A Resource-Based Paradigm for the Configuring of Technical Systems from Modular Components.

M. Heinrich           E.W. Jüngst
Daimler-Benz AG        Research Institute Berlin

Abstract
In the resource-based paradigm the interfaces through which technical systems, their components and their environment interact are modelled as abstract resources, and each technical entity is characterized by the types and amounts of resources it supplies, consumes and uses. This intuitive model, derived in one application area, is shown to be in concordance with the design rationale of modular component systems. A simple self-organizing configuring inference procedure for the resource-based paradigm, resource-balancing, with a description of the environment of the technical system as the requirement specification, is derived from the basic acceptance criterion for configurations. Five levels of knowledge are defined for this paradigm and introduced in a simple representation scheme which, through its inherent locality and mutual isolation of component knowledge, allows efficient acquisition and maintenance of even large component knowledge bases. First experiences with the implementation and use of these ideas in the prototype shell COSMOS are reported.

AI Topic: knowledge representation, inference, configuring
Domain Area: modular component systems
Language: CommonLISP + Frame Oriented Language Kit
Status: field test of shell prototype, on-going research
Effort: 3 PY
Impact: novel basic model for the configuring task with a decisively improved maintainability of the knowledge base.

1. Introduction
Configuring is the construction of a technical system according to the requirements of a specification by selecting, parametrizing and positioning instances of suitable existing component types from a given catalogue [24]. Thus, configuring itself does not involve the creation of new component types [13]. Creating new component types for addition to the component catalogue would rather be the innovating task of designing.

Configuring of technical systems is an ubiquitous industrial engineering task and has been a domain for expert system application ever since the longtime paragon R1/XCON [1]. Most configuration expert systems agree in using frames or objects for the representation of factual knowledge about component types [2, 3]. For the knowledge about how to use the factual knowledge and how to find a good configuration quickly, two approaches are used. Expert systems in the tradition of XCON [1] represent the actions of a human configuration expert through production rules and mimic the human in a by-rote performance. The other approach is to find and represent the principles that guide the human configuration expert. The prevalent constraint-based model [4-10] treats the functional specification of the technical system and the relations between components as constraints on the components and their attributes. Such constraints can be used to check hypothetical configurations in a generate-and-test methodology [5, 11] or as guidelines for selection of components [7, 9, 12]. If "key components" which represent the different functional constituents of the technical system can be identified [4, 13], and the component catalogue can be organized into taxonomies with root classes corresponding to the "key components", the configuration task can be viewed as a classification problem and implemented with the taxonomy as a decision tree [12, 14], with improvement possible by making partial choices [15].

When "key components" themselves can be configured, e.g. by recursively applying the process of functional specification, constraint propagation and component selection [10, 13], a hierarchical decomposition of the technical system is achieved. A mixed hierarchy of "has-parts"-relations and "has-specialization"-relations is an elegant and efficient representation scheme for such knowledge [7, 9, 14].

A more specific principle, that components in a technical system only connect at specific interfaces or "ports", has been recognized as an implicit understanding [16], as a prime source of constraints on components [17, 18, 19, 20] and as a distinct "architectural" type of constraint [13]. When for each component the knowledge about the component types that it may be connected to and those components that may be contained in it is kept as an attribute of the component [17, 18], maintainability of the knowledge base is much enhanced over purely rule-based expert
systems, where these constraints are mixed up with sequencing knowledge, thereby increasing the very large number of rules that have to be maintained [21]. However, the direct references between the component types [13, 22, 23] make it necessary that every component description in the knowledge base is re-examined and possibly changed whenever a new component type is introduced to the catalogue or deleted from it. This re-examination is a very demanding and time-consuming activity and could well be responsible for the huge amount of work spent on maintenance of large knowledge bases for configuration expert systems [21].

Our own studies of modular component systems and of the configuring of technical systems from modular components led us to a very general principle used by human experts that subsumes the connectability principle: the principle that systems and components interact mainly through interfaces which can be thought of as resources and that the resources demanded and the resources supplied by components have to be balanced. This principle, which has been independently recommended as a consistency check [11], we propose as a basic model for the configuring of technical systems [24].

2. The resource-based model

The concept of resource is an intuitive abstraction of the interactions between components and between a technical system and its environment. The notion of resource includes all kinds of extensive (accumulative) physical, technical and commercial entities, both real and virtual, that might be supplied by one component of a system and consumed (exclusively) or used (temporarily or in common) by another component (see fig. 1). Examples of resource types from the realm of computer systems are electrical current at a certain voltage, cooling power, floor space (physical entities), card slot space, input ports, memory capacity, software procedure interfaces, bus connectors (technical entities), purchase capital, construction work-time, software licenses, supervisor attention (commercial entities).

A resource-based model characterizes a technical entity mainly by the types and amounts of resources it supplies, consumes and uses. With a resource-based model, the technical system, its components and its environment can be described in a single common paradigm. The description of the environment, stating those resources and amounts demanded of the technical system and those resources and amounts provided for the technical system, obviously can play the role of technical specifications for the technical system.

![Figure 1](image1.png) **Figure 1** basic relationships of components and resources in the resource-based model of technical systems

We developed the resource-based model first for the configuring of modular computer systems, where practically all constraints are resource-based. But resource-based models apply as well to other modular technical and organizational systems. This can be explained from the fact that a technical system is always built for a purpose, to provide some service, i.e. some real or abstract resource or resources for use or consumption by its environment. Such resources do not arise by themselves or out of nothing, but are supplied by components of the technical system. This, after all, is the sole reason for a component becoming part of the technical system. The components may in turn themselves require other resources for their functioning which have to be supplied by further components. At the end of that "chain of supply", some resources have to be supplied by the environment to the technical system for use or consumption by its components (see fig. 2).

![Figure 2](image2.png) **Figure 2** resource-based model of a technical system and its environment

The components available for building the technical system are not arbitrarily designed but determined by considerations of cost-effectiveness in the trade-off between
universality, which leads to low cost through high-volume production, and adaptation, which reduces the cost of designing-in, manufacturing and assembling the components into a product. Most cost-efficient for application areas where there are only few and standardized resource types but large variations in the amount of resources from case to case are modular component systems.

A modular component system is a collection of types of modules each designed with the objectives of working well together and of being easily configured into technical systems for a wide spectrum of requirements. The modules, therefore, are usually designed to supply a certain optimized amount of only one resource or of a suitable combination of a few resources. Common basis for the design of the modules is the system design of that modular component system which identifies and specifies the physical and technical interfaces and the resources that are exchanged via these interfaces. This system design stays virtually unchanged during the lifetime of the modular component system and spans several generations of modules. Even in the case where the system design is altered, the practical considerations of upward-compatibility will only allow the addition of some new types of interfaces and resources, which does not affect any of the older modules types. Thus, for a modular component system, the component types will fit quite naturally into the resource-based model.

3. Resource-Balancing

3.1 The Principle

The idea behind the resource-balancing principle is simple: A configuration is not acceptable unless the resources which the environment and the components demand are each balanced by the resources which the components and the environment can maximally supply [11, 24]. This suggests a basic configuring algorithm with the resource model, which most human experts employ consciously or subconsciously: Starting from the resources demanded by the environment as stated in the requirements specification for the technical system, focus on a resource type not yet balanced, determine the list of component types which can supply that resource, select one component type from the list, incorporate a component of that type into the technical system, and repeat that process until for every resource the required amount is balanced by the amount of resources supplied by components or by the environment, with backtracking on the decisions as the simplest strategy to cope with dead-ends and with the situation when insufficient resources are supplied by the environment [24].

3.2 The levels of knowledge

This configuration process corresponds to "reasoning from first principles". It requires only

- system knowledge, i.e. knowledge about the resources in the system specification for the modular component system, and
- catalogue knowledge, i.e. the technical specifications of each component typically contained in the manufacturers catalogue,

to find a formally viable configuration if one exists.

The heuristic knowledge that only human experts can provide from their experience with the configuration process is on three levels:

- exception knowledge, e.g. knowledge about idiosyncrasies of components not contained in the catalogue, which is necessary to achieve a correct configuration.
- evaluation knowledge, e.g. knowledge about a measure of quality of the configuration and about how to predict it on the component level during the configuration process, that will help achieve a good configuration.
- performance knowledge, e.g. knowledge about some advantageous sequencing of decisions, that will lead to an acceptable configuration quickly.

![Figure 3](levels_of_knowledge_in_a_resource-based_model.png)

In the application areas we studied, the exception knowledge was mainly concerned with simple and compound incompatibilities between component types in a configuration and with special requirements for the positioning of components.

The measure of quality of a correct configuration is usually its cheapness given that it satisfies the qualitative and quantitative requirements. That a configuration is not acceptable unless its price is below a given limit can be expressed simply by making the cost of a component a (commercial) resource that has to be provided by the environment. When more than one component is applicable, a local decision about the component with the least overall
cost is necessary; the list of suitable component types must be sorted in order of decreasing cheapness, with the quotient of (units-of-resources provided) and (price of component) as the figure-of-merit, and the first component type of the list must be selected. To account for the cost of further components entailed by the selection of a component when computing the figure-of-merit, the cost of the resources consumed or used by that component must be estimated and added to its purchase price.

In the performance knowledge, the human experts need not concern themselves with the basic organization of the configuration process like they must in most rule-based approaches, but only with the fine-tuning of an self-organizing configuration process which is already quite efficient due to the supply-consumption-characteristics of the components. For simple modular component systems, no fine-tuning is necessary at all, and a static sequencing priority assigned heuristically to the resources has yet proved adequate for optimizing the sequence of decisions for others.

4. Knowledge representation

With the resource-based model each of these five distinct levels of knowledge (see fig. 3) can be acquired incrementally and quite independently and can be represented in well-structured knowledge bases. This leads to a decisive improvement in the maintainability especially of the knowledge base which will scale-up well for the large knowledge bases encountered in practical applications.

The system knowledge, i.e. knowledge about the types of resources that arise from the system design of the modular component system, can be organized in a resource taxonomy based on resource similarity, which can be exploited for resource substitution decisions.

Fig. 4 shows an excerpt from the resource taxonomy of an example (PLC component system AEG-MODICON A500). For each resource type, the name, unit of measurement, value type (integer or real), cardinality, usage type (common or exclusive), resource generability type (create-as-required or check-that-sufficient), positioning type (contain or connect or ignore) and existence (real or virtual) can be specified. The knowledge about resources is acquired and represented very early in the knowledge acquisition process and tends to become stable after a very short time. The system knowledge needs then never to be changed later except through the addition of new types of resources, which cannot affect any of the previously entered knowledge about system and components.

The catalogue knowledge is the largest and most volatile knowledge base. It contains the knowledge about the types of component that are available for configurations. The ideal representation paradigm for the component types are the classes of an object-oriented language, where the similarities between component types can be used to construct a taxonomy with the catalogue components as leaves of the class tree. Fig. 5 shows an excerpt from such a component taxonomy for a prototype that configures programmable logical controllers (PLC) from a subset of the large modular component system AEG-MODICON A500. The types of resources that a component may typically supply, consume or use are introduced as value-slots at suitable superclasses of the class tree together with default values for the amount of resource. Thus the catalogue components can easily be entered into the knowledge base by specializing an appropriate superclass and filling in the specific values of the resource amounts for that type of component. In the example of Fig. 5, the resource "purchase-price" is introduced at the root class "component", as every component will cost money. The resource "supported-weight" is introduced at classes "hardware" and "firmware", which have physical existence. The resource "cage-slot" is introduced at classes "board" and "card-cage", which consume resp. provide it, etc. etc.
The heuristic knowledge of human experts, i.e. exception knowledge, evaluation knowledge and performance knowledge, has natural places of attachment in the classes of the component taxonomy or of the resource taxonomy:

The exception knowledge is attributed to the component types or superclasses it applies to, with different possible levels of expressiveness. Knowledge about simple incompatibility between components may be attached to a component as a list of incompatible components that will be checked against when a new component is selected, or as a rule set of e.g. predicate logic expressions about unacceptable partial configurations, or as a method for the checking procedure. The same applies to positioning exceptions, which may override the automatic positioning implied by the consumption of position-bearing resources.

The evaluation knowledge is easily expressible by the purchase-price value for the component, and by specifying the per-unit-value as a resource attribute, which can be used in a standard procedure as default for the cost entailed by the component in the computation of the figure-of-merit. If necessary, overriding procedures can be specified for resource types or superclasses and even further overriding procedures for component types or superclasses.

The performance knowledge about quick ways to reach an acceptable configuration is expressed as influence on the sequence in which resources are considered for balancing. It can be represented by a static priority attributed to resource types or superclasses in a value-slot. Instead of using a priority number which - though efficient - does not retain knowledge about the reason for the priority assignment, a double list of resources with asserted higher resp. lower relative priority should be kept for each resource type. After a priority knowledge change, the resulting sequencing can be efficiently generated from this point-logic temporal constraint network without inequalities. Of course, as with the other knowledge, hooks for the introduction of overriding dynamic priority assignment rules or methods can be provided easily in the resource definitions.

This knowledge representation scheme, by describing each component type only in terms of the resources each instance of that component type supplies, consumes and uses, suppresses direct references to other components and effectively isolates the knowledge about components from each other. All the knowledge, including any heuristic knowledge, even in form of rules, is organized locally within the compact and efficient structure of the component and resource taxonomies, and can be found and accessed quickly and predictably by a human expert. A new component type thus can be added to the component catalogue without reference or change to older component type descriptions, and any component type description can be removed - together with all the knowledge pertaining to it - without consequence for the rest of the component knowledge base. Combined with the effect of representing not "by-rote" performance but "principle-based" knowledge, even very large knowledge bases can be expected not to show any of the maintenance problems rule-based configuration expert systems are plagued with.

The universality of this knowledge representation scheme for all kinds of modular technical systems makes it economically feasible to provide powerful interactive standard tools for the acquisition and maintenance of the knowledge bases. The taxonomic structure of the component and resource type catalogues is exploited best with the familiar browsers displaying the class tree graphically and allowing access to the knowledge interactively via the class node icons.

5. A Configuration Shell for Modular Systems

The resource-based model and the resource-balancing principle was developed from studies for our configuration expert system shell COSMOS (Configuration Shell for Modular Systems). COSMOS is based on an in-house hybrid expert system building tool FOLK (Frame Oriented Language Kit) implemented as an extension of COMMON LISP. The expert system shell itself takes only 15% of the entire code. Fig. 6 shows the system structure of COSMOS...
with the resource catalogue and the component catalogue. Most of the effort has so far been spent towards knowledge acquisition and maintenance tools and towards debugging and explanation facilities that support the domain expert in the development and tuning of the application-dependent knowledge base for the configuration shell. The resource types and component types and their superclasses are classes in FOLK, the interactive tools are based on browsers that are easily constructed as specializations of a generalized browser included in FOLK, with dynamically constructed fill-in-the-blanks masks and menus for value selection.

The same MMI technique is used for entering the description of the environment which is used as the specification of the technical system to be configured. For the specification of the communication structure in the configuration of distributed computer systems, a browser is used for entering and maintaining the factual knowledge about the actual network of computer stations and communication links. Besides that it serves as primary MMI for the accessing of the technical specifications of stations and links. Very little effort has been spent here, as the specification data interface is most likely to need adaptation when the developed expert system is integrated into an embedding software environment, after all.

The inference engine of the expert system shell COSMOS uses a blackboard architecture (see fig. 7) with a decision record and a balance sheet. The decision record contains, for each decision step, the list of all viable component types in order of decreasing utility, where the first and highest-ranking component type is considered selected. The balance sheet tallies for each type of resource the amount required by the selected components against the amount supplied by the selected components.

The environment, which states the requirements specification, is entered as the selected component in the first decision. In each inference step, from the column of the selected components in the decision record, the balance sheet is updated, from the balance sheet the unbalanced resource with the highest priority is determined, from the component catalogue all component types that can supply that resource are evaluated in their prospective environment and listed in order of decreasing utility (here a general check of component quality excludes all incompatible or substandard component types), and this list is taken as next entry to the decision record. Backtracking occurs when the list is empty and is performed by returning to the previous decision in the decision record and deleting the first component type of that list, which is added to a list of all rejected components for that decision step, and proceeding from there. An impossible configuration task is detected if backtracking reaches the first line which contains the environment as the selected component.

The explanation engine is yet simple. For each step, a reconstruction of the balance sheet to the state prior to the selection of a component, together with knowledge of the resource considered and the evaluation results for the list of viable components, allows to explain the "why select ...?" - questions. The list of rejected component types, which includes the attribute relevant for the rejection, takes care of the much more interesting "why not select ...?" - questions.
6. COSMOS experience and future work

Though aware of many possible improvements to the inference procedure, in our first implementation of the basic configuration expert system shell COSMOS we wanted to explore how well the simple basic algorithm abstracted from human expert behavior would serve in practical applications. We speedily included an improvement that enables components to balance certain resources locally, which is a necessity e.g. for spatially distributed systems with local power supplies.

For some hierarchically structured component types of subsystem complexity, however, the correct knowledge representation through resources was still quite tedious to derive. For a more suitable handling of such components, and other benefits, future work will be directed towards specific knowledge representations for structured components and composite resources.

COSMOS has been successfully used for a number of prototype expert systems for the configuring of Programmable Logic Controllers (AEG Modicon PLC Series A500 and A120) that select hardware (from cabinets to cables), firmware and operating system software, document the configuration and specify the resources required of the environment including purchase price, construction work-time, floor space and power consumption. A typical configuring task for the Series A500 is performed in 1 minute, the largest configurations (3 cabinets) in 5 minutes on a COMPAQ 386/25. An extension of the shell towards automatic generation of configuration-specific installation manuals is intended.

The acquisition and maintenance of knowledge for the component catalogues proved to be as easy as expected of our resource-based knowledge representation. Through the COSMOS tool set, knowledge can be entered by an expert without intercession of a knowledge engineer. In the first expert system prototype developed with COSMOS for the large Series A500 PLC family, the knowledge base containing the most important quarter of the total knowledge (including communications and structured components) was acquired from paper catalogues and entered in less than four person weeks. The bulk of maintenance work, i.e. the introduction of new components, the phasing out of discontinued components and the updating of prices, can now be performed by marketing personnel alone. Most notable from the point of view of the marketing was that the information in the component catalogue, while organized locally like a paper-based catalogue, is free of its side-effects and, with the tools provided, much easier to maintain. Some ideas about printing future paper catalogues directly from the knowledge base are entertained.

Field tests are currently performed in a wider spectrum of applications, from configuration of address reading and letter sorting systems to approximate configuration of low voltage switchgear for quoting purposes.

An application area of configuration expert systems not yet exercised is the use of the knowledge acquisition component and the configurator for the planning of new component types. A hypothetical new component is entered into the catalogue and evaluated by repeating the configuring of a library of typical past orders and comparing the configurations with previous results. The description of the hypothetical new component then constitutes the technical specification for the design of the new component. For the design of a totally new modular component system, system applicability can be tested by building a knowledge base of the planned resources and component types in the shell and evaluating the set of planned components with the expected standard mix of applications. The description of the components can then be used as formal requirement specifications for the design of the components and as the knowledge base for the configurator expert system.

7. Conclusions

The resource-based paradigm reflects the design rationale of modular component systems and captures a basic principle that human expert configurators are guided by. Resource-balancing is reasoning from "first principles" and "deep knowledge", but with a simple self-organizing basic inference process.

In the resource-based model, knowledge about configuring technical systems is layered into five distinct levels with well-defined responsibilities and intuitive appeal. These knowledge levels allow a well-structured knowledge representation with compact and mutually isolated knowledge for each component type.

This isolation of knowledge about components from each other is the key to efficient maintenance of even large component knowledge bases. As first experiences corroborate, the resource-based paradigm overcomes the maintenance bottleneck for the case of configuring technical systems from modular components.

The universality of the resource-based paradigm makes elaborate tools economically feasible. Through the available tools and the inherent power of the resource-based model, the human experts can readily shun the services of knowledge engineers in knowledge acquisition and need not become involved with routine knowledge maintenance.

For research, the resource-balancing principle holds much potential for more advanced inference techniques, and the resource-based model could be an interesting starting point for model-based design.
References:


