Requirements for Configuring Complex Software-Based Systems

Christian Kühn

DaimlerChrysler AG
Research and Technology
T721, 70546 Stuttgart, Germany
christian.kuehn@daimlerchrysler.com

Abstract

Numerous applications exist already for the knowledge-based configuration of technical systems, however the majority of approaches are confined to the structural composition of the systems, without looking at the system's behavior. The reactive behavior of the produced system plays a central role in the configuration of software-based systems. Therefore this paper proposes to combine knowledge-based configuration methods with system validation and verification technology. The necessary derived demands on modeling techniques and problem solving methods will be presented in this paper.

1 Introduction

The high number of faults in software-based technical systems shows, that – in spite of a multitude of existing methods and supporting systems – the complexity of system development is not completely controlled. The reason is the high time demand and therefore the high cost of system development as well as quality control. In particular, in security critical applications, e.g. in space and traffic technology, airplane and vehicle development, errors can lead to catastrophic consequences.

The possibility of reducing this expense using a suitable automation of routine work, is provided by existing methods of knowledge-based configuration. In particular the automatic or partly automatic execution of validation and verification during the system development, offers an increased potential, to both improve the quality of the configured hard- and software as well as reducing the high expense of quality control.
A large number of successful applications exist already for the knowledge-based configuration of technical systems, e.g. for computer systems, elevators, drive controllers, rolling mills, industrial stirring devices, image processing systems, machine parts, communication systems, aircraft vehicle cabins, liquid crystals and many more (see [Günter and Kühn 99]). The majority of configuration methods are confined to the structural composition of the systems, without looking at their behavior. Reactive system behavior can however be complex in software-based systems and therefore the correct reaction should be guaranteed.

After a short description of an application scenario – the configuration of software-based vehicle electronic systems (sec. 2) – and an introduction to knowledge-based configuration (sec. 3), a description of a proposed method to combine the knowledge-based configuration methods with the technology of system validation and verification (sec. 4) follow. The derived demands on modeling techniques and problem solving methods are presented in sec. 5. The paper concludes with a summary and an outlook (sec. 6).

2 Application Scenario: Configuration of Software-Based Vehicle Electronic Systems

In the field of current vehicle electronics two trends can be clearly identified: The first is noted for increasing the scope of vehicle electronics, the other is noted for the increase in functionality using software. The introduction of software provides the possibility of using universal control systems in future instead of specialized control systems for different functions (e.g. control systems for engines, gears, suspension, etc.). These can

- be programmed easily
- carry out several sub functions at the same time
- be modified later – in other words "reconfigured" (cf. [Kreuz and Roller 99]).

This trend leads to increased demands on the development of the electronic system within the vehicle. On the one hand there will be an immense variety of possible variants, and on the other the variants will be subject to an increasing change through the simple realization of software updates.

The configuration of vehicle electronic systems can carry out two different kinds of tasks:

- The configuration of electronic systems of a vehicle series. In other words, the development of an electronic system, which can be used in every vehicle of a vehicle series, without the need for complex alterations. This means that every permitted option must be considered and consequently all potential variations must be thought of beforehand.
- The individual configuration of electronic systems for each vehicle that is produced.

The first scenario involves a very complicated task, where the electronic system of the vehicle series is configured in principal for maximum equipment. The second case – the individual vehicle configuration – involves a task, which is carried out for a large number of vehicles and therefore requires a high degree of automation. In particular for trucks, where the difference between individual vehicles can be very large, such a procedure can be compelling.

The electronic system configuration should in particular ensure the economic efficiency and optimization of costs, cabling, weight and consider system resources e.g. memory and processor performance. A particular problem arises with the consideration of the communication of control systems over data busses (e.g. CAN) in vehicles (e.g. as partially shown in figure 1). The bus communication becomes less and less clear with the increasing complexity of hard- and software structures. The consideration of the demands on the system's dynamic communication reaction (e.g. throughput or security) plays an essential role in the development support of future electronic systems.

3 Knowledge-Based Configuration

Knowledge-based configuration belongs to the class of synthesis tasks. Under the term configuration, the step by step assembly and parameter setting of objects from an application domain to problem solution or configuration is understood, where certain restrictions and given task objectives should be fulfilled (see [Günter and Kühn 99; Stumptner 97]). The process of configuration consists of a sequence of configuration steps.

A configuration problem comprises the following components:

- A set of objects in the application domain and their properties (parameters).
- A set of relations between the domain objects. Taxonomical and compositional relations are of particular importance for configuration.
- A task specification (configuration objectives) that specifies the demands a created configuration has to accomplish.
- Control knowledge about the configuration process.

A large number of different applications exist already for knowledge-based configuration, particularly in the field of technical systems. An overview of the technologies, applications and systems is given in [Günter and Kühn 99]. Some of the successfully implemented technologies used in knowledge-based configuration are as follows:

- Structure-orientated procedures: The alignment of the problem solving process along a hierarchical given structure of the solution area. The structure is represented in AND-OR-trees, whereby the AND-link and the OR-link show the compositional and the taxonomical relationships of configuration objects respectively. In this way the solution structure can be declaratively given as a graph, which during the configuration process can be expanded step by step. A solution exists after all as an instantiated sub-graph.
- Constraints: Restrictions between objects can be represented and evaluated with the help of constraints. Using a constraint, a relationship between objects and their properties can be specified in a knowledge base and can be evaluated by constraint propagation. Using constraints for configuration requires the incremental accumulation of the constraint network in line with the further development of partial solutions. At a given time, the complete number of constraints to be considered is not known (except in the final solution). The complete evaluation of constraint relations is not

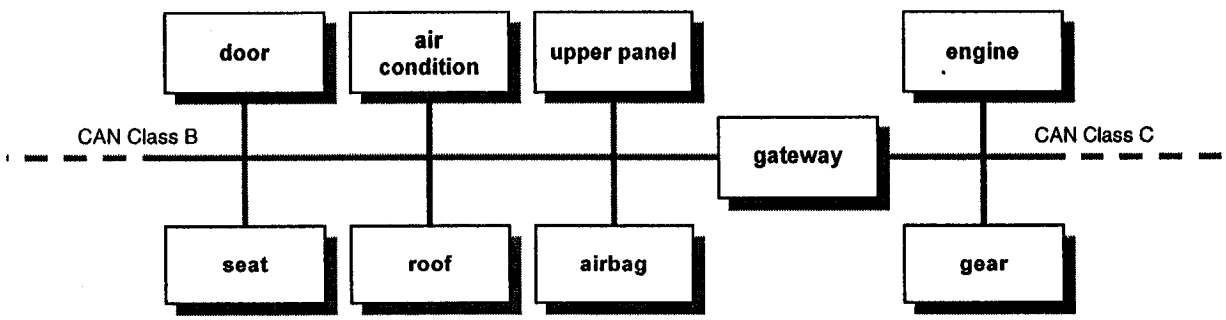

Figure 1: An excerpt from the electronic system of a vehicle: Networking of control units using a CAN-bus system

possible due to the magnitude of the search space. For this reason using constraint propagation rather than constraint satisfaction is to be considered for knowledge-based configuration.

- Case-based Configuration: Case-based problem-solving methods are identified, where knowledge about already solved tasks is saved and is used for the solution of new tasks. Using case knowledge to support configuration does not always have to be concerned with a complete case. The takeover of parts is also possible. The reason being is the assumption that similar tasks lead to similar solutions [Pfitzner 93].
- Resolving of configuration conflicts: Using heuristics means that decisions can prove to be disadvantageous during the course of further problem-solving or can even be incompatible with the rest of the solution parts. In this situation, either the revision of decisions or the repair (that means the direct modification) of a partial solution can be applied to retransform the current partial solution into a consistent state.

More techniques exist beside the aforementioned, e.g. rule-based or resource-based procedure methods, but will not be dealt with further here (see [Günter 95; Günter and Kühn 99] for more information).

A few new methods investigate the integration of simulation technologies in knowledge-based configuration (see e.g. [Bradshaw et al. 91; Günter and Kühn 97; Kühn and Günter 98; Stein 95]). One goal is to carry out simulation restricting sub solutions and calculating values. The other goal is to check the behavior of a complete or partially configured system through simulation, which enables an assessment of the (sub-)configuration.

4 Configuring Complex Reactive Systems using Validation and Verification

The above mentioned approaches for integrating simulation into knowledge-based configuration, principally take the behavior of a configured system into consideration. But these approaches are not sufficient to develop complex reactive and software-based systems, because they are not designed for a high variance of possible system behavior ("state explosion problem", see [Yoneda and Schlingloff 97]). Instead the deployment of formal methods for validation and verification as well as systematic test methods are necessary.

Validation serves to detect faults in the system's specification, inclusive of the requested system behavior. Thus the goal is to find a consistent and complete specification. The automation of the validation process is only partially possible, since there is no other document that could serve as a reference to check the specification (cf. [Peleska and Siegel 97]). Verification should guarantee the correct behavior of the designed system, which has to be checked against the specification. This can either mean a formal verification, i.e. a verification based on formal correctness proofs, or the application of test mechanisms. Using systematic test strategies – although the correct system behavior is not really proved – it is possible to develop sufficient faultless systems (see [Liggesmeyer et al. 98]). Existing approaches applicable for embedded systems provide an extensive coverage of test sequences, which are automatically generated and executed. The increasing number of tests can be seen as a convergence to a full proof (cf. [Peleska 96; Schlingloff, Meyer and Hülsing 99; Peleska and Siegel 97])

Performing validation and verification phases during configuration – not only at the end of the configuration process – provides the possibility to reduce test cases and detect faults and inconsistencies early. In this way validation and verification can act as a basis for making configuration decisions and – as well as guaranteeing the quality of a solution – can lead to an improved configuration process efficiency.

Figure 2 roughly illustrates a suggestion for the integration of validation and verification steps into the configuration process. Here, the configuration process is composed of alternating synthesis (configuration) and analysis (validation and verification) steps. It can be significantly advantageous not to give a complete specification of the configuration task at the beginning, but to refine and modify the specification in parallel to the configuration process. Thus together with the progressing refinement of the configuration solution, the user can also refine the specification for the outstanding solution parts.
Each configuration step can be followed by a verification phase and each refinement or modification of the specification can be followed by a validation phase. As mentioned above, validation can be performed only partially, e.g. for analyzing the existence of contradictions, the incompleteness or execution problems (e.g. deadlocks or lifelocks) in the specification. Besides this, the user should be allowed to change the specification interactively during the configuration process when the user himself detects errors in the specification.
The use of testing methods in combination with formal methods can be beneficial for verification, as the formal verification can be utilized for proving the correct behavior of the safety-critical system parts and the testing methodology can be applied for the residual system. If any incorrectness is found during the verification, the configuration process might run into a conflict handling phase, in which backtracking or directly repairing the incorrect solution parts might be performed.

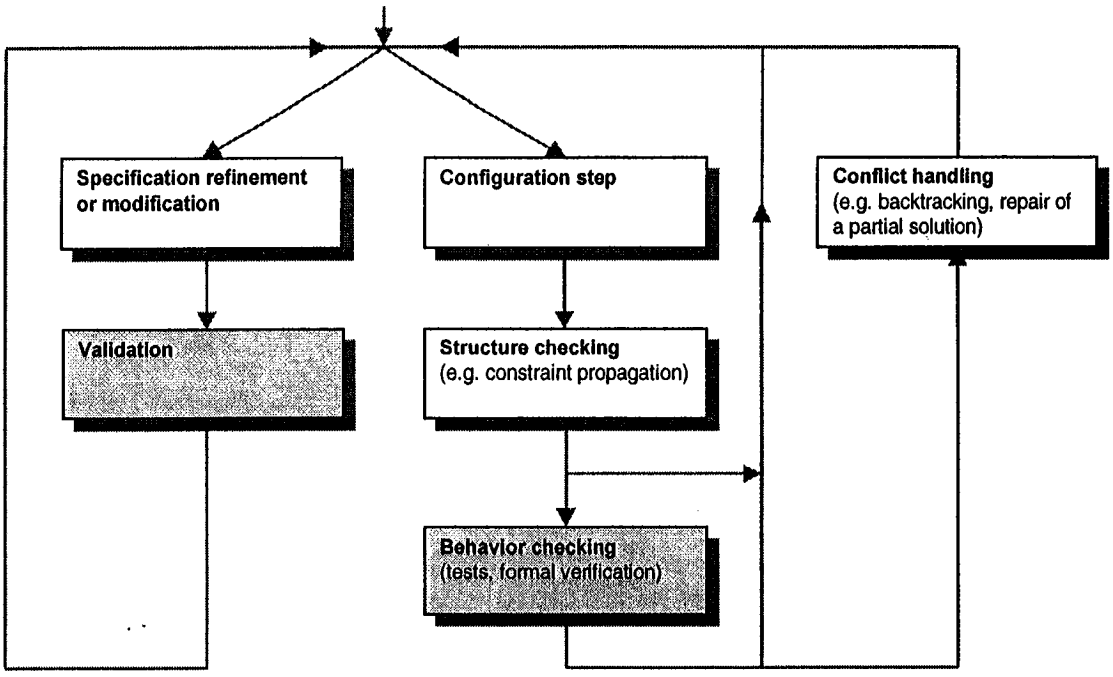

Figure 2: Possible integration of validation and verification phases into the configuration process

5 Demands on Modeling Techniques and Problem Solving Methods

In order to carry out knowledge-based configuration on software-based systems as described above, methods for knowledge representation as well as for problem solving must be provided, which meet the following demands:

- *Modeling of software and hardware*
  A suitable software model is needed, which provides an adequate modeling of software components together with their properties and interfaces. A functional or behavioral description of the software is especially necessary. This should be an abstract or qualitative, but formal representation of the software functionality and dynamic behavior.
  Furthermore hardware components as well as dependencies between the software and hardware structures must be modeled according to configuration.
- *Formal specification of the demanded system behavior*
  Besides the modeling of the demands on the system's static properties (as described for example in [Thärigen 95]), the demands on the system behavior also play an essential part. The safety demands and conditions are important. The requirements to be met in order to process such configuration demands are their formal specification. It might be necessary to register functional as well as time demands (real time demands).
- *Heuristic knowledge for software configuration*
  Appropriate methods for the formulation of heuristic, domain dependent knowledge are needed in order to draw conclusions from specification characteristics resulting in solution characteristics, which include software structures and properties.
- *Integration of validation and verification phases into the configuration control*
  The question of what the collaboration of synthesis and analytic validation and verification processes might look like has to be solved. The control of the configuration process has to be provided with additional control knowledge to handle the integration of validation and verification which is shown in figure 2. This entails the following questions: How can a sufficiently efficient configuration run be achieved in spite of extensive validation and verification processes? When and how often should a validation or verification be activated? Which parts of the configuration will be analyzed? What kind of analysis steps will be made? How can the control of the configuration make use of the results from the analysis processes? How can these results be used to apply conflict solving mechanisms – e.g. different variants of backtracking (chronological, dependency-directed, knowledge-based), repair of partial configurations, interactive conflict solving or acceptance of conflicts (for details see [Günter 93])?

- *Selection of a suitable analysis procedure*
  Knowledge about the conditions for different possible procedures is needed. During configuration it has to be decided, what kind of quality assurance must be applied in different cases. E.g. should a formal security analysis take place in the current state, if at all possible, should statistical testing techniques be applied, should systematic/deterministic checks be performed, or should an exhaustive test be carried out? In general a formal analysis of the entire system cannot be performed. Consequently a formal verification should be limited to safety critical areas.
- *Controlling the validation and verification procedures*
  Furthermore knowledge to control the validation and verification processes is needed, e.g. for test case generation, test execution and the preparation of test results.
- *Quality Statements*
  Rating the quality, i.e. the conformity of the solution with the specification, can be very difficult for the user. E.g. when an engineer allows the configuration to be done partially or even entirely by a configuration system. Therefore a configuration system that produces a quality and reliability statement on the configured (partial) system in line with the configuration itself would be advantageous. According to this knowledge has to be provided.

6 Summary and Outlook

Configuring complex software-based systems, which are able to carry out a reactive behavior – e.g. the software-based electronic systems of vehicles in the future – can be a very sophisticated task. Configuration techniques are required that exceed current configuration approaches, in order to construct a system when the reactive behavior is taken into consideration. In order to do this we suggest the integration of validation and verification techniques into knowledge based configuration. Accordingly, a knowledge representation is needed which allows the suitable modeling of software, the formal specification of the required system behavior, the representation of heuristic knowledge for the software configuration and the modeling of knowledge about the validation and verification methods.

Our work in this context is still in its infancy, but in the future we will attempt to find solutions to solve these problems. Further areas, which are important for the concrete realization of a corresponding configuration methodology – e.g. the integration into existing product data models, workflow concepts, the integration into CAD, simulation and verification systems as well as user interfaces for the maintenance of the different knowledge – will be postponed at this stage.

References

Bradshaw, J.; Young R. M.. 1991. Evaluating Design Using Knowledge of Purpose and Knowledge of Structure. *IEEE Expert* 6 (2):33-40.

Günter, A. 1993. Verfahren zur Auflösung von Konfigurationskonflikten in Expertensystemen. *Künstliche Intelligenz* 7(1):16-22, Germany.

Günter, A. ed. 1995. *Wissensbasiertes Konfigurieren – Ergebnisse aus dem Projekt PROKON.* St. Augustin, Germany: infix Verlag.

Günter, A.; Kühn, C. 1997. Einsatz der Simulation zur Unterstützung der Konfigurierung von technischen Systemen. In Mertens, P.; Voss, H. eds. *Expertensysteme 97 – Beiträge zur 4. Deutschen Tagung Wissensbasierte Systeme (XPS-97)*, 93-106. Bad Honnef am Rhein, Germany.

Günter, A.; Kühn, C. 1999. Knowledge-Based Configuration – Survey and Future Directions. In Puppe, F. ed. *XPS-99: Knowledge Based Systems – Survey and Future Directions, Proceedings 5th Biannual German Conference on Knowledge Based Systems*, Springer Lecture Notes in Artificial Intelligence 1570, Germany.

Kreuz, I.; Roller, D. 1999. Knowledge Growing old in Reconfiguration Context. In *Proceedings of the AAAI Workshop on Configuration.* Orlando, Florida.

Kühn, C.; Günter, A. 1998. Combining Knowledge-Based Configuration and Simulation. In Kleine Büning, H. ed. *Beiträge zum Workshop „Simulation in Wissensbasierten Systemen" (SiWiS-98)*, report tr-ri-98-194, Universität-GH Paderborn, Germany.

Liggesmeyer, P.; Rothfelder, M.; Rettelbach, M.; Ackermann, T. 1998. Qualitätssicherung Software-basierter technischer Systeme – Problembereiche und Lösungsansätze. *Informatik-Spektrum* 21(5):249-258, Germany.

Peleska, J. 1996. *Formal Methods and the Development of Dependable Systems.* Postdoctoral thesis, University of Kiel, Germany.

Peleska, J.; Siegel, M. 1997. Test Automation of Safety-Critical Reactive Systems. *South African Computer Journal* 19:53-77.

Pfitzner, K. 1993. Fallbasierte Konfigurierung technischer Systeme. *Künstliche Intelligenz* 7(1):24-30, Germany.

Schlingloff, H.; Meyer, O.; Hülsing, T. 1999. Correctness Analysis of an Embedded Controller, In *Proceedings of the DASIA-99*, Lisbon, Portugal. Forthcoming.

Stein, B. 1995. *Functional Models in Configuration Systems.* Ph.D. diss., Universität-GH Paderborn, Germany.

Stumptner, M. 1997. An overview of knowledge-based configuration. *AI Com* 10(2):111-126.

Thärigen, M. 1995. Wissensbasierte Erfassung von Anforderungen. In Günter, A. ed. *Wissensbasiertes Konfigurieren – Ergebnisse aus dem Projekt PROKON*, 89-96. St. Augustin, Germany: infix Verlag.

Yoneda, T.; Schlingloff, H. 1997. Efficient Verification of Parallel Real-Time Systems. *Journal of Formal Methods in System Design* 11(2):187-215.