Int J Artif Intell Educ (2016) 26:489–503
DOI 10.1007/s40593-015-0061-0

ARTICLE

Web Delivery of Adaptive and Interactive Language Tutoring: Revisited

Trude Heift[1]

Published online: 22 August 2015

Abstract This commentary reconsiders the description and assessment of the design and implementation of *German Tutor*, an Intelligent Language Tutoring System (ILTS) for learners of German as a foreign language, published in 2001. Based on our experience over the past 15 years with the design and real classroom use of an ILTS, we address a number of technological and pedagogical issues as they relate to the approach and motivation of designing an ILTS. We also discuss some of the core limitations of *German Tutor* that eventually resulted in *E-Tutor*, a modified and enhanced ILTS implemented in 2003 and further expanded in 2009. The commentary concludes with a description of the core contributions of *German Tutor* as a learner-centered ILTS. Since its initial implementation in 1998, *German Tutor* has been studied as a component of regular classroom instruction and this body of research has informed not only system upgrades and development but also studies of learner-computer interactions or, computer-assisted language learning (CALL), more generally.

Keywords Intelligent Language Tutoring Systems · Natural language processing · Learner modeling · Online language learning

Introduction

This commentary revisits our description and assessment of the design and implementation of *German Tutor*, an Intelligent Language Tutoring System (ILTS) for learners of German as a foreign language, published in 2001. It first describes our motivation and initial approach to designing an online, NLP-based ILTS and reports on our experiences of the challenges we faced in implementing an ILTS for regular classroom instruction. *German Tutor* was specifically designed for web delivery and, given the technical limitations of processing speed at the time, system performance and efficiency were

✉ Trude Heift
heift@sfu.ca

[1] Linguistics Department, Simon Fraser University, Vancouver, Canada

key factors that prompted our modular approach to system design. We adopted a heterogeneous three-tier architecture that integrated multiple servers and several programming languages. A primary Web server interacted with the client machine via CGI scripts and JAVA technology while the intelligent and adaptive components resided on a separate server dedicated to answer processing. In addition to tackling these technological challenges, we also placed a strong emphasis on pedagogical issues to achieve an interactive and adaptive system that can serve a vast diversity of students. The error-detection algorithms were not limited to a particular native language user group and identified the source of an error in order to provide error-contingent feedback. Individualization of the learning process was achieved through a dynamic student model that modulated feedback messages and provided remedial tasks suited to learner expertise.

Nevertheless, the initial system was limited to the content commonly taught in the first semester of university language instruction and, for this reason, a significant system upgrade was performed in 2003 (and further expanded in 2009) which also involved rewriting some of the initial computer code to not only achieve a wider coverage of grammatical phenomena but also to provide additional learning tools. These upgrades also included an extension of the activities types to no longer be limited to grammar practice but also target the learners' pronunciation, vocabulary skills, listening/reading comprehension, cultural knowledge and writing skills at both the beginner and intermediate proficiency levels. This commentary details the system's progress to date both in terms of computational as well as pedagogical upgrades. Finally, it discusses its core contributions as a learner-centered ILTS that has been implemented and studied in a regular language learning classroom for more than a decade. These user studies have not only informed system upgrades and development (see Heift and Caws 2014) but also contributed to the body of research in learner-computer interactions or, computer-assisted language learning (CALL), more generally.

Motivation

At the time when our article was written, web-based language learning activities were limited to multiple choice-like exercises with simple string-matching answer processing techniques. Learner errors were addressed with generic corrective feedback (e.g., "wrong, try again!") by making little or no reference to the source of an error. The language learning process was also not individualized with respect to learner and task variables, or feedback, for example. A few Web-based authoring programs for language learning such as HotPotatoes (Arneil and Holmes 1999) had just surfaced but web-based applications that made use of NLP technology for language learning had not yet been implemented, at least not beyond a prototype stage and thus not in regular classroom instruction. However, there were also many open pedagogical questions in CALL that required empirical testing. For instance, what feedback types are most effective with different learners? What are the distinct learner profiles of students working in a CALL environment? How can we devise ways of individualized instruction suited to a variety of learners, by at the same time addressing the needs of individual learners?

Naturally, a web-based language learning platform also opened up the possibility to examine the working behavior and linguistic performance of a very large and diverse

learner group which, from a research perspective, was an attractive proposition for researchers working in the field of second language acquisition (SLA). However, in order to answer those and similar pedagogical research questions, a system capable of processing and analyzing learner input was required. This entailed challenges not only from a computational point of view due to slow processing speeds at the time but also from a practical perspective due to the limited availability of web-based tools to construct interactive multimedia applications. Language learning applications have additional demands in that these systems not only must deal with well-formed input but, in addition, and much more challenging, they must be able to process learner errors in a pedagogical way. This requires a linguistically sophisticated system equipped, for instance, with a grammar and a parser which analyzes sentences from the student and detects grammatical and other errors. The feedback modules of the system then correlate the detailed output of the parser with an error-specific feedback message. Individualization and adaptivity is achieved with techniques of learner modeling by taking into account various learner and task variables (see e.g., Xu and Bull 2010; Dimitrova et al. 2007).

Approach and Core Limitations

From a computational point of view, the implementation of *German Tutor* involved two servers to distribute and thus divide the processing load of the computational analysis and user interaction. The intelligent and adaptive parts of the answer checking process resided on a dedicated UNIX machine with two CPUs running Prolog while the Java and HTML serving and pre-processing responsible for the interaction between the user and the Prolog server was handled by a Web and Language server. The answer checking process was implemented as a modular approach whereby the flow of user input, e.g., a sentence, passed through various answer checking modules. Accordingly, the system was organized in such a way that if any error checking module detects an error, further processing is blocked. As a result, only one error at a time is displayed to the learner. This decision was also motivated by pedagogical concerns given that previous research (e.g., Van der Linden 1993) had shown that displaying more than one feedback message at a time makes the correction process too complex for the learner. From a computational point of view, interrupting the error processing mechanism, however, also assisted in distributing the queue for each module in case of multiple, simultaneous users.

Given the two main components of the system, i.e., the intelligent answering processing/adaptive parts and the web server, in hindsight we note that the modular NLP components with their error checking and processing techniques turned out to be a good design choice and they are still in use after 15 years, albeit in slightly modified ways. From the point of processing speed, for instance, a modular approach to answer processing, although far less needed nowadays due to much improved processing speeds, has allowed us to efficiently expand the activity types over the years and deliver them to a very large user group by selectively calling on some of the answer processing modules but not on others. For example, in a learning activity that asks for a one-word answer, word order errors are not possible. Thus the answer checking mechanism for those activity types only needs to rely on modules that check for spelling and grammatical rule violations and the remaining error checking modules

can be ignored. As a bonus, this also cuts down on the number of false positives given that each module targets very specific error classes and thus limits the error scope.

In taking a broader perspective on the processing of learner errors, the choice of writing a grammar, or more generally, an error processing technique that is not tied to the learner's native language was also a key decision. Unlike systems in the past which commonly established the most likely errors based on the learners' native language and then built the system's error analysis around those errors (e.g., Schuster 1986), *German Tutor* can potentially handle all possible errors, albeit in varying degrees of specificity. This is essential for a web-based system that serves learners with various native language backgrounds.

One of the core limitations of the approach taken in *German Tutor*, however, which was mainly due to technological limitations at the time, was the implementation of client-end JAVA applets which called the Web Server asking for exercise data, passing the user's name, module, and number of exercises to retrieve. In the late 1990s, web applications were synchronous: a user action within the browser interface resulted in a request to the server that forced the browser to perform a full page refresh. Java applets were a promising alternative for a more dynamic interface, but by the early 2000s, end users were encountering compatibility issues running applets on Microsoft and Sun's competing Java Virtual Machine (JVM) implementations. Furthermore, applets require downloading, installation, and possibly the installation of a compatible Java Runtime Engine (JRE), all of which may require an administrator's permission to install on network computers. Proprietary solutions, like Adobe's Flash, were rejected for similar reasons even if this implied that, in some instances, we were unable to implement certain activity types at the time (e.g., drag-and-drop).

In the interest of achieving the widest compatibility as part of our system updates in 2003, we opted instead to rewrite the interface in plain-vanilla HTML 4.0 using frames and Javascript for dynamic content. HTML frames provided a means of dynamically updating a section of the browser page, like a feedback or help field, without the need for extra plug-ins or special permissions. These decisions, however, were pivotal given that our ILTS is used every semester by hundreds of language learners in regular classroom instruction with limited user support.

Progress to Date

German-Tutor was always intended to be used for real classroom instruction such as a CALL component in a blended language learning environment. For this reason, much thought was given to building a system that requires little user support and system maintenance, and ages well. In 2003, we expanded the language learning content of *German Tutor* which, at the time, was limited to the content commonly taught in the first semester of university language instruction. As part of the upgrade, the system was renamed to *E-Tutor* and now covers the first three semesters of university German, that is, the entire grammar of German commonly taught in a university language program. *E-Tutor* follows the grammatical and vocabulary sequence of *Deutsch: Na klar!* (Vansant et al. 2011), a textbook commonly used for language learners of German in North America and thus the system lends itself especially well for use in conjunction with regular face-to-face instruction.

System Design and Learning Content

While the underlying computational design of the system (see Fig. 1), for reasons discussed above, stayed the same with respect to its modular approach to answer processing, the Prolog grammar was rewritten in Java. Client-end Java applets were replaced by JSP pages communicating directly with Java servlets for error-checking, feedback, grammar help and dictionary queries. A JSP-driven Java servlet solution was chosen to aid maintainability, scalability and efficiency. The end result was a significant (10:1) improvement in system responsiveness and load handling. The transition to a pure Java stack was driven to an extent by the decision to expand language learning content, which in turn required a set of utilities and authoring tools. Language learning content (e.g., activity types and exercises) as well as additional language tools (e.g., dictionaries, morphological look-ups, lexicons) were also developed in C/C++, facilitated by the relatively easy translation between C++ and Java.

With these computational upgrades in place which now allowed us to serve large user groups and handle the entire German grammar typically taught to language learners at a beginner to intermediate level, we then extended the learning content by implementing 10 different activity types for each of the 15 chapters (see Fig. 2) with a custom-made authoring tool. The learning activities cover vocabulary and grammatical constructions typically taught at a beginner (e.g., subject-verb agreement, article usage) and intermediate level (e.g., relative clauses, passive constructions).

The activity types allow students to practice chapter-related vocabulary and grammar. In addition, there are learning activities for listening and reading comprehension, culture and writing, and a pronunciation unit with extensive audio recordings of native speakers (see Heift 2010b). The format of the activity types reflect different learning goals. For the pronunciation exercises, students listen to a sentence and are then asked to fill in the missing sounds targeted in a given chapter. For vocabulary practice, learners have to choose the correct word or provide missing inflections. For the grammar exercises, students fill in blanks, translate sentences, or write sentences with prompts provided. For listening comprehension, students write out a pre-recorded dialogue. For reading comprehension and knowledge of culture, students answer multiple choice questions that test their comprehension of a text. Finally, there is a writing task that focuses on the chapter theme which students, however, submit for correction to their instructor.

In addition to extending the learning content, we also added several learning tools (e.g., English-German, German-English dictionaries, vocabulary flashcards) and multimedia in

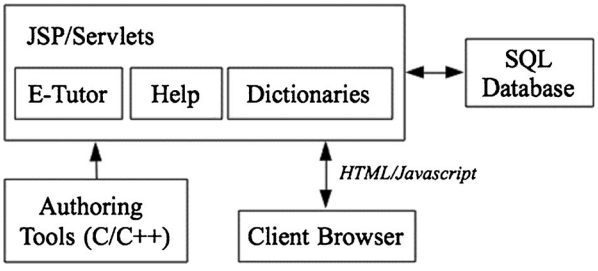

Fig. 1 *E-Tutor* architecture

Fig. 2 Chapter overview of *E-Tutor*

the form of authentic pictures, cultural information on Germany and its people all of which we embedded in a newly designed interface. Naturally, given that the system is used in regular classroom instruction by several NA universities each semester, we also included extensive help pages that cover detailed explanations on pedagogical content (e.g., activity types, grammatical terminology) and system usability (e.g., displays, navigation).

Learner Feedback

We also made additional use of our fully specified lexicon as part of the NLP component of the system by, for instance, implementing a context-sensitive help function that learners can access when receiving feedback on a particular error. In the case of a grammatical error, the system links to the grammatical paradigm relevant to the student's mistake. These paradigms are constructed during runtime and geared at a very specific learner mistake. For instance, in Fig. 3, the learner was asked to construct a sentence with the words provided by the prompt. The interface for the learner feedback consists of a display field with five tabs that are consistent for all activity types: feedback, history, grammar help, dictionary, and error profile.

Under the *Feedback* tab, the system displays the user feedback by highlighting the error in the learner's input followed by a error-specific, metalinguistic explanation of the error ("You made an error with the verb *werden. The subject is singular.*"). In the case of a lexical error (e.g., spelling), the system links to that particular word in *E-*

Tutor's bilingual dictionary (*Dictionary* tab) which consists of approximately 20,000 entries. If the learner committed a grammatical mistake such as the one displayed in Fig. 3, the system links to the grammatical paradigm that corresponds to the particular learner's error, in this instance, to the main verb *werden* (to become). The learner accesses the grammatical paradigm by clicking on the link. The *History* tab allows learners to examine their previous submissions for an exercise by also still linking to the context-sensitive help options that the system displayed for each individual submission. This then allows learners to revisit the grammatical paradigms or vocabulary items that were violated in their previous input. The *Error Profile* tab displays the error statistics by showing the kinds and number of errors that previous learners have made with this particular exercise.

Tracking and Learning Modeling

In addition to the improvements on learner feedback and help options, the system upgrade also included the implementation of an extensive tracking system. Besides a unique student ID and a time stamp, the log records the entire interaction between the computer and the student. This includes the activity type, the student input, the system feedback, and navigation patterns. This information is stored in the Report Manager which is part of *E-Tutor's* learner model. From a user perspective, the Report Manager provides access to a bookmarking system that tracks exercise completion for each user

Fig. 3 Learner feedback with a help link to an inflectional paradigm (*verb* '*werden*')

and exercise set. For each activity type, the system saves the information for each student and session between visits to the site (see the icons in Fig. 2 next to each activity type). This allows students to always continue their work from their last visit. This was an important upgrade to the system given that we expanded the learning content and activities significantly and thus students were unable to perform the learning activities in one sitting. In addition to the learner's progress, the Report Manager also saves information on the student's performance. The journaling system of the Report Manager records prior inputs along with detailed error reports which students can inspect at any time and mail to their instructors (see Heift 2010b).

The information on students' performance stored in the Report Manager is also used by the learner model of *E-Tutor* which provides a dynamic assessment of each learner by considering past and current learner performance and behavior in relationship to a particular learning activity (see also Amaral and Meurers 2007; Heift and Schulze 2007). Thus the system's interaction with each student is individualized as to the kinds of errors that are targeted in the student input as well as the ways in which they are communicated to the learner. For instance, the system might change the order in which errors are displayed for a particular learner depending on the learner's past performance or the focus of the learning activity. The amount and wording of the feedback can also be adjusted according to the learner's prior error history or for different skill levels of learners given that some students require more detailed feedback on their errors than others. Thus their need for additional assistance or practice with a language learning activity is monitored by *E-Tutor's* learner model and can even be addressed with remedial exercises by targeting areas of weaknesses of particular students.

More generally, the feedback messages in *E-Tutor* can readily be altered and adjusted to a particular exercise because the system separates feedback generation from linguistic analysis and learning activity. As a result, the learner model of *E-Tutor* obtains much flexibility in that, for instance, some errors might even remain unreported for some students and certain learning activities if reporting is deemed unnecessary from a pedagogical point of view. To achieve this, the learner model keeps track of each student's performance on a variety of grammatical phenomena (e.g., agreement, modals, verb particles) based on the information obtained from the grammar and the parser. The goal of the parser and the grammar is the generation of phrase descriptors, each of which describes a particular grammatical constraint, its presence or absence in the input sentence and the student's performance on this constraint. Phrase descriptors correspond to structures in the learner model and are the interface medium between the learner model and the grammar of the system (see Heift and Schulze 2003).

A phrase descriptor is implemented as a frame structure that models a grammatical phenomenon (e.g., subject-verb agreement, noun gender, etc.). Each member of the frame consists of a name followed by a value. If the grammatical phenomenon is present in the student's input, the value is either *correct* or *error* depending on whether the grammatical constraint has been met or not, respectively. If the grammatical constraint is missing, the feature value is *absent*. In addition to the grammatical features defined, the grammar uses a type *descriptor* representing the description of the phrase that the parser builds up. This type is set-valued and is initially underspecified in each lexical entry. During parsing, the values of the features of *descriptor* are specified. Ultimately, descriptor records whether the sentence is grammatical and what errors were made.

For each grammar constraint, the learner model keeps a counter for each student with a score for each individual grammar skill. The score increases when the student provides evidence of a successful use of that grammar skill, and decreases when the student provides evidence of an unsuccessful use of that grammar skill. The amount by which a student's score increases or decreases can vary depending on the current value of the score. When a student makes an error on a particular grammar skill, the message they receive then depends on their score for that skill. If they are ranked as novice, they will receive a more informative message than if they are ranked as an expert. Since the score for each grammar skill is independent of the score for the other grammar skills, a student may be expert at subject-verb agreement, but novice at forming the passive—and receive the appropriate message.

The score information is also used for a variety of remediation and assessment tasks. By comparing the learner model at the beginning and end of a session, we can provide a summary of the mistakes that a student made during that session (see Fig. 4). In our current system, these are summarized into general categories such as "Article Inflection", "Verb Inflection", etc. These groups are set by means of a parameter file. Similarly, we can also identify the grammar skills where the student was correct and provide a "positive" of what the student did right. At present we show a list of the errors at the end of each exercise set.

Further, students can also examine their learner model since initial system usage and identify their current strengths and weaknesses by also comparing them to previous users who completed the same exercises. For instance, Fig. 5 shows the number of correct responses as well as a break-down of all errors (totals and percentages) for a particular user and those for all previous users (4378 in total). Fig. 5 further indicates that, for all exercises completed, the student performed much better on word order than previous users while making more grammar and less spelling mistakes. Note that the search terms the learner can specify on top of the page can be adjusted to only display the results for certain chapters, exercise types and individual exercises.

Learner Corpus

Another fairly major update to *E-Tutor* followed in 2009 by which time a stronger focus on web-based exploratory learning had become prevalent and, given that *E-Tutor* had been used by learners of German for over a decade, it lent itself well for this purpose. We created a learner corpus consisting of several million responses submitted by roughly 5000 previous learners who had completed the activity types of *E-Tutor* between 2003 and 2008. We provided an interface to the learner corpus that, for instance, allows learners to examine interlanguage or task-specific phenomena and the benefits in this respect have been well documented (see Borin and Dahllöf 1999; Braun 2005; Gaskell and Cobb 2004; Granger 2003). Naturally, concurrent users can also compare their work to that of thousands of previous learners who have completed the same exercises in the same learning environment and examine their respective error patterns. This tool was fairly easily constructed due to the linguistic analysis of user input and the log that the system constructs from all user inputs.

The learner corpus, however, also allowed us to conduct an extensive statistical analysis with the goal to produce a ranked list of errors for each exercise, activity type and chapter based on prior student performance during those years. For each error profile, we then

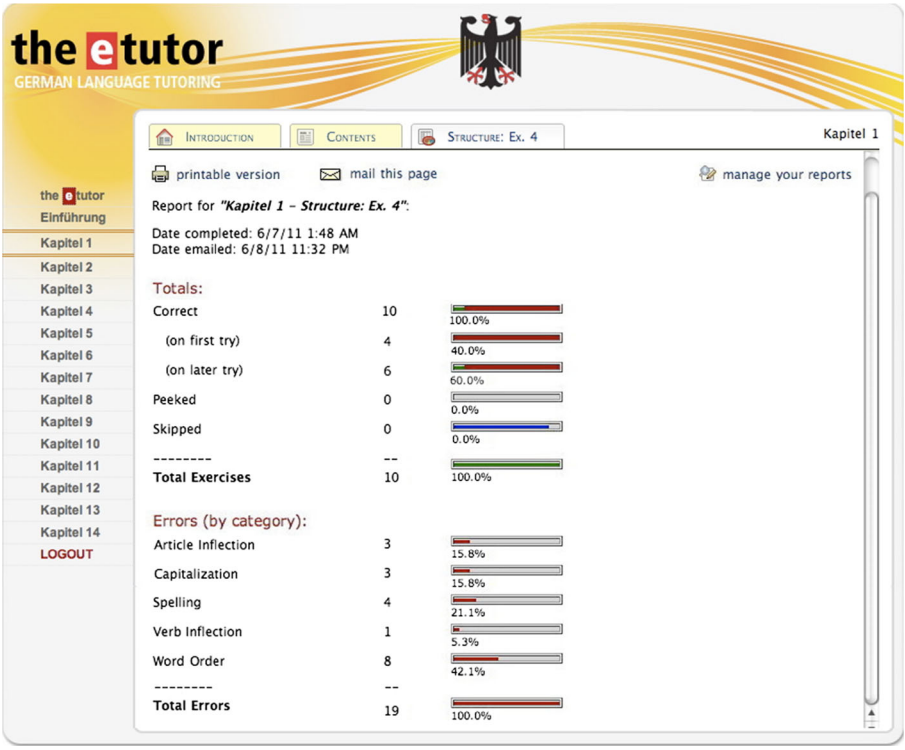

Fig. 4 Summary page

generated exercise-specific hints that the system displays when students start an exercise (see Fig. 6). Research (e.g., Ellis et al. 2001) has shown that this type of preemptive feedback may not only lead to more successful task completion but also reduce potential frustration by marking critical features in the language task. Again, and as with the error-specific feedback in response to a learner error, the preemptive feedback of *E-Tutor* forms part of the learner model in that the kinds and number of hints displayed by the preemptive feedback can be adjusted based on prior learner performance.

Core Contributions and Practical Impact

Core contributions of *German Tutor* (*E-Tutor*) relate to its emphasis on language learning pedagogy and its implementation and use in the regular language learning classroom. The pedagogical underpinnings of the system were to provide error-contingent learner feedback by, at the same time, being adaptive to the needs of a large user group, especially that of a language learning class at a university level. For this reason, *E-Tutor* is not to compare with, for instance, Tactical Language and Cultural Training Systems developed for the US military, or commercial programs such as Rosetta Stone given their different learning environments and targeted user groups, or more generally, their distinctive theoretical underpinnings and respective goals. Most importantly, these language learning environments are primarily designed for self-study

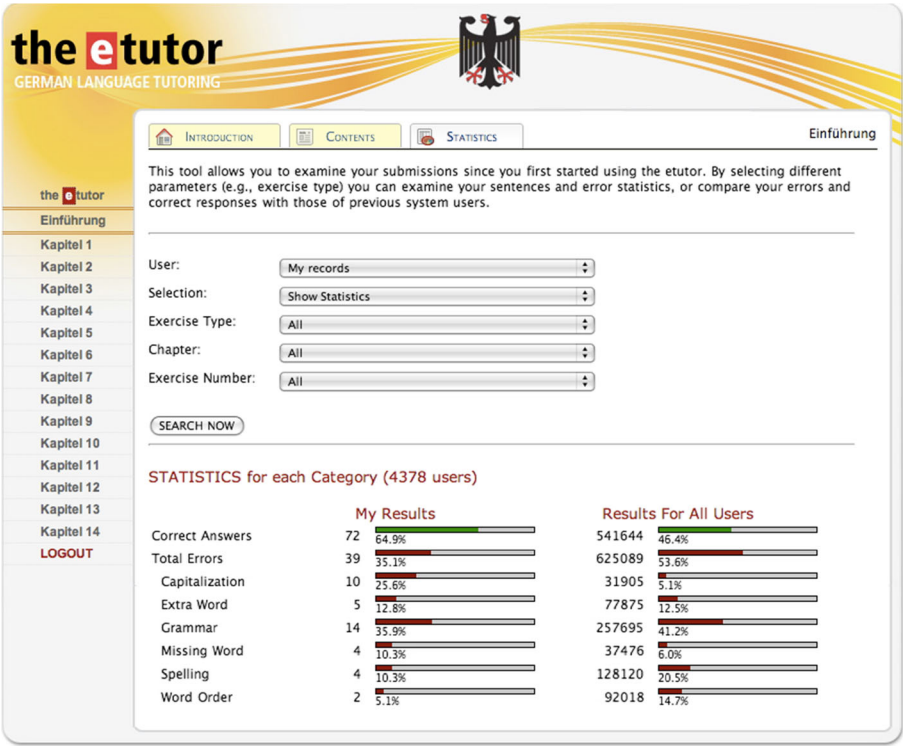

Fig. 5 Comparative performance reports

in that they do not follow a strictly graded curriculum as present in a language course. Any application used in a real language learning classroom, however, needs to keep a record of the student's work and performance that can be shared with instructors and its lexical and grammatical content must follow the same or, at least, a similar sequence as the one reflected by the remaining class materials, i.e., a language learning course book. This is generally not the case with off-the-shelf software that targets user groups outside an academic setting.

The fact that the goals of *German Tutor* were achieved and supported by several research projects conducted with large user groups in real classroom settings (e.g., Heift 2004, 2010a; Rimrott and Heift 2008), however, possibly raised awareness that a stronger emphasis on learner-centered CALL applications for the language learning classroom is needed. Subsequent user studies in similar environments were also perhaps encouraged by the pedagogical issues raised in our 2001 article (see e.g., Amaral and Meurers 2011). By now, researchers have gained significantly more insight into the effectiveness of different types of learner feedback in CALL and learner modeling, even through prototyping applications for research purposes.

More specifically, and in view of the powerful capacity of ICALL systems for generating error-specific feedback to learners, researchers have examined the effectiveness of different types of corrective feedback on L2 grammar to determine its impact on learning outcomes and/or learner-computer interactions. Overall, this line of research has shown that students generally benefit from explicit, error-specific (i.e.,

Fig. 6 Preemptive feedback with a help link to an inflectional paradigm

metalinguistic) feedback because they subsequently perform better on particular target language structures and/or because students' grammatical awareness is subsequently raised. One of the early ICALL studies (Nagata 1993), for instance, investigated different feedback types for learning Japanese particles and found that error-specific metalinguistic feedback that explained the functions and semantic relations of nominal relations in a sentence was more effective than generic, traditional feedback (e.g., *wrong, try again*!). A number of studies followed (e.g., Bowles 2005; Heift 2004; Heift and Rimrott 2008; Murphy 2007; Nagata 1996; Peterson 2010) that generally supported the benefits of error-specific feedback in a CALL environment. Some studies also examined learner error correction behavior in response to distinct feedback types thus focusing on the learning process and behaviour as opposed to learning outcomes. For instance, Heift (2002) identified different learner types, or personas, in the way they engage in error correction and access help options. Her study results indicate that the majority of students (85 %) prefer to work through the error correction process on their own, and to access a correct answer provided by the system only occasionally. Language skill, however, seems to be a determiner in that lower language proficiency learners make more use of system help while high performers may be skipping trivial material (see also Hegelheimer and Chapelle 2000). In another study, Heift (2003) examined different exercise types to determine differences among learners in their practice behaviour of German word order. For all exercise types, students received error-specific feedback. The results indicate that students using a drag-and-drop

interface performed significantly better than those using multiple choice, but only marginally better than the typed-entry group. The more flexible word order practice afforded by the drag-and-drop interface in addition to other benefits such as eliminating typing errors and ease-of-use argue in its favour (see also Heift 2010a; Heift and Rimrott 2012). Finally, from a learner modeling perspective, an ILTS that has been used for several years also provides a rich source of longitudinal data. Heift's (2008) study, for example, provides some evidence that there are noticeable differences among learners with respect to grammar acquisition and these distinctions are not apparent by simply considering aggregate data for all students. Her longitudinal data illustrate the differences and variations found among students as well as the learner's own variability. The data call for a student model that can make distinctions at a very fine-grained level by emphasizing the dynamic and non-linear process of language learning and support the predictions and assumptions made by researchers working in Dynamic Systems Theory and SLA.

Evidently, this line of research shows that an ILTS that is used in regular classroom instruction also provides a rich source for researchers to address and answer a variety of pedagogical and computational questions which in turn inform system upgrades and development. However, from a more practical point of view, the efforts, time, and resources involved in designing, implementing, and maintaining a system for real classroom use are certainly not to be underestimated. Nevertheless, there are a number of ILTSs used for regular classroom instruction by now. For instance, ROBO-SENSEI (Nagata 2009) is a system for learners of Japanese which was developed roughly around the same time as *German Tutor*, or TAGARELA is an ILTS for learners of Portuguese developed a few years ago (Amaral and Meurers 2011). Significant effort has also gone into creating applications for language learning by augmenting language learning materials with additional lexical and morpho-syntactic information for student access (e.g., dictionaries, corpus look-up tools). Similar AI techniques have also been used to make linguistic features of a text more salient and to help students develop their language awareness (e.g., Amaral et al. 2006; Knapp 2004; Wood 2011). While these language tools have a somewhat different scope and application than an ILTS, they share computational algorithms and similar pedagogical underpinnings with their strong focus on the needs of individual learners.

Conclusion

Undoubtedly, significant progress has been made since our article was published in 2001, especially when it comes to the processing of learner input, learning modeling, and the availability of language learning tools more generally. As for *E-Tutor* as a learner-centered CALL program, the user studies which have been conducted with the system have not only informed system upgrades and development but also contributed to the body of research in learner-computer interactions or, CALL, more generally. These results are especially valuable given that the user studies have been conducted in a regular classroom setting over several years which, by now, have also allowed for longitudinal analyses of user-computer interactions.

Nevertheless, language learners would likely benefit from additional, meaningful interactions with an ILTS. While our article from 2001 emphasizes a pedagogical

approach to language tutoring, learners might further benefit from engaging in a dialogue with the system and/or other concurrent users. This would allow learners, for instance, to clarify the sources of errors and feedback during system use. Such a dialogue module coupled with the ILTS we created would likely enhance the learner-computer interactions and further advance research in language learning with technology.

Acknowledgments The author owes deep gratitude to the many students, friends and, last but not least, family members who worked with her on the development of *E-Tutor* over the last decade. Most important is Chris Rolfe, who programmed most aspects of the current version of *E-Tutor* and performed the audio recordings. Parts of this research were also supported by several Social Sciences and Humanities Research Council (SSHRC), Canada, grants.

References

Amaral, L., & Meurers, D. (2007). In C. Conati & K. F. McCoy (Eds.), *User Modeling 2007: Proceedings of the Eleventh International Conference*, Lecture Notes in Computer Science (pp. 340–344). Wien: Springer.

Amaral, L., & Meurers, D. (2011). On using intelligent computer-assisted language learning in real-life foreign language teaching and learning. *ReCALL, 23*(1), 4–24.

Amaral, L., Metcalf, V., & Meurers, D. (2006). Language awareness through re-use of NLP technology. Paper presented at the Pre-conference Workshop on NLP in CALL: Computational and Linguistic Challenges. *CALICO 2006, Honolulu, Hawaii.*

Arneil, S., & Holmes, M. (1999). Juggling hot potatoes: decisions and compromises in creating authoring tools for the Web. *ReCALL, 11*(2), 12–19.

Borin, L., & Dahllöf, M. (1999). A corpus-based grammar tutor for education in language and speech technology *EACL'99. Computer and Internet Supported Education in Language and Speech Technology. Proceedings of a Workshop Sponsored by ELSNET and the Association for Computational Linguistics* (pp. 36–43). Bergen: Association for Computational Linguistics.

Bowles, M. (2005). Effects of verbalization condition and type of feedback on L2 Development in a CALL Task. PhD diss., Georgetown University.

Braun, S. (2005). From pedagogically relevant corpora to authentic language learning contents. *ReCALL, 17*(1), 47–64.

Dimitrova, V., McCalla, G., & Bull, S. (2007). Open learner models: future research directions (Special Issue of IJAIED Part 2). *International Journal of Artificial Intelligence in Education, 17*(3), 217–226.

Ellis, R., Basturkmen, H., & Loewen, S. (2001). Preemptive focus on form in the ESL classroom. *TESOL Quarterly, 35*, 407–432.

Gaskell, D., & Cobb, T. (2004). Can learners use concordance feedback for writing errors? *System, 32*, 301–319.

Granger, S. (2003). Error-tagged learner corpora and CALL: a promising synergy. *CALICO Journal, 20*(3), 465–480.

Hegelheimer, V., & Chapelle, C. (2000). Methodological issues in research on learner–computer interactions in CALL. *Language, Learning and Technology, 4*, 41–59.

Heift, T. (2002). Learner control and error correction in ICALL: browsers, peekers and adamants. *CALICO, 19*(3), 295–313.

Heift, T. (2003). Type or drag, but don't Click: a study on the effectiveness of different CALL exercise types. *Canadian Journal of Applied Linguistics, 6*(3), 69–87.

Heift, T. (2004). Corrective feedback and learner uptake in CALL. *ReCALL, 16*(2), 416–431.

Heift, T. (2008). Modeling learner variability in CALL. *Computer-Assisted Language Learning, 21*(4), 305–321.

Heift, T. (2010a). Prompting in CALL: a longitudinal study of learner uptake. *Modern Language Journal, 94*(2), 198–216.

Heift, T. (2010b). Developing an intelligent language tutor. *CALICO, 27*(3), 443–459.

Heift, T., & Caws, C. (2014). Constructing a data-driven learning tool with recycled learner data. *International Journal of Computer-Assisted Language Learning and Teaching, 4*(4), 79–93.
Heift, T., & Rimrott, A. (2008). Learner responses to corrective feedback for spelling errors in CALL. *System, 36*(2), 196–213.
Heift, T., & Rimrott, A. (2012). Task-related variation in computer-assisted language learning. *Modern Language Journal, 96*(4), 525–543.
Heift, T., & Schulze, M. (2003). Student modeling and ab initio language learning. *System, 31*(4), 519–535.
Heift, T., & Schulze, M. (2007). *Errors and intelligence in computer-assisted language learning: parsers and pedagogues*. New York: Routledge. ISBN 0415361915.
Knapp, J. (2004). A new approach to CALL content authoring. *Doctoral Dissertation*, Universität Hannover, Hannover.
Murphy, P. (2007). Reading comprehension exercises online: the effects of feedback, proficiency and interaction. *Language, Learning and Technology, 11*, 107–129.
Nagata, N. (1993). Intelligent computer feedback for second language instruction. *Modern Language Journal, 77*, 330–338.
Nagata, N. (1996). Computer vs. workbook instruction in second language acquisition. *CALICO Journal, 14*, 53–75.
Nagata, N. (2009). Robo-Sensei's NLP-based error detection and feedback generation. *CALICO Journal, 26*(3), 562–579.
Peterson, K. (2010). Implicit corrective feedback in computer-guided interaction. Does mode matter? PhD diss., Georgetown University.
Rimrott, A., & Heift, T. (2008). Evaluating automatic detection of misspellings in German. *Language, Learning and Technology, 12*(3), 73–92.
Schuster, E. (1986). The role of native grammars in correcting errors in second language learning. *Computational Intelligence, 2*(2), 93–98.
Van der Linden, E. (1993). Does feedback enhance computer-assisted language learning. *Computers & Education, 21*(1–2), 61–65.
Vansant, J., Clyde, M., & Di Donato, R. (2011). *Deutsch, Na Klar! An introductory German course*. Boston: McGraw Hill.
Wood, P. (2011). Computer assisted reading in German as a foreign language. Developing and testing an NLP-based application. *CALICO Journal, 28*(3), 662–676.
Xu, J., & Bull, S. (2010). Encouraging advanced second language speakers to recognise their language difficulties: a personalised computer-based approach. *Computer Assisted Language Learning, 23*(2), 111–127.