Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 84 (2016) 49 – 56
Procedia Computer Science

7th International conference on Intelligent Human Computer Interaction, IHCI 2015

Initial implementation of natural language turn-based dialog system

Alexander Wachtel*, Sebastian Weigelt, Walter F. Tichy

Karlsruhe Institute of Technology, Chair for Programming Systems Prof. Walter F. Tichy, Am Fasanengarten 5, 76131 Karlsruhe, Germany

Abstract

Our prototype implements a natural language dialog system for Excel spreadsheets. The work is motivated by a pilot study which shows that novice users have difficulties with the formula language of Excel and need interactive assistance. JustLingo[1] allows spreadsheet calculations in ordinary, written English. Furthermore, users can extend the system functionality by explaining of the calculations to the prototype. The keyword-based system analyzes the input using natural language processing techniques and translates it into Excel formulas. It asks for missing information such as operands and target cells and provides alternatives if there are ambiguities. It also handles references to previous inputs, allowing step-by-step construction of calculations. An evaluation shows that it properly resolves 82 % of references and correctly interprets 79,5 % of overall user input. Although far from perfect, the prototype demonstrates that computers may soon begin to take on simple, limited-domain programming tasks based on natural-language input.

Peer-review under responsibility of the Organizing Committee of IHCI 2015

Keywords: Natural Language Processing; Natural Language Interfaces; Dialog Systems; End-User Programming; Spreadsheet

1. Introduction

Since their invention, digital computers have been programmed using specialized, artificial notations, called programming languages. However, only a tiny fraction of human computer users can actually work with those notations. An alternative would be ordinary, natural language. Ordinary language would enable almost anyone to program and would thus cause a fundamental shift in the way computers are used. Rather than being a mere consumer of programs written by others, each user could write his or her own programs[2]. The idea of programming in natural language was first proposed by Sammet in 1966[1], but enormous difficulties in this area have resulted in progress being disappointingly slow. In recent years, significant advances in natural language techniques have been made, leading, for instance, to IBM's Watson[3] computer winning against the two Jeopardy! World champions, Apple's Siri routinely answering wide-ranging, spoken queries, and automated translation services such as Google's becoming usable. However, programming in natural language remains an open challenge[4,5].

[1] http://ps.ipd.kit.edu/english/JustLingo.php
* Corresponding author. Tel.: +49-721-608-46320 ; fax: +49 721-608-42817.
E-mail address: Alexander.Wachtel@kit.edu

Peer-review under responsibility of the Organizing Committee of IHCI 2015
doi:10.1016/j.procs.2016.04.065

The first prototype allows users to express spreadsheet calculations in ordinary typed English and converts the natural language input into arithmetic spreadsheet formulas. In general, spreadsheets have been used at least 7000 years for information representation and correlations within the data[6]. Meanwhile, spreadsheet programs such as Excel Microsoft become ubiquitous. It is estimated that each year hundreds of millions of spreadsheets are created[7]. A small knowledge base and an interactive dialog system assist in the accomplishment of computational tasks. The approach uses a limited vocabulary and a small set of syntactic patterns organized as a grammar. Relying on 92 patterns covering all basic arithmetic operations, both syntactic and semantic analysis can be performed simultaneously. This limited range of possibilities enables us to derive most of the semantics from the user query precisely. By recognizing references to previous inputs the system supports step-by-step construction of complex expressions. It also requests missing information and asks the user for clarifications in case of ambiguities. Rather than relying on the completeness of particular inputs, the system establishes a context which makes it usable for inexperienced users.

Perhaps the latter insight is the most significant: In the past, computers were expected to follow instructions blindly, without a notion of right or wrong or what users expected. Natural language programming, on the other hand, requires a domain-aware counterpart that can ask for clarification, thereby overcoming the chief disadvantages of natural language, namely ambiguity and imprecision.

2. Pilot study

Initially, we needed to find out how real users would interact with spreadsheets in natural language. For this purpose, we recruited subjects and asked them to describe how to solve problems with the aid of a spreadsheet. Spreadsheets with data were provided together with the tasks. The subjects were asked to imagine explaining to a human partner how to solve a given task and to write down what they would say. Even though there were no suggestions to ask the imaginary partner for help, many participants did just that. The problems to be solved were selected from the first four chapters of the textbook Excel 2013 Step by Step[8]. The study consists of 35 questions, six of which are mathematical problems (i.e. calculation of sum and average, rounding decimal values, copying an existing formula to other cells, conditional functions) and another six are data and document manipulation tasks (i.e. inserting and sorting of columns, creating tables and diagrams, labeling cells).

2.1. Observations

Overall, there were 57 participants. In a self-assessment, 12 % considered themselves as experts, 72 % advanced users, and 16 % beginners with regard to spreadsheets. The participants' experience on working with Excel was on average 9.5 years, with a maximum of 25 years and a minimum of 1 year. The following discusses a sample scenario from the empirical study that involves arithmetic functions within an Excel worksheet (see Figure 1).

	A	B	C	D	E	F
1	Grades overview					
2						
3	Name	Daily grades 1	Daily grades 2	Daily grades 3	Final exam	Average
4	Marcel	35,7	35,3	20,0	40,0	
5	Andrea	75,0	95,0	65,0	95,0	
6	Simon	100,0	95,0	75,0	85,0	
7	Andrew	90,0	95,6	60,0	80,8	
8	Mike	100,0	35,0	70,0	45,0	
9	Tyler	95,5	100,0	85,1	90,0	
10	Anna	60,0	40,0	35,0	35,0	
11	Mary	80,0	85,0	80,0	90,4	
12	Flinn	65,0	20,0	70,9	40,6	
13	Rick	30,3	20,7	40,0	20,0	
14	Alex	80,0	60,0	85,0	20,0	
15	Average	73,77	61,96	62,36	58,35	

Fig. 1: Example scenario in spreadsheet

The problem to be solved was stated as follows: In the column "Average" the mean of all 4 grades should be calculated for each student. Only the average value regardless of the weighting mentioned before should be calculated. Your task is to perform that calculation specifically for the student Marcel and to write it to F4.

The descriptions given by the participants reflect their experience level. Beginners simply ask for help without any further specification. In that case a system does not get enough information to fulfill the required task:

- *"I need to have a cell that averages the students grades, how would I do that?"*
- *"Friend, will you help me figure out the formula for this problem?"*

Advanced users provide more information for the system. In that case the system can recognize a few parts of the task. Important information is still missing to get the job done:

- *"How can I form the arithmetic average value of an arbitrary amount of values?"*
- *"Take every grade in the row which is named Marcel and calculate the arithmetic mean of it and write the result into cell F4."*

Experts provide complete descriptions. No dialog is needed:

- *"Could you insert the average value of Daily grades 1, Daily grades 2 and Final exam into Average?"*
- *"Please put the average of row 4 column B to E in F4."*

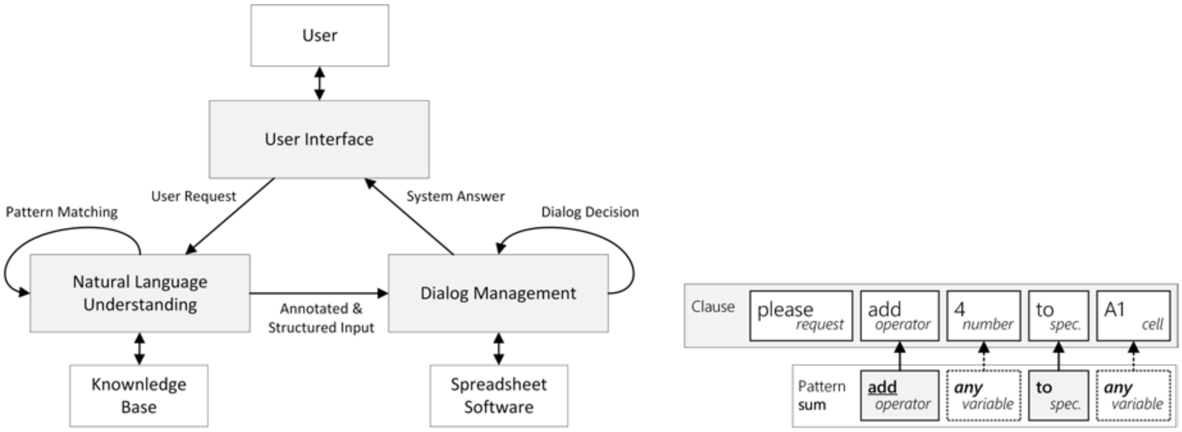
2.2. Results and findings

The study demonstrates that a natural language interface to spreadsheet programming helps inexpert users. The results show that 17 % of experts, 80 % of advanced and 100 % of beginners ask for assistance in at least one task. On average, experts needed help with 1.5 of twelve, advanced users with 3.5 and beginners with 6 tasks. The study provides evidence that spreadsheet calculations are in fact a scenario of sufficient complexity to be of value, especially for novices. However, every user needs assistance with complex spreadsheets. The interactive system should help inexperienced users by asking for missing information because natural language input contains gaps, have the ability to resolve references to already provided information, and successfully resolve ambiguity in natural language input.

3. Natural Language Understanding

The prototype is implemented as an Add-In for Microsoft Excel and looks like a common instant messaging system. The dialog system allows users to express spreadsheet calculations in ordinary typed English and converts the natural language input into arithmetic spreadsheet formulas. All calculations are stored as valid Excel-formulas, which allows future manipulation of the spreadsheet data. A small knowledge base and an interactive dialog system assist in the accomplishment of computational tasks. The approach uses a limited vocabulary and a small set of syntactic patterns organized as a grammar. Relying on 92 patterns covering all basic arithmetic operations, both a syntactic and semantic analysis can be performed simultaneously. This limited range of possibilities enables us to derive most of the semantics from the user query precisely. By recognizing references to previous inputs the system supports step-by-step construction of complex expressions. It also requests missing information and asks the user for clarification in case of ambiguities. Rather than relying on the completeness of particular inputs, the system establishes a context which makes it usable for inexperienced users. The system architecture consists of a user interface responsible for human interaction as well as a natural language understanding and a dialog management unit (see Figure 2a). User request is passed to the natural language understanding for annotation and matching to the keywords contained in the knowledge base. Dialog management builds a tree representation for a valid spreadsheet formula and generates a response. Individual modules are explained in the order in which they generate output that leads to a valid final result. All mechanisms necessary to meet the objectives are described for input *"Please add 4 to A1. Then multiply it by 7."*

In a first step the natural language understanding (NLU) unit performs essential language analysis relying on a basic vocabulary specifically built to cover the system's domain. Synonyms are substituted using a handcrafted synonym database. Mathematical terms and numerical values as well as references to regions within the spreadsheet are tagged. In the following step the system groups elements representing a sentence or clause to enable subsequent analysis to be performed in sets. Following this, the semantic meaning of any annotated sentence is derived. For this purpose patterns were designed with each pattern consisting of a sequence of keywords and placeholders. The latter can either take a single column, row, cell, number, or arbitrary elements of any type. All patterns are compared with each clause matching the keyword.



(a) (b)

Fig. 2: Architecture overview of the prototype (a) and Examplary pattern matching (b)

As part of the given example the pattern *sum* matches the first clause, with keywords *add* and *to*, and placeholders *any* (see Figure 2b). The placeholders are filled with the respective elements (*4* and *A1*). By successively matching additional patterns within the placeholder elements, more complex sentences can be transformed into a semantic representation. It builds a tree structure consisting of operators and operands which makes it easier to determine whether a valid arithmetic expression has been provided. Entirely identified patterns are mapped onto spreadsheet formulas, with the respective placeholders serving as operands, while others have to be processed by the dialog management unit. This approach allows the system to understand everything the user inputs, as long as the compatible pattern already exists in the knowledge database. If the pattern is missing from the knowledge database, the system fails to understand the user input and asks for a new task. Therefore, we enable users to define their own pattern without access to the source code by programming in different steps using the natural language turn-based dialog system. Furthermore, incompletely matched patterns are handled with active dialog.

4. Dialog Management

The purpose of the dialog management unit (DMU) is to deal with the tree structure that has been created by the NLU unit, resolve references, create a valid spreadsheet formula and generate a human-like response to the user input.

Missing information and references. If the tree represents a valid expression, no dialog is needed. In this case the DMU only creates user feedback. The dialog system tells the user that it understands the input and creates a formula, which is also displayed. Depending on the particular user query, the system asks if and where to store the result. In many cases the user input leads to partial trees. If so, the dialog system tries to recover through reference resolution (e.g. "Sum up 4 and 5. Multiply it by 7.", see Figure 3a). On the one hand, the system is capable of identifying explicit references such as *it*, *this result* or *that sum*. Implicit references as in the phrase *Now add 5* on the other hand are likewise recognized whenever a phrase is missing an operand. In that case the system finds the previous tree structure in the dialog history and incorporates the referenced clause at the exact position of the referencing pronoun. Whenever the prototype is not able to find an appropriate reference resolution, it informs the user and asks for the missing information.

Ambiguous information. A major disadvantage of natural language is ambiguity. The prototype deals with ambiguous user input by deriving more than one semantic tree representation. Each of them is mapped onto a spreadsheet formula and displayed to the user (e.g. "Calculate 5 divided by 3 times 2." could be interpreted as either (5/3)*2 = 3.33 or 5/(3*2) = 0.83, which must be resolved, see Figure 3b). The user can choose the desired alternative via dialog (e.g. "Take the first one.").

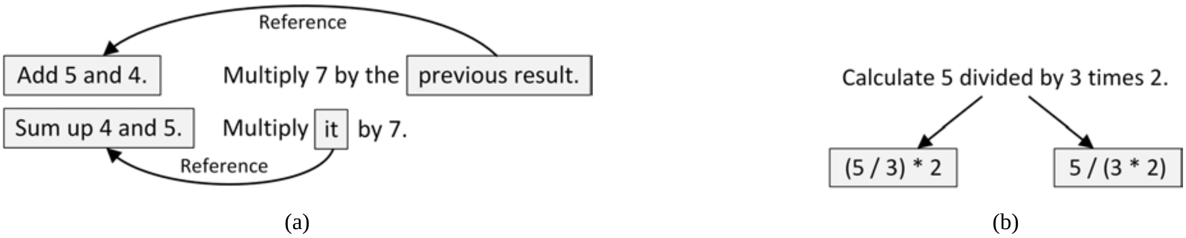


Fig. 3: Resolution of references (a) and ambiquities (b)

End-User Programming. Motivated by the difficult extensibility of the given software systems we implement an interface for automatic generation of new pattern or formula. This interface enables user to describe in ordinary written English the new formula and has been integrated in the previously described dialog system. We implemented a finite automation with four states: General request. In the first step, the user provides the request for creating new formula to the dialog system. Then, the prototype enters into the first state and expects user input for pattern name, user-specified formula and the user order to store the provided pattern in the database. A user-specified formula can not be used for further calculation without a well-defined and unique name. In this step, the prototype checks the already existing database entries for the new name. The dialog system asks the user in this state for the valid formula. Users are allowed to provide a complex formula in several inputs and make use of implicit and explicit references to the previous parts of the formula. If the user input of the formula is complete, it can be saved in the database and used under the provided user-friendly name.

Response generation. The DMU generates a response to the user input as a human-like dialog system. The system output is a set of abstract response segments that will be phrased in natural language. A single sentence is represented by a response segment that abstractly specified all required parameters. Those parameters will be later integrated at a certain position in the response. Each segment consists of one or more rules. A rule consists of one or more alternative phrases that are picked randomly. The same applies for vocabulary entries representing a single word.

5. Evaluation

As natural language is highly dynamic and differs widely among humans, it is necessary to evaluate the dialog system with multiple users. First, a study of a series of mathematical tasks has been done involving 17 participants initially unfamiliar with the dialog system. 15 of them had been using *Microsoft Excel* occasionally for at least five years, while two had never used it. The participants used the dialog system in order to accomplish several spreadsheet tasks. Some tasks built upon one another to encourage the use of references. Most interesting in this context is the promising overall success rate as well as the results regarding the three main objectives (as seen in Table 1). The system is not only successful in resolving references to the previously provided information, but also acts actively and asks relevant questions depending on the dialog context:

System: How can I help you?
User: Calculate 1000 multiplied by the sum of E6 and E7
System: Alright. For computing 1000 multiplied by the sum of E6 and E7 you require the formula (1000*(E6+E7)) which results in 9000. I can store this somewhere in the worksheet if you like?
User: now add 3000
System: For computing 1000 multiplied by the sum of E6 and E7 plus 3000 you require the formula ((1000*(E6+E7))+3000) and the output is 12000. I can write that somewhere in your worksheet?
User: Save that formula in E8
System: Alright, something else then. It was put to E8.

Total	
Number of tasks	170
Dialog iterations (Number of Human-Computer Interaction)	316
Successfully solved	79,5 %
Objectives	
Frequency of system queries	17,1 %
Successful reference resolution	82 %
Successful ambiquity resolution	34,8 %

Table 1: Evaluation result of the dialog system

Afterwards, the participants were asked to subjectively assess the system's overall performance. More than 80% rated the system to be intuitively usable and 70% perceived a distinct additional value for themselves. The queries of the dialog system were considered helpful in 68% of the cases. Novices particular benefited from the dialog. Also noticeable regarding future work is the observed effect that users, although the system has no restrictions, adapt to the prototype within a few requests by using formulations they experienced as processed properly by the system.

Furthermore, another study has been done involving other 10 participants, half of them initially unfamiliar with our prototype. Six of them considered themselves as beginner, three as advanced users and only one as expert regarding spreadsheets. The task was to calculate the zero points of the formula (a) using the quadratic equation, also known as midnight formula (see formula (b)):

$$(a) \quad f(x) = x^2 - 6x + 5 \qquad\qquad (b) \quad x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

This functionality is not available by a single formula in Microsoft Excel. The goal was to create new pattern for calculation and to expand the functionality of the given prototype by using the dialog system. The active dialog system allows users to program new formulas step by step using single lower case characters. After a few human-computer interactions the formula is too complex and can not be written easily by the user without our prototype. After saving the new formula, it can be successfully used in further calculations. As seen in Table 2 the system interactively reacted to the users input and successfully solved about 85% of the given tasks.

Total	
Number of tasks	120
Dialog iterations (Number of Human-Computer Interaction)	641
Successfully solved	85,83 %
Successful Tasks	
User Request	100 %
Name Definition	100 %
Provide New Formula	60 %
Store & Use New Formula	83,3 %

Table 2: Evaluation result of the end-user programming tasks

6. Related Work

Programming in natural language. In 1979 Ballard et al.[10,11,12] introduced their Natural Language Computer (NLC) that enables users to program simple arithmetic functions using natural language. Our prototype extends the idea with a dialog system component for step-by-step construction of complex expression and enables users to perform tasks they otherwise would not be able to accomplish. Although NLC resolves the references as well, there is no Human-Computer-Interaction with a dialog system. User feedback has been only provided by the common result output. NLyze[13], an Add-In for Microsoft Excel that has been developed at the same time as our prototype, behaves in a similar manner.This approach uses a separate domain-specific language for logical interpretation of the user input. Another approach of programming in natural language is the Pegasus system[14] that creates limited executable programs from natural language using an intermediate knowledge and reference representation. Metafor introduced by Liu et al.[15] has a different orientation. Based on user stories the system tries to derive program structures to support software design. A different approach regarding software design via natural language is taken by RECAA[16]. Besides supporting the user while building a natural language specification, the system can automatically derive UML models from the text and also keep model and specification consistent through an automatic feedback component. A limited domain end-to-end programming is introduced by Le. SmartSynth[17] allows synthesizing smartphone automation scripts from natural language description.

Natural Language Dialog Systems. Many dialog systems have already been developed. Commercially successful systems, such as Apple's Siri und Google's Voice Search[18,19] are characterized by the cover of many domains. The reference resolution makes the systems also look very natural. However, there is no dialog interaction besides the results output and the output of the error messages. The Mercury system[20] designed by the MIT research group is a telephone hotline for automated booking of airline tickets. Mercury guides the user through a mixed initiative dialog towards the selection of a suitable flight based on date, time and preferred airline. Furthermore, Allen[21] describes a system called PLOW developed at Stanford University. As a collaborative task agent PLOW can learn to perform certain tasks, such as extracting specific information from the internet, by demonstration, explanation, and dialog. Geiselmann [6], [7] presents the implementation of a dialog system for a robot and observes that most of the misunderstandings in human robot communication are due to new formulations and missing mechanisms to deal with meta-communication and elliptical utterances.

End-User Programming. The main question in this area of research is, how to allow users to interact with the computer more easily and allow general users, who are not software developers and have no access to the source code, to program a computer system[22]. In 2009 Myers[23] provides an overview of the research in the area of End-User Programming. Nearly 90 million people in US use computers at work, 50 million use spreadsheets at work, 12 million considered themselves programmers in a self-assessment and only 3 million people are professional programmers. As Myers summarized many different systems for End-User Programming have been already delevoped - Whyline, Visual Programming or programming by example and programming by demonstration for spreadsheets. However, from our knowledge point, there are no End-User Programming systems such as our prototype that can be manipulated by the user with natural language. During a study in 2006 Ko[24] identifies six learning barriers in End-User Programming systems: Design, Selection, Coordination, Use, Understanding and Information barriers. Peyton Jones[28] encapsulates a computation in spreadsheets as a function and proposes a mechanism to define functions using only standard spreadsheet cells, formulas and references. Sestoft[29] extends it by increasing expressiveness and emphasizing execution speed of the functions thus defined by supporting recursive and higher-order functions, and fast execution by a careful choice of data representation and compiler technology. Burnett[26] shows with research language Forms/3 that procedural and data abstraction, and graphics output can be supported in the spreadsheet paradigm. Cunha[27] realizes techniques for model-driven spreadsheet engineering that employs bidirectional transformations to maintain spreadsheet models and synchronized instances. Begel[25] introduces voice recognition to the software development process. His approach uses program analysis to interpret speech as code, thereby enabling the creation of a program editor that supports voice-based programming.

7. Conclusion and future work

We present the first implementation of prototype for an assistant usable in rather technical domains that uses natural language understanding, programming in different steps and active dialog management system and allows inexperienced users to manipulate spreadsheet data by using the natural language. Motivated by a pilot study the system is able to resolve ambiguities by providing alternatives to choose from. Furthermore, the dialog system handles references to previous results and expressions, allowing the construction of complex expressions step-by-step. The prototype also creates new formulas by using End-User Programming concepts. The evaluation of the prototype exceeded expectations. The system helped users to solve tasks and received positive feedback from nearly two thirds of the users. Inspired by the Turing Test[9], we asked 17 independent spreadsheet users to formulate requests for particular calculation tasks. Each task was answered by both, the prototype and a human, independently. Afterwards the participants were encouraged to identify the computer generated response. This however turned out to be surprisingly hard to decide. With 34 decisions made in total, 47.1% falsely identified the dialog system answer as human. This result indicates that the prototype is capable of generating suitable responses for sufficiently specific requests within the language domain.

However, plenty of work still needs to be done - the implemented tasks are quite simple. Programming languages assist with repetitive tasks that involve use of loops and conditionals. This is what is often challenging for spreadsheet users. The goal is to implement Excel scripts called macros from natural language input. We are exploring ways to extend the patterns with the help of the dialog system. Furthermore, the system needs to be extended for handling graphs and charts. Another open challenge is to switch from typed to spoken user inputs. Our first implementation uses the internal speech recognition tool in Windows Operating Systems, but still needs improvement. Our prototype is a stepping stone on the path to general programming in natural language, both written and spoken. Programming today requires years of training. With natural language, programming would become available to everyone. We believe that systems like our prototype are a reasonable approach for end user software engineering and will therefore overcome the present bottleneck of IT proficients.

References

1. Jean E. Sammet. The Use of English as a Programming Language. Communication of the ACM, Volume 9, March 1966.
2. Walter F. Tichy. Universal Programmability - How AI Can Help. Artificial Intelligence Synergies in Software Engineering. May 2013.
3. David Ferrucci. Building Watson: An Overview of the DeepQA Project. Association for the Advancement of Artificial Intelligence. 2010.
4. Hugo Liu. Toward a programmatic semantics of natural language. Visual Languages and Human Centric Computing. 2004.
5. Charles L. Ortiz. The Road to Natural Conversational Speech Interfaces. IEEE Internet Computing. March 2014.
6. Matthew Hurst. The interpretation of tables in texts. PhD. 2000.
7. Robin Abraham. Header and Unit Inference for Spreadsheets Through Spatial Analyses. 2004.
8. Curtis D. Frye. Microsoft Excel 2013, Step by Step. O'Reilly Media, Inc. 2013.
9. Alan M. Turing. Computing machinery and intelligence. 1950.
10. Bruce Ballard. Programming in natural language: NLC as a prototype. Association for Computing Machinery (ACM). 1979.
11. Alan Biermann. Toward Natural Language Computation. American Journal of Computational Linguistics. 1980.
12. Alan Biermann. An experimental study of natural language programming. Int. J. Man-Machine Studies. 1983.
13. Sumit Gulwani. NLyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. SIGMOD 2014.
14. Roman Knöll. Pegasus - First Steps towards a Naturalistic Programming Language. Darmstadt University of Technology. 2006.
15. Hugo Liu. Metafor: Visualizing stories as code. 10th international conference on Intelligent user interfaces. 2005.
16. Sven Körner. Transferring Research Into the Real World - How to Improve RE with AI in the Automotive Industry. August 2014.
17. Vu Le. SmartSynth: Synthesizing Smartphone Automation Scripts from Natural Language. MobiSys. 2013.
18. Jerome R. Bellegarda. Spoken Language Understanding for Natural Interaction: The Siri Experience. Springer New York. 2014.
19. Jason D. Williams. Spoken dialogue systems: challenges and opportunities for research. December 2009.
20. Stephanie Seneff. Response planning and generation in the MERCURY flight reservation system. 2002.
21. James Allen. PLOW: A Collaborative Task Learning Agent. Association for the Advancement of Artificial Intelligence. 2007.
22. Henry Liberman. End-User Development: An Emerging Paradigm Human-Computer Interaction Series, Volume 9. 2006.
23. Brad A. Myers. Invited Research: Overview End-User Programming. CHI. 2006.
24. Andrew J. Ko. Six Learning Barriers in End-User Programming Systems. Visual Languages and Human Centric Computing (VLHCC). 2004.
25. A. Begel, Spoken Language Support for Software Development, Ph.D. Thesis, Berkeley, 2005.
26. M. Burnett. Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm. March 2001.
27. J. Cunha. Bidirectional Transformation of Model-Driven Spreadsheets. Springer Lecture Notes in Computer Science. 2012.
28. S. Peyton Jones. A user-centred approach to functions in excel. 2003.
29. P. Sestoft. Sheet-defined functions: Implementation and initial evaluation. 2013.