[*Translating and the Computer 24*: proceedings of the…International Conference…21-22 November 2002, London (Aslib, 2002)]

Controlled Authoring at SAP

Christian Lieske, Christine Thielen and Melanie Wells

SAP AG, Germany

[christian.lieske, christine.thielen, melanie.wells]@sap.com

Dr. Andrew Bredenkamp,

DFKI / acrolinx GmbH, Germany

andrewb@acrolinx.com

Abstract

After assessing various tools on the controlled language market, SAP contracted the German Research Centre for Artificial Intelligence (DFKI) as a development partner for the SAP Knowledge Authoring - Text Enhancement (SKATE) project. The main emphasis of the project for SAP was to provide a solution for automating the check against their in-house standards and guidelines. The outcome of the project is the SKATE software, which is intended for use by technical writers, translators and copyeditors. SAP's main development languages German and English are supported by the software directly within the authoring and translation environment. This paper describes the SKATE software, and SAP's experiences of preparing the software for large-scale use.

1 Introduction

Ensuring consistency and quality in large-scale, multilingual, distributed documentation infrastructures is a complex and expensive task - available off-the-shelf applications are not able to support the automatic checking of in-house terminology and style (aka *controlled language/controlled authoring applications)* and manual checking processes are often difficult to maintain, disappear under time pressure and are very costly. Some of the difficulties involved in introducing controlled language lie in convincing people that a 5%-10% investment in the documentation production process can lead to an overall reduction of costs for multilingual documentation. If you consider that you can hope to achieve a conservative 2%-3% reduction in the translation costs for each language in the chain after the controlled language (less volume, higher standardization, better translation recycling options (Translation Memory), fewer queries due to better clarity, etc.), it becomes clear that this investment can become very worthwhile.

This paper describes how SAP and the Natural Language Processing experts of the DFKI together developed an application that remedies this situation within the project *SAP Knowledge Authoring - Text Enhancement* (SKATE). The SKATE endeavour was challenging in several ways:

1. SAP was the first company in the software sector to introduce a controlled language application and
2. The SKATE software was ultimately intended for use by any technical writer, translator and copy-editor working for SAP.
3. Support was needed for both German and English (further languages may be added in the future)
4. SKATE had to be integrated with SAP's sophisticated documentation and translation processes

The foundation of the SKATE software is an environment for creating customized language checking components which has been developed by the German Research Center for Artificial Intelligence (DFKI). The environment is an open system designed to be used by organisations who wish to integrate their own language standards/style rules. In SKATE, the style rules specific to SAP were developed in-house at SAP, whilst the DFKI developed more generic grammar checking rules. During

the project, SAP put particular emphasis on usability and user interface issues to ensure acceptance by users.

SAP's user feedback and structured usability testing helped to improve the system design of the original DFKI environment, especially for the user interface and user interaction design (see section 3).

2 A Controlled Authoring Application for SAP

SAP's group for MultiLingual Technology conducted a market analysis, and a subsequent evaluation of existing controlled authoring applications in 1999. The outcome was, that off-the-shelf products available on the market, did not fulfil all of SAP's needs. Accordingly, it was decided to set up the project *SAP Knowledge Authoring - Text Enhancement* (SKATE) to develop a controlled authoring application which would take these needs into consideration. This chapter sketches highlights of the project's software development life cycle, and depicts how the resulting application can currently be used.

2.1 Requirements Analysis

The first development-related activity of the controlled authoring project at SAP, was a detailed requirements analysis. The main results of this project phase were that the application

1. must be customizable in-house (most importantly to include SAP-specific style rules)
2. needs to support software-related texts (accordingly, it has to be able to cope with common words and wording in this domain)
3. must support German and English (with an option for additional languages)
4. should be very easy to use (since in the end any technical writer, translator and copy-editor working for SAP may use it)
5. must fit seamlessly with SAP's sophisticated documentation and translation processes (for example, it should be easy to integrate the application into Microsoft Word®, one of SAP's most important editors)
6. must enable both interactive use and background batch-processing (which should summarize checking results in reports)
7. must be powerful and highly scalable (due to the high number of possible users, and the large volumes which may have to be checked)

2.2 Design Decisions

Obviously, certain design decisions were required for a controlled authoring application to fulfil SAP's needs. Most notably, the technology to be used for the actual checking of style rule violations had to be selected. This section describes some of the design decisions which were made during the project. Since the software development lifecycle followed an iterative model, not all of the decisions were made at the beginning of the project. Certain aspects of the user interface, and user interface design for example were only made after the rigid usability tests which were conducted later in the project.

2.2.1 Language Checking Software

The market analysis and evaluation of existing controlled language tools had already indicated that only a few applications provided a good starting point for an SAP-driven development project. After the discussion of detailed requirements, and licensing options, SAP ultimately decided to partner with the German Research Center for Artificial Intelligence (DFKI). Their Flexible Language and Grammar Checker (FLAG) development environment thus became the backbone of the SKATE Language Checking Software (LCS). The most important modules of the LCS are:

- **Part of Speech (POS) Tagger** - (Brants, 2000) tags each sentence element with a detailed grammatical category based on the Penn Treebank tagset. This tagger is statistical, so copes well with words which are not stored in its lexicon.

- **Part of Speech (POS) Lexicon** - lists words with possible tags for the POS tagger. Tags for each word are also weighted according to frequency of occurrence within a particular corpus[1]
- **Morphological Analyser** - (Petitpierre and Russell, 1995) provides a morphological analysis of words; analysis can be used in checking rules that specify, for example, that certain elements must be third person singular
- **Chunker** - (Skut and Brants, 1998) identifies noun phrases (NPs) and verb phrases (VPs); results can be used by checking rules to check whether certain elements are within an NP
- **Rule Engine** - matches results from the Natural Language Processing components (for example, the chunker) against formalized checking rules

2.2.2 User Interface/User Interaction

SAP's interest in usability (see section 3) led to several modifications of the user interface which existed in the original DFKI environment. The modifications fall into the following areas:

1. **more ergonomic user interaction:**

   make most common actions as simple and easy as possible (single click on a marker to remove it, double click on a marker to see information related to it (e.g. the name of the rule that has been violated)

2. **single information area:**

   use a Web browser (rather than a combination of Web browser and Microsoft Windows®/Word® dialog boxes) to display information; this also resulted in more independence from Microsoft Word^ and other options for help resources (see section 2.3.2.4)

3. **better configurability:**

   enable, for example, different levels of information for expert and novice users.

2.2.3 Checking Reports

From the very beginning, SAP's usage scenarios included both interactive use and background batch-processing. In each scenario, a need to summarize checking results in special reports was identified. In order to be as flexible as possible, it was decided to base the reports on XML documents which are generated server-side by the Language Checking Software. These XML documents (which have been formalized by means of an XSD schema, see figure 1) capture all of the information that might be interesting for the different agents in the usage scenarios (e.g. authors, checking rule developers, copy-editors). Information in the XML documents includes the following:

- administrative information (<admin> element): user profile (e.g. user name, settings, input text) the checking workflow and the user environment (e.g. version of the used LCS)
- results of a language checking session (<results> element): for each checking type (style, grammar, terminology, unknown words) a summary of the found errors and a listing of the violations with an error description and context information (<style_or_grammar_error> element)

With this XML-based design it is possible to generate different views on the checking results and different output formats (for example HTML and PDF). It is possible to give detailed listings of the errors with the corresponding context information, e.g. for the evaluation of checking rules, or to calculate statistics for quality assurance purposes. Users can view the reports directly with a single mouse click, or collect them for further processing.

[1] The weightings were modified during the project to adapt them to SAP's technical documentation.

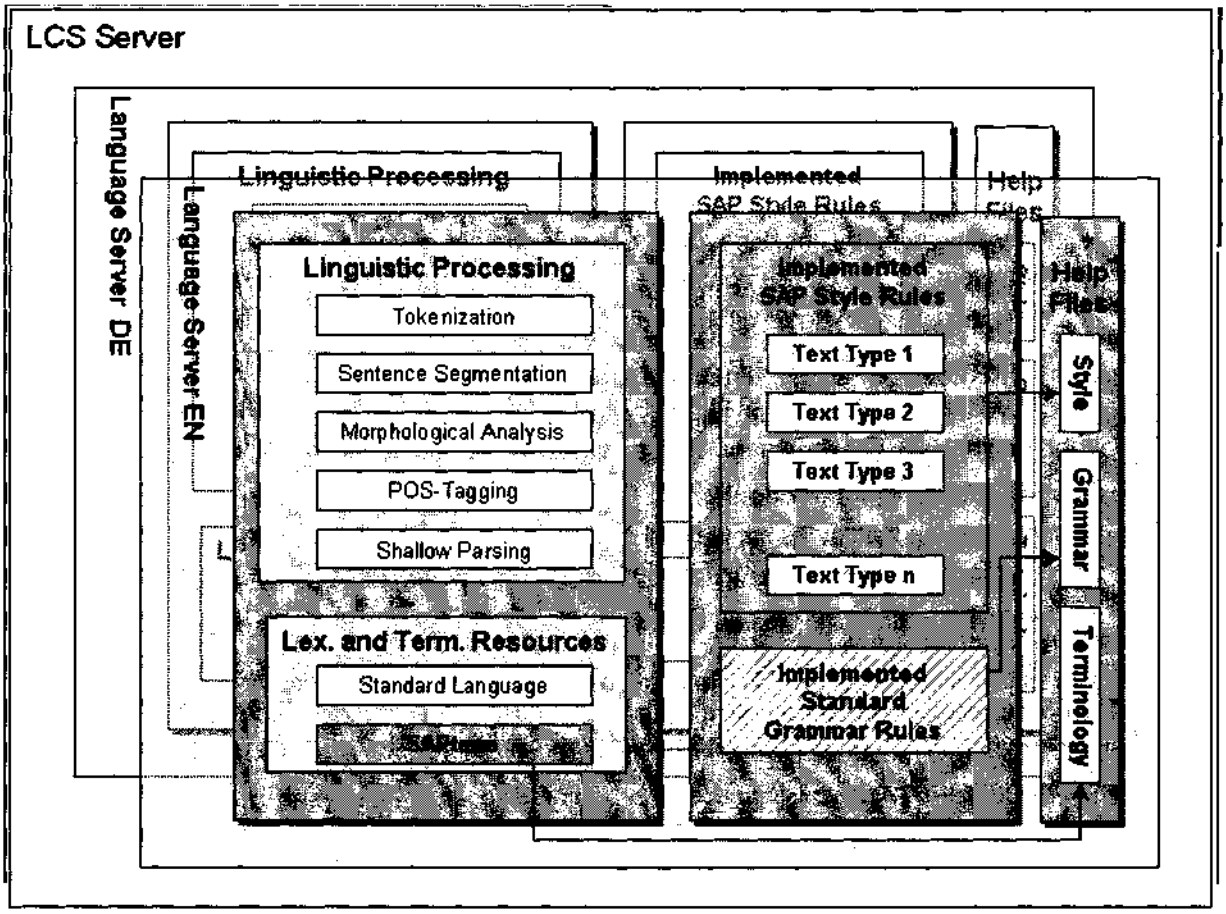


Figure 2: LCS server

2.3 Implementation

2.3.1 Language Checking Software

For ease of maintenance and scalability, SKATE is based on a Language Checking Software (LCS) with a client-server architecture. The typical end-user environment (e.g. Microsoft Word® editor) attaches to the LCS client via COM calls. The LCS client and LCS server communicate via HTTP and TCP/IP.

A typical interaction is as follows:

1. The LCS client sends a text (document segments, whole document or a set of documents) to the LCS server together with a user profile
2. The LCS server replies to the client with a set of results
3. The user can view the results directly in the edited text and as an XML report that can be displayed in a browser or evaluated by further transformations

The LCS server is a multi-user server implemented in Java with some underlying processing components implemented as libraries in C/C++. It runs on Microsoft Windows NT/2000®, Solaris or Linux. For each language (German, and English), SAP operates one server. A meta-server re-routes requests to the appropriate language server.

Each language-specific LCS server consists of language-specific processing components (e.g. for tokenization, part-of-speech tagging, morphological analysis), lexical resources and a number of implemented style and grammar checking rules (different rule sets are defined for different text types) and help resources for checking results (see figure 2). The processing components, the general lexical resources and parts of the grammar checking rules have been provided by the DFKI. The creation and administration of all other SAP-specific resources and checking rules are part of the customizing process done at SAP.

2.3.2 Linguistic Data and Help Resources

The LCS processing components make use of large amounts of linguistic data consisting of

- checking rules for style and grammar checking
- SAP terminology and general lexicons for style, grammar and terminology checking
- abbreviation lists for text segmentation and tokenization

Checking rules and SAP terminology are linked to corresponding help resources so that the author gets appropriate help during language checking (e.g. style and grammar guidelines, examples, term definitions).

2.3.2.1 Checking Rules

One of SAP's main requirements for the development of a controlled authoring application was the customization of the LCS for SAP-specific writing standards, i.e. the in-house implementation of style checking rules. After some training on the error specification formalism used by the system (Bredenkamp et al., 2000) this work was performed by computational linguists in the MultiLingual Technology (MLT) group at SAP.

Concerning general grammar checking the DFKI provided a set of rules based on standard grammar literature (for German: Duden 9,1997) and a (small) provided text corpora of SAP texts. For English the grammar rules were heavily reworked at SAP based on information found at various internet sites and some relevant literature (e.g. Hargis, 1998) and real-life errors found in the documentation.

Rule Selection

For the first introduction of the controlled authoring application, SAP decided to only implement a small number of checking rules, to allow user acceptance to be monitored. The selection of relevant style rules took some time because SAP has a large volume of existing writing standards (e.g. for German: SAP HORIZON L156, 2002) to be considered before any decisions could be made about which were the most relevant rules to be implemented. A rule is considered relevant, if the violation of this rule

- makes a text difficult to read and comprehend
- makes (human or machine) translation difficult
- occurs frequently
- damages the corporate image
- can be implemented

During the whole selection process it was necessary to consult with various SAP-internal groups involved in the process of documentation production (e.g. standardization group for technical writing, authors, translators, copy editors) and to evaluate existing documentation at SAP and relevant literature (e.g. The Chicago Manual of Style, 1993).

Concerning the selection of grammar rules, especially for English, the rules were reworked and expanded at SAP after several test phases. For German grammar checking, SAP observed that violations of the most general grammar rules do not occur very often and still need to decide which of the implemented rules are to be included.

Currently there are 203 rules implemented for German and 100 rules implemented for English:

Checking Type	German	English
Style	25	29
Grammar	178	71
Total	**203**	**100**

Table 1: Implemented Checking Rules

One difficulty in implementing relevant checking rules includes obtaining suitable and extensive test material, such as large amounts of SAP texts that still contain writing errors and, if possible, the corresponding edited texts that contain the (marked) corrections of these errors.

There is never enough time in documentation production for thorough and central copy-editing and therefore only small collections of such test material are available. But high-quality test material is crucial for the user-oriented development of checking rules, to determine which rule violations occur

(frequently) in the documentation and to ensure that our implemented rules cover all or most variants of an error type. There are new initiatives at SAP to establish central copy-editing for all text types and the SKATE software is currently being used for large-scale language checks (approx. 3200 documents / 750,000 words) as part of a quality assurance project for SAP R/3 Enterprise documentation. A secondary aim of this project is to collect valuable data to adapt the checking rules.

Rule Implementation

The linguistic processing components of the server process the text sent by the client. As a result the input text is segmented into sentences and each sentence is split into words or tokens (tokenization). Each token is enriched with multiple linguistic information or feature structures and used by the error specification formalism (Bredenkamp et al., 2000) for the implementation of checking rules developed in the FLAG (Flexible Language and Grammar Checking) project. The FLAG formalism can be described as a pattern matching language (including regular expressions, negation, etc.) over linguistic feature structures. A rule interpreter applies the rules to each sentence of the input text and if a pattern matches, then the corresponding rule with its assigned action is triggered.

Example

One style rule defined in the SAP writing standards (German and English) is the rule for passive voice:

> *In general, use the active voice. The passive voice makes it difficult for the user to figure out who is doing what to whom. Make it clear whether the user or the system performs an action.*

The task is now to formalize this rule with the error specification formalism to detect the use of passive voice in a sentence, like

The status of other objects cannot be checked.

so that the author could rephrase the sentence, e.g. into

You cannot check the status of other objects. (if the user is the agent)
or
The system cannot check the status of other objects. (if the system is the agent)

Based on this sample data, we can write a simple error description that searches for any inflected forms of "be" that occur together with a past participle in one sentence.

It is not possible to formalize all SAP writing standards or general grammar rules with the FLAG formalism. The linguistic components do not provide any semantic information with which we could formalize such rules as "Develop and support one idea in each paragraph". Another limitation of the formalism is that we cannot implement supra-segmental rules such as "Put the main information in the first sentence, and then elaborate in the following sentences" because processing is sentence-based and we cannot consider preceding or subsequent sentences. In addition, the checking of any formatting standards is not possible because we have no access to any inline information (e.g. italics, bold, font size) or to structuring information (e.g. for headings). Formatting standards need to be supported by other appropriate methods like document templates or document type definitions or schemas for XML documents. Nevertheless the formalism is powerful enough to describe local syntactic and lexical patterns in quite a simple and efficient way.

2.3.2.2 Terminology and General Lexicons

SAP Terminology

The whole SAP corporate terminology for over 30 languages is stored and administrated in a concept-oriented terminology database (SAPterm). The two main development languages, German (DE) and English (EN), were taken for the development of a first controlled authoring application. All other languages are translated from one development language or the other and so will already benefit from improvements in consistency etc. from this project. For these two languages SAPterm currently (September 2002) contains 65,938 German and 63,744 English terminology entries organized into more than 40 software components (e.g. terminology for financial accounting or for personnel management).

The import of company-specific terminology is an important part of the LCS customization. Therefore the LCS was delivered with an import tool that converts SAP terminology from an agreed XML format into an internal format used by the LCS.

It has been intended from the beginning of the project to use the *Open Lexicon Interchange Format* (OLIF, McCormick et al., 2001, www.olif.net) as terminology exchange format, but the support for OLIF downloads from SAPterm has not yet been completed. As an interim solution we agreed on an alternative simple XML format, that could easily be generated from the *Trados MultiTerm*® format, which is supported in SAPterm.

The import of SAP terminology is component-based, so that the user can check texts against the terminology of one or more software components for which the documentation is written. The import program creates two full-form term tables for each component, one for allowed terms and one for unallowed terms and the corresponding help resource (see section *Help Resources)* for each term from the XML exchange file. The user can then look up each unallowed term within SKATE to find the allowed variant.

The LCS can currently only access 10 term tables at any one time, meaning that we can only import the terminology for 5 SAP software components at one time. This could be solved by setting up several LCS servers with differing customized terminology.

During the development of the controlled authoring application, SAP's main focus was on the style checking functions. Currently there are basic functions available for term checking to get an initial insight into how the user would use such a function. The users are quite happy with a term checking function integrated in the authoring environment where they do not have to look up terms separately in SAPterm. But it is still unclear whether the organization of term data in SAPterm sufficiently lends itself to satisfactory term checking.

The importing and checking of terms is currently being re-worked to improve performance and usability. The highest priority is to get a seamless and automatic import from SAPterm (via OLIF) to SKATE.

General lexicons

The DFKI provided a general German (old and new German spelling) and a general English lexicon. These lexicons are generally stable, but are continually maintained and updated by DFKI/acrolinx (acrolinx are DFKI subcontractors for this project). The lexicons have been built over a number of years using extensive corpus analysis and collection. The German lexicon contains around 800,000 words (full-form), the English approximately 200,000 words. It is important that these lexicons are maintained separately from specific (SAP) terminology, to avoid them being "contaminated" - although the distinction between technical terminology and general technical words is not always obvious.

2.3.2.3 Abbreviation Lists

For the correct segmentation of texts, it was also necessary to configure the system with abbreviation lists for general vocabulary and for specific SAP terminology. Both lists were generated for German (2026 abbreviations) and English (919 abbreviations) at SAP, so that the update of the lists according to changes of SAPterm can be done in-house. General abbreviations were manually collected from standard lexicons (e.g. Duden) and internet resources, SAP abbreviations were semi-automatically generated from SAPterm and other software that uses lists of abbreviations (machine translation and translation memory software).

2.3.2.4 Help Resources

During language checking the system displays help information for each checking type (style, grammar, terminology) in a standard browser. When the user selects a marked section of text to get further information on this error type or term, the browser opens with more information referring to the selected error candidate.

From the implementation point of view, this means that each checking rule and term must be linked to a corresponding help resource, and is consequently also part of the customization process. As mentioned above, the generation and linking of help resources for terminology is done by the terminology import tool, but for the checking rules additional manual work is necessary.

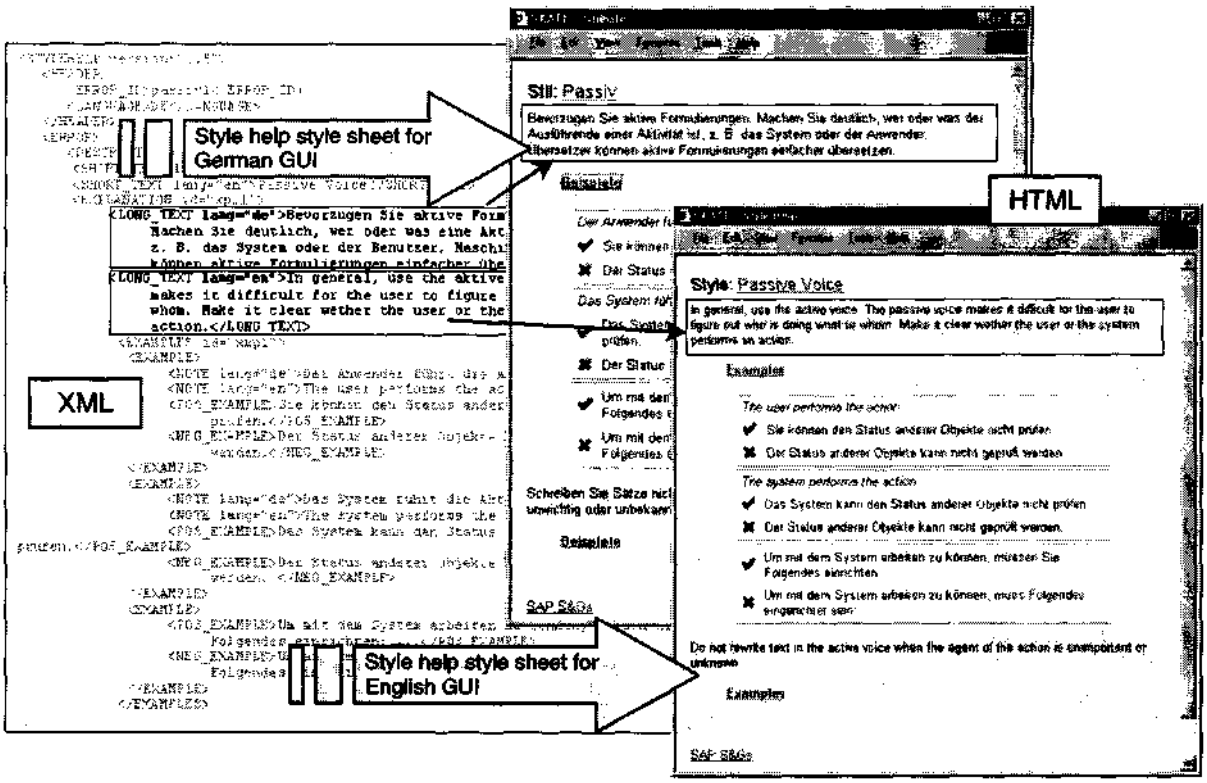


Figure 3: Multilingual help for style checking (LCS version 0.9)

XML for administration and generation of help resources

In the first versions of the LCS, the help texts had to be directly included in the implemented rule code. Editing and translating the help texts (for different GUI languages) was difficult and complicated and not practicable for complex texts, e.g. texts including positive and negative examples. SAP suggested reorganizing the help resources as XML documents and generating HTML output from the XML data. This has simplified the rule code, which now contains links to the help resources instead of the full help texts for each rule. For each GUI language the respective HTML document is linked to the rule. The current Microsoft Word® LCS client supports English and German as GUI languages and with the current configuration, English users can get help on German texts in English and German users can get help on English texts in German.

XML makes it possible to handle multilingual content in a very simple way. We can assign a language attribute (e.g. lang="de") to all translation-relevant elements and with language-specific transformations (e.g. XSL style sheets) we can generate the output (see figure 3). For the SKATE style and grammar help, SAP created an XML resource that contains the multilingual content for each implemented rule and then defined different XSL style sheets for each GUI language. For example, the style sheet for the German style help only takes the German content and formats the HTML output according to the layout definitions, including any static text elements (e.g. title, headings, labels). If a different layout is required, only the style sheets need to be changed. If the content of the help elements requires changing, only the XML data needs to be changed and formatting considerations can be ignored.

2.4 Maintenance

As described in the preceding sections, there are several tasks necessary to keep the SKATE software running and up to date (e.g. for changed terminology or style rules). We see the following regular maintenance issues:

General software administration

- Installation of new versions (server and clients)
- (Re)configuration of servers for German and English (property files)
- Administration and evaluation of logging information (for improvement measures)

Customization of linguistic resources

- Downloads and conversion of SAPterm data
- Implementation and validation of new checking rules
- Generation and administration of help resources

Rollout and support

- SKATE training for authors and translators
- Technical support for authors and translators

2.5 Typical Use

2.5.1.1 Interactive Use

During interactive language checking, authors, translators or copy editors use the SKATE tool to check a document or document section for specific writing standards (style, grammar, terminology). The user can view the checking results in the document immediately and revises the text according to these results. Additionally, the tool generates a report for subsequent off-line evaluation of the checking results.

Interactive language checking consists of the following five steps:

1. User calls the tool

In the LCS MS Word client 0.9 the user calls the SKATE tool using the *Check* icon on the SKATE toolbar or with *Tools -> SKATE Language Checking -> Check...*(figure 4).

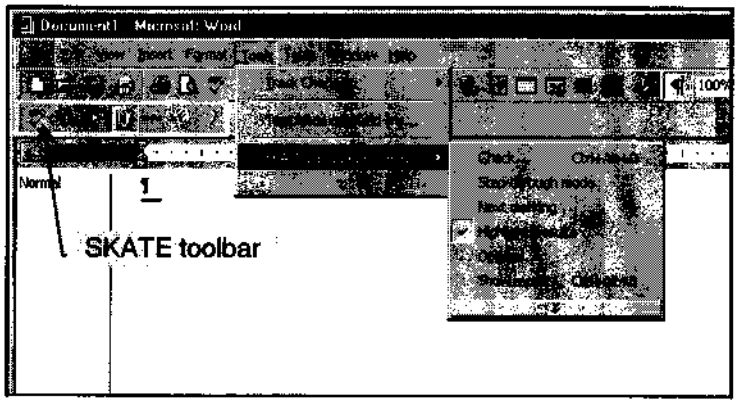


Figure 4: Start of the SKATE tool

2. User configures the tool

A dialog box opens (figure 5), where the user chooses the checking type (style, grammar, terminology) and can set the text properties (Options...: language, text type, subject area).

3. User starts language checking

The user starts the language check with the Check button on the SKATE Checks window (figure 5).

4. User views and processes checking results

When the checking is finished, a results window opens with a summary of the checking results (e.g. 6 style errors, 1 grammar error, 3 illegal terms) and the option of displaying the checking report. The user can then view the marked errors directly in the document, either by stepping through from one marking to the next or by selecting individual markings. In both cases the user can get more information from the SKATE context menu (right mouse click) (figure 6).

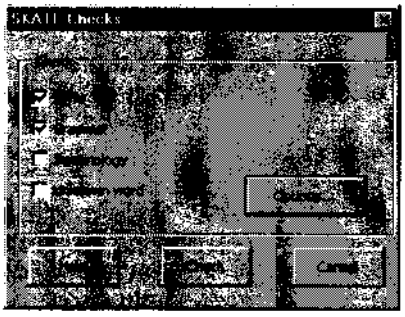


Figure 5: SKATE checks

5. User completes language checking and closes the document

When the user has finished the processing of the checking results he closes the document. Any remaining unprocessed markings are removed with the closing the document (after a warning is given). The system saves a copy of the checking report in the same directory as the document.

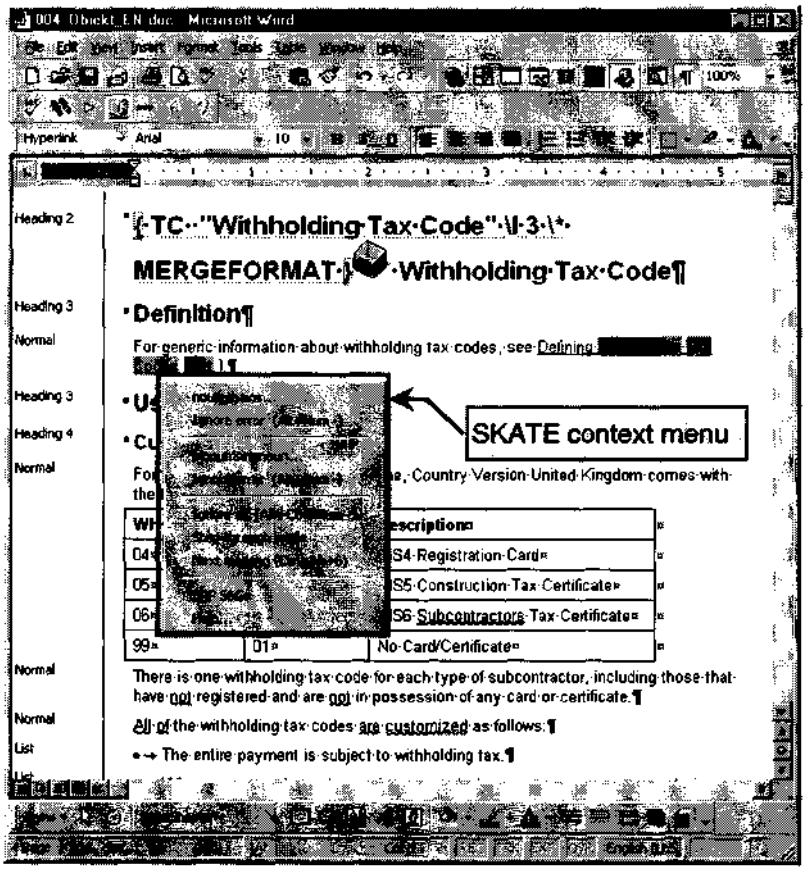


Figure 6: SKATE Checking results and context menu

2.5.1.*2* Non-interactive Use

Non-interactive language checking or batch language checking is currently still in development. For example, during batch checking, quality managers would use the SKATE tool to check a large number of documents (probably overnight) to assess the overall quality or release suitability of the documentation. How else can you achieve or guarantee a reliable quality check for a documentation release encompassing 50 million words and with a very short time-to-market? Users would receive an overview document listing summarized real error counts, with the possibility to drill-down to individual document level in problem cases. It would even be conceivable for the summarized report to contain a list (or a link to a list) of problem documents that have already been flagged as containing too many errors that should be top priority for manual editing.

3 Usability and User Acceptance

From the beginning of the SKATE project, possible SAP users of the software were involved in its development phases. The cooperation of SAP authors, translators and copy editors enabled the development of a controlled authoring application that would be accepted and used by its intended users, i.e. great emphasis was laid on usability and user acceptance. LCS version 0.7 underwent an initial intensive test phase, the SKATE Hands-on Experience (SHOE), which led to a second test phase, the SKATE User Day, which was supported by the SAP Usability Lab and had direct input from SAP's usability experts and the DFKI's programming experts. These usability tests finally led to a redesign of the graphical user interface (GUI) and the user interaction within the LCS Microsoft Word® client.

3.1 Usability Testing

The main goal of the usability testing was to form concrete ideas about the current usability of the software and generate ideas for possible improvements. In addition, the testers also tested the functional coverage of the software (e.g. rule coverage, false/missing alarms) and general software characteristics (e.g. robustness, performance, installation).

3.1.1 SKATE Hands-On Experience

The SKATE Hands-on Experience (SHOE) took place at SAP in Summer 2001. Testing was only done on the German prototype and all 13 of the testers were SAP German authors, including one member of the SAP standardization group for German documentation. Testers worked through a number of test cases for some test texts provided and for texts that they provided themselves. They were then asked to fill out a general survey about the testing and a survey for each specific test-case. The focus was given to style checking, whereby grammar checking and terminology checking were only marginally tested. Detailed feedback was obtained for 29 test cases and general feedback from all participants. These numbers are not representative for all SAP authors (currently there are approx. 250 authors working at SAP), but even this small number of testers gave us a lot of valuable feedback and, most importantly, confirmation that the software would be accepted after a number of improvements. In more detail, the observations of the testers can be categorized as follows:

1. **Tool-related observations**
   - Usability: tool is easy to learn, look & feel is poor, dialogs could be clearer, general usability could be improved
   - Robustness: acceptable for single user, unacceptable for more than 3 users
   - Speed and Resource usage: acceptable for single user and small documents, unacceptable for multiple users and for longer texts, parallel work on other Microsoft Word® documents not possible
2. **Rule-related observations**
   - General error detection: a majority of testers (8/13) thought that most errors were detected
     correctly
   - Checking rules: some rules yield too many false alarms, checking rules should be selectable, additional rules requested
   - Help texts: most testers thought that the help texts could be made clearer with examples and a better layout
   - Checking reports: content of the reports is too extensive, layout poor
3. **Other observations**
   - Terminology checking: needs to be more fine-grained (tested version did not include term checking by software component)
   - Error processing: processing is slow and laborious
4. **General observations**
   - Acceptability: most testers would use the tool, providing the reported problems were fixed and the rules made more accurate

3.1.2 SKATE User Day

The results of the SKATE Hands-on Experience made it clear that the usability of SKATE had to be drastically improved for it to be accepted by users. In September 2001, we therefore held a SKATE User Day at SAP in which an SAP usability expert and DFKI representatives also participated. There were a number of testers from documentation and translation and the English prototype also made its debut. The testers used the SKATE tool to check and correct a provided text and while doing this they were observed and interviewed by members of the SKATE development team. After that the testers and observers discussed their findings directly with the usability expert and the software developers. The results from this, together with the results from the SHOE, were then formalized and turned into a detailed specification for the new user interface and user interaction design (see the following section). By the end of 2001 the specification was implemented in the current version (0.9) of the SKATE software.

3.1.3 User Interface and User Interaction Design

Based on the results of extensive usability testing we decided to completely rework the existing user interface and interaction design. This included

- a consistent interface (texts and layout of the SKATE toolbar, menu and dialog windows)
- the separation of the SKATE checking options and configuration options
- more system feedback (status bar, summary statistics at the end of checking)
- a new interaction design (step-through mode, SKATE context menu)

- a new style and grammar help (browser-based)
- a new checking report (XML scheme definition and XML style sheets)
- a standard Microsoft Windows® design

For example, the version of SKATE that we used in the Hands-On Experience tests only included minimal help information for the user to view when a style or grammar error occurred (Figure 7). After the redesign, users can get more information on style or grammar rules and corresponding examples on demand in the browser. We abandoned the initial approach of offering two user levels (beginner, expert) with two different levels of help information, because it was too inflexible in cases where an experienced user wanted to get more information.

As described in section 3.2.1 the work invested in usability testing was extremely worthwhile and the pilot users were especially satisfied with the user-friendly interface.

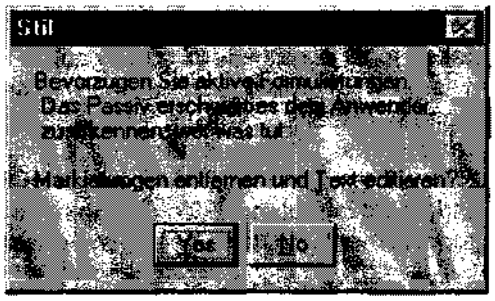


Figure 7: SKATE German style help (version 0.7)

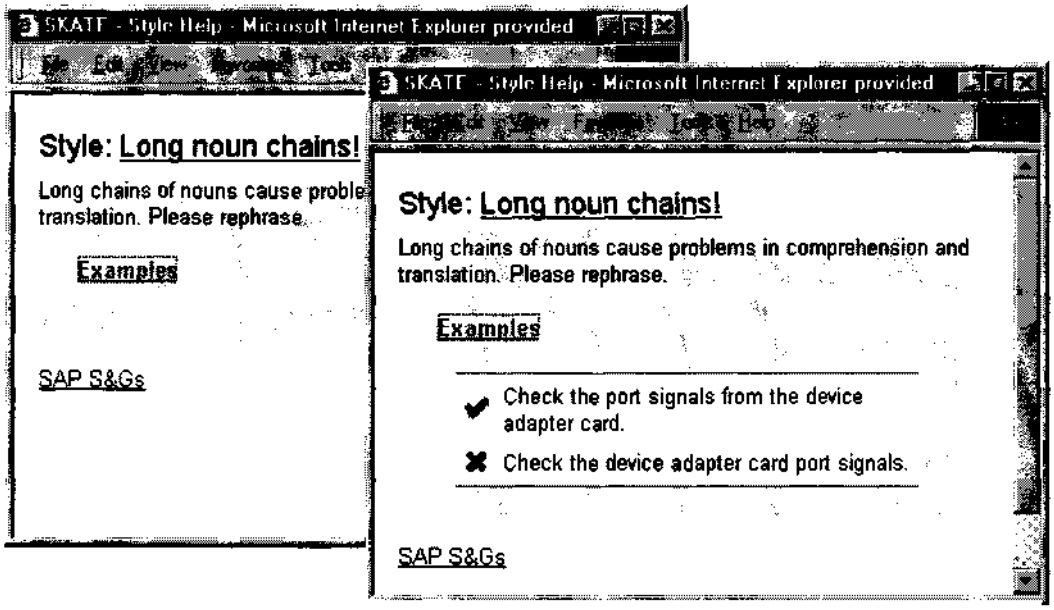


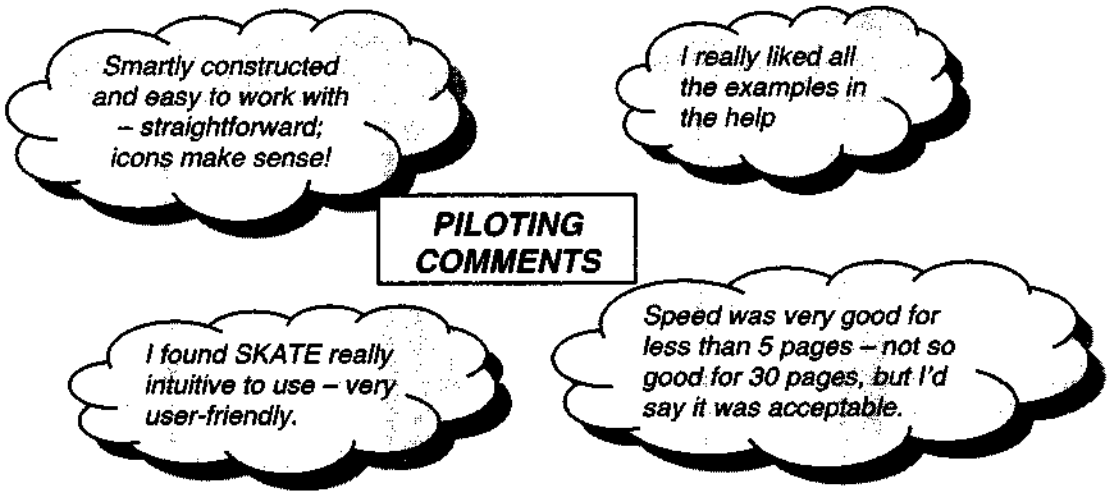
Figure 8: SKATE browser with (un)expanded examples

3.2 Roll-Out

The roll-out of the SKATE tool at SAP started with a piloting phase where a small number of users (authors and translators) used the tool as part of their information development. It was planned to open the tool for a controlled availability (15-20 users, see section 3.2.2) in May 2002 but this had to be postponed until Fall 2002 to allow for a number of organizational issues including maintenance contract negotiations, budgeting, internal rollout decisions, fitting in with software release planning for the documentation environment and participation in quality assurance projects.

3.2.1 Piloting

SKATE piloting took place at the start of 2002 with a small number of SAP users with varying profiles. These included authors, translators and internal standards and guidelines specialists. This was the first time that the English prototype was properly tested alongside the German (ignoring the SKATE User Day). The tests were mainly to evaluate the usability of the software (with its new redesigned help format, interface and user interaction implemented from the results of the SKATE User Day). The results and feedback from the piloting were generally positive and were presented to managers (along with general project details) in May 2002, when the decision was taken to start a controlled availability for SKATE along with the next authoring software upgrade.



3.2.2 Controlled Availability

This decision means that controlled availability of the tool is starting now! SAP are aiming for 15-20 users with mixed profiles (authors, translators, copy-editors, documentation managers) for each language. This should allow SAP to work within the current server constraints, but to still prepare the administration and the system for larger-scale rollout. It should also enable rule violations and false alarms to be satisfactorily monitored and an early assessment of new areas/types of documentation. Due to the complexity of the SAP documentation landscape (subject areas, terminology, documentation types) it is in everybody's interest to introduce SKATE slowly at SAP so that the effects of adding new subject areas or documentation types can be correctly monitored. For example, it may be necessary to add new style or grammar rules or even whole new rule sets when a new usage area is added.

In the meantime, SKATE has already been used at SAP within a high-profile quality assurance project for documentation, to provide statistics on the number of style, grammar and terminology violations in a subset of the SAP documentation (approx. 3,000 documents). The results of this project can only be positive for SKATE and SAP:

user awareness
management awareness
large-scale testing of programmed rules
large-scale testing of linguistic resources
concrete error statistics for documentation
pinpointing the most common errors

3.2.3 General Availability

In addition to controlled availability and large scale testing, SAP has discussed various other internal usage scenarios for SKATE:

1. As a Service

A central group could provide language checking as a central service for all user groups. The advantages to this would be that there would be fewer installations and the service providers would develop a very good overview of the quality of the documentation and the common errors contained in it. Disadvantages would be inconvenience for users and loss of impact of results .

2. As a Desktop Application

Each author, translator, reviewer, proofreader, copy-editor, documentation manager, quality assurance manager, etc. would have the client software installed and would run checks on documents as and when they feel the need. This is SAP's long-term goal for SKATE. This scenario also includes batch jobs on large numbers of files to check that a certain number of errors is not exceeded before delivery to customers.

4 Future Plans

The DFKI/acrolinx are interested in providing new clients and languages and in improving terminology, grammar and style checking. SAP are looking for a successful implementation and rollout (consolidation of position) before expanding the project further. Although there are plans afoot for new text types, clients and additional languages. The current development plans include:

- an optimisation in the performance and scalability of the client and server components
- an expansion of the terminology component to support OLIF and a larger number of subject areas
- an improvement in reporting and batch job processing
- an update for the background linguistic resources.

These developments will be undertaken by DFKI and acrolinx as part of the SKATE maintenance agreement with SAP.

5 Conclusions

During the SKATE project, the SAP members of the development team were always very careful to avoid the phrase "Controlled Language" with all of its possible negative connotations. To encourage acceptance of the SKATE tool within SAP, it was always described as a "text enhancement" or even a "quality enhancement" tool. Experience so far has shown, however, that these fears were entirely misplaced. Acceptance of SKATE at SAP is widespread on all levels (management and users) and resistance has not been experienced so far. Even though rollout has not been completed, various internal organizations describe SKATE as an integral part of the quality assurance process for documentation and include the software as an important part of future plans, and most potential users want to get their hands on it. The economic situation and the constant pressure to reduce costs and time-to-market have apparently overtaken all the fears of being "controlled". It may also be that regular status meetings, and the publishing of the testing results have already made people feel more comfortable with the idea. Whatever the reason, it cannot be denied that management support on all levels is absolutely essential for the successful development and introduction of new tools as is the necessary technical infrastructure.

If we had to invent/design the SKATE LCS all over again, then usability issues would appear earlier on the agenda. These issues became much more important than we expected at the beginning of the project.

Depending on the profile of the user, there are different perceptions of quality. With the SKATE tool, however, we believe that we have a tool that we can use to systematically check large volumes of documentation to start producing reliable quality benchmarks for the first time. SAP's Standards&Guidelines and existing human quality checks can only sustain and support this endeavour and we expect a great deal of synergy between the two processes.

6 Bibliography

Brants, T. (2000), TnT - A Statistical Part-of-Speech Tagger", in *Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000,* Seattle, WA, 2000.

Bredenkamp, A., Crysmann, B. and M. Petrea (2000), "Looking for Errors: A declarative formalism for resource-adaptive language checking", in *Proceedings of the 2nd International Conference on Language Resources and Evaluation,* Athens, Greece.

Duden 9 (1997), "Duden, Richtiges und gutes Deutsch: Wörterbuch der sprachlichen Zweifelsfälle", Wissenschaftlicher Rat der Dudenredaktion, 4. Auflage, Dudenverlag, 1997.

Hargis, G. (1998), "Developing Quality Technical Documentation", chapter "Words to Watch for Clarity", Prentice Hall, 1998.

McCormick, S., Lieske, C. and G. Thurmair (2001), "The Open Lexicon Interchange Format Comes of Age", in *Proceedings of the MT-Summit VIII,* Santiago de Compostela, Spain, 2001. (http://www.olif.net/olif2/documents/olifMtSummitVIII.pdf)

Petitpierre, D. and G. Russell (1995), "MMORPH - The Multext Morphology Program", Multext deliverable report for the task 2.3.1, ISSCO, University of Geneva, February 1995.

Skut, W. and T. Brants (1998), "Chunk tagger - statistical recognition of noun phrases", in *Proceedings of the ESSLLI Workshop on Automated Acquisition of Syntax and Parsing,* Saarbrucken, Germany, 1998.

The Chicago Manual of Style (1993), 14th Edition , University of Chicago Press, 1993.

SAP HORIZON L156 (2002), "Allgemeine Standards und Richtlinien für das Schreiben bei SAP", internal HORIZON document L156, SAP AG, 2002.