A DATA MODEL FOR KNOWLEDGE REPRESENTATION IN COLLABORATIVE SYSTEMS

Thomas Tamisier[1]* and Fernand Feltz[2]

Centre de Recherche Public - Gabriel Lippmann,Department Informatics, Systems, Collaboration (ISC)
41, rue du Brill, L-4422 Belvaux, Grand-Duchy of Luxembourg
**Email:* tamisier@lippmann.lu

ABSTRACT

Decision support systems are nowadays used to disentangle all kinds of intricate situations and perform sophisticated analysis. Moreover, they are applied in areas where the knowledge can be heterogeneous, partially un-formalized, implicit, or diffuse. The representation and management of this knowledge becomes the key point to ensure the proper functioning of the system and to keep an intuitive view upon its expected behavior. This paper presents a generic architecture for implementing knowledge-base systems used in collaborative businesses, where the knowledge is organized into different databases, according to the usage, persistence, and quality of the information. This approach is illustrated with Cadral, a customizable automated tool built on this architecture and used for processing family benefits applications at the National Family Benefits Fund of the Grand-Duchy of Luxembourg.

1 INTRODUCTION

Automated reasoning systems have shifted from experimental trials within a specialized community to multi-purpose tools, used in a wide variety of real domains. The purpose of these techniques is to provide a model of the reasoning, as well as a means to perform it, in a mechanical, standardized, and justified way (Gordon, 1994). Decision support systems have in fact taken a significant importance in solving problems that can be formalized through inference rules and combinational cases, notably for identification, planning, optimizing, and decision making tasks (Fehling, 1993; Groothuis et al., 2000). These techniques also gain ground in the juridical and administrative domains, especially for the efficient and equitable handling of problems consisting of opening rights or according some pre-defined status (cf. the Softlaw / RuleBurst Expert System).

In this context, we are engaged in a long-term project with the *Caisse Nationale des Prestations Familiales* (National Family Benefits Fund) of the Grand-Duchy of Luxembourg for the development of a customized decision support system. The system, called Cadral (for *CALcul du DRoit ALlocataire* or Claimant Allowance Calculation), will be used to process applications for opening rights relative to the attribution of public benefits, where the automatic treatment of the demands appears as the only means of coping with the continuously increasing amount and complexity of the work in response to the demographic and economic expansion of the country. Indeed, the administrative procedures handled here are very complex because of the local open-economy where individual cases pertain to different national, supra-national, and bilateral legal frameworks.

In the forthcoming version, Cadral sends a reply to an application for benefits. The information and material supplied through the application is formalized and passed as inputs to the engine processing the rules. Upon completion of the inferences, the claim is accepted, rejected, or Cadral asks for more information or additional documents. An extension performing the automatic calculation of the benefits is targeted for the subsequent versions of Cadral.

Several obvious advantages are brought by the computerized approach; we briefly recall the three outputs of Cadral most wanted by our administrative partner. The first is to deliver the benefits more quickly. Also the electronic

processing will guarantee the full equitability of the process and make the institution more transparent to the public. Last, in the future, Cadral will be used to enable the beneficiaries and the Fund to perform individual or macro simulations of the payments.

As for efficiency, most of the time automation will perform a full processing of the files. However, in some more complex cases, according to the particularity of the socio-economic situation of the beneficiary, Cadral will isolate the decisions that can be made automatically and transmit the file for manual handling for those requiring the intervention of a human operator. Nowadays, the basic tasks such as checking files against database entries or applying the right procedure are the source of overwork for the Fund, whereas less than 5% of the cases require a particularly subtle interpretation or care by a human operator.

The computerized approach is, moreover, the only means to ensure the rigorous handling of the applications. When processing a file, the inference engine first validates the coherence of the rules applied in the particular case and then ensures that an identical and consequently equitable answer is given whenever possible. Otherwise, the decisions left to the discretionary intervention of a human operator are precisely circumscribed. The trace of the reasoning, along with references to the juridical texts, can be used to justify the decisions with respect to the law and provide protection against possible complaints from the claimant against the administration.

In the next version, when a module is added to process applications including the exact calculation of the payment, Cadral will enable the beneficiaries as well as the board of the Fund to perform simulations by entering customized input data. In regard to the beneficiaries, the Benefits National Family Benefits Fund is building an interactive website to allow access to personal files and the ability to fill, view, and manage demands online. A simulation function will enable beneficiaries to check the acceptability of their demand, learn the amount they are to receive, and prevent them from sending incomplete applications. On the other side, macro-simulation will ease planning tasks within the Fund. They will be able to predict basic figures according to demographic statistics and to appreciate in advance the impact of changes in the legislation for family benefits.

The remainder of the paper is as follows. Section 2 describes the main ideas for modelling the knowledge. Section 3 gives implementation details for building the expert system. Section 4 presents the graphical interface. Section 5 further discusses the organization of the knowledge bases and the integration of Cadral. Finally, concluding remarks deal with the actual state and the continuation of the works.

2 DATA SPLITTING

The starting point for building Cadral is the Luxembourg national legislation regarding family benefits. It consists however not of a monolithic structure but of a constellation of national laws, supra-national decisions, and international regulations.

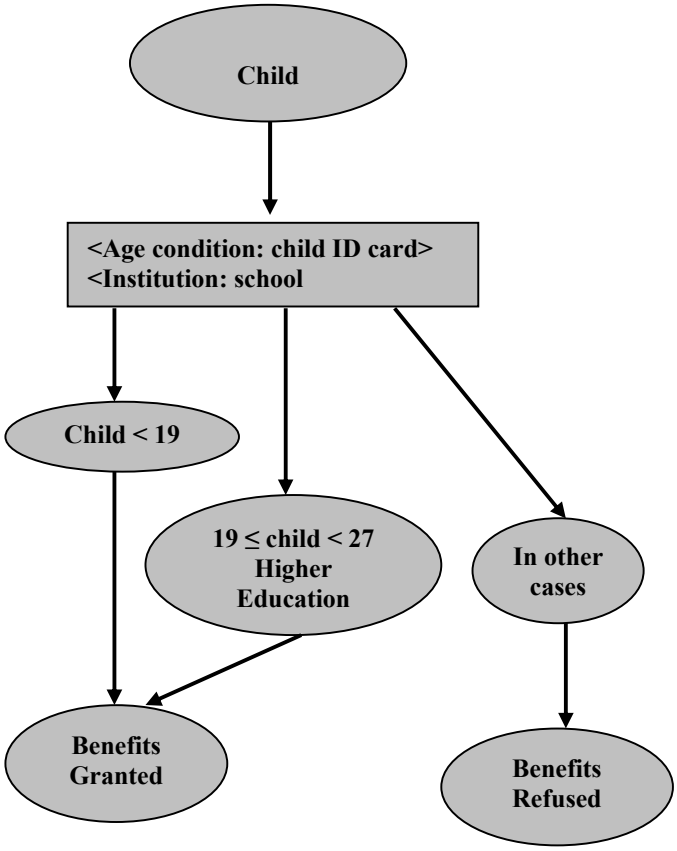
Modeling juridical texts into logical formalism has been proposed in order to directly apply inference mechanisms to the context of the law (O'Callaghan et al., 2003). Several formalisms and resolution algorithms are available and well-tried, such as the first-order and backward-chaining architecture of Prolog, the modal or deontic logics (Eijlander, 1993). However, the heterogeneous nature of the texts (complex formalism of general regulations vs. precision of national administrative code) and the numerous implicit definitions that are used (e.g. certificate validity, pre-natal, post-natal allowance...) make the translation into a formal and single-view computer language a very long and detailed challenge.

Nevertheless, in the daily work of the Fund, legislation is not constantly referred to. Operators have in their mind the condensed information that is relevant for most of the cases and refer to the law only when necessitated by some subtlety of a treatment. Accordingly, instead of whole mathematical modeling of the law, we decided to concentrate on the explicit model of the mental procedures that govern the processing of the applications and the relations between these procedures and the legislation. The operating knowledge of Cadral is therefore a procedural modeling of the legal texts.

An example of a procedure might be: "If a child is going to school and is younger than 19, the claimant is entitled to receive the child benefit." Such a procedure is modeled as a multi-valued acyclic n-ary graph, with nodes representing a factual state (e.g. child going to school) used as a condition and the edges denoting the necessary steps (e.g. showing a school certificate) to enter the state.

Moreover, we ensure that a state is always unique in the graph (no two nodes have the same label) though it is fully possible to go to the same state by different paths. Let us consider the text of one law article. This text consists of several alineas (i.e. paragraphs), each associated with a different state. The modeling of the full law article is therefore a procedural graph, where all the states are distinct and such that we can define an isomorphism, which associates every node in the graph with an alinea. When the procedures are translated into a collection of inference rules for an expert system, such isomorphism is used concurrently with the trace of the inference engine in order to memorize the legal references made during the reasoning performed according to the procedures.

Figure 1 illustrates the graph-based modeling of the legal texts. The root node is associated with alinea 1 of the (simplified) text and the two following nodes (in going down the figure) with alineas 2 and 3 respectively. Two additional concluding leaves for accepting or rejecting the applications show the final status.



Article: Simplified Process of Child Benefit
$1. A child benefit is granted to support the education of every child.
$2. The benefit is granted if the child is younger then 19.
$3. The benefit is granted if the child is between 19 and 27 and is enrolled in a university curriculum.

Figure 1. Graph modeling of a simplified law article

3 PROGRAMMING CADRAL

Our graph-procedural modeling of the law has guided our choice relative to the technology used to model the procedures and infer with them. The reasoning on a law article consists, indeed, in proceeding from one state to another according to the procedures and the conditions (labelling the edges) that are satisfied. All the paths and all the states must be effectively checked in order to ensure that no case provided by the law for a given application is left out. This consideration orientates our choice towards a rule-based inference system proceeding by parallel forward-chaining (contrary to the Prolog like scheme, which proceeds by backward resolution in a depth-first manner).

The resolution kernel of Cadral is developed on top of the Soar IA architecture. Soar is a general-purpose rule language whose inference engine is based on the Rete algorithm (Forgy, 1982) and works in a forward-tracking manner. The rules (also called productions) are "if A then B" statements whose meaning is: "if the situation A is satisfied, then create (or produce) the situation B." Soar's purpose is to propose a Unified Theory of the Cognition (Laird et al, 1987), and the system is backed as the most suitable language for intelligent agents programming (Fehling, 1993). One advantage of Soar is that it can communicate in many ways (through sockets or procedural routines) and allows users to place in the rules requests for information concerning the allowance demand or the claimant data.

However, because of its general purpose, writing rules in the Soar language can soon become intricate, relative to the syntax itself as well as to the management of the inference algorithm (Laird, 1999). Moreover, when slightly modifying the rule base, the behavior of the whole base can change drastically in a way that is not intuitive. For this reason, the heart of Cadral is an intermediate language with simplified syntax that is compiled into true Soar formalism. This upper-level layer is designed to provide the user with useful or necessary subroutines in view of the specialized topic of the program. All the subroutines are documented with a stable and proven behaviour, corresponding to the expected modeling of the procedures. In particular, the intermediate language implements the required controls on the Soar resolution engine in order to manage the notion of state used within the procedures and on-the-fly communications.

This high-level language also makes it possible to run a graphical editor in which the procedures are directly modeled in the shape of graphs. This editor is based on the tool Jgraphpad (cf. the JGraphPad tool), developed on Jgraph, the Java core graph visualization library that features a powerful array of graphical functionality and easily notably allows users to draw graphs and export them into miscellaneous formats. In Cadral, graphs drawn with JGraphPad are exported into GXL (cf. the Graph eXchange Language), written in the XML standard.

We now review Cadral on the whole from the implementation point of view. All the modules of Cadral run on a Windows station, though they are portable to Unix or Mac OS. For editing the rule base, the user draws the graphs of the procedures within the Jgraphpad editor and saves them in GXL files. The compilation from GXL to the intermediate language and from this to Soar rules files is accomplished through two compilers written in C++ with the Lex & Yacc packages (cf. the Lex & Yacc packages).

Thus, running Cadral consists of starting the Soar engine after launching the Soar rules generated files. Several interfaces are provided for running Soar, among them are the TSI library, which incorporates Soar into a Tcl/Tk (cf. the Tcl/Tk Scrip Language) application, and SGIO (cf. the Soar General Input Output extension), C++ libraries to be linked with a C++ executable. We currently run Cadral on both interfaces in executable test modules. All communications (concerning the data concerning the allowance demands, the claimant records in the databases, or the result of the resolution) are simulated through standard input / output, and the trace of the resolution is generated in a text file.

4 KNOWLEDGE EDITORS

Let us recall the modeling discussed in Section 2. When the administrative knowledge is translated into a collection of inference rules for an expert system, the modeling of a full piece of the law (a collection of articles) produces a

procedural graph, showing both the inter-dependence of the rules and the references to the legal texts. For editing the knowledge, Cadral proposes on the one hand a Rule Editor, in which the user calls the routines of the programming language of Cadral implemented in top of Soar and analytically translates the administrative procedures into detailed rules. These rules are then directly passed to the Soar-based engine. Figure 2 shows a shot of the Editor.

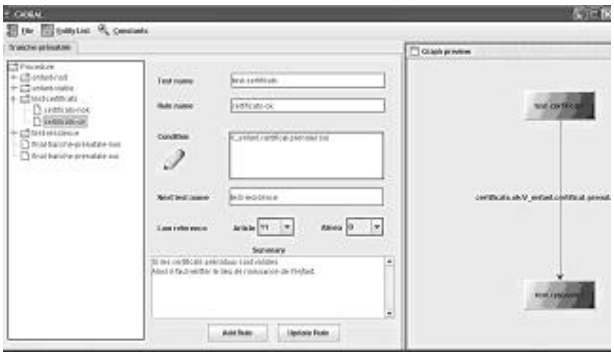


Figure 2. Cadral Rule Editor

On the other hand, for visualizing as a whole the body of laws formed by a set of rules, it is useful to have an intuitive and synthetic overview of the knowledge used by the resolution engine. Cadral therefore offers to compile together a set of rules (e.g. all the rules governing the attribution of a given allowance) and to display a graphical view of the corresponding piece of legislation. However, a good understanding of the relational model depends on the quality of the graph drawing. In offering the most readable display of the subjacent structure, the main problems are the layering of the graphical hierarchy and the minimization of edge crossings. The rest of this section presents the approach followed to address these problems.

The graphs we obtain here are more precisely *Directed Acyclic Graphs* (or digraphs), which can be seen as a generalization of trees: they contains no cycles, and their nodes can be assigned to layers so that all (directed) edges point in the same direction, usually downward. For further details, refer to a general presentation such as Di Battista et al. (1999). Among all well-known methods for hierarchical layout design, the most popular is presented in Sugiyama et al. (1981). We have implemented a modification of this approach adapted to the requirement of our legal graphs. Let us start with a short description of the application of this method to our legal graphs.

To begin with, we compute for each node its predecessors and successors lists. We then calculate the rank of every vertex, which helps us to assign levels to the vertices. We sort the nodes by level. With this step we get a hierarchical representation of the graph. We count the number of edge crossings. If this number is 0, then the hierarchical graph is planar, and we have reached a solution. If not, we build a proper hierarchy; we reverse and split edges until each edge is directed downwards and connects two nodes on neighbouring layers. The newly introduced nodes are called dummy nodes. We then reorder the nodes on each layer using the barycenter heuristic (Radwan et al., 2002).

During this implementation of the Sugiyama algorithm, we encountered 2 problems and refined the method as follows.

(1) If two (or more) vertices in the same layer have an equal crossing number, in what order should we place the vertices of this layer? The quality of the produced drawing without edge crossing depends indeed on the order in which vertices are placed on each layer. We have two options: either we leave the order of these vertices unchanged or we reverse them. By testing both cases, we determined that the number of edges crossing differs in the two considered cases. The number of edge crossing is smaller when we reverse the vertices with same values.

(2) The other problem appears when we run the algorithm on planar graphs, i.e. graphs that are already without any edge crossing. It is possible to obtain a result with at least one edge crossing after running the algorithm. To overcome this problem, we first assign levels to the graph, and we then count the number of edge crossings. If there is none, we stop the algorithm and take the layered graph as a result. Moreover, to be sure that we get at the next step, a crossing number inferior to the previous one, while running the algorithm, we compare the number of edge crossings after every iteration and end the process when it increases.

Figure 3 shows an example of the graphic visualization obtained for a piece of legislation after the application of the complete method described in this section.

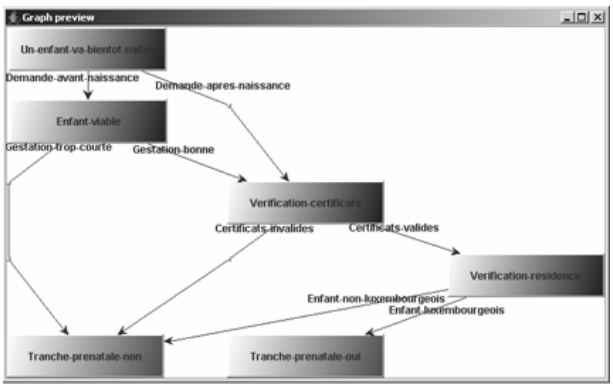


Figure 3. Cadral Graphical Viewer

5 OPERATIONAL WORKFLOW

Having shown how to edit and run the decision system, we complete the overview with some considerations about the data management and integration of the software tool into an operational infrastructure.

All Cadral's knowledge is organized into three kinds of data: procedural data (processing rules), reference data (law), and application data (individual cases). The procedural knowledge manages metadata to retrieve and interpret the other two datasets. The choice of data tables as the primary format to store the procedural information, which is then translated into engine rules for edition or processing or graphs for visualisation, is justified to ease cooperation between the Fund's operators for running and updating Cadral.

In order to be integrated into the working infrastructure of the Fund, Cadral must fulfill basic requirements in terms of interface, usage, and usability of the response. First, Cadral is designed to receive as inputs all the data that can be contained in the application files. An interface is thus maintained with a customizable list of these input data, the list being edited when changes occur in the law concerning benefits requirements. For now, the data are entered manually, but interfaces will be developed for communicating with an OCR-enabled processing module for the paper documents and files and with the Website for applications filled out online.

Still regarding the interface, Cadral must be able to retrieve all the parameters that are to be taken into account for the processing of the application, including checking for the presence of required certificates. The data is here dispatched into several databases tables, and special care must be taken to recombine all the information correctly. For example, a cross-border worker is entitled to receive child allowance, but the amount paid by the Fund will have to be deducted from the amount of the allowance of the same kind possibly received from the country of residence.

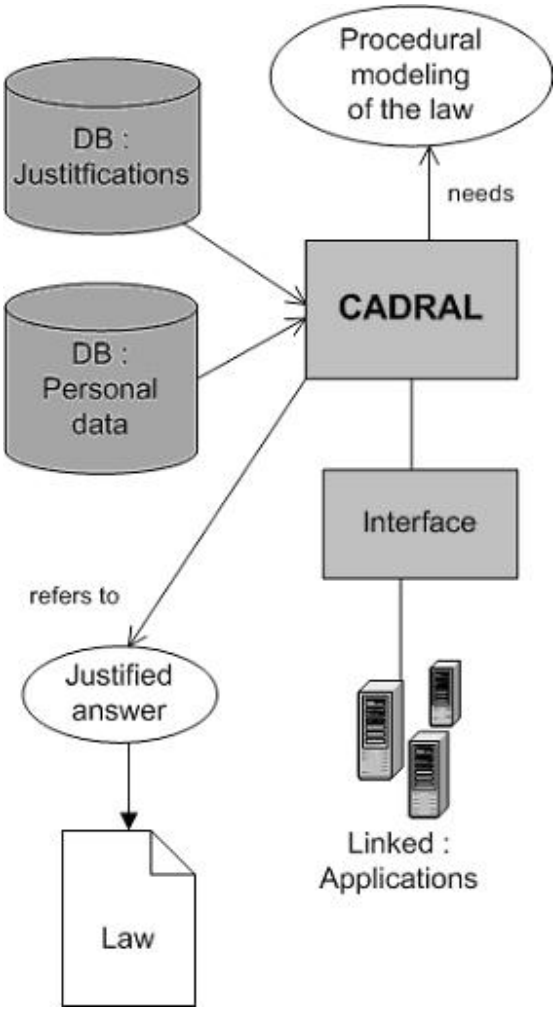


Figure 4. Integration of Cadral

Concerning its practical use, Cadral is characterized by its flexibility. The knowledge it contains to process all applications is a set of legal texts, and as such, they are subject to frequent evolution, dismissal, or addition. In a similar way, the situation for individual data of the beneficiaries can be either modified because of change in circumstances life or adjusted according to new juridical status. All these reasoning and demographic data must be stored so as to be easily updated by a non-specialist operator of the administration.

Lastly, dealing with juridical issues, the answer provided by Cadral must be deterministic and justified. Due to either intrinsic incoherence or errors in the translation into computerized procedures, the juridical knowledge of Cadral can be erroneous or incomplete. In the traditional manual handling, the incoherence is detected and solved by the operator or the hierarchy. With an automatic processing, the situation could lead to unwanted responses. It is therefore necessary to proceed with a set of rules and an inference mechanism that guarantee the exactness of the knowledge and the uniqueness of the returned decision. Moreover, the reasoning process provided by Soar is in every case traceable. In addition, links are maintained from the rules to a database of legal references, in order to exhibit a juridical justification of the decision.

Figure 4 illustrates how the resolution engine, or kernel of Cadral, interacts with all peripheral components.

6 CONCLUSIONS

A decision system for automatically processing administrative applications has been presented. This customizable system will be used for the attribution of family benefits. The raw knowledge of Cadral is a comprehensive set of juridical texts. However, though references to the law are necessary in some intricate cases and for justifying the results, direct interpretation of the texts can be avoided in most cases. This article has discussed the feasibility of a method based on the modeling of the legal framework into simpler procedures. This approach is completed by a graphical visualisation of the juridical knowledge and an operational integration in a computerized infrastructure. Future works include primarily the two following axes: first, a macro-editing package of the knowledge working from the graphical representation in order to ease the update of Cadral without rewriting separate rules and, second, a calculation module, which will allow a fully automated handling of the applications for benefits and will be used to develop a micro (for a single user) and a macro (for a population) simulation tools.

7 ACKNOWLEDGMENTS

The authors wish to thank Michel Neyens and Claude Nicolas, from the *Caisse Nationale des Prestations Familiales*, Luxembourg, for their continuous cooperation and support.

8 REFERENCES

Di Battista, G., Eades, P., Tamassia, R., & Tollis, I.G. (1999) *Graph Drawing: Algorithms for the Visualisation of Graphs*. Prentice Hall.

Eijlander, P. (1993) The Possibilities and Limitations of Using Intelligent Tools for Drafting Legislation. *Legal Knowledge and Information System, Proc. of Jurix Conference* (pp 5-10). Amsterdam, the Netherlands.

Fehling, M.R. (1999) Unified Theories of Cognition: Modeling Cognitive Competences, *Artificial Intelligence, Vol. 59* (1-2), pp 295-328.

Forgy, C.L. (1982) Rete: a Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem. *Artificial Intelligence, Vol. 19*, pp 17-37.

Gordon T. (1994) From Jhering to Alexy - Using Artificial Intelligence Models in Jurisprudence. *Legal Knowledge and Information System, Proc. of Jurix Conference* (pp 19-32). Amsterdam, the Netherlands.

Groothuis M., & Svennsson J. (2000) Expert System Support and Juridical Quality, *Legal Knowledge and Information Systems, Proc. of Jurix Conference* (pp 1-10). Amsterdam, the Netherlands.

The Graph eXchange Language tool. Retrieved November 22, 2006 from the World Wide Web: http://www.gupro.de/GXL.

The JGraphPad tool. Retrieved November 22, 2006 from the World Wide Web: http://www.jgraph.com/jgraphpad.html.

Laird, J.E., Newell, A., & Rosenbloom, P.S. (1987) Soar: an architecture for general intelligence. *Artificial Intelligence, Vol. 33* (1), pp 1-64.

Laird, J.E. (1999) The Soar User's Manual, version 8.2. Technical Report , the University of Michigan.

Lex & Yacc Packages. Retrieved November 22, 2006 from the World Wide Web: http://dinosaur.compilertools.net.

O'Callaghan, T.A., Popple, J., & McCreath, E. (2003) *Building and Testing the Shyster-Mycin Hybrid Legal Expert System.* Technical Report, Faculty of Engineering and Information Technology, Australia National University, Camberra. Retrieved November 22, 2006 from the World Wide Web: http://cs.anu.edu.au/software/shyster.

Radwan, A., Girgis, M., & Ghanem, A. (2002) A Study of Barycentre Algorithm for Hierarchical Graph Crossing Minimization. *International Journal on Intelligent Cooperative Information Systems, Vol. 2* (2), pp 13-22.

Soar General Input Output extension. Retrieved November 22, 2006 from the World Wide Web: http://sourceforge.net/projects/sgio.

The Softlaw / RuleBurst Expert-System. Retrieved November 22, 2006 from the World Wide Web: http://www.softlaw.com.au.

Sugiyama, K., Tagawa, S., & Toda, M. (1981) Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man and Cybernetics, SMC-11* (2), pp 109-125.

The Tcl/Tk Script Language. Retrieved November 22, 2006 from the World Wide Web: http://www.scriptics.com.