Representation of Failure Context for Diagnosis of Technical Applications

B.D. Netten

Delft University of Technology
Faculty of Information Technology and Systems
Zuidplantsoen 4
2628 BZ Delft
The Netherlands
tel: +31-15-278.3823
fax: +31-15-278.7141
http://www.kbs.twi.tudelft.nl/People/B.D.Netten/
e-mail: B.D.Netten@cs.tudelft.nl

Abstract. Development and maintenance of a case-base is known to be one of the most difficult problems in real-world application of case-based diagnosis systems. The complexity of large technical applications result in large numbers of possible inter- and intra-system failures. Development of fault diagnosis systems for a rolling stock application and a battery emergency backup system showed that a base of fault cases is insufficient to accurately diagnose failures. The context of where and how problems occur should also be considered, such as the configuration and operational conditions of the application. This paper presents a model for the representation of this context for technical applications. The context information is applied for the representation, retrieval and retainment of cases.

1 Introduction

Case-based reasoning provides significant advantages over other techniques for developing and maintaining diagnosis systems. On the other hand, the performance of case-based diagnosis systems, in terms of retrieval time, accuracy and coverage, may be less than for other techniques. For many applications, diagnosis system performance should satisfy minimum requirements before they can be put into practice. Development tools are required to evaluate and improve the performance of case-based diagnosis systems.

There are several strategies for developing case-based diagnosis systems. A typical case-based reasoning approach is to retain a sufficient number of cases from the real world from which performance can be improved afterwards. Development tools or procedures are needed to support the development of a vocabulary (e.g. [1]), abstraction or deletion strategies ([2]), or induction of relevant features (e.g. [3]). Another approach is to first develop a model of the diagnosis system and then acquire

In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

diagnostic knowledge in the form of cases (e.g. [4]). This approach is appropriate when a history of real-world cases is not a priori available and the application is too complex to develop an accurate behavioural model.

In technical applications, domain knowledge is available from the three sources mentioned above; i.e. real-world cases, analytical cases, and domain models. Combination of these sources provides additional background information to support development and maintenance of diagnosis systems, and to evaluate their performance.

This paper presents a partial domain model for the configuration and operational conditions for technical applications. The next section briefly presents the BRIDGE approach to improve the efficiency of a diagnosis system. The remainder of the paper presents the development of the domain model (Sections 3 and 4), and the evaluation of the competence of diagnosis systems (Sections 5 and 6).

2 Case-Based Diagnosis in BRIDGE

The objective for the BRIDGE project[i] is to improve performance of operational diagnosis systems as a means to improve safety, availability, reliability, maintainability and life cycle costs of large technical applications. Improved performance in operational diagnosis comprise a significant increase in the level of diagnosis automation, coverage of problems, accuracy of diagnoses, and in ensuring guaranteed response times for critical on-line diagnosis situations. The initial two-step approach in [5] is further developed in the BRIDGE project (Fig. 1).

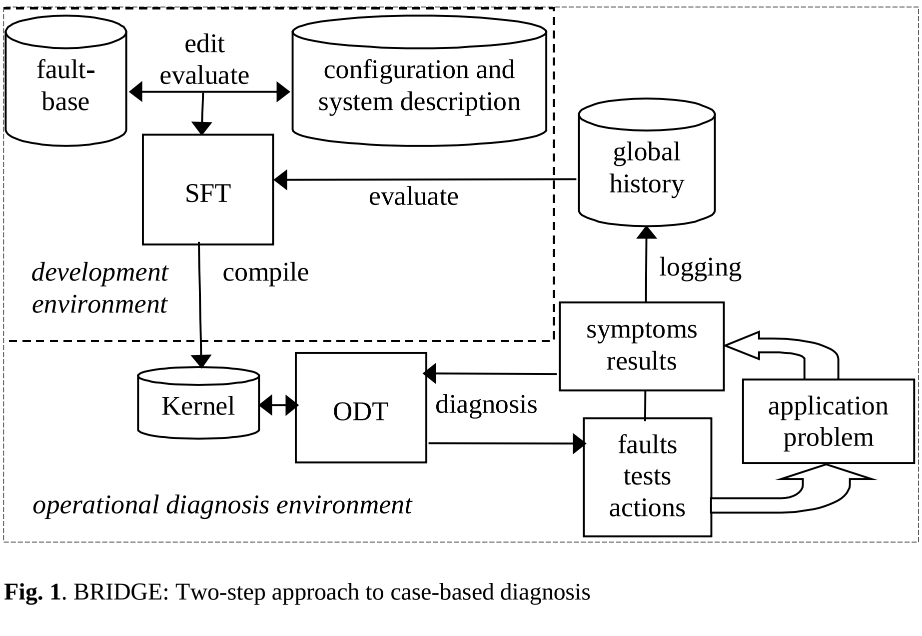

Fig. 1. BRIDGE: Two-step approach to case-based diagnosis

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

The first step addresses the development and maintenance of three knowledge bases; a *Fault-Base* with analytical cases, the *Configuration and System Description* with a partial model of the application, and the Global History with real-world cases. The *Support Function Tool* (SFT) contains a set of tools to edit and evaluate these knowledge bases. The SFT also compiles this knowledge into a *Kernel*, which is applied solely for operational diagnosis.

The *Operational Diagnosis Tool* (ODT) performs the on-line and real-time diagnosis of the application. The ODT receives input in the form of symptoms and test-results describing a new problem. The ODT diagnoses this problem and suggests lists of possible faults, additional tests and recovery actions. All input and diagnoses are logged in the global history. To guarantee real-time response, the ODT only calls the kernel with compiled knowledge. The ODT cannot retrieve or adapt diagnoses from the three knowledge bases, or communicate with the SFT.

The performance of the ODT can be predicted from properties of the fault-base and configuration and system description. The competence of the ODT depends on the knowledge being compiled from the SFT. Diagnosis efficiency is determined by the ODT. This paper addresses the evaluation of the competence of the knowledge bases in the SFT. A more detailed presentation on the ODT and its performance is given in [6].

3 Applications

The BRIDGE project particularly targets on diagnosis of large industrial applications. Two industrial applications are currently being developed in the BRIDGE project.

3.1 Central Advice System for Rolling Stock

Bombardier Transport Division BN produces and maintains railway transportation systems, such as trams and trains. The systems on board of transport vehicles are becoming more complex, intelligent and interdependent, due to the need for more comfort, higher availability, better maintainability, lower life cycle cost, and higher efficiency. A Central Advice System (CAS) is being developed to support fleet and maintenance management for one of BN’s customers. The application is presented in more detail in [7].

The ODT in CAS support drivers, maintenance teams, and fleet managers. The information in CAS is specific for operational conditions.

- While a tram is in service, a driver is only notified for events that are critical for safety or the service schedule. In most situations quick intervention is needed to be able to continue the journey or to keep the vehicle in service until the next maintenance activity. The available resource (time, tools and skills) for additional tests and actions are restricted.

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

- Maintenance teams use the ODT to diagnose failures during troubleshooting, preventive or corrective maintenance. Resources are available to either temporarily or completely resolve a problem.
- Managers receive information about vehicle and tram health states, impact on service. Fleet management uses this information for more accurate allocation of trams and vehicles for service. Maintenance management receive information on required personnel, time, tools and spare parts for interventions. They can also improve their short term planning of corrective and preventive maintenance. On long term, this information is used to forecast and optimise resources, such as stock, parts, staff, and the workshop.

3.2 Battery Monitoring System

DACS is developing a system for diagnosis and monitoring of large and distributed lead storage batteries. These battery systems are applied for generating stations or emergency back-up energy. The reliability and performance of these batteries are critical and can be optimised when given proper and regular maintenance. At regular intervals, the batteries are inspected for voltage of the battery and of individual cells, cleanliness, electrolyte levels, ambient temperature, etc.

A battery unit consists of several battery systems, each of which is made up of many cells. Battery units are installed at geographically different places. The configuration of each battery unit and type of batteries and cells can be different. Inspection of batteries is a tedious, time-consuming and accurate job when performed manually. The centralised battery monitoring system should perform these inspections remotely, automatically, accurately, and at regular time intervals. The system should also perform a diagnosis to identify when, where and what maintenance is required.

The system is developed as a client-server application. A client runs at every site, and remotely logs in on the server. Clients are implemented in Excel and connected to the data acquisition devices. BRIDGE is implemented in the server to perform the diagnosis and control the inspections for different clients, battery types and configurations, operators, etc.

3.3 Large technical applications

In [5] a prototype for the rolling stock application is presented in which cases are defined with features for the causal effects of faults; i.e. symptoms and test-results. The rapidly evolving developments in technical systems strongly increase the complexity of diagnosis problems. Managing these problems becomes very difficult. The complexity for diagnosis problems results from the complexity in the configuration and operational conditions of an application. For each problem that appears, several possible causes exist that are often linked with multiple systems and with a high impact on further service.

Experience with the prototype showed that the causal effects provide insufficient information. The proper recovery actions not only depend on the causes, but also on

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

their impact on safety and service, the operational status, and the availability of resources.

Industrial applications are complex configurations of many components. Each component is characterised by its type, position in the configuration, its assemblies and parts, symptoms, faults and failure modes, etc. Information about the configuration is important to accurately locate the place of causes and effects, and to distinguish between inter- and intra-system failures.

An operational condition defines the conditions in which a problem is diagnosed and resolved. An operational condition defines the status of the application and the availability of resources to perform the diagnosis and recovery actions. The accuracy of a diagnosis system, and the relevance of hypotheses, is seriously affected if tests or actions cannot be performed under the given operational conditions. The objectives, time constraints, and available information for diagnosis may significantly change over time.

The similarities and differences in configurations and operational conditions do provide essential information to evaluate and improve the accuracy of diagnosis. The configuration and operational conditions define a context for retrieving and storing cases; i.e. they determine the relevance of cases and their features (symptoms, tests and actions). The context extends the vocabulary and structure of the case-bases. The context also provides a partial domain background for evaluations (Section 5 and 6).

4 Knowledge Representation

An object model was developed in the BRIDGE project to represent the three knowledge bases mentioned in section 2. Fig. 2 gives a simplified OMT model [8].

4.1 Cases

Fig. 2 presents cases in the lower left dotted box. Two types of cases exist.

- A **failure case** defines a real world problem. The case contains the effects of a problem as a set of symptoms and test results, a set of recovery actions that have been performed, and a diagnosis of the fault. Failure cases are defined in the global history for real-world problem cases.
- A **fault case** defines an analytical can for a fault. The case defines the complete lists of all possible causal effects and recovery actions. Fault cases are defined in the case-base. Fault-cases are abstractions of failure cases. The complete list of effects covers multiple failure cases for the same fault.

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

4.2 Context of operational conditions

Following operational conditions have been generalised from the two industrial applications:

- A *health st*ate defines the impact of a problem on the safety or functionality of an application. Health states are monotonically ordered by the severity of their impact, e.g. "normal", "degraded functionality", "alarm". Health states determine the priority for hypothesising on faults, tests and actions in critical situations.
- An *operational mode* defines the status of the application, e.g. "stand-by", "in-service", or "out-of-service". The operational mode determines when symptoms, tests or actions can be obtained or performed, whether resources are available to diagnose and resolve problems, and whether faults are critical.
- The availability of resources to perform tests and actions. Resources are defined for the level of skill or authorisation of personnel, and the available time and cost to perform the tests and actions. The accuracy of a diagnosis system may be seriously affected if the necessary resources are not available in critical situations.

Operational conditions describe part of the system, or application, and are defined in the *system description* in BRIDGE. Fig. 2 also presents the main associations between the case-base and the operational conditions:

- The fault of a case has an impact on the health state of the application.
- A symptom is associated to the worst impact on the health state caused by any of its associated cases.
- Tests or actions can be performed in one or more operational modes, by one or more operator levels, and can only be performed with zero or more tools.

4.3 Configuration

The configurations in both applications consist of many similar types of components[1], such as vehicles, systems, modules, parts, batteries and cells. Components of the same type may generate the same set of symptoms, exhibit a common set of faults, and a common set of tests and actions can be performed on them. Component types also share possible aggregations; i.e. assemblies and parts.

Upon configuration of components, they are instantiated from their component types, and positioned in a unique aggregation. The place in this configuration further determines the behaviour of the component. This is represented by the definition of the place of their associated cases and their effects.

The complexity resulting from the configuration is typical for technical applications considered in BRIDGE. The identification of the place of occurrence of cases and their effects enable to diagnose inter- and intra-system failures.

---

[1] The word "component" or "sub-component" will be used to identify an assembly or part in the aggregation.

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

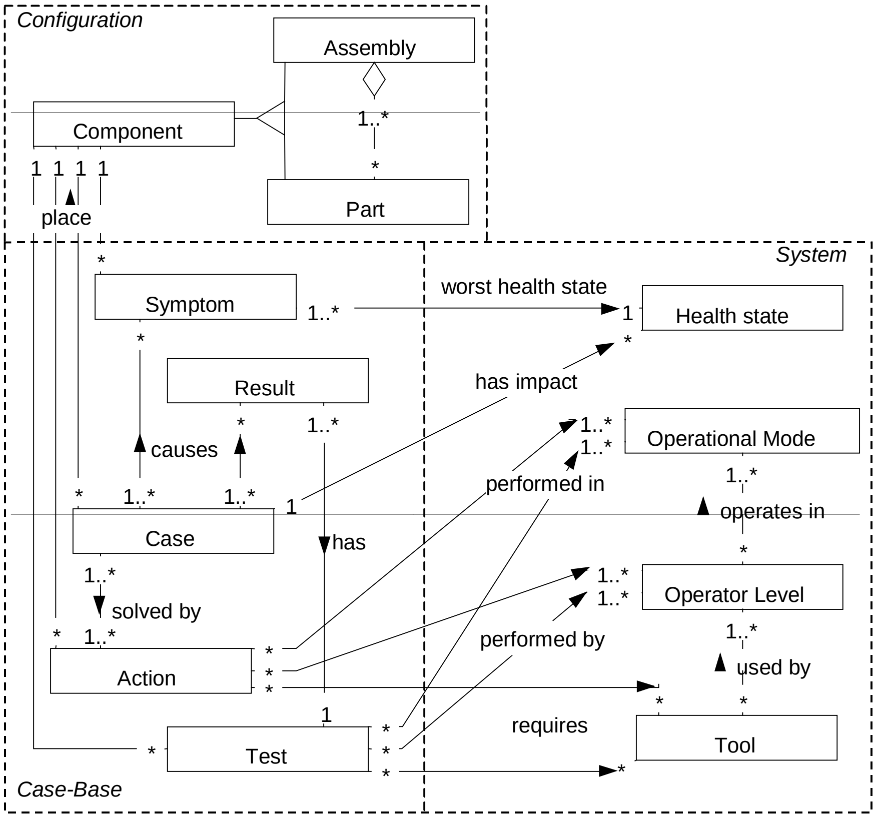

Fig. 2. Object model of the cases and context

5 Development and Maintenance of Diagnosis Systems

The SFT supports the development and maintenance of diagnosis systems. The SFT contains several tools. Editors enable the definition of all items and relations of Fig. 2 in the knowledge base. Evaluation tools are developed for data integrity, competence, and prediction of the ODT's response time. A Tool is provided for automatic generation of kernels for the ODT. A fault-base is development in three steps:

1. First, the configuration and system description is defined for a new application. This is a relatively easy task, assuming that the configuration and operational conditions of the application are well known.
2. The definition of components is extended for symptoms, tests and actions.
3. Faults are defined in fault-cases.

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

The data model, of which Fig. 2 is a simplification, contains many complex relations. The integrity of the knowledge bases has to be evaluated for these relations as a pre-condition for diagnosis performance and for the other evaluations. A model is built for the relations in the knowledge bases, such as the configuration, prioritised lists of health states, hierarchies of operator levels, and the context for symptoms, tests, results, actions and cases. The places of causal relations are also evaluated against the configuration and the places of fault cases, symptom, and test. The model also serves as a reference model for the generation of the ODT kernel.

6 Competence of a diagnosis system

This section presents the usage of the configuration and system description for evaluation of competence of the ODT. The competence of the ODT depends on the knowledge compiled from the case-base in the SFT. Specialised diagnosis systems can be built for the ODT by selectively compiling parts of the case-base. The case-base is filtered for specific operational conditions and parts of the configuration. The competence of the ODT depends on the coverage of the case-base, and the reachability of faults and actions, after filtering. This section considers a case-base after filtering. It should be noted that the coverage and reachability change with the operational conditions of a filter.

[9] defined the coverage and reachability of cases to evaluate the competence of retrieval and adaptation. Similar evaluation of competence for BRIDGE requires a redefinition in the context of a retrieval system and the filtering of operational conditions.

6.1 Coverage of a fault-case

A fault-case is an abstraction of a set of failures, which are caused by the fault and can be resolved by the same action. A fault-case defines the fault, its recovery action, and its set of symptoms and test-results should cover a set of failures. The certainty factor associated with each feature (symptom, test-result) identifies its relevance for the fault-case. The fault-case is also retrieved for new problems with a subset of symptoms and test-results or for which not all tests have been performed.

A fault-case thus covers as set of failures with a subset of symptoms and test-results, that are caused by the same fault and can be resolved by the same action. Given a fault-base **C** of fault-cases. Given a fault-case $\mathbf{c} \in \mathbf{C}$, defining fault **f** and recovery action **a**, causing a set of symptoms and test results that can be observed under these operational conditions **I**. The coverage of fault-case **c** is defined by a set of failures **p** in the problem space **P**:

$$\text{Coverage}(\mathbf{c}) = \{ \mathbf{p} \in \mathbf{P} \mid \mathbf{f}, \mathbf{a}, \mathbf{i} \in \mathbf{I} \} \tag{1}$$

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

This definition is valid for a given filtering of the operational conditions, and assumes the same place for faults, actions, symptoms and tests. The union of the coverage of all fault-cases defines the coverage of the case-base.

The objective for diagnosis can also be to suggest recovery actions. Actions can be defined for multiple fault-cases. The coverage of an action can be defined as the union of the coverage of its supporting fault-cases.

6.2 Reachability of faults, actions and failures

A consequence of filtering operational conditions is that fault-cases may not be distinguished from other cases and that their faults and actions cannot be retrieved. This could occur when additional tests to refine diagnoses cannot be performed, or when actions cannot be performed under the given operational conditions. The reachability of a fault **f** and action **a** can be defined, under the conditions as in Sect. 6.1, where **I** is the set of symptoms and test-results that can be obtained, and **A** is the set of actions that can be performed, under the given operational conditions:

$$\text{Reachable}(\mathbf{f}) = \{\ \mathbf{c} \in\ \mathbf{C} \mid\ \mathbf{f}, \mathbf{i} \in\ \mathbf{I}\ \} \tag{2}$$

$$\text{Reachable}(\mathbf{a}) = \{\ \mathbf{c} \in\ \mathbf{C} \mid\ \mathbf{a} \in\ \mathbf{A}, \mathbf{i} \in\ \mathbf{I}\ \} \tag{3}$$

Another criterion is the reachability of failures under given operational conditions. Upon evaluation of failure-cases from the global history, the set of symptoms and test-results of a failure-case should be observable, and fault-cases should be available in the filtered case-base. The reachability of a failure-case **p** can be defined under the conditions of above as:

$$\text{Reachable}(\mathbf{p}) = \{\ \mathbf{c} \in\ \mathbf{C} \mid\ \mathbf{f}, \mathbf{a} \in\ \mathbf{A}, \mathbf{i} \in\ \mathbf{I}\ \} \tag{4}$$

6.3 Evaluation of faults and failures

During development and maintenance, modifications to the fault-base are evaluated to preserve or improve competence. Evaluations distinguish two approaches.

Retaining failure-cases from the global history. The fault-base can be evaluated for every new failure-case from the global history. The fault-base can be updated in a process similar to the deletion strategy presented in [9]. Coverage and reachability criterion mentioned above can be improved by extending the set of symptoms and test-results for a fault-case or by insertion of a new fault-case.

It is important to note that the performance can also be improved. Complex failure-cases implicitly relate symptoms, test-results, faults, and actions over several operational conditions. New symptoms or test-results could be identified that can be observed in the first operational modes (e.g. in service). Extensions of fault-cases for these features will speed-up diagnosis in the future.

Evaluating manual modifications to the fault-base. Knowledge engineers can also modify fault-cases directly. Failure-cases are not available for evaluation of competence and the above mentioned criteria should be simplified accordingly. The coverage of a fault-case can only be evaluated for its set of symptoms and test-results, while reachability can only be evaluated for faults and actions. The reachability criteria will assure that fault-cases can still be diagnosed if one or more symptoms or test-results are eliminated. To preserve the coverage, fault-cases are deleted only when they are covered by other cases. In practice, symptoms and test-results will only be eliminated.

7 Current Status

The BRIDGE tools are initially developed for PC's running Windows'95 or WindowsNT. The tools are implemented in C++, the database is implemented in SQL-like relational database Access and linked through ODBC. In April 1998, the development of the BRIDGE alpha version is nearly finished. The integrity evaluator and the model for competence evaluation are already available. Evaluators for competence and performance of diagnosis systems will be available for the beta version in September. It should be emphasised that the BRIDGE diagnosis environment (ODT) is currently implemented on WindowsNT to simplify initial development. For real-time industrial application, the ODT has to be implemented in an embedded environment with a real-time operating system.

8 Related Work

BRIDGE is not a hybrid system with a model- and a case-based reasoning module, like for example in ADAPtER ([10]). BRIDGE uses only part of the domain model for the configuration and system descriptions, which are the easiest parts to develop for large technical applications. The case-base is used as the knowledge source to compensate for the omission of a behavioural model.

BRIDGE does not support adaptation like in ADAPtER or INRECA ([11]). On-line adaptations may result in incorrect diagnoses or invalid actions, and cannot be allowed in safety or time critical applications. It is entirely up to an operator to perform or adapt actions.

CASUEL is a common case representation language in which and object oriented domain model could also be defined, together with specific rules for case completion, similarity and adaptation ([12]). CASUEL is more generally applicable than the objectives set out in BRIDGE. The configuration and system description might also have been defined in a set of text files. In the BRIDGE project, however, existing SQL-like relational databases are preferred over text files to develop and maintain the case-base and the configuration and system descriptions.

The fault-base is developed initially from abstract cases, while real-world cases are retained afterwards to improve the abstract cases. Good performance and competence is obtained throughout the development process and avoids major efforts for quality improvement presented in [1].

8 Conclusions

BRIDGE is a tool for the development of real-time diagnosis systems from a case-base of faults and failures. This paper presented part of the tool for the development and maintenance of the case-base. Real-time diagnosis is performed in a separate tool, which is not the subject of this paper.

Evaluation of the integrity, competence and performance of case-based diagnosis system are considered essential for real-world application. Additional domain knowledge is required for such evaluations. For technical applications, the configuration and operational conditions of the application provide the necessary information and can be acquired relatively easy. A model for the configuration and operational conditions of technical applications has been presented. The model enables evaluations that improve the quality of diagnosis systems; i.e. integrity conflicts are avoided, competence is improved and evaluated for all operational conditions, and response times can be guaranteed.

References

1. Heider R., Auriol E., Tartarin E., Manago M.: Improving the quality of case bases for building better decision support systems. In: Bergmann R., Wilke W. (eds.): 5th German Workshop on Case-Based Reasoning, Bad-Honnef, (1997)
2. Smyth B., Cunningham P.: The Utility Problem Analysed. In: I. Smith and B. Faltings (eds.): Advances in Case-Based Reasoning, Springer Verlag (1996) 392-399
3. Aha D., Breslow L.A.: Refining conversational Case Libraries. In: D.B. Leake and E. Plaza (eds.): Case-Based Reasoning Research and Development (1997) 267-278
4. Trott J.R., Bing Leng, An Engineering Approach for Troubleshooting Case Bases. In: D.B. Leake and E. Plaza (eds.): Case-Based Reasoning Research and Development (1997) 178-189
5. Netten B.D., and R.A. Vingerhoeds: Large-Scale Fault Diagnosis for On-Board Train Systems. In: Veloso M., Aamodt A. (eds.): Case-Based Reasoning, Research and Development. Lecture Notes in Artificial Intelligence 1010, Springer Verlag, Berlin (1995) 67-76
6. Naks T., Motus L., Holt J.: Timing Analysis in OO System Life-cycles. In: Proc. of the first IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing. April 20-22, Kyoto (1998) 327-334
7. Coen L. De, Netten B.D., Vingerhoeds R.A.: Cental Advice System for Fleet Management and Operations. In: Developments in Mass Transit Systems, IEE Conference Publication No. 453, London, 22 April (1998) 315-320

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)

8. Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W.: Object-Oriented Modeling and Design, Prentice Hall, New Jersey (1991)
9. Smyth B., Keane M.: Remembering To Forget. In: 14th Int. Joint Conf. on AI, Montreal, Canada (1996)
10. Portinale L., Torasso P.: On the Usefulness of Re-using Diagnostic Solutions. In: ECAI'96 Workshop on Adaptation in Case-Based Reasoning (1996)
11. Wilke W., Bergmann R.: INRECA: Induction and Reasoning from Cases. In: ECAI'96 Workshop on Adaptation in Case-Based Reasoning (1996)
12. Manago M., Bergmann R., Conruyt N., Traphörner R., Pasley J., Le Renard J., Maurer F., Weβ S., Althoff K.-D., Dumont S.: CASUEL: A Common Case Representation Language., ESPRIT project 6322 deliverable D1, ftp://ftpagr.informatik.uni-kl.de/pub/CASUEL/Documentation

[i1] This research is partially funded by the European Commission, ESPRIT project 22154, BRIDGE Real-Time Intelligent Diagnosis Tool for Large Technical Applications. The author would like to acknowledge the other project partners for their contributions; Bombardier Transportation Div. BN, Data Acquisition and Control Systems, MiTime Ltd., University of Wales Swansea, and Tallinn Technical University. The LIMITS project was funded by the European Commission as project CP-94-1577.

Representation of Failure Context for Diagnosis of Technical Applications, B.D. Netten
In: Advances in Case-Based Reasoning, B. Smyth, P. Cunningham (Eds.)