The Resource-Based Paradigm: Configuring Technical Systems from Modular Components.

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

Abstract

In the resource-based paradigm the interfaces through which technical systems, their components and their environment interact are modeled 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. Four 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.

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 development of new component types [13]. The development of new component types, e.g. for addition to the component catalogue, rather is a typical task in the domain of design.

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 was independently recommended as a consistency check [11], we proposed as a basic model for the configuring of technical systems [24, 25]. A very similar paradigm of the demand and supply of resources was presented as a "computational market model for distributed configuration design" [26].

The fact that a major industrial design task (configuring) involves only components selected from catalogues has also been recognized in Mechanical Engineering and has led to research on "Catalogue Design", but with little information interchange between the Mechanical Engineering and the AI communities. A basic idea is that catalogue design involves two phases, the selection of a system structure and the selection of real components for the generic elements in the chosen system structure. The main research thrust, however, currently is towards techniques for the design of the system structure. Genetic algorithms seem to do very well for this task [27].

The Constructive Problem Solving (CPS) approach [28], with a rigorous formal treatment under an abductive reasoning scheme, follows a similar idea. A semantic model of the system structure is incrementally built up from generic components, the permissible attribute ranges of the generic components are increasingly constrained through the external specifications, the emerging structure and the emerging mutual dependencies of the components, and the final configuration is won by choosing a conforming set of components from the given component type catalogue. A logic-based calculus with about forty construction rules was proven sound and complete [29]. Besides the complexity issue through the lack of heuristic metalevel guidance rules, however, a basic problem with logic-based incremental approaches is that logically consistent intermediate configuration states could be isolated by arbitrarily many inconsistent states.

We re-present here the Resource-Based Paradigm because it gives important insight into the human conceptualisation of the configuring task, opens a novel line of reasoning for the heuristic guidance of configuring, discloses a type of constraints basic for configuring problems for which no sophisticated algorithms are known, and yet proves to be quite efficient in important practical application areas.

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).

The resource-based model characterizes a technical object 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 (see fig. 2).

We re-present here the Resource-Based Paradigm because it gives important insight into the human conceptualisation of the configuring task, opens a novel line of reasoning for the heuristic guidance of configuring, discloses a type of constraints basic for configuring problems for which no sophisticated algorithms are known, and yet proves to be quite efficient in important practical application areas.
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 to become part of the technical system. These 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. 3).

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

for 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 two further levels:

- 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.

Additionally, exception knowledge must be represented. Instead of expressing incompatibility of components (our first approach), it has proven most effective to describe hopeless situations in terms of resource balance situations. For more complicated situations, simple rules which change the priority of resource satisfaction or component placement have proven adequate.

3.3. Resource Quality

The simple resource model described above is not sufficient to adequately represent reality because often resources which superficially seem the same will in some aspects really be slightly or significantly different, e.g. electrical current is only equivalent when delivered at the same voltage and polarity or frequency, or, more accurately yet, within a specific voltage interval. This fact we express by associating quality attributes with the resource types, e.g. voltage intervals or voltage and tolerance, polarity or frequency with electrical current, rotational speed with torque, access time with memory capacity, baud rates with RS232-connections. Resource-Balancing becomes much more intricate then, as the qualities have to be compared in order to determine which provided resources are compatible with which required resources.

3.4. Knowledge representation

With the resource-based model each of these four distinct levels of knowledge (see fig. 4) 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. The quality attributes are represented in value-slots of the resource type classes.

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. 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.

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 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.

The performance knowledge about quick ways to reach an acceptable configuration is represented by a static priority attributed to resource types or superclasses in a value-slot.

The exception knowledge, captured as rules on resource types and amounts, is represented in value-slots of resource types and their superclasses as part of the performance knowledge.

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, avoids all 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.

4. 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).

4.1 The COSMOS Architecture

Figure 5 shows the system structure of COSMOS with the resource catalogue and the component catalogue. Both resource types and component types are represented as classes in an object-oriented language. Most research effort was 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 visualized in browsers, the interactive tools are based on those browsers augmented with dynamically constructed fill-in-the-blanks masks and menus for value selection.

The inference engine of the expert system shell COSMOS uses a blackboard architecture (see fig. 6) with a decision record, resource model and balance sheets, and the agenda. 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 sheets tally for each type of resource the amount required by the selected components against the amount supplied by the selected components.

The environment, carrying 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 sheets the unbalanced resources
are determined and their treatment put on the agenda, the action with the highest priority is chosen, either the selection of a component or the positioning of a component.

For component selection, all component types from the component catalogue that can supply that resource type or a compatible resource type are evaluated in their prospective environment and listed in order of decreasing utility and this list is taken as next entry to the decision record.

Backtracking occurs when the list is empty. It is performed by returning to the previous decision in the decision record, 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. Backtracking also occurs when a hopeless resource balance situation (according to exception knowledge) is detected. An impossible configuration task is determined 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.

4.2. The COSMOS experience

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.

COSMOS has since been successfully used for a number of prototype expert systems, e.g. for the configuring of Programmable Logic Controllers, Conveyor Belt Systems, and Switchgear System Controllers. A commercial re-implementation has been successfully applied to the configuring of PLC systems including software and also including the configuring of the communication network hardware (Conquest, AEG/Schneider/MODICON).

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. The knowledge (including communications and structured components) was acquired from paper catalogues and entered with less than four person weeks effort. Some training and experience, of course, is needed for the design of the basic representation structure and the first representation of exceptions. 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. The transfer of price information from EXCEL databases to the COSMOS knowledge base is automated. Most notable from the point of view of 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.

5. Hierarchical Configuring

The Resource-Based Paradigm is not limited to configuring of classical modular technical systems like PLC where the components are concrete and the structure of the product is simple. A very important industrial application area are industrial plants and installations which are constructed from a large variety of different modular component family systems. Such applications are well within the scope of automatic configuring by a suitable expansion of the resource-based approach to both the real and abstract components of such plants.
In our approach, the top modeling level consists of the process elements as viewed by the project planning engineers. In many application areas, these abstract process elements exhibit a modular behaviour and functional capabilities which can be easily captured and adequately expressed in a resource model. The overall task is then solved by selecting and connecting enough of suitable such process elements. However, each of the abstract process elements itself will usually need to be configured from smaller abstract or real components. The resources provided by the abstract component determine the resources required of its parts. In many cases, the internal parts of the process elements can be treated like additional components of the whole system, as long as the implications for the scope of the resource-balancing are taken into account. Alternatively, the abstract component can be thought of as the environment for its internal component, and configured in isolation.

In both cases, the configuring process should be structured into hierarchical layers, and the configuring done successively with knowledge bases for resources and components appropriate for just one level.

An application domain well suited to this hierarchical configuration approach is the construction of conveyor-belt systems. Figure 7 sketches the multiple layers and illustrates the component relationships typically encountered:

Figure 7. Resource-Based Multi-Level Hierarchical Configuring of a Conveyor Belt System starting with the Process Task
The requirement specification is given on the Process Level in form of a "Transport Task" Process and its partial transport resource requirements, e.g. transport distance at a certain velocity, or changes in transport direction.

On the Process Level, the required resources are then satisfied by selecting suitable catalogue components, e.g. Belt Segments, until the required transport distance is reached. These components may be thought of as abstract or real. Each Belt Segment then posts its required resources, e.g. mechanical power of certain amount and quality and control of certain type, which is satisfied on the Technology Level by selecting suitable motors and gears or control programs and entailed sensors/actors, respectively.

A quality attribute is often used as a means to express mechanical or geometric constraints at the interface over which the resource is exchanged. A typical example in mechanical design is the exchange of mechanical power between a motor and load. The quality attributes then are the rotational speed of the motor axle, the standard for the shaft coupling and the standard for the flange mounting of the motor. In the example of Fig. 7, the mechanical interface is represented in the quality attribute "Stub" at one level and as the quality attribute "ShaftForm" at another.

The motor needs a certain amount of switchable electrical power at a certain voltage, which is satisfied by selecting a Motor Control Center (MCC) unit of suitable power handling capability, and subsequently satisfying the requirements of the MCC by selecting a MCC Frame to house the unit. The voltage and the frequency of the electrical power are affixed as quality attributes to the exchanged resource "electrical power".

The control program, on the other hand, requires the resources of suitable subroutine software and sensors, which in turn require memory, sensor inputs and CPU power from a control system, e.g. a Programmable Logic Controller. At that level, the quality and power of the resource-based configuration technique has already been established.

With such a complex system, even a resource-based configuration algorithm will get into trouble when the number of component types to choose from gets large enough.

A much more profound problem, however, is that this conveyor belt system is not located at one point, where it doesn’t matter who is supplied by whom, but is distributed over space and has a specific and necessary internal structure, where it is important that the right amount of resource is supplied at the right place.

In systems with a spatial extent or internal structure, the balancing of resources is usually much more intricate than the basic idea conveys. In such systems, it is necessary, but not sufficient that resources are balanced within the whole system. Instead, most resources must be balanced within some scope.

For resources which are involved in establishing "part-of-" or "contained-in-" relationships, which we call "containing" resources due to the role they play in the placement of components, the scope is simply the container (the component which provides the containing resources). For some other resources, it is obvious that they are scoped similarly to containing resources, e.g. mass.

In many domains, like the conveyor-belt transport systems, the service of a technical system does not only depend on the number of components that constitute it but also on the structure in which the components are connected, e.g. the question of which other belt segments are controlled by the control system instance depicted in Figure 7, and which other control system instances control the other belt segments and transport system subsystems.

For catalogue design of such technical systems, the total configuration task has to be partitioned into configuration on multiple levels. For each level, the refinement can be done in one of three ways:

- the system / subsystem is configured out of smaller components from a catalogue, with a fixed structure, as for the "control system level" in Figure 7.
- the resources that the system / subsystem has to provide are decomposed and chunked into separate requirements manually or automatically, as in the "transport task",
- the system / subsystem is replaced by or hierarchically refined into a decomposition of generic components and links, where the components determine how the higher level resources are transformed into the lower level resources and the links determine which component has to provide which resources to which components. This technique will be prevalent on the process level and the technological level.

6. 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 four 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:


