Configuring from Observed Parts

Lothar Hotz [1]

Abstract. This paper presents a conceptual framework that allows the configuration of aggregates using existing parts observed in reality. Typical problems that can be solved with such an approach are recognition problems that construct aggregates from given observations. One instance of the conceptual framework is given by the system SCENIC, which allows the interpretation of video scenarios showing table-laying scenes.

1 Introduction

Configuration is the construction of aggregates using parts of a certain domain. In a typical configuration schema, starting from a given goal object that represents the aggregate to be configured, this aggregate is decomposed in its parts step by step. Thus, a goal-driven, top-down approach is typically used for constructing a configuration. Furthermore, the parts that should be in the aggregate are previously described in a configuration model. This configuration model describes implicitly all configurations that can be constructed. Typically a knowledge acquisition process creates that configuration model. A knowledge engineer acquires existing components to be configured and formalizes them in the configuration model. In a second step, the configuration process uses the configuration model for configuring new products. During this configuration process the configuration model is seen as fixed. If parts exist, that should be integrated into the aggregate, they are defined in a task specification on the basis of the configuration model before the configuration process starts. No further existing parts are considered, beside those. Thus, during the configuration process the real existing parts are only considered indirectly through the configuration model. The configuration process relays on the fact that the real world is in the state as it was during knowledge acquisition. If new parts are created in the real world during the configuration process, those components are not taken into account. Only if the configuration model is updated (e.g. by a third step - an evolution step), those new parts could be taken into account. Summarizing, we speak of a three-step approach — knowledge acquisition, configuration process, evolution process. The relationship between the configuration model and the real world is only established during knowledge acquisition, the task specification and probably an evolution process, but not during the configuration process. An example of such an approach is given in [6].

In the standard configuration schema, as often in knowledge-based systems, a gap exists between the real world and the configuration model. However, this schema is suitable for non dynamic, technical domains, e.g. hardware configuration [8, 2, 7, 11, 9]. For situations with a number of changes (e.g. software configuration) or even dynamic situations (e.g. interpretation of video scenes) this three-step approach can be improved by taking into account the parts that can be observed during the configuration process.

In this paper, we discuss the *configure-from-observed-parts* problem, or *CofOP-Problem* for short. This problem focuses on configurations that can be derived from real existing and observed parts at a first place. Only in a second step requirements are included to compute the desired configurations. In this case, it can be ensured that the resulting configuration has really counterparts in the real world.

In Section 2, we have a closer look at the CofEP-Problem. Section 3 focuses on structural configuration as a starting point. As the core of the paper, we present a conceptual framework (see Section 4), a general system (see Section 5) as well as a concrete instance of such a system, a system for interpreting video scenes (Section 6). The paper concludes with a discussion (Section 7) and an outlook in Section 8.

2 Problem Description

The *configure-from-observed-parts* (CofOP-Problem) can simply be rephrased with following questions:

1. Given a number of parts, what aggregates can be built with these parts? What can we do with the things that exist?
2. Now, we have a partial aggregate made from existing parts, what is missing to make this aggregate to fulfill given requirements?
3. What role do aggregates play in finding a final configuration? What customer requirements can be fulfilled with the aggregates that are constructed?

In the CofOP-Problem we do not start the configuration from requirements and than try to infer needed parts, but we start from the things that exist and infer the use of the constructed aggregates.

Typical application examples are those where it is not clear in the beginning what should be configured, or what the aggregate will look like. In these cases, goal objects can not be given, but only the parts, which should form the aggregate to be constructed. Thus, a more explorative task has to be performed for handling the CofOP-Problem. Examples are:

- Systems that interpret images; existing parts in this case are shapes or objects an image processing system can identify in the images.
- Systems that interpret videos; existing parts in this case might be tracked objects that can be identified by a tracker (e.g. SCENIC see Section 6).
- Systems that build software; existing parts in this case are software components provided by an asset store or software configuration management system.

An extension of the CofOP-Problem is given with the question:

4. What happens if a new part is observed during the configuration process?

[1] HITeC e.V., University of Hamburg, Germany, email: hotz@informatik.uni-hamburg.de

In this case, not only previously given parts are considered in the configuration but parts that are dynamically created. Examples are parts that are created in a development process or dynamic situations of a video film, which should be interpreted.

3 A Structural Configuration System

A typical structural configuration system designed to support the configuration of aggregates based on component descriptions in a knowledge-base is organized in four separate modules:

Concept Hierarchy *Concepts* are described using a highly expressive concept description language, and embedded in a taxonomical hierarchy (based on the `is-a` relation) and a compositional hierarchy (based on the structural relation `has-parts`). Parameters specify concept properties with value ranges or sets of values. *Instances* selected for a concrete configuration are instantiations of these concepts.

Constraints *Constraints* pertaining to properties of more than one object are administered by a constraint net. Conceptual constraints are formulated as part of the conceptual knowledge base and instantiated as the corresponding objects are instantiated. Constraints are multi-directional, i.e. propagated regardless of the order in which constraint variables are instantiated. At any given time, the remaining possible values of a constraint variable are given as ranges or value sets.

Task Description A configuration task is specified in terms of an aggregate which must be configured (the goal) and possibly additional restrictions such as choices of parts, prescribed properties, etc. Typically, the goal is the root node of the compositional hierarchy.

Procedural Knowledge Configuration strategies can be specified in a declarative manner. For example, it is possible to prescribe phases of bottom-up or top-down processing conditioned on certain features of the evolving configuration.

A stepwise configuration could be executed by the configuration system according to the following basic algorithm:

```
Task specification
Repeat
   Determine current strategy
   Determine possible configuration steps
   Select from agenda and execute one of
          { aggregate instantiation,
            aggregate expansion,
            instance specialization,
            parameterization,
            instance merging }
   Propagate constraints
   Check for conflict
```

An aggregate is completely configured if all properties of the aggregate have been parameterized, all its required parts have been completely configured, and all constraints are satisfied. A conflict is encountered when the constraint net cannot be satisfied with the current partial configuration. In this case, automatic backtracking occurs. Backtracking can be controlled by procedural knowledge to achieve ”intelligent backtracking”. An example of such a system is KONWERK [1, 3]. For a more general ontology for configuration see [10]; for an overview of standard approaches see [4, 13].

4 Conceptual Framework

The CofOP-Problem requires a distinction of the following aspects[2]:

The reality (*Realität*) are things in the outside, materialized world. In our case, the real existing parts that can be used for configuring span the reality.

Substantiality (*Wirklichkeit*) covers everything the system knows about the reality. From the view point of the system the substantiality constitutes the real world. The substantiality is divided into the outer and inner substantiality.

Evidence space (the outer substantiality, perceived reality) (*Äussere Wirklichkeit*) is the representation of the reality in the system. The real world is *reflected*. The evidence space represents the observed parts that can be used in the configuration. Objects in this space are called *evidence*. Evidence is seen as given and not alterable.

Hallucination space (the inner substantiality) (*Innere Wirklichkeit*) is the representation of the things that might exist, i.e. that can be imagined. Objects in this space are called *real objects* because seen from the system point of view, those objects constitute the real world, even if they do not exist. Each real object might have or might not have evidence; it might or might not be justified in the evidence space, i.e. in the reality. Thus, real objects can be justified by evidence or can be imagined (*hypothesized* or *hallucinated*). Real objects that are imagined (i.e. have no evidence) are called *imagined objects* or *hypotheses*. Those real objects that are in fact in the reality (i.e. have evidence) are called *perceived objects*. Imagined and perceived objects can be distinguished from each other. However, real objects, if justified or not, can be part of aggregates, thus, can form the resulting configuration. In the hallucination space new inferences about the substantiality can be made.

The domain-dependent configuration model (*die Bedeutung*) contains the knowledge about a certain domain, typically represented in a knowledge base, i.e. configuration model. This model is used to interpret the substantiality and to infer new real objects.

5 Technical Aspects

In this section, technical issues concerning the realization of a system that solves the CofOP-Problem are presented.

5.1 Architecture

The general architecture consists of two systems, a low-level sensoric system and a high-level inference system. The low-level system establishes the connection to the reality. Examples are:

- an image processing systems which computes shapes or objects of an image;
- a warehousing system providing a constantly updated, complete description of all existing parts of a technical system;
- a software asset store or configuration management system providing existing software components.

The output of the low-level system is mapped to the evidence space and represented as evidence in the high-level system (see Figure 1).

The high-level inference system is a configuration system, that employs a certain upper model. This upper model represents the conceptual framework described in Section 4. The upper model consists of:

[2] This distinction is inspired by Spitzing’s distinctions made in the photography research, see [12].

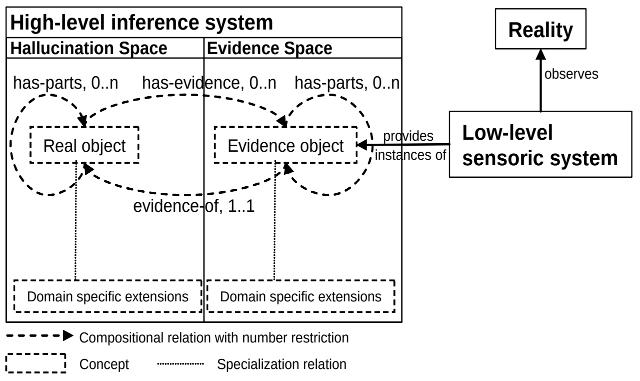

Figure 1. General architecture including the upper model representing the conceptual framework.

- the concept `evidence-object`. Its instances represent the parts that exist in the reality, i.e. its instances form the outer substantiality. Evidence-objects can have parts, if the low-level system is able to observe those, or have no parts, if the low-level system can only observe non-structured objects, which is typically the case.
- the concept `real-object`. Its instances and parts represent all what is known and what is hallucinated by the system, i.e. its instances form the inner substantiality.

The concepts `evidence-object` and `real-object` are related to each other by the relation `evidence-of` and its inverse `has-evidence`. A real object may have evidence or may have not, an evidence object should always be related to a real object; both relational aspect are represented with number restrictions. Real objects with a `has-evidence` relation of cardinality 0 represent imagined objects, real objects with a `has-evidence` relation of cardinality greater 1 represent perceived objects. Thus, there are no concepts for imagined or perceived objects, because every real object can be in both roles for distinct situations.

The relation `has-evidence` has following semantics: If an instance of the concept `evidence-object` exists, than an appropriate instance of the concept `real-object` is created. However, if an instance of the concept `real-object` exists *no* instance of the concept `evidence-object` is created, because evidence objects represent the reality, i.e. only the low-level sensoric system can trigger the creation of such instances. Thus, they represent imagined objects as long as no evidence objects are observed by the low-level system.

Domain dependent configuration models are added to the upper model by specializing the concepts `real-object` and `evidence-object`. The specializations of `evidence-object` strongly depend on the low-level sensoric system's facilities and the data it can deliver, e.g. structured objects or no structured objects, types of objects or descriptions via properties.

The relation `evidence-of` establishes a mapping from the evidence space into the hallucination space. This mapping is domain dependent. The mapping has to be made with the information provided by the evidence object.

5.2 Operational Aspects

The configuration steps introduced in standard configuration systems can be used for configuring an aggregate with observed parts (see Section 3 and [5]). However, following aspects are of major importants for handling the CofOP-Problem:

Configuration from bottom to top, from parts to aggregates: Processing integration steps from parts to aggregates is the first step performed when using observed parts. For example the configuration of the `evidence-of` relation for an evidence object is an integration step.

The bottom-up steps can continue into the hallucination space as long as unique decisions can be made. Thus, all inferences that are justified by the reality are drawn. One may think of this phase as the determination of those things that can be inferred from the evidences; or the answer to the question "What can be done with the parts that exist?" (i.e. Question 1. of Section 2).

Top down for generating hypotheses: In this second step a goal-object can be selected, which represents certain requirements. Starting from this goal-object the normal configuration process is performed. For example, a specialization is done by selecting an appropriate sub concept, which represents a hypothesis made from top-down. Through decomposition further hypotheses are generated and integrated in the configuration. One may think of this phase as an exploration of high-level concepts, which might be responsible for the objects and occurrences observed so far. Furthermore, this phase identifies such parts that are missing for constructing a complete configuration. Missing parts are those real objects that do not have evidence, i.e. the imagined objects (Question 2. of Section 2).

Use of bottom-up generated objects: If through top-down analysis a required part is identified, it has to be checked, if an appropriate real object already exists in the configuration, because it could have been generated in the bottom-up phase. This operation creates a configuration that contains both, imagined real-objects and evidence-based, perceived real-objects (Question 3. of Section 2).

Fuse real objects: If a part is generated from top-down and one from bottom-up, it could be the case, that those two parts are in fact the same. In this case only one part should be in the resulting configuration. If these generated parts fulfill a certain fuse condition, which ensures that they should really be one object, these parts can be fused. This means, that one real object is created, which contains all relations and parameters of the former two parts. The parameter and relation values of the fused object are computed by building intersections. This means that real objects that are related to the former two parts are also considered to be fused (*deep fuse*).

Spontaneous instantiation: To handle dynamic aspects a configuration step is needed that instantiates an arbitrary concept of the evidence space. For example, if a new objects enters a video scene, a new evidence instance should be created. Thus, the low-level system and the high-level system should be synchronized by synchronization points or steady polling. When new evidence instances are created at time t, those have to be integrated in the configuration even if the defined configuration procedure would focus on another configuration step at time t. Furthermore, backtracking has to be initiated, because the inferred conclusions could be wrong, because the evidence base is changed (Question 4. of Section 2).

Backtracking to the hypotheses generation: During configuration conflicts might occur as seen above. Because in the described schema the bottom-up steps are seen as unique inferences from the evidences, only the top-down hypotheses generation can be backtracked. The first hypothesis which was made can be seen as a backtracking point. While the evidence instances including their inferences stay in the configuration, all made hypotheses are withdrawn and the configuration process proceeds with another hypothesis.

6 SCENIC - an Example Applications

In this section, we shortly present the experimental system SCENIC (SCENe Interpretation as Configuration) which utilizes configuration technology for concrete scene interpretation experiments [5].

The SCENIC System was built employing the conceptual framework presented in this paper. The example scenes are taken from a table-laying scenario: A video camera is installed above a table and observes a tabletop. Human agents, sometimes acting in parallel, place dishes and other objects onto the table, for example, cover a dinner-for-two. It is the task of the scene interpretation system to generate high-level interpretations such as ”place-cover” or ”lay-dinner-table-for-two”. Occurrences of this kind are complex enough to involve several interesting aspects of high-level scene interpretation such as temporally and spatially constrained multiple-object motion, a knowledge base with compositional structure, and the need for mixed bottom-up and top-down interpretation steps.

According to the conceptual framework we introduced above, SCENIC consists of a low-level system, in this case an image processing system. Via a video camera and a tracking system the low-level system observes the table. A so called *geometric scene description* (GSD) is generated by the low-level system, which represents the objects and their properties seen in the video film. The high-level system consists of KONWERK and a configuration model containing the presented upper model and a domain specific model for table-laying scenarios (see Figure 2). KONWERK performs the symbolic interpretation subtask in SCENIC.

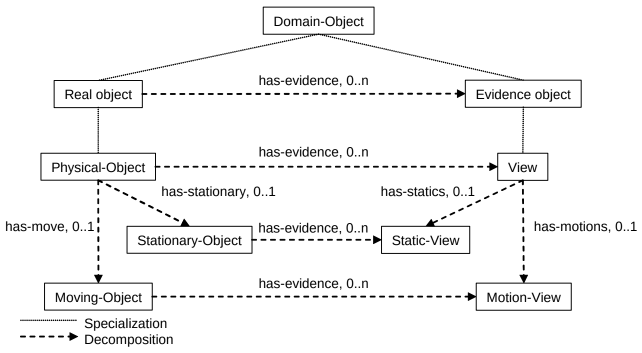

Figure 2. An extract of the domain specific specialization of the upper model presented in Figure 1.

From the GSD evidence objects are created during configuration. The GSD contains the observed objects as well as their motion in time and space. The bottom-up strategy integrates the evidence objects into real objects and those into aggregates of the hallucination space. For example, in SCENIC transports (like a hand-plate transport) are identified during this phase. When the bottom-up strategy is exhausted, top-down expansion begins. In SCENIC a hypothesis about a possible table-scene is generated as an aggregate, e.g. `dinner-for-two`. This aggregate is decomposed and further hypotheses are generated. This way hypotheses about a further development of a scene are generated by the typical configuration steps (e.g. aggregate decomposition see Section 3). During this expansion of possible occurrences the real objects created by the bottom-up strategy are used if possible. Thus, a configuration or *scene interpretation* is constructed, which consists of real objects with evidence and those without evidence, i.e. imagined and perceived real objects. In Figure 3 perceived objects are in realistic shape, imagined objects with possible positions are shown with icons.

A fuse operation in SCENIC occurs, when the low-level system cannot provide distinct information about an object. For example, a certain object can only be identified as a *dish*, not as a *cup* (e.g. the cup on the right in Figure 3 cannot clearly separated from the saucer underneath). From top-down it is inferred that this object has to be a cup, because the hypotheses generation determines that at this certain position of a cover only a cup can exist. At this point a real-object

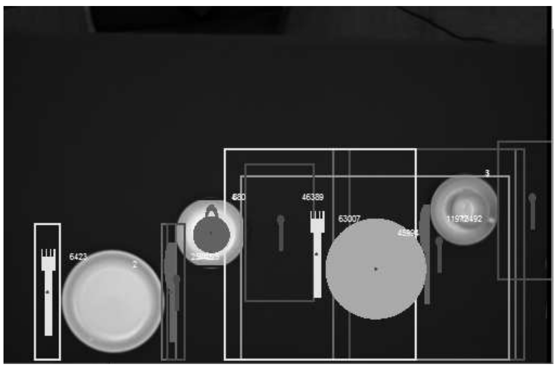

Figure 3. Result of a scene interpretation: realistic shape for evidence objects, hypothesized objects including their possible spatial regions are represented with icons and rectangles.

is created from top-down, representing a hypothesized cup, and a real-object is created from bottom-up, representing a dish. Those two real objects can be fused, if a fuse condition (like position and time overlap) is satisfied. This fusion represents the insight of the system, that the previously not identifiable dish is in fact a cup, given certain hypotheses. This possibility to disambiguate low-level sensoric input through high-level inferences is unique to this approach.

SCENIC was used to interpret video scenes consisting of about 300 frames. During this sequence 28 evidence objects (including those evidence objects representing motions) are processed, 81 imagined real-objects are created for the hypothesis *dinner for two* (again including those representing motions). For keeping track of spatial and time relations 130 constraints are created during configuration. 51 interpretation steps were needed to obtain the first intermediate scene interpretation (see Figure 3), using 90 sec of CPU time (1.8 GHz PC). Backtracking and additional 8 interpretation steps were needed to arrive at an alternative intermediate interpretation for the hypothesis *single dinner* using additional 45 sec of CPU time.

The configuration model consists of 50 types of real objects and 34 conceptual space and time constraints. The operational aspects presented in Section 5.2 are covered by 8 strategies.

7 Discussion

Data-driven approaches may take existing parts into account, if they refer to them during the configuration process. Data-driven approaches perform bottom-up construction of aggregates from their parts. If those parts are identified during the configuration process, a similar scenario as described in this paper has to be handled. However, the conceptual framework presented in this paper clearly separates hypotheses and evidences. Thus, following four situations can be distinguished:

1. Hypothesis, which is supported by an evidence, i.e. the relation `has-evidence` is established (*perceived object*).
2. Hypothesis, for which no evidence is yet found. This is due, when the real object has no relation to an evidence object (*imagined object*).
3. Evidence, which is already interpreted, i.e. the relation `evidence-of` is established (*accepted evidence*).
4. Evidence, for which it is not yet decided how the evidence should be interpreted. This is due, when the evidence has no relation to an imagined object (*not accepted evidence*).

For data-driven approaches employing a schema, where evidences are instances of concepts, case 2 cannot be represented and be used for reasoning, because no distinguishable instances for evidence objects and hypotheses exist.

A further opportunity to use the presented conceptual framework is its ability to support the evolution process, e.g. in software configuration. To handle the question: What happens if a new observed part comes into play during the configuration process that has no mapping into the hallucination space? This would indicate evidence for something which is still unknown for the system (not accepted evidence). However, because the evidence-object is generic each new developed part (e.g. a software component) can be represented by an instance of `evidence-object` and thus can be considered *during* the configuration process. If such a part cannot be mapped to a real object the configuration system would raise a conflict. Thus, this conflict could be handled by evolving the configuration model according to the new developed part [6].

8 Summary and Outlook

The configuration from observed parts is a problem in the area of recognition or interpretation of images and videos. But also in technical configuration like elevators or software-intensive systems a representation of observed parts, which should be used for configuration, is of importance. In this paper, we present a conceptual framework that explicitly represents observed parts and distinguishes them from other parts, without an existing representative in the reality.

An example for an instance of the conceptual framework is given by the system SCENIC, which interprets video films about table-laying scenes. In an upcoming system, which will be used for interpreting images of buildings, photographed from the front or from a satellite, we will also apply the described framework. The goal here is to identify regions in buildings like windows, doors, roofs, chimneys as well as aggregates like row of houses or roofs. Similar to SCENIC from a low-level system geometric scene descriptions are provided, which are interpreted with the high-level configuration system. The imagined real objects will be used to give feedback to the low-level system; for example, to focus on a certain part of the image for identifying a certain detail of a building.

ACKNOWLEDGEMENTS

This research has been supported by the European Community under the grant IST 027113, eTRIMS - eTraining for Interpreting Images of Man-Made Scenes.

REFERENCES

[1] A. Günter, *Wissensbasiertes Konfigurieren*, Infix, St. Augustin, 1995.

[2] A. Günter and R. Cunis, 'Flexible Control in Expert Systems for Construction Tasks', *Journal Applied Intelligence*, **2(4)**, 369–385, (1992).

[3] A. Günter and L. Hotz, 'KONWERK - A Domain Independent Configuration Tool', *Configuration Papers from the AAAI Workshop*, 10–19, (July 19 1999).

[4] A. Günter and C. Kühn, 'Knowledge-based Configuration - Survey and Future Directions', in *XPS-99: Knowledge Based Systems, Proceedings 5th Biannual German Conference on Knowledge Based Systems*, ed., F. Puppe, Springer Lecture Notes in Artificial Intelligence 1570, Würzburg, (March 3-5 1999).

[5] L. Hotz and B. Neumann, 'SCENIC Interpretation as a Configuration Task', Technical Report B-262-05, Fachbereich Informatik, University of Hamburg, (March 2005).

[6] L. Hotz, K. Wolter, T. Krebs, S. Deelstra, M. Sinnema, J. Nijhuis, and J. MacGregor, *Configuration of Industrial Product Families - The ConIPF Methodology*, Aka Verlag, Berlin, 2005.

[7] S. Marcus, J. Stout, and J. McDermott, 'VT: An Expert Elevator Designer that uses Knowledge-based Backtracking', *AI Magazine*, 95–112, (1988).

[8] J. McDermott, 'R1: A Rule-based Configurer of Computer Systems', *Artificial Intelligence Journal*, **19**, 39–88, (1982).

[9] K.C. Ranze, T. Scholz, T. Wagner, A. Günter, O. Herzog, O. Hollmann, C. Schlieder, and V. Arlt, 'A Structure-based Configuration Tool: Drive Solution Designer DSD', *14. Conf. Innovative Applications of AI*, (2002).

[10] T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen, 'Towards a General Ontology of Configuration', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1998)*, *12*, 357–372, (1998).

[11] E. Soloway and al., 'Assessing the Maintainabiliy of XCON-in-RIME: Coping with the Problem of very large Rule-bases', in *Proc. of AAAI-87*, pp. 824–829, Seattle, Washington, USA, (July 13-17 1987).

[12] G. Spitzing, *Foto Psychologie*, Beltz, Weinheim, Basel, 1985.

[13] M. Stumptner, 'An Overview of Knowledge-based Configuration', *AI Communications*, **10(2)**, 111–126, (1997).