Rethinking Reusable Resources

David M. de Matos*, Ricardo Ribeiro†, Nuno J. Mamede*

L^2F – Spoken Language Systems Laboratory – INESC ID Lisboa
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{david.matos,ricardo.ribeiro,nuno.mamede}@l2f.inesc-id.pt
*IST – Instituto Superior Tecnico
†ISCTE – Instituto Superior de Ciências do Trabalho e da Empresa

Abstract

We address the common and recurring problem of data reuse, focusing on the following topics: (i) the current state of affairs (in particular, problems with data); (ii) requirements for change; (iii) the proposed solution (its problems and advantages, as well as related work in this area), including the canonical-, I/O-, and data transformation models; (iv) maintenance issues; (v) implementation and deployment aspects; (vi) conclusions and future directions, including results from work done so far and aspects that merit future work.

1. Introduction

Although much effort has already been devoted to data-related problems, the fact is that existing resources are very difficult to reuse, especially if that reuse does not take the original philosophy of the resources strictly into account: resources otherwise useful become closed to users using some other model. For example, for someone working in morphology: while various data sets are available, e.g. SMorph (A¤-Mokhtar, 1998), PAROLE (PAROLE, 1998), and Palavroso (Medeiros, 1995), and despite the fact that all have similar description capabilities, still they present the user with serious interoperability issues.

In general, the problems that afflict data sets and their reusability refer to miscellaneous incompatibilities: (i) at the description level, i.e., how existing objects are described (the problem manifests itself frequently as tag incompatibility); (ii) at the level of what is described: some descriptions may describe objects missing from other descriptions; (iii) basic incompatibilities: format/representation: XML (W3C, 2001a) vs. tabular data; and (iv) expressiveness: e.g. "United States of America" as a single entity vs. composition of separate entities.

In the following we discuss the topics of the abstract.

2. Requirements for change

To address the incompatibility issues presented above, we identified a set of requirements: (i) preserving existing information (this is in an "at least" sense); (ii) allowing data reuse among different applications; (iii) allowing data to be imported/exported across existing formats (existing applications keep working, but they now use potentially better/richer data); and (iv) easy maintenance and documentation of changes.

These requirements are ideal in the sense that they may be addressed in various ways. A given solution for one of them may be optimal, but not suit all of them: some may be better than others and some solutions may give rise to new problems. Our proposal seeks to find a balance, minimizing the negative aspects while meeting the requirements.

3. Dynamic repository

We propose a canonical model for storing/manipulating data, and a dynamic maintenance model for keeping the data model synchronized with new data developments. Even though a canonical model has its own set of problems, it presents distinct advantages: it is easier to maintain and document a single format than multiple different ones; the effort dedicated to maintenance tasks is concentrated, possibly further improving them; it allows for deeper understanding of data, which in turn facilitates reuse (the reverse would require a user to understand multiple models).

3.1. Related work

From literature, we find related work in this area. What seems to be different is that some of the models we could call canonical did not address the issue of remaining current nor how to adapt to new developments. Illustrative models include Multilex (Paprotte and Schumacher, 1993), Genelex (Antoni-Lay et al., 1994), PAROLE/SIMPLE (PAROLE, 1998; SIMPLE, 2000), and the EAGLES recommendations.

The event that triggered most of these reusability efforts was the workshop *Automating the Lexicon: Research and Practice in a Multilingual Environment* (Walker et al., 1995) that took place in 1986. Then, several projects were launched that addressed this issue. The EUROTRA-7 Study (EUROTRA-7, 1991) was concerned with accessing the feasibility of designing large scale reusable lexical and terminological resources. The main contributions of this study were an initial specification of a model for a reusable lexicon and several recommendations regarding the importance of standardization. Another important project was Multilex. This project aimed at providing specifications of standards for multilingual lexicons. The result was a preliminary design for a reusable multilingual lexicon, that continued the work previously started during EUROTRA-7. The Genelex project had as main objective the development of a generic, application-independent model of lexicon. This model is commonly described as "theory welcoming" since it tries to accommodate data from competing theories. The Genelex model was adopted (and adapted) in projects like PAROLE/SIMPLE which aimed at the development of the core of a set of natural language resources for the European Community languages.

4. Conceptualization

This section presents design concepts without regard for implementation details.

Any sufficiently expressive high-level modeling language should be suitable for describing our models: one such is UML (Booch et al., 1999; OMG, n.d.); another would be XML Schema Definitions (XSD) (W3C, 2001b). Also to consider is their being amenable to automated processing, and their usefulness as model documentation languages (both UML and XSD fulfill these criteria: XSD, directly; UML, partly, via XMI (OMG, 2002)). We chose UML for its relative ease of use and rapid learning curve.

Since UML can be converted into XMI (i.e., XML), it allows a wide range of processing options. This, in turn, allows for the repository's data model to be used as the starting point for a set of processes that not only create the actual database, but also facilitate access to its data items (this may be done, e.g., through the use of code automatically generated from the UML model).

In addition to the above, UML diagrams provide a useful source of documentation for the current state of the repository model. In fact, meta-information present in the UML diagrams may even be included in the database, thus enriching the data sets already there with a meta level.

4.1. Canonical model

The canonical model consists of a set of class diagrams that specify the entities involved in the description of language components. Such components are morphological entities, inflexion paradigms, predicates and their arguments, and so on.

The canonical model is based on existing large coverage models, i.e., we seek a broad linguistic description that crosses information from various levels, including but not limited to morphology, syntax, and semantics. Examples of existing models, as mentioned before, are the ones resulting from the PAROLE project and its follow-up, the SIMPLE project.

4.2. Data input and output models

Data input/output models are used to describe external formats, i.e., formats of data to include in or to obtain from the repository. These models may already exist in some form (e.g. an SGML DTD) or they may be implicit (e.g. SMorph, ispell (Gorin et al., 1971–2003) use tabled data).

We isolate these models to clearly separate the repository's canonical model from the outside world. Nevertheless, we maintain open the possibility of interaction with other ways of storing/representing data. The following aspects must be taken into account.

4.2.1. Information aggregation

The repository is not limited in its capacity for storing objects by differences in the various descriptive levels of data to be imported, nor because of information concerning a particular domain. In fact, the repository is able to support multiple levels and domains, as well as the relationships between their objects, thus becoming an important asset for the tasks of information aggregation and organization.

4.2.2. Multiple levels

We consider multiple information levels referring to the ones described in the literature (morphology, syntax, and so on). But we are not limited to these "traditional" descriptions: it may be of interest to include support for other levels, e.g. one halfway between morphology and syntax. The design of the database must provide support both to existing descriptions and to descriptions resulting from either cross-references of existing data or from including new data in the repository. Evolution to improve support must, however, ensure that current uses remain possible.

4.2.3. Multiple sources

In addition to the aspect presented in §4.2.2., we must also consider the existence of multiple information sources in the context of a given domain: data may originate from different projects and/or applications. The main concern here is maintaining the expressiveness of the original data, as well as the integrity of their meaning and the consistency of the data already in the repository. The danger stems from using different formats and descriptions for stored and imported data. As an example, morphology models defined by the PAROLE project are much more expressive than those defined by, say, a morphological analyzer such as JSpell (de Almeida and Pinto, 1994). The repository must be able to import/export both data sets according to their original models.

The coexistence of multiple sources is a non-trivial problem, especially if the canonical model assumes links between description levels: importing data from sources without those links may require additional assumptions. An example: PAROLE/SIMPLE morphological entities may be associated with syntactic units and these with semantics units; in contrast, syntactic data from project Edite (Marques da Silva, 1997), while also associated with semantic information (different from that of SIMPLE), is not directly associated with the morphological level.

Regarding integrity, consider a morphological entity: it may be defined in different ways by different models. However, when stored in the repository, it must be represented as a single object with the semantics of each original source model. This implies that the canonical model must be sufficiently flexible and expressive to ensure that the semantics of imported objects is not destroyed.

4.2.4. Relationships and non-linguistic data

Beyond data items, which may come from various independent sources and possibly unrelated domains, the database must contemplate the possible existence of relationships between the objects it stores. We have seen examples of those relationships (e.g. between morphological and semantics objects, or those existing between syntax and semantics objects). Other relationships may be created and stored, to account for any aspect deemed of interest: e.g. relationships with non-linguistic data, such as ontologies.

In general, relationships are not restricted in what concerns the number of related objects, i.e., the database supports any multiplicity.

4.3. Data transformation models

These models allow resources from the repository to be adapted to diverse applications. Some of these applications may precede the repository and require proprietary formats. This compatibility issue is just one example of the more general problem of exporting data described according to the canonical model to formats described according with outside models. The export capability is of great importance, since the repository must ensure its usefulness for existing applications.

Two sets of models have, thus, been defined: the first contains models of the transformations needed for converting from data described by external models and the canonical model. The second set contains models of the transformations needed for converting from data described by the canonical model and external models.

4.4. Programming interface

More than being just a source of passive data, the repository supports "online" uses. To support online clients, the repository must support some kind of communication mechanism with its users, regardless of them being humans or machines. Thus, in addition to being able to import/export data with existing formats, the repository must also provide a clearly defined interface (e.g. a programming interface).

5. Implementation

We now present implementations for each of the previous concepts.

5.1. The canonical model

Implementing the canonical model consists of defining the model proper and deploying it using some kind of data storage solution. Requirements as defined in §4.1. must be satisfied.

Work on the modeling task started with the study of existing large coverage models defined by the PAROLE/SIMPLE projects. Their models, published as SGML DTDs, were enriched according to the requirements for supporting both the new concepts and existing concepts that underwent some refinements. The resulting data model differs from the original, but is still very close and has proven to be sufficient for covering other models.

We chose a relational database (RDB) to implement the repository. RDBs confer flexibility to the global design of the system using it. The flexibility is directly linked to fine data granularity provided by database tables and by the operations provided to work with them, e.g., dynamic changes are possible, making it possible to perform changes to data structures while in use. RDBs are also flexible in the possible views they allow to be defined over data: they allow finer selection according to the client's interests.

Any candidate RDB engine must possess some way of verifying and enforcing data integrity constraints (e.g. references to foreign keys). The exact nature of these mechanisms is not important in itself, but must be taken into account when processing data.

Our choice for storage and data management was MySQL (MySQL, n.d.). Tables and integrity maintenance constraints were generated using XSLT scripts taking as input the original UML repository models. Note that only the canonical model diagrams are used in this task, i.e., the data input/output and data transformation models are not used.

5.2. Access to the canonical repository

For convenience and flexibility, a network interface should be provided. This feature, present in almost all modern RDBs, should not prove difficult to implement. It may be either a proprietary or an open protocol implementing some kind of distributed SQL transport mechanism. Examples are ODBC (Microsoft Corporation, n.d.) and JDBC (Sun Microsystems, Inc., n.d.). We privileged openness, since it facilitates portability and maintenance.

Since our main source of interaction would come from a set of C++ applications we started by defining a programming interface for this language. A connectivity library (DTL/ODBC (Gradman and Joy, n.d.)) was used to link the lower level RDB access with the higher level program interface (a set of automatically generated C++ classes representing database concepts). The generation of these classes was done using XSLT, taking as input the original canonical model UML diagrams. Since this was the method using the database itself, we are able to ensure parallelism and minimize mismatches in concept mapping.

Regardless of these methods, access to the repository is open to other methods. This is one of the advantages of using a RDB engine as a separate data management agent. In particular, use of other languages is possible, as long as they support the concepts in the repository, e.g., via the object abstraction. We introduce this requirement to prevent the high costs associated with explicit management of non-native concepts in the target language. Another requirement is that a low-level RDB interaction library (either native/proprietary or open/standard) exists that supports the chosen language. Once again, this is to avoid pursuing expensive solutions.

5.3. Data input and output models

As mentioned above, these models are used to describe data to be imported/exported to/from the repository, i.e., to be converted to/from the canonical data model.

These models may be described using UML (same advantages as for the canonical model), but other data description languages, such as XML Schema Definitions (XSD), may be acceptable as long as their expressiveness is deemed sufficient for automatic processing and documentation purposes. If the original description does not exist, it is possible that one or more UML models may cover the data to be processed. Selecting the appropriate external model will depend on the current canonical model and on how well the external model allows the external data to be mapped onto the canonical representation.

These models do not require further implementation or support (they are assumed to be supported by some outside application/system). In what concerns our work, they are to be used as input for the code derived from the data transformation models (see §4.3.).

5.4. Data transformation models

Our work with these models is limited to test cases. Namely, we defined input transformation models for the Portuguese data resulting from the PAROLE/SIMPLE projects. Although preliminary, at the time of this writing, the work allows us to envision the best way of implementing other filters for loading arbitrary data. Output transformation models have not been explicitly implemented: currently, we obtain data from the database, either through the programming interface, associated with the canonical model, or directly, via SQL commands.

6. Maintenance

There are two main aspects regarding maintenance. The first is repository content management: this aspect accounts for future expansion both of data content and expressiveness of data models, i.e., their descriptive power.

The second maintenance aspect concerns management of data models: this item covers aspects relating to miscellaneous remodeling operations and possible redefinition of the canonical model. This implies transition operations between successive canonical models, which are instantiations of data import/export operations, albeit possibly more complex than the ones used by applications such as a morphological analyzer.

Content maintenance aspects and model maintenance aspects remain to be fully addressed. Data model maintenance has been partially addressed by the use of UML diagrams and subsequent code generation operations that allow full access to the corresponding repository data.

7. Final remarks and future directions

Results so far, obtained with large data sets, allow us to conclude that our approach addresses the requirements stated above. Moreover, given the complexity of the data loaded (the entire Portuguese Language PAROLE/SIMPLE lexicon) into the database and the noted absence of major problems, we envision easy progress towards the end of including new data from other sources.

We are also able to conclude that our work points to a more general solution to the problem of data reuse and integration. In addition, it opens the door to seamless integration with other data description levels, such as language-oriented ontologies.

8. References

Antoni-Lay, Marie-Helène, Gil Francopoulo, and Laurence Zaysser, 1994. A Generic Model for Reusable Lexicons: the Genelex Project. *Literary and Linguistic Computing*, 9(1):47–54. Oxford University Press.

Aıt-Mokhtar, S., 1998. *L'analyse presyntaxique en une seule etape*. Thèse de doctorat, Universite Blaise Pascal, GRIL, Clermont-Ferrand.

Booch, Grady, James Rumbaugh, and Ivar Jacobson, 1999. *The Unified Modeling Language User Guide*. Addison-Wesley Longman, Inc. ISBN 0-201-57168-4.

de Almeida, Jose João Dias and Ulisses Pinto, 1994. Jspell – um modulo para a analise lexica generica de linguagem natural. In *Encontro da Associação Portuguesa de Linguıstica*. Evora.

EUROTRA-7, 1991. Feasibility and project definition study of the reusability of lexical and terminological resources in computerised applications.

Gorin, R. E., Pace Willisson, Walt Buehring, and Geoff Kuenning, 1971–2003. International ispell. `http://www.gnu.org/software/ispell/ispell.html`.

Gradman, Michael and Corwin Joy, n.d. *Database Template Library*. See: `http://dtemplatelib.sf.net/`.

Marques da Silva, Maria Luısa, 1997. *Edite, um sistema de acesso a base de dados em linguagem natural. Analise Morfologica, Sintactica e Semântica*. Tese de mestrado, Instituto Superior Tecnico, Lisboa.

Medeiros, Jose Carlos, 1995. *Processamento Morfologico e Correcção Ortografica do Português*. Tese de mestrado, Instituto Superior Tecnico, Lisboa.

Microsoft Corporation, n.d. ODBC – Microsoft Open Database Connectivity. Specifications and implementations may be found, among other places, at: `http://msdn.microsoft.com/library/en-us/odbc/htm/odbcstartpage1.asp`, `www.iodbc.org`, or `www.unixodbc.org`.

MySQL, n.d. *MySQL Database Server*. MySQL A.B. See: `http://www.mysql.com/products/mysql/`.

OMG, 2002. *XML Metadata Interchange (XMI) Specification, v1.2*. Object Management Group (OMG). See: `www.omg.org/technology/documents/formal/xmi.htm`.

OMG, n.d. *Unified Modelling Language*. Object Management Group (OMG). See: `www.uml.org`.

Paprotte, Wolf and Frank Schumacher, 1993. MULTILEX – Final Report WP9: MLEXd. Technical Report MWP8-MS Final Version, Westfalische Wilhelms-Universitat Munster.

PAROLE, 1998. *Preparatory Action for Linguistic Resources Organisation for Language Engineering – PAROLE*. `http://www.hltcentral.org/projects/detail.php?acronym=PAROLE`.

SIMPLE, 2000. *Semantic Information for Multifunctional Plurilingual Lexica – SIMPLE*. `http://www.hltcentral.org/projects/detail.php?acronym=SIMPLE`.

Sun Microsystems, Inc., n.d. JDBC Data Access API. See: `http://java.sun.com/products/jdbc/`.

W3C, 2001a. *Extensible Markup Language*. World Wide Web Consortium (W3C). See: `www.w3c.org/XML`.

W3C, 2001b. *XML Schema*. World Wide Web Consortium (W3C). See: `www.w3c.org/XML/Schema` and `www.oasis-open.org/cover/schemas.html`.

Walker, D., A. Zampolli, and N. Calzolari (eds.), 1995. *Automating the lexicon: Research and practice in a multilingual environment*. Oxford: Oxord University Press.

Acknowledgments

This paper was supported by the DIGA Project (POSI/PLP/41319/2001).