

Toward the Interoperability of Language Resources

July 13-15, in conjunction with the 2007 LSA Summer Institute



Issue Statement

Michael Cochran
SIL International

As we look at the integration of applications to support the tasks of language research and documentation, we focus on strategies for allowing integrated yet somewhat independent tasks to be accomplished. The “Unix approach” involves multiple independent applications that can be chained together to accomplish a task. The “Microsoft approach” involves fairly monolithic applications that help do specific, integrated tasks.

Looking at focused, independent applications, we find they are: (1) great for users who are doing independent tasks; (2) hard for users doing integrated tasks (and for the support staff who must make the applications work together); and (3) easier for developers. In contrast, monolithic, integrated applications are: (1) great for users doing very integrated tasks with large integrated data sets that would be hard to pass data back and forth between independent applications; but (2) hard for developers, as they have to understand how the tasks interrelate.

Both independent and integrated applications are used in language work. Some tasks are so integrated that the applications that support them must be integrated. Other tasks are independent enough that the cost of integrated applications is not warranted. As additional independent applications are developed, the number of integrations between applications that must be developed increases dramatically. For example, two applications require two integrations (one in each direction), whereas three applications require six integrations, and five applications would require twenty in order to support all possible paths for interaction. In order to address this proliferation of interactions standards for interoperation can be defined that will address many of the issues. It should be noted however that the interpretation of that data that influence the behavior of an application demand integration beyond what data standards usually provide. This level of integration is often only available through integrated applications.

Several different issues must be solved. Solutions are different for an integrated suite than for independent applications. Learning the applications (graphical user interfaces, training materials, terminology, and so on) may be easier across a suite than for a set of independent tools. Sharing data between applications (synchronizing shared data and keeping it up to date) is often harder with independent tools. Synchronizing data may also involve addressing data integrity issues (e.g., when a class of noun is changed it needs to propagate through all applications). The setup of the language data (discussed below) influences all applications that use the data and must be equivalent in all the applications.

Several issues relate to the entry and display of language data, and how various applications can be made to support the language data in the same way. They include: encodings (Unicode vs. hacked encodings); characters that are not in the Unicode standard; script behaviors that are not supported by current operating systems; strategies for entering language data; the actual fonts, keyboards, and conversion mappings created for particular scripts, including the International Phonetic Alphabet.



A core issue in sharing data between applications involves the setup information outlined above. For each language used, each application must agree on keyboards, fonts, special characters, conversion strategies and mapping files to share data.

Another issue is publishing from the applications. Publishing is a specialized domain in itself. Most linguistic research results in publishing. Applications must target publishing tools that support the correct publishing quality. These tools must also support the language issues outlined above.

Another thing that needs to be integrated is task help and training. SIL has been experimenting with the integration of knowledgeware into our language software applications. This allows users access to “just in time” training materials to help them to do linguistics tasks in which they are engaged.

Many issues exist relative to the mapping of data between different applications. As an example, consider Toolbox and Phonology Assistant (PA), independent applications that access the same data to accomplish their particular tasks. In Toolbox, the phonetic data is in one of the fields for each of the lexical records. In the past, you would need to save your Toolbox data, and then create a utility to import the phonetics part of the data from the Toolbox file into PA, so that you could do phonology work. This demanded that the tools be able to import/export relative to each other, and the data in the applications were somehow mapped to each other. At the very least, the user had to be told how the fields in the different applications are related, and be given strategies for doing tasks that require both tools.

Our new approach is to have PA read the Toolbox file directly. The user can identify the fieldname for use as phonetic data. Then all of the phonological pattern matching and research is done on the Toolbox data itself. When the user moves into the Toolbox file and makes changes, then PA rereads the data and reapplies the search. For FieldWorks, PA also reads all of the setup information and directly uses it so that there is no need to enter any of the setup information.

In conclusion, we are aware of and continue to address issues related to integration of language applications. We have solutions to some of the issues above and plan to resolve others. I would like to hear from others thinking about these issues and how they have addressed them or intend to address them.

