This article has been accepted for publication in a future issue of a journal/proceedings, but has not been fully edited. Content may change prior to final publication. Citation information: Belli, F., Bonin, H.E.G., Jedrzejowicz, P., Development and Validation of FIREX - A Knowledge-Based System for Tranport of Dangerous Goods and Fire Department Consulting, Annual Review in Automatic Programming, Vol. 14, Part I, Pergamon Press, Oxford etc., pp. 69-74

DEVELOPMENT AND VALIDATION OF FIREX—A KNOWLEDGE-BASED SYSTEM FOR THE TRANSPORT OF DANGEROUS GOODS AND FIRE DEPARTMENT CONSULTING

F. Belli*, H. Bonin*, H. Gerdes*, W. Filipowicz and P. Jedrzejowicz****

**Institute for Systems Analysis and Informatics, Hochschule Bremerhaven, FRG*
***Computer Science & Systems Analysis Department, Merchant Marine Academy, Gdynia, Poland*

Abstract. Formal aspects of expert systems have been studied thoroughly both in Artificial Intelligence and Computer Science. The more the expert systems become mature for a wide application, the more attention will be paid to a transparent development of these systems. The objective is to gain more confidence in the system, especially when used in safety-critical areas.

This paper summarizes the preliminary results of our approach to the development and validation of the system FIREX ("FIRe avoidance and combat EXpertise") which will be materialized in PROLOG and LISP in cooperation with the fire-brigade department of the city of Bremerhaven who is also the end user of the system.

Keywords. Software Development and Validation, Reliability, Knowledge Based Systems, Dangerous Goods, Operations Research, PROLOG, LISP, Expert System Shell S.1.

INTRODUCTION

Nowadays, there exist a great number of computer supported data bank systems to handle the problems which are due to appear when dangerous goods are concerned, e.g. transport and stowage of such goods, or defending of human health and environment when they catch fire. Examples for german and international data bank systems are INFUCHS /1/, CHEMDATA /2/, CD-ROM for Dangerous Substances /3/, BAM-Dangerous Goods /4/, EXIS /5/, IgL-Catalogue /6/, FEUDAL /7/ or Danger-Data /8/. These Systems mainly give information about critical features of dangerous substances and explain how they should be treated. Well structured, international systems identify and classify dangerous substances uniquely, e.g. IMDG-Code and UN numbers /9, 10/.

What has motivated us to develop an expert system to handle dangerous goods - although we have been informed on the existing data bank systems with excellent services? The above mentioned data bank systems give information usually on one single substance. An expert system is supposed to achieve specific information on several substances; especially when one has to make an optimal decision concerning interaction of such substances. The objective is the reduction of the risk when dangerous goods are to be handled. As an example, we consider the question: How should they be stowed onto or into a ship, i.e. in which sequence and in which protective environment? Apart from minimizing the risk of damage of the shipment as well as the environment, also the costs of the transport, stowage, and storage are to be minimized. Moreover, one must take the "worst case" into account, even if everything has been accomplished to exclude this case: What should be done when "something goes wrong", e.g. leakage, fire on board etc /11/? What is the best method to minimize the damage on the shipment, and especially on human health and environment, during the combat against the undesired events of the worst case? Roughly spoken, we have been trying to simulate the activities of an human expert in this field.

For the construction of an appropriate system to be consulted about the transport of dangerous goods and fire combat, are knowledge and methods necessary from following related areas:

- Environmental protection. Dangerous substances and their physical and chemical reactions, fire defense and combat.
- Methods of Computer Science, especially data and knowledge based systems.
- Operations research to determine the optimal decision.

The conception of the knowledge based system FIREX ("FIRe avoidance and combat EXpertise") considers following potential applications:

- Decision support. Carrying and stowage of dangerous goods must be well planned and carried out. We mentioned above some of the interesting optimization problems. They concern minimizing the risk subject to the costs.
- Training and tutoring. For the fire brigade, a difficult problem during the education of the ongoing staff is their training. On one hand, the potential team leaders must be trained thoroughly, best "on the job". On the other hand, the realistic simulation of fire combatting theatres is expensive and risky. FIREX is supposed to be employed here, mostly to evaluate the trainee's suggestions to plan and control the fire combat actions.
- Fire combat. After a sufficient successful duration of the deployment in the training, the FIREX is supposed to be deployed also in the "reality", depending on the experiences which will then have been made in the education.

FIREX will be developed by the Institute for Systems Analysis and Informatics ("INSI") at the Hochschule Bremerhaven, Computer Science & Systems Analysis Department of the Merchant Marine Academy Gdynia (Poland), and Fire Brigade Department of the city of Bremerhaven.

PROBLEMS WITH DANGEROUS GOODS

Dangerous goods (we will synonymously use "dangerous substances") are those which are inherently dangerous as well as those which are liable to chemical reaction with other ones. The latent hazard entails the need for segregation between

different substances. Fire may spread more rapidly if inflammable substances are not effectively separated from ignition sources. Substances which are inflammable should be segregated from those which may explode when involved in fire.

Transport of Dangerous Goods

General remarks. In shipping, four segregation distances have been introduced taking into account potential hazard as well as the construction of the store. These distances are abbreviated and referred to as following:

1: away from,
2: separated from,
3: separated by complete compartment from,
4: separated longitudinally by an intervening complete compartment from.

(Compartment means enclosed space of which boundaries should be resistent to fire and water.)

In addition to these distances, we introduce several terms which will be used in the optimization procedure as will be stated later:

"#": not known,
"0": no segregation recommended,
"5": prohibited (substances involved can not be shipped together).

For the shipment of goods, nine classes have been introduced as a guideline for the first and rough assessment of properties of any dangerous substance. The classes were further divided into subclasses. The table containing segregation distances between subclasses (classes) is published in every guide to storage and stowage of dangerous goods /10,12/.

In addition to a segregation which may be general as between whole classes of substances there may be a need to segregate a particular substance from materials which could contribute to its hazard. It refers to the whole class or sometimes to the particular type of substances. Properties of some substances may be considered different with respect to their flashpoint.

For the sake of fulfilling any requirement each dangerous substance is described on an individual sheet in IMDG ("International Maritime Dangerous Goods") Code book /9,12/.

To maintain proper distance one should select the worst case from the segregation distances with respect to:

- classes segregation table,
- special segregation with the selected classes,
- special segregation with the selected types,
- segregation requirement depending on flashpoint limits.

Classes and limits of flashpoint are well defined for every substance. Proper assignment of the substance to any type appears to be more complicated and sometimes expert knowledge is to be consulted.

Precising the Problem Status. Transportation and stowage of dangerous goods requires an expert knowledge on their properties and characteristics as well as ability to make the decisions assuring safe and efficient cargo handling. Important part of the required knowledge on dangerous goods has already been assembled in /9,12/. This book does solve, however, neither classification nor segregation problems. To decide on how the different substances should be segregated or what is the minimal compartments requirement to store or to carry dangerous goods consignment (called a load) one has to apply segregation rules, i.e. an expert knowledge, and solve an optimization problem. To support both a small decision support component, called DANLOAD, has been developed. This component supports the user by providing information and/or evaluation with respect to the following:

- Dangerous goods characteristics.
- Segregation between different substances.
- Minimal compartments required to store or to carry dangerous goods consignment.

This subsection briefly reviews the optimization procedure.

Optimization Procedure. Minimal number of separated compartments required to carry the dangerous goods (consignment) as well as allocation of substances are generated. The problem at hand is equivalent to the following one:

Given: Graph $G = (V,E)$ and integer k (number of compartments) with $k \leq |V|$, where V is a set of nodes (substances) with the cardinality $|V|$ and E is a set of edges (i.e. for every substance there is an edge if the segregation distance is 2,3,4 or 5.)

Question: Is there such assignment (stowage plan)
$f: V \rightarrow \{1,2,\ldots,k\}$
that for each $u,v \in V$ implies
$f(u) \neq f(v)$ if $(u,v) \in E$.

The problem may be defined less formally:

Given: a set of substances, and segregation distances for each pair of substances.

Goal: Assign substances to the separated compartments in such a way that inside each there are substances with segregation distances less or equal 1 (i.e. allowed distances are #, 0 and 1).

It was proved that such a problem is NP-complete and hence unlikely to be solved by efficient (i.e. polynomial time) optimization algorithms /13/.

As the number of dangerous goods carried in one load is not likely to exceed an amount 15 to 20, the following exact algorithm has been applied:

1. For given graph $G = (V,E)$ find all minimal bases (i.e. subgraphs $G1 = (V1,E1)$ where $V1 \subseteq V$ and $E1 \subseteq E$ for which every $e \in E$ is incident with at least one $v \in V1$.)

2. Find all minimal covers of the set V with the subsets $V1$ (i.e. $\bigcup_k V_{1k} = V$). The smallest k is the number of necessary compartments.

3. Generate all possible assignments of substances to compartments.

To avoid uncontrolled time and core consumption the solution space was limited both in terms of the number of solutions and the problem size. For larger cases suboptimal algorithms are used /14/.

Fire Combat and Extinguishing

Water will be used in most cases to extinguish fire, but it can also have catastrophic consequences if applied wrongly, e.g. when carbide is concerned. The second component of the FIREX takes methods and means on fire defense and fire combat into account /11/. In the present stage, the

knowledge on extinguishment methods and tools will be implemented. The interactions of several dangerous goods will be considered. If no common, compatible extinguishment means are available then the transport and stowage of such dangerous goods must not be carried out, even if it is not prohibited by law. In such cases, the system will produce a warning.

The knowledge basis is structured according to the representation scheme "object-attribute-value". The objects are real or mental items of the application environment. Objects have features, called attributes, with which they will be associated. Attributes can have different values. We differentiate between the objects of static knowledge, inference knowledge, and strategic knowledge.

"Classes" and "Types of classes" are elements of the static knowledge. Any problem environment of the application/user defines a class. During the consultation, FIREX creates instances of a class, e.g. of the class "goods". Each instance defines then the domain of the substances. The goal of an instance is to assemble data on the instance. An attribute represents an information on an instance, e.g. appropriate extinguishing means as water, or method as jetting water.

The values of the attributes can be determined during the dialogue with the user or through the inference mechanism of the system. If such an assignment can not be performed, they retain their initial value "unknown". FIREX uses attributes for the properties of the substances, as to state (solid, liquid, gas), color, odour etc. Other attributes contain decision moments or danger classes.

The inference knowledge will be represented by means of rules of precisely specified decision tables. A rule checks the values of the attributes and invokes an appropriate action when the conditions hold. Rules will be implemented for various purposes, as to legal regulations, determination of the danger class etc.

The strategic knowledge controls the inference process by means of "Control Blocks" and "Functions". A control block determines when an instance is to be created, or an attribute is to be assigned to a value, or an output is to be produced. Functions constitute sets out of attribute values, e.g. intersection of extinguishment means of substances.

An explanation component can be materialized by means of the constructs "WHAT, HOW, WHY" of the expert system shell S.1 (Framentec SA). This explanation component can elucidate the results inferenced by the system, e.g. why a shipment must not be loaded. As the entire consultation will be protocolled, the user can work with event numbers (identification of a question). He/She can also work with a combination of classes, instances, and attributes.

Structure and Development of the "FIREX" System

Traditional methods of the requirements engineering, e.g. SA ("structured analysis"), could not have been applied to the system development in our logic and object oriented approach. The main reason is the tedious refinement of the graphical SA model of the FIREX. This model has to be combined with the partially verbal entries of the data dictionary in order to cope with the complexity which is necessary to detect inconsistencies and incompletenesses of the requirements definition. Some assistance was given by decision table systems /15,16/. We must admit, however, that we could not accomplish completely any formal specification of the system requirements which would have been necessary for a formal verification of the system. Our semi-formal specification can be used to semi-formally validate the system as will be described in the next section.

For the implementation of the final, operational system we chose PROLOG for a logic programming paradigm. Once the functional requirements had been specified, PROLOG proved very efficient for constructing prototypes rapidly.

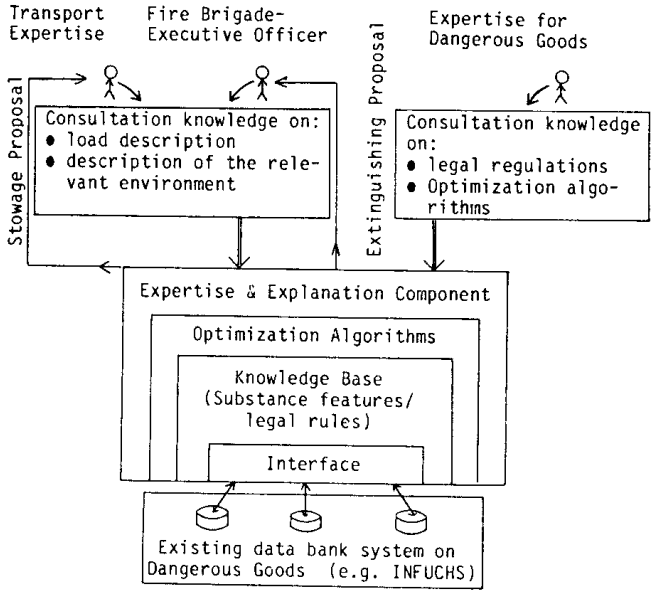

Fig. 1. The Structure of the FIREX System.

PROLOG Implementation in Examples. The segregation table can be easily mapped into PROLOG facts as the following examples depicts. The rule

```
segregation_rule(['5.1','0'],['2.2','5.1']).
```

means that the substances which belong to the class 5.1 should be segregated from the substances of the classes 2.2 and 5.1 according to the rule 0. The other entries of the matrix will be implemented similarly.

Also the substance lists can be implemented easily, e.g. the PROLOG fact

```
substance_list(['5.1','Calciumpermanganat',
                    'Ammoniumpersulfat',..]).
```

represents that substances Calciumpermanganat and Ammoniumpersulfat belong to the class 5.1.

Exceptions of the rules can also easily be mapped:

```
exception(['2','Calciumpermanganat'],
              'Ammoniumpersulfat',
              'Ammonium..',..]).
```

The above fact determines that Calciumpermanganat should be segregated from Ammoniumpersulfat etc. according to the rule 2 although the segregation table determines the rule 0.

Once the facts have been specified, the transportation expertise can be implemented in a straight forward manner. We will not demonstrate this here as it has been published elsewhere /17,18/ in most detail.

VALIDATION OF FIREX

"Testing in the Large" of FIREX

Testing of complex information systems and knowledge based systems is a high-demanding activity. Formal methods are precise but shown to be hardly feasible and error-prone (see e.g. /19, 20/).

Due to its safety relevance, the system FIREX must have a high degree of reliability and confidence. Guided by the results of our previous and to date work /20, 21/, we have been studying following problems:

- "Test Problem". Can the theoretical sound test methods of the traditional software engineering (e.g. "Test Theory" of Goodenough/Gerhart, or "Mutation Theory" of DeMillo/Lipton, or "Domain Theory" of Howden /22/) be deployed to validate expert systems? If not, can they be extended or modified to accomplish this purpose?

- "Reliability Problem". One major problem during the validation of a system is the determination of test termination, i.e. releasing the system under test. Can the existing software reliability models and methods /19,21,22/ be applied to determine and to assure the reliability of expert systems? Moreover, can the existing methods be used to optimize the structure of such systems /23/)?

We will here exclude the second problem as it is a research direction of its own. Further, we will exclude the formal methods to verify knowledge based systems as this is the objective of ambitious theoretical projects and is beyond the scope of this paper.

Testing, Test Cases, and Test Environment for PROLOG Programs

A test can be carried out in order to detect errors in a system and, if any, to localize them. The objective is thus the diagnosis and correction of errors to exclude later system malfunctions and/or failures.

A test can also be carried out to check the degree of the fulfillment of the relevant properties of the system. This will also be called "measuring the quality metrics", e.g. the efficiency concerning the execution time of a program.

Tests are usually carried out on a running system, i.e. dynamically through the execution of the software on hardware. A test can also be performed analytically, without running the test object. The latter activity will be called static testing.

A Test Case consists of test input data and expected output of the system which must be determinable and reconstructable. Test cases must be prepared in advance, i.e. before tests start.

Test cases for PROLOG programs consist of falsified predicates of the clauses and arguments of the predicates. Such clauses are known as "queries" or "procedure calls" which then invoke a PROLOG program. Additionally, the system reaction (i.e. answers) must be determined in advance. The arguments must be chosen partly instantiated, partly variable to verify the system behaviour in any kind of inference.

Example: Provided that the knowledge base rudimentarily stated in the last section has been extended to a PROLOG procedure, PROLOG can generate answers to questions, e.g.

```
?- stowage('Calciumpermanganat',
           'Ammoniumpersulfat', RULE).
```

which materializes the question how both substances should be stowed. The system has to answer:

```
RULE='2'
```

as an exception rule exists here.

However, PROLOG can also check the correctness of a rule:

```
?- stowage('Calciumpermanganat',
           'Ammoniumpersulfat', '0').
```

which materializes the question whether both substances should be stowed according to the rule "0". The system has to answer:

```
No
```

as the exception overrides the regular rule.

A Test Environment consists of tools to generate test cases, to drive (i.e. execute) the test, and to report the results of the test. The test environment should also include tools to carry out static testing.

The Test Environment for FIREX

Once we have decided the logic and object oriented paradigm to be the programming style and PROLOG to be the major implementation language of FIREX, the necessity raised to develop a test environment for PROLOG programs. On this objective a subproject called PRO-Test (PROLOG-Test-Environment) has been working.

The rationale of PRO-Test is given in Fig. 2. In the following, we will briefly elucidate the structure of this test environment.

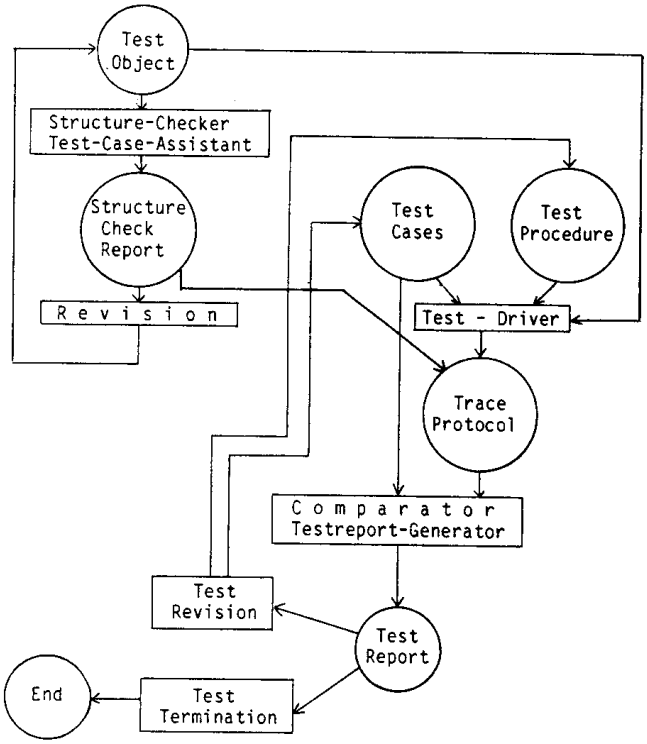

Fig. 2. The structure of PRO-Test

Static Testing

Type checking is one of the main instruments of static testing /24, 25, 26/. PRO-Test needs type specification as input and assists the user to develop test cases.

The Structure Checker. PROLOG was originally designed to have the only construct "Clause", consisting of the only structure "Term". The programmer can create his/her objects without any restrictions as to boundaries of data types, number of arguments of predicates of clauses etc. This "liberty without a restriction" has been proved, however, error prone in practice as the involved translator has no chance to carry out the usual static, i.e. lexical and syntactic, checks. To have these benefits, we introduce following declarations required in each PROLOG program:

- Types and structure of the functors
- Types and structure of the predicates

Moreover, the user has to attach examples of minimal number for each defined functor and predicate. The system uses these examples as references and as a basis when test cases will be produced.

A functor is primitive when the structure it defines is self-contained. A compound functor involves another functor with its belonging structure. To process declarations on compound structures, the system takes into account the primitives which have already been declared and are contained in the compound structure, i.e. the possible inconsistencies will be uncovered or incomplete parts will be "unified" where it is possible. Repetitive declarations will be uncovered and thus excluded. For primitives the user selects the reference instances which are likely to reveal errors.

Based on the declaration, the Structure Checker can reveal type conflicts and incompletenesses. Moreover, the introduced constraints cause a better understanding of programs.

Test Case Assistant. The declaration of predicates has the structure of the clauses and queries. Through "qualification" of the arguments, i.e. setting them to variables or to constants, test cases can be produced. The qualification will be performed semi-automatically: The system produces all combinations from which the user can make his/her selection for the ongoing test. In a dialogue with the user the system combinatorically produces a set of test cases. In this dialogue the user has the following options:

- Enter a special value.
- Enter a subset of the reference instances.
- Enter a number which indicates the cardinality of the randomly generated subset of the reference instances

Belonging to each test case, also the expected result as system output has to be determined.

To gain meaningful test cases, the system interacts with the user to qualify the clauses which have been declared and referenced by examples.

Dynamic Testing

As depicted in Fig. 2, the Test Driver executes the program, i.e. the test object, according to the specified test procedure applying to prepared test cases. The Comparator sets the expected results (as contained in the test cases) against the results produced by the execution. The inconsistencies as well as the test coverage will be documented by the Test Reporter. As an example, a report on a "passed" test may have following appearance:

"All test cases successfully executed,
10 clauses out of totally 100 have been activated."

Example for a "failed" test:

"Following test cases failed:
Test Case 7: Expected "Yes" Real Result: "No"
Test Case 9: Expected "3" Real Result: "3,4"
78 clauses out of totally 100 have been activated."

SYSTEM STATUS TO DATE AND CONCLUSION

Programming Environment

The transport component of the FIREX has been implemented in PROLOG. We used several PROLOG systems to materialize prototypes in order to gain experience with these systems and the problem. The optimization nucleus has been implemented in C and then linked with the main system.

The FIREX component which materializes the fire combat consulting has been implemented both in the expert system shell S.1 and TLC-LISP (IBM-AT; Apollo DN 3000).

We deliberately chose the above listed "conglomerate" of languages, tools, and systems, partly as we enjoyed to experiment with the subject and the technical environment. For some parts, however, we were forced to leave the logic and object oriented world in order to implement classical OR-algorithms, as in DANLOAD, where C is used in the imperative way.

Preliminary Experiences and Theses

Up to date, we have been exercising on the prototypes of the both components of FIREX. Our first experiences can be summarized as follows:

- There is a wide gap between well-known "toys" or experimental knowledge-based systems which have been published excessively and the needs of the "real" users of "real" expert systems. Without an access to the existing data bank systems and without appropriate implementation of intrinsic OR algorithms, the utilization of an expert system as a decision support tool is considerably confined.

- The traditional life cycle model to develop software systems ("waterfall model") is not adequate to implement knowledge-based systems. The "rapid prototyping" model could be applied, i.e. the requirements should be validated by means of prototypes of the expert system to be developed before their design and implementation start. During the implementation one should utilize the rationale of the "programming in large".

- The special tools which are available at the moment to develop expert systems, e.g. expert system shells as to S.1, are not mature/efficient to be applied in reality and safety relevant environment of the FIREX as it includes classical data/information processing components besides modern knowledge processing.

  We must credit, however, such systems for their comfort to materialize the user interface in a quasi-natural language.

- Traditional test methods to verify software can

hardly be applied for knowledge-based systems. Further work to precise the terms used is necessary, e.g. to determine the term "module", "error" etc.

- The existing models and metrics, e.g. complexity, fault-tolerance, to define and quantify the reliability of software systems must be heavily modified to be applied in expert systems environment.

- The expertise or knowledge to be processed is a highly perishable item. Therefore appropriate tools are necessary to manage the configuration and to control the versions of the knowledge base.

One of the common problems of developing complex systems with sophisticated services to the user is the materialization of the requirement "ease of use and learn". FIREX has to fulfill high expectations of the user. To reach these objectives, the end user, namely the trainer and the trainees and authoritative experts of the fire brigade, participate on the development of the FIREX.

REFERENCES

/1/ "INFUCHS", Informationssystem für Umweltchemikalien, Chemieanlagen und Störfälle, Umweltbundesamt Berlin, Gruppe Z2, Information und Dokumentation, UMPLIS, Bismarckplatz 1, D-1000 Berlin 33.

/2/ "CHEMDATA", National Chemical Emergency Centre, B7.22 Harwell Laboratory, United Kingdom Atomic Energy Authority, Oxfordshire OX 11 ORA.

/3/ "Gefahrgut-CD-ROM, (Hommel: Handbuch der gefährlichen Güter; CHEMDATA (vgl. 6); Einsatzakten für Chemieereignisse des Schweizerischen Feuerwehrverbandes), Springer-Verlag GmbH & Co.KG, Postfach 105280, D-6900 Heidelberg 1.

/4/ "BAM-Gefahrgut/Tankfahrzeuge & -container", Datenbank Gefahrgut der Bundesanstalt für Materialforschung und -Prüfung (BAM), Fachgruppe 1.5 (Gefahrgutumschließungen aus Metallen), Unter den Eichen 87, D-1000 Berlin 45.

/5/ "EXIS", Chemical and Hazardous Materials Information and Transport Regulations Database", Exis Limited, 38 Tavistock Street, London WC2E 7PB.

/6/ "IgL-Datenbank", Informationen gefährliche Ladung", Hapag-Lloyd AG, 1430 Ladungsservice, Postfach 102626, D-2000 Hamburg 1.

/7/ "FEUDAL-System", Feuerwehr-Daten-Leit-System", Keudel av-Technik GmbH, Am Stocker 2, D-6331 Waldsolms (incl. Schulungsprogramme wie "Feuerwehr im Einsatz - Du hast das Kommando", vgl. 6800, Ausgabe 1/ Januar 1987, S.15-17).

/8/ "Danger-Data", Gefahrgut-Computer SHARP CE-150", Albert Ziegler GmbH & Co. KG, Postfach 1680, D-7928 Giengen/Brenz.

/9/ "IMDG-Code", International Maritime Dangerous Goods Code, London 1983 (International Maritime Organization); United Nations, Secretariat Committee of Experts on the Transport of Dangerous Goods, Definitions and Criteria for Classes.

/10/"Bundesgesetzblatt Teil I Nr. 30 vom 4. Juli 1986", Anlageband (ca. 3000 Seiten); Quester, H. (Hrsg.), Gefahrgut Vstr, Anlageband 2, VerkehrsVerlag Düsseldorf.

/11/"FwDV, Feuerwehr-Dienstvorschriften", z.B. "FwDV 12/1", Einsatzleitung - Führungssystem, Deutscher Gemeindeverlag, 1982.

/12/"Stowage & Segregation Guide to IMDG-Code", K.O. Storck Verlag, Hamburg.

/13/Garey, M.R., D.S. Johnson, "Computers and Interactability", W.H. Freeman and Co., 1979.

/14/Korzan, B., "Elements of Graph Theory and Networks", WNT, Warszawa, 1978.

/15/Bonin, H., "Die Planung komplexer Vorhaben der Verwaltungsautomation", R.v.Decker & C.F.Müller, Heidelberg, 1988.

/16/Bonin, H., "Software-Konstruktion mit LISP", Arbeitsberichte des Fachbereichs 2, Hochschule Bremerhaven, 1987/7 (Version 2.4).

/17/Belli, F., "Einführung in die logische Programmierung mit PROLOG", Bibliographisches Insitut - Wissenschaftsverlag, Mannheim etc., B.I.-Hochschultaschenbuch, Band 630, zweite erweiterte Auflage, 1988.

/18/Belli, F., "PROLOG-Systeme in der Praxis - Überblick mit Anwendungsbeispielen in IF/-Prolog, MPROLOG, Turbo-PROLOG und anderen PROLOG-Systemen", Arbeitsberichte des Fachbereichs 2, Hochschule Bremerhaven, 1987/10 (Version 2), erscheint bei Bibliographischem Insitut - Wissenschaftsverlag, Mannheim etc., B.I.-Hochschultaschenbuch, Band 631, 1988.

/19/Belli, F., P. Jedrzejowicz, "Ein Ansatz zur Zuverlässigkeits-Optimierung fehlertoleranter Systeme", it - Informationstechnik (ehemals "Elektronische Rechenanlagen"), 2/1987, S. 61-68.

/20/Belli, F., M.A.J. Burford, "CADAS: A Tool for Designing Reliable Embedded Software and Supporting Testing 'in the large'", Tagungsband 2. GI/NTG/GMR-Fachtagung "Fehlertolerierende Rechensysteme", Informatik- Fachbericht Nr. 84, Springer-Verlag, Berlin, New York etc., 1984, pp. 101-113.

/21/Belli, F., P. Jedrzejowicz, "Software Reliability - Modelling and Optimization", to appear in "EIK - Elektronische Informationsverarbeitung und Kybernetik", (Journal of Information, Processing and Cybernetics), 1988/8.

/22/Belli,F., "Ein Bündel heuristischer Methoden zur Kostenoptimalen Bestimmung und Sicherung von Software-Zuverlässigkeit", Informatik-Fachbericht Nr.83,("Software-Fehlertoleranz und Zuverlässigkeit"), Springer-Verlag, Berlin, New York etc., 1984, pp.253-268.

/23/Belli, F., P. Jedrzejowicz, "Optimization of Reliability and Structure of Stoftware, The next Step after Reliability Modeling", Arbeitsberichte des FB2, HS Bremerhaven, 1988/1 (VO).

/24/Mycroft, A., A. O'Keefe, "A Polymorphic Type System for PROLOG", Artificial Intelligence 23, 1984, pp. 295-307.

/25/Milner, R., "A Theory of Type Polymorphism in Programming", J. Comput. System Sci. 17(3) 1987, pp. 348-375.

/26/Bäcker, A. et al., "Eine Übersicht über die Prolog-Programmierumgebung ProLab", Tagungsband des GI-Workshop 'Sprachspezifische Umgebungen', Technische Hochschule Darmstadt 1988, S. 141-155.

Acknowledgement

This work was supported in part by Deutscher Akademischer Austauschdienst (DAAD), Senator für Bildung, Kunst und Wissenschaft of the Hansestadt Bremen (Research Project "Reliable Systems"), and Deutsche Forschungsgemeinschaft (DFG).