Multilingual natural language generation for multilingual software: a functional linguistic approach

John A. Bateman
Language and Communication Research Department of English Studies
University of Stirling
Stirling FK9 4LA
Scotland, U.K.

Christian M.I.M. Matthiessen
Speech, Hearing and Language Research Centre (SHLRC)
Macquarie University
Sydney NSW 2109, Australia

Licheng Zeng

ABSTRACT
In this paper we present an implemented account of multilingual linguistic resources for multilingual text generation that improves significantly on the degree of re-use of resources both across languages and across applications. We argue that this is a necessary step for multilingual generation in order to reduce the high cost of constructing linguistic resources and to make NLG relevant for a wider range of applications—particularly, in this paper, for multilingual software and user interfaces. We begin by contrasting both a weak and a strong approach to multilinguality in the state of the art in multilingual text generation. Neither approach has provided sufficient principles for organizing multilingual work. We then introduce our framework where multilingual variation is included as an intrinsic feature of all levels of representation. We provide an example of multilingual tactical generation using this approach and discuss some of the performance, maintenance and development issues that arise.

INTRODUCTION

Natural Language Generation (NLG) is the automatic production of natural language ‘texts’ (written or spoken, long or short, hyper or linear) on the basis of some computer-internal representation of information. It can be regarded as a set of generic methods for providing flexible, on-demand, reader-oriented textual information. Multilingual NLG (MLG) is the logical extension of this enterprise to consider the techniques, representations and relevant applications for the automatic production of texts in various languages within a single system. Typically in MLG, the information to be expressed is held broadly constant and the desired target languages vary; this has been suggested as an effective alternative to machine translation (MT) when producing documents in various languages (cf. Zubov, 1984; Kittredge, Iordanskaja and Polguère, 1988; Rösner, 1992; Hartley and Paris, 1995), and as a strategy for broadening access to information and information systems in a multilingual society (e.g., Glass, Polifroni and Seneff, 1994; Alexa, Bateman, Henschel and Teich, 1996; Schütz, 1996; Somers, Black, Nivre, Lager, Multari, Gilardoni, Ellman and Rogers, 1997).

‘Full’ MLG involves several substantial overheads however. Most significantly, the information to be expressed must be explicitly represented in a ‘prelinguistic’ abstract form (such as in a knowledge base) and linguistic resources (grammars, lexicons, mappings from semantics to grammar, etc.) must be developed and maintained for each language covered. As a consequence, most areas of software design, including software internationalization and localization, would have until recently considered MLG too technology-intensive and experimental to warrant application. But this situation is changing: first, the boundaries between desirable software features and traditional areas of investigation in NLG are being progressively blurred (cf. Karkaletsis, Spyropoulos and Vouros, 1998) and, second, the ability of NLG to...
meet practical application requirements forms an increasingly central concern for NLG systems (cf., e.g., Rambow and Korelsky, 1992; Reiter and Mellish, 1993; Reiter, Mellish and Levine, 1995; Coch, 1996; Lavoie and Rambow, 1997).

One consequence of this changing situation is that traditional methods of constructing multilingual software will not provide a sound basis for advanced software design. When user interfaces, program diagnostics and online documentation are expected not only to become increasingly user-friendly and user-adaptive but also to be provided in the language of choice of the user, more powerful and flexible methods will need to be employed. In contrast to the maintenance of distinct language-specific message catalogues or the combination of language-parameterized message fragments or templates, an appropriate MLG architecture has much to recommend it as a provider of multilingual capabilities. According to Karkaletsis et al. (1998), for example, advantages include the tailoring of messages to a user’s level of expertise and to the current software context, a reduction in the cost of development of new and localized software versions, and an enforcement of externally imposed writing rules and content requirements. For this endeavor to succeed, however, it is crucial that the above mentioned overheads for MLG be reduced to such an extent that the increased functionalities provided offset the additional effort and necessary changes in work practices involved.

The first source of overheads mentioned, the provision of abstract representations of the informational content of software messages, is currently the subject of a number of investigations (cf. Paris and Vander Linden, 1996; Hoppenbrouwers, van der Vos and Hoppenbrouwers, 1996; Karkaletsis et al., 1998) and there are a range of solutions being pursued. However the second type of overhead, the provision of the linguistic means for the expression of such messages, has not so far been rigorously considered. Linguistic resources for message generation are still too often constructed on an ad hoc basis for the purposes of particular demonstrations, prototypes or restricted applications. It is precisely this second area, therefore, that provides the focus for this paper. In particular, we ask where the phenomenon of multilinguality is best to be ‘placed’ in an MLG system for effective and appropriate multilingual generation functionalities to be achieved for multilingual software. This is a fundamental MLG design decision since it determines the extent to which the processing strategies and linguistic resources developed (e.g., grammars, lexicons, semantic ontologies, etc.) can be re-used across languages. Re-use is a critical factor for the cost and development time of providing multilingual solutions: the more that can be realistically re-used, the less overhead is invoked when expanding on the languages offered. Moreover, the benefits of re-use grow as the expressive requirements for an MLG component increase: providing different ‘phasings’ of a single message according to the user’s level of expertise or stated style preference (cf. Bateman and Paris, 1989), for example, requires correspondingly more complex linguistic resources, all of which then have to be repeated multilingually.

We organize our discussion as follows. There are now several systems where multilingual generation of some kind is attempted. In Section 2 we show how these accounts have not succeeded in minimizing the linguistic overhead introduced by attempting multilingual generation and so do not provide an optimal basis for further development. In Section 3 we introduce an implemented framework for multilingual linguistic description that does maximize the degree of re-use that is achievable across languages. It succeeds in this by virtue of its very explicit position on the placement of multilinguality—a position based on the notion of a multilingual functional typology that we have developed drawing on insights from functional linguistics. In Section 4, we provide an example of multilingual message generation using multilingual resources organized as we propose and, in Section 5, go on to discuss some of the implementation, development, and maintenance issues that are raised. Finally, in Section 6, we conclude by summarizing our discussions and drawing attention to some of the more general issues for future applications for which our approach lays foundations.
Figure 1: Two simple models for multilingual generation

THE ‘PLACEMENT’ OF MULTILINGUALITY IN MULTILINGUAL GENERATION SYSTEMS

The basic generation functionality with which we will be concerned in this paper is the following: a generation component should be able to generate the “same message in more than one language using the grammar, lexical and morphological rules for the supported languages” (Karkaletsis et al., 1998, p324). In addition to this, we also keep in mind that it would be useful “to be able to express the same message in a single language in different ways, according to user expertise, style and plans” (ibid., p323). Within NLG, such components are typically referred to as ‘tactical generators’, reflecting a traditional division of the generation process into at least two phases: determining what is to be communicated (‘strategic generation’), and determining how that content is to be expressed (‘tactical generation’). Although our own account does not particularly follow this division (which originates in Thompson (1977)), it is nevertheless useful for expository purposes and so will be adhered to here.*

Two simple possible architectures for multilingual tactical generation are contrasted in Figure 1; they differ precisely in their placements of multilinguality. The architecture on the left places multilinguality outside of the generation system entirely: different languages simply require different generators, each with its own algorithms and data structures. Within these individual generators there is no ‘multilinguality’ at all: each component is only aware of the one language with which it is concerned. Rössner (1992) accordingly labels this treatment of multilinguality ‘weak multilinguality’. In contrast to this, the architecture on the right is said by Rössner to exhibit ‘strong multilinguality’. Here there is a commitment to a single generation algorithm that is used across all of the languages to be covered by the multilingual system.

Weak multilinguality, despite its apparent duplication of necessary processing components, is often claimed to be a simpler option organizationally—largely because it avoids the need for close (or any) cooperation between components. Rössner (1994, p130) goes so far as to suggest that this approach is therefore a ‘safer’ engineering option which might well be justifiable for applications-oriented generation.

*We will also generally restrict our attention to issues of sentence generation rather than the generation of longer stretches of text—although it is worth noting that the majority of the points we make concerning multilinguality find even stronger support when the complexities of text generation proper are addressed.
systems. The prior existence of tactical generators in targeted languages is also a consideration in its favor. Rösner’s own TECHDOC system (cf. Rösner and Stede, 1994), for example, a system for the automatic production of technical instruction manuals in (primarily) English and German, initially incorporated the Penman sentence generator (Mann and Matthiessen, 1985; Matthiessen and Bateman, 1991) for the generation of English and Rösner’s own SEMTEX generator (Rösner, 1987) for German—two completely unrelated components. Weak multilinguality is also adopted when there are differences in theoretical interests among those working on the respective generation components and/or a lack of an effective solution to distributed development work. Thus, the recently completed GIST project (European Union, LRE: cf. Not and Stock, 1994), a system for generating instructions for completing administrative forms in English, Italian and German, also adopted weak multilinguality for its tactical generation components, using three completely independent tactical language generators each based on a different kind of linguistic theory, even though much of the coverage necessary for Italian and German had then mostly to be developed or implemented within the project.

These alleged organizational simplifications of the weak multilingual architecture are, however, more than offset by several inherent drawbacks. When considering weak multilinguality for any part of an MLG system as a whole, it is necessary to ensure that appropriate input forms for each component are created: since these input forms are independent and defined only by the tactical generators for which they serve (there is no ‘standard’ input specification language for tactical generators), there is no guarantee of even approximate commensurability. An API-‘wrapper’ (e.g., Vander Linden, 1994) must be defined to insulate the rest of the system from variation in the inputs required and this wrapper may need to do more or less work depending on those generators’ individual details. It is not, in general, possible to predict how much work is involved in making any given monolingual generator compatible with the demands of the multilingual API: this may require additions to the coverage of the generator, collapsing of distinct alternatives into single alternatives, or matching combinations of input features with particular outputs. Many of these problems arise because in the weak multilingual approach there is also no guarantee that the expressive capabilities of the distinct generation components are at all comparable: if one system is capable of responding well to highly differentiated inputs and another not, then there will be a severe discord in the quality of texts generated in those languages. Usually, components are being gathered in an MLG system precisely because similar texts are to be generated across the languages covered: synchronizing the coverage and input specifications of a heterogeneous set of tactical generators is then clearly an organizational overhead.

In contrast to the architectural heterogeneity of weak multilinguality, strong multilinguality has apparent advantages of modularity and generalization. There is a clear separation of process (generation) and data (linguistic description) and the generation process must from the outset be designed to be adequate for handling a variety of language specifications—making it more likely to support general solutions. Most current MLG systems in fact now exhibit some variant of this architecture, albeit often because experimental systems are being constructed in which the individual language descriptions are also being built anew in the, sometimes idiosyncratic, forms that those systems require. The multilingual message generation systems of Glass et al. (1994), Somers et al. (1997) and Karkaletsis et al. (1998) and many others are all of this kind: there is little re-use of existing generation resources—neither across languages within each system nor from resources externally developed. Such approaches are therefore still paying significant overheads for their multilingual abilities—overheads that primarily involve increased development times for more sophisticated levels of generation functionality or functionality that is inherently restricted in scope.

The strong multilingual position alone is not then itself sufficient to allow distinct language descriptions to lever off one another to encourage re-use. Indeed, there is no theoretical requirement that the concrete language descriptions be related in any way beyond being expressed in a common formalism or implementation language. This is problematic because the range of variation possible across the different language versions of a message can appear substantial. Consider, for example, the following software help message that might be produced for a drawing application or any product where the user can construct a diagrammatic representation.

Start the MAKE-PLINE command by choosing Polyline from the Polyline flyout on the Draw
Localized versions of this software need to produce messages that express approximately the same information; this will include phrases such as:

**Chinese:**

xuanze huitu gongjubang shang duobianxing fubiao zhong de duobianxing, kaishi zhixing MAKE-PLINE mingling

select (draw toolbar-on) (polyline flyout-in)-of polyline, start-execute (MAKE-PLINE command)

**Russian:**

Zapustite komandu MAKE-PLINE, vybrav punkt Polyline v palitre Polyline na paneli instrumentov Draw

Start (command MAKE-PLINE), choosing (option Polyline) (in flyout Polyline (at toolbar of instrument named Draw))

**Spanish:**

De la barra de tareas despligue el menu ‘Dibujar’ y elija la opción Polyline para ejecutar la orden MAKE-PLINE

(From the toolbar) open (the menu ‘Draw’) and select (the option Polyline) in order to execute (the command MAKE-PLINE)

To aid comparability, we have added approximate indications of structural and lexical correspondences—these are for the most part for current expository purposes and should not be taken as indicating ‘translation equivalences’ or similar notions.

There are many superficial, and not so superficial, differences between these messages: individual phrases are expressed using a variety of word orders (e.g., ‘MAKE-PLINE command’ vs. ‘komandu MAKE-PLINE’); there are different grammatical constructions expressing the relationship between the starting of the command and the means by which that starting is achieved, i.e., the choosing (e.g., ‘by choosing’ in English vs. Russian ‘vybrav’ (a participal form expressing means) vs. Spanish ‘para’ (‘in order to’); some languages use determiners (‘the Polyline flyout’, ‘le orden MAKE-PLINE’), others not (‘komandu MAKE-PLINE’, ‘MAKE-PLINE mingling’); some languages have extensive morphological contributions, others hardly any; and so on. We return to some of these in Section 4 below when we consider their generation. Relevant here is that it is certainly not to be assumed that the development group concerned with localization into any one of these languages has any knowledge about the requirements of the others. Individual language development thus becomes an essentially monolingual endeavor. In the terms of the diagram of Figure 1, the data sets for the distinct languages in the strong multilingual model remain unrelated, and the resulting systems, although implementationally strongly multilingual, are in fact more akin to weak multilingual systems with similar attendant problems.

The adoption of distinct language descriptions is far from ideal as a complete and cost-effective multilingual solution. It is well established that there are commonalities between languages within and across descriptions and developing descriptions independently does not allow these commonalities—or congruences as we shall now term them—to be used. This is important for achieving an effective multilinguality because the existence of congruences between language descriptions has also been demonstrated to have significant practical consequences. Several experiments have been carried out in which new descriptions of languages not previously covered by a system have been ‘derived’ from existing descriptions. Work within the Core Language Engine framework has been particularly significant in this respect and development times for extensive language capabilities—primarily analysis—in previously uncovered languages have been reported to be dramatically reduced by description re-use (cf. Alshawi, Carter, Gambäck and Rayner, 1992; Rayner, Carter and Bouillon, 1996). Similar results have also been achieved in MLG systems. For example, even though Rössner’s description of TECHDOC, as mentioned above, orients itself to weak multilinguality, in fact the system soon shifted to a strong multilingual architecture—tactical generation being provided for both English and German by the Penman generation system. Moreover, the necessary German generation resources were derived from the existing English resources by editing them...
where the languages diverged. The re-use approach was also adopted from the outset in the Drafter sys-
tem (e.g., Paris and Vander Linden, 1996). In Drafter, software designers are encouraged (by a suitable
development interface) to provide software specifications that are also interpretable by an MLG compo-
ent so that ‘first drafts’ (in English and French) of instructions for using the software can be produced
automatically. In this system the French generation resources were again produced by editing existing En-
glish resources (actually the same resources that were edited in TECHDOC: the Nigel systemic-functional
generation grammar of English (cf. Mann and Matthiessen, 1985)) wherever the two languages diverged.

In summary, then, both theoretical approaches to contrastive linguistic descriptions—particularly func-
tional ones as we describe in the next section—and practical results in MLG system construction argue for
the existence of substantial opportunities for re-use across language descriptions. The strong multilingual
position is compatible with this situation but does not predict it. Strong multilinguality alone accordingly
provides no mechanisms for deliberately making use of such re-use in order to reduce development time
and costs, nor for supporting subsequent maintenance. Comparability in coverage and performance is also
unenforced and it remains for the concrete language descriptions to deliver the goods; there is nothing
in the architecture itself that demands this or even offers support. An even ‘stronger’ multilinguality is
required.

**INTRINSICALLY MULTILINGUAL GENERATION GRAMMARS**

Two General Goals for Multilingual Descriptions

We have noted that re-use of linguistic descriptions is possible across languages and have suggested that
such re-use could reduce the overheads of achieving MLG functionality. To uncover usable commonalities
most effectively, however, it is necessary to adopt highly systematic approaches to the linguistic organi-
zation of the messages that a system is to generate. In this section, we present a framework for linguistic
description that provides an explicit representation of congruences and incongruences within language de-
scriptions along several dimensions important for multilinguality. We will term this broader treatment of
multilinguality *intrinsic multilinguality*. With it, there is a better guarantee that data can be re-used across
languages and across tasks, and that individual language descriptions remain synchronized. Moreover, it
allows a finer characterization concerning how much of the work of generation can be shared across lan-
guages in an MLG system: the greater degree of intrinsic multilinguality achieved, the more benefits for
multilingual processing accrue.

Our framework consists of a multilingual extension to the concept of functional typologies as pursued
in systemic-functional linguistics (cf. Halliday, 1978). It provides for a high degree of intrinsic multilin-
guarity in our linguistic descriptions, and hence in the computational representations of those descriptions,
while simultaneously satisfying two, more often conflicting, goals that we have identified for effective
multilingual descriptions (cf., Bateman, Matthiessen, Nanri and Zeng, 1991a):

- **Integration** of the different languages so that commonality is separated from particularity and re-
  used: resources should *maximize the factoring out* of generalizations across the languages of the
  system and the particulars of individual languages in the linguistic resources;

- **Integrity** of each individual language so that it can be used separately: integrated resources should
  support consistent access from *both* the point of view of their multilinguality and the point of view
  of the individual languages.

Stating commonality in terms of multilingual resources while maintaining language-specific integrity
and difference can be achieved if a description/system makes it possible to ‘extract’ monolingual and con-
trastive perspectives of the linguistic resources defined. The question of separateness or integration is then
transformed into a matter of *view and use* of those resources: it must be possible to take a monolingual
view, focusing only on the resources of one language, and equally, it must be possible to take a multilingual view, concentrating on contrasts and commonalities. On top of this, we require that the language description includes those dimensions of organization along which language divergence and similarity can be expressed and that are most beneficial for achieving highly integrated descriptions.

**Systemic Resources: Organization**

The basic building blocks of our intrinsically multilingual descriptions are functional typologies. Functional typologies are built up within systemic-functional linguistic (SFL) approaches by forming classification hierarchies, called system networks (Halliday, 1966). Particular theoretical weight is placed on the disjunctions that are posited to hold over groups of features. Each disjunction, or grammatical system, is seen as a point of abstract functional choice: that is, the grammatical systems capture those minimal points of alternation offered by a linguistic representation level in order to express, or ‘construe’, the information maintained at the next ‘higher’ (i.e., more abstract) level in the linguistic system. This is a representation of the so-called paradigmatic axis of linguistic description and is a subsumption lattice rather than a list because abstract choices exhibit inter-dependencies: for example, the lexicogrammatical ‘choice’ in English of asking a polarity-seeking question (Did you . . . ?) and asking a wh-question (Where is . . . ?) is only available given the logically prior lexicogrammatical ‘choice’ to ask a question at all. The existence of the prior choice is motivated both by the semantics it may express and by the commonalities in syntactic form that both English yes/no questions and wh-questions share. SFL is therefore from the outset organized around ‘commonalities’ and ‘differences’ in grammatical and lexical patterning.

The orientation to semantic criteria when constructing the network classification is an important key for maximizing effective multilinguality. The semantic-functional coherence of the paradigmatic organization overall is given preference over generalizations concerning possible structures. This makes it more likely that the classifications represented by the system network carry over across languages: that is, paradigmatic organizations can often be maintained as congruent across language descriptions when a purely structural classification would not. Declarative/interrogative, for example, has strong semantic support and can usually provide a beneficial organization for widely (structurally) diverging languages: we would therefore expect that most language descriptions would include a grammatical system containing the choice between the grammatical features ‘declarative’ and ‘interrogative’.

Crucially, however, there is no requirement that paradigmatic descriptions of distinct languages are entirely congruent: this would reduce the account to a, clearly false, grammatical interlingua. Intrinsic multilinguality means that congruency can be maintained when and where it occurs but is not enforced when the language facts would be violated. This is represented by defining language-restricted ‘partitions’ over the general multilingual paradigmatic organization. Information that is valid only for a particular language is accessed by examining the collection of specifications given within all the partitions that mention that language. Any areas of paradigmatic organization that are congruent across two or more languages are simply tagged as belonging to the partitions relevant. An example of such a multilingual paradigmatic classification is shown in Figure 2; this shows a fragment of the system network for the multilingual English/Chinese/Japanese trilingual clause grammar first introduced in Bateman, Matthiessen, Nanri and Zeng (1991b). The grammatical systems with options ‘declarative’ and ‘interrogative’ are congruent with respect to Chinese, English, and Japanese (as well as many other languages) and so appear in all three partitions unchanged: the commonalities have been factored out as required.* The interlingual congruence is partial, however, and the figure captures this by also containing partitions that are valid only for Chinese and/or Japanese.

This kind of multilingual description is useful for a number of reasons: most significant here is the fact that the choice points represented provide a re-usable hook both ‘upwards’ to the input specifications for the tactical generation component and ‘downwards’ to the various ways in which the choices are expressed structurally. Both aspects can be utilized for linguistic resource development and maintenance.

---

*The feature ‘element’ replaces the more German-centric ‘wh-interrogative’.
Figure 2: Multilingual system network with conditionalized partitions
The structural expression of the choices made in a paradigmatic description such as Figure 2 is the task of realization statements associated with individual grammatical features. Realization statements set constraints on configurations of syntactic constituents, thereby specifying properties of the linguistic unit being generated. Allowable constraints include statements of linear precedence, immediate dominance, ‘unification’ of functional constituents, and further type constraints to hold for subconstituents. The resulting structures represent both the ‘constituency organization’ of the messages, i.e., which elements are to be grouped together to form larger syntactic structures, and the ‘functional labeling’ of those constituents, i.e., an explicit indication of the functional role that each constituent plays in the larger structure of which it forms a part—e.g.: Actor, Theme, Process, etc. As an example, the contrastive choice between ‘wh-interrogative’ (i.e., ‘element’) and ‘yes/no-interrogative’ (i.e., ‘polarity’) is realized in English differentially by either one or other of the two constraint sets:

- include a Wh-element (indicated by the realization statement: [+ Wh]) and order this before the Finite-element (the finite part of the verbal group, i.e., that carrying agreement and tense; realization statement: [Wh \^ Finite]),
- order the Subject after the Finite-element [Finite \^ Subject] and conflate (‘unify’) the Finite-element with the (interpersonal aspect of the) Theme [Theme / Finite].

These contribute respectively to the following two contrasting grammatical fragments where we can see that the constraints of the realization statements are satisfied:

<table>
<thead>
<tr>
<th>Who is going?</th>
<th>Are you going?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wh/Theme/Subject Finite Process</td>
<td>Finite Subject Process</td>
</tr>
</tbody>
</table>

Realization statements thus provide a kind of ‘template construction language’ where the templates concerned are actually the well-formed and functionally motivated syntactic expressions of a language. A detailed description of realization statements and of their place in a systemic description is given in, for example, Matthiessen and Bateman (1991, pp95-97).

Functionally labeled constituents and their configurations form a further level of possible congruence across languages. This is illustrated in Figure 3, which shows the functional structures for the English and Chinese versions of our example software message from Section 2. Here we can see that even with this, rather low, level of linguistic abstraction, some significant differences between the languages are neutralized. Individual ‘functional configurations’ are sometimes very similar cross-linguistically and, when different, are often different in systematic ways. All the clauses here, for example, include a ‘Process Goal’ configuration reflecting the similar semantics involved: some action (the ‘Process’) is performed on some object (the ‘Goal’); similarly, the structures employed for the expression of locations are in both languages expressed by a combination of a spatial relation (‘Minorprocess’) and the location itself (‘Minirange’), although the relative orderings of these are different. Organizing language descriptions along dimensions such as these begins to bring out relationships of similarity between languages that might superficially appear to be quite different.

It is also clear, however, that these syntagmatic congruences are limited; there are often substantial variations in grammatical structure even within quite closely related languages. An account that was based solely, or even primarily, on such a level of representation would have to give up on re-usable congruent descriptions at this point. In contrast, a multilingual systemic classification hierarchy allows us to go much further. Functional typologies cleanly separate the classification hierarchy (the paradigmatic axis of description) from the structural configurations over which the classification holds (the syntagmatic axis). The representation of congruence and incongruence in paradigmatic choices is unaffected by local incongruence in structure: thus, as suggested in Figure 2, the paradigmatic options of a lexicogrammar may be held common across languages even when they are expressed structurally in quite different ways. Minor complexities in structural realization are in effect ‘hidden’ from higher level processes—encapsulating them...
**English:**

<table>
<thead>
<tr>
<th>Clause 1 (end-state)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process</strong></td>
</tr>
<tr>
<td>Deictic</td>
</tr>
<tr>
<td>Start the MAKE-PLINE command</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clause 2 (means)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sb</strong></td>
</tr>
<tr>
<td>Thing</td>
</tr>
<tr>
<td>Deictic</td>
</tr>
<tr>
<td>Locative-qualifier</td>
</tr>
</tbody>
</table>

by choosing Polyline from the Polyline flyout on the Draw toolbar

<table>
<thead>
<tr>
<th>Clause 1 (means)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process</strong></td>
</tr>
<tr>
<td>Qualifier (of flyout)</td>
</tr>
<tr>
<td>Minirange</td>
</tr>
</tbody>
</table>

**Chinese:**

<table>
<thead>
<tr>
<th>Clause 1 (means)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process</strong></td>
</tr>
<tr>
<td>xuanze</td>
</tr>
<tr>
<td>select draw toolbar on polyline flyout in of polyline</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clause 2 (end)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phase</strong></td>
</tr>
<tr>
<td>kaishi</td>
</tr>
<tr>
<td>start execute MAKE-PLINE command</td>
</tr>
</tbody>
</table>

**Abbreviations:**
- Class.: Classifier
- MProc: Minorprocess
- Sb: Subordinator
- Q-marker: Qualifier-marker

Figure 3: Contrasting functional structures for the English and Chinese example software message
in the syntagmatic description of individual languages where they belong. It is also possible (and very frequently used for closely related languages) that language conditionalizations be applied to individual components within the syntagmatic descriptions of any feature. This is illustrated in Figure 4 for the interrogative/declarative grammatical system as it might appear for the languages English and German. Here we can see that statements both of commonalities (outside of particular language partitions) and of differences (in particular language partitions) are explicitly represented: the system remains unchanged—i.e., congruent with respect to the German and English language views—and partial congruence is also maintained in the realizations. This provides a strong basis for development and maintenance, as we describe in Section 5 below. These possibilities for conditionalization begin to indicate the flexibility of the placement of multilinguality in our account. Furthermore, in addition to the grammatical systems and constraints on structure (realization statements) illustrated here, we also apply similar conditionalization to all levels of linguistic description represented in the systemic-functional framework, including lexical entries, punctuation rules, logical forms, example test suites, and the mapping rules from semantics to the grammatical features shown in system networks.

Multilingual systemic descriptions clearly meet the demand for individual language description integrity that we set out at the beginning of this section: the framework provides multilingual resources from which language-specific views can be built. The monolingual view contains those parts of the resources that are within partitions specific to the language in question: it can simply be ‘lifted out’ from the multilingual resources—and, as we describe in the next section, it is possible to organize generation processes in such a way that they only refer to this view. At the same time, there is a multilingual view capturing integration, where congruence across (some set of) languages is represented by congruence in the specifications without the need for additional statements of correspondence.

This integrated approach to intrinsic multilinguality can be seen clearly in the following example. The draft EAGLES (‘Expert Advisory Group on Language Engineering Standards’, a European Union project) recommendations on morphology (Monachini and Calzolari, 1994) list the linguistic features necessary for describing the morphology of nine European languages. Whereas this report presents its information as a list of descriptions for each language making reference to the features which apply in any particular case and those which do not, we can straightforwardly re-express this information in the form of multilingual system networks. This allows us to view the entire account both monolingually for each language and contrastively for any subset of the full range of languages covered. Figure 5(a-b), for example, shows two
Figure 5: Distinct monolingual views (a, b) and a bilingual view (c) automatically extracted from an integrated multilingual resource for morphology

automatically extracted monolingual views drawn from the full integrated multilingual resource; the information of these two views is combined in Figure 5(c) in an automatically extracted bilingual contrastive network. In this bilingual view, features without explicit language specification are valid for both of the languages displayed. This makes it particularly clear that these two languages are virtually congruent with respect to this area of their grammars, differing only by two features in the grammatical system GENDER.

Language developers can work with any of these views as required. Individual language descriptions can be considered in isolation or similar grammatical constructions or classifications can be examined cross-linguistically, comparing the resources of different languages or adopting treatments from other languages as convenient. Depending on the task—whether monolingual, bilingual, or multilingual—access to the resources is managed appropriately via the conditionalizations they contain. Moreover, since it is the responsibility of any implementation of this framework to construct such definitions internally when merging descriptions of different languages, it is possible that no user ever seeks to view the entire multilingual definition. Individual developers can focus on particular languages without needing to consider the full set. Some of the utility of this can be seen from Figure 6, which does display a fragment of the full multilingual resource in a single integrated view of the nine languages discussed in the EAGLES report; integrated views such as this which involve more than two languages show their language partitions by attaching language labels to each grammatical feature. The single definition for the nine languages is clearly complicated, but once treated as a multilingual system its information can be readily decomposed and viewed for subsets of the languages covered. Language developers can effectively disregard those areas of the description that are not relevant for the languages or tasks on which they are working.

Although straightforward, the form of intrinsic multilingual representation we have described succeeds in providing very general support for multilinguality. It goes considerably beyond one now quite common model of multilinguality in which linguistic resources are divided into ‘core areas’ that are shared and language-specific modules that refine or update the core specification. Such models range across approaches to formal syntax (e.g., Pollard and Sag, 1994, p58), to multilingual lexicons (e.g., Cahill and Gaz-

*The graphical views we use in the remainder of this paper are provided by the visualization tools of the KPML multilingual systemic-functional development environment that we describe below; these views are more effective in color since there is extensive use of color-coding.

*Note that the meaning of these features is not at issue here; what is important is allowing a language developer to focus on components of the multilingual information as required, whatever that information might be. We do not, therefore, explain any of these features that the EAGLES report proposes for morphology.
Figure 6: Fragment of the integrated multilingual view of morphology for 9 European languages according to the EAGLES recommendations
The examples discussed here, however, show that this is not suitable as a general view of multilinguality. The provision of a core grammar might appear a useful organizing decision on the basis of the French/Spanish data shown in Figure 5, where the features ‘trns’ and ‘common’ of the GENDER system would then be part of a Spanish-specific non-core partition and the rest are accepted within the core. But if we extracted instead the language pair German/Danish from the same multilingual resource as shown in Figure 7, then the situation already looks quite different. Here it seems that only the feature ‘neuter’ would belong to the core, and the rest (including ‘masculine’ and ‘feminine’ that were in the core suggested by French/Spanish) are in language-specific non-core partitions belonging to either German or Danish.

While still relatively simple in these morphology examples, the occasions for significant mismatches of potential ‘core’ material are considerably more numerous in the grammar resources proper. Partitions need to be defined as required by the congruencies found necessary for particular language descriptions and this does not divide naturally into ‘core’ and ‘non-core’. Even within traditional sets of ‘typologically-related’ languages (e.g., English, German, French, Dutch), some languages will be more closely related than others and, moreover, the closeness of the relationship will vary depending on what part of the linguistic system one is examining. This leads to one of the most important problems with the ‘core grammar’ approach from the perspective of engineering a practically useful multilinguality: as more languages are considered, less and less useful information is left remaining in the so-called core. This reduces the support for expressing congruence. In contrast to this, multilingual system networks move beyond any ‘hardcoded’ common-noncommon placement of multilinguality such as is involved in a a core-noncore division, a theory-description split, or the selection of an interlingua. They instead provide a formulation of appropriate dimensions of description of linguistic information that maximizes the opportunities for capturing congruences across languages wherever these might occur. This is a precondition for reducing the cost and time of development for multilingual language descriptions to maximal effect.
AN EXAMPLE OF MULTILINGUAL TACTICAL GENERATION: INTRINSIC MULTILINGUALITY IN ACTION

Having now introduced our theoretical framework for multilingual descriptions, we turn in this and the following section to the MLG functionality that such resources support. We have pursued two main implementation directions: first, providing an extensive grammar development environment and generation engine that supports large-scale multilingual development (KPML: Bateman, 1997), and second, producing a testbed for further expansion of the theoretical principles of multilingual description and their implementation, including extensions for multimodality and more sophisticated inferential use of the information inherent in a multilingual description (MULTEX: Zeng, 1995; Zeng, 1996). KPML is currently implemented in ANSI-standard Common LISP, making extensive use of the Common Lisp Interface Manager (CLIM), and runs under Unix and Windows; MULTEX is implemented in JAVA, indicative of the fact that we expect significant increases both in the presence of NLG technology on the World-Wide Web and in the use of web-style access for much information delivery.

Our example of tactical generation focuses on the process of generation and the kinds of cross-linguistic variability that are readily supported. As we indicated in Section 2, a multilingual tactical generator expects as input a specification of the informational content of the message it is to produce. This therefore provides the starting point for our discussion. For concreteness, we will describe the generation process from the perspective of the KPML system, although similar constructs can be located in most systemically based generators. The generation algorithm is essentially that of the Penman monolingual text generation system (cf. Mann and Matthiessen, 1985; Matthiessen and Bateman, 1991), on which KPML is based.

The input specification for tactical generation expected by KPML is our multilingual extension of the Penman ‘Sentence Plan Language’ (SPL: Kasper, 1989). Considerable care was taken in the original design of SPL to make it as ‘grammar-free’ as possible. This ensures that an application program does not need to know details of the grammatical treatments of individual languages. Normally SPL input specifications are produced automatically by the text planning and information-extraction ‘strategic’ phase of generation before being passed to the tactical generator. However, it is also conceivable that such input specifications (or, more typically, input specification fragments) be authored directly and used as complex semantic templates; such an approach to generation has been pursued for example by DiMarco, Hirst, Wanner and Wilkinson (1995).

SPL expressions consist of lists of typed semantic variables and the relations (slots) that these variables enter into:

(var-0 / type :relation1 var1 :relation2 :var2 ...)

Most SPL relations provide information about semantic roles, circumstantial information (time, place, etc.) and so on, although some also refer to aspects of the textual (e.g., :theme) or interactional (e.g., :speechact) meanings desired for the message: these latter become particularly important when sentences are to be generated in sequence as part of a coherent text. Semantic types are generally drawn from a knowledge base model of a domain and are therefore given by the application rather than the generator. In order that the generator can interpret such application concepts, however, they are typically subordinated to a general multilingually validated hierarchy, called the Upper Model.* Linguistic resources used with KPML are defined so that they provide methods of expression for each Upper Model concept. Domain concepts that are subordinated to Upper Model concepts therefore inherit possibilities for linguistic expression from their superconcepts.

Figure 8 shows a typical SPL for sentence generation; this is the particular input required to generate the example software message used in the previous sections. In the figure, examples of semantic variables are rel, s, c0, cl, etc., while rst-means, manner, start, software-command, etc. are semantic

---

*The Upper Model originated in work by Halliday, Mann and Matthiessen in 1984/5. It is still undergoing development and extension as new languages are considered (cf. Bateman, Henschel and Rinaldi, 1995; Halliday and Matthiessen, in press).
types and \texttt{:class-ascription c1 or :name polyline} are examples of relations (the domain of the relation is the immediately dominating semantic variable, the range of the relation is the immediately dominated variable or element). As appropriate for the sentences this SPL should produce, the semantic types in the figure are primarily selected from a CADCAM application domain.

Generation begins when such an SPL specification is passed to the tactical generator. This involves providing the generator with a pointer to the SPL variable ‘\texttt{rel}’, from which all the other information given in the SPL can be interrogated as needed, and ‘traversing’ the system network that defines the grammar (cf. Figure 2 above). Traversal of the system network triggers a set of semantics-grammar mappings that attempt to classify the given SPL object (i.e., ‘\texttt{rel}’) according to a grammatical description appropriate for its semantic type. Semantics-grammar mappings (the so-called ‘chooser and inquiries’ of a Penman-style architecture such as KPML) often consider several distinct aspects of the meaning to be expressed while making their decisions: including the textual status of the information to be expressed, as this is relevant for pronominalization, ellipsis, word-ordering phenomena, etc., and interpersonal information, as this is relevant for speech act force, etc. Each step of grammatical classification—i.e., each selection of a grammatical feature from the system network—leads to a further set of semantics-grammar mappings, providing an increasingly fine classification. In the present case, the semantic type of the variable ‘\texttt{rel}’ is a conjunction of ‘\texttt{rst-means}’ and ‘\texttt{manner}’; these are both general concepts taken directly from the Upper Model. This leads the semantics-grammar mappings to opt for realization as a grammatical clause, and more specifically, as a clause involving a combination of two further clauses as subconstituents: one representing an end-state, and another representing the means by which that end-state is achieved. Traversal finishes when no further paths through the network compatible with the input specification are found; by this time, a complete grammar will have produced a complete grammatical fragment for that specification.

The semantics-grammar mappings activated during traversal also establish associations between grammatical constituents and components of the SPL input specification. Thus, in this case, the end-state clause becomes associated with the SPL variable ‘\texttt{s}’ and the means clause with the variable ‘\texttt{c}’. The generation
process then recurses and starts a new cycle through the system network for each of these grammatical constituents. And, whereas on the first traversal of the grammar the semantics-grammar mappings were ‘looking at’ the variable ‘rel’, they now consider instead the semantic variable relevant for their respective grammatical constituents. When generating the end-state clause, for example, the semantic type and associated semantic roles of the variable ‘s’ will be used for driving classification; this then results in a grammatical unit being generated that is appropriate both for the semantics specified by ‘s’ and for the grammatical context of that end-state clause as a subconstituent of the larger clause realizing ‘rel’. The semantic type of ‘s’ is given in the SPL as start, which is a domain concept that must be linked to an appropriate Upper Model concept so that the semantics-grammar mappings can classify it. For the present domain and example, start is a concept that is appropriately realized by a clause involving both a doer (a grammatical ‘Actor’) of the action (a grammatical ‘Process’) and something to which the action is done (a grammatical ‘Goal’): i.e., ‘the user (:Actor) starts (:Process) the command (:Goal)’; however the SPL input specifies that the communicative force of the utterance is imperative and this allows the ‘Actor’ to be omitted. The grammatical structure generated thus far is then:

<table>
<thead>
<tr>
<th>Clause (end)</th>
<th>Clause (means)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>Goal</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Similar fragments can be found in both of the Chinese and English functional structures shown in Figure 3 above. Cross-language variations arise because the particular grammar network followed and semantics-grammar mappings applied are given by the current view that is adopted of the multilingual resources loaded. ‘Development-time’ multilinguality is thus generally reduced to ‘generation-time’ monolinguality; and, since the views in force during the generation of any single sentence are—from the perspective of the generation algorithm—indepen
dent, it is natural that the structures produced from a single input specification can be widely divergent.

If, for example, the current view requires the means-clause to be ordered before the end-clause (i.e., a conditionalized realization statement such as [Means\^End] applies: cf. Figure 4) then structures appropriate for the Chinese and Spanish example versions of our message are generated; conversely, if the active language view has the realization [End\^Means], then structures compatible with our English and Russian examples are generated. Similarly, considering the generation of the grammatical structures for ‘on the toolbar’ (from the semantics given in the SPL, i.e., ‘c2 :spatial-locating c4’), when a realization [Minorprocess\^Minirange] appears, structures appropriate for English, Russian and Spanish are generated, while a realization statement [Minirange\^Minorprocess] generates structures for Chinese (cf. Figure 3). These differences extend down to the selection of particular lexical items—both closed-class items that are inserted directly by the grammar, and open-class items that are more commonly associated with particular domain concepts. Language-specific conditionlization can be specified for such items at the point of linkage with a domain concept, in the grammar’s selection of lexical features in realization statements, or even within individual ‘multilingual’ lexical items where all that might vary across views is the ‘spelling’.

More significantly, wherever there is a choice in generation, the individual grammar specifications make their selections as appropriate for the grammar and semantics of the language view being used for generation. For example, it is not obligatory to order the end-clause before the means-clause in English, nor means before end in Spanish: this needs to be ‘decided’ by the tactical generator. In general, this will depend on language-internal factors, which may or may not be partially congruent with the factors relevant in other languages; this congruence is maintained in the definition of the multilingual resources in order to ease development, but is not allowed to play any, possibly distracting, role during generation. Generated sentences are not then restricted to being literal ‘translations’ of one another; they must first respect the requirements of their own languages. Thus it is not necessary to ‘translate’ the English ‘choose from the Polyl ine flyout’ using a precisely equivalent (and wrong) Russian or Chinese prepositional phrase involving selection from. The Russian and Chinese language views are free to provide their own classifications of the
SPL input specification in order to produce their preferred 'choose in' -type expressions (i.e., *vybrav* *v*... and *xuanze* ... *zhong* respectively).

This kind of variation is captured straightforwardly in our account by the conditionalization that is allowed in the semantics-grammar mappings: these can opt to send the interpretation of a given SPL element down completely different paths of the grammar if language varieties require it. This phenomenon is in fact extremely pervasive multilingually (see also Delin, Scott and Hartley, 1996) and there several further cases in our example software message alone. It is therefore particularly important to avoid reduction in grammatical congruence simply because supposed 'translation equivalents' appear to be differently structured. There is a natural extension of this flexibility to involve considerations of style, even within single languages, but this is beyond the scope of the current paper. Important to note in conclusion here is that intrinsic multilinguality preserves congruence where possible even when faced with this degree of cross-language variation. Re-usability in the linguistic descriptions is thus combined with the generally claimed advantage of MLG that it is possible to generate more varied and natural 'translation equivalents' than would normally be considered possible with machine translation (cf., e.g., Vander Linden and Scott, 1995)

USE AND EVALUATION OF THE APPROACH: ISSUES

Evaluating NLG components is a difficult enterprise—primarily because there are still substantial differences between input representations, processing methods, and goals. Even restricting the area to applications of MLG, there are still difficulties in finding sufficiently comparable but nevertheless diverging systems that would support contrastive evaluation. Individual projects rarely have the resources to consider a range of approaches to MLG as potential or actual providers of their generation needs. Evaluations are still therefore mostly whole-system evaluations, where the particular approach taken to generation in general, or to multilinguality in particular, has not been at issue.

While rigorous contrastive evaluations of MLG and the particular approach to MLG that we have proposed in this paper therefore remain to be done, we can already point to some broad characteristics and results of the methodology and its implementations as used in a number of distinct generation projects and systems. KPLML, for example, is freely available, fully documented and has already been adopted in a number of text generation projects as either the resource development environment or generation engine or both; these projects include DRAFTER (Paris, Linden, Fischer, Hartley, Pemberton, Power and Scott, 1995) for English/French, GIST (Not and Stock, 1994) and HEALTHDOC (DiMarco et al., 1995) for English, TECHDOC (Rösser and Stede, 1994) for the resource development and maintenance of English/German, KOMET (Bateman and Teich, 1995) for experiments with English/Dutch/German/Greek/Japanese, AGILE (Teich, Steiner, Henschel and Bateman, 1997) for Czech/Russian/Bulgarian, and GUME (Aguado, Bañón, Bateman, Bernardos, Fernández, Gómez-Pérez, Nieto, Olalla, Plaza and Sánchez, 1998) for Spanish. In this section we use the experiences gained in these projects as a preliminary round of informal evaluation.

New Language Acquisition

There are two common scenarios in the use of MLG technology: first, generation is required in some language(s) for which there do not exist appropriate linguistic resources and those resources have to be developed, and second, generation is required in some language(s) where a generation capability does exist but that capability has first to be tailored to the requirements of a particular application. The two scenarios can of course combine. In this subsection we focus on the first scenario; the second scenario is addressed in the subsection following.

The general strategy of developing coverage for a new language by re-using components of existing languages is well established and is widely documented as bringing reductions in development times (cf., e.g., Lee, Okumura, Muraki and Kim, 1991; Okumura, Muraki and Akamine, 1991; Rayner et al., 1996). The
earliest attempt to use the same strategy with systemic–functional generation grammars was the TECHDOC project’s development of a German grammar on the basis of the English grammar of the Penman system; here a similar reduction in development time was found. However, the creation of new descriptions by reusing existing descriptions has actually proceeded by ‘code scavenging’ whereby the rules of one language are opportunistically edited to approximate another. Following this process, the newly acquired linguistic resource is an additional, monolingual description that is not formally related back to its origins. This is theoretically and practically less than ideal since the source descriptions might change—e.g., be updated or extended—and it would be beneficial if such changes could also be applied to the relevant components of the new descriptions created on the basis of those sources.

With our approach to intrinsic multilinguality, we have attempted to support this process of resource acquisition in a manner that is both theoretically and implementationally more well-founded. The explicit placement of multilinguality along the dimensions of linguistic description involved have allowed software tools to be developed that focus on precisely those areas of a linguistic description that are most likely to require changing when extending them to another language. At the same time, the implementation of the strong partitioning required for multilingual descriptions guarantees that a language developer can make such changes without endangering the integrity of existing resources. This scheme was used in a pilot experiment for Dutch (Degand, 1996) where it was found that the types of adaptions required were indeed of a quite restricted nature. Adaptions such as adding/removing realization statements for the new language description, growing more specific system network fragments, and closing off areas of the system network for the new description are now provided as basic operations for the language developer. Subsequent experiences with this methodology for both French and Spanish, as well as ongoing work for Czech, Russian and Bulgarian appear to support further its effectiveness. For the experimental systems so far targeted, usable tactical generation capabilities have been produced within six person-months for all of the languages concerned by developers previously unfamiliar with the system. The methodology and corresponding support tools are illustrated in more detail in Bateman (1997).

It also appears to be the case that the form of the resources constructed by this method are generally well-suited to subsequent extension for further applications—thereby encouraging re-use. Newly developed resources can also be freely ‘re-merged’ with the previously existing linguistic resources for closer contrastive work. Intrinsic multilinguality then improves on the raw code scavenging methodology in three ways: first, detailed software tools for such development is supported; second, the relatedness of source and target resources is preserved by integration; and third, the languages among which similarities may be recognized are extended beyond those related by traditional structural typology.

**Maintenance and Development**

Having acquired some set of linguistic resources, those resources nevertheless will require extension, correction, and customization to particular application needs. Language engineering tasks such as these pose considerable problems. It is therefore a justifiable engineering concern that a combination of linguistic descriptions from distinct languages, as is entailed by intrinsic multilinguality, might result in systems that are significantly more complex and difficult to maintain and extend than their simpler, non-combining counterparts (cf. Rössner, 1994, p.132). It appears, however, that this problem is largely avoided by the provision of visualization and generation tools that always respect both the language partitions defined in an intrinsically multilingual resource and the particular viewing demands specified by the language developer. We illustrated some of these aspects for the simple case of morphology in Section 3; the language developer only sees the combination of languages that is selected and can change those resources as required. No dependencies or congruences with other language views are allowed to influence generation.

The problem is thus ‘reduced’ to that of dealing with any large-scale set of monolingual resources, and certainly the main difficulties reported by developers to date when seeking to expand the coverage of some set of existing resources are: (i) ascertaining precisely what is already covered and what not, and (ii) locating precisely which components in the overall resources either are responsible or should be responsible for the generation behavior desired. We have found that learning to find their way around a
large resource set is the biggest hurdle to fast resource development that language developers new to the system face. Therefore, a variety of potential solutions to this problem are currently being investigated. Central to all of these is a reliance on the notion of ‘example sets’, or test suites as were used extensively in the Penman system for recording the current state of development of the English grammar. An example set consists of a set of example records, which together should indicate the coverage of a given resource set. A full example record consists of an input specification (typically SPL), the string(s) that should result when that specification is used for generation, the grammatical structure generated, and the full set of grammatical features selected from the grammar during the generation of that structure. The KPML development environment allows the language developer to inspect examples of any of the features of the grammar for which examples exist (both monolingually and multilingually if required); conversely, the language developer can inspect any example that exhibits grammatical phenomena similar to those to be covered in order to see exactly where in the linguistic resource the corresponding decisions were made.

This also appears to provide an effective strategy for organizing the resource development work involved in producing a specific generation capability for some application when that capability already exists in another language. First, the approximate targeted coverage of the desired resources is codified in terms of an example set for the existing resources that spans the kinds of constructions to be covered. Then, this example set serves as an index into the grammar of the new language showing likely places that will require alteration to customize generation both to the application and to the new language. The provision of such ‘guides’ speeds development time considerably since language developers do not need to locate the relevant parts of the grammar themselves.

Finally, when the language developer changes resources in the places the previously described strategies will have indicated, possible dependencies within the language partition being worked on must also be taken into account. A typical generation grammar in the systemic-functional form will consist of around 700 grammatical systems (Figure 2 contained only 7) and the connectivity between systems is sometimes complex. Tracking down the possible implications of any change, although not an issue multilingually due to language partitioning, is still very much of an issue for development within any one language view. Here the paradigmatic organization of a large-scale systemic-functional grammar typically shows a further formal property that can be beneficially employed for easing this problem. Early work on systemic descriptions of English (Halliday, 1978, p113) noted the emergence of ‘functional regions’: i.e., particular subareas of the grammar that are concerned with particular areas of meaning. Dependencies within functional regions tend to be far more numerous and complex than dependencies across functional regions and this therefore establishes a natural modularity within a large-scale grammar. Applying this to language development, changes are considered first in the context of their immediate region and subsequently following the dependencies between regions. This also provides a good basis for distributed grammar development since developers can announce that they are interested in the expressive potential of some functional region and work on this independently of parallel work on other functional regions. The dependencies between regions then define the required ‘input’/‘output’ behavior of the region.

Functional regions have now also been defined multilingually and specialized tools are provided for visualizing and manipulating a grammar according to the organization of its functional regions. A partial region connectivity graph for the English grammar is shown in Figure 9. This ‘meta-network’ also serves as a ‘menu’ for accessing further graphical views of the grammar network as well as selections from test suites illustrating use of the resources contained within a region. Dependencies between regions are thus clearly indicated and can be followed up during development. The extent to which these kind of large-scale views of linguistic resources can reduce development times is still being investigated. Experiences with the English grammar, however, which has by far the longest history of development of the languages with which we have been concerned, strongly suggest that continued scaling-up and adaption is rendered a relatively constrained and approachable task.
Resource Re-use

It now appears that the paradigmatic organization of a multilingual systemic-functional network can be significantly shared across languages. For example, a range of multilingual generation development tasks and experiments were carried out by the KOMET project between 1992 and 1997; in particular, generation capabilities for six languages (Greek, English, German, Dutch, French, and Japanese) were developed for several trial research prototypes and demonstrators. If the monolingual views of these six languages are considered in isolation from one another, then the total grammar size (in May 1997) was 4537 grammatical systems. When the same information is considered multilingually, as an integrated intrinsically multilingual grammar, then there are only 1470 systems, indicating a high degree of congruence across descriptions. More details of these figures are given in Bateman (1997, p46).

Generation Performance

The generation algorithm used by KPML has the practical advantage of being deterministic as Reiter (1994) and others have since recommended for practical NLG systems. Its speed during generation is however conditioned entirely by the efficiency of the LISP that is used as implementation platform. This can vary dramatically from provider to provider and, more significantly, from hardware platform to hardware platform. Some example generation time statistics are shown in Table 1, showing the generation times for sentences ranging in length from 10 to 60 words generated on two machine types. Generation with KPML is still currently most commonly performed with the full KPML development environment, which incurs significant overheads because of the extensive debugging and maintenance tools resident. It is a design feature of the KPML environment that information is maintained for each grammatical unit generated—information not just about the results of generation, but about all details of how that result was achieved.
This is ideal for debugging/development but is not appropriate for final application behavior. It is also clear from the table that the PC LISP implementations used so far are incurring significant degradations in performance for longer sentences (e.g., 40–50 words), presumably brought about by excessive garbage collection brought about by LISP’s large memory allocation. This degradation is not apparent for the UNIX LISP implementation. Until such garbage collection overheads occur, generation times remain for both platforms approximately linear with respect to sentence length.

While sufficient for demonstrations and prototypes, performance will clearly need to be improved on the PC platform for more serious applications. This may be achieved by providing a speed-efficient reimplementation of the core generation algorithm, which is straightforward. A recent example of the effectiveness of such a solution is provided by the monolingual English tactical generator REAL PRO (Lavoie and Rambow, 1997). REAL PRO provides a highly speed-efficient tactical generator written in C++ for one component of a generation framework that has been used successfully in several important MLG systems; it is capable of generating 40–50 word sentences in less than a second on a 150MHz PC. Similar generation times are now being achieved with the JAVA implementation of our MULTEXT system. Therefore, for the future, we will probably separate our development environment from the generation component that is to be used in applications, allowing each to support those aspects of intrinsic multilinguality that are most useful for the differing tasks involved.

### Summary

We have selected several areas of experience that have arisen in applications of our framework in a variety of MLG systems. There are, of course, many more that should be discussed but space precludes this here. In general, however, we believe that the results obtained from applications of our framework so far are encouraging. They are still no substitute for proper evaluation however. Securing resources for such evaluation must therefore be seen as one of the most important areas where work is now necessary. However, undertaking such detailed and application-specific evaluations can only be done in collaboration with consumers of the technology we describe here.

### CONCLUSIONS AND DIRECTIONS

In this paper, we have focused on the provision of multilingual generation and the kind of architecture that can best support the diverse needs of practical applications, including particularly the generation of software diagnostic messages, online help or online documentation. We have argued that to achieve effective MLG it is necessary to determine a view of multilinguality that is maximally supportive of the re-use of linguistic descriptions. The effort involved in constructing particular language descriptions is very high, and such re-usability offers considerable potential savings. We showed in Section 2, however, that the representation of ‘multilinguality’ in the vast majority of MLG systems was still overly narrow: it did not support the enterprise of MLG as it should. In contrast to this, we have presented an implemented framework that makes cross-language congruence practically usable for supporting MLG development. This allows us to establish a representation that explicitly captures substantive similarities in language descriptions. In trial applications of the framework to date, significant reductions in development times for sophisticated grammars, as well as an easing of the problems of large-scale resource maintenance, appear to have been

<table>
<thead>
<tr>
<th>sentence length (words)</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
<th>55</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (in seconds)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sparc (Ultra, 64Mb RAM):</td>
<td>0.17</td>
<td>0.25</td>
<td>0.35</td>
<td>0.46</td>
<td>0.57</td>
<td>0.71</td>
<td>0.83</td>
<td>0.98</td>
<td>1.15</td>
<td>1.33</td>
<td>1.52</td>
</tr>
<tr>
<td>PC (200MHz, 64Mb RAM):</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>13</td>
<td>18</td>
<td>23</td>
<td>30</td>
<td>36</td>
<td>48</td>
<td>55</td>
</tr>
</tbody>
</table>

Table 1: Example generation time statistics with the current KPML implementation
achieved.

The move to more flexibility and quantity of software documentation is leading to more opportunities for exploiting NLG technology. Areas which support this development include possible NLG solutions to the problems of maintaining consistency across and within subsequent versions of documentation, cost-effective provision of differential styles of documentation that are dependent on user, task and application situation, and user-accessible presentations of information maintained by knowledge-intensive software. Providing context-specific or user-specific styles of documentation of such diversity across all the languages where such documentation is to be provided will severely overload any approach to software localization that maintains catalogues of canned messages or templates. The range and flexibility of those messages combined with the fact that many of their components will nevertheless be broadly similar guarantees that any such solution would be extremely inefficient. For applications of this kind, the overhead of not considering MLG technology will itself start to assume significant proportions.

Acknowledgements: Mattheissen and Zeng gratefully acknowledge the financial support of the Australian Research Council for the work reported in this paper. We would also like to thank the following who have worked on grammar development within the computational framework we have been developing and have provided some of the examples used in this paper: Guadalupe Aguado, Alberto Bañón, Esperanza Nieto, Serge Sharoff and Lena Sokolova.

REFERENCES


URL: http://www.ercim.inria.fr


URL: http://www.darmstadt.gmd.de/publish/komet/gen-um/newUM.html


Kasper, R. T. 1989. A flexible interface for linking applications to PENMAN’s sentence generator, Proceedings of the DARPA Workshop on Speech and Natural Language. Available from USC/Information Sciences Institute, Marina del Rey, CA.


