7th German Workshop on Case-Based Reasoning, Würzburg, March 3-5, 1999, pp. 120-130.

# Verification of Case-Base Integrity in BRIDGE

B.D. Netten

Delft University of Technology,
Faculty of Information Technology and Systems,
Zuidplantsoen 4, 2628 BZ, Delft, Netherlands
http://www.kbs.twi.tudelft.nl/People/Staff/B.D.Netten
B.D.Netten@cs.tudelft.nl

**Abstract.** The performance of a case-based diagnosis system should be validated before it can be applied in a critical operational environment. This paper proposes a framework for a validation process of case-based diagnosis systems. This framework is developed from existing work on expert system validation and on evaluation and improvement of the performance of case-based reasoning systems. First results on implementation in the BRIDGE[1] project are also presented.

## 1 Introduction

Diagnosis systems play an essential role in maintaining the safety and functionality of applications. The performance of a diagnosis system has to satisfy minimum requirements. For example, the system must accurately diagnose failures that have a severe impact on the functionality and safety of the application. Before a diagnosis system can be applied, its has to be validated to satisfy the minimum performance requirements.

Case-based reasoning (CBR) is applied successfully for diagnosis in industrial applications. Case-based reasoning has several advantages over other techniques like model-based reasoning. Reasons for their initial success are the minimal initial efforts required for developing an operational system and their applicability in weak domains. At the same time, the performance of an initial diagnosis system may be poor, and substantial effort is required to improve the quality of the case-base and the retrieval and adaptation mechanisms. Research has been reported on a number of issues related to case-based reasoning performance. Integration into a framework for the validation of the performance seems to be missing though.

Validation has become an important issue in the BRIDGE[1] project. Initially, the set of performance requirements grew rapidly in the early phase of the project and demanded an automated and more formal approach to the development and maintenance of case-based diagnosis systems. The validation process also provides objective criteria to retain knowledge from the case-base. This paper proposes a framework for the validation of case-based diagnosis systems. The final section presents the first step of this framework in BRIDGE tool: the verification of knowledge base integrity.

# 2 Related Work

## 2.1 Validation of Knowledge Based Systems

This section summarises issues and terminology for database and knowledge-based system validation that will be reused in following sections.

**Database Integrity.** Databases are validated on the consistency of data and relationships with a set of integrity rules. Three types of integrity rules are identified in [13]: attribute, entity, and referential integrity rules. Attribute integrity rules define constraints for the validity of attribute values. Entity integrity rules define how table indices should be defined and maintain the uniqueness and irreducibility properties of database entities. Referential integrity rules define the integrity of relations between tables.

**Validation process of knowledge-based systems.** The terminology for validation of knowledge-based systems from [14] will be used here. The quality of a system is perceived as a set of concepts, which are specified in a set of measurable criteria. *Validation* is the process of verifying and evaluating a system for the predefined specifications.

*Verification* is the process of validating a system for the formal specifications, or in other words [15], the process of checking that the system is built in the right way. Verification addresses concepts that can be formally specified, such as redundancy, consistency, and completeness. These specifications can be measured unambiguously and hence, verification can be automated.

Concepts that cannot be formally specified, such as user-satisfaction and explanation facilities, cannot be measured unambiguously. If heuristic measures exist, pseudo-formal specifications can be defined. Alternatively, test examples can be defined. Evaluation is the process of validating a system for these pseudo-formal specifications and tests. Evaluation requires human judgement or acknowledgement.

System quality is affected by internal errors in the knowledge base and external errors as they are perceived from a user’s point of view. Structural verification is the process of verifying the knowledge base for internal errors. Functional verification is the process of verifying the system for external errors. The externally experienced errors result from internal errors. External inconsistencies for example result from internal errors or from a paradox in the real world. Many external errors can thus be avoided through structural verification.

**Approximate Reasoning.** Verification becomes more difficult for approximate reasoning systems in which more redundancies and inconsistencies may be perceived. For example, non-redundant rules may have such small effects on the conclusions that they are perceived (informally) as redundant. "Adequacy criteria" are introduced in **[15]** for functional verification, which can be interpreted as:

- Results are significantly different if they comprise different sets of selected goals.
- Every rule should be effectively tested. An effective test for a rule yields significantly different results when the rule would be removed from the KB.

Evaluation for the adequacy criteria requires laborious testing and user involvement.

Specifications can be verified either within the KB (syntactically) or with additional knowledge (semantically) such as facts, domain models or general knowledge [16]. The next section presents the type of specifications defined in the BRIDGE diagnosis system. These specifications require both syntactical and semantical verification.

The validation process can also be applied to revise the knowledge base. For (semi-) automatic revision of detected errors, it is important to distinguish revisable from indisputable knowledge [16].

### 2.2 Performance of Case-Based Reasoning Systems

The performance of case-based reasoning systems depends on the size and coverage of the case-base, and the effectiveness of retrieval and adaptation mechanisms (e.g. [1]). In many applications, a review committee evaluates system performance and revises the case-base (e.g. [2]). This manual process is laborious and error prone. Formal verification procedures are necessary for validation, conflict checking and revision.

Various aspect about the validation of performance have been reported individually. Methodologies for the development process [3] and maintenance process [4] integrate much of this work, but seem to concentrate on evaluating and improving the performance rather than validating it. Individual aspects have been reported for example on empirical performance evaluation (e.g. [4], [5], [6]), performance improvement by improving the vocabulary (e.g. [2]), abstraction or deletion strategies (e.g. [6], [7]), induction of relevant features (e.g. [8]), or the acquisition of cases from domain models [9]. Refiner(+) [10] verifies and refines a case-base for consistent classification. A next step is to develop an integrated framework for the formal verification and validation of performance.

## 3 BRIDGE Case-Based Diagnosis System

This section presents the case-based diagnosis system for large technical applications from the BRIDGE project [11]. BRIDGE is an integrated case- and model-based diagnosis system.

The case-base in BRIDGE can be divided in two parts:

- A fault-base containing analytic cases for faults, which are generalizations of failure cases.
- A global history of real-world failure-cases.

A case represents a problem or fault, and contains:

- The cause (*fault*)
- The effects or observables (*symptoms* and *test-results*)
- *Actions* to resolve the cause or reduce its effects on functionality and safety.

To improve diagnosis performance, a context model is developed, which defines a vocabulary and structure for case representation. The model is also used for case indexing and similarity. The knowledge base in BRIDGE contains the context model and the case-base (Fig. 1 is a revised version of the object model from [11]).

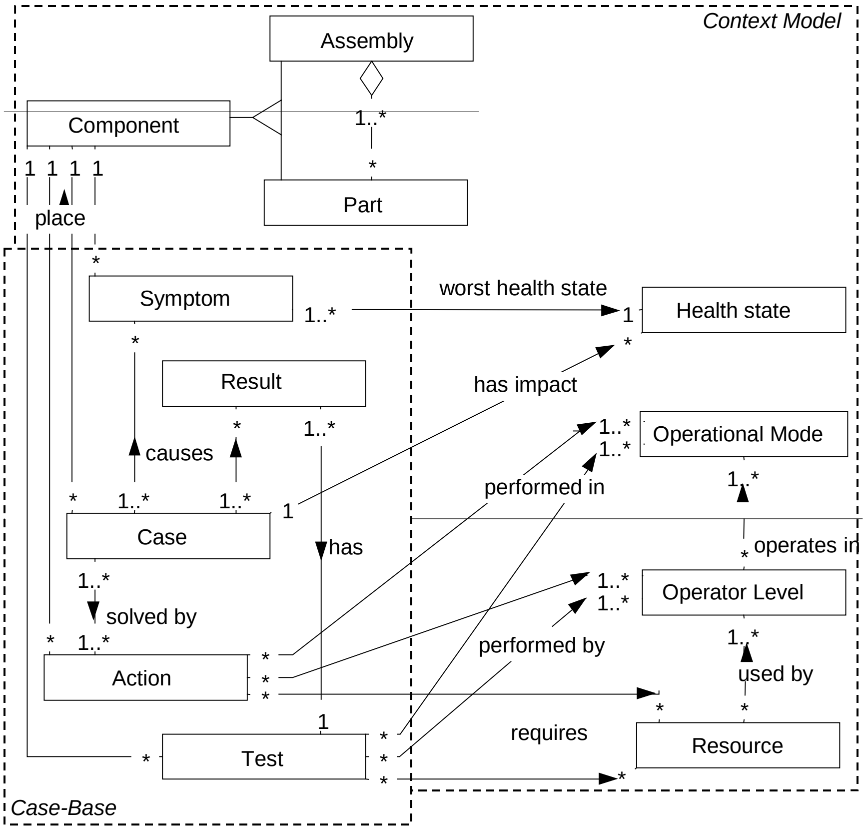


**Fig. 1: Object model of the context model and cases**

The context model is a partial domain model for the configuration and operating conditions. The configuration model allows decomposition of diagnosis problems and identification of inter- and intra-component problems. An operating condition defines the condition of an application during on-line diagnosis and repair. An operating condition is defined by the operational mode and health state of the application, and the operator level and available resources for diagnosing and repairing the application. An example of a critical operating condition of a train is: operational mode = Driving while in Service, health state = Alarm level, operator = Driver.

On-line diagnosis is performed in an iterative process of four steps:

1. Identification of the operating conditions and place in the configuration.
2. Input of observed effects and retrieval of possible diagnoses.
   Input is obtained from the effects observed during the current or a previous operating condition. A cause is reused as a diagnosis of the problem.
3. Additional input is requested to increase diagnosis accuracy; i.e. to confirm or differentiate diagnoses.
4. Actions are reused as a (temporary) solution to the diagnosed problem(s).

Diagnosis is performed in a changing environment when the operating conditions change over consecutive iterations.

### 3.1 Performance Specifications for the Case-Based Diagnosis System

Performance requirements are specified as functional requirements for the on-line diagnosis efficiency and competence.

Efficiency depends on the redundancy and decomposition of the knowledge base. Redundancy is reduced in three ways:

- Fault-cases are indexed and retrieved rather than failure-cases.
- Case features are defined uniquely and only referenced in cases.
- Generalization of components and their aggregation.

The knowledge base is hierarchically organized allowing a reduction of the problem space to parts of the configuration and current operating conditions. The response time for diagnosing a set of observations is a measure for system efficiency that can be guaranteed in BRIDGE [12].

System competence is specified by integrity, reachability, accuracy, and coverage criteria:

- The integrity, or consistency, of the knowledge base with the object model is assumed. The integrity requirement is discussed later.
- A problem can be solved correctly if the problem is reachable, according to the definition of reachability in [7]. This functional criterion can be redefined to structural criteria [11]. A problem is reachable if:
  - A fault-case with the correct diagnosis and actions is reachable in the current operating condition, and
  - The actions can be determined with sufficient accuracy.

  The definition of case reachability is modified in [11] from problem reachability in [7]. A case is reachable if the observables distinguish it from other cases. The

current and previous operating conditions determine which inputs can be observed, and thus restrict the reachability of cases. The reachability of diagnoses and actions depend on the reachability of cases.
In BRIDGE, actions are reused from cases. Appropriate actions are determined from case similarity, which depends on the reachability of cases and the similarity measure. Accuracy is thus a more stringent criterion than case reachability. Accuracy is defined by a minimum similarity value for the case with the correct diagnosis and action, and a minimum difference in similarity to any case with another diagnosis and action.

- Coverage is measured in two ways:
  - The fault-cases should cover all case features and a subset of the entities in the context model. The fault-base is incomplete, or the context model is redundant, if for example:
    - An observable or action is not referenced in any fault-case.
    - A leaf component in the configuration tree is not covered in one or more fault-cases.
    - A health state (except the normal health state) is not covered in any case.
  - All real-world failure-cases are covered by one or more fault-cases.

Performance requirements are specified for one or more operating conditions. For the example of a critical operating condition above, it is required that:

- All fault-cases with “Alarm level” are reachable with sufficient accuracy.
- All failure-cases causing a health state = “Alarm level” should be covered in the fault-base.
- Maximum response time is guaranteed.
- Only those symptoms, tests, actions and diagnoses are presented that are relevant, necessary and that can be performed under the given operating conditions.

# 4 An Approach for Validation of the Case-Based Diagnosis System

The validation processes for database and rule-based systems can also be applied in a modified form for the validation of case-based diagnosis system performance as specified earlier. Validation of case-based reasoning systems is usually not possible, because the necessary information is either not available or specified. Instead, the performance is evaluated for a set of test cases. In BRIDGE, information is provided for validation: performance specifications are formulated, a partial domain model is provided as a context model, and indisputable knowledge is identified in several stages of the validation process.

Database integrity rules also apply for the data and relations in the context model and case-bases. The structural and functional views also apply to case-based diagnosis systems:

- The structural view checks for internal errors, such as redundancy, inconsistency and incompleteness of the context model and cases. Internal errors provide information feedback for updating individual fault-cases.

- The functional view checks for external errors in the reachability and coverage of diagnoses and actions. External errors provide feedback for updating the similarity measures, context model and fault-base.

Several modifications are required to the validation process of rule-based systems. For case-based diagnosis systems the performance, rather than quality, should be verified, while similarity measures introduce a form of approximate reasoning. Accuracy, coverage, and reachability may be interpreted as "adequacy criteria" for verifying "redundancy", "consistency" and "completeness" of the context model and case-bases. This will be discussed below.

The validation approach aims to maximize automatic verification and minimize evaluation. Validation process consists of 5 consecutive processes for the verification of integrity, coverage, reachability, and response time, and for the evaluation of test cases. Verification of response time is further discussed in [12].

### 4.1 Integrity Verification

All following validation steps assume the integrity of the knowledge in the context model and case-bases with the object model from Fig. 1. The object model is formulated in a set of attribute, entity, and referential integrity rules. Following internal errors can be detected:

- Redundant definitions of entities and relations.
- Invalid attribute values and references to non-existing entities (inconsistencies).
- Undefined entities in the context and case features (incompleteness).

Integrity is verified in 3 steps.

1. The context model is verified independently of the case-bases. During this step the context knowledge is revisable. Any conflict can directly be restored.
2. The integrity of the features in the case-base and their relations to the context is verified. The context knowledge is assumed indisputable while the features are verified or modified. A conflict in the feature definition is restored directly. A conflict in the context model requires an iteration back to step 1.
3. The integrity of case-definitions, case-feature or case-context relations is verified. Both the context model and the features are assumed indisputable while cases and case-relations are verified. A conflict is either restored directly in the case or incrementally via steps 1 or 2.

### 4.2 Verification of Coverage

The coverage of the case-based diagnosis system is the set of problems it can solve. Verification of the coverage of the fault-base requires additional domain knowledge. The context model and failure-base represent a part of the domain knowledge and specify a minimum requirement for the coverage of the fault-base.

The entities in the context model are associated to normal and abnormal behavior of the application. For example:

- The configuration tree is expanded for components associated with observables, faults and actions. These leaf components should thus be covered by fault-cases.
- The observables and actions should also be covered by fault-cases.
- Health state levels indicating abnormal behavior should be covered by fault-cases.

Coverage of these entities can be verified and additional knowledge can be obtained from the failure-base or the domain expert.

All known real-world failure cases should be covered in the fault-base and context model; i.e. at least one fault-case is defined with the same fault, actions, and observables, and the operating conditions should match the relations in the context model. Following errors can be detected:

- When a failure case cannot be categorized as an auxiliary or support case of any of the fault-cases, then the fault-base is incomplete.
- When the operating conditions for entities in the failure-case do not match those in the fault-base, then the context or feature model is incomplete.

### 4.3 Verification of Reachability and Accuracy

The functional requirements for the reachability and accuracy of diagnoses and actions can be formulated as a set of structural requirements for the reachability and accuracy of fault-cases. In BRIDGE, requirements are specified for critical operating conditions and for the general situation in which all inputs can be observed.

The observable set of inputs can be too restrictive to accurately diagnose a problem. This can be verified from the intersections and differences in the sets of inputs for cases and their similarity values. If a fault-case cannot be identified with sufficient accuracy, then it is either redundant or incomplete. Categorization of a case, i.e. as auxiliary, spanning or support cases, is decisive for updating of the fault-base. These internal errors are directly related to external errors in the retrieval and reuse of diagnoses and actions. Functional verification of the reachability and accuracy of diagnoses and actions only becomes necessary when internal errors are detected.

For critical operating conditions, requirements are specified in general terms from the context model. The requirements should be satisfied by cases from a particular health state or part of the configuration. The observable input is restricted to a branch in the configuration, by operational mode, operator level, or available resources.

A global requirement is that every fault-case is reachable with sufficient accuracy in any situation; i.e. when all inputs are available. The global requirement is less stringent and should only be verified for cases not verified for any critical operating condition.

### 4.4 Evaluation of test cases

Operational experience will reveal hidden problems and faults. The observable inputs are insufficient to accurately diagnose these cases. Upon detection of such an error a dilemma is posed. Updating or addition of a fault-case will increase the coverage but also violate the accuracy requirements. The diagnosis system cannot be validated for these problems. Fault-cases for hidden problems cannot be verified automatically and should be evaluated separately.

# 5 Integrity Verification in BRIDGE

The integrity verification process is implemented in BRIDGE and operational, while the following validation steps are still under development. Integrity rules are formulated with a name (identifier), constraint relation, and a violation response.

Application independent rules for entity and attribute integrity are straightforward; i.e. presence of entities and attribute values, uniqueness of entity identifiers, data types and value domains. Upon violation, the user is prompted to either accept a default value or define a valid value.

Application dependent rules verify the referential integrity for the associations in the object model from Fig. 1, such as the hierarchical structures in the context model. Local integrity rules verify binary relations between pairs of entities:

- Generalization and instantiation of components, their configuration, and aggregation, and the inheritance and specialization of features and cases.
- Ordering of health states by their severity on functionality and safety.
- Generalization of operational modes.
- Organization of operator levels and teams, and their deployment by skills (systems in the configuration) and operational modes.
- Priority ordering of resources, and organization of tools in toolboxes.
- Location of features and cases in the configuration.
- Impact of cases on functionality and safety (health states).

Global integrity rules verify paths of binary relations through the fault-base and context:

- Loop detection in the hierarchical structures.
- Observability of features; i.e. deployment of resources by operators, skill and deployment of operators during operational modes.
- Intra- and inter-system causal relations; i.e. relative location of features from cases and vice versa.

The location of causal relations poses a significant problem for integrity verification. Failures are recorded with causes and effects in particular instances of components, which are absolute locations. Generalization of failures into fault-cases requires three transformations of entity locations:

- Causes and effects are typical for types of components and are generalized as features of component types or sub-trees in the configuration.
- A fault may cause a particular effect in more than one component. The fault-case defines the relative location of an effect, which is a path through the configuration relative to the fault's own location.
- For diagnosis of effects, the relative locations in the fault-cases should be transformed into locations of cases relative to effects.

Relative causal-effect and effect-cause locations are many-to-many relations in the configuration of which the integrity is verified by generalizing and expanding paths in the configuration.

Integrity verification is statically, during indexing of the fault-base. The verification procedure follows the three-step approach and the directionality of the object associations. Upon violation, the inconsistency is presented to the user with a default modification or a dialog box to browse the already verified and indisputable knowledge.

Experience with the two industrial applications in the BRIDGE project is that, despite substantial development efforts from domain experts during 1 year, many integrity conflicts are still detected in the knowledge base. It also provides valuable and adequate information to refine and extend the knowledge base. Integrity verification is therefore considered essential in the development of case-based diagnosis systems.

## References


1. Smyth B., Cunningham P., The Utility Problem Analysed. In: Smith, I., Faltings, B. (eds.): Advances in Case-Based Reasoning, Lecture Notes in Artificial Intelligence, Vol. 1168, Springer (1996) 392-399.
2. 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) 85-100.
3. Bergmann R, Breen S., Fayol E., Göker M., Manago M., Schmitt S., Schumacher J., Stahl A., Wess S., Wilke W., Collecting Experience on the Systematic Development of CBR Applications Using the INRECA Methodology. In [17], 460-470.
4. Heister F., Wilke W., An Architecture for Maintaining Case-Based Reasoning Systems. In [17], 221-232.
5. Lieber J., A Criterion of Comparison between Two Case Bases. In: EWCBR-94, (Eds.) Keane M., Haton J-P., Manago M., (1994) 205-214.
6. Portinale L., Torasso P., Performance Issues in ADAPtER, a Combined CBR-MBR Diagnostic Architecture. In: Proc. AAAI Spring Symposium on Multimodal Reasoning, Stanford, AAAI Press, (1998)
7. Smyth B., Keane M.: Remembering To Forget. In: 14th Int. Joint Conf. on AI, Montreal, Canada, (1996)
8. 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
9. 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
10. Sleeman D., Craw S.M., Ellery I.G., Sharma S., Edwards P., Machine Learning, Knowledge Base Refinement and Knowledge Acquisition: Systems which Cope with Incomplete & Inconsistent Knowledge Bases. In: EKAW91, (Eds.) M. Lister, B.R. Gaines, (1992) 332-342.
11. Netten B.D.: Representation of Failure Context for Diagnosis of Technical Applications. In [17], 239-250
12. Motus L., Naks T.: Handling Timing in a Time-critical Reasoning System – a Case Study. Submitted to: IFAC Symp. On AI in Real Time Control. October 5 – 8, Grand Canyon Arizona (1998)
13. Date C.J., An Introduction to Database Systems, 6th edition, Addison-Wesley Publ. (1995)
14. Laurent J.P.: Proposal for a valid terminology in KBS Validation. In: 10th European Conference on Artificial Intelligence, (Ed.) B. Neumann (1992) 829-834
15. Tepandi J.: Comparison of Expert System Verification Criteria: Redundancy. Chapter 4 in: Validation, Verification and Test of Knowledge-Based Systems. (Eds.) M. Ayel, J.-P. Laurent, John Wiley & Sons (1991)

16. Loiseau S.: Checking and restoring the consistency of Knowledge Databases. In: Encyclopedia of Computer Science and Technology, Vol 36. (Eds.) Kent A., Williams J.G (1996) 15-34
17. Smyth B., Cunningham P., (Eds.), Advances in Case-Based Reasoning, Springer, (1998)

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