# The Tacton View of Configuration Tasks and Engines

**Klas Orsvärn, PhD**
**Tomas Axling, MSc**

Date: 1999-04-21

Tacton Systems AB
Saltmätargatan 7
113 59 Stockholm, Sweden
klas.orsvarn@tacton.com
tomas.axling@tacton.com



**Abstract**

This paper presents the view of product configuration taken by Tacton Systems AB. We first define the product configuration task in general. We then distinguish three types of configuration tasks, describe different interactivity requirements, and discuss three common types of configuration engines. Finally, Tacton Configurator is briefly described.

## Product Configuration

In a *configurable product line*, custumization is combined with rational production and handling, by using a set of pre-defined components which can be combined in many different ways to satisfy a wide range of requirements.

In a *product configuration* task, the case input is a requirements specification, and the case output is a quantified list of components in a configurable product line which together satisfy the requirements and obey the technical restrictions of the product line, optionally with a description of how the component instances should be connected. The output is called a *configuration.*

It is usually the case that the same functional requirements can be satisfied with different combinations of components, i.e. different configurations are possible. In this case, the output configuration should be sufficiently optimal, implicitly or explicitly, according to some criterion, which is usually some kind of cost.

The above definition of product configuration, consistent with that of Mittal 1989, is very wide. For example, a computer program can be regarded as a configuration in the above sense, as well as a VLSI circuit. However, a programming language or the set of logical gates in digital circuitry is not regarded as a configurable product line. Product configuration is regarded as a routine task, although it requires a great deal of knowledge and skills. Programming and VLSI design are regarded as creative engineering tasks. So, what makes product configuration routine?

Inherent in the notion of configurable product line is not only its pre-defined set of components, but also a modular break-down structure. The break-down structure is a part-of structure of abstract configurable parts. The leaves of the part-of structure are components.

In contrast to a component, a configurable part can satisfy a wide range of varying requirements. The requirements input to a configuration task, is the requirements on a configurable part.

A configuration can be regarded as an instantiation of a configurable part. A configurable part can be instantiated in many different ways, to satisfy different requirements.

The modular break-down structure is not the only property that makes product configuration routine. Another important property of a configurable product line is that the requirements it can satisfy can be represented with a pre-defined set of parameters, where each parameter has a pre-defined set of possible values. The set of possible values of a parameter may be infinite in principle, but in practice it is finite. A parameter may represent an attribute of the configuration as a whole, or an attribute of a part of the configuration, such as a key component selection.

The specific requirements on a configuration is an assignment of values to parameters (more generally; restrictions on parameter values).

The configuration requirements may refer to several configurable parts, and several instances of those configurable parts, if these have interdependencies.

In *re-configuration*, an existing configuration is given as input, in addition to the requirements. The output is a description of the difference between the existing configuration and the new configuration needed to satisfy the input requirements. Usually, the new configuration is an expansion of the existing configuration, which has more functionality or more capacity. The creation of a new configuration from scratch is a special case of re-configuration, where the existing configuration is empty.

A *configuration engine* is a computer program which performs the configuration task, i.e. match input requirements with a suitable configuration. It uses a

representation of relevant knowledge of the configurable product line, called a *configuration model*.

## Types of configuration tasks

Even with the above more restrictive definition of product configuration, it is still a very wide definition. It is also still an NP-complete problem. Hence, we cannot expect to find a single configuration engine which is the most suitable tool for all configuration tasks. Since run-time performance is crucial, trade-offs are needed, and the suitable trade-offs will vary depending on the nature of the specific product line or context of use.

We have identified three kinds of configuration tasks, which are relevant to distinguish when discussing types of configurators: features-and-options, compositional, and network.

### Features-and-options configuration

In features-and-options configuration, which may also be called variant configuration, the configuration consists of a pre-defined set of modules, which may be mandatory or optional, and each module must be realised by one of several alternative components. There are restrictions on which components may be combined, and the selection of one component may also require the selection of a set of auxiliary components.

Configurable product lines based on mechanics, are usually structured for features-and-options configuration. Parameter setting of software products is another example.

When requirements are expressed in a product oriented way, the requirements are expressed by selection which optional parts should be included, and for each part selecting which one of the alternative components should be used.

### Compositional configuration

The following are distinguishing features of compositional configuration:

- The quantity of components of a certain type varies greatly between configurations, usually depending on capacity requirements.
- It is necessary to make sure that some of the component instances in a configuration can be connected, in order ensure that the configuration is valid (the connections can be made at pre-defined points).
- There are different plausible ways to connect the component instances in a configuration.
- There are restrictions on how component instances may be connected, and these restrictions may be context dependent, e.g. the legal connection of one component instance may depend on the actual connection of other component instances.
- Requirements are to a large extent expressed functionally rather than in terms of component attributes.

Compositional product lines are most often within the electronics domain, where circuit boards are typical examples of components that may have many instances, which must be connected by cabling and/or sub-rack positioning.

Features-and-options configuration is a special case of compositional configuration, and is often used for part of the configuration of compositional product lines. Thus, a configurator capable of solving compositional configuration is also capable of solving features-and-options, but the opposite does not hold.

### Network configuration

Network configuration is compositional in the sense that the number of modules is not pre-defined. The distinguishing feature of network configuration is that the requirements are expressed by selecting nodes ("components") and connecting them together. Connections are made at pre-defined points. Some connections of nodes are not valid. A node may be a component, a class of components, or a configurable part. If the node is a class of components, a component selection must be made. Different alternative components may have different connection restrictions. If the node is a configurable component, it may in turn be configured according to features-and-options configuration or compositional configuration. Its parameter value assignments may depend on the network configuration. Network configuration may thus be used to configure a network of nodes where each node itself needs to be configured.

Layout configuration is a special kind of network configuration, where the nodes in the network represent classes of components. The task is a to lay out a valid network of nodes, where for each node a selection between different alternative components must be made. A typical example of layout configuration is interior design of different kinds, e.g. office furniture or kitchens. The logic of layout configuration is usually similar to that of product oriented features-and-options configuration, i.e. to make sure that components combinations are valid.

## Interactivity requirements

This section will briefly describe different kinds of interactivity that a configuration systems may provide. The importance of these requirements differ between product lines and user categories.

The basic functionality according to the definition of the product configuration task is to propose a configuration that is correct and satisfies the requirements. This can be done in a more or less interactive way. A high degree of interactivity simplifies the end-user's task of finding a suitable configuration, and exploring different alternatives.

In batch-mode, the configurator will take the complete requirements as input and try to create a configuration as output. If the requirements cannot be satisfied, i.e. they are inconsistent, the configurator will say so. Ideally, it will also be able to tell the user how to change the requirements so that they can be satisfied.

When the risk of inconsistent requirements is high, batch-mode configuration can be very frustrating for the end-user. *Propagation* is a functionality that propagates the consequences of each selection entered by the end-user, so that all alternatives presented to the end-user are consistent with previous selections. The propagation provided by a configurator may be incomplete, in which case not all consequences are propagated. In this case, the entered requirements may still be inconsistent.

An interactive configurator will allow the user to enter requirements in any order. The user normally wants to begin with the most important requirements, and relative importance of requirements may differ greatly between users (for some customers, the power is the most important property of a car, whereas others may regard the colour as more important). Similarly, it should be possible to make changes to the requirements given, i.e. to retract a requirement and perhaps enter a new selection.

Another desirable functionality is to explain why a specific alternative is inconsistent with previous selections, i.e. which previous selections to change in order to select this alternative.

The configurator should be able to propose default values for each selection, to simplify requirements entry. The set of default values should represent the best configuration according to some criterion.

## Types of configurator engines

We distinguish three common types of configuration engines, described below.

### Product structure with restriction tables

Requirements are expressed by selecting optional parts and alternative components in the part-of break-down structure. Restrictions are represented in a data-base, with tables representing valid, invalid, and required combinations of (usually) component pairs. The configuration engine assists the user by making sure that selections obey the restrictions, and by selecting required components. The task of determining which combinations of selections is valid corresponds to determining satisfiability of propositional logic, which is NP-complete. Therefore, this kind of engine usually only checks a selection for validity after it has been made.

### Constraint solver

A constraint solver engine represents the configuration problem as a set of variables, which represent either requirement parameters or part of the configuration. Given requirements expressed as value assignments to some parameters, a constraint solver engine represents the solution as the domain of valid values of the other variables. One advantage of a constraint solver engine is that it offers a high degree of interactivity. Consequences of one parameter value assignment are directly propagated to other parameters. However, the degree of interactivity differs between different constraint solver engines.

Some constraint solver engines are capable of searching for a provably optimal solution when there is more than one possible configuration, and may be able to apply different optimisation criteria, such as cost or space.

Since propagation is an NP-complete problem, constraint solver engines usually offer incomplete propagation. The degree of incompleteness differs between solvers. It is desirable that variables with domains presented to the user are propagated completely, so that the user knows which of the presented values are valid. If the user selects an invalid value, some constraint solvers are capable of suggesting which previous selection to change in order to make the invalid selection valid.

Compositional configuration tasks require *dynamic constraint solvers*. In order to represent the connections between component instances, each instance needs to be represented with one variable. Since the number of instances of a component may vary greatly in compositional configuration (e.g. in the order of zero to thousands), the solver must be able to generate new variables depending on the problem at hand.

### Constructive search

A constructive search engine will search for one suitable configuration that satisfies the requirements. The engine gradually constructs the configuration solution as it searches. With a constructive search engine, it is not feasible to select provably optimal solutions, since it would require all possible solutions are generated and compared. However, it usually allows local optimisation, i.e. optimisation for a sub-problem, either in the selection between candidate components, or for a larger part of the problem. This may be sufficient to generate a suitable configuration, i.e. a sufficiently optimal solution.

An advantage of a constructive search engine is that its configuration process is easier to understand, and it is therefore easier to structure the configuration model for efficiency in search. Moreover, in many product lines it is difficult to define a cost function, so that the provably optimal solution can be selected. The reason is usually that some optimality criteria are soft, i.e. they express preferences that are not strict and may be conflicting. Soft criteria in configuration tasks are often naturally expressed in terms of the sequence of steps in the configuration process, e.g. “if you do this before that, the result will be sufficiently optimal”. Procedural knowledge is easier to express with a constructive search engine.

Another advantage of constructive search is that the engine only represent one solution at a time. This means that it solves a simpler task than the constraint solver engine, with less functionality, and can thus have a better run-time performance.

A hybrid engine will use a constraint solver for requirements capture, and constructive search to find a suitable configuration.

A constructive search engine is less interactive than a constraint solver, but is usually more efficient than a constraint solver for complex compositional configuration where complete propagation or optimisation are not required.

## Tacton Configurator

Tacton Configurator has two engines, a dynamic constraint solver and a constructive search engine. It is capable of solving all types of configuration problems described above, and combinations thereof.
Based on experiences of product configuration systems since 1987, in a wide range of product lines, Tacton Configurator was originally developed as a research prototype at the Swedish Institute of Computer Science, under the name Obelics (Axling, 1996, Orsvarn, 1996). Tacton Configurator was released as a product in 1997, and has since been applied for configuration of a wide range of product lines, from simple features-and-options, to complex networked compositional configuration, for sales configuration of quotations as well as installation configuration.

## Acknowledgements

The view of configuration presented here has been influenced throughout the years by many people at the Swedish Institute of Computer Science and Tacton Systems AB.

## References

Mittal, S. Frayman, F. (1989). Towards a generic model of configuration tasks. Proceedings of the 11th IJCAI.

Axling, T. and Haridi, S. (1996). A Tool for Developing Interactive Configuration Applications. Journal of Logic Programming 26(2).

Orsvärn, K. and Hansson, O. (1996) Prestudy of Configuration of a Naval Combat Management System, In Working Notes of AAAI Fall Symposium on Configuration, MIT, Nov. 1996.