Cognitive Systems Research 83 (2024) 101185

Contents lists available at ScienceDirect

Cognitive Systems Research

journal homepage: www.elsevier.com/locate/cogsys

Semantic configuration model with natural transformations

Viacheslav Wolfengagen [a,*], Larisa Ismailova [a], Sergey Kosikov [b], Igor Slieptsov [b], Sebastian Dohrn [b], Alexander Marenkov [b], Vladislav Zaytsev [b]

[a] National Research Nuclear University "Moscow Engineering Physics Institute", Moscow 115409, Russian Federation
[b] NAO "JurInfoR", Moscow 119435, Russian Federation

ARTICLE INFO

Action editor: Tingting Liu

Keywords:
Information process
Configuration
Morphing
Cognitive preference
Semantic net
Functor
Natural transformation

ABSTRACT

In the present work, efforts have been made to create a configuration-based approach to knowledge extraction. The notion of granularity is developed, which allows fine-tuning the expressive possibilities of the semantic network. As known, the central issues for knowledge-based systems are what's-in-a-node and what's-in-a-link. As shown, the answer can be obtained from the functor-as-object representation. Then the nodes are functors, and the main links are natural transformations. Such a model is applicable to represent morphing, and the object is considered as a process, which is in a harmony with current ideas on computing. It is possible to represent information channels that carry out the transformations of processes. The possibility of generating displaced concepts and the generation of families of their morphs is shown, the evolvent of stages of knowledge and properties of the process serve as parameters.

1. Introduction

Configuration is used in science and technology, including in various areas of computer science: in software engineering, information technology, information systems, databases, Artificial Intelligence … . At the same time, specific interpretations of configuration look different in form. We see our task in the variety of forms to find a pattern and explore the idea of configuration.

We have built a representative end-to-end example illustrating the essence of configuration as a process:

– parts of the system are assumed as varying;

– the connections of components to the whole also can vary.

Variable components are conveniently represented by *functors*, and the connections between them – counterpart dependencies, *morphisms* – reflect the essence of the ongoing transformations and are called *natural transformations*. The same terminology is used in the functor category. The amount of knowledge involved from category theory is quite elementary, it can be gleaned in any book on this subject.

More meaningful is the cross-cutting example we establish and use, which is 'biologically inspired'. This is a particular process of *transforming* a tadpole into a frog. It is well known that in this case *morphing* occurs — a kind of *natural transformation*, also modeled by a functor category. At the same time, a particular tadpole (named Hugo) changes its properties during configuration-morphing, *becoming* a frog (transforming into a frog). But the cognitive component of the process 'remembers' how the transformation occurs over 'stages'.

The example we have considered, as it turns out, gives a *characterization*. In fact, this is a *theory of semantic configuration*. From the point of view of predicative semantic networks, as noted in the literature on AI, this is a difficult, hardly ever solvable task. We have considered one example, but once it is established, other examples come to mind. Some of them are reviewed by a team of authors. One of these is the *entanglement problem* called as the 'fast Smith problem'. It should be noted that the way to solve this problem in a generalized form is only outlined till now.

In this sense, the representation of configuration-as-a-process that we put forward can be productive and effective.

Much of the work in computing deals with configuration. Most often this is related to applied development. Recently, there has been interest in endowing an information system with customizable, parameterized semantics.

1.1. Aims and scope

In this paper, we mainly consider semantic configuration. In this case, the system is endowed with a family of parameters for which tuning is performed.

Particular attention is paid to the *configurator* — the core of system configuration. The notion of configuration is studied from the standpoint of modern computing, when information processes are taken as

* Corresponding author.
E-mail addresses: jir.vew@gmail.com (V. Wolfengagen), jir.lyui@gmail.com (L. Ismailova), kosikov.sv@gmail.com (S. Kosikov), igor.slieptsov@mail.com (I. Slieptsov), sdohrn@mail.ru (S. Dohrn), qwexly@gmail.com (A. Marenkov), zvs2403@gmail.com (V. Zaytsev).

https://doi.org/10.1016/j.cogsys.2023.101185
Received 25 May 2023; Received in revised form 2 October 2023; Accepted 7 November 2023
Available online 10 November 2023

the basis, and their interaction should be tuned. Under these assumptions, configuration studies were not performed yet. There are several difficulties along the way:

– usually algebraic methods with set-theoretic interpretation and finite signature are chosen for modeling;

– insufficient attention is paid to the phase of abstraction;

– in fact, developers immediately proceed to the algorithmization phase;

- in practice, cumbersome heterogeneous architectures arise.

In this paper, the functor category is taken as a metatheory. In its graph representation, functors are placed at nodes, and natural transformations between them are arcs, and the representing diagram is *commutative*. From the point of view of the flow of information processes, channeling occurs, and the previously noted shortcomings of algebraic models can be overcome.

In general, the way of basic research of configuration as a process is proposed in this work. As it turns out, this can be done while remaining within the framework of the doctrine of computational thinking as it was stated by Association for Computing Machinery.

1.2. Key terminology

A configuration is the knowledge of what a working (actual) system is, selected from a set of its possible options.

Configuration in software engineering is understood in a slightly different context and is generally based on a similar conceptual framework, but with pronounced specializations.

The configuration is *under control* if the system definition configuration matches the system implementation configuration. If some parts of these configurations do not correspond to each other, then one speaks of configuration collisions.

Configuration *management* is the technical discipline of systems engineering that ensures that the proper (conceived, approved) configuration of a system is maintained throughout its life cycle. Simply assume, that configuration management is the practice of ensuring version compatibility (non-collisions!) and completeness of system parts (non-collisions!) throughout the life cycle. Configuration management is the practice of systems engineering management and is concerned with maintaining the integrity of a system throughout its entire life cycle. As part of this practice, various types of specifications for purchased/manufactured equipment/products are issued — BOM (bill of materials, list of components).

Relegating configuration management to systems engineering means that configuration necessarily applies to the entire system. The configuration *manager* is a systems engineer, along with a requirements engineer, a system architect, an integrator.

1.3. Novelty

The *novelty* of the proposed configuration method is the possibility of controlled granulation of a family of semantic maps. In this case, a generalized configurator by V. Lefebvre appears. A representative end-to-end example is constructed, illustrating the resulting computational effects and the corresponding cognitive processes. It is based on a semantic map whose nodes are variable domains and whose links are natural transformations. One of the useful mathematical properties of the constructed model is the ability to support end-to-end design technologies.

In general, a new analytical study of the notion of configuration has been carried out.

Another extractable advantage is the extension of the computational thinking approach to a wide class of semantic modeling problems. For further research, it is planned to develop special tools, because known semantic configuration systems do not have such mathematical properties.

1.4. The originality

The original essence of the study is as follows. In this paper, we propose a semantic configuration model equipped with natural transformations. The configurator is a family of semantic maps, parameterized by ‘knowledge stages’ and ‘course of events’ evolvent. Functors are located at the nodes of the semantic map, and natural transformations serve as links between them. The representing graph is a commutative diagram on which it is possible to set and solve the problem of channeled distribution of information.

1.5. State-of-the-art

Increasing the cognitive capabilities of operating information systems and services continues to be in the focus of attention of manufacturing companies. For example, Microsoft is currently actively developing its Azure service, especially in terms of cognitive search capabilities. A feature is supported that retrieves search results based on the semantic meaning of search terms (not just keywords). This is done using modern language models based on *transformers* that are supported by the search engine.

The implementation avoided training large-scale AI models or managing expensive infrastructure to deploy them. At the same time, we can optionally enable the *semantic configuration* feature using an intuitive interface that helps clients easily configure *semantic search* for their datasets. Practically, this looks like the ability to include fields by category, such as document or catalog title, descriptive text or detailed content, and keywords or terms associated with the document.

In other words, semantic configurations have become part of the *index definition* and can easily be created simply in the Azure portal. The achievement is that there is no need to re-index the content to create a semantic configuration or use semantic search. AI can bring benefits ranging from improved reception to data exploration. Because there are many valuable enterprise data resources, the challenge is to bring as much ‘analytics’ into the product as possible.

This may be ‘unavailable’ content; about 80% of business-related data is in unstructured formats such as PDF, PowerPoint, Word documents, JPEG, CSV, and so forth. The implementation of data enrichment by AI methods allows us to extract structure, analytical information and transform information from existing data. This shows the *growth of the cognitive component* of the service.

Although the achievements of AI in the field of information technology are of interest and are extremely in demand, nevertheless, not everything is all right in the field of artificial intelligence itself. The confusion, misdirection of emphasis, and duplication of effort ongoing in research on knowledge extraction and representation have been noted. There has been a push to overcome the confusion on this subject so far. There appears to be a loss of touch with the core ideas and assumptions on which the biggest breakthroughs in AI were based.

Theoretical constructions tend to become more complex and are equipped with an ever-increasing mass of subtleties and nuances that are difficult to comprehend, and terminological inconsistency is observed from work to work. The lack of conceptual unity does not facilitate the exchange of acquired experience, and many efforts go unnoticed, dissolving into burdensome details. It is well known that the design of something is guided by the requirements of the integrity of the description and the unity of the idea. Design involves the *gradual refinement* of the components, their functionality and relationships (Schubert, 1976). The importance of choosing an appropriate indexing is also discussed in works on semantic networks (Woods, 1997).

We can take the liberty of suggesting that the semantic web is based on the category of functors with natural transformations, attracting the mathematical ideas of Scott (1980).

There were a lot of attempts to study the meaning of node. A node is a point of intersection/junction in a data network (Brachman, 1977).

In an environment where all devices are accessible through a network, all of these devices are considered as nodes. The individual definition of each node depends on the type of network to which it belongs.

The notion of a 'psychological configuration' – psychological settings, – is also extended to networks, which is understood as some initial principle of integrity and stability, which determines the features of the structural and functional organization and regulation of a person's mental activity in the process of perception in general (Brachman, 1979), and the data transmission network in particular.

Even in the early generic papers, when solving problems of knowledge representation, semantic networks are most often used. There are questions about what should be placed in their links (Woods, 1975), and what should be better represented in the nodes (Allen & Frisch, 1982). The answer to these questions contributes to the *configuration* of the presented knowledge, which means the solution of a certain range of tasks.

This state of affairs is reflected in numerous discussions on *software configuration* (Jaeger, 2016; Tao & Plaisted, 2021), problems of development and application of *configurators* (Cobanli, 2011; de Jong, 1988; McGuinness et al., 1998; Myllärniemi et al., 2012; Raatikainen et al., 2018), *model configuration* issues (Noorian, Bagheri, & Du, 2016; Sun, Cho, Gray, & White, 2011; Uta et al., 2021). The terminology base depends on the development context and varies from case to case.

As it was noted, there is a *dependency* between the generality of understanding and flexibility of context. Thus, a notion of 'configuration' tends to take into account the idea of *relative location*, and therefore considers an object in relation to various fields of view and 'correspondences'.

Note that in computing 'configuration' means the way a computer or computer system is put together; a specific set and arrangement of internal and external components, including hardware, software, and devices. Also this is the way a software program or device is set up for a particular computer, computer system, or task; the specific settings for a program or device. More general understanding leads to more flexible contexts, so that 'configuring' tends to take into account the idea of relative arrangement and hence considers the object relatively to disparate fields of view and 'reference points'.

For example, in the semantics of programming languages, an instruction or an expression is configured by adjusting to the environment of the calculation and producing either a new state of the computation or a value. In semantic networks, the execution of logical operations over frames causes changes in the ISA hierarchy of both concepts and frames, derived concepts and frames are generated at the intensional level, and associated database relations are formed at the extensional level. This applies to knowledge representation systems of a procedural or declarative nature. We just mention these features because, though interesting and important, the detailed study is out of the current paper scope.

For knowledge acquisition or mining systems, this interpretation is used more generally. In this regard, an extended interpretation from interdisciplinary positions is adopted. Thus, configuring (lat. configuratio – mutual arrangement) is considered as a special logical and methodological device, a mental technique for synthesizing knowledge of different subjects, different ideas about the same object. The solution of these problems is connected with the need to carry out a *theoretical synthesis* of knowledge developed in various scientific subjects.

In the starting point, we single out and distinguish: (a) the process of *representing* an object as a system and (b) the process of *constructing* an object according to some project.

1. Representing an object as a system: To solve some cognitive tasks, an object must be depicted as divided into elements. The image should capture the connections and relationships that turn 'dismemberment' into a kind of 'wholeness'. Elements, as well as connections and relationships, can be distinguished in different ways depending on the 'standards' in which the researcher reflects the object and which determine its structure.
2. Constructing an object as a system: When solving practical problems, special objects are built. They are built from a set of parts according to a special 'project-image', which guides the 'constructor'. The 'constructor' takes the 'image' to another realm. He embodies it in new material. The resulting object has a structure because it is created.
3. Examining an object that is constructed: The researcher may be faced with a special task — to isolate or *remove* from the object the structure with which it is endowed. But the object, as an object of study, acquires the structure of its depiction (for 'to see' a structure in an object means to depict it). Therefore, it is necessary to choose such a system representation of the object that *reflects exactly the structure* acquired by it. From this need follows a natural principle by which the explorer should be guided: he should *borrow the 'design' that guided the 'constructor' and use it in a new function* — as a means of system representation.

1.6. Preliminary contribution of the authors

The contribution of the team of authors to the problem is reflected in the works (Ismailova, Wolfengagen, & Kosikov, 2022d; Ismailova, Wolfengagen, Kosikov, & Andronov, 2023), where various techniques for constructing computational models are considered. Suggestions for semantic modeling are contained in Wolfengagen, Ismailova, and Kosikov (2022), (Wolfengagen, Ismailova, Kosikov, & Babushkin, 2021). Examples of *parameterized semantic constructs* are given in Ismailova, Wolfengagen, and Kosikov (2022e), (Ismailova, Wolfengagen, & Kosikov, 2022a, 2022b, 2022c). Interesting model solutions (Wolfengagen, Ismailova, Kosikov, & Dohrn, 2022) are summarized and specified in Ismailova, Wolfengagen, and Kosikov (2021c), (Ismailova, Wolfengagen, & Kosikov, 2021d). Preliminary work on *channeled modeling* of information processes has been done in Ismailova, Wolfengagen, and Kosikov (2021a), (Ismailova, Wolfengagen, & Kosikov, 2021b; Kosikov, Ismailova, & Wolfengagen, 2021).

To answer all the questions raised, the content of this work is divided into five main sections. Section 1 contains a general discussion of configuration issues, necessary to understand the direction of the discussion in the main content of the work. Section 2 covers the influential applied related works.

The important issue of database configuration is beyond the scope of this paper. In Section 3 we follow the general methodological guidelines of V. Lefebvre on the configuration model. Section 4 develops a notion of configuration in accordance with the main criteria of computational thinking, which implements the configuration model. Finally, we develop a computational model of semantic configuration with transformers in the form of natural transformations. Ideas of conceptual mathematics are involved in its construction. Section 5 contains the reasons in favor of effectiveness of computational thinking when solving the selection of configurator and configuring tasks.

2. Related works

The *mental rotation* ability was studied in Gentile and Lieto (2022) as an essential spatial reasoning skill in human cognition and has proven to be an essential predictor of mathematical and STEM skills, critical and computational thinking. Despite its importance, little is known about when and how mental rotation processes are activated in games explicitly targeting spatial reasoning tasks. In particular, the relationship between spatial abilities and TetrisTM has been analyzed several times in the literature. The results obtained are for the TetrisTM game by explicitly modeling mental rotation via an ACT-R based cognitive model controlling a virtual agent. But there was a need to improve its educational effectiveness. The more or less similar way was used in developing an ontology-based tool for dynamic generation, classification and recommendation of novel contents in online libraries (Barbera, Lieto, & Pozzato, 2022). In Lieto, Pozzato, Perrone, and Chiodino

(2019) there was described an approach to goal-oriented knowledge capturing that, differently from the standard knowledge acquisition pipelines, employs a dynamic conceptual reframing mechanism relying on a non monotonic reasoning procedure.

A notion of *configuring* in computer science field is a multi-meaning one and deserves a more detailed consideration, especially in connection with particular application areas.

There is a discussion in Falk, Frauenberger, and Kannabiran (2022) on how shortening or lengthening design processes configure decision making, and there have been repeated calls for developing time-sensitive discourses in HCI and design research, and for re-examining engagement with power. In response, they explore the relationship between time and decision making in design processes in order to better understand how this configures power structures. There was an attempt to analyze two design cases: a short-term hackathon and a long-term design process.

In Rust, Picard, and Ramparany (2022) there is a discussion of resilient distributed constraint reasoning to autonomously configure and adapt IoT environments, there was investigation of multi-agent techniques to install autonomy and adaptation in IoT-based smart environment settings, like smart home scenarios. They particularly make use of the smart environment configuration problem (SECP) framework, and map it to a distributed optimization problem (DCOP). This consists in enabling smart objects to coordinate and self-configure as to meet both user-defined requirements and energy efficiency, by operating a distributed constraint reasoning process over a computation graph.

A survey study in Balogova and Brumby (2022) of how users configure video-conference tools for online meetings concludes that many knowledge workers now spend prolonged hours on video calls each day. However, it is unclear how people set up their videoconferencing tools now that they are highly accustomed to this communication medium. To investigate this, an online questionnaire is distributed that explored 115 users' videoconferencing setup preferences, asking them about their typical video and camera setup for meetings. Structure of reporting of results around four themes: (1) video layout preferences, (2) camera preferences, (3) self-view window preferences, and (4) multitasking behavior during meetings. Results show that participants preferred using the active speaker view when joining large meetings with a single key presenter, and the grid view when on social calls and meetings requiring collaboration.

Configuring federative, hierarchical attention-enhanced meta-learning network for personalized federated learning was studied in Gao, Wang, Liu, Zhang, and Ma (2023). Federated learning, as a distributed machine learning framework, enables clients to conduct model training without transmitting their data to the server, which is used to solve the dilemma of data silos and data privacy. It can work well on clients having similar data characteristics and distribution. However, it has some limitations where the dataset of clients may be different in distribution, quantity, and concept in many application scenarios.

Automated identification of security-relevant configuration settings using NLP is studied in Stöckle, Wasserer, Grobauer, and Pretschner (2023). To secure computer infrastructure, there is a need to configure all security-relevant settings. There is a need of security experts to identify security-relevant settings as well, but this process is time-consuming and expensive. An evaluation shows that trained classifiers do not perform well enough to replace the human security experts but can help them classify the settings.

Configuration for continuous integration builds is considered in Zhang, Chen, Hu, Peng, and Zhao (2023). Despite the benefits, continuous integration (CI) can incur high costs. One of the well-recognized costs is long build time, which greatly affects the speed of software development and increases the cost in computational resources. While there exist configuration options in the CI infrastructure to accelerate builds, the CI infrastructure is often not optimally configured, leading to CI configuration smells. Attempts have been made to detect or repair CI configuration smells.

An exploratory study (Randrianaina, Tërnava, Khelladi, & Acher, 2022) of benefits and limits of incremental build of software configurations. has being done. This is important as software projects use build systems to automate the compilation, testing, and continuous deployment of their software products. As software becomes increasingly configurable, the build of multiple configurations is a pressing need, but expensive and challenging to implement. The current state of practice is to build independently (a.k.a., clean build) a software for a subset of configurations. While incremental build has been studied for software evolution and relatively small changes of the source code, it has surprisingly not been considered for software configurations.

A self-adaptive framework for configuration tuning was considered in Wang, Hoffmann, and Lu (2022). This is due to software systems increasingly expose performance-sensitive configuration parameters, or PerfConfs, to users. Unfortunately, the right settings of these PerfConfs are difficult to decide and often change at run time. To address this problem, prior research has proposed self-adaptive frameworks that automatically monitor the software's behavior and dynamically tune configurations to provide the desired performance despite dynamic changes. However, these frameworks often require configuration themselves; sometimes explicitly in the form of additional parameters, sometimes implicitly in the form of training.

Efficient configuration of optimization algorithms (de Souza, Ritt, & López-Ibáñez, 2022) is desirable in applications. Propose is of a set of capping methods to speed-up the automatic configuration of optimization algorithms. This aids to build a performance envelope based on previous executions of known configurations, which defines the minimum required performance for new configurations. Then, the performance envelope to evaluate new configurations is used, stopping poor performers early. There arise different methods to aggregate previous executions into a performance envelope, and evaluate them on several configuration scenarios.

Fuzzing configurations of program options were used in Zhang, Klees, Wang, Hicks, and Wei (2023). While many real-world programs are shipped with configurations to enable/disable functionalities, fuzzers have mostly been applied to test single configurations of these programs. There was an empirical study to understand how program configurations affect fuzzing performance. It was found that limiting a campaign to a single configuration can result in failing to cover a significant amount of code.

Black-box optimization in a configuration system (Kucher, Balyo, & Christensen, 2022) gives a useful hint how to overcome an 'under-definability' in configuring process. The product configurator Merlin is a CPQ solution (Configure, Price, Quote) that enables fast, error-free configuration and quotation generation for products with many variants. However, there is a strict need to formulate the set of criterions.

Multi-objective optimal configuration method for off-grid integrated system is proposed in Liu, Tan, Lin, Chen, and Wang (2022). Based on the loss function, there have been developed four different types of mathematical models in order to satisfy the needs of different residents, but they cover just several particular cases without generalization.

The dynamic product configuration user interface as a vision motivated by the cyber–physical production systems domain is analyzed in Fadhlillah, Feichtinger, Gutiérrez Fernández, and Rabiser (2023). Cyber-Physical Production Systems (CPPSs) are large-scale industrial systems in which hardware and software are deeply intertwined. CPPS software has to be highly variable to support frequently changing customer and hardware requirements. Managing the overall variability of such large-scale industrial software is challenging.

An idea of enhancing the configuration tuning pipeline of large-scale distributed applications using large language models (Somashekar & Kumar, 2023) seems to be promising. While the information in product manuals and technical documents guides the tuning process, manual collection of meta-data for all application parameters is laborious and not scalable.

Improved regression models for algorithm configuration is proposed in de Souza and Ritt (2022). Offline algorithm configuration methods search for fixed parameter values for a given set of problem instances. For each parameter, such methods perform an equivalent to a constant regression, since the parameter value remains constant for any problem instance.

Understanding configuration dependencies of file systems (Mahmud, Zhang, Gatla, & Zheng, 2022) looks as a nice idea. The contributors identified a prevalent pattern called multi-level configuration dependencies and built a static analyzer to extract the dependencies and leverage the information to address different configuration issues. The preliminary prototype is able to extract 64 multi-level dependencies with a low false positive rate.

Interactive feature modeling with background knowledge for validation and configuration (Vandevelde, Callewaert, & Vennekens, 2022) is somewhere closer to our proposal in the current paper. Feature modeling enables a straightforward representation of a product's features, components, and the relations between them. The modelers can interact with the feature model and its *background knowledge* to explore the problem space on-the-fly.

The list of related works grows by the time and could be continued without big efforts. But state-of-the-art here shows the we have a variety of particular proposals, what to configure. However, a notion of configuration and/or configuring is left, as it was. This demonstrates the needs to advance the studying a notion of configuration in-principal. In addition, there is no unity in mathematical approaches in use, all of them are aimed to solving the particular applied tasks. In this paper there is an attempt to study configuring using a sound mathematical and meta-mathematical ground.

3. Configuration by V. Lefebvre

It can, of course, be immediately stated that the notion of configuration is understood as some initial principle of integrity and stability, which determines the features of the structural and functional organization and regulation in the process of perception. Let us show that this path is representative and even constructive.

3.1. Configurator

Configuring involves building a special structural model of the configurator. With the help of it, multi-subject knowledge is accommodating in a single theoretical representation of some complex, systemic object. The configurator serves as an ideal image of the structure of an object, explains and substantiates existing knowledge, shows which aspects of the object they are projections of. For the first time, configurators, as a special class of models, were apparently singled out by V.A. Lefebvre (Lefebvre, 2001; Lefebvre & Smolyan, 1968).

Let there be several different system representations of one object, and the elements into which the whole is divided are fundamentally distinct in different system representations. The object appears to be projected onto several screens. Each screen sets its own division into elements, thus generating a certain structure of the object.

The screens are connected to each other in such a way that the researcher is able to correlate different pictures without going through the object itself. Such a device, synthesizing various system representations, was called a *configurator*.

A simple example of a configurator is a Cartesian coordinate system in geometry. The correlation of knowledge from different subject areas with each other cannot be carried out in the plane of the knowledge itself, but involves their opposition to the object. Correlation of knowledge becomes possible only through the construction of a kind of ontology that represents the reality of the object as such (despite the fact that the status of ontology can be attributed to one of the particular knowledge, the functional and interpretative difference between knowledge related to the object and ontology acting in functions of the image of the object as such).

In the process of knowledge synthesis, the role of such an auxiliary 'ontology' is performed by the configurator. In general, working with the configurator involves two types of mental movements:

(1) from existing knowledge and theoretical schemes to the configurator, in order to determine the structure of the object;

(2) from the configurator to the synthesized knowledge and theoretical schemes, in order to substantiate and interpret them as projections obtained from a certain perspective of the object.

3.2. Performing configuration

The implementation of configuration requires a special over-subject organization of thinking, which was perceived as specifically methodological effort. Configuration leads to such a transformation of theoretical schemes and knowledge that their synthesis becomes possible. After the implementation of the synthesis, the configurator can be eliminated in the structure of the theory, but more often it is retained as a basic model and becomes the basis of a new discipline.

As a more complex example of a configurator, *functor category* can be used, producing *semantic maps*. Configuration management in software, is known as attachment of Software Configuration Management Patterns. Configuration is also understood as overcoming contradiction in conflicting structures. Configuration, according to the idea of V. Lefebvre, overcomes conflicts through the algebra of conflicts as a broad approach in science and technology. But this has not yet been brought to practical technologies.

Of course, it begs to immediately write down an evaluating mapping that assigns values to program constructs:

$$\|\cdot\|\cdot : \text{expression} \times \text{environment} \to \text{value}.$$

In general, this is the right way to focus on the main parameters and vary them:

$$\text{configuration} : \text{system description} \times \text{parameter} \to \text{configured system}$$

At the same time, everything is in a harmony with the main thesis that software configuration management is a systems engineering process designed to track changes in the configuration metadata of software systems.

4. Configuration process

Because the notion of configuration is understood as some initial principle of integrity and stability, which determines the features of the structural and functional organization and regulation in the process of perception, then a suitable mathematical idealization should not be cumbersome. Let us start with the *assumption* that it is possible to select *objects* in the subject area and establish *relationships* between them. Objects are accessible to perception through *properties* (is there an object that does not exhibit properties?), and stable links between objects are considered as relationships (are there objects that do not participate in relationships?).

Configuration consists in establishing the properties of objects and fixing the relationships that arise between them. An example of configuring an image of some subject area is a semantic network, which may change over time.

4.1. Configuration

An object or phenomenon from the subject area can be viewed in a certain field of view FoV, which can change. The notion of the field of view is quite neutral, it is important that these are the conditions under which the perceived object or phenomenon manifests its *properties* (is there an object without properties?). If we talk about the reflection of an object or phenomenon, then we are dealing with one or another image of the field of view, a reflection of its type RoV. This arises

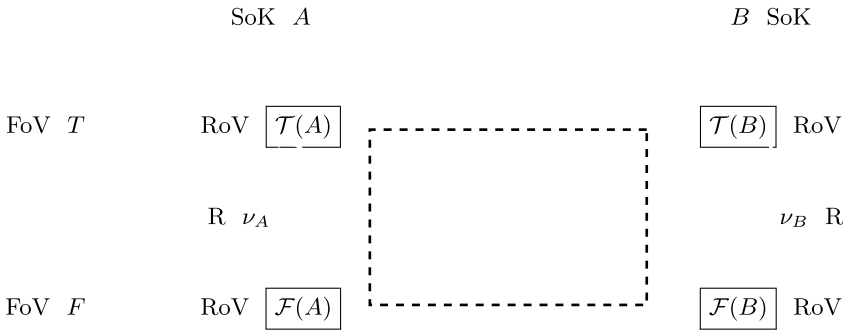

Fig. 1. Configuration. (Abbreviations: *SoK* — Stage of Knowledge, *FoV* — Field of View, *RoV* — Reflection of View, *R* — doing Reflecting; A, B — stages, T, F — properties, $\mathcal{T}, \mathcal{F}$ — functors, ν — natural transformation.).

during the implementation of the act of reflection R. We assume that each stage of knowledge SoK is associated with a field of view and reflection of its appearance. Thus, in the 'later' stages, the knowledge of the reflection of an object or phenomenon may change. What does all of this mean? Exactly that in this explanatory system 'stages' appeared as parameters. Let us look for a suitable semantic network.

4.2. The knowledge graph nodes

Let us put forward the *assumption* that a semantic network is based on the category of functors with natural transformations.

Let S be the category of all sets and arbitrary functions. Take a *contravariant* functor $\mathcal{F}$ in the category C. This is the mapping $\mathcal{F} : A \to S$, which maps each domain A in C to the set $\mathcal{F}(A)$ and with each mapping $f : B \to A$ from C associates function $\mathcal{F}(f) : \mathcal{F}(A) \to \mathcal{F}(B)$ (notice the change in order!) such that

$$\begin{array}{rcl} \mathcal{F}(1_A) & = & 1_{\mathcal{F}(A)} \text{ u} \\ \mathcal{F}(f \circ e) & = & \mathcal{F}(e) \circ \mathcal{F}(f), \end{array} \tag{1}$$

provided that $f : B \to A$ and $e : C \to B$ in C.

We need a transformation representing the transformation of functors. Take two contravariant functors $\mathcal{T}$ and $\mathcal{F}$. The transformations $\nu : \mathcal{T} \to \mathcal{F}$ are called 'natural transformations' of functors. This means that each domain A in C has an associated function $\nu_A : \mathcal{T}_A \to \mathcal{F}_A$ such that If $f : B \to A$ from C, then $\nu_A \circ \mathcal{F}(f) = \mathcal{T}(f) \circ \nu_B$ to S.

The first question that arises is what to place in the *nodes* of the semantic network. In Fig. 1, the solid border rectangles enclose the 4 actual nodes of the network, while the central rectangle with a dashed border is left as the place where the *edges* will be.

Let properties T, F of some individual-object be in the field of view of FoV, they correspond to reflection RoV, to which the functors $\mathcal{T}, \mathcal{F}$ are associated, respectively, and with respect to the stages (A determines the 'actual stage', B is a variable). Nodes are generated in the target semantic network

$$\{\mathcal{T}(A), \mathcal{T}(B); \mathcal{F}(A), \mathcal{F}(B)\}$$

Note that this figure does not contain information about the development of stages from A to B.

4.3. The knowledge graph links

If the semantic network needs information about relationships between stages or reflections of object properties, then mappings are needed that correspond to edges. In the category C we associate the development of stages with the function $f : B \to A$ (the stages 'evolve' from A to B!), and with the change of properties we associate the function $g : T \to F$ (properties 'evolve' from T to F). Fig. 2 shows a semantic network with nodes and edges, and at the same time it is a commutative diagram in the category of functors, which is a strong mathematical argument.

4.4. Fine configuration

For every T in C we set

$$H_T(A) = \{h | h : A \to T\} \tag{2}$$

And if $f : B \to A$ in C then mapping $H_T(f)$ takes $h \in H_T(A)$ and transforms it to $h \circ f \in H_T(B)$. It is easy to show that H_T is a contravariant functor. It is called the *representative* functor corresponding to T.

Let $g : T \to F$ in C. There is a natural transformation $H_g : H_T \to H_F$; this is because $h \in H_T(A)$ can be naturally mapped to $g \circ h \in H_F(A)$. The composite mapping for $f : B \to A$ to C takes $h \in H_T(A)$ and transforms it to $g \circ h \circ f \in H_F(B)$ and there are two equivalent ways to compute it, by composition associativity. On this basis, the resulting diagram commutes. It is known that the only natural transformations $\nu : H_T \to H_F$ have the form H_g for some $g : T \to F$ in C. Fig. 3 shows a fine configuration of the semantic net, it is obtained from the previous Fig. 2 by substituting H_T for $\mathcal{T}$ and H_F instead of $\mathcal{F}$.

4.5. Fine configuration with channeling

Fig. 4 shows the appearance of the information channel $H_g(f)$, which follows from the commutativity of the representing diagram. Here C_f, C_f^g are *displaced* concepts.

Assume the following story (Schubert, 1976): One day Johnny caught a tadpole. He called it 'Hugo' and kept it in a jar. A year later, when Johnny set Hugo free, Hugo was a frog. Question: was Hugo a tadpole or a frog when Johnny caught it? In a semantic net in which 'tadpole' was predicated of Hugo in the permanent mode, this question is not decidable; Hugo might already have been a frog when Johnny caught it. Thus we need to record that Hugo was a tadpole at the time Johnny caught it.

Now this story is easy to read, noting that T are tadpoles, F are frogs, g is the Hugo morph, f is the evolvent along which stages A, B of knowledge unfold, C_f^g – morphed tadpoles, $H_g(f)$ – channel that opens the possibility of morphing. This means that such stories can be 'read' in this configuration.

5. Discussion

As a result, the paper presents a synthesis of the main provisions of the method of constructing the configurator by V. Lefebvre, computational thinking and the basic ideas of conceptual mathematics (category theory).

The configurator basically answers the question of *what* to represent. Configuring involves building a special structural model of the configurator. With the help of it, multi-subject knowledge is moved in a single theoretical representation of some complex, systemic object. The configurator serves as an ideal image of the structure of an object,

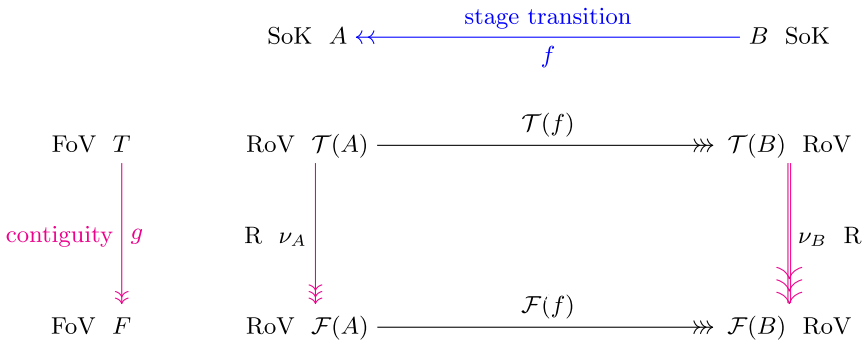

Fig. 2. Configurator.

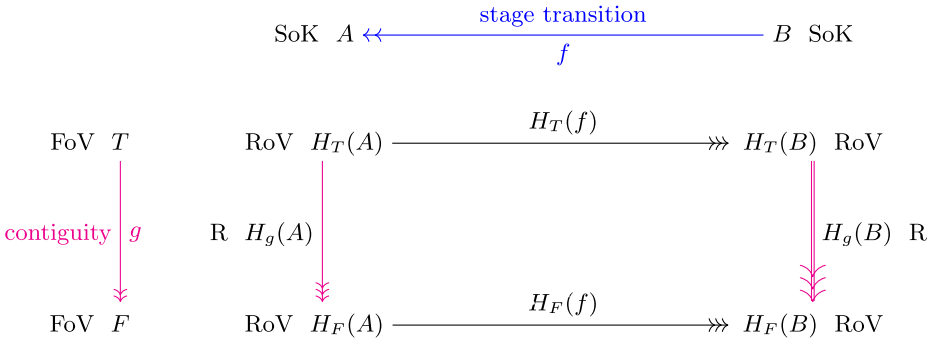

Fig. 3. Fine configurator.

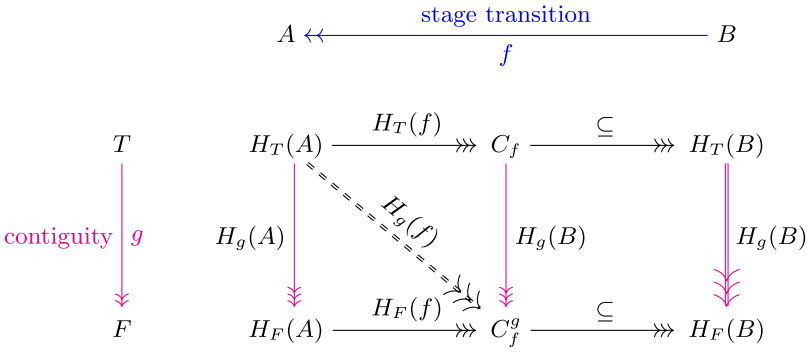

Fig. 4. Channeled fine configurator.

explains and substantiates existing knowledge, shows which aspects of the object are its projections. In other words, the object is *known* precisely through families of projections.

Computational thinking largely answers the question of *how* to represent: in short, it provides a systematic alternation of object projection families and plays the role of a tool.

Whether computational thinking is used in computing or another subject area, the process of computational thinking can be broken down into four parts or steps.

1. Decomposition

The first step in computational thinking is decomposition. This is reflected in Fig. 1. While it may go by different names depending on the school of thought, the basic process is the same: to solve a complex problem, we must first break it down into smaller, more manageable pieces. Decomposition is an important part of computational thinking because it helps make the problem more manageable (you have probably heard the expression 'the best way to eat an elephant is one bite at a time'). It also helps problem solvers better define and understand the problem being solved, allowing them to simplify the problem through pattern recognition and abstraction.

2. Pattern recognition

Part of computational thinking is also pattern recognition. It is the process of identifying patterns or connections between different parts of a larger problem. This is reflected in Fig. 2. The goal of pattern recognition is to simplify the problem even further by discovering where details may be similar or different, as well as creating a continuous understanding of a more complex problem.

3. Abstraction

Abstraction is the process of extracting the most relevant information from each decayed task. This is reflected in Fig. 3. It helps to define or summarize what exactly needs to be done to solve the problem as a whole. This step in the computational thinking process helps to determine how these important details can be used to solve other areas of the same problem.

4. Algorithmic thinking

The final component of computational thinking is algorithmic thinking. This phase more closely corresponds to Fig. 4 It is the process of determining a step-by-step solution to a problem that can be replicated for a predictable and reliable outcome. For the modern definition of computational thinking related to computing, this decision would

be a step-by-step process that would be completed by the computer. However, this process can also be completed partially or completely by humans.

Conceptual mathematics – category theory, – plays the role of a shell or an envelope in which everything is packed. It is entrusted with the role of implementing semantic configuration. A semantic map is generated, the nodes of which are functors, and the links are mappings between functors, or, in the terminology of category theory, natural transformations.

Now this is not so difficult to bring-in the Figs. 1, 2, 3, and 4 the 'biologically inspired' interpretation in terms of tadpole and its morphing into frog (remember, that this a *pattern* giving rise to other applications!).

It is important that all the objects in nodes are to be understood as *processes* — in a harmony with the assumptions of computational thinking. For further possible extensions of the given approach see Wolfengagen, Ismailova, and Kosikov (2024) with an example of solving the task of process entanglement.

6. Conclusion

In this paper, the issue of the role and place of configuration was discussed from the standpoint of system analysis. The problem is not as simple as it might seem. On the one hand, the term itself is considered well-known, but there is no exact definition in the literature. On the other hand, this gives rise to a very broad and rather free interpretation of this term, which is strongly dependent on the applied field of knowledge or engineering.

1. An attempt was made to translate the broad discussion about configuration into the context of software engineering and cognitive modeling. Moreover, in recent years there has been an increased interest in cognitive search on the Web with the development of semantic search mechanisms and semantic configuration that is emerging before our eyes.

2. In the field of programming, configuration is discussed in the context of the programming field — the establishment of an abstract syntax of the language, the choice of one or another semantics that best reflects the features of the use of software environment constructs, the formation of an effective computational model, the construction of programming technology, and, if necessary, the involvement of a unified database. The important issue of database configuration is beyond the scope of this paper. All of these steps in the design of a software system reflect the introduction of configuration aspects into software engineering, although it is not directly called that.

3. Particular attention is paid to the requirements for the computational model, since it is the model that supports the configuration notion chosen when designing the system.

4. At present, a unified idea of configuration has not yet been developed, individual model solutions are mathematically diverse and often have cumbersome descriptions of a highly specialized nature.

5. The idea of the configuration model, introduced by V. Lefebvre, is used in solving problems of semantic modeling and binding to cognitive stages. At the same time, the apparatus of conceptual mathematics makes it possible to use as an abstraction a functor category with natural transformations that reflect semantic transformers used in practice of cognitive search systems.

6. The developed notion of configuration is made in accordance with the main criteria of computational thinking and implements the semantic model of configuration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

Allen, J. F., & Frisch, A. M. (1982). What's in a semantic network? In *Proceedings of the 20th annual meeting on Association for Computational Linguistics* (pp. 19–27). USA: Association for Computational Linguistics, http://dx.doi.org/10.3115/981251.981256.

Balogova, K., & Brumby, D. (2022). How do you zoom?: A survey study of how users configure video-conference tools for online meetings. In *CHIWORK 2022, 2022 Symposium on human-computer interaction for work*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3533406.3533408.

Barbera, C., Lieto, A., & Pozzato, G. L. (2022). An ontology-based tool for dynamic generation, classification and recommendation of novel contents in online libraries. In R. Damiano, S. Ferilli, M. Striani, & G. Silvello (Eds.), *CEUR workshop proceedings: vol. 3286, Proceedings of the 1st workshop on artificial intelligence for cultural heritage, ai4ch 2022, co-located with the 21st international conference of the Italian association for artificial intelligence* (pp. 1–12). CEUR-WS.org, URL https://ceur-ws.org/Vol-3286/01_paper.pdf.

Brachman, R. J. (1977). What's in a concept: Structural foundations for semantic networks. *International Journal of Man-Machine Studies*, *9*(2), 127–152. http://dx.doi.org/10.1016/S0020-7373(77)80017-5.

Brachman, R. J. (1979). On the epistemological status of semantic networks. In N. V. Findler (Ed.), *Associative networks* (pp. 3–50). Academic Press, http://dx.doi.org/10.1016/B978-0-12-256380-5.50007-4, URL http://www.sciencedirect.com/science/article/pii/B9780122563805500074.

Cobanli, O. M. (2011). Integrating end-users to the design process through design competitions. In *Proceedings of the 2011 conference on designing pleasurable products and interfaces*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2347504.2347568.

de Jong, P. (1988). The Ubik configurator: A fusion of messages, daemons, and rules. In *OOPSLA/ECOOP '88, Proceedings of the 1988 ACM SIGPLAN workshop on object-based concurrent programming* (pp. 197–199). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/67386.67443.

de Souza, M., & Ritt, M. (2022). Improved regression models for algorithm configuration. In *Proceedings of the genetic and evolutionary computation conference* (pp. 222–231). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3512290.3528750.

de Souza, M., Ritt, M., & López-Ibáñez, M. (2022). Efficient configuration of optimization algorithms. In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 17–18). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3520304.3534078.

Fadhlillah, H. S., Feichtinger, K., Gutiérrez Fernández, A. M., & Rabiser, R. (2023). Dynamic product configuration user interface: A vision motivated by the cyber-physical production systems domain. In *Proceedings of the 17th international working conference on variability modelling of software-intensive systems* (pp. 88–90). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3571788.3571803.

Falk, J., Frauenberger, C., & Kannabiran, G. (2022). How shortening or lengthening design processes configure decision making. In *Nordic human-computer interaction conference*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3546155.3547726.

Gao, Y., Wang, P., Liu, L., Zhang, C., & Ma, H. (2023). Configure your federation: Hierarchical attention-enhanced meta-learning network for personalized federated learning. *ACM Transactions on Intelligent Systems and Technology*, http://dx.doi.org/10.1145/3591362, Just Accepted.

Gentile, M., & Lieto, A. (2022). The role of mental rotation in TetrisTM gameplay: An ACT-R computational cognitive model. *Cognitive Systems Research*, *73*(C), 1–11. http://dx.doi.org/10.1016/j.cogsys.2021.12.005.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2021a). Cognitive system to clarify the semantic vulnerability and destructive substitutions. *Procedia Computer Science*, *190*, 341–360. http://dx.doi.org/10.1016/j.procs.2021.06.044, 2020 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: Eleventh Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050921012898.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2021b). Equalities between combinators to evaluate expressions. *Procedia Computer Science*, *190*, 332–340. http://dx.doi.org/10.1016/j.procs.2021.06.058, 2020 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: Eleventh Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050921013053.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2021c). A mathematical model of the feature variability. *Procedia Computer Science*, *190*, 312–316. http://dx.doi.org/10.1016/j.procs.2021.06.041, 2020 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: Eleventh Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050921012850.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2021d). A semantic model for indexing in the hidden web. *Procedia Computer Science*, *190*, 324–331. http://dx.doi.org/10.1016/j.procs.2021.06.043, 2020 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: Eleventh Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050921012874.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2022a). Applicative approach to construe a computational model of concepts and individuals. *Procedia Computer Science*, *213*, 463–470. http://dx.doi.org/10.1016/j.procs.2022.11.092, 2022 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: The 13th Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050922017835.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2022b). Conceptual hanger for displaced concepts: a framework for information processes variability. *Procedia Computer Science*, *213*, 588–595. http://dx.doi.org/10.1016/j.procs.2022.11.109, 2022 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: The 13th Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050922018087.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2022c). Elements of semantic analysis based on lambda-calculus. *Procedia Computer Science*, *213*, 471–476. http://dx.doi.org/10.1016/j.procs.2022.11.093, 2022 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: The 13th Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050922017847.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2022d). Lambda-calculus, combinators and applicative computational technologies. *Cognitive Systems Research*, *76*, 93–100. http://dx.doi.org/10.1016/j.cogsys.2022.10.002, URL https://www.sciencedirect.com/science/article/pii/S1389041722000468.

Ismailova, L., Wolfengagen, V., & Kosikov, S. (2022e). A prototype system for supporting a network of information graphs with the ability to assess the nature of the subject's knowledge. *Procedia Computer Science*, *213*, 16–20. http://dx.doi.org/10.1016/j.procs.2022.11.033, 2022 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: The 13th Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050922017240.

Ismailova, L., Wolfengagen, V., Kosikov, S., & Andronov, S. I. (2023). The applicative approach to the synthesis of a data structure with the given combinatory characteristic. *Cognitive Systems Research*, *77*, 88–93. http://dx.doi.org/10.1016/j.cogsys.2022.10.010, URL https://www.sciencedirect.com/science/article/pii/S1389041722000602.

Jaeger, T. (2016). Configuring software and systems for defense-in-depth. In *Proceedings of the 2016 ACM workshop on automated decision making for active cyber defense* (p. 1). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2994475.2994483.

Kosikov, S., Ismailova, L., & Wolfengagen, V. (2021). Data enrichment in the information graphs environment based on a specialized architecture of information channels. *Procedia Computer Science*, *190*, 492–499. http://dx.doi.org/10.1016/j.procs.2021.07.001, 2020 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: Eleventh Annual Meeting of the BICA Society. URL https://www.sciencedirect.com/science/article/pii/S1877050921013922.

Kucher, M., Balyo, T., & Christensen, N. (2022). Black-box optimization in a configuration system. In *Proceedings of the 26th ACM international systems and software product line conference - volume B* (pp. 229–236). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3503229.3547041.

Lefebvre, V. A. (2001). *Theory and decision library 30, Algebra of conscience: Revised edition with a second part with a new foreword by anatol rapoport* (2nd ed.). Springer Netherlands.

Lefebvre, V., & Smolyan, G. (1968). *New in life, science, technics. Series "Mathematics, cybernetics" 1968, Algebra of conflict, no. 03*. Znanie.

Lieto, A., Pozzato, G. L., Perrone, F., & Chiodino, E. (2019). Knowledge capturing via conceptual reframing: A goal-oriented framework for knowledge invention. In *K-CAP '19, Proceedings of the 10th international conference on knowledge capture* (pp. 109–114). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3360901.3364422.

Liu, P., Tan, B., Lin, Z., Chen, Y., & Wang, F. (2022). Multi-objective optimal configuration method for off-grid integrated energy system. In *ICECC 2022, The 2022 5th international conference on electronics, communications and control engineering* (pp. 86–91). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3531028.3531042.

Mahmud, T., Zhang, D., Gatla, O. R., & Zheng, M. (2022). Understanding configuration dependencies of file systems. In *Proceedings of the 14th ACM Workshop on hot topics in storage and file systems* (pp. 1–8). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3538643.3539756.

McGuinness, D. L., Isbell, C., Parker, M., Patel-Schneider, P., Resnick, L. A., & Welty, C. (1998). A description logic-based configurator on the web. *SIGART Bulletin*, *9*(2), 20–22. http://dx.doi.org/10.1145/1056754.1056756.

Myllärniemi, V., Ylikangas, M., Raatikainen, M., Pääkkö, J., Männistö, T., & Aaltonen, T. (2012). Configurator-as-a-service: Tool support for deriving software architectures at runtime. In *WICSA/ECSA '12, Proceedings of the WICSA/ECSA 2012 companion volume* (pp. 151–158). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2361999.2362031.

Noorian, M., Bagheri, E., & Du, W. (2016). Quality-centric feature model configuration using goal models. In *Proceedings of the 31st annual ACM symposium on applied computing* (pp. 1296–1299). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2851613.2851959.

Raatikainen, M., Tiihonen, J., Männistö, T., Felfernig, A., Stettinger, M., & Samer, R. (2018). Using a feature model configurator for release planning. In *Proceedings of the 22nd international systems and software product line conference - volume 2* (pp. 29–33). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3236405.3236411.

Randrianaina, G. A., Tërnava, X., Khelladi, D. E., & Acher, M. (2022). On the benefits and limits of incremental build of software configurations: An exploratory study. In *Proceedings of the 44th international conference on software engineering* (pp. 1584–1596). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3510003.3510190.

Rust, P., Picard, G., & Ramparany, F. (2022). Resilient distributed constraint reasoning to autonomously configure and adapt IoT environments. *ACM Transactions on Internet Technology*, *22*(4), http://dx.doi.org/10.1145/3507907.

Schubert, L. (1976). Extending the expressive power of semantic networks. *Artificial Intelligence*, *7*(2), 163–198. http://dx.doi.org/10.1016/0004-3702(76)90003-5, URL https://www.sciencedirect.com/science/article/pii/0004370276900035.

Scott, D. (1980). Relating theories of the λ-calculus. In J. Hindley, & J. Seldin (Eds.), *To H.B. Curry: Essays on combinatory logic, lambda-calculus and formalism* (pp. 403–450). Berlin: Academic Press.

Somashekar, G., & Kumar, R. (2023). Enhancing the configuration tuning pipeline of large-scale distributed applications using large language models (idea paper). In *ICPE '23 Companion, Companion of the 2023 ACM/SPEC international conference on performance engineering* (pp. 39–44). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3578245.3585032.

Stöckle, P., Wasserer, T., Grobauer, B., & Pretschner, A. (2023). Automated identification of security-relevant configuration settings using NLP. In *Proceedings of the 37th IEEE/ACM international conference on automated software engineering*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3551349.3559499.

Sun, Y., Cho, H., Gray, J., & White, J. (2011). Supporting feature model configuration using a demonstration-based approach. In *Proceedings of the 2nd international workshop on product line approaches in software engineering* (pp. 55–59). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/1985484.1985498.

Tao, T., & Plaisted, D. (2021). Interactive online configurator via Boolean satisfiability modeling. In *IAIT2021, The 12th international conference on advances in information technology*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3468784.3468785.

Uta, M., Felfernig, A., Le, V.-M., Popescu, A., Tran, T. N. T., & Helic, D. (2021). Evaluating recommender systems in feature model configuration. In *Proceedings of the 25th ACM international systems and software product line conference - volume a* (pp. 58–63). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3461001.3471144.

Vandevelde, S., Callewaert, B., & Vennekens, J. (2022). Interactive feature modeling with background knowledge for validation and configuration. In *Proceedings of the 26th ACM international systems and software product line conference - volume B* (pp. 209–216). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3503229.3547039.

Wang, S., Hoffmann, H., & Lu, S. (2022). AgileCtrl: A self-adaptive framework for configuration tuning. In *ESEC/FSE 2022, Proceedings of the 30th ACM joint European software engineering conference and symposium on the foundations of software engineering* (pp. 459–471). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3540250.3549136.

Wolfengagen, V., Ismailova, L., & Kosikov, S. (2022). Cognitive technology to capture deep computational concepts with combinators. *Cognitive Systems Research*, *71*, 9–23. http://dx.doi.org/10.1016/j.cogsys.2021.10.001, URL https://www.sciencedirect.com/science/article/pii/S1389041721000747.

Wolfengagen, V., Ismailova, L., & Kosikov, S. (2024). Computationally inspired cognitive modeling. *Cognitive Systems Research*, *83*, 1–10. http://dx.doi.org/10.1016/j.cogsys.2023.101175, URL https://www.sciencedirect.com/science/article/pii/S1389041723001092.

Wolfengagen, V., Ismailova, L., Kosikov, S., & Babushkin, D. (2021). Modeling spread, interlace and interchange of information processes with variable domains. *Cognitive Systems Research*, *66*, 21–29. http://dx.doi.org/10.1016/j.cogsys.2020.10.016, URL https://www.sciencedirect.com/science/article/pii/S1389041720300905.

Wolfengagen, V., Ismailova, L., Kosikov, S., & Dohrn, S. (2022). Cognitive system for traversing the possible worlds with individual information processes. In V. V. Klimov, & D. J. Kelley (Eds.), *Biologically inspired cognitive architectures 2021* (pp. 596–601). Cham: Springer International Publishing.

Woods, W. A. (1975). What's in a link: Foundations for semantic networks. In D. G. Bobrow, & A. Collins (Eds.), *Representation and understanding* (pp. 35–82). San Diego: Morgan Kaufmann, http://dx.doi.org/10.1016/B978-0-12-108550-6.50007-0, URL https://www.sciencedirect.com/science/article/pii/B9780121085506500070.

Woods, W. A. (1997). *Conceptual indexing: A better way to organize knowledge*: *Tech. Rep. SMLI TR-97-61*, Sun Microsystems, Inc., URL http://www.smli.com/technical-reports/1997/abstract-61.html.

Zhang, C., Chen, B., Hu, J., Peng, X., & Zhao, W. (2023). BuildSonic: Detecting and repairing performance-related configuration smells for continuous integration builds. In *Proceedings of the 37th IEEE/ACM international conference on automated software engineering*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3551349.3556923.

Zhang, Z., Klees, G., Wang, E., Hicks, M., & Wei, S. (2023). Fuzzing configurations of program options. *ACM Transactions on Software Engineering and Methodology*, *32*(2), http://dx.doi.org/10.1145/3580597.