Analytica Chimica Acta, 235 (1990) 27–40
Elsevier Science Publishers B V , Amsterdam – Printed in The Netherlands

Expert system for precision testing in validation of liquid chromatographic methods

J A VAN LEEUWEN *, L M C BUYDENS, B G.M. VANDEGINSTE [a] and G. KATEMAN
Department of Analytical Chemistry, Catholic University of Nijmegen, Toernooiveld, 6525 ED Nijmegen (The Netherlands)
M MULHOLLAND
Philips Scientific, Cambridge, CB1 2PX (Great Britain)

(Received 3rd July 1989)

ABSTRACT

After a liquid chromatographic method has been developed, it must be validated to establish its limitations in daily use Method validation is becoming increasingly important as stricter rules are applied by regulatory authorities. Precision testing is a vital step in this validation; both intralaboratory testing and interlaboratory testing are needed. In an intralaboratory test, repeatability and ruggedness tests are usually done Expert systems are available for both tests Here they are integrated to form an intralaboratory precision-testing expert system, special integration architecture is described Important features of the integrated system are a supervisor containing planning knowledge about the tests and a common data structure containing all the objects necessary for an expert system in this area

Many applications of artificial intelligence in analytical chemistry have been described recently [1–5]. Although the types of knowledge vary, most applications are found in expert systems for spectral interpretation (infrared, ultraviolet, mass or NMR) or elucidation of chemical structures [2–5]. The expert system described here was developed as part of a project (Esprit project 1570) that evaluates the use of expert systems in the development of liquid chromatographic (LC) methods [6]. In this project, method development is divided into four domains: first guess of conditions, selection of criteria for optimization [7], optimization of instrumental parameters and operating conditions [8] and validation of the method [9–11]. In this paper, the structure and implementation of the validation part is described.

When a full validation program is run, the five features of performance tested are the accuracy, precision, sensitivity, selectivity and limitations of the method. Because a full program for method validation involves testing in different laboratories, it would be of little use to try to tackle the entire validation in one expert system, as it would be very difficult to control expert systems located at different sites. The size of such a system would also become very large. Therefore the method validation is tackled in parts. The expert system described here is concentrated on precision tests that can be done in the laboratory where the LC method is developed. It is intended to give the analyst validating the method as much certainty as possible that this method will not fail in a collaborative interlaboratory test.

For most analysts in a routine laboratory, the performance of a precision test is not straightforward. Decisions must be made about which tests to apply, the extent of each test and the acceptability of the results. If a method fails the test, it is also necessary to specify the method again, with consideration of the problem that led to its failure.

[a] Present address Unilever Research, Vlaardingen, The Netherlands

Expert systems that advise on parts of the problem of precision testing are available [9,10]. For an analyst to use these systems most beneficially, the basis of the test procedures and their relations to each other must be known. An integrated system based on existing systems would eliminate this requirement.

An integrated system involves several modules of a heuristic as well as an algorithmic nature. Integration of modules with different problem-solving techniques, like calculations and production rules, requires a flexible architeture, in which the different modules are connected so that they can exchange as much information as possible. It must, for instance, be possible to have modules implemented in the usual techniques for expert systems as well as modules implemented in spreadsheet packages and normal programming languages like C. In principle, the architecture should not enforce too many constraints on the structure and implementation of the modules because this would damage their performance. Especially in this study, where new modules are integrated with existing expert systems, it is important not to change the existing systems too much and not to place too many restrictions on the new modules. A feature of the design for integration described here is the development of a kind of backbone for LC expert systems, a data structure that can serve as a basis for many different expert systems on validation of LC methods. A framework of concepts describing the basics of LC can be accessed by every module in the integrated system. The objects in the common data structure are represented formally on paper so that they can be implemented in any suitable technique for expert systems or in any programming language. By using the common data structure and the modules as building blocks, an expert system is built on intralaboratory precision testing. Because of its modular structure, the separate parts of the system can be activated independently. The system also contains planning knowledge on when to activate which module.

The repeatability test and the ruggedness test are the main parts of the integrated system. These tests were developed independently in trials involving several experts. The integrated system contains the integrated knowledge of these experts. This is a typical feature of so-called second-generation expert systems, which contain the knowledge of more than one expert, apply different inference techniques in different parts of the system, and can also select the next part of the system to be consulted. The so-called blackboard architecture is suitable for the implementation of such systems [12].

PRECISION TESTS

The purpose of a precision test is to establish the random deviation from the mean in a certain analysis. Precision testing normally consists of repeatability and (interlaboratory) reproducibility tests [13]. In a repeatability test on a method, the same sample is analyzed (usually 10 or 25 times) under the same conditions by the same analyst. Because the test site does not change, a repeatability test normally does not require much manpower and time. In contrast, reproducibility tests are relatively costly; identical samples are tested in different laboratories to examine the precision of the method under slightly changing conditions. To avoid excessive problems during a reproducibility test, a ruggedness test can be added to the precision test [14]. In a ruggedness test, the effects of changing parameters such as temperature, accuracy of the analyst, etc., are simulated, so that the likely performance of a method in a reproducibility study can be assessed. Possible problems can be identified and resolved before the actual reproducibility test. Especially in LC, a large number of factors may influence method performance. A ruggedness test on the right factors can greatly reduce costs during a reproducibility study and can be applied in the laboratory where the method was developed, like the repeatability test.

A repeatability test combined with a ruggedness test can be seen as an in-house precision study. This type of testing is advisable for every method that is destined for general use. However, ruggedness tests are rarely done; even repeatability testing is not yet part of many standard operating procedure, probably because of lack of experience in many routine laboratories.

Stand-alone expert systems

Previous work was aimed at the development of expert systems that could advise on parts of the problem of precision testing. For instance, expert systems on repeatability testing [9] and on choice of factors in ruggedness testing [10] have been developed. The latter system has been developed further to provide a complete system for ruggedness testing. Both systems are described briefly below.

The repeatability system. The repeatability system advises the user on the performance of repeatability tests on the injection and sample preparation of the LC procedure. The results of the repeated experiments are processed to see if the method is repeatable within the specified limits, i.e., the (relative) standard deviations are calculated and interpreted. If a problem with the method is indicated, the system diagnoses the problem and proposes remedies.

The repeatability system contains three modules, which cover the set-up of the test, interpretation of results and diagnosis cure. On the basis of a description of the method, the system selects a repeatability test for the injection and sample preparation procedures and provides a description of the experiments to be done. The spreadsheet structure of the second module allows input of the experimental results; the relative standard deviations are calculated and assessed for acceptability. If they are acceptable, consultation stops. If they are not, possible causes of the error are diagnosed and if the problem can be identified, the system advises on possible solutions, e.g., check for adequate degassing or for a loose grating in the detector. The first and third modules are implemented in a commercial expert-system shell; they contain mainly heuristic rules and frames to represent the objects in the system. The second module is implemented in a spreadsheet package because it is more algorithmic in nature and mostly concerns calculations.

The ruggedness system. The ruggedness system advises the user on a complete test, from selection of factors to interpretation of results. A ruggedness test consists of various experiments that simulate the changes to be expected when a method is transferred from one laboratory to another.

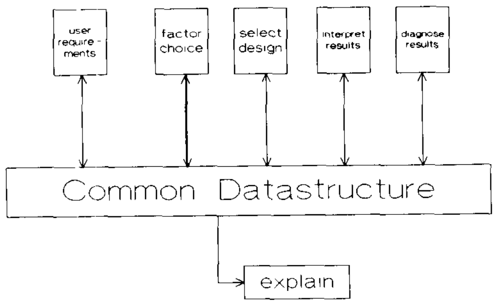

Fig 1. The ruggedness system

Because the number of possible influencing factors is large (about 40), the test must be efficient. If most of the interactions between factors can be neglected (which is normally a realistic assumption), the most efficient experimental design is a fractional factorial design. But even when factorial designs are used, the number of possible factors is too large, so that the important factors must be selected. The use and interpretation of experimental design require experience which is not commonly available in routine laboratories.

The ruggedness system is designed to eliminate the problems that prevent ruggedness testing from being part of a normal procedure for method validation. Various modules advise on the different steps in the ruggedness test (Fig. 1). First, the user enters a description of the LC method, which should include only the facts necessary for ruggedness testing. The system then advises on which factors to test; this is done on the basis of the input description and the requirements specified by the user (e.g., the expected usage of the method). If the user wants to change, add or delete any factors in the output advice, this is stored by the system and the user is warned that modifications have been made. When the set of factors selected by the system has been accepted by the user, the system selects an experimental design for the test; the number of experiments is usually 8–32. After the experiments have been done and the results have been put in, the interpretation module produces either suitability criteria or main effects. Main effects indicate that there are problems with the method; the system indicates if the problems are serious enough for rejection of the method. Normally, the user is only warned that certain factors should not vary too much or that certain parameters are not reliable.

As in the repeatability system, heuristic and algorithmic processes are used in the ruggedness system. The heuristic processes are implemented in an expert-system shell; the selection of factors and the selection of designs are rule-based. Interpretation of the experimental results is implemented in C language. The diagnostic module is implemented in an expert-system shell with frame-based reasoning.

INTEGRATION OF THE SYSTEMS

Integration of the repeatability and ruggedness systems involves the development of several new items and the adaptation of some parts of the separate systems. It is vital for all modules of the integrated system to use the same concepts as the basis for reasoning, so that flexible communication is possible between the various modules. Communication between modules in a system can often be done by simple transfer of files, but such communication would be insufficient here. For an integrated expert system, most of the facts produced by one module must be available to all other modules in the system. It is also important that the modules are not consulted in a standard sequence. If a file-transfer system were developed for this integration, all possible consultation sequences would have to be implemented, which would become unrealistically complex.

If all modules in the system must use the same concepts for reasoning, it is better to merge all existing concepts in one common data structure that forms the basis for all systems. In this study, the common data structure is built as a blackboard structure; all the modules of the integrated system can read information from and write information to this structure. The blackboard architecture was chosen because it allows integration of modules which use different inferencing or problem-solving techniques. This approach has some consequences for the user interface of the separate systems. If the user interfaces were not changed, the integrated system would sometimes ask for the same information twice, if the information were important for more than one module. When the information is available in the common data structure, a new module for method description must replace the separate modules of the stand-alone systems. In the architecture proposed here, this is the only essential adaptation of the existing systems.

The integrated system needs a supervisor to decide when to consult which module. The supervisor contains knowledge on each module, its input variables, its output variables and its status. On this basis, the supervisor decides which route should be taken to find an efficient precision test. With the supervisor, a level of meta knowledge (knowledge about the expertise in the system) is introduced into the system. Only the supervisor can trigger the modules; the modules cannot trigger themselves or each other. Because of the hierarchical control structure, it is relatively easy to implement additional levels of control, e.g., to integrate other tests like accuracy and sensitivity.

The method description module, the common data structure and the supervisor are the new items in the integrated system (Fig. 2). Adaptation of the existing systems to the common data structure and the method description module requires only minor changes. In future, new modules can easily be added if they use the concepts of the common data structure.

Method description

In this module, a full description of the LC method to be tested can be entered. The module contains knowledge on which LC methods can be assessed by the system. The method description module is normally the first to be consulted, thus the user is confronted with the limitations of the system at the start. If the system accepts the description of a method, the user can be confident that a valid consultation has been started and a valid conclusion will be reached. The only exception is when a very incomplete method description is provided. Normally, the system can work with an incomplete method description but if much information is missing, the system may not reach valid conclusions. The point at which the system loses its full validity is believed to be when 30% of the method description is missing.

The system contains knowledge about the normal concepts of an LC method. The user is not asked for any feature of the method that is not in line with previously given answers. For example, if a diode-array detector is involved, the user will be asked for wavelength, time constant and attenuation. When a refractive index (RI) detector is specified, the user will be asked the RI range and the temperature of the detector. If a column extraction is specified for sample preparation, the wash volume and the extraction volume will be needed. When a filtration is specified, the pore size of the filter will be required. However, the user can overrule the knowledge in the system by volunteering information. At any time during the method description, the user can enter information that he is not asked to enter. This is done by filling areas in the method description that are not filled by the system. Volunteering information is not always advisable. If the system does not need certain information it may have good reasons. Information volunteered at the wrong place may confuse the system and invalidate its conclusions.

Another feature of the method description module is the guidance given to the user to ensure that the description contains all the information relevant for the particular consultation. The system will not ask for more information than it needs. If, during a consultation, it becomes clear that only a repeatability test is necessary, the user will only have to specify a few parameters of his method, mainly related to the expected usage of

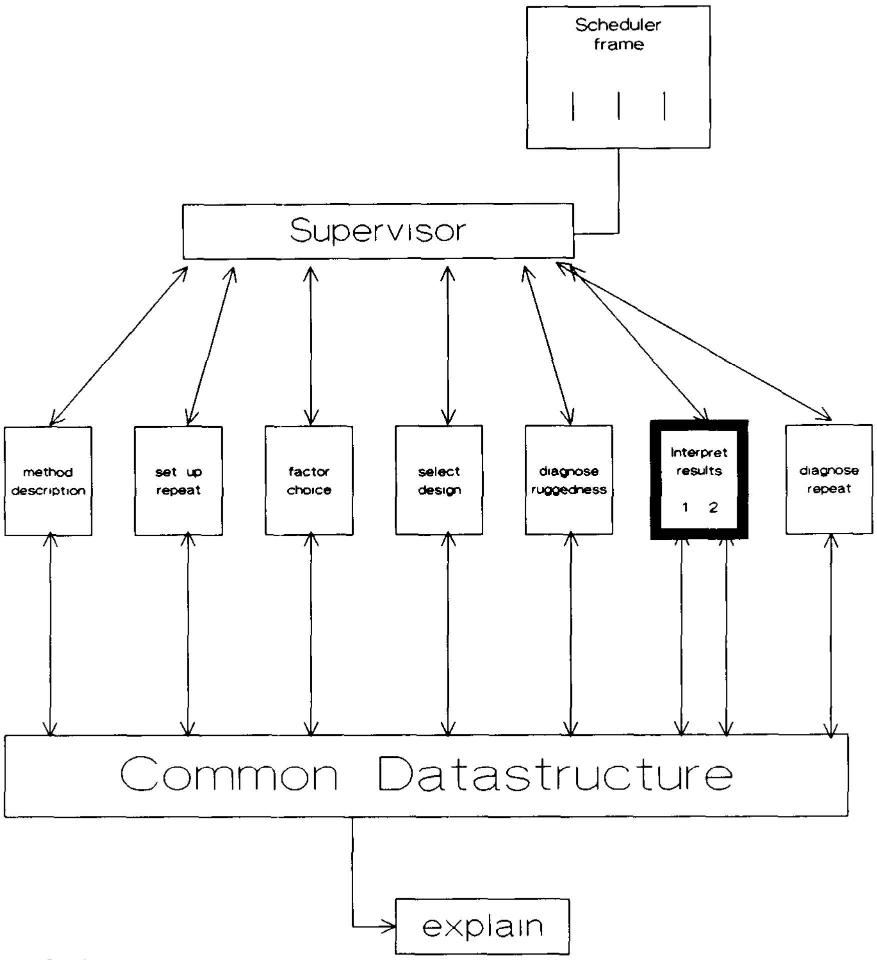

Fig 2 The precision expert system

the method. If a ruggedness test is necessary, many more features are required. Of course, when a ruggedness test follows a repeatability test, the information produced during the first test remains available for the second.

Finishing the method description defines the basis for the following consultation. The system will ask for further information if necessary but the user cannot change his method description after leaving the method description module. If changes are essential, a new consultation must be started.

Common data structure

The basis of every expert system is a description of the objects about which it can reason. All necessary objects must be described in such a way that misinterpretation is impossible. In this case, all objects are represented in a network of frames. An object is described by giving a frame the name of the object and defining a list of all the relevant properties of the object. Every property has a number of possible values which are also defined in the frame. For every object, a so-called O(bject) A(ttribute) V(alue) triplet is created. A typical LC object is a column, which is easily described in a frame (see Table 1). The column is defined by properties (attributes) like length, particle size, functionality, internal diameter, etc., each of which has several possible values. For functionality, for example the list including ODS, nitrile, C8, C18, etc. For column length, the range may be between 2 and 100 cm.

The different objects in a knowledge domain are related. For instance, the object "LC method" has parts like sample preparation, column and detector. Relations between objects can be of a general nature common to different knowledge domains. The definitions of general relations are normally provided by the expert/system shell. A particularly useful example of a general relation between objects is "inheritance". Inheritance allows division of frames into more specific subframes; its use implies that all attributes present in a general frame will always be attributed in the subframes, i.e., subframes inherit certain attributes from the more general frame. An example of the use of inheritance in representing relations between LC objects can be seen in the description of a detector (Table 2). Here, a detector has only two properties: it is always of a certain type (UV, RI or diode array) and it always has a time constant. Real detectors have other properties that place them in a subclass of detectors, e.g., a UV detector will have a wavelength property that distinguishes it from a RI detector. The UV and RI detectors are thus subframes of the detector frame.

Another useful general relation is instantiation, which means specifying the exact feature of an

TABLE 1

Example showing the information in a frame for a column

Object	
Attribute 1	Attribute 2
value 1.1	value 2 2
value 1 2	value 2 2
	value 2 3
Column	
Tradename: Spherisorb, Hypersil, Nucleosil, Partisil, µBondapak, Lichrosorb	
Functionality. C8, C18, SI60, ODS, Nitrile, Phenyl, PAC	
Particle size (µm), REAL	
Column length (cm): REAL	
Batch number: INTEGER	
Internal diameter (mm) REAL	
Column 1	*Column 2*
Tradename: Lichrosorb	Tradename. µBondapak
Functionality. ODS	Functionality: C18
Particle size 4	Particle size 5
Column length: 20	Column length 30
Batch. 23475435	Batch 3498745678
Internal diameter 4 6	Internal diameter: 4

TABLE 2

Example of inheritance detector

Detector	
Type. variable UV, fixed UV, diode array, refractive index	
Time constant (s) REAL	
UV detector	*RI detector*
Wavelength (nm) REAL	RI const . REAL
Attenuation REAL	Temp (°C): REAL
	Range high, low
UV detector 1	
Type variable UV	
Time constant: 0 5	
Wavelength: 276	
Attenuation 0.1	

example. Defining a frame means the introduction of a certain concept into the common data structure. During a consultation, a specific example of the concept will be defined. An instantiation of a frame has only a subset (normally one) of the possible values for each attribute. Examples of instantiation can be seen in Tables 1 and 2; instantiations define so-called *is-a* relations, e.g., in Table 1, column 1 *is-a* column. Because instantiations are created during a consultation, they are not part of the common data structure, because they are deleted after each consultation. Figure 3 summarizes all the objects and their general relations in the common data structure.

Inheritance enables a network of relationships between objects to be defined. There are, however, other types of relations in the knowledge domain that cannot be described with the inheritance concept, e.g., relations between attributes. Such relations can be defined explicitly by a functional

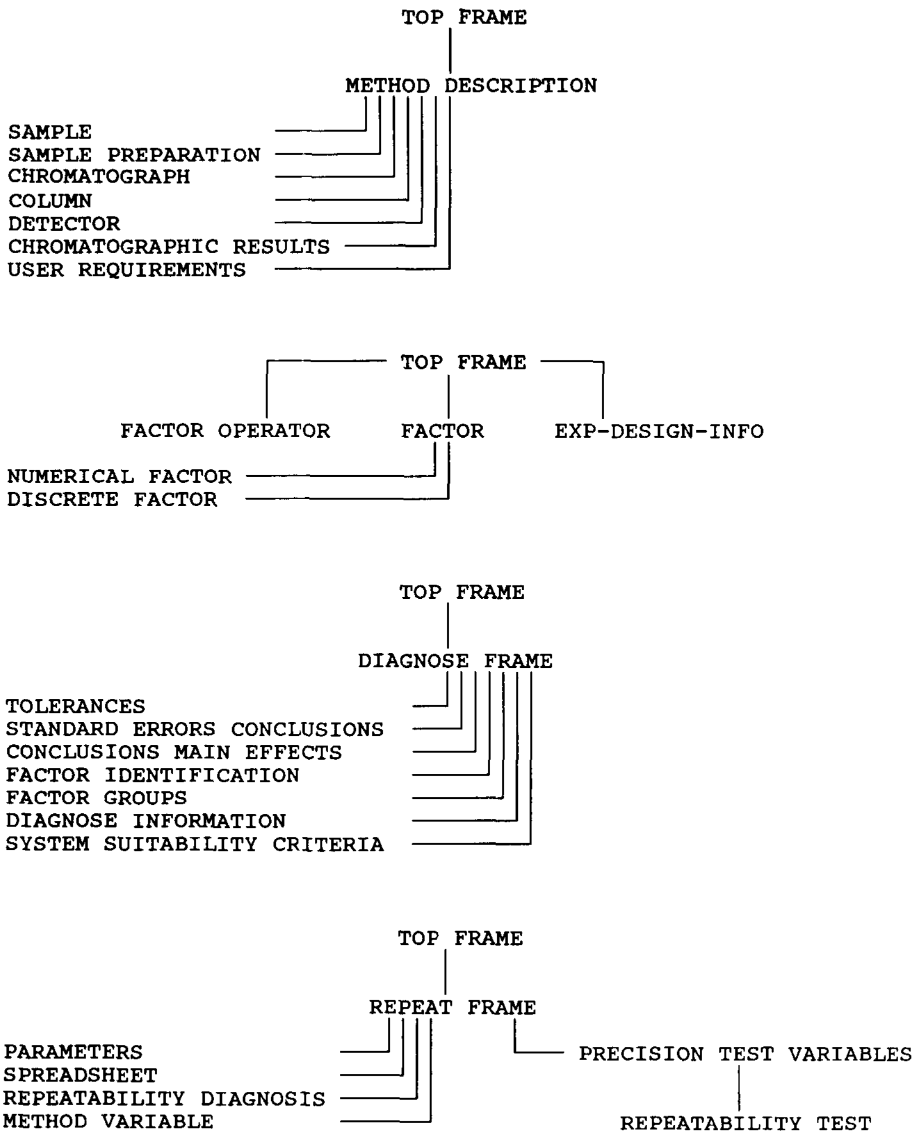

Fig 3. Objects in the common data structure

relationship, in which the names of the attributes are included with the nature of the relationship between them. An example of a functional relationship is the definition that the description of a sample preparation cannot include the attributes shake (min) and sonicate (min) simultaneously, because it is unnecessary to use both sonication and shaking in one procedure. The functional relationships are usually of a more complicated and specific nature than the general relations, and are defined in the language of the expert system, e.g., Lips. This makes them less comprehensible and flexible than the general relations but a functional relationship is used only once or twice so that generalization is unnecessary.

The objects and relations together form the common data structure, which contains much knowledge about LC and method validation. Because it is impossible to change the objects and the relationships during a consultation, the common data structure is the static backbone of the system. The common data structure developed for this expert system provides a basis for a complete description of a procedure for method validation in LC, and could easily be extended for another LC application.

Supervisor

The supervisor contains the knowledge on when to activate which module. This meta knowledge represents the knowledge of a manager who decides which tests are necessary and when they should be done. The decisions of the supervisor are not much related to the activities of the modules. The supervisor only needs information on the input and output parameters of every module and some information from the method description module.

At the moment, the supervisor knowledge is represented in rules (Table 3). The rules act on a simple separate frame, the scheduler (see Fig. 2). The modules cannot operate on this scheduler frame. In the scheduler, information is stored about the state of the modules. This structure is capable of handling the knowledge for the precision-testing system with its eight modules. Because the amount of information stored in the scheduler will become very large when more modules are added, the supervisor is being extended to include more frames and an additional so-called task level between the modules and the planning knowledge.

TABLE 3

Rules in the supervisor

```
(DEFINE-RULE PRECISION TEST 3
  ( doc-string "general rule 100 10-12-87"
    sponsor select-test-sponsor)
   (INSTANCE ?user is user-requirements
     WITH lab-number 1 OR 2)
THEN
   (INSTANCE PREC-TEST IS TEST
     WITH GLOBAL PERFORMANCE CHARACTER-
       ISTIC PRECISION
     WITH CONTRIBUTORY CHARACTERISTIC
       REPEATABILITY
     WITH CONTRIBUTORY CHARACTERISTIC
       RUGGEDNESS))
(DEFINE-RULE PRECISION TEST 2
  ( doc-string "general rule 110 10-12-87"
   ·sponsor select-test-sponsor)
   (INSTANCE ?APP IS APPLICATION
     WITH lab-number 1 or 2)
THEN
   (INSTANCE PREC-TEST IS TEST
     WITH GLOBAL PERFORMANCE CHARACTER-
       ISTIC PRECISION
     WITH CONTRIBUTORY CHARACTERISTIC
       REPEATABILITY))
(DEFINE-RULE PRECISION TEST 1
  ( doc-string "general rule 120 10-12-87"
   :sponsor select-test-sponsor)
   (INSTANCE ?user is user-requirements
     WITH analyst-number 1
     WITH instrument-number 1)
THEN
   (INSTANCE PREC-TEST IS TEST
     WITH GLOBAL PERFORMANCE CHARACTER-
       ISTIC PRECISION
     WITH CONTRIBUTORY CHARACTERISTIC
       REPEATABILITY))
```

Architecture of the system

A complete precision testing expert system can be constructed with the building blocks described above. The system is based on the common data structure of frames that represent all the physical and mental objects necessary for a precision test, i.e., descriptions of the sample, method, tests, results and diagnosis. Working on the framework are the modules that each contain knowledge on a certain part of the test procedure. The modules communicate through the common data structure by placing variable values in it and reading from it. Direct contact between modules (e.g., for communication of variables shared by them) is not possible, thus the supervisor can keep track of all activities in the system. The supervisor can see which modules can be triggered at a certain moment because it contains a list of the input and output parameters of all modules. A module can be activated only if all its input parameters are known; it will be deactivated only when all its output parameters have been established.

The supervisor acts as a kind of switchboard operator connecting modules to each other according to the state of the system at a certain moment. The supervisor also contains knowledge on the priority of the modules if a situation occurs where more than one module can be activated at the same moment. Thus the supervisor decides on the best scheme to follow, using its meta knowledge.

Comparison with a blackboard

At first sight, the architecture described above resembles blackboard architecture [12]. Blackboards are well known artificial-intelligence techniques for the integration of expert systems. Blackboard architecture also uses modules (knowledge sources) that can communicate with each other only via a framework of objects, the blackboard (see Fig. 4). However, in blackboard architecture, the knowledge sources trigger themselves when the state of the blackboard is such that they can contribute to the solution of the problem. The scheduler then decides which of the knowledge sources can be activated. In an ideal situation, several knowledge sources can be activated at the same time. The blackboard architecture offers possibilities for parallel processing. The scheduler therefore does not contain any planning knowledge.

In the architecture used here, the supervisor calls up the modules that can add to the solution of the problem. The scheduler frame contains all the conditions for activating the modules, and this allows the implementation of planning knowledge about the activation of the modules. Recently, the literature on blackboards has shown a shift to-

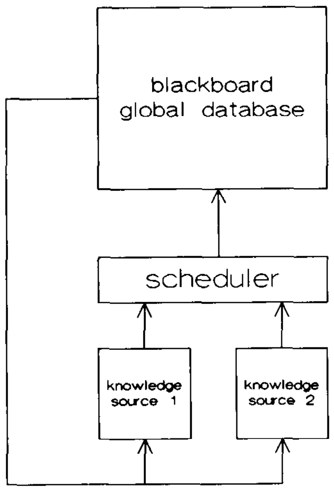

Fig 4 Blackboard architecture

wards the implementation of planning knowledge similar to that used here. An example of such a blackboard was described recently [15].

In principle, the architecture used here does not allow processes to run in parallel unless they are specified as possible simultaneous processes. For the application of precision testing in LC method validation, however, this is not (yet) a disadvantage.

PROGRAM FLOW

Consultation of the system normally starts with specifying the needs of the user (Table 4A). The appropriate modules are then loaded by the system. In a complete consultation, all modules are loaded from initial method description to diagnosis of the ruggedness test. In the method description module, the user can enter all the information that he has on the method (Table 4B). The supervisor then decides, on the basis of the expected usage, which tests are needed (repeatability, ruggedness or both).

This information is transferred to the scheduler frame, which decides on the next step. In general, it will be a repeatability test and the system will activate the relevant module, which uses information about the expected usage to select a suitable test. It also activates a spreadsheet in which the test is implemented. After the user has done the test and entered the results, the expert system activates the next module to diagnose the results and advise on their acceptability, etc. (see above).

The same procedure applies when the ruggedness part is activated. Normally, ruggedness test-

TABLE 4

Screen dumps of the integrated precision-testing system

(A) Intelligent scheduler

```
Supervisor control  NO
Describe method· YES
Perform repeatability test· NO
Select factors to be tested  YES
Select experimental design  YES
Perform diagnosis: YES

Run a total consultation  NO

Start consultation

Loading  part
         file
```

(B) Chromatograph

```
Screens > chromatograph questions > options

Main questions and answers                        Related answers
Flow rate (ml min⁻¹)     : 1 5                     Minimum solvent (%)    25
Number of solvents       : 2                       Solvent 1 (%)          75
pH                       . 2 5                     Solvent 2 (%)          25
Buffer conc  (M)         ·                         Solvent 3 (%)          ·
Additives                :                         Solvent 4 (%)
Injection volume (μl)    . 10
Temperature mode         CONTROLLED                Minimum additive (%)  ·
                                                   Additive 1            : 0 % (  )
                                                   Additive 2              0 % (  )
                                                   Additive 3              0 % (  )
                                                   Additive 4              0 % (  )
                                                   Additive 5              0 % (  )
                                                   Temperature ( ° C)      40
```

(C) Selected factors

```
> sample > chrom  > detector > column > data > options <

Sample prep.          Chromatograph          Detector               Column
factors.              factors                factors                factors

weight          –     pH            *        RI-range        –      Manufacturer· *
shake time      –     temperature  : –       filter         · –     Batch         –
sonicate time   *     buffer       . –       wavelength      *
heat temp       –     solvent      . *       UV-time-con     –
pore size 1     *     additive       –       RI-time-con     –      Data handling
pore size 2     –     flow rate    . –                              factors
wash vol        –
extract vol     –                                                   1 factor  *
extraction      –
centrifuge      –
dilution        –

No modifications made
```

TABLE 4 (continued)

(D) Selected experimental design

Experimental design SATURATED-FRACTIONAL-FACTORIAL-DESIGN
Number of factors 7
Number of levels 3
Number of dummy factors 0
Number of experiments· 15

Show factors
Wavelength warning

(E) System suitability criteria

The results of any analysis should always fall within the ranges of the values given below

	Minimum found at exp.: 6		**Maximum found at exp.: 14**	
Comp	1	2	1	2
RT	217 333	277.0	267 333	439 0
Area	58 449	572.325	255.746	102 058
Height	646 832	7772 02	2841.85	2096.0
C area	297 621	296.896	6.979	7.0895
C hgt·	296 535	296.271	6 982	6 8935
P count	2 5399	5.7842	3.1108	6.0617
Result	2425 15	2614.22	3637.18	2871 17

(F) Main effects

This screen shows the input given by the user
It also gives a short description of the results.
When a change is made to the input values, the results may change
A better description of the results can be obtained by selecting the appropriate screen from the 'screens' menu

Usage. ONCE ONLY
Length of run· > 10 ⩽ 25
Number of users· ⩾ 3
Number of instruments. ⩾ 3
Number of laboratories 1 OR 2

Preparation repeatability test: REPEAT 1
Injection repeatability test REPEAT 5

Write ss

ing is only done after a successful repeatability test, but the user may elect to forego the repeatability test and proceed with the ruggedness test. The ruggedness part consists of four modules (see Fig. 1), which are usually consulted in sequence. The factor choice module advises on which factors are likely to influence method performance (Table 4C). The select design module then advises on a suitable experimental design to test these factors with a minimum of experiments (Table 4D) and a C program is activated in which the user enters his experimental results. The diagnosis module is then activated. If the results are satisfactory, the module will report this and provide system suitability criteria (Table 4E). If the results are not satisfactory, the user is informed of the main effects that are outside the tolerance limits and of the factors causing the problem (Table 4F). A solution to the problems that does not affect the method itself, is to respecify the levels for testing the factors. When these levels are specified at narrower intervals, the method is more likely to pass the ruggedness test but, of course, operation of the method in the laboratory will have to be kept under more rigid

TABLE 5

Results of the test case

Method description

Sample	
name	aspirin, salicylic acid
formulation	tablet
of components	2
Sample preparation	
take sample	take no of doses
No of tablets	1
add solvent (ml)	250
add internal standard	no internal standard
dissolve sample	sonicate
sonicate minutes (min)	15
dilution 1	*no dilution*
dilution 2	no
extraction	filtration
filter pore size (μm)	10
Chromatograph	
solvent no	2
solvent 1 (%)	75
solvent 2 (%)	25
pH	2 5
buffer conc (M)	
injection volume (μl)	10 0
temperature ($^\circ$C)	40
flow rate (ml min^{-1})	1 5
Column	
tradename	Spherisorb
functionality	C18
particle size (μm)	7 0
column length (cm)	25
batch number	1
internal diameter (mm)	4 0

Detector	
type	variable UV
attenuation	0.1
time constant (s)	0 1
wavelength (nm)	295
Results	
minimum resolution	4.0
min retention time (s)	240
overall run time (s)	600
worst peak symmetry	1 4
Requirements	
usage	$> 10 \leqslant 25$
No. of lines	4
purpose	stability indication
regulatory	USA
standard method	USP
interlaboratory	$\geqslant 3$
number of analysts	$\geqslant 3$
average run length	$> 10 \leqslant 25$
number of instruments	$\geqslant 3$

Ruggedness test

Factors chosen by the system

Factor	Nominal level	Lower level Upper level
Sonication time	15	12
		18
Pore size	10	5
		20
Data handling	–	–
pH	2.5	1 5
		3 5
Solvent	25	20
		30
Manufacturer	Spherisorb	other
		other
Wavelength	295	290
		300

TABLE 5 (continued)

Experimental design
Reflected saturated fractional factorial design
11 factors
4 dummy factors
3 levels

Interpretation of experimental results

Main effect	Factor
24.0%	sonicate time
23.7%	pore size
24.0%	solvent
27.2%	manufacturer
22 5%	wavelength

Repeatability test

Test advised by the system
10 times repeated injection of sample
5 times repeated sample preparation

Interpretation
Relative standard deviations

Injection of sample	<1%
Sample preparation	<1%

Diagnosis
No problems

control. In this case, the factor choice module is activated again, the factor levels are adapted to fit the test, and the whole procedure is repeated. Control of this operation is again placed in the scheduler frame that keeps track of the number of times a module is activated and also checks if the new activation has yielded a result.

The architecture around the scheduler frame also makes it possible to load only one module that can be consulted as a stand-alone expert system. For some modules, this can be practical.

The prototype was developed in the Goldworks expert-system shell, Version 1.1 [16]. An IBM PC/AT with 8 Mbyte extended memory was used. The spreadsheet packages Lotus 123 and MS-C were used for implementation of the tests.

RESULTS OF A TEST CASE

To illustrate the capacities of the system, the test case used was a full in-house precision test of an LC method for the determination of aspirin and salicylic acid. Both repeatability and ruggedness tests were done and the results were interpreted by the system. Real experimental data were used so that a comparison of the system performance in a real-life situation was possible. Table 5 shows a complete description of the results.

The conclusion of the system for the repeatability test was that the method was satisfactory within the specified limits. As the method had previously undergone similar tests [9], this was to be expected.

The set-up of the ruggedness test corresponded with the expert's ideas and suggestions. The only difficulties appeared in the interpretation of these results. The system reported several problems that were not encountered in the real ruggedness test. For instance, the system suggested that the method was not rugged in the measurement of the concentrations of the analyte. As this is the crucial parameter, such a conclusion would be serious, but in reality these problems were not encountered. The difficulty may be due to too rigid interpretation by the system or to problems overlooked in real life. To test this, a full reproducibility study is needed.

Conclusion

The expert system described here is a prototype that must still undergo a full evaluation study, which will be reported elsewhere. The philosophy of using a common data structure as a blackboard for various modules will be investigated further. The possibility of adding new modules for testing accuracy and selectivity will be crucial for success. The addition of modules on extensions to the ruggedness test and other modules on method development will also be investigated.

Part of this research was supported by the European Community in Esprit project P1570 Expert Systems in Chemical Analysis.

REFERENCES

1 M A Tischler and E A. Fox, Comput Chem , 4 (1987) 235
2 C G Enke, A P Wade, P T Palmer and K J Hart, Anal Chem , 59 (1987) 1363A.
3 S Moldoveanu and C A Rapson, Anal Chem., 59 (1987) 1207

4 K. Janssens, W Dorinne and P. van Espen, Chemom. Intell. Lab. Syst., 4 (1988) 147.

5 G.J Kleywegt, H J Luinge and H.A van 't Klooster, Chemom. Intell. Lab. Syst., 2 (1987) 291.

6 D. Goulder, T. Blaffert, A Blokland, L Buydens, A Chhabra, A. Cleland, N Dunand, H. Hindriks, G. Kateman, H. van Leeuwen, D. Massart, M. Mulholland, G Musch, P. Naish, A. Peeters, G. Postma, P. Schoenmakers, M. Desmet, B. Vandeginste and J. Vink, Chromatographia, 26 (1988) 237.

7 A Peeters, L. Buydens, D.L. Massart and P.J Schoenmakers, Chromatographia, 26 (1988) 101

8 P.J. Schoenmakers, N Dunand, A. Cleland, G. Musch and T Blaffert, Chromatographia, 26 (1988) 37

9 M Mulholland, J A van Leeuwen and B.G.M Vandeginste, Anal. Chim. Acta, 223 (1989) 183.

10 J A. van Leeuwen, B.G.M Vandeginste, G. Kateman, M. Mulholland and A Cleland, Anal Chim. Acta, 228 (1990) 145.

11 M Mulholland, N Durand, A. Cleland, J.A. van Leeuwen and B G.M. Vandeginste, J. Chromatogr., 485 (1989) 283

12 N.P Nii, the Artif. Intel. Magazine, (1986) 38

13 W.J. Youden and E H Steiner, Statistical Manual of the Association of Official Analytical Chemists, AOAC, Washington, DC, 1975

14 M. Mulholland, P.J. Naish and D.R Stout, Chemom Intell. Lab. Syst., 5 (1989) 263.

15 B.R Maitre, T. Laasri, F Mondot, F Charpillet and J P Haton, paper presented at the 9th International Workshop on Expert Systems and Their Applications, Avignon, May, 1989.

16 J.A. van Leeuwen, B G M Vandeginste, G J Postma and G. Kateman, Chemom Intell. Lab Syst., 6 (1989) 239.