Logic-based Network Configuration and Management

Salvador Abreu and Joaquim Godinho
Universidade de Évora

spa@di.uevora.pt
jjg@sc.uevora.pt
Largo dos Colegiais, 2, 7000 Évora, PORTUGAL
http://www.uevora.pt/

Keywords: *Logic in Databases, Deductive Databases, Knowledge Bases, Network Management, Network Configuration.*

Abstract: *Universidade de Évora's Integrated Information System (SIIUE) aims at representing the entire universe of concepts useful for the man-agement and day-to-day operation of the Organization. It relies on a first-order logic Description Language to define the schema, represent data and computed relations. This representation is used to generate an object-relational database that is actually used to interface to the system. In this article we describe an application of this framework to issues of network and services planning, configuration and monitoring.*

Preface

Universidade de Évora has developed and is currently deploying a general-purpose information system [1], geared initially towards interacting with faculty members and staff but which aims to gradually fulfill the whole of the organization's internal information needs.

The approach chosen for SIIUE [1] can be summed up in the following design principles:

1. Use a Logic Programming basis for defining both structure and data. The LP dialect used for this purpose is known as the Information System COnstrution language (ISCO), and is partially described in [1].
2. Structure the information in an object-oriented, multiple inheritance scheme. This approach allows for a good re-use of information and is quite intuitive in specifying complex class hierarchies as is inevitable in the case of SIIUE.
3. Specify constraints between data in the Logic Description language. Especially when associated with a form of Constraint Programming (such as constraints over finite domains), Logic Programming can be a very effective tool to specify a problem declaratively.
4. Interact with the system through the Database management system. Considering that most of the information fits the relational model, it seems reasonable to opt for this approach towards interfacing to the system. Moreover, by choosing this path we can make use of an existing efficient implementation for a DBMS.

This approach has already proven to be quite adequate for solving a problem that is underspecified and of arguably large scope, while having to endure severe restrictions on the human resources involved in its design and implementation. It allowed us to go from an initially ill specified requirement to a functional, although partial, implementation in under one year with very limited human resources (total development effort was under 2 MY).

The purpose of this article is to describe our experience with using the SIIUE information system framework to deal with network administration tasks.

The article is structured as follows: in the next section we briefly describe the specification source language used for programming SIIUE (ISCO), sec-tion 3 discusses the application of SIIUE to the problem of Network Config-uration, in section 4 we illustrate the potential of SIIUE for the automatic generation of configuration files for certain network support systems. In section 5 we take a brief look at the issue of failure diagnosis under SIIUE and conclude in section 6 with directions for future work.

ISCO: The Logic Description Language

Before delving into the issue of how to link an RDBMS with a Prolog system, a brief description of the Logic Description Language, ISCO, is in order. To describe the system's classes and their behavior, a modeling language could be used. Instead of resorting to an existing language such as UML [2], we decided to implement a new such language, based on a first-order logic.

We feel that this approach provides more flexibility at the meta-level than would be feasible with other tools. Presently the only form of programming is textual but there is on-going work geared towards developing a visual programming language to satisfy the same design goals and sharing the same semantics (see section 6 for more details.)

The approach chosen involves a stylized logic program where the relation between the entities is described, in an object-oriented fashion. The entity taxonomy used for the information system is introduced as a type system with multiple inheritance.

Rather than formally describing the syntax and semantics of ISCO, let's just say it can be thought of as Prolog with a few extensions.[1]

From the logic description (ISCO) program, several passes are made to generate different types of derived components, of which the main ones are:

1. An SQL file with all the instructions necessary to construct an object-relational database that corresponds to the information described in the ISCO source. This covers both the schema and the data.
2. A set of PHP3 [5] class definitions, to be used for web page construction. PHP3 (PHP Hypertext Processor) is an embedded scripting language for web servers, which is actively being used to develop the web site of Universidade de Évora.
3. A Java package, basically a collection of Java classes which mirror the ISCO hierarchy. These are fitted with constructors that in-

1 These include the modularity mechanism known as Contextual Logic Programming [4] and a very-high-level ODBC interface.

terface to the SQL database previously generated using JDBC. The goal is to simplify the construction of applications that make use of SIIUE.
4. Documents describing the class hierarchy, with comments on every class' usage, on the attributes that are used as external keys, etc. This is especially useful in describing the actual system to non-technical persons. The format of the documentation can be either of HTML or LaTeX, the latter being used to produce high-quality cross-referenced PDF files.

The dialect of Prolog being used is enhanced with a modularity extension, we are implementing a mechanism similar to that of Contextual Logic Programming [4], as it provides an elegant and effective framework in which to structure computations while retaining the simplicity of Prolog. In particular contextual definitions will allow for a much cleaner way of defining higher-level procedures.

The Prolog implementation being used (GNU Prolog [3]), being the successor of CLP(FD) also provides the constraint logic programming paradigm, which is a very useful extension to the traditional Prolog programming style in that it allows for problems to be solved by providing a-priori search-space pruning, through the constraint propagation mechanism.

Using SIIUE in Network Configuration

SIIUE is a general-purpose information system, with deductive facilities and several interfaces. It has been successfully applied to issues of the institution's academic activity and is now being applied to the problems related to network planning, configuration and management.

The final purpose of SIIUE, in its network component, is mostly to allow the generation of automatic configurations of several equipments and also the diagnosis and preventive maintenance of the whole network. Nevertheless that isn't possible without a good model of the network, in its physical and logical aspects, which presumes a representation of the physical infrastructure, of the installed equipments, of their connections and locations, of their configurations and the available services.

By simultaneously modeling the physical infrastructure and the equipment configuration, it is possible to immediately detect inconsistencies between these two levels the physical and the logical—for instance, a computer physically connected to a router, where both network configurations aren't compatible or are simply inconsistent. These are the two levels that are represented in ISCO and summarized in the next two points.

One of the advantages of this approach is the possibility to easily - and as the situations are identified - specialize the existing representation or introduce intermediate levels in order to bring out common features of different hierarchy nodes. Such a characteristic seems fundamental, as the model may be conceived in attendance to the existing equipment, but allowing the gradual introduction of new technological solutions. The same principle may be applied to the network configuration—for example, the evolution to IPv6—which must be modeled in such a flexible way as to allow the evolution of both the standards and the equipment.

Locations and devices

As previously described, a network's logical representation focusing on issues such as routing and firewall configurations, DNS and DHCP needs a minimum physical representation of the installed equipment, namely regarding network interfaces, and their interconnection.

Nevertheless, regarding network analysis and diagnostic this physical representation must be more accurate in order to cover several other aspects such as physical location, paths and hardware characteristics.

The model being deployed is centered around two top-level classes, associated with the ***network***and ***location***concepts. Two other entities complement the integration of the two levels: ***connection-Point***and ***connection***, the latter representing a pair of instances of the former.

With the proposed model we aim to achieve three purposes:

- Represent all the physical infrastructure — from terminal equipment to racks and network equipment — in the most complete and scalable way possible, allowing for increasing levels of detail.
- Characterize all the existing interconnections, whether we intend to represent effective connections of active equipment or only the installed cabling infrastructure.
- Associate a location to any network infrastructure or equipment, deploying a representation sufficient to immediately detect static (ie. pertaining to the definition itself, not to an anomaly in operation) definition inconsistencies such as: equipment X installed in room A is connected to jack Z which is installed in room B.

Automatic Configuration

The issue of being able to guarantee the completeness and correctness of the configurations for the various network-based services is an important one for the structure with that responsibility within the Organization. Having the information from which these services are configured described in a Logic Programming language allows us to comfortably perform all sorts of verifications on the data, thereby providing us with the assurance that it is correct.

Several services are amenable to being automatically configured by a general-purpose information system with the flexibility and characteristics of SIIUE, these include DNS, DHCP and user-related services such as accounts, mailboxes and group aliases.

LDAP directories

LDAP is now widely used as a directory service, mostly for the purpose of implementing user groups and as the basis for authentication services but its potential goes beyond these uses. This is basically due to LDAP's simple and generic structuring mechanism.

LDAP directories, when used to describe large sites, are error-prone in that the features that make LDAP successful - its simplicity and extensibility - are also its weakness, because so much is left to interpret by the clients. This kind of situation requires that behavior specified by rules be specially coded within the directory and "programmed around" by client applications.

User and Group Profiles

Since SIIUE already has a representation for the entire organic structure of the University, including that pertaining to people —faculty, staff and students— it makes sense to use this information to automate the maintenance of user accounts and mailboxes as well as group aliases.

People are represented as individuals, through the class indivìduo, in-dependently from the nature of their relation with the Organization.

The relationship between people and groups (in the sense of structure within the Organization") is reflected by a "member" relation (membro in the portuguese-flavoured actual implementation.) This relation is represented by a class from which a few others are derived, in order to designate special membership situations (such as the president of a group.)

5.Another hierarchy rooted on class contacto is used to represent all manners of contact information for entities within the Organization; this includes e-mail addresses, phone numbers, web pages, room numbers, etc. for individuals or groups. Note that a part of this information is used to determine the names of the aliases while another part is used to determine the definitions.

Yet another class hierarchy deals with who teaches what, to whom and when. This hierarchy is rooted on class lecciona ("teaches".) This sort of information is used to automate the construction of a very rich set of functional e-mail aliases, which include for example:

- The faculty members currently teaching some course.
- The faculty members currently teaching some course to a specific set of students.
- The members of the Scientific Council.
- The faculty members who taught a given course in a specified semester.
- The professor responsible for a given course.
- All users of all machines owned by the History Department.
- The Physics Department's secretariat.
- etc.

Some of the functionality of these aliases is already accounted for by standards such as LDAP and practical tools that use it. However, our option to define our own scheme is backed by the ease with which it was possible to extract the information from SIIUE (the tool is the same), together with the fact that, using this approach, we are essentially centralizing the management of all user information.

This approach to defining groups also allows us to generate web pages with the structure of the Organization, in which we may present various informations related do the group, including its composition which is also derived from the member relation.

Using SIIUE to define users and groups allows us to relegate the responsibility of updating this information to other entities within the Organization, such as the Administrative Services. This way, many of the Computing Services' administrative duties can be reduced because tasks such as creating user accounts and managing most lists become fully automated.

DNS and DHCP

Two of the most critical basic network services running on the Organization's network are DNS and DHCP. For several years, the configuration files for both of these services were already being generated from a single source, which was processed via a collection of M4 macros and AWK scripts. This approach was reasonably functional but started to require a lot of attention as the size of the network grew: it became evident that hand-editing configuration files does not scale well.

By incorporating the required information into SIIUE, it is feasible to automate the construction of all configuration files related to these services while retaining the declarative nature of the specification. For example, we may:

- Specify that all "inner" networks have a gateway at the highest address, just below the broadcast address.
- Indicate that all machines owned by the Department of Mathematics have an MX record pointing to a specific machine.
- In recognition of the fact that every network associated with a separate physical building has a similar topology, a deductive datum (ie. a data line with a nonempty Prolog body) may be written to algorithmically specify some of the characteristics of the network: its local DNS and DHCP server, the local HTTP proxy, etc.

Firewall Definitions

Most routers in Universidade de Évora's network also act as firewalls, some-times just doing transparent proxying of HTTP requests but frequently actually performing some filtering or IP masquerading. The task of configuring the firewall hosts can be quite complex: as an example one such system running on our network has over 800 distinct filtering rules.

Maintaining these rules coordinated with the evolution of the network can be delicate.

Another issue that can de-stabilize the operation of these systems is the firewall's hardware and software configuration itself. In the case of Linux routers, the exact way in which the firewall rules are specified has changed twice: from Linux version 2.0 to version 2.2 the "flat" rulespace has been replaced by the "IP Chains" scheme. This is about to be replaced for Linux 2.4 with yet another mechanism. In certain situations the 7.router may exceptionally not even be a Linux box but a commercial router proper, such as a Cisco or Lucent device.

SIIUE is being used to generate the firewall rules for these equipments, thereby guaranteeing the correctness of the rules currently in use, at least from the point of view of being up-to-date.

Network Diagnostics

One of the major issues involved in network management is the recovery from anomalous situations. The availability of the entire network and services topology as a Logic Program or deductive database allows for the adoption of advanced diagnostic and troubleshooting strategies.

We already had experimented along the lines of dependency-based net-work monitoring (see [7] for such an extension to the mon [8] network moni-toring system). The experience using mon was interesting and indicated that the approach of having a monitoring system continuously watching various services on several hosts can indeed speed up fault detection, sometimes more so than SNMP-based approaches.

The mon system, being programmed in Perl, lacks the sophistication in programming complex diagnostics that can easily be achieved in a Logic Pro-gramming language. Moreover, when using mon to watch over a network and services topology which is continuously evolving, the timely maintenance of configuration files quickly becomes a tedious and error-prone endeavor.

We are currently working on a service monitoring system which directly uses SIIUE. In particular, this system has the complete knowledge of the network topology and also knows about the wiring as well as other related low-level issues. The availability of a full model of the network allows us to apply logic program debugging-like techniques and can hopefully lead to very informative diagnostics, so that we can envision the automatic generation of detailed procedures for failure recovery for use by the field personnel, customized for each individual failure.

Conclusions and Future Work

SIIUE [1] has already proven useful by permitting a number of practical applications to be developed on relatively short notice and with restricted human resources. The combination of an RDBMS with a Logic Programming core was shown to be useful in that it allows for elaborate computations to be performed on the information contained in the database.

The widening of SIIUE's scope to include networking topology, services and user information allows for the relatively easy automation of several tasks which were manually performed.

While this article doesn't present a completed system, it does outline a specific line of work which is currently being pursued at Universidade de Évora which builds upon the experience gained while developing the SIIUE framework and its first applications. Various other on-going research and development projects at Universidade de Évora are related to this work, namely a Natural Language interface, a Visual Programming language to complement ISCO and a Geographic Information System to deal with spatial issues.

Acknowledgements

The authors would like to thank Luìs Quintano and Mário Filipe for various discussions pertaining to the work described herein. An acknowledgement is also due to the administration of Universidade de Évora for its support in the development of the SIIUE project, in which the present work is integrated.

References

[1] Salvador Pinto Abreu. A Logic-based Information System. In Pontelli and Santos-Costa [6].
[2] Grady Booch, Jim Rumbaugh, and Ivar Jacobson. Unified Modeling Language User Guide. Addison Wesley, December 1997. ISBN: 0-201-57168-4. See http://www.awl.-com/cp/uml/uml.html.
[3] Daniel Diaz and Philippe Codognet. GNU Prolog: Beyond Compiling to C. In Pontelli and Santos-Costa [6].
[4] Luìs Monteiro and António Porto. A Language for Contextual Logic Programming. In K.R. Apt, J.W. de Bakker, and J.J.M.M. Rutten, editors, Logic Programming Languages: Constraints, Functions and Objects, pages 115-147. MIT Press, 1993.
[5] The PHP Hypertext Processor. http://www.php.net/.
[6] Enrico Pontelli and Vitor Santos-Costa, editors. 2ndInternational Workshop on Practical Aspects of Declarative Languages (PADL'2000), number 1753 in Lecture Notes in Computer Science, Boston, MA, USA, January 2000. Springer-Verlag.
[7] Tiago Severina and Salvador Pinto Abreu. Monitorização inteligente de uma rede de computadores. In CRC'98 - I Conferência de Redes de Computadores, Universidade de Coimbra, 1998. (in Portuguese).
[8] Jim Trocki. Mon - Service Monitoring Daemon, 1998. See URL http://www.kernel.org/software/mon/.