorname{sig}^*\big(\operatorname{sig}(P)\big) \arrow[r, dashed, "\varepsilon_P"] \arrow[d, "\operatorname{sig}^*\big(\operatorname{sig}(u^{\mathrm{op}})\big)"'] & P \arrow[d, dashed, "u^{\mathrm{op}}"] \\
\operatorname{sig}^*\big(\operatorname{sig}(Q)\big) \arrow[r, dashed, "\varepsilon_Q"] & Q
\end{tikzcd}$$
□

Theorem 2. *There exists an AO-extension*

$$\begin{aligned} \operatorname{ac}\colon & \text{c-DESC} \hookrightarrow \text{c-AO} \\ \colon & A \mapsto \langle A, 1_{\operatorname{sig}(A)}\rangle,\ g \mapsto \langle g, \operatorname{sig}(g)\rangle \end{aligned}$$

such that $\operatorname{ac}^*(f)$ *explicates every* c-AO-*morphism* f *and* ac-*modularization of the aspectual structure of every* c-AO-*object is a trace of its universal explication.*

Proof. By Proposition 9, $\operatorname{ac}(A) \in \operatorname{Ob}\text{c-AO}$ for each c-DESC-object A, so the pair of maps ac is indeed a functor. By Theorem 1, the functor $\operatorname{mod} \circ \operatorname{cao}$ is right adjoint to it, where $\operatorname{cao}\colon \text{c-AO} \hookrightarrow \text{AO}$ is a full embedding. Further, let v be a map that sends each c-AO-object to some universal explication. Since every trace is an epimorphism (see Proposition 3), an explication q of a c-AO-morphism $f\colon S \to R$ is determined by the equality $q \circ v_S^{\mathrm{op}} = v_R^{\mathrm{op}} \circ \operatorname{mod}(f)$ uniquely. A map that sends f to q is a morphism function of a functor ac* that is left adjoint to ac, and the family of c-AO-morphisms $\langle \upsilon_S^{\mathrm{op}}, 1_{\mathrm{str}(S)}\rangle \colon S \to \langle \mathrm{dom}\, \upsilon_S, 1_{\mathrm{str}(S)}\rangle$, $S \in \mathrm{Ob}\,$c-AO, is the unit of this adjunction. We verify that the counit of this adjunction is an identity: for arbitrary c-DESC-object P, c-AO-object S, and c-DESC-morphism $p\colon \mathrm{ac}^*(S) \to P$, and we find a unique c-AO-morphism $r\colon S \to \langle P, 1_{\mathrm{sig}(P)}\rangle$ with $\mathrm{ac}^*(r) = p$. By definition, $p \circ \upsilon_S^{\mathrm{op}} = 1_P \circ \mathrm{mod}(r)$. Hence $r = \langle p \circ \upsilon_S^{\mathrm{op}}, \mathrm{sig}(p)\rangle$, and it is impossible to define the morphism r otherwise since the functor mod is faithful (Theorem 1). □

Corollary 2.1. *A universal explication of an AO-model is unique up to isomorphism provided that it exists.*

Definition 14. A *core AO-technology* over an arbitrary formal design technology **AR** is a modularizable AO-technology over **AR** generated by the functor ac.

For formal technologies over which not all AO-models have a universal explication, weaker approaches to modularization of aspectual structure exist. In particular, an aspectual labeling of an arbitrary AO-model can be represented at modular level by a partial morphism of a modular base. Recall that a partial morphism of an object X to Y in a category C is a pair of C-morphisms with common beginning, one of which is a monomorphism and is directed to X (it delimits a "part" of object X on which the partial morphism is defined) and another is arbitrary and is directed to Y (it represents the action of a partial morphism) [1, Definition 28.1(1)]. If the category C has enough pullbacks, then composition of partial morphisms can be defined: composition of a pair of arbitrary partial morphisms $m\colon X \leftarrow A \to Y\ {:}f$ and $m'\colon Y \leftarrow B \to Z\ {:}f'$ is a partial morphism $m \circ m''\colon X \leftarrow G \to Z\ {:}f' \circ g$, where $m''\colon A \leftarrow G \to B\ {:}g$ are edges of pushout $f \circ m'' = g \circ m'$ (m'' is a monomorphism since monomorphisms are stable under pullbacks [1, Proposition 11.18]). It can be verified directly that the class of all C-objects and the class of all their partial morphisms with such a composition law comprise a category (an identity morphism of an object T in it looks like $1_T\colon T \leftarrow T \to T\ {:}1_T$, and, more generally, every C-morphism $p\colon T \to S$ induces a partial morphism $1_T\colon T \leftarrow T \to S\ {:}p$, so C is included into this category as a subcategory).

Labeling of an arbitrary AO-model induces a partial c-DESC-morphism as follows. Recall that the counit ε of the adjunction $\mathrm{sig}^* \dashv \mathrm{sig}$ consists of monomorphisms (see the beginning of the proof of Proposition 2). It is possible to delimit a discrete "part" in the modular base of an AO-model by the counit and act on it by the discrete implementation of the aspectual labeling.

Definition 15. A *partial modularization* of an arbitrary AO-model $\langle A, l\colon \mathrm{sig}(A) \to L\rangle$ is the following partial c-DESC-morphism from the modular base A to discrete implementation of the aspectual structure L:

$$\varepsilon_A\colon A \leftarrow \mathrm{sig}^*\big(\mathrm{sig}(A)\big) \to \mathrm{sig}^*(L)\ {:}\mathrm{sig}^*(l).$$

Obviously a partial modularization of an AO-model does not represent a refinement that produces the modular base of the AO-model from its aspectual structure since the action of a partial modularization might not be a trace of a refinement. Nevertheless an "approximation" of modularization of an aspectual structure can be constructed using a colimit of a c-DESC-diagram that represents a partial modularization (provided that it exists: an edge of a colimit of a partial morphism parallel to its action can be considered as a universal extension of a partial morphism to all its domain). In literature, colimits of diagrams that represent partial morphisms of certain specific kind are employed at formalizing MDE technologies to calculate results of multistep model editing procedures [32]. In essence, a connection (in a sense explained in Sec. 1) of the modular base with the discrete implementation of the aspectual structure of an AO-model is constructed. By Proposition 8, for each AO-model that belongs to an aspectual core precisely its universal explication is obtained this way. Here we show that a modularization of the aspectual structure of every modularizable AO-model can be constructed this way since the following weak form of Proposition 8 holds.

Proposition 10. *The diagram of a partial modularization of every modularizable AO-model has a colimit whose edge parallel to the action of the partial modularization is a modularization of the aspectual structure of the model.*

Proof. Let $\mathrm{am}\colon \text{c-DESC} \to \text{m-AO}$ be an arbitrary AO-extension, and let χ be the unit of the adjunction $\mathrm{am}^* \dashv \mathrm{am}$. Choose an arbitrary modularizable AO-model $S = \langle A, l\colon \mathrm{sig}(A) \to L\rangle \in \mathrm{Ob}\, \text{m-AO}$. Recall that $\mathrm{str}(\chi_S) = 1_L$, so $i \circ \mathrm{int}(\chi_S) = l$, where $i = \mathrm{asp}\Big(\mathrm{am}\big(\mathrm{am}^*(S)\big)\Big)$ is an isomorphism (by Proposition 6). Let $u = \varepsilon_{\mathrm{am}^*(S)} \circ \mathrm{sig}^*(i^{-1})$ and verify similarly to the proof of implication (i) $\Longrightarrow$ (iii) of Proposition 8 that the equality $\mathrm{am}_*(\chi_S) \circ \varepsilon_A = u \circ \mathrm{sig}^*(l)$ (the counit equality for the modularization of aspectual structure) determines a pushout, i.e., a colimit of the diagram of the partial modularization of AO-model S. Indeed, an arbitrary pair of c-DESC-morphisms $a\colon A \to X \leftarrow \mathrm{sig}^*(L) :\! y$ that satisfies the condition $a \circ \varepsilon_A = y \circ \mathrm{sig}^*(l)$ induces a m-AO-morphism $\langle a, j \circ \mathrm{sig}(y)\rangle\colon S \to \mathrm{am}(X)$, where $j = \mathrm{asp}\big(\mathrm{am}(X)\big)$. By the definition of adjunction, there exists a c-DESC-morphism $q\colon \mathrm{am}^*(S) \to X$ such that $\mathrm{am}(q) \circ \chi_S = \langle a, j \circ \mathrm{sig}(y)\rangle$. Applying the functor am_* to this equality we obtain $q \circ \mathrm{am}_*(\chi_S) = a$. So

$$\mathrm{sig}(q \circ u) \circ l = \mathrm{sig}(q) \circ i^{-1} \circ l = \mathrm{sig}\big(q \circ am_*(\chi_S)\big) = \mathrm{sig}(a \circ \varepsilon_A) = \mathrm{sig}(y) \circ l.$$

Hence $q \circ u = y$ since the morphism l has a right inverse and the functor sig is faithful. Since $\varepsilon_{\mathrm{am}^*(S)}$ is an epimorphism we conclude that q is a colimit arrow of a diagram of the partial modularization of modularizable AO-model S.

$$\begin{tikzcd}
\mathrm{sig}^*\big(\mathrm{sig}(A)\big) \arrow[rr, "\varepsilon_A"] \arrow[d, "\mathrm{sig}^*(l)"'] & & A \arrow[d, dashed, "\mathrm{am}_*(\chi_S)"'] \arrow[rdd, "a"] & \\
\mathrm{sig}^*(L) \arrow[r, dashed, "\mathrm{sig}^*(i^{-1})"] \arrow[rrrd, "y"'] & \mathrm{sig}^*(L) \arrow[r, dashed, "\varepsilon_{\mathrm{am}^*(S)}"] & \mathrm{am}^*(S) \arrow[rd, dotted, "q"'] & \\
 & & & X
\end{tikzcd}$$

□

Proposition 11. *A modularization of the aspectual structure of every modularizable AO-model is a regular epimorphism.*

Proof. This proposition follows from Proposition 10 similarly to implication (iii) $\Longrightarrow$ (ii) of Proposition 8. □

4. Design of Design Technologies

The category-theoretic approach developed above allows constructing formal technologies for design of technologies that comprise the theoretical basis of MDE. Denote by **CONF**, **SPEC**, and **ARCH** categories that consist of all formal configuration, specification, and design technologies, respectively, and all their morphisms. There exist obvious "forgetful" functors **desc**: **CONF** $\to$ **CAT**, **conf**: **SPEC** $\to$ **CONF**, and **spec**: **ARCH** $\to$ **SPEC**.

The functor **desc**: $\langle \text{c-DESC}, \mathrm{Conf}\rangle \mapsto \text{c-DESC}$ is faithful and has a left adjoint **desc***: **CAT** $\to$ **CONF**: $\text{c-DESC} \mapsto \langle \text{c-DESC}, \varnothing\rangle$ with an identity as the adjunction unit. Hence it can be employed to extract interfaces from formal configuration technologies. Capabilities to assemble complex configuration technologies are determined by the class of all **CONF**-diagrams that are sent to preconfigurations by the functor **D**(**desc**) which will be denoted as DSConf. The following characterization of the class of all **desc**-preconfigurations holds (cf. [26, Proposition 15]).

Proposition 12. *A diagram $\Sigma\colon X \to$ **CAT** that has a colimit is a* **desc***-preconfiguration if and only if each edge of its colimit σ_I, $I \in \mathrm{Ob}\, X$, preserves colimits of all $\Sigma(I)$-diagrams whose colimits are preserved by all functors $\Sigma(f)$ where f is an X-morphism with a domain I.*

Proof. Consider a diagram $\Sigma\colon X \to$ **CAT** with a colimit whose vertex will be denoted by S and edges will be denoted by $\sigma_I\colon \Sigma(I) \to S$, $I \in \mathrm{Ob}\, X$. If the diagram Σ satisfies the condition of the proposition, then every **CONF**-diagram $\Xi \in \mathbf{Ddesc}^{-1}(\{\Sigma\})$ has a colimit with vertex $\Big\langle S, \bigcup\limits_{I \in \mathrm{Ob}\, X} \sigma_I \circ \mathrm{Conf}_I \Big\rangle$, where Conf_I

is the class of all configurations of the technology $\Xi(I)$, and edges σ_I. Conversely, assume that there exist an X-object I and a $\Sigma(I)$-diagram Δ such that all functors $\Sigma(f)$ with $f \in \operatorname{Mor} X$ and $\operatorname{dom} f = I$ preserve colimits of diagram Δ, but σ_I does not preserve them. The **CONF**-diagram

$$\begin{aligned}\Theta &\colon X \to \mathbf{CONF}\\ &\colon J \mapsto \langle \Sigma(J),\ \{\Sigma(f)\circ\Delta \mid f\colon I \to J \in \operatorname{Mor} X\}\rangle,\ h \mapsto \Sigma(h)\end{aligned}$$

cannot have a colimit with a vertex sent to S by the functor **desc** and edges σ_I. Consequently, the functor **desc** does not preserve colimits of diagram Θ, notwithstanding the fact that $\mathbf{desc}\circ\Theta = \Sigma$. □

In particular, all discrete **CAT**-diagrams are preconfigurations. In contrast, consider the following example of a **CAT**-diagram that has a colimit but does not satisfy the condition of Proposition 12. Let $\boldsymbol{U}$ be a category $1 \to 0 \leftarrow 1'$, Σ be a **CAT**-diagram of embeddings $\boldsymbol{U} \hookleftarrow \boldsymbol{U}\setminus\{0\} \hookrightarrow \boldsymbol{U}$. The two-dot discrete diagram $\ulcorner 1 \urcorner \amalg \ulcorner 1' \urcorner$ has a colimit in $\boldsymbol{U}$ but not in a category $\operatorname{colim}(\Sigma)$.

As far as refinements of formal technologies are concerned, they shall represent processes of software systems implementation, i.e., transitions from configuration technologies of interfaces to configuration technologies of their implementations. An arbitrary functor $\operatorname{sig}\colon \text{c-DESC} \to \text{SIG}$ can be treated as a rule of extracting interfaces from models in a technology $\langle\text{c-DESC}, \text{Conf}\rangle$ provided that it satisfies conditions (ii)–(v) of Definition 1. It can be considered as **CONF**-morphism $\operatorname{sig}\colon \langle\text{c-DESC}, \text{Conf}\rangle \to \langle \text{SIG}, \operatorname{sig}\circ\text{Conf}\rangle$ that satisfies these conditions. Denote by r-CONF the class of all such **CONF**-morphisms. The *formal technology of design of configuration technologies* is a quadruple

$$\mathbf{SCONF} = \langle \mathbf{CONF},\ \text{DSConf},\ \mathbf{desc},\ (\operatorname{Ob}\mathbf{CONF}, \text{r-CONF})^{\text{op}}\rangle.$$

A technology of design of specification technologies is constructed on the basis of the technology **SCONF**. Remarkably formal constructions of aspect-oriented approach are employed for this: as shown below, a functor of extracting interfaces that enhances an arbitrary formal configuration technology to a specification technology is precisely its explicable labeling in the technology **SCONF**. Consequently, a core AO-technology over **SCONF** is duly called the *formal technology of design of specification technologies* (see Corollary 3.1 below) and denoted as **SSPEC**. So a category of interfaces represents an aspectual structure of their implementations (for example, an assortment of roles that components play within systems [18]). Therefore, it is possible to reduce the costs associated with assembling complex specification technologies by employing aspect-oriented techniques. Moreover, we will prove that the construction of an AO-technology is, in turn, consistent with respect to the aspectual structure of technologies: if a modular design technology is aspectually complete (in the sense that assembling of systems in it does not invade aspectual structure), then its transformation to an AO-technology over it is a refinement. In view of this we introduce the following concept.

Definition 16. A formal design technology **AR** is called *aspectually complete* if the class Conf is aspectually closed with respect to the functor mod.

Proposition 13. *A formal technology* **AR** *is aspectually complete if and only if the class of all configurations of every AO-technology over* **AR** *coincides with* $\mathbf{D}\text{mod}^{-1}(\text{Conf})$.

Proof. The sufficiency of the condition directly follows from Definition 8. To prove the necessity, consider an arbitrary functor ai that generates an AO-technology over **AR**. By Corollary 1.2 and Definition 7, we have

$$\mathbf{D}\text{int}^{-1}(\text{AOInt}_{\text{int}}) \subseteq \mathbf{D}\text{ai}^{-1}(\text{AOInt}_{\text{ai}}) \subseteq \mathbf{D}\text{mod}^{-1}(\text{Conf}).$$

Since $\mathbf{D}\text{int}^{-1}(\operatorname{sig}\circ\text{Conf}) = \mathbf{D}\text{mod}^{-1}(\text{Conf})$, by assumption $\operatorname{sig}\circ\text{Conf} \subseteq \text{AOInt}_{\text{int}}$. Hence

$$\mathbf{D}\text{mod}^{-1}(\text{Conf}) \subseteq \mathbf{D}\text{ai}^{-1}(\text{AOInt}_{\text{ai}}) \subseteq \mathbf{D}\text{mod}^{-1}(\text{Conf}).$$ □

For example, the discrete event modeling technology SM is aspectually complete (see Corollary 4.6), while the technology of design of configuration technologies **SCONF** is not (see below).

Theorem 3. *An AO-model over* **SCONF** *has an explication if and only if it is a specification technology; all of its explications are universal. A refinement of AO-models over* **SCONF** *has an explication if and only if both its domain and codomain are specification technologies.*

Proof. Choose an arbitrary formal specification technology $\mathrm{SC} = \langle \text{c-DESC}, \mathrm{Conf}, \mathrm{sig}\colon \text{c-DESC} \to \mathrm{SIG}\rangle$ and let $\mathrm{SConf} = \mathrm{sig} \circ \mathrm{Conf}$. The morphism of formal configuration technologies $\mathrm{sig}\colon \langle \text{c-DESC}, \mathrm{Conf}\rangle \to \langle \mathrm{SIG}, \mathrm{SConf}\rangle \in \text{r-CONF}$ is a trace of the unique explication of the technology SC. This refinement is invertible since the functor sig^*, which is left adjoint and right inverse to sig, preserves all colimits [29, Sec. V.5] and thus induces a **CONF**-morphism $\mathrm{sig}^*\colon \langle \mathrm{SIG}, \mathrm{SConf}\rangle \to \langle \text{c-DESC}, \mathrm{Conf}\rangle$ (hence the technology **SCONF** supports tracing). By Proposition 9, an explication of the technology SC is universal.

Let $\langle \mathrm{cm}, \mathrm{sm}\rangle\colon \mathrm{SC} \to \langle \text{c-DESC}', \mathrm{Conf}', \mathrm{sig}'\colon \text{c-DESC}' \to \mathrm{SIG}'\rangle$ be a **SPEC**-morphism with $\mathrm{cm} \in \text{r-CONF}$, so in particular $\mathrm{sm} \circ \mathrm{sig} = \mathrm{sig}' \circ \mathrm{cm}$ and $\mathrm{Conf}' = \mathrm{cm} \circ \mathrm{Conf}$. Consequently, there exists a **CONF**-morphism $\mathrm{sm}\colon \langle \mathrm{SIG}, \mathrm{SConf}\rangle \to \langle \mathrm{SIG}', \mathrm{sig}' \circ \mathrm{Conf}'\rangle$. We need to verify that $\mathrm{sm} \in \text{r-CONF}$. Consider a functor $\mathrm{scm} = \mathrm{sig}' \circ \mathrm{cm} \in \text{r-CONF}$. The functor $\mathrm{scm}^* = \mathrm{cm}^* \circ \mathrm{sig}'^*$ is left adjoint to it (denote by ν the counit of this adjunction). The equality $\mathrm{sm} \circ \mathrm{sig} = \mathrm{scm}$ implies that $\mathrm{sm} \circ \mathrm{SConf} = \mathrm{scm} \circ \mathrm{Conf} = \mathrm{sig}' \circ \mathrm{Conf}'$.

$$\begin{tikzcd}
\text{c-DESC} \arrow[r, "\mathrm{sig}"] \arrow[d, "\mathrm{cm}"'] \arrow[dr, "\mathrm{scm}"] & \mathrm{SIG} \arrow[d, "\mathrm{sm}"] \\
\text{c-DESC}' \arrow[r, "\mathrm{sig}'"] & \mathrm{SIG}'
\end{tikzcd}$$

This equality also implies that $\mathrm{sm} = \mathrm{scm} \circ \mathrm{sig}^*$, so the functor sm is faithful (as a composition of faithful functors) and functor $\mathrm{sig} \circ \mathrm{scm}^*$ is left adjoint to it with an identity as the adjunction unit. Indeed, $\mathrm{sm} \circ \mathrm{sig} \circ \mathrm{scm}^* = \mathrm{scm} \circ \mathrm{scm}^* = 1_{\mathrm{SIG}'}$ and for an arbitrary $A \in \mathrm{Ob}\,\mathrm{SIG}'$, $X \in \mathrm{Ob}\,\mathrm{SIG}$, $f\colon A \to \mathrm{sm}(X)$ there exists a (unique since the functor sm is faithful) SIG-morphism $g\colon \mathrm{sig}\big(\mathrm{scm}^*(A)\big) \to X$ that satisfies the condition $\mathrm{sm}(g) = f$: let $g = \mathrm{sig}\big(\nu_{\mathrm{sig}^*(X)} \circ \mathrm{scm}^*(f)\big)$, whence

$$\mathrm{sm}(g) = \mathrm{scm}\big(\nu_{\mathrm{sig}^*(X)} \circ \mathrm{scm}^*(f)\big) = 1_{\mathrm{scm}(\mathrm{sig}^*(X))} \circ \mathrm{scm}\big(\mathrm{scm}^*(f)\big) = f.$$

Verify that the functor sm lifts colimits of a diagram $\mathrm{sig} \circ \Delta$ for each diagram $\Delta \in \mathrm{Conf}$. If ξ is a colimit of diagram $\mathrm{sm} \circ (\mathrm{sig} \circ \Delta)$, then there exists a colimit δ of diagram Δ such that $\mathrm{sm} \circ \mathrm{sig} \circ \delta = \xi$ since the functor scm by its construction lifts colimits of all diagrams from the class Conf. So, by Proposition 2, cocone $\mathrm{sig} \circ \delta$ is a colimit of diagram $\mathrm{sig} \circ \Delta$ which is sent to ξ by the functor **D**sm.

Finally, consider arbitrary diagrams $\Delta \in \mathrm{Conf}$ and $\Xi \in \mathrm{Ob}\,\mathbf{D}\mathrm{SIG}$ such that $\mathrm{sm} \circ (\mathrm{sig} \circ \Delta) = \mathrm{sm} \circ \Xi$. We have $\mathrm{scm} \circ \Delta = \mathrm{scm} \circ \mathrm{sig}^* \circ \Xi$. Hence $\mathrm{sig}^* \circ \Xi \in \mathrm{Conf}$. Consequently, $\Xi = \mathrm{sig} \circ \mathrm{sig}^* \circ \Xi \in \mathrm{SConf}$. □

Corollary 3.1. *The core AO-technology over* **SCONF** *has* **SPEC** *as a category of models and* **conf** *as a functor of extracting interfaces.*

Corollary 3.2. *Let* $\mathbf{AR} = \langle \text{c-DESC}, \mathrm{Conf}, \mathrm{sig}, \text{r-DESC}\rangle$ *be an arbitrary formal design technology. For each functor* ai *that generates an AO-technology over* **AR** *the functor* $\mathrm{mod}\colon \mathrm{AO} \to \text{c-DESC}$ *constructed from its constituents induces an* **ARCH**-*morphism of* $\mathrm{AO}_{\mathrm{ai}}(\mathbf{AR})$ *to* **AR**. *The technology* **AR** *is aspectually complete if and only if the functor* **spec** *sends this morphism to a trace of an explicable refinement of specification technologies.*

Proof. Consider triple $\mathrm{ao}_{\mathrm{ai}} = \langle \mathrm{mod}, \mathrm{int} \circ \mathrm{ai}^*, \mathrm{mod}(-^{\mathrm{op}})^{\mathrm{op}}\rangle$. By condition (ii) of Corollary 1.1 and Definition 7, this triple is an **ARCH**-morphism of $\mathrm{AO}_{\mathrm{ai}}(\mathbf{AR})$ to **AR**.

$$\begin{tikzcd}
\langle \mathrm{AO}, \arrow[d, "\mathrm{mod}"'] & \mathbf{D}\mathrm{ai}^{-1}(\mathrm{AOInt}_{\mathrm{ai}}) \arrow[d, phantom, "\mathbf{D}\mathrm{mod}\ |\!\cap"] & \mathrm{AO} \arrow[r, "\mathrm{ai}"] \arrow[d, "\mathrm{mod}"'] & \mathrm{INT}, \arrow[d, "\mathrm{int}\circ\mathrm{ai}^*"] & \text{tr-AO}^{\mathrm{op}}\rangle \arrow[d, "\mathrm{mod}(-^{\mathrm{op}})^{\mathrm{op}}"] \\
\langle \text{c-DESC}, & \mathrm{Conf}, & \text{c-DESC} \arrow[r, "\mathrm{sig}"] & \mathrm{SIG}, & \text{r-DESC}\rangle
\end{tikzcd}$$

If **AR** is aspectually complete, then, by Theorem 1 and Proposition 13, $\mathbf{conf}\big(\mathbf{spec}(\mathrm{ao}_{\mathrm{ai}})\big) \in \text{r-CONF}$, so the **SPEC**-morphism $\mathbf{spec}(\mathrm{ao}_{\mathrm{ai}})$ is a trace of a refinement in the technology **SSPEC**, and this refinement is explicable by Theorem 3. Otherwise, $\mathrm{mod} \circ \mathbf{D}\mathrm{mod}^{-1}(\mathrm{AOInt}_{\mathrm{mod}}) \neq \mathrm{Conf}$, so the **SPEC**-morphism $\mathbf{spec}(\mathrm{ao}_{\mathrm{mod}})$ is not a trace of a refinement. $\square$

Note that the formal technology **SCONF** is aspectually incomplete itself. In order to verify this, we construct a **SPEC**-diagram Ξ that has a colimit in comma category $\mathbf{desc} \downarrow \mathbf{CAT}$, whose vertex does not belong to a category of AO-models over **SCONF** notwithstanding the fact that **CONF**-diagram $\mathbf{conf} \circ \Xi$ is a configuration in **SCONF**. Consider the following categories and functors:

$$
\begin{aligned}
&X = (\{A, B\},\ \{1_A,\ r\colon A \to A,\ 1_B\colon B \to B,\ p_1, p_2, q_1, q_2\colon A \to B\}),\\
&\quad \text{where } r \circ r = 1_A,\ \ p_i \circ r = q_i \quad (i = 1, 2),\\
&X_i = (\{A, B\},\ \{1_A, r, 1_B, p_i, q_i\}) \subseteq X \quad (i = 1, 2),\\
&X_0 = (\{A, B\},\ \{1_A, r, 1_B\}) \subseteq X,\\
&Y = (\{A\},\ \{1_A, r\}) \subseteq X_0,\\
&\mathrm{ex}_i\colon X_i \hookrightarrow X \quad (i = 0, 1, 2),\\
&\mathrm{jx}_i\colon X_0 \hookrightarrow X_i \quad (i = 1, 2),\\
&\mathrm{yx}\colon Y \hookrightarrow X_0,\\
&\mathrm{xy}\colon X \to Y\colon A \mapsto A,\ B \mapsto A,\ r \mapsto r,\ p_i \mapsto r,\ q_i \mapsto 1_A \quad (i = 1, 2),\\
&\mathrm{xy}_i = \mathrm{xy} \circ \mathrm{ex}_i \quad (i = 0, 1, 2).
\end{aligned}
$$

Every functor xy_i $(i = 1, 2)$ is faithful, and the functor $\mathrm{jx}_i \circ \mathrm{yx}$ is left adjoint to it with an identity as the adjunction unit. Denote by Ξ the following **SPEC**-diagram that consists of embeddings of formal technologies.

$$\langle \mathrm{jx}_1, \mathrm{xy}_0\rangle\colon \langle X_1, \varnothing, \mathrm{xy}_1\rangle \hookleftarrow \mathbf{conf}^*\big(\mathbf{desc}^*(X_0)\big) \hookrightarrow \langle X_2, \varnothing, \mathrm{xy}_2\rangle :\!\langle \mathrm{jx}_2, \mathrm{xy}_0\rangle,$$

$$\begin{tikzcd}
A \arrow[loop above, "r"] \arrow[r, shift left, "p_1"] \arrow[r, shift right, "q_1"'] \arrow[d] & B \arrow[dl] & \hookleftarrow & A \arrow[loop above, "r"] \arrow[d] & B \arrow[d] & \hookrightarrow & A \arrow[loop above, "r"] \arrow[r, shift left, "p_2"] \arrow[r, shift right, "q_2"'] \arrow[d] & B \arrow[dl] \\
A \arrow[loop above, "r"] & & & A \arrow[loop above, "r"] & B & & A \arrow[loop above, "r"] &
\end{tikzcd}$$

Let $\mathbf{dc}\colon \mathbf{SPEC} \hookrightarrow \mathbf{desc} \downarrow \mathbf{CAT}$ be an embedding of categories. Diagram $\mathbf{dc} \circ \Xi$ has a colimit in the category $\mathbf{desc} \downarrow \mathbf{CAT}$ with edges

$$\langle \mathrm{ex}_1, 1_Y\rangle\colon \langle X_1, \varnothing, \mathrm{xy}_1\rangle \hookrightarrow \langle X, \varnothing, \mathrm{xy}\rangle \hookleftarrow \langle X_2, \varnothing, \mathrm{xy}_2\rangle :\!\langle \mathrm{ex}_2, 1_Y\rangle.$$

The functor xy is not faithful, so $\mathrm{colim}(\mathbf{dc} \circ \Xi)$ is not an AO-model over **SCONF**. Hence diagram Ξ is not aspectually determined. At the same time, **CONF**-diagram $\mathbf{conf} \circ \Xi$ is a configuration in **SCONF**. Indeed, consider an arbitrary diagram $\Delta\colon Z \to X_1$. If its schema is empty, then it has no colimit since neither A nor B are initial X_1-objects. If the category Z is nonempty and connected, then diagram Δ has a colimit if and only if for every $I \in \Delta^{-1}(A)$ each of the following sets consists of at most one element:

(i) $\Delta\big(\mathrm{Mor}(I, J)\big)$ for each $J \in \Delta^{-1}(A)$,

(ii) $\displaystyle\bigcup_{J \in \Delta^{-1}(B)} \Delta\big(\mathrm{Mor}(I, J)\big)$

(if $\Delta(\mathrm{Ob}\, Z) = \{A\}$, then $\mathrm{colim}(\Delta) = A$; otherwise, $\mathrm{colim}(\Delta) = B$ provided that the condition above holds). If the category Z is discrete and contains more than one object, then diagram Δ has a colimit if and only if $\Delta(\mathrm{Ob}\, Z) = \{B\}$ (and $\mathrm{colim}(\Delta) = B$). In all cases described above, if diagram Δ has a colimit, then the functor ex_1 preserves it. Similarly, the functor ex_2 preserves colimits of all X_2-diagrams. By Proposition 12, the diagram $\mathbf{desc} \circ \mathbf{conf} \circ \Xi$ is a **desc**-preconfiguration as desired.

On the basis of the results obtained above, a formal technology of design of design technologies is constructed. Specification technologies fail to serve as interfaces for design technologies since the functor $\mathbf{spec}\colon \mathbf{ARCH} \to \mathbf{SPEC}$ that "forgets" a category of refinements is not faithful. Nevertheless, it has a left adjoint

$$\begin{aligned}\mathbf{spec}^*\colon &\mathbf{SPEC} \to \mathbf{ARCH}\\ &: \langle \text{c-DESC}, \text{Conf}, \text{sig}\rangle \mapsto \langle \text{c-DESC}, \text{Conf}, \text{sig}, \text{Iso}\,\text{c-DESC}\rangle,\end{aligned}$$

where the unit of this adjunction is an identity, and $\mathbf{spec}^* \circ \mathbf{conf}^* \circ \mathbf{desc}^* = \mathbf{triv}$. It is reasonable to extract interfaces from design technologies by the same means as the functor **conf** extracts interfaces from specification technologies, viz. by "forgetting" a functor of extraction of interfaces (sic!). Therefore, an interface of a design technology is a triple $\langle \text{c-DESC}, \text{Conf}, \text{r-DESC}\rangle$ that satisfies conditions (i), (vi)–(viii) of Definition 1. Such triples are called *design formalisms* in [28]. A morphism of a formalism $\langle \text{c-DESC}_1, \text{Conf}_1, \text{r-DESC}_1\rangle$ to a formalism $\langle \text{c-DESC}_2, \text{Conf}_2, \text{r-DESC}_2\rangle$ is a pair of functors $\langle \text{cm}\colon \text{c-DESC}_1 \to \text{c-DESC}_2, \text{rm}\colon \text{r-DESC}_1 \to \text{r-DESC}_2\rangle$ that satisfies conditions (i), (ii), and (iv) of Definition 3. Denote by **FORM** a category of all formalisms and all their morphisms. There exist the following forgetful functors:

$$\begin{aligned}\mathbf{form}\colon &\mathbf{ARCH} \to \mathbf{FORM},\\ &: \langle \text{c-DESC}, \text{Conf}, \text{sig}, \text{r-DESC}\rangle \mapsto \langle \text{c-DESC}, \text{Conf}, \text{r-DESC}\rangle,\\ \mathbf{fconf}\colon &\mathbf{FORM} \to \mathbf{CONF},\\ &: \langle \text{c-DESC}, \text{Conf}, \text{r-DESC}\rangle \mapsto \langle \text{c-DESC}, \text{Conf}\rangle,\end{aligned}$$

that satisfy the condition $\mathbf{fconf} \circ \mathbf{form} = \mathbf{conf} \circ \mathbf{spec}$. The functor **form** is faithful and has a left adjoint

$$\begin{aligned}\mathbf{form}^*\colon &\mathbf{FORM} \to \mathbf{ARCH},\\ &: \langle \text{c-DESC}, \text{Conf}, \text{r-DESC}\rangle \mapsto \langle \text{c-DESC}, \text{Conf}, 1_{\text{c-DESC}}, \text{r-DESC}\rangle\end{aligned}$$

with an identity as the adjunction unit (cf. construction of the functor mod* in the proof of Theorem 1). Configurations of design technologies are selected from preimages of **form**-preconfigurations to ensure consistency with configurations of specification technologies: consider the class of **CONF**-diagrams

$$\begin{aligned}\text{PForm} = \{\Delta \mid &\mathbf{D}(\mathbf{fconf})^{-1}(\{\Delta\}) \subseteq \text{FPConf},\\ &\mathbf{D}(\mathbf{conf})^{-1}(\{\Delta\}) \subseteq \text{CSConf},\\ &\mathbf{D}(\mathbf{conf} \circ \mathbf{spec})^{-1}(\{\Delta\}) \subseteq \text{PCspec}\},\end{aligned}$$

where FPConf is the class of all **form**-preconfigurations, CSConf is the class of all configurations in the technology **SSPEC**, PCspec is the class of all **ARCH**-diagrams with colimits preserved by the functor **spec**. Note that the class PForm contains all discrete **CONF**-diagrams. Refinements of design technologies generalize refinements of specification technologies in order to provide traceability for which they shall be consistent with respect to discrete implementation of interfaces.

Definition 17. An $\mathbf{ARCH}^{\text{op}}$-morphism $\langle \text{cm}, \text{sm}, \text{rm}\rangle^{\text{op}}\colon \mathbf{AR} \to \mathbf{AR}'$ is called a *refinement of design technologies* if it satisfies the following conditions.

(i) Pair $\langle \text{cm}, \text{sm}\rangle$ is a trace of a refinement of a specification technology $\mathbf{spec}(\mathbf{AR})$ to $\mathbf{spec}(\mathbf{AR}')$.
(ii) There exists a functor drm that is right inverse to rm and satisfies the condition $\text{drm}(i) = \text{cm}^*(i)$ for each $i \in \text{Iso}\,\text{codom}\,\text{cm}$.

For example, an **ARCH**-morphism $\text{ao}_{\text{ai}}\colon \text{AO}_{\text{ai}}(\mathbf{AR}) \to \mathbf{AR}$ constructed in the proof of Corollary 3.2 is dual to a refinement in the case where the technology **AR** is aspectually complete, and all refinements in it are traceable (in particular, the discrete event modeling technology SM satisfies this condition). Moreover, under this condition the refinement $\text{ao}_{\text{int}}^{\text{op}}$ is invertible and **ARCH**-morphism right inverse to its trace is induced by an AO-extension: it looks like $\langle \text{am}(-), 1_{\text{SIG}}, \text{am}(-^{\text{op}})^{\text{op}}\rangle\colon \mathbf{AR} \to \text{AO}_{\text{int}}(\mathbf{AR})$, where am is an arbitrary AO-extension of $\mathbf{AR}$. Noninvertible refinements may generally be found among other refinements of the kind $\mathrm{ao}_{\mathrm{ai}}^{\mathrm{op}}$.

Denote by r-ARCH the class of all $\mathbf{ARCH}$-morphisms dual to refinements of design technologies. A *formal technology of design of design technologies* is a quadruple

$$\mathbf{SARCH} = \langle \mathbf{ARCH}, \mathbf{D}(\mathbf{conf} \circ \mathbf{spec})^{-1}(\mathrm{PForm}), \mathbf{form}, (\mathrm{Ob}\,\mathbf{ARCH}, \text{r-ARCH})^{\mathrm{op}} \rangle.$$

All refinements of design technologies are traceable as well as refinements of other kinds of technologies: a right inverse to a **form**-image of an arbitrary r-ARCH-morphism $\langle \mathrm{cm}, \mathrm{sm}, \mathrm{rm} \rangle \colon \mathbf{AR} \to \mathbf{AR}'$ looks like $\langle \mathrm{cm}^*, \mathrm{drm} \rangle \colon \mathbf{form}(\mathbf{AR}') \to \mathbf{form}(\mathbf{AR})$. In addition, the technology **SARCH** is counified as well as **SCONF** and **SSPEC**.

By Corollary 3.1, there exists an **ARCH**-morphism

$$\langle \mathbf{spec}, \mathbf{fconf}, \mathbf{spec}(-^{\mathrm{op}})^{\mathrm{op}} \rangle \colon \mathbf{SARCH} \to \mathbf{SSPEC}.$$

In turn, Corollary 3.2 (upon any choice of the technology **SCONF** for **AR**) along with Proposition 7 implies that there exists an **ARCH**-morphism

$$\langle \mathbf{conf}, \mathbf{desc}, \mathbf{conf}(-^{\mathrm{op}})^{\mathrm{op}} \rangle \colon \mathbf{SSPEC} \to \mathbf{SCONF}.$$

This chain of **ARCH**-morphisms can be extended by one more step by introducing a “formal technology of design of categories of models”: let

$$\mathbf{SCAT} = \big\langle \mathbf{CAT}, \mathbf{desc} \circ \mathrm{DSConf}, 1_{\mathbf{CAT}}, \big(\mathrm{Ob}\,\mathbf{CAT}, \mathbf{desc}(\text{r-CONF})\big)^{\mathrm{op}} \big\rangle.$$

The quadruple **SCAT** is a counified formal design technology that supports tracing: the class of all refinements in it is dual to the class of all faithful functors that have a left adjoint with an identity as the adjunction unit (which are precisely **desc**-labelings). There exists an **ARCH**-morphism $\langle \mathbf{desc}, 1_{\mathbf{CAT}}, \mathbf{desc}(-^{\mathrm{op}})^{\mathrm{op}} \rangle$:$\mathbf{SCONF} \to \mathbf{SCAT}$. The technology **SCAT** is constructed in such a way that this **ARCH**-morphism is a trace of a refinement of a design technology induced by a universal explication of the specification technology **spec**(**SCONF**) (see the first paragraph of the proof of Theorem 3). Thus, configuration technologies are produced from categories of models by means of a refinement in a formal technology of design of design technologies.

5. Technologies Induced by Labeling

Consider the process of performing the major technological procedures of MDE, viz. constructing some domain-specific design technology. The first step is naturally the choice of language for the formal representation of models of programs. The techniques of assembling complex models provided by the language constructs (macro substitution, links resolution, recomposition, and so on) form the morphisms of formal models that induce a category c-DESC. Once it has been built, as a next step one may choose the class of configurations Conf and obtain the configuration technology. However, it is more practical to move straight to the choice of the functor of extracting interface sig since the conditions (ii) and (iii) of Definition 1 dictate its form sufficiently stringently. It shall coincide up to isomorphism with some faithful coreflector in c-DESC (an example was given in Sec. 1 when we considered the discrete event modeling technology SM). The class of configurations can then be defined at interface level via selecting from sig-preconfigurations. The domain-specific specification technology is obtained this way. Complex technologies can be synthesized from technologies of this kind by means of modular and aspect-oriented techniques formalized in a technology for design of specification technologies (cf. Corollary 3.1), e.g., to support several types of models at the same process.

To obtain the design technology it remains to specify the class of refinements Mor r-DESC. As mentioned in the Introduction, the main criterion of optimal choice of refinements is ensuring the widest possible capabilities for traceability. Refinements shall be traceable (in particular, dual to activities of integration), nontrivial (exceed isomorphisms), and at the same time not excessively costly in tracing. An acceptable compromise is a requirement that refinements shall allow tracing inclusion components into the system, viz. integration activities that do not invade the internal structure of components and hence can be traced "for free" [24]. Employ the semantics of traceability presented in Sec. 2. If a component X is included into a system S (i.e., there exists an inclusion represented by a c-DESC-morphism $m\colon X \to S$) and S is further refined into a system T (i.e., there exists a traceable refinement represented by an r-DESC-morphism $r\colon S \to T$), then X shall be included into T in such a way that composition (in the category c-DESC) of a trace $r^{\mathrm{op}}\colon T \to S$ with the inclusion of X into T produces m.

This condition is easy to satisfy at the level of interfaces (for an arbitrary morphism m) by Definition 4: desired inclusion of $\mathrm{sig}(X)$ into $\mathrm{sig}(T)$ looks like $s{\circ}\mathrm{sig}(m)$ where s is a morphism that satisfies the condition $\mathrm{sig}(r^{\mathrm{op}}) \circ s = 1_{\mathrm{sig}(S)}$. Hence a refinement r that provides traceability for inclusions shall allow "lifting" each sig-morphism of $\mathrm{sig}(X)$ to $\mathrm{sig}(T)$, whose composition with $\mathrm{sig}(r^{\mathrm{op}})$ produces a sig-image of given inclusion m, onto the level of models to a morphism of X to T. As we will see later (Definition 18), in category theory this condition turns out to be a generalization of the definition of the concept of an initial morphism.

Therefore, it is reasonable to construct well-traceable refinements starting at the level of interfaces by fixing the class L of all valid labelings of formal models, viz. sig-images of traces. By statement (ii) of Proposition 3, it shall consist of retractions and contain all isomorphisms. Traces of all refinements belong to the class $\mathrm{sig}^{-1}(\mathrm{L})$ so this class can be considered as consisting of integration activities that most heavily invade the structure of components. Therefore, noninvasive inclusions can be defined as integration actions that are "orthogonal" to them. In category theory there is the formal notion of orthogonality which gives the following characterization of the class of all inclusions: any action of integration is uniquely (up to isomorphism) factorized to an inclusion composed with a morphism that is sent to a retraction by the functor sig. Classes of morphisms that induce factorization of such a kind are known as components of factorization system [6] or factorization structure [1]. We will provide the exact definition below while here we draw a classical example: a pair $(\mathrm{Epi}, \mathrm{Mono})$ in the category **Set** is a factorization system where monomorphisms represent inclusion of subsets and an epimorphism labels its domain by elements of its codomain. Orthogonality of inclusions to traces means that at the level of interfaces any activity of integration of a component into a system is reduced to an inclusion of a result of some labeling of the component. In other words, the impact of the environment provided by the system upon the interface of the component consists in "coarsening," viz. cancelling some refinement which is determined uniquely up to isomorphism.

In order to propagate the noninvasive nature of inclusion to the discrete structure of models an extra condition of preservation of discreteness is asserted by which only sig-discrete components may be included into a discrete system. Recall that a c-DESC-object S is called discrete if ε_S is an isomorphism where ε as usual is the counit of the adjunction $\mathrm{sig}^* \dashv \mathrm{sig}$. For example, in technologies over **Set** usually all monomorphisms preserve discreteness in the sense stated above. In addition, in order to provide traceability at the level of configurations the counifiedness condition is asserted. Refinements are selected from the class $\mathrm{sig}^{-1}(\mathrm{L})^{\mathrm{op}}$ by capability to trace inclusions of components. In this way, the class of labelings L induces a formal design technology that we will concisely call an L-technology.

Following [24], we will show that technologies induced by classes of labelings "nicely" yield to aspect orientation. All refinements in them are invertible in the sense of Definition 4, and monomorphic inclusions are noninvasive. The procedure of inducing such technologies can be traced along refinements of specification technologies of a certain kind (it is natural with respect to an appropriate class of **SPEC**-morphisms), in particular, it can be "descended" to a level of interfaces and "lifted" to a level of AO-technologies. Therefore, the category tr-AO can be constructed as a comma category of a certain kind, and (subject to some extra restrictions) aspectual completeness is provided. We will consider the discrete event modeling technology SM as an indicative example of a design technology induced by a class of labelings.

Definition 18. Let ff: $C \to D$ be an arbitrary faithful functor, M be an arbitrary class of C-morphisms. A C-morphism $f\colon T \to S$ is called M-*initial* if for every M-morphism $m\colon X \to S$ and every D-morphism $k\colon \mathrm{ff}(X) \to \mathrm{ff}(T)$ such that $\mathrm{ff}(f) \circ k = \mathrm{ff}(m)$ there exists a C-morphism $k^+\colon X \to T$ such that $\mathrm{ff}(k^+) = k$.

Denote by M-Init the class of all M-initial morphisms. This concept is constructed as a generalization of the known category-theoretic concept of initial morphism [1, Definition 8.6], which is precisely a Mor-initial morphism, i.e., M-initial for any class M. In many familiar cases, category c-DESC has a factorization system of the kind $(\mathrm{sig}^{-1}(\mathrm{Iso}), \text{Mor-Init})$, e.g., when the functor sig is topological [1, Exercise 21M(3)]. This fact can be generalized to M-initial morphisms as follows. Let M be an arbitrary class of morphisms of some category. We will say that a relation M-Difip(e, f) holds between morphisms e and f if for every commutative square $f \circ u = m \circ e$ with $m \in \mathrm{M}$ there exists a unique "diagonal" morphism d such that $d \circ e = u$ and $f \circ d = m$.

$$\begin{tikzcd}
\cdot \arrow[r, "u"] \arrow[d, "e"'] & \cdot \arrow[d, "f"] \\
\cdot \arrow[r, "m"'] \arrow[ur, dotted, "d"] & \cdot
\end{tikzcd}$$

Note that if e is an epimorphism, then it is enough to find a morphism d such that $d \circ e = u$ in order to verify M-Difip(e, f). Recall that relation Difip (diagonal fill-in property), which is the same as our Mor-Difip, is used in category theory for definition of a factorization system. Specifically, a factorization system is a pair of classes of morphisms (E, F) such that:

(i) $\mathrm{Iso} \circ \mathrm{E} \subseteq \mathrm{E}$;
(ii) $\mathrm{F} \circ \mathrm{Iso} \subseteq \mathrm{F}$;
(iii) $\mathrm{F} \circ \mathrm{E} = \mathrm{Mor}$;
(iv) $\mathrm{Difip}(\mathrm{E}, \mathrm{F})$.

We enlist a few properties of an arbitrary factorization system according to [1, Chap. 14]. A (E, F)-factorization of every morphism (i.e., its representation as composition $f \circ e$ with $e \in \mathrm{E}$ and $f \in \mathrm{F}$) is unique up to isomorphism. Class E is closed under composition and contains all isomorphisms. It is weakly left cancellable in the sense that the conditions $e \in \mathrm{E}$ and $e' \circ e \in \mathrm{E}$ imply $e' \in \mathrm{E}$. It is also closed under formation of coproducts and pushouts (provided that these exist) in the sense that a coproduct of E-morphisms and an edge parallel to an E-morphism in a pushout belong to E. Dual statements hold for F.

Proposition 14. *Let* $\mathrm{M} \subseteq \mathrm{Mor}\,$c-DESC. *A* c-DESC-*morphism* f *is* M-*initial if and only if the condition* $\mathrm{sig}(e) \in \mathrm{Iso}\,\mathrm{SIG}$ *implies* M-Difip(e, f) *for every* c-DESC-*morphism* e.

Proof. Assume that $f \in$ M-Init and $\mathrm{sig}(e) \in \mathrm{Iso}\,\mathrm{SIG}$ (so $e \in \mathrm{Epi}\,$c-DESC), $f \circ u = m \circ e$ for some $u \in \mathrm{Mor}\,$c-DESC and $m \in \mathrm{M}$. We have $\mathrm{sig}(f) \circ (\mathrm{sig}(u) \circ \mathrm{sig}(e)^{-1}) = \mathrm{sig}(m)$. Hence, by Definition 18, there exists a c-DESC-morphism $d\colon \operatorname{codom} e \to \operatorname{dom} f$ such that $\mathrm{sig}(d) = \mathrm{sig}(u) \circ \mathrm{sig}(e)^{-1}$. Since the functor sig is faithful, $d \circ e = u$, so d is a desired diagonal.

Conversely, assume that M-Difip$(\mathrm{sig}^{-1}(\mathrm{Iso}\,\mathrm{SIG}), f)$ holds for some c-DESC-morphism $f\colon T \to S$. Choose arbitrary morphisms $m\colon X \to S \in \mathrm{M}$ and $k\colon \mathrm{sig}(X) \to \mathrm{sig}(T)$ so that $\mathrm{sig}(f) \circ k = \mathrm{sig}(m)$. Denote $f^* = \mathrm{sig}^*\big(\mathrm{sig}(f)\big)$ for an arbitrary c-DESC-morphism f. We have $f^* \circ \mathrm{sig}^*(k) = m^*$. Hence

$$f \circ \big(\varepsilon_T \circ \mathrm{sig}^*(k)\big) = \varepsilon_S \circ f^* \circ \mathrm{sig}^*(k) = \varepsilon_S \circ m^* = m \circ \varepsilon_X.$$

Since $\mathrm{sig}(\varepsilon_X) = 1_{\mathrm{sig}(X)} \in \mathrm{Iso}\,\mathrm{SIG}$, there exists a diagonal $d\colon X \to T$ such that $d \circ \varepsilon_X = \varepsilon_T \circ \mathrm{sig}^*(k)$, so $\mathrm{sig}(d) = k$. Consequently, the c-DESC-morphism d can be employed as k^+ from Definition 18, so $f \in$ M-Init.

$$\begin{tikzcd}
X^* \arrow[r, "\mathrm{sig}^*(k)"] \arrow[dr, "m^*"'] \arrow[dd, "\varepsilon_X"'] & T^* \arrow[r, "\varepsilon_T"] \arrow[d, "f^*"] & T \arrow[dd, "f"] \\
 & S^* \arrow[dr, "\varepsilon_S"'] & \\
X \arrow[rr, "m"] \arrow[uurr, dotted, "d"] & & S
\end{tikzcd}$$

□

Proposition 15. *Let* M ⊆ Mor c-DESC*. Every* c-DESC*-morphism with a discrete codomain is* M*-initial if and only if every* M*-morphism with a discrete codomain has a discrete domain.*

Proof. Assume that every c-DESC-morphism with a discrete codomain is M-initial. If $m\colon X \to S \in \mathrm{M}$ and $\varepsilon_S \in \mathrm{Iso}\,\text{c-DESC}$, then $\varepsilon_S \circ m^*\colon X^* \to S \in \text{M-Init}$. So, by Proposition 14, a commutative square $(\varepsilon_S \circ m^*) \circ 1_{X^*} = m \circ \varepsilon_X$ has a diagonal d such that $d \circ \varepsilon_X = 1_{X^*}$. Hence ε_X is an isomorphism since it is a left-invertible epimorphism [1, Proposition 7.43].

Conversely, assume that every M-morphism with a discrete codomain has a discrete domain. Choose arbitrary morphisms $f\colon T \to S$, $m\colon X \to S \in \mathrm{M}$, $k\colon \mathrm{sig}(X) \to \mathrm{sig}(T)$ so that $\mathrm{sig}(f) \circ k = \mathrm{sig}(m)$ and $\varepsilon_S \in \mathrm{Iso}\,\text{c-DESC}$ (hence $\varepsilon_X \in \mathrm{Iso}\,\text{c-DESC}$ as well). We have $\mathrm{sig}(\varepsilon_T \circ \mathrm{sig}^*(k) \circ \varepsilon_X^{-1}) = k$, so $f \in \text{M-Init}$. □

Definition 19. A formal specification technology $SC = \langle \text{c-DESC}, \mathrm{Conf}, \mathrm{sig}\rangle$ is said to *support* L*-labelings* for some class L of SIG-morphisms if the following conditions hold:

(i) every L-morphism is a retraction;
(ii) there exists a class M ⊆ Mor c-DESC such that the pair $(\mathrm{sig}^{-1}(\mathrm{L}), \mathrm{M})$ is a factorization system in c-DESC and every M-morphism with a discrete codomain has a discrete domain;
(iii) the class Conf is closed under pushes by $\mathrm{sig}^{-1}(\mathrm{L})$-morphisms.

In this case, the quadruple $\mathrm{SC_L} = \langle \text{c-DESC}, \mathrm{Conf}, \mathrm{sig}, (\mathrm{Trl}\,\mathrm{L})^{\mathrm{op}}\rangle$ where $\mathrm{Trl}\,\mathrm{L} = (\mathrm{Ob}\,\text{c-DESC}, \mathrm{sig}^{-1}(\mathrm{L}) \cap \text{M-Init})$ is called a design technology over SC *induced by a class of labelings* L (an L-technology over SC), M-morphisms are called $\mathrm{SC_L}$*-inclusions*.

Proposition 16. *Every* L*-technology* $\mathrm{SC_L}$ *is a counified formal design technology that supports tracing with* L *as the class of all labelings and*

$$\begin{aligned}\mathrm{RegMono}\,\text{c-DESC} \subseteq \mathrm{Mono}\,\text{c-DESC} \cap \text{Mor-Init} &= \mathrm{Mono}\,\text{c-DESC} \cap \mathrm{M} \\ &= \mathrm{Mono}\,\text{c-DESC} \cap \text{M-Init} \subseteq \mathrm{M} \subseteq \text{Mor-Init} \subseteq \text{M-Init}.\end{aligned}$$

Proof. First of all, we verify that the class M-Init is closed under composition. Choose arbitrary M-initial morphisms $f\colon T \to R$, $g\colon R \to S$, M-morphism $m\colon X \to S$, and SIG-morphism $k\colon \mathrm{sig}(X) \to \mathrm{sig}(T)$ such that $\mathrm{sig}(g \circ f) \circ k = \mathrm{sig}(m)$. Find a c-DESC-morphism $k^+\colon X \to T$ such that $\mathrm{sig}(k^+) = k$. Since $g \in \text{M-Init}$ and $\mathrm{sig}(g) \circ (\mathrm{sig}(f) \circ k) = \mathrm{sig}(m)$, by Definition 18, there exists a c-DESC-morphism $n\colon X \to R$ such that $\mathrm{sig}(n) = \mathrm{sig}(f) \circ k$. Consequently, $\mathrm{sig}(g \circ n) = \mathrm{sig}(m)$. Hence $g \circ n = m$ since the functor sig is faithful. We will verify that $n \in \mathrm{M}$, so that the desired c-DESC-morphism k^+ exists by Definition 18 since $f \in \text{M-Init}$. Let $n = n' \circ e$ and $(g \circ n') = n'' \circ e'$ be $(\mathrm{sig}^{-1}(\mathrm{L}), \mathrm{M})$-factorizations. So $m = g \circ n = n'' \circ (e' \circ e)$ is a $(\mathrm{sig}^{-1}(\mathrm{L}), \mathrm{M})$-factorization of an M-morphism. Hence $e' \circ e \in \mathrm{Iso}\,\text{c-DESC}$. Therefore e is left-invertible. Every left-invertible epimorphism is an isomorphism [1, Proposition 7.43]; consequently, $n \in \mathrm{M} \circ \mathrm{Iso}\,\text{c-DESC} = \mathrm{M}$ as desired.

$$\begin{tikzcd} & T \arrow[d, "f"] \\ & R \arrow[d, "g"] \\ X \arrow[uur, dashed, "k^+"] \arrow[ur, "n"] \arrow[r, "m"] & S \end{tikzcd}$$

Condition (ii) of Definition 19 guarantees that $\mathrm{Iso}\,\text{c-DESC} \subseteq \mathrm{sig}^{-1}(\mathrm{L}) \cap \mathrm{M}$. It follows that, in particular, $\mathrm{Iso}\,\mathrm{SIG} = \mathrm{sig}\big(\mathrm{sig}^*(\mathrm{Iso}\,\mathrm{SIG})\big) \subseteq \mathrm{sig}(\mathrm{Iso}\,\text{c-DESC}) \subseteq \mathrm{L}$; consequently, $\mathrm{sig}^{-1}(\mathrm{Iso}\,\mathrm{SIG}) \subseteq \mathrm{sig}^{-1}(\mathrm{L})$. So, by the same condition (ii) of Definition 19, $\mathrm{Difip}(\mathrm{sig}^{-1}(\mathrm{Iso}\,\mathrm{SIG}), \mathrm{M})$ holds. Hence $\mathrm{M} \subseteq \text{Mor-Init}$ by Proposition 14. Since, by the very same condition (ii) of Definition 19, the class $\mathrm{sig}^{-1}(\mathrm{L})$ is closed under composition, $\mathrm{Trl}\,\mathrm{L}$ is indeed a subcategory of c-DESC that contains all isomorphisms. (A stronger statement is actually proved: all c-DESC-objects and all M-initial morphisms comprise a subcategory of c-DESC that has a factorization system $(\mathrm{Mor}\,\mathrm{Trl}\,\mathrm{L}, \mathrm{M})$ [24, Proposition 4].) Along with condition (iii) of Definition 19 this implies that SC_L is indeed a counified formal design technology.

Further, by condition (i) of Definition 19, for each $(\mathrm{Trl}\,\mathrm{L})$-morphism $t\colon T \to S$ there exists a SIG-morphism s such that $\mathrm{sig}(t) \circ s = 1_{\mathrm{sig}(S)}$. Since $1_S \in \mathrm{M}$, by Definition 18, there exists a c-DESC-morphism t^+ such that $t \circ t^+ = 1_S$. Hence SC_L supports tracing.

The final part of condition (ii) of Definition 19 and Proposition 15 imply that $\mathrm{sig}^*(\mathrm{L}) \subseteq \mathrm{sig}^{-1}(\mathrm{L}) \cap \text{M-Init} = \mathrm{Mor}\,\mathrm{Trl}\,\mathrm{L}$. Hence $\mathrm{sig}(\mathrm{Mor}\,\mathrm{Trl}\,\mathrm{L}) \supseteq \mathrm{sig}\big(\mathrm{sig}^*(\mathrm{L})\big) = \mathrm{L}$.

It remains to verify the chain of inclusions of classes of morphisms specified in the condition of the proposition. Since the functor sig due to having a left adjoint preserves regular monomorphisms [29, Sec. V.5], every regular c-DESC-monomorphism is initial [1, Proposition 8.7(3)]. Further, let $m\colon P \to Q$ be an arbitrary M-initial c-DESC-monomorphism, and let $m = m' \circ e$ be its $(\mathrm{sig}^{-1}(\mathrm{L}), \mathrm{M})$-factorization. Hence e is a monomorphism, so $\mathrm{sig}(e) \in \mathrm{Iso}\,\mathrm{SIG}$ since the functor sig due to having a left adjoint preserves monomorphisms [29, Sec. V.5]. By Proposition 14, the commutative square $m \circ 1_P = m' \circ e$ has a diagonal d such that $d \circ e = 1_P$. Consequently, $e \in \mathrm{Iso}\,\text{c-DESC}$. Hence $m \in \mathrm{M}$. In summary, $\mathrm{Mono}\,\text{c-DESC} \cap \text{M-Init} \subseteq \mathrm{M}$. Finally, the condition $\mathrm{M} \subseteq \text{Mor-Init}$ is verified above in the second paragraph of the proof. □

Proposition 17. *In an arbitrary* L*-technology* SC_L*, every* $\mathrm{sig}^{-1}(\mathrm{L})$*-morphism is a trace of a refinement if and only if the functor* sig *is an equivalence of categories* c-DESC *and* SIG*.*

Proof. Consider the following chain of equivalent statements:

$$\begin{aligned}
&\text{sig is an equivalence}\\
&\quad\overset{\text{by the definition of equivalence of categories}}{\iff} \varepsilon \subseteq \mathrm{Iso}\,\text{c-DESC}\\
&\quad\overset{f\circ\varepsilon_A=\varepsilon_B\circ f^*\text{ for each } f\colon A\to B}{\iff} \mathrm{sig}^{-1}(\mathrm{Iso}\,\mathrm{SIG}) = \mathrm{Iso}\,\text{c-DESC}\\
&\quad\overset{\text{consider that } \mathrm{L}\supseteq\mathrm{Iso}\,\mathrm{SIG} \text{ and } \mathrm{M}\supseteq\mathrm{Iso}\,\text{c-DESC}}{\iff} \text{M-Difip}\big(\mathrm{sig}^{-1}(\mathrm{Iso}\,\mathrm{SIG}), \mathrm{sig}^{-1}(\mathrm{L})\big)\\
&\quad\overset{\text{by Proposition 14}}{\iff} \mathrm{sig}^{-1}(\mathrm{L}) \subseteq \text{M-Init}\\
&\quad\overset{\text{by Definition 19}}{\iff} \mathrm{Mor}\,\mathrm{Trl}\,\mathrm{L} = \mathrm{sig}^{-1}(\mathrm{L}).
\end{aligned}$$

□

Proposition 18. *In an arbitrary* L*-technology* SC_L *every refinement is a* c-DESC*-isomorphism if and only if* $\mathrm{L} = \mathrm{Iso}\,\mathrm{SIG}$*.*

Proof. The necessity follows from Propositions 16 and 3. To prove the sufficiency, note that, by condition (ii) of Definition 19 and Proposition 14, the condition $\mathrm{L} = \mathrm{Iso}\,\mathrm{SIG}$ implies $\mathrm{M} = \text{Mor-Init}$. In turn, every initial c-DESC-morphism that is sent to an isomorphism by the functor sig is an isomorphism itself [1, Proposition 8.14]. ☐

Theorem 4. *Let* $\mathrm{SC}' = \langle \text{c-DESC}', \mathrm{Conf}', \mathrm{sig}' \colon \text{c-DESC}' \to \mathrm{SIG}' \rangle$ *be an arbitrary formal specification technology,* $\langle \mathrm{cm}, \mathrm{sm} \rangle \colon \mathrm{SC} \to \mathrm{SC}'$ *be a* **conf**-*trace such that the functor* sm *is an isomorphism of categories. If* SC *supports* L*-labelings, then* SC′ *supports* sm(L)*-labelings and the class of all* $\mathrm{SC}'_{\mathrm{sm(L)}}$*-inclusions looks like* $\mathrm{cm(M)} \circ \mathrm{Iso}\,\text{c-DESC}'$ *where* M *is the class of all* $\mathrm{SC_L}$*-inclusions.*

Proof. Let $\dot{\mathrm{L}} = \mathrm{sig}'^{-1}\big(\mathrm{sm(L)}\big)$, $\dot{\mathrm{M}} = \mathrm{cm(M)} \circ \mathrm{Iso}\,\text{c-DESC}'$, ε' be the counit of the adjunction $\mathrm{sig}'^{*} \dashv \mathrm{sig}'$, and ζ be the counit of the adjunction $\mathrm{cm}^{*} \dashv \mathrm{cm}$. We will employ the following properties of the **SPEC**-morphism $\langle \mathrm{cm}, \mathrm{sm} \rangle$:

(a) $\mathrm{sm} \circ \mathrm{sig} = \mathrm{sig}' \circ \mathrm{cm}$ (by condition (iii) of Definition 3);
(b) $\mathrm{cm}\big(\mathrm{sig}^{-1}(\mathrm{L})\big) \subseteq \dot{\mathrm{L}}$ (by (a) $\mathrm{sig}'\big(\mathrm{cm}(q)\big) = \mathrm{sm}\big(\mathrm{sig}(q)\big) \in \mathrm{sm(L)}$ for each $q \in \mathrm{sig}^{-1}(\mathrm{L})$);
(c) $\mathrm{sig} = \mathrm{sm}^{-1} \circ \mathrm{sig}' \circ \mathrm{cm}$ (by (a));
(d) $\mathrm{cm}^{*}(\dot{\mathrm{L}}) \subseteq \mathrm{sig}^{-1}(\mathrm{L})$ (by (c)

$$\mathrm{sig}\big(\mathrm{cm}^{*}(h)\big) = \mathrm{sm}^{-1}\Big(\mathrm{sig}'\Big(\mathrm{cm}\big(\mathrm{cm}^{*}(h)\big)\Big)\Big) = \mathrm{sm}^{-1}\big(\mathrm{sig}'(h)\big) \in \mathrm{L}$$

for each $h \in \dot{\mathrm{L}}$ having that $\mathrm{cm} \circ \mathrm{cm}^{*} = 1_{\text{c-DESC}'}$).

Verify all conditions of Definition 19 for the technology SC′ and class of SIG′-morphisms sm(L).

(i). Every functor preserves retractions. Hence sm(L) consists of retractions.

(ii). Let

$$\dot{\mathrm{M}}' = \{\mathrm{cm}(m) \mid m \in \mathrm{M},\ \zeta_{\mathrm{codom}\,m} = 1_{\mathrm{codom}\,m}\} \circ \mathrm{Iso}\,\text{c-DESC}'$$

and establish the existence of a factorization system $(\dot{\mathrm{L}}, \dot{\mathrm{M}}')$ in c-DESC′. At first, verify that $j \circ g \in \dot{\mathrm{L}}$ for each $j \in \mathrm{Iso}\,\text{c-DESC}'$ and $g \in \dot{\mathrm{L}}$. By statement (d) and condition (ii) of Definition 19,

$$\mathrm{cm}^{*}(j \circ g) \in \mathrm{Iso}\,\text{c-DESC} \circ \mathrm{cm}^{*}(\dot{\mathrm{L}}) \subseteq \mathrm{Iso}\,\text{c-DESC} \circ \mathrm{sig}^{-1}(\mathrm{L}) \subseteq \mathrm{sig}^{-1}(\mathrm{L}).$$

Hence

$$j \circ g = \mathrm{cm}\big(\mathrm{cm}^{*}(j \circ g)\big) \in \mathrm{cm}\big(\mathrm{sig}^{-1}(\mathrm{L})\big) \subseteq \dot{\mathrm{L}}$$

taking statement (b) into account. Further, by condition (ii) of Definition 19, for each c-DESC′-morphism f there exists a $(\mathrm{sig}^{-1}(\mathrm{L}), \mathrm{M})$-factorization $\mathrm{cm}^{*}(f) = n \circ e$ that is turned by the functor cm into an equality $f = \mathrm{cm}(n) \circ \mathrm{cm}(e)$, which, by statement (b), is an $(\dot{\mathrm{L}}, \dot{\mathrm{M}}')$-factorization. Verify $\mathrm{Difip}(\dot{\mathrm{L}}, \dot{\mathrm{M}}')$. If $(\mathrm{cm}(m) \circ i) \circ u = v \circ l$ for some $m \colon A \to B \in \mathrm{M}$, $i \in \mathrm{Iso}\,\text{c-DESC}'$, $l \in \dot{\mathrm{L}}$, $u, v \in \mathrm{Mor}\,\text{c-DESC}'$, then denoting $w = \mathrm{cm}^{*}(i \circ u)$ we have

$$m \circ (\zeta_A \circ w) = \zeta_B \circ \mathrm{cm}^{*}\big(\mathrm{cm}(m)\big) \circ w = \big(\zeta_B \circ \mathrm{cm}^{*}(v)\big) \circ \mathrm{cm}^{*}(l).$$

Hence by $\mathrm{Difip}(\mathrm{sig}^{-1}(\mathrm{L}), \mathrm{M})$ (condition (ii) of Definition 19) and statement (d) there exists a c-DESC-morphism d such that $d \circ \mathrm{cm}^{*}(l) = \zeta_A \circ w$. Applying the functor cm to this equality, we obtain that $\mathrm{cm}(d) \circ l = i \circ u$, so $i^{-1} \circ \mathrm{cm}(d)$ is the desired diagonal.

$$\begin{tikzcd}
{} \arrow[r, "\mathrm{cm}^{*}(u)"] \arrow[d, "\mathrm{cm}^{*}(l)"'] & {} \arrow[r, "\mathrm{cm}^{*}(i)"] & \mathrm{cm}^{*}\big(\mathrm{cm}(A)\big) \arrow[r, "\zeta_A"] \arrow[d, "\mathrm{cm}^{*}\big(\mathrm{cm}(m)\big)"] & A \arrow[d, "m"] \\
{} \arrow[rr, "\mathrm{cm}^{*}(v)"'] \arrow[urrr, dotted, "d"] & & \mathrm{cm}^{*}\big(\mathrm{cm}(B)\big) \arrow[r, "\zeta_B"'] & B
\end{tikzcd}$$

Verify that for each $\dot{\mathrm{M}}'$-morphism $p \colon X \to Y$ with $\varepsilon'_Y \in \mathrm{Iso}\,\text{c-DESC}'$ the condition $\varepsilon'_X \in \mathrm{Iso}\,\text{c-DESC}'$ holds. Let $p = \mathrm{cm}(r) \circ t$ for some $r \colon P \to Q \in \mathrm{M}$ and $t \colon X \to \mathrm{cm}(P) \in \mathrm{Iso}\,\text{c-DESC}'$ such that $\zeta_Q = 1_Q$ and $\mathrm{cm}(Q) = Y$. Statement (c) and rules of constructing composition of adjunctions of functors [29, Sec. IV.8] imply that $\varepsilon_Z = \zeta_Z \circ \mathrm{cm}^{*}\big(\varepsilon'_{\mathrm{cm}(Z)}\big)$ for each c-DESC-object Z. In particular, by assumption, $\varepsilon_Q = \mathrm{cm}^{*}(\varepsilon'_Y) \in \mathrm{Iso}\,\text{c-DESC}$. Hence, by condition (ii) of Definition 19, $\varepsilon_P \in \mathrm{Iso}\,\text{c-DESC}$.

Consequently, substituting P for Z, we obtain that c-DESC-morphism $\mathrm{cm}^*(\varepsilon'_{\mathrm{cm}(P)})$ is left-invertible. So it is an isomorphism since it is an epimorphism (the counit ε' consists of epimorphisms and the functor cm^* having right adjoint preserves all colimits [29, Sec. V.5], in particular, all epimorphisms). We conclude that $\varepsilon'_{\mathrm{cm}(P)} = \mathrm{cm}\big(\mathrm{cm}^*(\varepsilon'_{\mathrm{cm}(P)})\big) \in \mathrm{Iso}\,\text{c-DESC}'$. Consequently, $\varepsilon'_X = t^{-1} \circ \varepsilon'_{\mathrm{cm}(P)} \circ \mathrm{cm}^*\big(\mathrm{cm}(t)\big) \in \mathrm{Iso}\,\text{c-DESC}'$.

It remains to verify that $\dot{\mathrm{M}} = \dot{\mathrm{M}}'$. On the one hand, by construction, $\dot{\mathrm{M}}' \subseteq \dot{\mathrm{M}}$. On the other hand, $\mathrm{Difip}(\dot{\mathrm{L}}, \dot{\mathrm{M}})$ holds (this condition is verified by literally the same reasoning as $\mathrm{Difip}(\dot{\mathrm{L}}, \dot{\mathrm{M}}')$ above), so $\dot{\mathrm{M}} \subseteq \dot{\mathrm{M}}'$.

(iii). Let $\tau \rightrightarrows \Sigma$ be a push of an arbitrary diagram $\Sigma \in \mathrm{Conf}'$ by an arbitrary family of $\dot{\mathrm{L}}$-morphisms τ. Consider the diagram $\Delta = \mathrm{cm}^* \circ (\tau \rightrightarrows \Sigma) = \mathrm{cm}^*(\tau) \rightrightarrows \mathrm{cm}^* \circ \Sigma$. By statement (d), family $\mathrm{cm}^*(\tau)$ consists of $\mathrm{sig}^{-1}(\mathrm{L})$-morphisms, so the definition of **conf**-trace and the condition $\mathrm{cm} \circ \mathrm{cm}^* \circ \Sigma = \Sigma$ imply that $\mathrm{cm}^* \circ \Sigma \in \mathrm{Conf}$. It follows that $\Delta \in \mathrm{Conf}$ by condition (iii) of Definition 19. So $\tau \rightrightarrows \Sigma = \mathrm{cm} \circ \Delta \in \mathrm{Conf}'$ by condition (i) of Definition 3. □

Corollary 4.1. *If* SC *supports* L*-labelings, then triple* $\mathrm{SSIG} = \langle \mathrm{SIG}, \mathrm{sig} \circ \mathrm{Conf}, 1_{\mathrm{SIG}} \rangle$ *is a formal specification technology that supports* L*-labelings in a way that a class* $\mathrm{Mor\,Trl\,L}$ *coincides with* L *and the class of all inclusions coincides with* $\mathrm{sig}(\mathrm{M})$ *where* M *is the class of all* $\mathrm{SC_L}$*-inclusions. The functor* sig *induces a refinement of the* L*-technology* $\mathrm{SSIG_L}$ *to* $\mathrm{SC_L}$*.*

Proof. By Corollary 3.1 and Proposition 7, we have $\mathrm{SSIG} = \mathbf{conf}^*\big(\mathbf{conf}^{**}(SC)\big) \in \mathrm{Ob}\,\mathbf{SPEC}$. Theorems 3 and 2 imply that the component of the unit of the adjunction $\mathbf{conf}^{**} \dashv \mathbf{conf}^*$ sends the formal technology SC to a trace $\langle \mathrm{sig}, 1_{\mathrm{SIG}} \rangle\colon \mathrm{SC} \to \mathrm{SSIG}$. Now apply Theorem 4. In particular, it was verified in its proof that every $\mathrm{SSIG_L}$-inclusion looks like $\mathrm{sig}(m) \circ i$ for some $m\colon P \to Q \in \mathrm{M}$ and $i \in \mathrm{Iso\,SIG}$, at that $\varepsilon_Q = 1_Q$. Hence, by condition (ii) of Definition 19, $\varepsilon_P \in \mathrm{Iso}\,\text{c-DESC}$ and we have $\mathrm{sig}(m) \circ i = \mathrm{sig}\big(m \circ \varepsilon_P \circ \mathrm{sig}^*(i)\big) \in \mathrm{sig}(\mathrm{M})$ having that class M is closed under composition and contains all c-DESC-isomorphisms.

Finally, since $\mathrm{SSIG_L} = \langle \mathrm{SIG}, \mathrm{sig} \circ \mathrm{Conf}, 1_{\mathrm{SIG}}, (\mathrm{Ob\,SIG}, \mathrm{L})^{\mathrm{op}} \rangle$, there exists an **ARCH**-morphism $\langle \mathrm{sig}, 1_{\mathrm{SIG}}, \mathrm{sig}(-^{\mathrm{op}})^{\mathrm{op}} \rangle\colon \mathrm{SC_L} \to \mathrm{SSIG_L}$ which is a trace of a refinement of design technologies (condition (ii) of Definition 17 is provided by the choice of the functor sig^* for drm having that Proposition 15 implies that $\mathrm{sig}^*(\mathrm{L}) \subseteq \mathrm{Mor\,Trl\,L}$). □

Corollary 4.2. *If* SC *supports* L*-labelings, then all* SIG*-objects and all* L*-morphisms comprise a subcategory of* SIG *that contains all* SIG*-isomorphisms.*

Proof. By Corollary 4.1. □

Corollary 4.3. *Let* $\mathrm{SC}' = \langle \text{c-DESC}', \mathrm{Conf}', \mathrm{sig}'\colon \text{c-DESC}' \to \mathrm{SIG}' \rangle$ *be an arbitrary formal specification technology,* $\langle \mathrm{cm}, \mathrm{sm} \rangle\colon \mathrm{SC} \to \mathrm{SC}'$ *be a* **SPEC***-morphism such that* cm *is a* **desc***-labeling and* sm *is an isomorphism of categories. If* SC *supports* L*-labelings and* SC′ *supports* L′*-labelings, then the condition* $\mathrm{sm}(\mathrm{L}) \subseteq \mathrm{L}'$ *holds if and only if the triple* $\langle \mathrm{cm}, \mathrm{sm}, \mathrm{cm}(-^{\mathrm{op}})^{\mathrm{op}} \rangle$ *is an* **ARCH***-morphism from* $\mathrm{SC_L}$ *to* $\mathrm{SC}'_{\mathrm{L}'}$*.*

Proof. Denote by M′ the class of all $\mathrm{SC}'_{\mathrm{L}'}$-inclusions. We will employ the counit ζ, the factorization system $(\dot{\mathrm{L}}, \dot{\mathrm{M}})$ in c-DESC′, and the statements (a)–(d) that are introduced in the same way as in the proof of Theorem 4.

First, assume that $sm(\mathrm{L}) \subseteq \mathrm{L}'$ so $\dot{\mathrm{L}} \subseteq \mathrm{sig}'^{-1}(\mathrm{L}')$ and verify that $\mathrm{cm}(f) \in \mathrm{Mor\,Trl\,L}'$ for an arbitrary $(\mathrm{Trl\,L})$-morphism $f\colon A \to B$. By statement (b) and assumption, $\mathrm{cm}(f) \in \dot{\mathrm{L}} \subseteq \mathrm{sig}'^{-1}(\mathrm{L}')$. Now employ Proposition 14 by verifying $\mathrm{M}'\text{-}\mathrm{Difip}\big(e, \mathrm{cm}(f)\big)$ for an arbitrary $e \in \mathrm{sig}'^{-1}(\mathrm{Iso\,SIG}')$. Let $\mathrm{cm}(f) \circ u = m' \circ e$ be an arbitrary commutative square with $m' \in \mathrm{M}'$. By assumption, $\mathrm{Difip}(\dot{\mathrm{L}}, m')$ holds, so $m' \in \dot{\mathrm{M}}$, i.e., $m' = \mathrm{cm}(m) \circ i$ for some $m \in \mathrm{M}$ and $i \in \mathrm{Iso}\,\text{c-DESC}'$. Applying functor cm^* to the square, we obtain the equality $\mathrm{cm}^*\big(\mathrm{cm}(f)\big) \circ \mathrm{cm}^*(u) = \mathrm{cm}^*\big(\mathrm{cm}(m)\big) \circ \mathrm{cm}^*(i) \circ \mathrm{cm}^*(e)$. Composing this equality with ζ_B from the left and applying the counit equality $f \circ \zeta_A = \zeta_B \circ \mathrm{cm}^*\big(\mathrm{cm}(f)\big)$, we obtain the equality $f \circ \zeta_A \circ \mathrm{cm}^*(u) = w \circ \mathrm{cm}^*(i) \circ \mathrm{cm}^*(e)$, where $w = \zeta_B \circ \mathrm{cm}^*\big(\mathrm{cm}(m)\big)$. Let $w = n \circ s$ be a $(\mathrm{sig}^{-1}(\mathrm{L}), \mathrm{M})$-factorization. By Corollary 4.1, the equality $\mathrm{sig}(w) = \mathrm{sig}(n) \circ \mathrm{sig}(s)$ is an $\big(\mathrm{L}, \mathrm{sig}(\mathrm{M})\big)$-factorization. So, by statement (c), we have $\mathrm{sig}(w) = \mathrm{sm}^{-1}\Big(\mathrm{sig}'\big(\mathrm{cm}(\zeta_B \circ m)\big)\Big) = \mathrm{sig}(m) \in \mathrm{sig}(\mathrm{M})$ and obtain $\mathrm{sig}(s) \in \mathrm{Iso}\,\mathrm{SIG}$. Finally, we have the commutative square $f \circ h = n \circ g$ in c-DESC, where $h = \zeta_A \circ \mathrm{cm}^*(u)$ and $g = s \circ \mathrm{cm}^*(i) \circ \mathrm{cm}^*(e)$. So, by statement (c), $\mathrm{sig}(g) = \mathrm{sig}(s) \circ \mathrm{sig}\big(\mathrm{cm}^*(i)\big) \circ \mathrm{sm}^{-1}\big(\mathrm{sig}'(e)\big) \in \mathrm{Iso}\,\mathrm{SIG}$. So, since $f \in \mathrm{M}$-Init and $n \in \mathrm{M}$, by Proposition 14, there exists a diagonal d such that $d \circ g = h$. Applying the functor cm to this equality, we obtain $d' \circ e = u$ where $d' = \mathrm{cm}(d) \circ \mathrm{cm}(s) \circ i$. Hence d' is the desired diagonal of the original square. Consequently, $\mathrm{cm}(f) \in \mathrm{sig}'^{-1}(\mathrm{L}') \cap \mathrm{M}'$-Init $= \mathrm{Mor}\,\mathrm{Trl}\,\mathrm{L}'$.

$$\begin{tikzcd}
{} \arrow[r, "\mathrm{cm}^*(u)"] \arrow[d, "\mathrm{cm}^*(e)"'] & \mathrm{cm}^*\big(\mathrm{cm}(A)\big) \arrow[rr, "\zeta_A"] \arrow[d, "\mathrm{cm}^*(\mathrm{cm}(f))"] & & A \arrow[ddd, "f"] \\
{} \arrow[r, "\mathrm{cm}^*(m')"] \arrow[d, "\mathrm{cm}^*(i)"'] & \mathrm{cm}^*\big(\mathrm{cm}(B)\big) \arrow[rrdd, "\zeta_B"] & & \\
{} \arrow[ru, "\mathrm{cm}^*(\mathrm{cm}(m))"'] \arrow[rrrd, "w"] \arrow[d, "s"'] & & & \\
{} \arrow[rrruuu, dotted, "d"] \arrow[rrr, "n"'] & & & B
\end{tikzcd}$$

Now assume that $\mathrm{cm}(\mathrm{Mor}\,\mathrm{Trl}\,\mathrm{L}) \subseteq \mathrm{Mor}\,\mathrm{Trl}\,\mathrm{L}'$ and verify that $\mathrm{sm}(l) \in \mathrm{L}'$ for an arbitrary L-morphism l. The final part of condition (ii) of Definition 19 and Proposition 15 imply that $\mathrm{sig}^*(l) \in \mathrm{Mor}\,\mathrm{Trl}\,\mathrm{L}$, so along with statement (a) we have $\mathrm{sm}(l) = \mathrm{sm}\Big(\mathrm{sig}\big(\mathrm{sig}^*(l)\big)\Big) = \mathrm{sig}'\Big(\mathrm{cm}\big(\mathrm{sig}^*(l)\big)\Big) \in \mathrm{L}'$. ☐

Corollary 4.4. *The specification technology* $\mathbf{spec}\big(\mathrm{AO}_{\mathrm{int}}(\mathrm{SC}_\mathrm{L})\big)$ *supports* L*-labeling, and the* L*-technology* $\mathbf{spec}\big(\mathrm{AO}_{\mathrm{int}}(\mathrm{SC}_\mathrm{L})\big)_\mathrm{L}$ *is a subtechnology of* $\mathrm{AO}_{\mathrm{int}}(\mathrm{SC}_\mathrm{L})$*. A morphism of AO-models over* SC_L *is an inclusion if and only if functors* mod *and* str *send it to inclusions (i.e., to an* M*-morphism and to a* $\mathrm{sig}(\mathrm{M})$*-morphism, respectively).*

Proof. Construct categories AO, LAB, and tr-AO and functors il, mod, int, asp, and str from constituents of the technology SC_L as described in Sec. 2. Let $\mathrm{M}_{\mathrm{AO}} = \{\langle a, x\rangle \mid a \in \mathrm{M},\ x \in \mathrm{sig}(\mathrm{M})\} \subseteq \mathrm{Mor}\,\mathrm{AO}$ and verify that a pair $(\mathrm{int}^{-1}(\mathrm{L}), \mathrm{M}_{AO})$ is a factorization system in category AO. Let $\langle f, b\rangle \colon \langle A, l\rangle \to \langle B, h\rangle$ be an arbitrary AO-morphism, let $f = m \circ e$ be a $(\mathrm{sig}^{-1}(\mathrm{L}), \mathrm{M})$-factorization, let $b = p \circ q$ and $\big(h \circ \mathrm{sig}(m)\big) = n \circ s$ be $\big(\mathrm{L}, \mathrm{sig}(\mathrm{M})\big)$-factorizations. Since $n \circ \big(s \circ \mathrm{sig}(e)\big) = h \circ \mathrm{sig}(m \circ e) = h \circ \mathrm{sig}(f) = b \circ l = p \circ (q \circ l)$ is an $\big(\mathrm{L}, \mathrm{sig}(\mathrm{M})\big)$-factorization, there exists a SIG-isomorphism i such that $p = n \circ i$ and $s \circ \mathrm{sig}(e) = i \circ q \circ l$. Hence $\langle f, b\rangle = \langle m, n\rangle \circ \langle e, i \circ q\rangle$ is an $(\mathrm{int}^{-1}(\mathrm{L}), \mathrm{M}_{AO})$-factorization with intermediate object $\langle \mathrm{dom}\, m, s\rangle$. A diagonal of an arbitrary commutative square $r \circ u = v \circ w$ in AO with $r \in \mathrm{M}_{\mathrm{AO}}$ and $\mathrm{int}(w) \in \mathrm{L}$ is a pair $\langle d, d'\rangle$, where d is a diagonal of a mod-image of the square and d' is a diagonal of its str-image.

Assume that $\langle f, b\rangle \in \mathrm{Mor}\,\mathrm{M}_{\mathrm{AO}}$ and $\langle B, h\rangle$ is a discrete object, i.e., $\varepsilon_B \in \mathrm{Iso}\,\text{c-DESC}$ and $h \in \mathrm{Iso}\,\mathrm{SIG}$ (by the proof of Theorem 1). Since $f \in \mathrm{M}$, we have $\varepsilon_A \in \mathrm{Iso}\,\text{c-DESC}$. In addition, the equality $\big(h \circ \mathrm{sig}(f)\big) = b \circ l$ determines an $\big(\mathrm{L}, \mathrm{sig}(\mathrm{M})\big)$-factorization of the $\mathrm{sig}(\mathrm{M})$-morphism $h \circ \mathrm{sig}(f)$, so $l \in \mathrm{Iso}\,\mathrm{SIG}$. Thus, $\langle A, l\rangle$ is a discrete object as well.

It remains to verify that, if $\langle f, b\rangle \in \mathrm{int}^{-1}(\mathrm{L}) \cap \mathrm{M}_{\mathrm{AO}}$-Init, then $f \in \mathrm{M}$-Init, so the category $\mathrm{Trl}\,\mathrm{L}$ constructed from constituents of the technology $\mathbf{spec}\big(\mathrm{AO}_{\mathrm{int}}(\mathrm{SC}_\mathrm{L})\big)$ is contained in tr-AO as a subcategory, and functor 1_{AO} induces an embedding of the technology $\mathbf{spec}\big(\mathrm{AO}_{\mathrm{int}}(\mathrm{SC}_\mathrm{L})\big)_\mathrm{L}$ into $\mathrm{AO}_{\mathrm{int}}(\mathrm{SC}_\mathrm{L})$. Let $\mathrm{sig}(f) \circ k = \mathrm{sig}(m)$ for some SIG-morphism k and M-morphism $m \colon X \to B$. If $\big(h \circ \mathrm{sig}(m)\big) = n \circ s$ is an $\big(\mathrm{L}, \mathrm{sig}(\mathrm{M})\big)$-factorization, then there exists an M_{AO}-morphism $\langle m, n\rangle \colon \langle X, s\rangle \to \langle B, h\rangle$. Hence there exists an AO-morphism $k' \colon \langle X, s\rangle \to \langle A, l\rangle$ such that $\mathrm{int}(k') = k$, so we can choose $k^+ = \mathrm{mod}(k')$ to satisfy Definition 18 for f. ☐

Corollary 4.5. *The category* tr-AO *induced by the technology* SC_L *coincides with the comma category* $\mathrm{sig}_\mathrm{L} \downarrow (\mathrm{Ob}\,\mathrm{SIG}, \mathrm{L})$ *where functor* $\mathrm{sig}_\mathrm{L} \colon \mathrm{Trl}\,\mathrm{L} \to (\mathrm{Ob}\,\mathrm{SIG}, \mathrm{L})$ *acts as* sig*. Therefore, a refinement in an*

AO-technology over $\mathrm{SC_L}$ *has an explication if and only if both its domain and codomain belong to an aspectual core.*

Proof. For each tr-AO-morphism $\langle t, b\rangle\colon \langle A, l\rangle \to \langle B, h\rangle$, we have $b \in \mathrm{L}$ since $b \circ l = h \circ \mathrm{sig}(t) \in \mathrm{L}$, $l \in \mathrm{L}$, and, by Corollary 4.1, class L is weakly left cancellable.

Further, by Propositions 16 and 9, every explication of the aspectual structure of an AO-model in an AO-technology over $\mathrm{SC_L}$ is universal, so an explicable refinement can take place only between elements of an aspectual core. We verify that each of their refinements is explicable. Assume that AO-models $\langle A, l\rangle$ and $\langle B, h\rangle$ have explications $s\colon P \to A$ and $r\colon Q \to B$, respectively. Let $q\colon P \to Q$ be an explication of the action of the tr-AO-morphism $\langle t, b\rangle$ along s and r, i.e., $q \circ s^{\mathrm{op}} = r^{\mathrm{op}} \circ t$. Since refinement s is traceable, $\mathrm{sig}(q) = b \in \mathrm{L}$. We establish that $q \in$ M-Init by means of Proposition 14: find a diagonal of an arbitrary commutative square $q \circ u = m \circ e$ with $\mathrm{sig}(e) \in \mathrm{Iso\,SIG}$ and $m \in \mathrm{M}$. Let s' be a c-DESC-morphism that is right inverse to s^{op}. By the same Proposition 14, commutative square $(r^{\mathrm{op}} \circ t) \circ (s' \circ u) = (q \circ s^{\mathrm{op}}) \circ (s' \circ u) = m \circ e$ has a diagonal d such that $d \circ e = s' \circ u$. Hence $(s^{\mathrm{op}} \circ d) \circ e = s^{\mathrm{op}} \circ s' \circ u = u$, so $s^{\mathrm{op}} \circ d$ is the desired diagonal of the original square. Finally, we have $q \in \mathrm{Mor\,Trl\,L}$.

$$\begin{tikzcd}
{} \arrow[r, "u"] \arrow[dd, "e"'] & P \arrow[r, "s'"] \arrow[dr, "1_P"] & A \arrow[d, "s^{\mathrm{op}}"] \\
& & P \arrow[d, "q"] \\
{} \arrow[uurr, dotted, "d"] \arrow[rr, "m"] & & Q
\end{tikzcd}$$

□

Corollary 4.6. *A technology* $\mathrm{SC_L}$ *is aspectually complete if and only if every* SIG*-diagram from the class* $\mathrm{str} \circ \mathbf{D}mod^{-1}(\mathrm{Conf})$ *has a colimit.*

Proof. The necessity of the stated condition follows from Definitions 16 and 7. To prove the sufficiency in view of condition (iii) of Definition 19, it suffices to show that an arbitrary diagram $\Delta\colon X \to \mathrm{AO}$ with $\mathrm{mod} \circ \Delta \in \mathrm{Conf}$ is aspectually determined. First, we describe the construction of a colimit of the LAB-diagram $\mathrm{asp} \circ \Delta$. The functor $\mathrm{int} \circ \Delta$ is related to $\mathrm{str} \circ \Delta$ by a natural transformation β (i.e., a SIG^X-morphism) with components $\beta_I = \mathrm{asp}\big(\Delta(I)\big)\colon \mathrm{int}\big(\Delta(I)\big) \to \mathrm{str}\big(\Delta(I)\big)$, $I \in \mathrm{Ob}\,X$ that induces **D**SIG-morphism $\langle \beta, 1_X\rangle\colon \mathrm{int} \circ \Delta \to \mathrm{str} \circ \Delta$. So for every colimits $\rho\colon \mathrm{int} \circ \Delta \to \ulcorner P \urcorner$ and $\upsilon\colon \mathrm{str} \circ \Delta \to \ulcorner Q \urcorner$ there exists a colimit arrow which is a unique SIG-morphism $u\colon P \to Q$ such that $\upsilon \circ \langle \beta, 1_X\rangle = \ulcorner u \urcorner \circ \rho$. It can be verified directly that $\mathrm{Difip}\big(\{\beta_I \mid I \in \mathrm{Ob}\,X\}, \mathrm{sig(M)}\big)$, which holds by Corollary 4.1, implies $\mathrm{Difip}\big(u, \mathrm{sig(M)}\big)$ (a general dual statement can be found in [6]). Consequently, $u \in \mathrm{L}$. Hence cocone $\langle \rho, \upsilon\rangle\colon \mathrm{asp} \circ \Delta \to \ulcorner u \urcorner$ is a colimit of the LAB-diagram $\mathrm{asp} \circ \Delta$.

Choose for ρ cocone $\mathrm{sig} \circ \delta$, where $\delta\colon \mathrm{mod} \circ \Delta \to \ulcorner A \urcorner$ is a colimit of the c-DESC-diagram $\mathrm{mod} \circ \Delta$ (its existence follows from condition (i) of Definition 1). Colimit υ exists by assumption, so given the above, it can be verified directly that cocone $\langle \delta, \upsilon\rangle\colon \Delta \to \ulcorner\langle A, u\rangle\urcorner$ is a colimit of diagram Δ, i.e., the functor $\langle \mathrm{mod}, \mathrm{str}\rangle\colon \mathrm{AO} \to \text{c-DESC} \times \mathrm{SIG}$ lifts its colimits. By Proposition 1 diagram Δ is aspectually determined. □

In order to illustrate the significance of the concept of an L-technology, we show that the discrete event modeling technology is the only technology over the specification technology $\mathrm{SS} = \mathbf{spec}(\mathrm{SM})$ induced by some class of labelings. Indeed, by condition (i) of Definition 19, class of labelings L over the technology SS shall consist of surjective maps of sets and, by Corollary 4.1, it shall be the first component of a factorization system in the category **Set**. There are four factorization systems in **Set** [1, Example 14.2(4)] of which the first component consists of surjections only in the following two: (Iso, Mor) and (Epi, Mono). Choice of the class Iso **Set** of all bijections of sets as L is hindered by condition (ii) of Definition 19: the class of all bijective **Pos**-morphisms does not constitute the first component of any factorization system in **Pos** since it is not closed under formation of pushouts (a vertex of a colimit of a **Pos**-diagram $\{a < b\} \leftarrow \{a, b\} \rightarrow \{a > b\}$ comprised from bijections $a \mapsto a$, $b \mapsto b$ is a singular set). On the contrary, the class Epi **Set** of all surjective maps satisfies all conditions of Definition 19: the class of all regular **Pos**-monomorphisms can be taken for M (category **Pos** has a factorization system (Epi, RegMono) and a domain of every regular **Pos**-monomorphism with a discretely ordered codomain is discretely ordered itself) and, in addition, class of scenario configurations CPos is closed under any pushes.

Verify that the category Trl Epi **Set** over the technology SS is dual to a category of scenario refinements r-Pos. First, choose an arbitrary surjective map of partially ordered sets $f\colon T \to S$ such that the condition $f(y) < f(y')$ is equivalent to $y < y'$ for each $y, y' \in T$: all such maps comprise class Mor r-Pos$^{\mathrm{op}}$ (see Sec. 1). Clearly the map f is monotonic. Choose arbitrary maps $m\colon X \to S \in$ RegMono **Pos** and $k\colon X \to T$ such that $f\big(k(x)\big) = m(x)$ for each $x \in X$ and verify that the map k is monotonic, which implies that $f \in$ (RegMono **Pos**)-Init. Choose some $x, x' \in X$ that satisfy the condition $x < x'$. Since the map m is injective, $m(x) < m(x')$; consequently, $k(x) < k(x')$ by the choice of the map f.

Conversely, let $t\colon P \to Q$ be an arbitrary surjective RegMono-initial map of partially ordered sets. Choose some $p, p' \in P$ with $t(p) < t(p')$ and verify that $p < p'$, which implies that $t^{\mathrm{op}} \in$ Mor r-Pos. Consider maps $s\colon \mathit{2} \to P$: $0 \mapsto p$, $1 \mapsto p'$ and $n\colon \mathit{2} \to Q$: $x \mapsto t\big(s(x)\big)$, where $\mathit{2}$ denotes the two-element linearly ordered set $\{0 < 1\}$. Clearly $n \in$ RegMono **Pos**, so the map s is monotonic by the choice of the map t. Consequently, $p < p'$.

With the discrete event modeling technology as an example it can be shown that restrictions asserted on **SPEC**-morphism $\langle \mathrm{cm}, sm\rangle\colon \mathrm{SC} \to \mathrm{SC}'$ in Theorem 4 in order to provide transition of its domain's capability to support some class of labelings to its codomain cannot be considerably weakened. Start from the fact that the scenario specification technology SS does not support (Iso **Set**)-labelings. The **SPEC**-morphism $\langle 1_{\mathbf{Pos}}, |-|\rangle\colon \mathrm{TS} \to \mathrm{SS}$, where $\mathrm{TS} = \mathbf{conf}^*\big(\mathbf{conf}(\mathrm{SS})\big)$ is a **conf**-trace, but its second component is not an isomorphism and TS supports (Iso **Pos**)-labelings. In turn, the **SPEC**-morphism $\langle \mathrm{dord}, 1_{\mathbf{Set}}\rangle\colon \mathrm{US} \to \mathrm{SS}$, where $\mathrm{US} = \mathbf{conf}^*\big(\mathbf{conf}^{**}(\mathrm{SS})\big)$ is not a **conf**-trace and US supports (Iso **Set**)-labelings.

Corollary 4.3 allows revealing the functorial nature of the procedure of inducing technologies by labeling. Denote by **SPL** a category whose class of object consists of all pairs $\langle \mathrm{SC}, \mathrm{L}\rangle$, where SC is a specification technology that supports L-labelings and a morphism of a pair $\langle \mathrm{SC}, \mathrm{L}\rangle$ to $\langle \mathrm{SC}', \mathrm{L}'\rangle$ is every **SPEC**-morphism $\langle \mathrm{cm}, \mathrm{sm}\rangle\colon \mathrm{SC} \to \mathrm{SC}'$ such that cm is a **desc**-labeling, sm is an isomorphism of categories, and $\mathrm{sm}(\mathrm{L}) \subseteq \mathrm{L}'$. By Corollary 4.3 there exists an embedding $\mathbf{ispl}\colon \mathbf{SPL} \hookrightarrow \mathbf{ARCH}$: $\langle \mathrm{SC}, \mathrm{L}\rangle \mapsto \mathrm{SC}_{\mathrm{L}}$.

Employing Corollary 4.4 we show that the procedure of AO-extension is natural with respect to this embedding. We will concisely denote the specification technology $\mathbf{spec}\big(\mathrm{AO}_{int}(\mathbf{AR})\big)$ as s-AO($\mathbf{AR}$). Let **SPLL** be a subcategory of **SPL** whose class of objects is the same as Ob **SPL**, and a morphism of an object $\langle \mathrm{SC}, \mathrm{L}\rangle$ to $\langle \mathrm{SC}', \mathrm{L}'\rangle$ is every **SPL**-morphism $\langle \mathrm{cm}, \mathrm{sm}\rangle\colon \langle \mathrm{SC}, \mathrm{L}\rangle \to \langle \mathrm{SC}', \mathrm{L}'\rangle$ that satisfies the condition $\mathrm{sm}(\mathrm{L}) = \mathrm{L}'$ (Theorem 4 ensures that the class of all these morphisms is nontrivial). For an arbitrary **SPLL**-morphism $\langle \mathrm{cm}, \mathrm{sm}\rangle$, construct a functor $\mathrm{aom}\colon \mathrm{AO} \to \mathrm{AO}'$ that induces an **SPLL**-morphism

$$\langle \mathrm{aom}, \mathrm{sm}\rangle\colon \langle \text{s-AO}(\mathrm{SC}_{\mathrm{L}}), \mathrm{L}\rangle \to \langle \text{s-AO}(\mathrm{SC}'_{\mathrm{L}'}), \mathrm{L}'\rangle.$$

The starting point is the fact stated in Sec. 2 that the category of AO-models AO constructed from constituents of the L-technology SC_{L} is a vertex of the limit of **CAT**-diagram with schema $\boldsymbol{U}\,\mathrm{sig}\colon \text{c-DESC} \to \mathrm{SIG} \leftarrow \mathrm{LAB} : \mathrm{dom} \circ \mathrm{il}$, where LAB is a full subcategory of the arrow category $\mathrm{SIG}^{\mathit{2}}$ whose class of objects consists of all L-morphisms. It is easy to verify that the **SPLL**-morphism $\langle \mathrm{cm}, \mathrm{sm}\rangle$ induces a natural transformation of such diagrams, which will be denoted by ξ with components $\mathrm{cm}\colon \text{c-DESC} \to \text{c-DESC}'$, $\mathrm{sm}\colon \mathrm{SIG} \to \mathrm{SIG}'$, $\mathrm{sml}\colon \mathrm{LAB} \to \mathrm{LAB}'$, where sml is a functor that acts as sm^{2}. Note that composition of **SPLL**-morphisms induces composition of such natural transformations. Let $\mathrm{aom} = \lim(\langle \xi, 1_{\boldsymbol{U}}\rangle)\colon \mathrm{AO} \to \mathrm{AO}'$ (the functor lim is defined by duality similarly to the functor colim that was introduced in Sec. 1).

By the definition of limit, $\mathrm{mod}' \circ \mathrm{aom} = \mathrm{cm} \circ \mathrm{mod}$ (hence $\mathrm{int}' \circ \mathrm{aom} = \mathrm{sig}' \circ \mathrm{mod}' \circ \mathrm{aom} = \mathrm{sig}' \circ \mathrm{cm} \circ \mathrm{mod} = \mathrm{sm} \circ \mathrm{sig} \circ \mathrm{mod} = \mathrm{sm} \circ \mathrm{int}$) and $\mathrm{asp}' \circ \mathrm{aom} = \mathrm{sml} \circ \mathrm{asp}$ (hence $\mathrm{str}' \circ \mathrm{aom} = \mathrm{codom}' \circ \mathrm{il}' \circ \mathrm{asp}' \circ \mathrm{aom} = \mathrm{codom}' \circ \mathrm{il}' \circ \mathrm{sml} \circ \mathrm{asp} = \mathrm{sm} \circ \mathrm{codom} \circ \mathrm{il} \circ \mathrm{asp} = \mathrm{sm} \circ \mathrm{str}$). Consequently, we have

$$\begin{aligned}\mathrm{aom}\colon & \mathrm{AO} \to \mathrm{AO}' \\ & : \langle A, l\rangle \mapsto \langle \mathrm{cm}(A), \mathrm{sm}(l)\rangle,\ \langle f, b\rangle \mapsto \langle \mathrm{cm}(f), \mathrm{sm}(b)\rangle.\end{aligned}$$

The functor aom is faithful since the right side of the equality $\mathrm{mod}' \circ \mathrm{aom} = \mathrm{cm} \circ \mathrm{mod}$ consists of faithful functors. In addition, the functor aom has left adjoint

$$\begin{aligned}\mathrm{aom}^*\colon & \mathrm{AO}' \to \mathrm{AO} \\ & : \langle A', l'\rangle \mapsto \langle \mathrm{cm}^*(A'), \mathrm{sm}^{-1}(l')\rangle,\ \langle f', b'\rangle \mapsto \langle \mathrm{cm}^*(f'), \mathrm{sm}^{-1}(b')\rangle;\end{aligned}$$

the unit of this adjunction is an identity. Further, the functor aom preserves colimits of all configurations in $\text{s-AO}(\mathrm{SC}_\mathrm{L})$ since functors $\mathrm{cm} \circ \mathrm{mod}$, int, and str preserve colimits of any configuration, and the functor sm being an isomorphism of categories preserves colimits of any diagram. Verify that the functor $\mathbf{D}\mathrm{aom}$ sends configurations in $\text{s-AO}(\mathrm{SC}_\mathrm{L})$ to configurations in $\text{s-AO}(\mathrm{SC}'_{\mathrm{L}'})$, i.e., that $\mathrm{sm} \circ \mathrm{AOInt}_\mathrm{int} \subseteq \mathrm{AOInt}_{\mathrm{int}'}$. Choose arbitrary diagrams $\Theta \in \mathrm{AOInt}_\mathrm{int}$ and $\Delta \in \mathbf{D}\mathrm{int}'^{-1}(\{\mathrm{sm} \circ \Theta\})$, let $\Sigma = \mathrm{aom}^* \circ \Delta$. We have $\mathrm{aom} \circ \Sigma = \Delta$ and $\mathrm{int} \circ \Sigma = \mathrm{sm}^{-1} \circ \mathrm{sm} \circ \mathrm{int} \circ \Sigma = \mathrm{sm}^{-1} \circ \mathrm{int}' \circ \Delta = \Theta$. Hence $\mathrm{mod}' \circ \Delta = \mathrm{mod}' \circ \mathrm{aom} \circ \Sigma = \mathrm{cm} \circ \mathrm{mod} \circ \Sigma \in \mathrm{Conf}'$ and also the diagram $\mathrm{str}' \circ \Delta = \mathrm{str}' \circ \mathrm{aom} \circ \Sigma = \mathrm{sm} \circ \mathrm{str} \circ \Sigma$ has a colimit. Similarly to the proof of Corollary 4.6, we conclude that diagram Δ is aspectually determined. So is its every push by $\text{tr-AO}'$-morphisms. Hence $\mathrm{sm} \circ \Theta \in \mathrm{AOInt}_{\mathrm{int}'}$.

It follows that pair $\langle \mathrm{aom}, \mathrm{sm}\rangle$ is indeed an **SPLL**-morphism of AO-technologies. Hence, by Corollary 4.3, there exists a functor

$$\begin{aligned}\mathbf{aom}\colon & \mathbf{SPLL} \to \mathbf{ARCH} \\ & : \langle \mathrm{SC}, \mathrm{L}\rangle \mapsto \mathrm{AO}_\mathrm{int}(\mathrm{SC}_\mathrm{L}),\ \langle \mathrm{cm}, \mathrm{sm}\rangle \mapsto \langle \mathrm{aom}, \mathrm{sm}, \mathrm{aom}(-^{\mathrm{op}})^{\mathrm{op}}\rangle.\end{aligned}$$

There exists a natural transformation of this functor to a standard embedding $\mathbf{ispl} \circ \mathbf{ispll}\colon \mathbf{SPLL} \hookrightarrow \mathbf{ARCH}$, where $\mathbf{ispll}\colon \mathbf{SPLL} \hookrightarrow \mathbf{SPL}$ is an embedding of a subcategory. This natural transformation sends each **SPLL**-object $\langle \mathrm{SC}, \mathrm{L}\rangle$ to **ARCH**-morphism $\mathrm{ao}_\mathrm{int}\colon \mathrm{AO}_\mathrm{int}(\mathrm{SC}_\mathrm{L}) \to \mathrm{SC}_\mathrm{L}$ constructed in the proof of Corollary 3.2. This illustrates the naturality of the AO-extension procedure for technologies induced by labeling.

Conclusion

Software engineering technologies like MDE and AOP have great potential as a means of reducing the cost of software systems design. It was verified in practical industrial projects that they allow replacing dozens of programmers by small groups that create tools for automatic generation of massive source code [27]. As a mathematical device for rigorous analysis and improvement of design processes, especially under conditions of joint use of diverse models, it is reasonable to involve category theory. Categorical structures reflect the key features of the technological procedures of software design, such as:

- extraction of an integration interface for a formal model of a program is represented by a functor of a special kind, viz. a faithful coreflector;
- system configurations ("megamodels") are found among diagrams whose colimits are determined by an appropriate faithful coreflector;
- refinements that offer optimal tracing capabilities are selected by a certain regular procedure from morphisms dual to those sent to retractions by a faithful coreflector.

Theoretical results of this kind are used in solving problems whose improvised ad hoc solutions' complexity increases with increasing scale and complexity of software systems. Such problems arise while reconciling models of various parts and aspects of the system with each other and with source code (in particular, via tracing), while achieving the required values of the integral system performance and other system-wide quality indicators, and so on. Category-theoretic structures that formally describe the possible ways to solve the problem on an abstract conceptual level are constructed: components involved in the solution are mapped to objects of appropriate categories, technological activities are mapped to morphisms, transitions are mapped to functors, etc. Properties of solutions are calculated in the categories, and a solution that delivers extremal value of the target functional (cost, reliability, etc.) is selected from alternatives. Then the construction that corresponds to the selected solution is interpreted in terms of an appropriate technology, and tools necessary to automate it are found or created. Examples of this approach are given in [27].

By developing such a category-theoretic approach, in the future it will be possible to significantly improve the level of intelligence of software engineering tools and to pass on the construction, analysis, and optimization of design procedures to them. Modern CASE-tools are basically unable to do it; they are capable only of mechanically executing commands issued by engineers. Employing the language of category theory for formal machine-oriented representation of technologies and design techniques, it is possible to reduce intellectual activity to recognition and calculation of categorical structures. Note that this approach could lead to significant improvement in efficiency of automated design of material products as well.

Implementation of an automatic categorical solver for algebraic specification development problems as a part of the SPECWARE technology [35] can be considered as one of the first steps in this direction. Creation of full-scale multi-model tools of intelligent computer-aided systems design opens up many promising directions for future research.

REFERENCES

1. J. Adámek, H. Herrlich, and G. Strecker, *Abstract and Concrete Categories*, Wiley and Sons, New York (1990).
2. N. Aizenbud-Reshef, B. Nolan, J. Rubin, and Y. Shaham-Gafni, "Model traceability," *IBM Systems J.*, **45**, No. 3, 515–526 (2006).
3. R. J. Allen and D. Garlan, "A formal basis for architectural connection," *ACM Trans. Software Engineering Methodology*, **6**, No. 3, 213–249 (1997).
4. J. Brichau, R. Chitchyan, A. Rashid, and T. D'Hondt, "Aspect-oriented software development: an introduction," in: *Wiley Encyclopedia of Computer Science and Engineering*, Vol. 1, Wiley and Sons, New York (2008), pp. 188–198.
5. C. Brunette, J.-P. Talpin, A. Gamatié, and T. Gautier, "A metamodel for the design of polychronous systems," *J. Logic Algebraic Programming*, **78**, No. 4, 233–259 (2009).
6. A. Carboni, G. Janelidze, G. M. Kelly, and R. Paré, "On localization and stabilization for factorization systems," *Appl. Categ. Structures*, **5**, 1–58 (1997).
7. Z. Diskin and T. S. E. Maibaum, "Category theory and model-driven engineering: from formal semantics to design patterns and beyond," in: *Proc. 7th Workshop ACCAT'2012, Electron. Proc. Theor. Comput. Sci.*, **93**, 1–21 (2012).
8. R. Douence, P. Fradet, and M. Südholt, "Trace-based aspects," in: *Aspect-Oriented Software Development*, Addison-Wesley, Reading (2004), pp. 201–218.
9. Eclipse Modeling Project, `http://www.eclipse.org/modeling/`, The Eclipse Foundation (2014).
10. A. Egyed, P. Grünbacher, M. Heindl, and S. Biffl, "Value-based requirements traceability: lessons learned," in: *Design Requirements Engineering: A Ten-Year Perspective. Design Requirements Workshop, Cleveland, OH, USA, June 3–6, 2007, Revised and Invited Papers*, Lect. Notes Business Inform. Processing, Vol. 14, Springer, Berlin (2009), pp. 240–257.
11. J. L. Fiadeiro, A. Lopes, and M. Wermelinger, "A mathematical semantics for architectural connectors," in: *Generic Programming*, Lect. Notes Comput. Sci., Vol. 2793, Springer, Berlin (2003), pp. 190–234.
12. J. Goguen, "A categorical manifesto," *Math. Structures Comput. Sci.*, **1**, No. 1, 49–67 (1991).
13. J. Goguen, "Categorical foundations for general systems theory," in: *Advances in Cybernetics and Systems Research*, Transcripta Books, London (1973), pp. 121–130.

14. O. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in: *Proc. 1st Int. Conf. on Requirements Engineering. Colorado Springs, 1994*, pp. 94–101.
15. B. Graaf and A. van Deursen, "Visualisation of domain-specific modelling languages using UML," *Proc. 14th IEEE Int. Conf. on the Engineering of Computer-Based Systems ECBS'2007. Tucson, 2007*, pp. 586–595.
16. I. Groher and M. Völter, "Aspect-oriented model-driven software product line engineering," in: *Transactions on Aspect-Oriented Software Development VI*, Lect. Notes Comput. Sci., Vol. 5560, Springer, Berlin (2009), pp. 111–152.
17. R. Guitart and L. van den Bril, "Décompositions et lax-complétions," *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, **18**, No. 4, 333–407 (1977).
18. S. Hanenberg and R. Unland, "Roles and aspects: similarities, differences, and synergetic potential," in: *Object-Oriented Information Systems. 8th Int. Conf., OOIS 2002 Montpellier, France, September 2–5, 2002. Proceedings*, Lect. Notes Comput. Sci., Vol. 2425, Springer, Berlin (2002), pp. 507–520.
19. *ISO/IEC Standards 9126. Information Technology — Software Product Evaluation — Quality Characteristics and Guidelines for Their Use*, ISO, Geneva (1991).
20. F. Jouault, B. Vanhooff, H. Bruneliere, G. Doux, Y. Berbers, and J. Bezivin, "Inter-DSL coordination support by combining megamodeling and model weaving," in: *Proc. 2010 ACM Symp. on Applied Computing*, Sierre (2010), pp. 2011–2018.
21. A. Kannenberg and H. Saiedian, "Why software requirements traceability remains a challenge," *J. Defense Software Engineering*, July/August 2009, 14–19.
22. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-oriented programming," in: *ECOOP '97 — Object-Oriented Programming: 11th Europ. Conf., Jyväskylä, Finland, June 9–13, 1997, Proceedings, Vol. 11*, Lect. Notes Comput. Sci., Vol. 1241, Springer, Berlin (1997), pp. 220–242.
23. D. S. Kolovos, R. F. Paige, and F. A. C. Polack, "The grand challenge of scalability for model driven engineering," in: *Models in Software Engineering: Workshops and Symposia at MODELS 2008, Toulouse, France, September 28 — October 3, 2008. Reports and Revised Selected Papers*, Lect. Notes Comput. Sci., Vol. 5421, Springer, Berlin (2009), pp. 48–53.
24. S. P. Kovalyov, "Formal approach to aspect-oriented scenario modeling," *Sib. Zh. Industr. Mat.*, **13**, No. 3, 30–42 (2010).
25. S. P. Kovalyov, "Modeling aspects by category theory," in: *Proc. 9th Workshop on Foundations of Aspect-Oriented Languages. Rennes, France, 2010*, pp. 63–68.
26. S. P. Kovalyov, "Diagrammatic description of software systems composition," *Vestn. Novosibirsk Gos. Univ. Ser.: Mat., Mekh., Informatics*, **12**, No. 3, 103–126 (2012).
27. S. P. Kovalyov, "Increasing efficiency of large-scale information-control systems design processes," *Proc. XII Russian Workshop on Control Problems VSPU-2014* [In Russian], ICS RAS, Moscow (2014), pp. 9291–9300.
28. A. Lopes and J. L. Fiadeiro, "Revisiting the categorical approach to systems," in: *Algebraic Methodology and Software Technology: 9th Int. Conf., AMAST 2002, Saint-Gilles-les-Bains, Reunion Island, France, September 9–13, 2002. Proceedings*, Lect. Notes Comput. Sci., Vol. 2422, Springer, Berlin (2002), pp. 426–440.
29. S. Mac Lane, *Categories for the Working Mathematician*, Springer, Berlin (1998).
30. M. Pinto, L. Fuentes, and J. M. Troya, "DAOP-ADL: an architecture description language for dynamic component and aspect-based development," in: *Generative Programming and Component Engineering: Second Int. Conf., GPCE 2003, Erfurt, Germany, September 22–25, 2003, Proceedings, Vol. 2*, Lect. Notes Comput. Sci., Vol. 2830, Springer, Berlin (2003), pp. 118–137.
31. V. R. Pratt, "Modeling concurrency with partial orders," *Int. J. Parallel Programming*, **15**, No. 1, 33–71 (1986).
32. A. Rutle, A. Rossini, Y. Lamo, and U. Wolter, "A formalisation of the copy-modify-merge approach to version control in MDE," *J. Logic and Algebraic Programming*, **79**, No. 7, 636–658 (2010).

33. D. C. Schmidt, “Model-driven engineering,” *IEEE Computer*, **39**, No. 2, 25–32 (2006).
34. I. Sommerville, *Software Engineering*, Addison-Wesley, Reading (2010).
35. Y. V. Srinivas and R. Jüllig, “SPECWARE: formal support for composing software,” in: *Mathematics of Program Construction ’95*, Lect. Notes Comput. Sci., Vol. 947, Springer, Berlin (1995), pp. 399–422.
36. F. Steimann, “The paradoxical success of aspect-oriented programming” *Proc. Int. Conf. OOPSLA’06. Portland, 2006*, pp. 481–497.
37. S. N. Vassilyev, “Knowledge formalization and control on the basis of positively constituted languages,” *Inform. Technol. Vychisl. Sist.*, No. 1, 3–17 (2008).

Serge P. Kovalyov
Institute for Control Problems, Russian Academy of Sciences, Moscow, Russia
E-mail: `kovalyov@nm.ru`