Using the FOCUS architecture for developing knowledge-based front ends: a KBFE for forecasting

A Prat, M Catot, P Fletcher, J Lorés and R Southwick*

Statistical packages, optimisation packages and numerical-algorithms libraries are widely used in industrial and scientific development environments. They represent an enormous body of very complex and valuable knowledge that is becoming increasingly difficult to access. End users of these systems have to cope simultaneously with the intricacies of the software and with the increasing complexity of the application-domain problems. For these systems, knowledge-based front ends can provide co-operative assistance to end users, enabling them to use the systems successfully, while preserving the knowhow contained in the libraries and packages. They are also valuable in extending the software's working life.

FOCUS is an ESPRIT-2 project whose goal is to develop generic tools and techniques for constructing and maintaining KBFEs for open user systems (e.g. libraries, reusable software components) and closed user systems (e.g. free-standing software, packages) for industrial and scientific applications. The participating partners are drawn from both industrial and academic institutions, providing a wide crosssection of software researchers, producers and users, and, although there are substantial research objectives, the project has taken a pragmatic approach, with the commercialisation of products developed playing a prominent role.

The paper describes part of the work undertaken during the twenty-fourth month of the ESPRIT II project, which began in December 1988 and ran for four years. The aims of the FOCUS project are stated, and a core element of the emerging FOCUS design strategy is described, namely a separable architecture for knowledge-based front ends, concentrating on the front-end harness, the problem solver and the back-end manager components. Some KBFEs developed using this architecture, and the tools created within the project that operate in the domain of forecasting, are also described.

Keywords: knowledge-based front end, interface separability, back-end manager, user interface, problem solver.

A wide and ever-growing range of application packages and libraries are now available to assist computer users in addressing a large variety of computing tasks. Some of these users write computer programmes in the traditional sense, and so are mainly interested in incorporating computational units (e.g. from libraries) in their programs. However, an increasing number of users are 'non-programmers' who make use of their chosen application package(s) rather than program in general purpose programming languages. Using the term 'user system' to refer to both application packages and libraries, we observe that computer users rightly expect that such systems should possess a number of qualities such as reliability, efficiency, flexibility etc., but there is also a discernible, growing demand for 'ease of use', particularly in the initial learning stages, and for continuing 'support in use' as the user gains more confidence and addresses more complex topics within his/her chosen user system. It is considered that the most effective way to provide such assistance is through the use of knowledge-based front ends (KBFEs) which contain explicitly represented knowledge of the user system and of the host environment in which that system is to be used.

PROGRAMME

The objectives of the projects were achieved within a programme of work which can be viewed as a three phase exercise. In Phase 1, state-of-the-art prototype KBFE systems (e.g. GLIMPSE (Nelder and Wolstenholme, 1986) were studied, and various other investigations carried out, to define and generate an initial set

Departmento de Estadistica, ETSEIB, Avda, Diagonal 647 08028, Barcelona, Spain
*Quintas, 2100 Ging Road, Palo Alto, CA 94303, USA
Paper received 5 December 1991. Revised paper received 21 September 1992. Accepted 22 September 1992

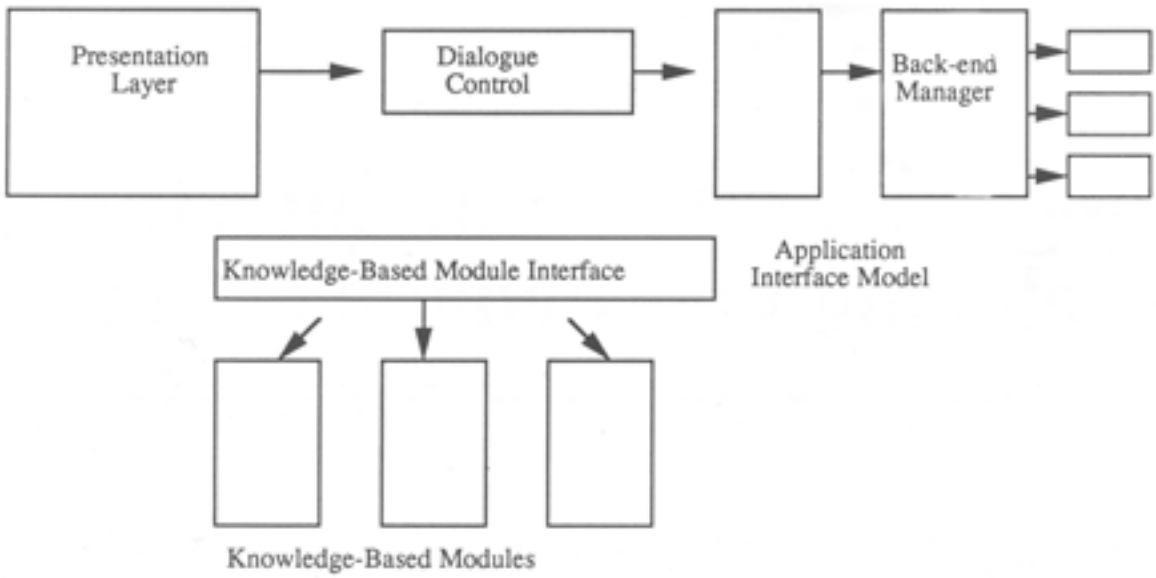

Figure 1. FOCUS architecture

of generalised tools, techniques and methodologies. In particular, an overall FOCUS architecture for KBFE construction was devised.

In Phase 2, first generation FOCUS KBFE prototype systems were constructed to validate the principles of the architecture and provide feedback to tool developers preparing the next set of tools. In Phase 3, the tools, together with refinements in the techniques and methodology produced by evaluation exercises were applied to produce a second and third generation of FOCUS KBFE prototypes, with particular reference to the perceived needs of the numerous marketplaces in which FOCUS technology could potentially be applied. In all phases of the project, on-site user evaluation took place, and feedback will be given to the developers of the KBFE prototypes and the tools.

KBFE architecture

The KBFE architecture described below (see Figure 1) is an extension of the Seeheim model (Green, 1985). It consists of a presentation layer, a dialogue control layer and an application interface model. The application interface layer contains an explicit application model. The purpose of this is to provide a clear and clean interface between the front and back ends.

In addition to these three user-interface layers (Pfaff, 1983), the KBFE also incorporates a number of intelligent knowledge-based modules (KBM) which encode domain knowledge useful for user guidance and assistance and a back end manager (BEM) which deals with implementation details of the back end applications. The rest of the KBFE, with the exception of these modules but including their communication and support software, constitutes the front end harness (FEH).

The various modules which comprise the KBFE can be distinct software entities. They need not be physically located on the same machine and can communicate over networks if necessary. Communication is accomplished through a standard message structure (Edmonds *et al.*, 1992).

Front-end harness

The front-end harness consists of three modules, as follows.

Presentation layer

This is the software component responsible for the management of the surface level human–computer interface. It is built on top of the X Windows system and interprets a high level description on the interface to produce interaction objects on the display. These distance the interface builder from much of the tedious programming required by the X Windows toolkits, and can be used, accommodating whatever 'look and feel' is required. The interaction objects are highly autonomous and store a good deal of what used to be called dialogue state information.

The presentation layer transforms user interaction into message structures which are, in turn, transmitted to the dialogue control module. These messages can be preprogrammed when an object is created or derived during an interaction sequence. This affords a considerable degree of flexibility in their structure and content.

Dialogue control

This module is central to the FEH. It manages and coordinates communication between the other FEH components and at the same time controls the dialogue with the user.

For simplicity, the dialogue control module treats all of the modules with which it communicates in the same way. Communication is effected through message structures and a protocol for their transmission. In addition to their content the messages contain an identifier and information relating to their route. Messages may give rise to a reply which will return as another message.

Application interface model

In order for the dialogue controller to communicate with the back-end manager it is necessary to map between the dialogue state and the back end tasks. This is accomplished by the application model, which contains representations of the back end functionality.

In classical user-interface management systems this model has often been implicitly embedded with the dialogue controller, thus negating the benefits of interface separation. In the FEH this model is explicit and with open user systems such as software libraries which may not have a great deal of user-interface structure.

The knowledge based model interface and the application interface model can in fact be implemented as a single software module.

A more detailed description of the front end harness can be found in Edmonds and McDaid, 1990.

Knowledge-based modules

Domain knowledge is encoded in a series of *knowledge modules*. Knowledge is represented declaratively using logic, in the form of PROLOG clauses, as the representational formalism. Note that domain knowledge is not application dependent: back end applications provide information that assists in the solution of a problem in the domain.

There may be several knowledge modules, each covering a different area of information. These include domain specific knowledge, which describes relationships between domain entities, application specific knowledge, which contains information on the use of an application package and strategic information on the use of an application package, and strategic knowledge, which contains methods useful for finding solutions to a particular problem.

Inference using this knowledge is performed using a logic-based problem solver (PS). The PS is an inference engine that incorporates a backward reasoning search strategy. This search strategy is the same as that used by PROLOG, and for a good reason: the PS solves queries just as PROLOG would, but has the additional functionality that makes it useful for use in a KBFE setting. The use of a PROLOG-based inference engine is well known (see, for example, Clark and McCabe, 1982). The problem solver is a conceptual descendent of one such system, APES (Hammond, 1982), and owes much to the original APES system.

Control of the inference process is specified through a set of meta-level statements in the domain programme. These are used, for example, to declare that certain information should be requested from the user, or from a back end application via the BEM. This separation of data and control facilitates a modular construction of the domain knowledge bases.

The problem solver has several notable features:

- *Blending transformer:* The PS does not employ a meta-interpreter, as do many PROLOG-based inference engines. Instead, it uses a programme transformation that 'blends' meta-level functionality into an object-level programme (Cosmadopoulos and Southwick, 1989).
- *Separation of data and control:* The PS operates on a logical model of a problem domain; information about how the problem solving process should be controlled is kept separate. The result is a clean representation of domain knowledge.
- *Managing user dialog:* If a problem solver interacts with a user, it must be prepared to behave in a manner acceptable to the user. The PS caters to the needs of users who may not know the answer to a system-generated question, or who change their minds.
- *Defaults, suggestions and help:* These guide and assist the user in answering unfamiliar questions, and can be either static (in the form of canned text), or context dependent (and computed 'on the fly').
- *Validity checking data:* Data entered by a user can be checked and immediately rejected if it does not satisfy predetermined constraints.
- *Explanation:* Currently, the PS offers standard 'how' and 'why' rule-trace explanations. In the case of explaining why user-entered data is invalid, however, a facility explains the reason for the failure.
- *Reason maintenance:* A system that reasons in a changing world needs some way of discovering the ramifications of changes that have occurred, and updating its view of the world accordingly. In the PS, this is accomplished through the use of a reason maintenance system (RMS). This records the logical dependencies that hold amongst program data, and allows a problem solver to exploit that information. With this record, an RMS provides several benefits: it avoids redundant computation, deals with inconsistency, manages the search process, and supports non-monotonic reasoning.

Communication between KBMs and other system components is managed by the dialogue control module, using the standard message structure found throughout the FOCUS system. In order to get some information from the user, for example, a message is dispatched to the FEH, which handles all the interaction with the user, and then sends a return message containing the appropriate data. This architecture allows a clean distinction between problem solving and interacting with external data sources, and it encourages a purely logical knowledge representation.

Back-end manager

While the front end harness and the knowledge based modules are concerned with the 'whys' and 'wherefores' of what the end user is or should be doing it is the BEM (Prat *et al.*, 1990) that takes care of the 'how', thus maintaining interface separability (Edmonds, 1992). Its prime responsibility therefore is that of mapping an application independent specification of the 'what' into an executable specification of the 'how' with respect to a particular application or package (i.e. mapping from a semantic to a syntactic specification), executing the application, and passing the relevant information back to the appropriate KBFE component.

To carry out this role the BEM must be 'programmed' by an expert user with knowledge about the syntax, semantics, functionality and environment of the application software. This information is stored in the form of specification objects (tasks and actions), back end I/O objects (templates and definitions extraction), and an environment knowledge base. Figure 2 illustrates the structure of the BEM showing its relationship with the rest of the KBFE architecture.

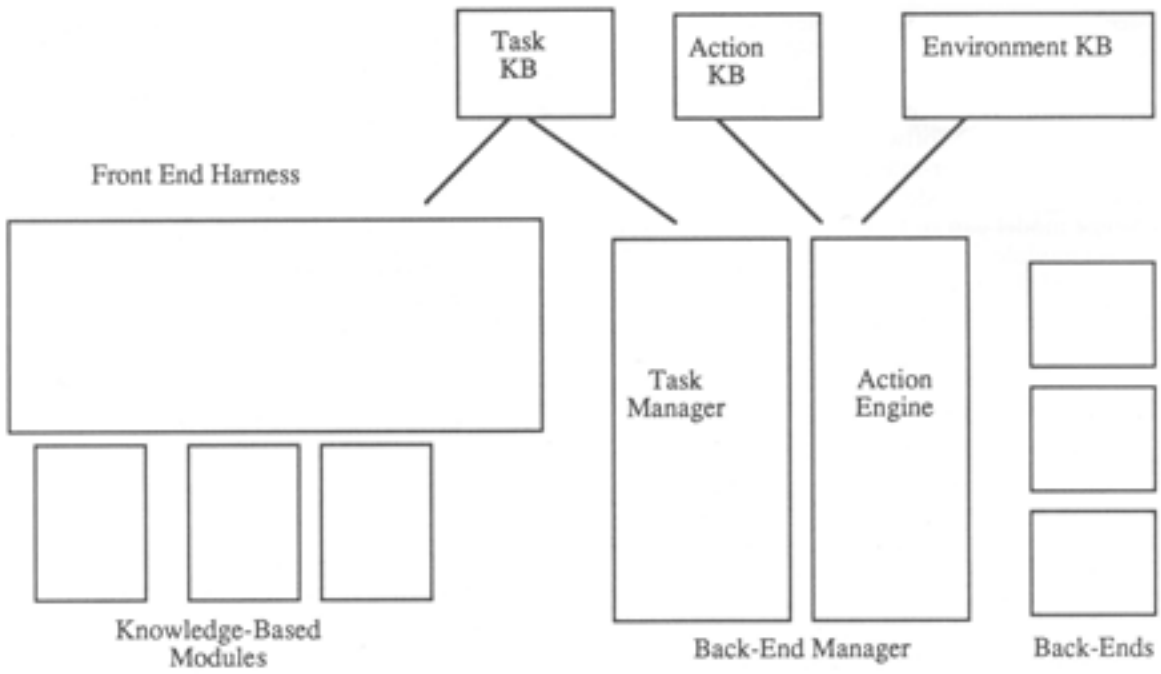

Figure 2. Back-end manager

KAFTS: A KBFE FOR FORECASTING

The Box–Jenkins methodology (Box and Jenkins, 1976) for modelling, and forecasting time series has long been considered one of the most efficient techniques, and it has been extensively used, especially for univariate data. The strategy of model identification, estimation and diagnostic checking requires that the person responsible for making the forecasts has considerable experience and knowledge. When the number of series to forecast is great (as in the case of a bank organisation) the task of modelling 'by hand' becomes almost impossible.

Here we present KAFTS, a KBFE for implementing the Box–Jenkins strategy, built using the FOCUS architecture on a back end subsystem consisting of standard statistical software and specialised custom built software.

Introduction

The class of ARIMA models provides a flexible set of parametric models that are useful in modelling a great variety of time series. In the case of seasonal data, the multiplicative form of such a model is

$$F_p(B)\, F_P(B^S)\,(1-B)^d\,(1-B^S)^D Z_t = q_0 + q_q(B)\, Q_Q(B^S) a_t \quad (1)$$

where S is the seasonal period, B is the backshift operator ($BZ_t = Z_{t-1}$), $F_p(B)$ is a polynomial of degree p known as the autoregressive polynomial for the nonseasonal part, $F_P(B^S)$ is a polynomial of degree P in B^S known as the autoregressive polynomial for the seasonal part, $q_q(B)$ and $q_Q(B^S)$ are the moving average polynomials of degree q and Q for the nonseasonal and seasonal part, respectively, d and D are the degrees of regular and seasonal differencing of the original series needed to achieve stationarity, and q_0 is a constant. We assume that Equation 1 is such that the zeros of all the polynomials are outside the unit circle, there are no common zeros between the autoregressive and moving average polynomials, and $\{a_t\}$ is a white noise sequence. Z_t are the observations or a Box–Cox transformation of them.

The well known strategy for modelling time series data developed by Box and Jenkins (see Equation 1) consists mainly of three steps:

- *Identifications:* Given the data, a guess is made for the values of d, D, p, q, P and Q. This guess is based in the examinations of the estimated autocorrelation function and the partial autocorrelation function of the data.
- *Estimations:* Given a model identified in the previous step, the parameters of this model (the coefficients in the polynomials) are estimated using the maximum likelihood method.
- *Diagnostic checking:* A set of diagnostic checks is applied to the residuals of the estimated model in order to assess its validity.

These three steps are applied repeatedly until a useful model is obtained.

Next, the forecast for future observations is obtained from the retained model by taking conditional expectations:

$$Z_t(1) = E(Z_{t+1}/Z_t, Z_{t-1}, \ldots)$$

where $Z_t(1)$ is the forecast made at time t for the value Z_{t+1} $(1 = 1, 2, \ldots)$

End users and objectives

The objectives of the KBFE are to provide a usable knowledge based interface to forecasting software.

The KBFE is aimed at two sets of end users. The system is to be used operationally by end users with little or no knowledge of the domain. For this group of end

Table 1. Information flow

Predicate name	Instantiated variables	Noninstantiated variables	Source of instantiation	Component responsible for instantiation
Time series	—	Data	User	FEH
Tentative model identification	Data	Identifiers	Back-end software (Idaut)	BEM
Estimate parameters	Data, identifiers	Model, diagnostics	Back-end software (SCA)	BEM
Check model	Data, identifiers, model	Diagnostics checked model	KBM domain rules	PS
Forecast	Data, identifers, checked model	Forecasts	Back-end software (GENSTAT)	BEM

users guidance must be at the operational level, both in terms of the use of the user interface and the practicalities of modelling time series in terms of data manipulation. Here, the KBFE will appear relatively automatic. However, provision must also be made for a second more statistically sophisticated end user: a time series analyst who may have to intervene with a particular time series and perform some manual manipulation of the modelling process. Assistance for this end user must, accordingly, be based on knowledge of the time series domain.

Encoding domain knowledge

By taking away many of the complicating considerations such as user and back end interaction, the problem solver has permitted the domain knowledge to be encoded in a declarative and natural manner. The following is an extract from the domain program which clearly defines how a forecast is obtained.

```
Forecast: time series (Data),
    tentative-model-identification (Data, Identifiers),
    estimate-parameters (Data, Identifiers, Model,
    Diagnostics)
    check-model (Data, Identifiers, Diagnostics, Model,
    Checked-Model)
    forecast (Data, Identifiers, Checked-Model, Fore-
    casts)
```

In terms of the encoding, it is irrelevant where the information comes from to satisfy each of the PROLOG goals. Table 1 shows the resulting information flow generated by the above definition of a forecast.

Back ends and BEM

The back end subsystem consists of three separate modules: Idaut (a FORTRAN program which performs the automatic identification of a tentative model given a time series), SCA (used for the estimation of model parameters and obtaining diagnostic information), and GENSTAT (used for data exploration and manipulation and forecasting). Idaut is used in a 'batch' mode, while SCA and GENSTAT are used interactively.

```
task (estimate-parameters (Data Identifiers, Model,
Diagnostics)):
  action-engine (specify-model      % action name
    [Data, Identifiers],            % input parameter
    [Model],                        % out parameters
    SCA],                           % backend/package
  action-engine (get-diagnostics,   % action name
    [Model],                        % input parameters
    [Diagnostics],                  % output parameters
    SCA).                           % back end/package
```

The above listing illustrates how a back end independent information request, or task from the knowledge based module, is mapped into a back-end specific (in this case SCA) set of actions. The action engine then executes both 'specify-model' and 'get-diagnostics' using TPL and XTL (Prat *et al.*, 1990) to generate the input and extract the appropriate output in order to instantiate 'model' and 'diagnostics'.

CONCLUSIONS

The forecasting KBFE described has provided a very useful testbed for the FOCUS architecture and has in principle vindicated the usefulness of the idea of interface separability. It has provided useful input to both tool developers and the developers of KBFE design methodologies. It will be further developed to take advantage of the new tools and techniques coming from the FOCUS project to include more sophisticated diagnostic checking and intervention analysis.

Two main issues have arisen in the development of this prototype. Firstly, the components of the architecture whilst providing a powerful presentation mechanism for the end user have not yet addressed sufficiently their interface to the KBFE developer. More effort is needed to develop them into a suitable KBFE development environment. Secondly, whilst providing the end user with the ability to control his/her presentation, by presenting interaction objects which can be moved, resized, scrolled etc., it still remains unclear as to what extent control of the presentation needs to be retained in some way so that users do not build themselves a screen that is overloaded with such interaction objects.

ACKNOWLEDGEMENTS

This work has been partially funded by the EC through ESPRIT 11 (Project 2620) and by the CICYT (Spanish Government) (TIC 880643). Ernest Edmonds provided valuable comments on an early draft of the paper.

REFERENCES

Box, G. E. P. and Jenkins, G. M. (1976) *Time Series Analysis; Forecasting and Control* Holden Day

Clark, K. L. and McCabe, F. G. (1982) 'PROLOG: a language for implementing expert systems' Machine Intelligence Vol 10 pp 455–470

Cosmadopoulos, Y. and Southwick, R. W. (1989) 'Using meta-level information for expert systems control: a 'blending' transformer approach' *in Research and Development in Expert Systems VI* Shadbolt, N. (Ed.) Cambridge University Press pp 54–65

Edmonds, E.A. (Ed.) (1992) *The Separable User Interface* Academic Press

Edmonds, E. A., Murray, B. S., Ghazikhanian, J. and Heggie, S. P. (1992) 'The re-use and integration of existing software: a central role for the intelligent user interface' People and Computers VII, Proceedings of the BCS HCI '92 Conference, York, Cambridge University Press

Edmonds, E. A. and McDaid, E. (1990) 'An architecture for knowledge-base front ends' Knowledge-Based Systems Vol 3 No 4 pp 221–224

Green, M. (1985) 'Report on dialogue specification tools' *in* Pfaff, G. E. (Ed.) *User Interface Management Systems* Springer-Verlag pp 9–20

Hammond, P. (1982) 'APES: a user manual' Technical Report 82/9, Department of Computing, Imperial College, London, UK

Nelder, J. A. and Wolstenholme, D. E. (1986) 'A Front End for GLIM' Computer Science and Statistics: Proceedings of the 18th Symposium on the Interface

Pfaff, G. E. (Ed.) (1985) 'User interface management systems' Proc. Workshop on User Interface Management Systems (1983)

Prat, A., Catot, J. M., Lores, J. and Fletcher, P. (1990) 'The back-end manager – an interface between a knowledge based front end and its applications subsystems' Knowledge-Based Systems Vol 3 No 4 pp 225–229

BIBLIOGRAPHY

Edmonds, E., McDaid, E., Prat, A., Fletcher, P. and Lorés, J. (1990) 'Systems architecture and KBFE/back-end interface specifications' FOCUS/LUTCHI/UPC/5/4.1-C (Internal Esprit Report)

NAG Fortran Library Manual (1990) NAG Limited, Oxford, UK

Prat, A., Ginebra, J., Catot, J. M. (1989) 'Expert system for forecasting' Proceedings of Development of Statistical Expert Systems, Eurostat News, pp 342–347

Prat, A. M. and Catot, J. M. (1985) 'Incorporating expertise in time series modelling: the STATXPS system' Statistical Software Newsletter Vol 11 (Aug) pp 55–62

Southwick, R. W. (1990) 'A reason maintenance system for backward reasoning systems' Proc. 5th International Symposium on Methodologies for Intelligent Systems, Knoxville, Tennessee, USA pp 102–109