Definitions to the Configuration Problem

Armin Wurzinger, 1528532

***Abstract*—The field of configuration and affiliated configuration problems has received a lot of attention in the last years. Different approaches have been developed in order to solve such configuration problems. Definitions of some common terms used in documents dealing with configuration can vary, making it harder to understand them. Therefore some publications regarding the configuration problem were examined for finding terms which were used ambiguously. In this paper we give an overview of the results of this research, and we disambiguate those definitions used inconclusively. Own definitions of ambigous terms are stated, if necessary. Furthermore, we conclude which existing definitions are appropriate for general use. Additionally, it is elaborated when its appropriate to use specific definitions instead of general ones.**

I. Introduction

Configuration is an important activity in the industry, where systems for configuration are being developed since more than 20 years [1]. While mass production is an effective way of reducing the costs for the industry, customers want products that meet their individual needs. This has led to the development of variable products, which are configured out of mass-produced parts.

With the upcoming of computers, another huge market that requires highly customizable products emerged: software. Software can usually be customized to run on various kinds of platforms and to suit each user's needs. Usually not every possible customization will work. Therefore, technologies have been researched that aid in the creation of customized, well-working products.

In this paper, we studied previous publications dealing with technologies to solve configuration-related problems. The terms defined by those publications are sometimes used inconclusively, therefore these defintions were disambiguated. Then we discuss why there are ambiguous definitions, and own definitions based on the examined ones are being provided, if necessary. Furthermore, it will be elaborated when its appropriate to use other definitions instead of the ones given in this research.

II. Configuration from the general point of view

In order to get a better understanding of *configuration*, we examine past attempts to define the term. Several papers dealing with configuration were studied, discussing some notable, different definitions.

Generally, when speaking of *configuration* the topic is seen from different points of view, overloading the term. In this section the term will be discussed from a general, abstract point of view. Another widely used interpretation of the term configuration is *configuration solution*, which we examine in Section III. The exact meaning is usually apparent due to the context.

A. Previous Definitions

The first definition we inspected is stated in [2];

> "Configuration is a special case of design activity with two key features:
>
> - The artifact being configured is assembled from instances of a fixed set of well-defined component types.
> - Components interact with each other in predefined ways."

This definition refers to *configuration* as being an activity. This activity involves the creation of artifacts. *Artifact* refers to the outcome of such activity, and does not restrict what can be configured. Compared to the term product the term artifact is more general; from an industrial point of view a product is often seen as something that is being assembled in a factory. We can see that in this definition artifacts are being assembled out of a fixed set of component types. It will be shown later that this set must not necessarily be fixed. It is kept very generally and informally, so it applies to all kind of various tasks besides the creation of technical products out of components. For instance, cooking can be seen as a *configuration* activity, where well-defined ingredients are being combined (components). Ingredients used in cooking will interact with each other in a predefined way.

Definitions in other publications are often based on this definition, but may either restrict or expand it. In [3] a second definition is given, which is more specific and tailored towards product configuration. *Product configuration* is a common activity that occurs in the industry. Products created by companies are often assembled out of existing components, for example, the *configuration* of a computer using components like a CPU or a mainboard is such activity;

> "Configuration is a design activity, where the configured product is built from a predefined set of component types that can be parameterized and interconnected on pre-enumerated connection points. Additional constraints are used to restrict the number of legal product constellations."

Opposed to the definition in [2] this interpretation does not simply refer to components, but more abstract component types which can be parameterized in order to represent an actual component. The way components can be combined is being restricted through pre-enumerated connection points (for instance, USB ports used in computers) and further constraints. In Section IV-B, it can be seen that this definition is more closely tied to *configuration problems*.

A third definition for configuration is given by [4]. According to this paper, the term is generally defined as task, with product configuration being an activity focused on solving this task;

> "Configuration as a task can be roughly defined as the problem of designing a product using a set of predefined components while taking into account a set of restrictions on how the components can be combined.
> The term product configuration is used to denote the routine engineering activity of this type in the sales-order-delivery process."

It can be observed that in this definition *product configuration* is being described as the engineering activity of solving a *configuration task*, as opposed to configuration being a design activity.

B. Discussion

In order to define *configuration* from a general point of view, the definition should be kept generally applicable. Therefore, from the three examined publications the definition given by [2] is the most appropriate. Furthermore several additional papers on this topic were inspected, like the proceedings of the workshop on configuration 2011 [6]. In this proceedings *configuration* is generally seen as an activity that assembles configuration artifacts out of components, which interact with each other.

However, according to [7] there are relevant problems where the set of components is not predefined. It can change dynamically during a configuration process. This issue is also being pointed out in [8], stating that predefined components usually refer to either a standard component, a standard component with variants or a parametric component, i.e. a component for which one or more attributes vary continuously. Furthermore, it is noted that this is usually enough to approximate common real world problems in the area of *product configuration*. Therefore, this definition is not enough for a very general definition of a *configuration*.

In [7] it is also criticized that the way components interact with each other are usually modeled as simple binary compatibility relations. The definition given in [2] however does not assume this. It simply states that the ways components interact with each other are predefined, not that they can be modeled with binary relations. More specific, restricted definitions, as seen in [3] and [4], are not suited for defining *configuration* in general, but to define distinct specialized activities such as *product configuration*. In [4] *product configuration* is being seen as a routine engineering activity. This does not apply for *configuration* in general, which can be just a general design activity.

C. Definition of the Author

Based on this examination, we define the term *configuration* from a general point of view as a less strict version of the definition by [2];

> Configuration is a special case of design activity with two key features:
> - The artifact being configured is assembled from instances of a not necessarily fixed set of well-defined component types.
> - Components interact with each other in predefined ways.

III. Configuration Solution

When speaking of *configuration*, one could refer to a solution for solving a given *configuration problem*. In this section past attempts to define the term from this perspective were examined. Such definitions are often formally specified, tailored towards a specific way of solving *configuration problems*.

A. Previous Definitions

The first definition examined is stated in [9];

> "A configuration (solution) S for a given configuration task $(V, D, CKB \cup REQ)$ is represented by an assignment $S = ins(v_1), ins(v_2), ..., ins(v_k)$ where
> - $ins(v_i) \in dom(v_i)$ and
> - S is complete and consistent with the constraints in $CKB \cup REQ$"

V is a set of finite domain variables, D represents variable domains. This approach is based on a constraint satisfaction problem, and V and D are abstractions referring to the set of components and their relations. There are several different approaches to solve configuration problem, where knowledge-based configuration systems form the foundation for a thriving industry [10]. Other approaches are rule-based or model-based configuration systems [10]. According to [9], *configuration* is typically knowledge-based, therefore this definition refers to a knowledge base CKB which is unified with the requirements REQ in order to generate a *configuration solution*.

However, it is not generally the case that a *configuration* can be expressed as being knowledge-based. Furthermore it is stated that this approach of representing a *configuration task* is primarily used in situations where all variables of the problem definition are relevant for the solution. Therefore, this definition does not seem appropriate to define a *configuration solution* in general, but can be used when dealing with such specific approaches.

In [11], a *configuration solution* (CS) is defined as

> "$CS := I, V, S$ where
> - I is a set of individuals, which are instances of components
> - V is a set of values, which are assigned to properties of individuals
> - S is a Boolean function, $S : Cr, R \rightarrow T, F$. The assignment of I and V makes the expressions Cr and R true
> - Cr a set of constraints imposed on components due to technical and economical factors
> - R a set of customer requirements, usually specified as constraints"

Again, it can be seen that this definition of a *configuration solution* is specific for the way of solving it. Compared to [9], there is no knowledge-base involved directly. The involvement of first order logic shows that this kind of interpretation would be used for example when treating a *configuration problem* as a constraint satisfaction problem. Intuitively, it can be described as trying to find components I, having assigned properties V which do not violate the constraints Cr. These

constraints represent technical or economical factors and a representation of a set of customer requirements. As this definition keeps both technical factors and customer needs in mind, it is appropriate to use it in a *product configuration* scenario.

A third, more general definition of a *configuration solution* is given in [12];

> "A configuration is a set of components and a description of the connections between the components in the set."

Compared to the two definitions that have been examined it can be clearly seen that this definition is not specific to a certain way of solving a *configuration problem*, nor does it state if a configuration problem is being solved at all. Due to its informal nature, without the use of first-order logic people lacking some logic knowledge can understand it. Comparing this to the definition of *configuration* from a general point of view, stated in Section II, we can see that this interpretation of a *configuration solution* basically fits being the outcome of such activity. It can be used to refer to all kinds of configuration activities, such as assembling a computer out of predefined parts.

B. Discussion

As seen in both [12] and [11], using *configuration* instead of *configuration solution* is common practice. It is usually apparent from the context if *configuration solution* is being referred to as *configuration*. Furthermore, the term *configuration solution* can be used instead of just *configuration* if one wants to be completely disambiguous.

Definitions of a *configuration solution* can be different depending on the approach which is being used in order to solve a *configuration problem*. If a paper presents a specific way of solving *configuration problems*, such interpretations can be used. For a general definition of a *configuration solution*, the definition given in [12] are approriate.

However, the interpretation assumes that components are being connected in some way. This contradicts with the general definition of *configuration* given in section II-A, referring to cooking being a *configuration* activity as an example. It is not obvious how the connection of ingredients could be described.

C. Definition of the Author

Based on the interpretation in [12], we give an own, general definition of a *configuration solution*:

> A configuration is a set of components and a description of the interactions between the components in the set.

IV. Defining Configuration Problem

When speaking of a configuration problem, it is often not clear what exactly is implied. We examined several papers dealing with configuration and it showed that the definition of a configuration problem varies. In Section IV-A previous interpretations of a *configuration problem* were examined. Then, in Section IV-B we give an own general definition of a *configuration problem*. Furthermore it is discussed when it is appropriate to use interpretations examined in section IV-A.

A. Previous Definitions

In this section past attempts to define the term *configuration problem* are being examined. It can be observed that such definitions are often formally specified towards a specific approach of solving them. A first definition of a *configuration problem* is given in [10];

> "In general we assume a configuration problem is described by two sets of logical sentences: DD, the domain description, and SRS, the particular system requirements which specify an individual configuration problem instance.
> A configuration then is a triple $(COMPS, CONNS, ATTRS)$ of components, connections, and attributes. $COMPS$ is a set of ground literals $type(c, t)$, $CONNS$ is a set of ground literals $conn(c_1, p_1, c_2, p_2)$, where c_1, c_2 are components and p_1, p_2 are ports, and $ATTRS$ is a set of ground literals of the form $val(c, a, v)$ where c is a component, a is an attribute name, and v is the value of the attribute."

The first logical sentence mentioned in the definition, DD, corresponds to the definition of *configuration* given in Section II-C. It basically describes the components and how they can interact with each other.

SRS is an abstraction for requirements on the artifact that gets configured. Such requirements could be used to model the needs of a specific customer in the case of product configuration. Referring to the example of cooking as a configuration activity, as stated in Section II-A, such requirements could model which ingredients a person does not like. From a general point of view the definition of a domain description and particular requirements to describe a *configuration problem* seems plausible. The problem that arises is that none, one or more valid *configuration solutions* have to be calculated out of the given DD without violating requirements specified in SRS. This definition can also be partly applied to the interpretation of a *configuration task* given by [12] and cited in Section V, where it suits the description of the input. The criteria (A) from the definition resembles the DD, whereas the criterias (B) and (C) can be seen as the SRS.

In [3], a more advanced definition of a configuration problem is given compared to the one in [10];

> "In general we assume a confguration problem is described by a triple $(DD, SRS, CONL)$ where DD and SRS are sets of logical sentences and $CONL$ is a set of predicate symbols.
> DD represents the domain description or confguration knowledge base, and SRS specifes the particular system requirements, which define an individual configuration problem instance. A configuration $CONF$ is described by a set of positive ground literals whose predicate symbols are in the set $CONL$."

It can be seen that this interpretation is a more formal version of the definition given by [10]. This definition introduces a set of predicate symbols $CONL$. An example is given where this set could include for instance the predicates $type$, $conn$, and val. As we can see, these predicates are also used when

defining a *configuration problem* in [10], cited in this section. In this definition it has been assumed that these predicates exist, without specifying them. Therefore, the definition in [3] is more precise, but also a bit harder to understand.

A more general definition of a configuration problem can be found in [14];

> "The product configuration problem is defined as characterised by two constituents:
>
> 1) A catalogue which describes the generic components in terms of their functional and technical properties and the relationship between both
> 2) User requirements and user preferences about the functional characteristics of the desired configuration"

This interpretation of a *configuration problem* is defined in a less formal way without using first-order logic. Instead, it is written from a general point of view. Compared to the previous two definitions, it can be seen that the first constituent corresponds to DD of the previous definition, and the second constituent resembles the requirement specification SRS.

B. Review of the examined previous definitions

As seen in Section IV-A, all three examined definitions of the term *configuration problem* are compatible with each other, but their generality, or level of abstraction differs. Therefore, none of them can be considered invalid and all of them provide a suitable definition of *configuration problem*. It depends on the context and the target audience which interpretation should be used. There is a link between a *configuration problem*'s representation and the algorithm, as stated in [2];

> "Like any problem-solving activity, solving a configuration task implies two different steps. First, we need to represent the problem. Second, we need algorithms that, based on the problem representation, will produce a solution."

As we can see, such algorithms are based on the specification of a *configuration problem*. Therefore, if seen from the reverse perspective, an (existing) algorithm already implies a certain way of representing the problem. Due to their use of first-order logic, the definitions in [3] and [10] are a good choice when working in the field of solving *configuration problems*. They can be used when a *configuration problem* is modeled as a constraint satisfaction problem for instance.

In case one wants to use a general definition of configuration problem, targeting an audience without some knowledge in the fields of logic and configuration, the definition given in [14] seems appropriate. This interpretation also suits to the definitions of *configuration* and *configuration solution* stated in this paper, which are defined in Sections II-C and III-C. In order to clearly distinguish the term *configuration problem* from the activity of *configuration* in general, it can be seen that in such problem different requirements are involved. A *configuration solution* is the outcome of solving a *configuration problem* according to this research.

V. Distinguishing Configuration Task

When examining several papers dealing with *configuration*, we saw that the terms *configuration task* and *configuration problem* are sometimes used interchangeably. It is often not clear if the terms *configuration task* and *configuration problem* actually have different meanings, therefore we distinguish them in this section. As it can be seen in [9], those terms are often being used mutually;

> "Based on this definition of a CSP, we can now introduce the definition of a configuration task (problem) and a corresponding configuration (solution)."

The term *configuration task* is then being defined as a constraint satisfaction problem, so formulated as a problem and not as a task. However, in [2] it is stated that a *configuration task* encloses a *configuration problem* and a way of solving it.

A. Previous Definitions

A first previous definition of configuration task is given in [5];

> "Configuration is the task of composing a customized system out of generic components."

In this definition the term *system* is being mentioned, meaning a group of related components that work together. In this context it can be seen as a more general interpretation of a product. However, this definition seems to be inappropriate since a *configuration task* does not refer to the activity of *configuration*. The difference in the semantics of these two words is that an activity refers to something that happens in general, whereas a task is more work-focused. Therefore, it should also be named *configuration task* to be precise. Furthermore, the terms *configuration task* and *(configuration) problem* are interchangeable, although they do not refer to the same. The definition of [5], and also the definition of product configuration as a design activity in Section II-A are clear examples where terms are being used disambigiously, which can be confusing.

Another interpretation is given in [12], a *configuration task* is described as being the task of finding a *configuration solution* to a given input, and not a representation of a problem;

> "Given:
>
> - (A) a fixed, pre-defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected at that port, and other structural constraints
> - (B) some description of the desired configuration and
> - (C) possibly some criteria for making optimal selections
>
> Build:
> One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the connections between the components in the set, or, detect inconsistencies in the requirements."

Informally, a task consists of a given input (representing a *configuration problem*) and should lead to a *configuration solution* as an output. However, as pointed out in [13] there is one implication within this definition, that makes it unsuitable for general use: It is too closely tied to the field of computer configuration, where the author of that paper is working in. The concept of connecting components using ports is hard to apply when configuring mechanical products. For example, it is not obvious how a gear can be connected to another gear using a port. Similar definitions can be seen in other papers. For instance, in [11], a *configuration task* is also specified as being the task of finding *configuration solutions* to a given input;

> "Given a set of predefined components, the task of product configuration is to find a configuration solution satisfying individual needs of customers without violating any constraints imposed on components due to technical and economical factors."

The input is then modeled formally, and being defined as a *configuration problem* (see section IV-A). Again, in this interpretation the two terms do not mean the same. Furthermore, it also has the problem that it assumes a set of predefined components. In Section II-C it has been concluded that this does not necessarily have to be the case.

The last definition of a *configuration task* being examined is given in [14];

> "A configuration task consists in finding the following answer: 1. One or more configurations that satisfy all requirements and that optimise the preferences if those requirements are consistent. 2. An explanation of failure in the other case."

Similar to the interpretation in [11], this interpretation also states that a *configuration task* consists of finding configurations that satisfy given requirements. Furthermore, it points out an important fact. A *configuration task* will always have an output, but this may not be a *configuration solution*. There can be constellations of requirements on a *configuration* that cannot be fulfilled.

B. Discussion

As we have seen when examining some different definitions of *configuration task*, we saw that the term is often being used with the term *configuration problem* inconclusively. In order to be precise, as seen in Section IV a *configuration problem* can be defined as being a representation of the problem of configuring artifacts. Therefore, a *configuration problem* refers to the description of problem instance, and is not a *configuration task*.

A *configuration task* refers to the task of finding a *configuration solution* to a given *configuration problem*. It may also happen that there is no possible *configuration solution* for a given input. These two terms should not be used interchangeably, as they do not mean the same. In case they are used mutually, it can be usually inferred from the context wheter the whole task or just the problem representation is meant.

The two terms *configuration*, referring to an activity, and *configuration task*, referring to a specific task, are also sometimes being used inconclusively, as seen in [5]. As an activity is more general than a task, those two terms should be used distinctively.

VI. Conclusion

The field of configuration includes colloquial approaches and applications. There is a number of papers presenting various related topics in this field. Therefore the terminology used throughout these publications varies. Whereas some publications use more general definitions, others give interpretations which are usually tailored to specific ways of approaching configuration. In order to disambiguate commonly used terms in the field of configuration, we provided several general definitions of *configuration*, *configuration solution*, *configuration task* and *configuration problem*. *Configuration* from a general point of view refers to the activity of configuring artifacts out of predefined components which interact which each other in some way. The term *configuration* is also often overloaded when actually a *configuration solution* is meant. A *configuration solution* is the result of a *configuration task*. The term *configuration task* refers to the task of solving a *configuration problem*. Such a *configuration problem* is the representation of a problem where one wants to configure an artifact given certain requirements.

It is perfectly fine to use alternative definitions of these terms in a paper as long as their meaning is not being implied to be known by the reader, but properly specified instead. Such alternative definitions can ease understanding when working with specific ways of solving configuration problems, for instance they could be tailored towards representing a configuration problem as a constraint satisfaction problem. However, it may be still useful if a reference to a more general definition of such term is given, and to state that a specific interpretation is being used. This can help to understand what a publication is all about more easily in future publications.

References

[1] G. Friedrich, D. Jannach, M. Stumptner, and M. Zanker, "Knowledge engineering for configuration systems," in *Knowledge-Based Configuration*, A. Felfernig, , L. Hotz, , C. Bagley, , and J. Tiihonen, Eds. Boston: Morgan Kaufmann, 2014, pp. 139 – 155. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124158177000116

[2] D. Sabin and R. Weigel, "Product configuration frameworks-a survey," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 42–49, Jul 1998.

[3] A. Felfernig, G. Friedrich, and D. Jannach, "Conceptual modeling for configuration of mass-customizable products," *Artificial Intelligence in Engineering*, vol. 15, no. 2, pp. 165 – 176, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0954181001000164

[4] T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen, "Towards a general ontology of configuration," *AI EDAM*, vol. 12, no. 04, pp. 357–372, 1998.

[5] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming*. Elsevier, 2006.

[6] K. Shchekotykhin, D. Jannach, and M. Zanker, Eds., *Workshop on Configuration*, 2011.

[7] J. Yuan, M. Shan, K. Wang, and C. Lin, "Solving ncsp-based configuration problem for mass customization," in *2007 International Conference on Mechatronics and Automation*, Aug 2007, pp. 2247–2251.

[8] F. Salvador and C. Forza, "Configuring products to address the customization-responsiveness squeeze: A survey of management issues and opportunities," *International Journal of Production Economics*, vol. 91, no. 3, pp. 273 – 291, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925527303002895

[9] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, Eds., *Knowledge-Based Configuration*. Boston: Morgan Kaufmann, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124158177000281

[10] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, "Consistency-based diagnosis of configuration knowledge bases," *Artificial Intelligence*, vol. 152, no. 2, pp. 213–234, 2004.

[11] D. Yang, M. Dong, and R. Miao, "Development of a product configuration system with an ontology-based approach," *Computer-Aided Design*, vol. 40, no. 8, pp. 863 – 878, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010448508001061

[12] S. Mittal and F. Frayman, "Towards a generic model of configuraton tasks." in *IJCAI*, vol. 89, 1989, pp. 1395–1401.

[13] D. C. Brown, "Defining configuring," *AI EDAM*, vol. 12, no. 04, pp. 301–305, 1998.

[14] M. S. B. Queva and C. W. Probst, "A framework for constraint-programming based configuration," Ph.D. dissertation, Technical University of Denmark/Danmarks Tekniske Universitet, Department of Informatics and Mathematical Modeling/Institut for Informatik og Matematisk Modellering, 2011.