

# Designing a Multi-Purpose CL Application

Ursula Reuther, Antje Schmidt-Wigger  
Institute of Applied Information Sciences (IAI)  
Martin-Luther-Str.14  
D-66111 Saarbrücken  
{ursel;antje}@iai.uni-sb.de  
+49-681-38951-{28;29}

## Abstract

CL development and especially the development of tools for CL convergence checking is a time and cost consuming task. The development of flexible CL checkers by reusing existing resources is a great challenge for the CL community today. The article presents some theoretical reflections and lessons learnt in ongoing CL projects on the subject of customisation and parametrisation, trying to show the best ways to achieve ease of handling and maximum reusability of software and lingware modules. The sub-modules are identified, together with a description of their general components and the limitations necessary for specific user settings.

## 1 Backdrop

Although Controlled Language (CL) was considered monolithic some years ago (cf. [Douglas 1996]), today's understanding of CL is that it "is not a single, immutable entity" and that it "has taken form in different applications and for different purposes." (cf. [Allen 1999]).

In this light, flexibility and portability are essential for CL applications. The idea of "one tool – many users" is above all relevant from a user's point of view: he wants to have a customised CL solution tailored for his own special purpose and needs – with the option of easily adapting or expanding when his situation changes. But from the developer's point of view, too, this aspect has to be given high priority when designing a multi-purpose CL – only such a basis allows for easy and cheap reuse of existing resources in a new business case.

Efforts toward extending an already existing standard in CL are described in [Wojcik et al. 1998] where AECMA SE was used as the basis to develop a general purpose CL (BTE - Boeing Technical English) for a variety of Boeing technical documents. In [Lalaude et al. 1998] a modularisation of CL rules is described aiming at the creation of various rule sets to be applied for different document types.

Both initiatives could be considered a kind of customisation: in the former case, an existing CL was adjusted to meet the requirements of a broader application domain, and in the latter case, an existing CL was adapted to differentiate between various document types. It is exactly these kinds of activities that form part of customer requirements when a company wants to implement

its individual CL application in its document production workflow, and they have to be responded to by CL tool developers.

Although there exist ready-to-use solutions for established standards (e.g. MAXit for AECMA SE), most providers of CL applications offer their tools on a project-service basis where user requirements are implemented when they arise. In this article, we would like

- to sketch different types of user requirements by investigating possible implementation scenarios,
- to identify necessary parameter settings, and
- to define the potential basis for a user-driven parametrisation of a core CL implementation.

## 2 Customisation Scenarios

In the following, various scenarios of “customised” CL application variants or of company-internal variants (a softer variant of customisation) are sketched out and the resulting parameter requirements are listed.

- *Scenario 1:* Same company, different user profiles (e.g. technical writer vs. editor)

The technical writer needs more guidance and more explanations at the linguistic level than the editor does. The editor defines the rules and needs to identify errors quickly; the technical writer has to apply the rules and should be supported in the learning process. Therefore the manner of **presenting and explaining the errors** must be different and thus parametrisable.

The editor needs, apart from a quick, reliable checking tool, information about e.g. errors which remain uncorrected although they have been reported to the user as well as about creation and use of new (ad hoc) terms and syntactic structures in order to evaluate the applicability and appropriateness of the CL and to modify it if necessary. This function is irrelevant for the technical writer. Thus, the **access to document-related information** must be offered as an additional function.

- *Scenario 2:* Same company, different types of documentation for different target groups (e.g. customer handbooks vs. repair manuals vs. parts lists in reduced text format)

The CL application should offer the possibility of loading and triggering the respective lexical and syntactic rules according to the document type in question. However, at least a common core lexicon is probably necessary to ensure harmonisation at company level. Thus, there must be the possibility of parametrising **the relevant linguistic data and resources**.

- *Scenario 3:* Different companies, same industrial sector and domain (e.g. parts supplier for the automotive industry vs. large manufacturing company in the automotive sector)

In the co-operation of a large company and its suppliers, documents accompany the parts delivered, and textual elements as well as part names are taken over and re-used in the final documentation. To ensure compatibility, a subset of the checking tools available in the document management system (DMS) of the large company is thus relevant for the supplier (e.g. terminology and abbreviation checking). Therefore a “flattened” version of the CL application should be available, i.e. **the inclusion of the functionalities** should be parametrisable.

- *Scenario 4:* Different companies, different industrial sector, but business partners (e.g. a publishing company producing and/or translating various types of documentation for the automotive industry vs. large company in the automotive sector)

Within this scenario, it might well be the case that additional or different features need to be checked during the QA process at the service provider's site. Therefore, although sharing the domain and the content of the documents, the CL application should allow for additional checking functions (e.g. extension to checkers for the target languages, cross-project checking, etc.), i.e. **type and combination of the CL functions** should be parametrisable.

- *Scenario 5:* Different companies from different industrial sectors with no business relation, same type of documentation

The requirement to be derived from this scenario is a modular and open design of all relevant components to fit in with each company's DMS in terms of content and in terms of format and technical requirements. The architecture of the CL checking facilities should be modular, with clear delimitation of language dependent and workflow dependent components to allow maximum reusability of the CL checker, i.e. **the development environment** must be flexible and thus parametrisable.

- *Scenario 6:* Different companies, same industrial sector

Competing companies could decide to unify a part of their document production processes to solve complex problems through common efforts. This is mainly a goal in terminological standardisation initiatives (e.g. SAEJ 1930). Foreign companies, too, can profit from an existing standard in their translation activities. As a requirement, these standardisation efforts should be reflected in the CL architecture e.g. through clear distinctions between **standardised and customer specific** terminology.

The emerging question now is: which, how many and how complex parameter settings should be left to the user (be it the individual author or the company as a whole), and which parameters should remain in the responsibility of the developer or the support service. In any case, each parameter setting should be reflected in an adequate modularisation of the application to allow for a user driven choice.

At IAI we are currently investigating this topic, since we have to cope with all of the above mentioned scenarios:

- Scenario 1–4 in a research project which aims at 'transferring' the CL application used by technical authors of repair instructions at BMW (cf. [Reuther 1998]) to other departments and companies (cf. section 4 for more details on the project),
- Scenario 5 within a feasibility study carried out for Siemens Information and Communication Networks for their operation and maintenance documentation of telecommunications equipment and systems (the only common denominator with respect to the initially treated documents being the German language) and
- Scenario 6 in the MULTIDOC project (cf. [Schütz 1998]) where CL is seen as one contribution to the improvement of the documentation workflow in the European automotive industry.

### 3 Approaches to Parametrisation

In any case and in all of the scenarios, we have to consider customisation both on the functional level and on the content level.

The **functional level** (programming and software issues) includes the choice of a basic programming platform and the integration of the CL application in the editor used for processing the documents at the customer's site. Other functional features are a user interface which allows the user to choose among various options as well as functions which are available only for certain users (e.g. statistics on usage, information about deliberately ignored error marks) or for certain user profiles (e.g. terminologists).

As for the portability of a piece of software to different platforms and interfacing different adjacent applications, the problem is not at all a CL specific one. To be successful on the market, porting has become a necessity, and a large range of filters is a minimum for the acceptability of a single product. Engineering methods such as modularisation, use of platform independent programming languages etc. try to respond to that problem. But they are not intended to be the subject of the present article.

The **content level** (lingware issues) involves two dimensions:

- linguistic content: vocabulary, style and grammar rules, and
- didactic content: how the error is shown and explained to the user.

As argued above, both linguistic and didactic content should be subject to parametrisation. In the following, we would like to explore the possibilities of how to parametrise the relevant types of information and related data on the content level.

Bearing in mind the requirements to be taken into account when developing customised variants of CL applications, the following approaches toward realising and implementing user-specific CL tool sets are viable:

#### 3.1 The single modules approach

In this approach, all components of a single CL application are kept separately from other ones. This implies that the complete lingware relevant for the customer in question has to be maintained and improved in isolation, even if it originates from a related lingware module. Thus, progress achieved for one customer is not available for other ones. This approach is not very developer friendly nor is it economic. Therefore, this (admittedly rather unrealistic) approach will not be discussed further. Suffice it to say that the single modules approach reflects the current market situation where different CL applications model different CL specifications for the same or different languages, only learning from each other through their respective publications.

#### 3.2 The satellite approach

Within this approach one linguistic core module and various sub-modules are assumed. The sub-modules include those lingware components which are specific to a certain application or to a certain user. A similar architecture is assumed in [Lalaude et. al 1998], where generic and specific CL rules are distinguished and where the specific ones are grouped in sets each of which is linked to particular document types. Here, the sub-modules or satellites represent the

customer-specific linguistic data which, in combination with the core module, result in the application tailored exactly to the customer's needs.

The core module should include those linguistic data which are valid in any application and which are shared by all users, i.e. general vocabulary, rules checking the spelling and the grammatical correctness and those style rules which can be said to hold for any CL application.<sup>1</sup> The sub-modules then include customer-specific terminology and 'corporate writing rules' plus a user-defined exception handling, when generally valid standards included in the core module have to be violated (cf. section 5.1).

This approach clearly has an advantage in comparison with the former one, because the core module, once defined and implemented, has to be maintained only once for all possible application scenarios. However, the more customers come into play, the more efforts have to be made to design and to maintain the various sub-modules. Furthermore, there exists the risk that the same work may be done twice in different sub-modules, either because they are developed in parallel according to specific user requirements, or, and this seems to be more likely, because more than one user has the same requirements which are, however, not shared by all users and therefore not included in the core module.

An argument sometimes put forward against a common core rule set is the strong competition among companies which are not willing to share "corporate style rules" with other companies. But in many cases the "uniqueness" of the rules is only a matter of presentation and corporate style; the underlying "writing rule", however, follows standard recommendations of technical writing and thus supports the approach of a common core module, at least from the developer's point of view.

### **3.3 The single source approach**

In what we call the "single source approach" (a term borrowed from technical writing), one single source of linguistic data (including grammar and style rules as well as ontological information) is assumed, from which all customer relevant data are compiled automatically and then stored in the specific application.

The advantage of this approach clearly lies in its efficiency as far as maintenance and further development is concerned. An additional advantage from a user's point of view is that new customers may, if they want, benefit from already existing requirements implemented for former customers. Similar lingware developments in parallel are thus avoided and synergies of different applications result in an optimisation the lingware components compiled and maintained within one single source. A similar approach is described in [Knops&Depoortere 1998], where a rule library with each rule being tagged with the appropriate customer label is the source for various applications.

Changing one customer's application profile in the above approach does not mean manipulating the linguistic data, but simply adapting the "shopping list" of rules, which is less risky and simpler. This way of generating "control on demand" is of course interesting for the developer, but the customer, too, might be interested in simply changing certain parameters in his application. For instance, suppression or adding of rules, changing numerical values within rules,

---

<sup>1</sup> Assuming that a CL is applied only in technical documentation, there exist style rules which belong to the general principles of technical writing and which therefore are to be included in the core module. If this assumption were abandoned, it might be difficult to define such generic style rules.

etc. might be an important and necessary feature of the application. Thus, this approach qualifies best for such a user-driven definition of a CL application.

## **4 The TETRIS Project**

The TETRIS project, which is funded by the German Federal Ministry of Economics and Technology (grant number VI B 4-00 30 48/9), is the follow-up project to MULTILINT (cf. [Reuther&Schmidt-Wigger 1998]) and aims at developing strategies and approaches for technology transfer in the domain of applied language technologies exemplified by the case of a CL application for German.

One of the project's main objectives is to "transfer" and to customise the existing pilot application of the MULTILINT prototype, used by technical authors in the After Sales department of BMW, for other companies, especially SMEs, in order to exploit the know-how and the linguistic resources of the MULTILINT system for other CL applications of German. Furthermore, the project aims at ensuring compatibility between documents produced in the different production chains the project partners are involved in.

The TETRIS project consortium consists of IAI, BMW and five new user companies. The companies already work together in close co-operation: BMW as the manufacturing company, plus a translation provider, a document publisher and distributor and two service providers in the domain of technical documentation for back-fitting and for customer information material. Their common activities cover all aspects of linguistic production, thus representing a solid basis for the validation of a "distributed concept" for CL.

It is a further main objective of TETRIS to investigate this distributed concept of CL, i.e. to investigate to what extent standards for a German CL can be derived from an application scenario where different use cases and different requirements come into play. In a first step, existing style rules available in the companies as well as rules extracted from relevant teaching material will be classified and compared, and common rules will be identified. This will be done in close co-operation with the project partners involved as well as with members of a working group of "tekomp", the German professional association for technical writing. The expected outcome will then consist of guidelines for a standardised CL for German which will be validated within the project.

To explore the kinds of parameters necessary for the efficient use of a common checking facility by different users, the present implementation of the MULTILINT tools will in the next chapter be taken as the basis for investigating the reasons for and the implications of parametrisation.

## **5 Parameters of the MULTILINT Checking Tools**

The MULTILINT toolbox (cf. [Schmidt-Wigger 1998]) comprises six core checking facilities which tackle the three different levels of limitation a Controlled Language has to cope with:

- general language correctness,
- lexical limitations, and
- syntactic limitations.

Each checking facility can be activated separately and relies on specifically defined resources. One general linguistic core component, however, creates the basis for all these checkers, i.e. a general morphological analyser plus a language specific morpheme lexicon.

### **5.1 General language correctness (spelling and grammar rules of German)**

A successful CL checker should include components for general language checking to ensure an overall quality of the documents produced. In fact, style checking components could run into difficulties if they are confronted with grammatically ill-formed input (cf. [Adriaens 1994]).

The definition of grammatical correctness should usually be common for all applications with the same source language. In most linguistic communities, the rules are defined by a central institution and a CL should not stand in contradiction to it.

But Corporate Language tends to be stronger in limitation in cases where general language rules allow for different writing possibilities. One prominent example is the case of the German Spelling Reform: it introduces in many cases alternative variants, whether in order to allow the older generation to continue with their habits, or whether in the case of rules too complicated to be followed. A Corporate Language should define its own preferences, in lexical and in grammatical matters.

In addition, some exceptions will always be found when Corporate Language comes into play. The most important aspect for German is the influence of English terms and orthography (e.g. 'English' separate compounding). The internationalisation of the market opens the door to English through American based industries' specific vocabulary, brand names<sup>2</sup>, and through the positive connotation English forms have acquired, at least in German. Thus, it is more and more the case that company specific terms or grammar rules violate German standard grammar. For example, at the lexical level, the noun *roadster* in BMW documents must start with a lower-case letter, which is against German spelling rules. Further grammar violations can be found in the field of compounding, where e.g. *BMW* is allowed only without a hyphen in compounds.

The above deviations from standard grammar have to be tackled in different ways, according to where the problematic structure is treated: either in the lexicon or in the grammar modules.

The MULTILINT spell checker relies on a complete morphemic lexicon in order to give a tagged basis to the checking facilities. Thus, unusual variant spellings are added in a user specific lexicon, together with proper names and brand names, and the standard spelling is filtered out only after its successful recognition via the general language analysis. Separate compounding, however, can only be recognised via the grammatical component, because it involves the treatment of a list of (graphemically defined) words.

The legalising of ungrammatical structures or, in other words, the parametrising of grammar checking, is easier from a procedural point of view, as this component is not based on a positive, but on a negative description of grammar conventions. Rules which could identify structures as ungrammatical for general language are simply not included in the rule set for a specific text type or for a specific Corporate Language (e.g. compound splitting), whereas stronger limitations can easily be added to the rule set.

The creation and maintenance of the user specific lexicon for spell check parametrisation can be performed by the user her-/himself, as long as this is done centrally and in co-ordination with all

---

<sup>2</sup> E.g. sometimes legal reasons forbid the use of a brand name – even if hyphenated – where it is incorporated in a German compound. The compound should therefore be circumvented by a prepositional phrase construction or – more easily – the compound is simply written in separate parts.

parties involved. At the grammar level, however, it is more difficult for the user to take action, because to him, the organisation and the interrelations of grammar rules might not be obvious. Excluding specific rules from the existing rule set should be feasible; adding new ones would require the involvement of a trained linguist.

## **5.2 Lexical limitations (terminology and consistency)**

Terminology checking relies on user defined terminological databases. It is thus mainly user driven. In Scenario 2 and 6, however, it could be interesting to have one common core terminology representing the standardised part of terminology at (inter-)national or company level, together with the possibility to add, reduce and replace peripheral items at will. For both scenarios, a “trace back” function should be available.

Terminology and consistency checking, in our approach, uses – besides general language specific morphosyntactic information – relations between lexical units which are derived from an ontological structure. Thus, scenario 5 includes not only a different user specific terminology, but also domain knowledge, i.e. the linguistic realisations of the concepts have to be adapted, and/or defined from scratch, for a useful term and consistency checking.

A further parameter to be taken into account in the domain of terminology is the access to terminological information. In an application targeted toward supporting a terminologist, it is important, among other things,

- to make available to him new term creations by the authors as possible term candidates,
- to enable him to extract relevant terminology from existing data material,
- to allow him to collect (deliberate) violations with respect to the official terminology,
- to use specific programmes for consistency checks, e.g. among different databases, etc.

For the technical author, on the other hand, a simple check against the corporate terminology is sufficient.

## **5.3 Syntactic limitations (style)**

Style checking involves different viewpoints. The basic viewpoint should be the ease of reading of a text, which means that the purpose a text is written for should be grasped quickly and without any misunderstanding. Additional requirements are concerned with the ease of retrieval and the ease of translation of the text.

Thus, three general types of weaknesses can be identified via stylistic considerations: ambiguity, complexity and redundancy. Ambiguity and redundancy relate to the unequivocal relation between meaning and text, while complexity is a hindrance to quick understanding and thus a possible source of misunderstandings.

Concerning the parametrisation of style rules, most freedom should be given with redundant structures or words, i.e. the choice of the adequate expression of a semantico-pragmatic content should follow user preferences (e.g. instructive sentences may be expressed by imperatives, infinitives, declaratives, modal verbs, etc.).

Other style variants are text type driven, which means that the intention of a text has to be reflected (i.e. parts lists being short and concise, repair instructions being instructive and unambiguous, etc.) as well as the user’s expectations about adequate style (i.e. customer documentation being of eloquent style).

To create variable sets of text type adequate and specific style rules, the developers have to take into account possible choices of relevant limitations, relying on psycholinguistic studies of readability and understandability or on linguistic analyses for identifying ambiguous cases.

Another possibility to achieve this goal is corpus based work. Traditionally used structures are declared as preferred structures, and the developer's task is to identify concurrent structures through style rules. This corpus work, again, should be done for each text type separately, but company specific stylistic preferences also have to be identified and taken into account.

In some companies, there already exists a company-internal Style Guide or at least some writing rules which have evolved on the basis of the same considerations above: standardising the existing documentation and avoiding future "stylistic trouble" in the domain of technical writing.

In practice, a combination of all these approaches is followed. As for now, new incoming texts and text types are checked against the complete rule set, thus identifying problem areas and being able to propose more or less strict corrections and changes.

In any case, the customer (in our context, this target group includes the technical writer as well as the company as a whole or sub-units) must have the possibility to set stylistic parameters reflecting the document type, and in combination with this option, he should be able to distinguish between company internal and external documentation.

#### **5.4 Error messages (didactic content)**

Horizontal to the above three categories of CL limitations, there is a further level for which parametrisation is of importance. The error messages displayed to the user and explaining the kind of error must also fulfil the requirement of being flexible. Therefore, various versions of error messages have been developed ranging from linguistically oriented ones via short, simple, and/or pedagogical ones through to one version especially made for technical writers with no linguistic background. The customer may include more than one version in his application, thus offering alternative versions to the end user which can be changed 'on the fly'. Furthermore, there is the option of displaying the error messages in another language than the document is written in: this is becoming more and more important, since many globally operating companies based in Germany (e.g. Siemens, Heidelberger Druckmaschinen) produce their documentation in parallel, both in English and German. Thus, the requirements placed upon the authors are the same as in the aeronautics industry, where all documents should be written in English, regardless of the author's mother tongue.

## **6 Preliminary Results**

As far as customisation through **parameter settings** is concerned, the following observations can be made for the above mentioned categories:

- as for grammatical checking, there is a need for defining exceptions both on a lexical and on a syntactic level,
- as for terminology and consistency checking, domain specific knowledge must be stored in a way such that it can be exploited efficiently by parameter settings, and the access to information must also allow for parametrisation,
- as for the checking at stylistic level, parameters must be foreseen for different text types and for different target groups,

- and, finally, at the level of the “social interface”, i.e. at the level of explanations, various parameters must be taken into account as well.

The single source approach is the most convenient one in order to realise this high degree of parametrisation.

Three groups involved in the CL definition process qualify for **parameter responsibility**: the developer, the company as the customer (through a centralised institution) and the technical writer as the individual end user.

- Regarding the goal of Controlled Language use, the company should play the key role in the parametrisation of the CL tool to ensure document quality and consistency in its overall workflow.
- The single user, i.e. the author, should only be able to choose among convenience features such as message type for himself; the choice of text type specific rules should be done automatically, i.e. by SGML marking of the document.
- The developer should take care of the parametrisation when general principles or standards are involved. Thus he could identify them and suggest them as a unified component to all his customers.

## 7 Conclusion

In this article, we have elaborated on the reasons for and on the requirements of multi-purpose CL development, investigating different general approaches with respect to different scenarios. The single source approach turned out to be the most promising one in terms of flexibility and ease of handling.

The approach has been realised in different projects, leading to insights into necessary and adequate parameters to be offered to the user in the customised CL tool.

Upcoming initiatives such as NCCAT (National Consortium on Controlled Language and Computer-Aided Translation) aim at bringing together different players in the field of CL, sharing experiences in the development and in the application of CLs and creating cross-industry standards in this field. In this light, IAI's current research and development activities in customising CL applications and the findings in the TETRIS project can be considered as one important step toward these goals.

## 8 References

Adriaens, G., 1994, Simplified English Grammar and Style Correction in an MT Framework: The LRE SECC Project, in: Proceedings of Aslib 1995.

Allen, J., 1999, Different Types of Controlled Languages, in: TC-Forum 1-99.

Douglas, S., 1996, Sublanguage in the sky, in: Annual Report 1995-96 of the Human Communication Research Centre (HCRC),  
URL: <http://www.hcrc.ed.ac.uk/AnnualReport96/Text/5.4-www.html>.

Knops, U., Depoortere, B., 1998, Controlled Language and Machine Translation, in: Proceedings of CLAW 98.

- Lalaude, M., Lux, V., Regnier-Prost, S., 1998, Modular Controlled Language Design, in: Proceedings of CLAW 98.
- Reuther, U., Schmidt-Wigger, A., 1998, MULTILINT Abschlußbericht, URL: <http://www.iai.uni-sb.de/de/multi-docs.html/bericht5.ps>.
- Reuther, U., 1998, Controlling Language in an Industrial Application, in: Proceedings of CLAW 98.
- SAE International, 1995, SAEJ 1930 - surface vehicle recommended practice. Technical report, issued 9.1991, revised 9.1995, The Engineering Society For Advancing Mobility Land Sea Air and Space.
- Schmidt-Wigger, A., 1998, KURD in MULTILINT – A Grammar and Style Checker for German, in: Proceedings of CLAW 98.
- Schütz, J., 1998, Multilingual Human Language Technology in Automotive Documentation Workflows, in: Proceedings of Aslib 98.
- Wojcik, R., Holmback, H., Hoard, J., 1998, Boeing Technical English – An Extension of AECMA SE beyond the Aircraft Maintenance Domain, in: Proceedings of CLAW 98.