Human Tutorial Instruction in the Raw

YOLANDA GIL, University of Southern California

Humans learn procedures from one another through a variety of methods, such as observing someone do the task, practicing by themselves, reading manuals or textbooks, or getting instruction from a teacher. Some of these methods generate examples, which require the learner to generalize appropriately. When procedures are complex, however, it becomes unmanageable to induce the procedures from examples alone. An alternative and very common method for teaching procedures is tutorial instruction, where a teacher describes in general terms what actions to perform and possibly includes explanations of the rationale for the actions. This paper provides an overview of the challenges of using human tutorial instruction for teaching procedures to computers. First, procedures can be very complex and can involve many different types of interrelated information, including: 1) situating the instruction in the context of relevant objects and their properties, 2) describing the steps involved, 3) specifying the organization of the procedure in terms of relationships among steps and substeps, and 4) conveying control structures. Second, human tutorial instruction is naturally plagued with omissions, oversights, unintentional inconsistencies, errors, and simply poor design. The paper presents a survey of work from the literature that highlights the nature of these challenges and illustrates them with numerous examples of instruction in many domains. Major research challenges in this area are highlighted, including the difficulty of the learning task when procedures are complex, the need to overcome omissions and errors in the instruction, the design of a natural user interface to specify procedures, the management of the interaction of a human with a learning system, and the combination of tutorial instruction with other teaching modalities.

Categories and Subject Descriptors: **H.5.m [Information interfaces and presentation]**: Miscellaneous

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Procedure learning, intelligent user interfaces, natural language interpretation, instruction, end user programming, interactive learning

ACM Reference Format:

Gil, Y. 2015. Human Tutorial Instruction in the Raw. ACM Trans. Interactive Intelligent Systems. 5(1), 2015.

1. INTRODUCTION

End users today are able to create numerous applications such as spreadsheets, web sites, and games. How can they do this with no programming background? They are empowered by interfaces and languages that are designed for a given type of task and are natural to use. These interfaces and languages are not necessarily simple, they can be quite complex and although they may require some effort to learn they are learnable within reason. End user programming interfaces have a very different flavor from programming environments, as they are more focused on concepts and

This work was supported by the Defense Advanced Research Projects Agency under grant HR0011-07-C-0060, by the Air Force Office for Scientific Research under award FA9550-11-1-0104, and by the National Science Foundation under award IIS-1117281.

Author's address: Y. Gil, Information Sciences Institute and Department of Computer Science, University of Southern California, 4676 Admiralty Way, Marina del Rey CA 90292, USA. Email: yolandagil@acm.org.

design rather than on editing raw code. Yet, we are far from interfaces that allow non-programmers to specify the kinds of tasks that they would like computers to automate for them.

[Kay 84] reminds us of the early days of programming, when code was mysteriously produced by a few select and painfully trained individuals. A bold estimate in 2000 anticipated that by 2005 there would be 2.75 million professional programmers in the United States, and 55 million end-user programmers [Boehm et al 00]. A more recent estimate predicted 90 million end user programmers by 2012 in the US alone [Scaffidi et al 06]. Of those, 55M would use spreadsheets and databases, while only 13M would describe themselves as programmers. [Adams 08] argues that programming applications have moved from having dozens of markets of millions of users (focusing on large software applications of universal appeal) to having millions of markets each with dozens of users (focusing on reusing generic services and components). The future may be one where there will be millions of markets of one user, each representing a personal need that cannot be fulfilled by off-the-shelf code developed by someone else. This would be a manifestation of what is known as the "long tail of programming" [Anderson 08].

The successful examples of end-user applications today typically focus on data manipulation through spreadsheets and web forms. However, there are no practical approaches that allow end users to specify procedures to process data, or to control a physical environment. Some approaches have been designed to learn procedures from examples provided by end users through demonstrations or through observation [Li et al 10; Castelli et al 10; Chen and Weld 08]. From a few demonstrations, a system induces a general procedure that generalizes from the particulars of the examples shown by the user. However, when procedures are complex it is hard to create demonstrations that cover the space of possible generalizations particularly if the user is to provide only a few examples.

A complementary approach is to teach procedures through tutorial instruction, a method commonly used by people to teach procedures to other people. In tutorial instruction, the teacher provides a natural language description of procedures using general situations and abstract objects [Clark et al 01; Webber et al 95]. This is in contrast with situated instruction or demonstrations where a particular state is used to illustrate the procedure [Huffman and Laird 95; Thomaz and Breazeal 08a]. Tutorial instruction is a concise way to communicate complex procedures, and can be supplemented with demonstrations or practice to improve learning [Fritz and Gil 11]. Tutorial instruction is a common form of instruction, and there are many Web sites that provide this kind of instruction to convey all kinds of procedures (e.g., http://www/wikihow.com, http://www.ehow.com, and http://howto.cnet.com).

Designing systems that can learn procedures from human tutorial instruction raises many research questions. What form can tutorial instruction take? What kinds of information will need to be conveyed to the system during instruction? In what modality would this information be conveyed? Will instruction provided in a natural way by a human contain adequate information for the system? What capabilities would such a system be expected to have?

This article aims to provide a better understanding of the challenges we face in developing systems that learn from human tutorial instruction by surveying previously published research that is scattered in the literature. The literature on instruction is vast, ranging from education approaches, tutoring techniques, psychological models of students, education software, the writing of educational materials, and natural (end-user) programming to name a few. This document draws from three major areas: cognitive science, end user programming, and natural language processing. The document is not organized by discipline, rather it

assembles findings from these disciplines from the point of view of the research challenges in learning from human tutorial instruction. It is not meant to be an exhaustive compilation, but rather to extract major lessons learned from seminal and representative papers.

The article is organized around two major sources of complexity in learning from tutorial instruction: the diversity of information that needs to be used to convey procedures, and the challenges that arise when human instruction is faulty.

The first theme is that extracting procedures from human instruction is challenging because they include many kinds of knowledge that need to be appropriately interrelated by the teacher and comprehended by the learner. As [Donin et al 92] explain:

> "Writing instructions for complex procedures is by no means an easy task (witness the great variability in quality of technical manuals and written instructions). The complexity of this task is due at least partly to the fact that **procedures themselves are complex relational structures and the mapping between these structures and a linear sequence of propositions expressed in discourse is not easy to define**."

To understand the sources of that complexity, we turn to studies of human cognition and the ability of learners to understand instruction in alternative forms that appear equivalent in content. Our focus is not to learn about cognitive limitations of human students, but about how human instruction is designed in practice to accommodate human learners. In other words, human teachers know how to teach human learners, so their instructions to a system will likely be designed with a structure that they are used to provide to human learners.

A second theme of the article is that procedures are challenging to teach because in order for end users to be able to provide instruction they need to be able to express it in a form that is natural for them to teach. This results in instruction plagued with ambiguity, omissions, and errors in the instruction, since natural human instruction is more often than not poor instruction. An average human as compared to a professional teacher will make many gaffes, including omitting important information and giving instructions that maybe easier to misinterpret. [Miller 81] speculates as a result of multi-year studies (emphasis is ours):

> "We speculate that […] **the direct translation of natural language programs into formal computer programs may be feasible only for rather simple problems**; for more complex ones we could envision as being necessary much more complicated interactive processes intervening between the subjects' initial specifications and their ultimate interpretations […]. This point of view assumes that people in general can develop solutions for problems of even high complexity, and it is just the manner in which they express the solutions that can cause translation difficulties. Another view – certainly not counter-indicated by our present data – is that **the locus of difficulty may well be conceptual, not expressional; that is, maybe subjects' solutions decrease in completeness with complexity** because subjects are less and less able to formulate conceptually adequate solutions, regardless of whether they are expressed in "thoughts," natural language, or computer programs."

1. **Background information**
   - Situation information
   - Device models
   - General principles
2. **Details of procedure steps**
   - Type
   - Objects and modifiers
   - Conditions
   - Effects
3. **Relationships among steps**
   - Ordering
   - Goal statements
   - Goal decomposition schemas
   - Structural schemas
   - Functional schemas
4. **Control structures to organize steps**
   - Iterations
   - Conditionals
   - Choices
   - Advice and policies
   - Exceptions

Fig. 1. Broad categories of information that appear in human instruction.

We draw from the literature on natural language processing for research on several analyses of corpora containing diverse written instructions. This helps to understand the sources of difficulty in interpreting naturally occurring instruction. We also draw from research on end-user programming, also known as natural programming, looking at how people express instructions that are to be implemented by a computer. There are many practical lessons learned from developing interfaces that enable people to instruct computers for tasks of varying complexity. There are also many field studies of how people approach programming and what programming concepts are more challenging for people to understand and therefore to teach.

As we review the literature, we give many real examples along the way. The examples include procedures to manipulate physical systems as well as procedures that could be implemented through software agents to manage web sites or personal devices (e.g., smart phones). Some examples are from instructional documents, such as manuals, and others are instructions given in an interactive dialogue setting. Through these examples we can best illustrate the challenges of learning from human tutorial instruction in more concrete terms.

The article begins describing the types of information that appear in instruction, grouped into four broad categories: background information, procedure steps, organization of the instruction, and control structures. Section 3 gives an overview of research and studies that reveal shortcomings in human instruction due to errors and omissions, organized along those four categories. Section 4 presents major research challenges in this area stemming from the complexity of the procedures to be learned, omissions and poor structure of the instruction, unrealistic assumptions on the student's background knowledge and skills, people's attitudes towards teaching computers, and lack of important teaching skills in people.

Table 1. Situated instructions are grounded on scenarios, in contrast with more general instruction (from [Alterman et al 91]).

General instruction	"Insert card"	"Make a phone call"
Situated instruction	"Insert Visa card in an ATM"	"Make a call from payphone"
		"Make a call from home"

Q: How do you go about buying an item for the office?
A: You mean something small like a paper holder?
Q: Yes, what do you do?
A: Well, first I have to find a catalog, an office equipment catalog, that lists the paper holder. When I found it in the catalog, I put down the vendor, the part number, the phone number of the vendor and so on… all that stuff in the purchase order…
Q: How do you continue?
A: Hmm, I'll call the vendor, they mostly have an 800 number, and ask for the current price... […] I'd put the price down on the purchase order, too…, hmm, and then I'd mail the purchase order to the propriety department… and I'd have to file a copy of the purchase order in our own books.

Fig. 2. Situated instructions illustrate the introduction of objects (from [Mahling and Croft 88]).

2. INFORMATION ABOUT PROCEDURES CONVEYED IN TUTORIAL INSTRUCTION

Tutorial instruction can include the broad categories of information, summarized in Figure 1. We describe throughout the rest of this section each type of information in turn, illustrating them with excerpts from real instructions in a variety of domains extracted from the literature.

2.1 Background Information

Background information includes descriptive statements about objects and situations that are relevant to the instruction and are assumed to be known by the student in order to understand the procedural instruction proper.

Situated instruction describes a procedure in a particular usage context. Whereas a demonstration of a procedure uses a specific state, situated instruction reflects a class of states rather than a specific one. This class of states is often referred to as a *situation* or a *scenario*. Whether instruction is situated or not is a matter of degree, one can imagine a whole lattice of procedure abstractions that are more or less situated. Table 1 shows a situated instruction where specific kinds of objects and properties are introduced, such as a card that is a VISA and an ATM.

Situated instruction requires the use of bindings and constraints for each of the steps in the procedure.

Studies have found that examples are preferred by learners when given both options, perhaps because they take less effort to process than instruction [LeFevre and Dixon 86]. However, teaching complex tasks is harder using examples alone. Situated instruction is a good compromise, since it is still based on generalities but grounded on specific situations [Mahling and Croft 88]. Figure 2 shows an example of situated instruction where objects are introduced by the teacher.

Procedure execution:

1. Pick-up receiver – executed
2. Listen for dial tone – executed
3. Dial -- executed
4. Listen for ring tone – failed

   Instruction provided:

   "The call you have made requires an initial deposit. Please hang up momentarily. Listen for dial tone, deposit the required coin, and dial your call again.

Fig. 3. Instructions can be given in response to execution failure and be situated in the context of the execution state (from [Alterman et al 91]).

A form of situated instruction may be given when executing a learned procedure. In this case, the execution occurs in a specific situation or state, but the instruction is provided in a more generic manner but still be situated in a way that is generalizing the specific state of the failure. Figure 3 shows an example.

Ontological information has been shown to facilitate the appropriate representations for learning new material [Slotta and Chi 06]. Hierarchical schemas are often descriptive background information about the task, in contrast with information that reflects the operation of the procedure [Steedhouer et al 00].

A common kind of descriptive information is about devices that need to be manipulated by the procedure. **Device models** have been shown to increase the rate and accuracy of learning, recall, and execution [Kieras and Bovair 84]. The key information in device models that facilitate instruction of procedures is about the specific configuration of the device, rather than general principles or motivation. That is, information that supports direct inference about the steps needed to operate the device.

Instruction may also include the presentation of **general principles** as background, which can be elaborated and generalized by the student to build procedures. Instruction may convey general domain-independent strategies that can be adapted very effectively by students [Chi and VanLehn 08]. This results in very versatile knowledge that can be applied to a variety of tasks. This is useful knowledge for non-recurrent tasks or complex situations that typically require drawing from general principles and background knowledge to select appropriate steps and design appropriate procedures. Alternatively, instructions can be given to describe many procedures that represent recurrent, routine tasks that each time have the same underlying structure and actions to be carried out.

In preparation for instruction, sometimes techniques are used to recall or develop the background information necessary for the lesson [Schwartz and Bransford 98] so the learner is better prepared to process the instruction.

2.2 Details of Procedure Steps

Information about steps is typically given in linear sequence. An example is shown in Figure 4.

Several kinds of information about steps can be specified: the kind of step to be taken, the objects to be used, constraints on those objects, step orderings, and enabling conditions among steps. These types of information are illustrated in Table 2.

The **type of step** to be taken can be indicated explicitly, or implicitly by mentioning some condition or state that hint to that action. Examples of each are (from [Dixon et al 88]):

1. Lift the receiver
2. Wait for the tone
3. Enter #87
4. Enter your identification code
5. Enter #
6. Wait for the tone
7. Enter *81*
8. Enter the appropriate gate number
9. Enter #
10. Wait for the tone
11. If you want to open more gate numbers: repeat steps 8. 9. 10
12. Enter #
13. Put the receiver down

Fig. 4. Instructions given as explicit steps for how to open gate numbers (from [Steehouder et al 00]).

Table 2. Instructions given as steps contain different types of information (from [Young 99]).

Type of information	Example
Type of step	Login
Objects and modifiers	Check out the student handbook from the circulation desk using your student ID
Ordering	Go to the registrar's office, then submit your form to the registrar
Conditions and effects	Pay your fees so that you can register for classes

Explicit: "Remove the diffuser and then unscrew the lightbulb."
Implicit: "With the diffuser off, unscrew the lightbulb."

[Dixon et al 88] showed that explicit actions are interpreted as important, while implicit actions are interpreted as lower level details of the procedure. In other words, the format of the instruction is used as a cue to discern the relative importance of steps. A learner who has significant background knowledge relies less on this kind of cue and more in their judgment, but when learners lack background about the instruction task the form of the instruction can affect their understanding and performance.

Objects and **modifiers** refer to the objects relevant to performing an action. Objects here refer not just to physical objects but to any constant or conceptual constraint on the form or qualification of the action. Some researchers have classified relations between objects and actions in case frames [Fillmore 68], where an action corresponds to a frame and each object has a role or fulfills a case in that frame. Case frames have been typically used for language interpretation and generation [Baker et al 98] rather than for reasoning or learning about process representations. Modifiers qualify the action, for example with temporal or resource constraints. For example, duration estimates and resource selection have been found to be important to describe specific types of processes such as project management [Pietras and Coury 94].

Conditions and **effects** may or may not be expressed in instruction, and when they are they are expressed in a variety of ways. This is exemplified in Figure 5. There are a variety of taxonomies of conditions and effects both in the linguistic and knowledge representation literature, though the instruction often does not explicitly state how the condition or effect must be interpreted or represented [Linden 94; Kosseim and Lapalme 95; Linden and Martin 95; Di Eugenio 98].

- *If a light flashes red*, insert credit card again.
- *When the 7010 is installed and the battery has charged for twelve hours*, move the off/stby/talk switch to stby.
- The battery low indicator will light *when the battery in the handset is low.*
- Return the off/stby/talk switch to stby *after your call.*
- *First, make sure the handset and base antennas are fully extended.* Then set the off/stby/talk switch to talk.

Fig. 5. Alternative expressions of step conditions with different meanings (from [Linden 94]).

A study by [Mahling and Croft 88] shows that a description of a situation (i.e., a pre-situation) can be used to recall procedures, concluding that human learners are able to infer preconditions that trigger a procedure since they were not taught those preconditions. The study also shows that learners were not able to fully describe the effects of steps when asked to do so (i.e., a post-situation), however when given a specific statement they knew whether it was an effect of a given step or not. Therefore, human learners are aware of the effects of steps even if the instructions may not specify them and they have to be inferred.

Humans may have alternative steps for a procedure, perhaps a prototypical one and several alternatives [Mahling and Croft 93].

2.3 Relationships Among Steps

Organizational information provides an expectation for how to interpret other information in the instruction and understand the relationships among steps.

Studies have shown that humans have difficulties processing and using instruction that only contains step information and does not offer a way to organize the steps. Step information is often missing the logic connections behind the steps [Steehouder et al 00]. This makes it harder for people to transfer what they have learned into other domains by making correspondences and analogies [Smith and Goodman 84; Eylon and Reif 84]. It is also harder for people to recall instructions given as a sequence of steps only [Smith and Goodman 84]. This might be an indication that people have difficulties inferring completely the missing organizational information, and therefore we could expect that making these inferences will also be challenging for computers.

The **ordering** of steps is typically implicit in that steps are listed one clause or sentence after another. Instruction may state a particular linear order when in fact many alternative orderings will work, and the alternatives may have to be derived by the student. Partial orderings may be indicated explicitly in the instruction. Concurrent execution of steps is also possible. Examples from [Linden and Martin 95] are:

> Sequence: "Firmly grasp top of phone handset and pull out."
> Concurrent: "Press and hold the mouse button while you move the mouse."

Goal statements can be considered a very simple kind of organizational information, providing useful context to interpret the steps in the instruction. In some cases the goal statement is given first, and in other cases the steps are given first [Steehouder et al 00; Dixon 87]. Figure 6 shows examples that mix step descriptions with organizational information. Note that this is also shown in some of the examples of the previous section. [Dixon 87; Dixon 82] found that human learners process much faster instructions that provide first an overview of the goals of the procedure and then details on each of the steps. A possible assumption is that

"After you program all channels, press the ENTER button to restore the normal operation function."

"Initiate the clock setting by pressing key 1 and 4 simultaneously."

"Push the timer button. The setting will be activated."

"If you want to add page headers and footers to the printed overview, click on Head/Page in the Image menu. Next, click on Handouts, and select the preferred options.

"You can make a wagon by drawing a big rectangle with two circles underneath."

"By drawing a long rectangle with two circles underneath you can make a wagon."

"This will be a picture of a wine glass. Draw a triangle on top of an upside-down T."

"To make a wagon draw a long rectangle with two circles underneath".

Fig. 6. Instructions containing organizational information given as a goal statement shown as underlined text (from [Steehouder et al 00] and [Dixon 87]).

Table 3. Goal statements often convey different types of key information for performing steps (from [Webber et al 95]).

Type of information	**Example**
Endpoint of a step	"Blot with clean tissues to remove any liquid still standing."
Timing between steps	"Sprinkle liberally with salt to extract the liquid that has soaked into the fabric. Then vacuum up the salt."
Enablement between steps	"Go over to the mirror to straighten your bow tie."
Partial enablement between steps	"Depress vacuum canister door release button to open door and expose paper bag."
Addition of steps	"Steam for two minutes to open mussels."
Expectations about state	"Use a screwdriver to open the paint can."

goal statements provide a framework for interpreting step information. The learners follow a guessing strategy, where they attempt immediately to guess the relationships between the steps. They spend extra time generating those guesses as well as possibly correcting their interpretation once the organizational portion of the instruction is given. Learners were often found to fail at such corrections and therefore have errors in performing the learned task. The harder it is to interpret the steps correctly, the more advantageous it is to provide the goal statements first. This view is supported by the work of [Di Eugenio and Webber 96], where the clause describing a step and the clause describing a goal mutually constrain one another.

Goal statements can be given in an action-oriented form (i.e., the accomplishment of an action or task) or in a state-oriented form (i.e., the accomplishment of a condition in a state). Goal statements can be seen as a simple case of organizational information which will be described in the next section.

Goal statements often include key information or constraints for the steps. Table 3 shows some examples. In some cases the constraints are implicit and the student must derive them [Webber et al 95]. For example, in "Depress vacuum canister door release button to open door and expose paper bag," the instruction does not mention that the vacuum door must be open in order to expose the paper bag, which only happens if an additional action is performed either by pulling the door open or by pushing the button while the vacuum is horizontal so the door falls with gravity.

Steps only

How to Replace a Flat Tire

1) Get a screwdriver
2) Use it to pry off the hubcap
3) Get a wrench
4) Use it to loosen the bolts that hold the wheel onto the rim
5) Get a jack
6) Place it under the car on the side of the damaged wheel
7) Raise the car

Goal Decomposition Schema

How to Replace a Flat Tire

A. The first goal is to remove the damaged wheel
- a) To accomplish this, you need to slacken the bolts that hold the wheel onto the rim
  - i) Before you can accomplish the latter, you need access to the bolts
    - 1) Get a screwdriver
    - 2) Use it to pry off the hubcap
  - ii) Now you can loosen the bolts
    - 1) Get a wrench
    - 2) Use it to loosen the bolts
- b) To get the damaged wheel off, you need to raise the car high enough to pull the wheel off
  - i) To raise the car, use a jack
    - 1) Get a jack
    - 2) Place it under the car on the side of the damaged wheel
    - 3) Raise the car

Fig. 7. Instructions for replacing a flat tire with steps only shown left and using a goal decomposition schema shown right (from [Smith and Goodman 84]).

In general, organizational information provides a framework for understanding how step information and other information fits into the procedure being taught, highlights what is important, and guides recall of prior knowledge that might be relevant to the instruction. It can be thought of as a schema that can be used as a roadmap to fit the specific steps of the instruction. This kind of information is common in all sorts of narratives and is often called expository or explanatory schemata [Britt and Larson 03].

The typical form of organizational information is as a **goal decomposition schema**. Although the procedure to be executed is a linear or partially ordered sequence of steps, a group of steps may accomplish higher-level goals that can themselves be grouped. As a result, there may be several levels of decomposition in the hierarchy. An example from [Smith and Goodman 84] is shown on the right hand side of Figure 7, contrasted with using only steps as in the left hand side.

Organizational information may be based on other kinds of information besides goals. A structural schema relies on the structure or components of the object of the instruction. A functional schema provides information stemming from the function that the object of the instruction. Figure 8 shows an example. Notice that some levels state general principles that are instantiated at lower levels. For example, statement V is an instantiation of statement III, and statement VI is an instantiation of statement IV.

Organizational information is referred to as semantic level instruction, while step information is often referred to as syntactic level instruction [Steedhouer et al 00]. Another way to look at the difference is that step information is tactical in that it contains information necessary for immediate execution of the procedure, but organizational information is more strategic in that its intention is to enable the learner to understand the context of the procedure and facilitate learning, recall, reuse, and transfer. It also facilitates failure recovery when unexpected situations arise.

Structural Information

I.	You will construct an electrical circuit that will light a small lamp when you press a switch.
II.	The components of the circuit will be installed in the yellow plastic console.
III.	Assembling a circuit requires that you get the major components ready, then connect them.
IV.	It is often the case that the components themselves have to be assembled first.
V.	The circuit has three major components (1) battery, (2) switch, and (3) small lamp.
VI.	As a way of starting things off, we will first have you assemble the battery.
VII.	In this case, the main components of the battery consist of two dry cells.
VIII.	And the minor components of the battery consist of wire, nuts, and bolts.
IX.	The first things that you will do is to wire together two bolts that will be placed in contact with the dry cells.
1.	Select the short red wire that has been stripped at both ends.
2.	Now you are to wrap one end of the wire around one of the short bolts.
3.	Next you are to wrap the other end of wire around another one.

Functional Information

I.	You will construct an electrical circuit that will light a small lamp when you press a switch.
II.	The components of the circuit will be installed in the yellow plastic console.
III.	In a circuit, electrical current flows from a source to a "consumer" (i.e., to something that requires current, like a lamp).
IV.	Current can flow only when the circuit's components are interconnected in a complete circle, each connection being made by a wire or other metal object that conducts electricity.
V.	This circuit's major components are battery, the source of the current; a lamp, the main consumer; and a switch, which in ON position forms a connection that allows current to flow.
VI.	The battery itself consists of two dry cells, and it is these dry cells that are the source of the current.
VII.	The dry cells have to be connected so that current can flow from the negative pole of one cell to the positive pole of the other.
VIII.	The first thing you will do is to make the wire connection that will later be used to link the two dry cells.
1.	Select the short red wire that has been stripped at both ends.
2.	Now you are to wrap one end of the wire around one of the short bolts.

Fig. 8. Instructions for assembling a circuit, using structural information on the left and functional information on the right (from [Smith and Goodman 84]).

2.4 Control Structures to Organize Steps

Control structures represent non-sequential combinations of instructions. They take a variety of forms in instruction, including iterations, conditionals, decision points, advice, and exceptions.

Iterations often appear in instruction. However, loop constructs are not always the preferred format of iteration. Rather than using loop constructs, instructions tend to indicate how groups of objects are often processed in aggregate operations [Myers et al 04]. For example:

"Move everyone below the 5th place down by one."

When iterations are specified as loops, particular expressions are preferred. Objects are typically processed as lists, rather than modeled as array structures with indices. Iterations over a list of objects more often take the form of taking an element, checking it and terminating the iteration if appropriate and otherwise processing it [Soloway et al 83]. This is in contrast with many programming languages that pick an initial element from the set and then loop over processing an element and then pick the next element to end the loop. [Onorato and Schvaneveldt 87] show that experienced programmers are much more likely to use loop constructions than other subjects, even when communicating with other humans.

GOAL: Select-text
Selection rule for goal: Select-text
If text is word, then Select-by-clicking-word
If text is arbitrary text, then Select-by-click-and-slide

GOAL: Make-a-copy
- [select: if you have little money use carbon-paper-method; if you have a xerox machine use xerox-machine-method]

Fig. 9. Choice selection policies (from [John and Kieras 96] and [Mahling and Croft 88]).

"To make a piercing cut, first drill a hole in the waste stock of the interior of the pattern. If you want to save the waste stock for later use, drill the hole near a corner in the pattern."
"Dust-mop or vacuum your parquet floor as you would carpeting. Do not scrub or wet-mop the parquet."
"To book the strip, fold the bottom third or more of the strip over the middle of the panel, pasted sides together, taking care not to crease the wallpaper sharply at the fold."

Fig. 10. Directive (or positive) advice and preventative (or negative) advice (from [Linden and De Eugenio 96]).

Conditional expressions are used in instructions to specify checks, applicability conditions, and object selection criteria. Conditional expressions use and, or, and not. An example (from [Miller 74]) is:

> "Put a card in box 3 if either the name's second letter is not 'L' or if its last letter is 'N'."

Instructions often contain information about how to generate choices and make decisions among them in different situations. They may indicate choices as well as preferences (or relative rankings) among choices. They may also indicate which of many choices is to be selected under a situation. Choice selection criteria are typically given through a set of rules, where under different conditions different options are pursued or ruled out. Figure 9 shows examples of such rules from [John and Kieras 96] and [Mahling and Croft 88].

Advice, **policies**, and **imperatives** refer to a form of information that is supposed to guide the student when confronted with a choice during procedure elaboration or execution. This information can guide the choice of objects, actions, orderings, or strategies. The distinctions are blurry, but policies refer to broad agreements within a community, imperatives refer to strong guidelines, and advice refers to any information that can be brought to bear in generating choices and in making a choice among several options.

We can distinguish between positive and negative advice. Positive advice or directives are given to point to good choices in the procedure. Negative or preventative advice points out actions and situations that would be undesirable. Figure 10 shows examples of both kinds of advice.

Advice is often given in a situated fashion. A particularly handy use of advice is to describe **exceptions** to a general procedure for particular circumstances.

2.5 Summary

The range of knowledge that can be specified about a procedure is very diverse, including the objects and principles that provide context to the procedure, the various steps that the procedure is composed of, the organization of the steps, the control structures used to coordinate among various steps. While specifying simpler procedures might involve just enumerating a few sequential steps, the description of

a procedure can becomes quite complex along all those dimensions. A system must be able to learn from all the range of knowledge specified, and to relate the different pieces of instruction into a coherent working procedure.

3. POOR TUTORIAL INSTRUCTION

In this section, we tackle the issue of how humans express in instructions the kinds of information that we mentioned in the previous section. We show that human instruction has a variety of omissions, errors, and other features that make it harder for the student to learn an appropriate procedure. Poor instruction leads to inefficient or limited learning. We discuss a variety of corpus analyses of textual rendering of tutorial instructions, and include references to the literature that point out how human learners seem to address these shortcomings. These faults occur naturally in human instruction, we need to design systems that can overcome these faults.

3.1 Omissions and Errors in Human Instruction

Human instruction has a variety of omissions that make it harder for the student to learn an appropriate procedure. [Galotti and Ganong 85] argue that human teachers follow Grice's maxims [Grice 75] of dialogue: 1) be no more or no less informative than is required; 2) be truthful; 3) be relevant; 4) make your contributions easy to understand; and 5) avoid ambiguity and obscurity. If the teacher violates these rules, the student is likely to be puzzled. They speculate that instruction such as: "1. Wet hair, 2. Apply shampoo, 3. Rinse, 4. Repeat steps 2 and 3 one time only" would be found laughable and even insulting. They report on a study where people are more likely to give more detailed instructions, in particular to include more control statements, if they write instructions for a Martian with no common sense than to another human. People assume that a human student makes inferences based on the instruction given, and they are mindful of the instructor's intentions. In their words, "it is bad form to belabor the obvious." Therefore, information is often left out intentionally either because it is less central to the instruction or because the teacher assumes that the student will infer it.

In many cases human instruction contains errors or is poorly designed. Errors are not necessarily correlated with programming expertise. For example, a study by [Brown and Gould 87] showed that 44% of the formulas created by expert spreadsheet users who had also programming background contained errors. A spreadsheet formula can be seen as a form of procedure. Another study [Kim and Gil 00] found that expert programmers and novices needed similar assistance in specifying correct procedures.

The ambiguity of natural language manifests itself in a variety of ways in tutorial instruction. [Furnas et al 87] illustrate with empirical data the tremendous variability in vocabulary when humans refer to the same object or action. Less than a dozen people out of a thousand were found to use the term that had been selected to refer to a specific computer command. Word usage for any given command was found to follow Zipf's distribution, with a few words used very frequently and the vast majority used rarely. Also, most words applied to only a few commands or objects. This means that having more words associated with objects does not imply more objects associated with a word. They propose unlimited aliasing as a possible solution. Iteratively collection of aliases from users improved interface design at little loss of precision. The convergence of the approach varies from domain to domain, showing in one of their experiments that after 100 subjects named a set of objects there was more than a ¼ chance that a new subject would propose a new term. [Bugmann et al 01] report that as they collected new examples of instructions new

words continued to appear and that the rate was not diminishing. They found that on average 42% of the instruction statements had new words. Among those, 65% had only one new word and 35% had between 2 and 6 new words. Other corpus analyses reveal the intrinsic ambiguity of natural language in expressing the same kind of information, particularly in analysis done to develop speech recognition and dialogue systems.

Despite the omissions and errors in human instruction, human learners seem to be resilient and able to learn from it. When information is missing in the instruction, the student has to make guesses regarding the missing information. [Lee and Dry 06] found that people are more confident in their guesses when they have to make less of them. Also their confidence is influenced by the accuracy of the advice as well as the frequency of the advice.

All these faults present challenges for the development of systems that learn from human instruction. A variety of studies have shown that these faults occur in information about each of the four major categories of information in instruction described in the previous section. In the rest of this section, we address each of the four categories in turn.

3.2 Missing and Erroneous Background Information

The assumptions that the teacher makes on the background information that the student has are key in enabling the interpretation of a given set of instructions. There are many studies that show this for a variety of kinds of background knowledge such as domain-specific knowledge, general principles, learning strategies, experiential knowledge, related knowledge to support transfer learning, and other skills. [VanLehn et al 07] found that when tutorial materials are prepared to the level of preparation that the student has, written instructions are just as effective as dialoguing with a tutor. [Kirschner et al 06] discusses how tutor guidance is less important when the learners have sufficiently high prior knowledge to provide internal guidance, citing numerous studies that support this.

3.3 Missing and Erroneous Information about Steps

Important information about steps is often also missing in human instruction. Necessary conditions and steps may be left out of the instruction. Contrast the instruction (from [Webber et al 95]):

> "Depress door release button to open door and expose paper bag."

with:

> "Holding canister horizontally, depress door release button to open door and expose paper bag."

where gravity is taken into account by placing the canister horizontally. Also contrast with this instruction where there is an explicit action to open the canister:

> "Depress door release button, then grasp the door and pull it open to expose paper bag."

where there is an explicit user action included to open the canister door.

[Lau et al 09] report that many instructions have missing steps as well as errors in the steps that were present. Some steps were indirectly specified in commentary, for example "if you click on the top button you will see the next page" was stated to mean that the button should be clicked next. Human learners assume that steps that are important to the procedure will be explicitly stated in the instruction, though

1. Pull out sharply in order to remove the phone.
2. To remove the phone, pull out sharply.
3. Pull out sharply for phone removal.
4. Pull out sharply for removing the phone.
5. For the phone, pull out sharply.
6. Remove phone by pulling out sharply.
7. Remove the phone. Pull out sharply.
8. The purpose of pulling out sharply is to remove the phone.
9. Pulling out sharply achieves the purpose of removing the phone.
10. Removing the phone involves pulling out sharply.
11. The method for removing the phone is to pull out sharply.

Fig. 11. Alternative forms of expressing a purpose relation between two clauses in a sentence [Linden and Martin 95].

they manage to learn when steps are implicit in the instruction [Dixon et al 88]. Conversely, when steps are relatively unimportant but stated explicitly, the initial inferences made about them tend to be erroneous and must be later corrected. This does not happen when learners have expertise in the subject matter, and use their own judgment to decide on the importance of the steps.

[Mahling and Croft 88] found that most people are very good at expressing task decomposition, sequencing, and preconditions, but are not very good at recalling effects of procedures and actions. [Wright and Hull 90] report that 50% of instructions had omissions of locations where the procedure was to take place. [Lauria et al 02] found that instructions rarely specified starting or final state, but rather focused on the action to be performed.

Ambiguity also affects the purpose of steps in the instruction. [Linden and Martin 95] analyzed a corpus of instructions from 17 diverse sources containing 6,000 words in 1,000 clauses. Figure 11 shows examples of alternative expressions of the same information: what to do to remove a phone. Table 4 identifies the major linguistic forms used and shows the number and percentage of occurrences in the corpus. Other studies show that there are many alternative language constructs to express the same kind of instruction information. [Kosseim and Lapalme 00] analyzed a large corpus of instructions from 15 different sources containing 13,000 words to convey 79 procedures of different domains and target readers. They found that the sentences could be classified as conveying nine types of information realized in seven categories of rhetorical relations. Table 4 shows the frequency of each type of information (sense) as well as mappings to rhetorical relations in Rhetorical Structure Theory (RST) [Mann and Thompson 88]. The type of information is shown for the entire corpus, for the subset of the corpus designed to teach how to execute procedures, for the subset of the corpus designed to explain procedures, and for the subset of the corpus designed for combining execution and comprehension. The analysis showed that the same type of information may be rendered using different rhetorical relations, as shown in Table 5. For example, step information ("required operation" category) is mostly presented as sequences, while effects ("outcome" category) are presented either as purpose, result, or means.

Table 4. Alternative expressions of a purpose relation (from [Linden & Martin 95]).

a. **Infinitive form with "to":** *To end a previous call*, hold down FLASH [6] for about two seconds, then release it. (Code-a-phone, 1989)
b. **Prepositional phrase with "for" (nominalization):** Follow the steps in the illustration below, *for desk installation*. (Code-a-phone, 1989)
c. **Gerund with a "for" preposition:** The OFF position is primarily used *for charging the batteries*. (Code-a-phone, 1989)
d. **Noun phrase with a "for" preposition (goal metonymy):** *For frequently busy numbers*, you'll want to use REDIAL [7], and the pause will have to be in Redial memory. (Code-a-phone, 1989)
e. **Simple imperative joined with "by" preposition:** When instructed (approx. 10 sec.) *remove phone by* firmly grasping top of handset and pulling out. (Airfone, 1991)
f. **Simple imperative in adjoining sentence:** *Return handset to wall unit from which it was taken*. Insert heel first as shown, then push top in firmly. (Airfone, 1991)
g. **Simple imperative joined with "so that":** Tilt pan down slightly at the rear *so that the fluid drains out.* (Reader's Digest, 1981)

	Initial	Final	Total (count)	Total (percentage)
(a) To-Infinitive	38	33	71	59.6%
(b) For-Nominalization	2	7	9	7.5%
(c) For-Gerund	0	3	3	2.5%
(d) For-Goal-Metonymy	1	5	6	5.0%
(e) By-Purpose	11	1	12	10.0%
(f) Adjoined-Purpose	4	0	4	3.3%
(g) So-That-Purpose	0	10	10	8.4%
Other	4		4	3.3%

3.4 Problems with the Organization of Instruction

The organization of instruction is another area where omissions and errors occur. [Wright and Hull 90] report that only 30% of instructions in their study contained overviews. [Eylon and Reif 84] found that students with less preparation were less able to assimilate hierarchical organization information. [Hoc 89] shows that the kinds of adequate abstractions needed to develop working instructions are in fact hard for people to design. To remedy this, [Van Merriënboer et al 03] describes how to sequence lessons with simple-to-complex strategies to teach complex tasks. One is a part-task approach where the learner starts with simpler tasks and builds up skills to the more complex tasks. Each instructional objective covers one of the sub-tasks. It is not until the end of the curriculum that the learner can practice the whole task. While this is not a practical approach for complex tasks that require a high level integration of constituent skills, part-task practice is useful for drilling on problems on recurrent aspects of an overall complex task. A second alternative is a whole-task approach where a simplified but real version of the entire procedure is presented at the beginning, designing subsequent lessons to cover other conditions for the task that uncover further complexity. This approach breaks down the complex tasks by identifying equivalence classes of problems, where the simplified version might correspond to a class of simpler task problems.

Table 5. Frequencies of types of information in instruction (senses) for a corpus and their mappings to rhetorical relations, from [Kosseim and Lapalme 00].

	Entire corpus		Execution Texts	Hybrid Texts	Comprehension Texts
Sense	Number of occurrences	%	%	%	%
ATTRIBUTE	158	11	3	17	95
REQUIRED OPERATION	762	52	65	40	29
CONDITION	164	11	11	12	9
OUTCOME	136	9	7	13	9
GUIDANCE	124	8	9	8	8
CO-TEMPORAL OPERATION	45	3	1	4	7
OPTION	34	2	2	3	3
PREVENTION	21	1	1	2	2
POSSIBLE OPERATION	15	1	1	1	2
OTHER	12	1	0	0	5
Total	1471	≈100	100	100	100

Sense	Rhetorical Relation						
	sequence	c-condition	elaboration	purpose	result	means	concurrency
ATTRIBUTE			100%				
REQUIRED OPERATION	98%	1%		1%			
CONDITION	2%	90%		4%	4%		
OUTCOME				28%	68%	4%	
GUIDANCE				31%		69%	
CO-TEMPORAL OPERATION							100%
OPTION		21%		79%			
PREVENTION	86%					14%	
POSSIBLE OPERATION		73%				27%	

3.5 Missing and Erroneous Information about Control Constructs

Control constructs are truly prone to errors, as humans appear to find many traditional programming constructs to be unnatural and hard to grasp. [Miller 81]

did a corpus analysis over instructions provided by different subjects, and categorized the types of information used. Figure 12 shows the six major categories and twenty-five subcategories used, as well as their frequency in two different instruction corpora. The frequency is also shown for those same categories as they appear in a set of programs written by students. There are major differences in terms of the amount of transfer of control statements, and the difference would likely be larger with a corpus of programs that had been fully complete and made robust to errors. Table 6 summarizes the major differences found between natural language expressions of procedures and typical characteristics and constructs in programming languages. Figure 13 gives an example that contrasts pseudo-code for a program with the natural language instructions for a task. The program follows a conditioned action style, in contrast the corresponding natural language instruction follows action qualification style (the arrow indicates the primary action). Note that control statements are a notorious differentiator.

Control constructs involving conditional expressions are notoriously hard for humans to express correctly. When human instructors express conditional expressions within a procedure they do not intend them to be interpreted by the rules of Boolean logic [Pane and Myers 00a]. The same was observed in studies of database query formulation [Androutsopoulos et al 95; Greene et al 90]. For example, in "Find the customers that are located in California and Nevada" the word 'and' is meant to be interpreted as a Boolean or. The word 'or' often means exclusive or. The difficulty in comprehending and stating Boolean expressions has been found to be correlated with their complexity [Feldman 00; Miller 74]. Even subjects with significant experience in formal logic have been found to make the same kinds of mistakes in complex queries than other subjects [Weiland and Shneiderman 92]. Using parentheses for grouping subexpressions was not found to help users [Greene et al 90]. Alternative mechanisms to natural language entry have been proposed that effectively reduce the error rates in constructing and interpreting Boolean expressions, including visual languages such as flow-based selection [Young and Shneiderman 93], logic gates [Green and Petre 96], and Karnaugh maps [Huo and Cowan 08] as well as textual alternatives such as query by example and tabular query forms [Pane and Myers 00b].

Conditionals and iterations are often incompletely specified in human instructions. An example of an incomplete partial conditional (from [Miller 81]):

> "(1) See if the age of the person is greater than 50;
> (2) Write his name down on a list."

and a loop with no explicit termination condition:

> "Wet hair, apply shampoo, rinse, and repeat."

It is worth mentioning here that although rules are a natural way to convey control knowledge, teaching procedures using this format has been shown to lead to inaccurate lessons. [Clark et al 04] points to several studies that provide evidence for this and argues that teaching complex tasks appropriately requires following a methodology that recognizes the role of rules and avoids well-known pitfalls. To solve a task, important cues from the environment must be recognized and associated with steps, which may be covert (cognitive) or overt (action) steps. Through practice, conditional cues and steps are mapped to rules that require much less cognitive effort and lead to better speed and performance. In the end, if-then rules are strung

---

1. Actions involving existing data structures
   (a) Files – go, get, use, look, open, select

(b) Records – same
(c) Records – movement to a different location
(d) Item – go, get, use, look, select
(e) Problem – problem statement, other given information

2. Actions involving new data structures
   (a) Creating new item/record (single)
   (b) Creating new item/record (multiple)
   (c) Manipulation – go, get, use, move, label

3. Attribute testing
   (a) Check on single record, attribute, and value mentioned explicitly
   (b) Check on multiple records, attribute, and value mentioned explicitly
   (c) Check on single record, only value explicit
   (d) Check on multiple records, only value explicit
   (e) Check on something other than attribute or value
   (f) Check on single record, attribute/value mentioned only implicitly or in incomplete linguistic structure
   (g) Check on multiple records, attribute/value mentioned only implicitly or in incomplete linguistic structure

4. Transfer of control
   (a) Iteration, repetition
   (b) Full conditional (with provision for both outcomes)
   (c) Partial conditional (provision only for successful test outcome; no provision for "ELSE" or no outcome)
   (d) Unconditional transfer or "GOTO"
   (e) Sequencing reference – "after, when, until"
   (f) Reference to terminating procedure ("stop")

5. Transformations
   (a) Explicit ordering of new data ("alphabetize")
   (b) Computations involving item information
   (c) Invoke some other general procedure

6. Miscellaneous
   (a) Nonprocedural comments

Content class	*Attrib. (%)*	*Nocont. (%)*	*Knuth (%)*
1. Existing data	40	21	16
2. New Data	16	45	18
3. Attribute test	27	4	7
4. Control	9	3	22
5. Procedures	7	26	27
6. Comments	1	1	10

Fig. 12. Types of information appearing in instructions by several subjects (top), and frequency of each in two different corpus (an attribute testing task and a noncontingent task) compared to a corpus of programs (supplied by Knuth), from [Miller 81].

Table 6. Comparison of common features of programming languages and their use in natural language specifications (from [Miller 81]).

Features	Programming languages	Natural language specifications
Data		
Declarations, etc.	Always explicit	Never occurred
References	Explicit, well-defined	Implicit, contextual
Examination/creation	Usually iterative, element by element	On aggregate basis
Indexing	By numerical/variable value, major aspect	Seldom occurred, then contextually defined (e.g. ”next,” “previous”)
Data types	Many, defined	No distinction
Format specs.	Many, explicit	Infrequent, contextual
Transfer of control		
Extent	Major aspect of programs and style	Seldom specified
IF-THEN-ELSE	Most used at present	When occurred, only partial – IF-THEN (no else)
IF (cond.) GOTO	Major feature	Never occurred
Uncond. GOTO	Was major, still common	Never occurred
Exception detec.	Important feature	Never occurred
Structure	Many types: recursion, co-routines, nonlinear	Basically linear block structures
Procedure calls	Frequent, specified completely	Mostly control mechanism, but context specified
Argument passing	Always explicit	Mostly implicit
General language		
Lexicon	Very limited, except for variable names	Can be rich and large, with many synonyms, may be restricted
Sentence type	Active imperative and conditional	Mainly active imperative, but can be declarative/conditional
Sentence syntax	Quite rigid	Extremely variable, may be very complex

together to generate behavior, and therefore if-then rules become a natural way to convey knowledge. For example, studies have shown that up to one third of the relevant cues are not included in instruction, and that the knowledge conveyed is often not sufficient to solve the task [Clark et al 04]. Cognitive task analysis offers an effective methodology for formulating lessons and exposing covert knowledge that is crucial to teach complex tasks [Clark et al 04].

```
DO END UNTIL TIME = 5:00PM
    I = 0
```

```
DO END OUT WHILE 1 < 200
     I = I + I
     OPEN BOX (I)
     J = 0
     DO  END.IN WHILE J < 12
          GET NEXT BALL
          IF RED THEN
               IF LARGE THEN
                    IF UNBROKEN THEN
                         J = J + 1
  ------------------→ PACK BALL IN BOX(I) CELL(J)
                         RETURN (END.IN)
                    ELSE RETURN (END.IN)
               ELSE RETURN (END.IN)
          ELSE RETURN (END.IN)
     END.IN
     CLOSE BOX (I)
END.OUT
END
```

(A) Program Normal Form

```
---------→ PACK LARGE RED DECORATIONS TWELVE TO A BOX.
            MAKE UP A TOTAL OF 200 BOEXES.
            STOP AT 5:00 PM IF NOT FINISHED.
            BE SURE TO PACK ONLY THE UNBROKEN ONES.
```

(B) Natural Normal Form

Fig. 13. A comparison of a program pseudo-code (A) versus natural language instruction for the same task (B) (from [Miller 81]).

3.6 Summary

In its natural form, human instruction is plagued with errors and omissions. This is a natural way for people to describe procedures, as too much detail is often considered too verbose and unnatural. This makes human instruction be far from the kind of complete and correct logical instruction set that a computer could interpret and execute. In order to learn from human instruction, a system must be able to cope with all these faults.

4. RESEARCH CHALLENGES IN LEARNING FROM TUTORIAL INSTRUCTION

We have described so far a range of difficulties in learning from tutorial instruction. We presented the different kinds of information that can be specified about procedures. We also discussed various kinds of faults that appear in human instructions, including errors and omissions as well as mis-organization in the presentation of information. We now reflect on how these characteristics affect the difficulty in learning from instruction, and discuss research challenges in several aspects of this task.

4.1 Learning Difficulty in Tutorial Lessons

The omissions and errors that human teachers commonly commit in tutorial lessons must be corrected in order to learn a proper procedure. To learn from a given

lesson, the system must consider how to fill possible omissions and how to fix the errors. This leads to many combinations of hypotheses that result in alternative possible models of the procedure to be learned. The student must then reason about the plausibility and likelihood of alternative models, perhaps based on background knowledge or what it has already learned. Concisely put, the difficulty of the learning task increases when the lessons contain more information because the procedures are complex, and when larger numbers of omissions and errors are present. We discuss the challenges of the learning task as we revisit the four broad categories of information that appear in procedural knowledge: background information, procedure steps, organization of the instruction, and control structures.

The first category is the specification of background information through the introduction of objects. Procedures are applied in a rich context that involves objects with specific properties and relationships. That is, instruction of procedures is often situated in that it is framed in the context of a general situation, specified by introducing a set of generic objects. For example, a lesson may start off by saying "suppose you have a route to follow, and a vehicle that cannot negotiate slopes of more than 30 degrees", which introduces two objects and their types plus a property constraint. If the instruction introduces all the objects and constraints that will be used in the lesson, it is easier for the student to understand how the objects are used throughout the procedure. If the instruction does not introduce some of the objects, then the student has to hypothesize what the objects are and what their types and constraints might be in the situations where the procedure must be applied. The more objects that are not properly introduced by the teacher, the harder the learning task because there are more hypotheses to explore. A different dimension that makes learning harder is the resolution of references when objects are similar. If there are several objects of similar types that are not specifically introduced by the teacher, it will be harder for the student to figure out which object references might be the same and which might refer to a distinct object and therefore figure out how many distinct objects need to be present. In those cases, the student has to hypothesize and explore more alternative assignments or permutations of the data objects.

In introducing new objects, another source of difficulty for learning is mixing the introduction of the objects within other expressions. A case of this is the need to introduce existential or universal quantification. In general, there can be multiple different interpretations of the same expression that lead to different uses of quantifiers, depending on the nature of the introduced objects and how they are related. When such quantification is not explicitly stated, the student needs to form multiple different hypotheses. The number of hypotheses will grow when the student has to consider alternative interpretations of multiple objects.

The second category is the description of procedure steps. Here, what makes learning procedures more difficult is the lack of necessary details in describing a procedure's steps. For example, the instruction may call a subprocedure as one of the steps but omit arguments that are required by the definition of the subprocedure. In that case, the student must hypothesize what objects may be used to invoked the subprocedure, perhaps also hypothesizing objects that were not introduced in the instruction. The more arguments missing and the more candidate objects, the larger the hypothesis space to explore. Steps can also be specified indirectly by mentioning the effect of an action but not the action itself. For example, the instruction may say "Before starting the engine make sure there is gas in the tank." In those cases, the student must rely on prior knowledge about actions that may have the effect mentioned, then insert the action where it may be appropriate in the procedure. The more candidate actions and insertion locations, the harder learning is. Instruction

may also lack information about what to do when exceptions arise. In these cases, the student is unlikely to infer what the missing information is and would have to wait for other opportunities to learn that, for example with follow up instruction, practicing on their own, or observing the teacher.

The third category is the specification of relations among steps, including ordering relations, causal relations, dataflow relations, resource relations, and temporal relations. A major source of difficulty is that instruction is always provided in a sequential order but a procedure's step structure may be quite complex and non-linear. Of immediate concern are step ordering relations. The instruction may specify steps in a convenient but not necessarily correct order. In addition, steps are given in a sequence when the actual dependencies among steps are better represented as a partial order. If the step ordering is not fully specified or incorrect, the student needs to reason about possible steps and step orderings, as well as other relationships among them such as causal or dataflow relations. The space to explore can grow quickly as the number of hypotheses or the degree of ambiguity is higher.

An important relationship among steps is based on how the results and effects of each step are used by others. Some actions have side effects and do not return any specific result, while other actions return an object that can be used as an argument of subsequent steps. The former typically correspond to representations of physical procedures while the latter are functions meant to be evaluated to find a value. If the student knows that a function is to be learned, then the student can reason about what each step is returning and the compatibility between the objects produced and returned by each substep as well as the entire function. If the student does not know whether a function or a procedure is to be learned, then the student must reason about the effects of each action (including side-effects and conditional effects) and the prerequisites of other actions in order to figure out how the substeps are related.

The fourth category is control constructs to organize steps. Control constructs such as conditionals and iterations are notoriously challenging for human teachers to specify in all the detail necessary in order for a procedure to be executable. That is, natural human instruction typically leaves out important things such as initialization and termination conditions for iterations, clauses in conditions, else statements, and the need to create temporary variables (e.g., counters) for iterations. When conditions and iterations are present, the student's analysis of the procedure becomes more complex. For conditional branches, the student needs to consider the alternative data flows and control flows possible. For iterations, the student may need to analyze several folds in order to understand how a loop needs to work. Iterations that are easier to learn involve processing sets of objects one at a time. Harder iterations to learn are those that require the student to set up loops with new variables and infer exceptional initial and termination conditions. Conditionals that are easier to learn involve checking the state for a new object or property value to manifest. More complex conditionals can mention disjuncts and negations whose scope may be hard to determine from the instruction.

Combinations and nestings of conditionals and iterations within a procedure make learning harder. When conditional statements are nested, instructions typically do not specify the scoping of each statement. When scoping is ambiguous, the student must create and explore alternative hypotheses. The more combinations, the larger the search space that the student must explore. Iterations can also be nested, raising the complexity of learning because of potential interactions between the objects in the inner and outer loops.

To sum up, the nature and amount of imperfections in the instruction affects the complexity of the learning task. At the hardest end of the scale is learning more complex procedures with more faulty instruction. At the more tractable end of the

scale is where research to date has focused, where users are either constrained to express simpler procedures or constrained to produce more precise and error-free instruction. Learning systems must be able to fill in gaps through any knowledge they already have, or perhaps have the capability of asking questions that expose where the instruction could be completed by the human teacher.

4.2 Natural User Interfaces for Tutorial Instruction

In order for non-programmers to provide tutorial instruction, they need interfaces that they find natural and allow them to specify instruction in a manner similar to how they would interact with a person. Using natural language is a good approach, but the difficulties to interpreting natural language are many. Several directions are possible, either by using graphical languages or constraining natural language.

Visual programming and other natural interface designs have been developed that are effective means to convey instruction. [Kelleher and Pausch 05; Ko et al 11] provide a thorough overview of such systems, which they call empowering systems since they aim to help non-programmers to specify behaviors for a computer system. Some of the best known systems are AgentSheets, Stagecast, Logo, Alice, Forms/3, and Hypercard. Successful techniques include programming by rehearsal by personifying components, the use of domino icons or comic strips to show before and after states for actions, 2-D grids and patterns for specifying conditions for behavior rules, physical metaphors to represent objects and behaviors, associating behaviors to interface components, object-centric commands that can be interpreted, commands with graded complexity in parameters, aggregate operations over objects, making specifications alive so they can always be tested even if partially specified, extending spreadsheets to create new data types and their associated behaviors, event-triggered behaviors, and visual dataflow languages. Most of these tools are designed to target specific tasks, objects, or behaviors. Yet they have been shown effective in experiments with non-programmers, and some are commercialized and have been used by thousands of users.

[Nardi 95] provides good arguments in favor of structured languages that are neither visual nor natural language text, and shows that they can be at least as natural and as effective as visual languages and interfaces, particularly for complex tasks. Nardi points to studies that show that visual languages are not more effective than textual languages, and that when they are the improvements are not huge. Spreadsheets and CAD systems are two perfect examples of languages that are not text and yet effectively used by end users to program their applications [Nardi 95; Kay 84]. The key is to offer primitives and operations that are appropriate for the type of task targeted by the system, to localize the complexity of the language so it is accessible, and to make it clear to the user what the side effects of any change are. Spreadsheets, for example, offer a textual language that contains a library of operations, and a clear model of how changes in a given cell affect other cells. Conditional constructs can be complex and nested but have clear local effects in that they only affect the cell where they are defined. Iterations are easily defined by aggregating groups of cells. Spreadsheets also expose clearly the propagation mechanisms of cell values, so users can anticipate the effects of their operations. Another useful feature is that they provide a sophisticated interactive browser. Nardi quotes [Brooks 87] to convey these points:

> "Software is very difficult to visualize. Whether one diagrams control flow, variable-scope nesting, variable cross-references, dataflow, hierarchical data structures, or whatever, one feels only one dimension of the intricately

> interlocked software elephant. If one superimposes all diagrams generated by the many relevant views, it is difficult to extract any global overview."

[Nardi 95] also points out that many formal languages are used by non-programmers for a variety of tasks. Unlike programming languages, these formal languages are used daily by large amounts of people. Examples include musical notation, algebra, baseball scoresheets, and knitting patterns. These languages prove that people are willing to invest the time to learn a new complex language as long as it is accessible and it is natural for the task they need to perform. Natural language is clearly not the most popular or convenient means to express instructions for those tasks. Nardi argues that the key to developing a successful language and interface for end users is to study the task that they are targeted for, and design the language and the interface to serve no less and no more than that task. Nardi quotes [Winograd and Flores 86]:

> "[Driving a car] is not achieved by having a car communicate like a person [i.e., through conversation], but by providing the right coupling between the driver and action in the relevant domain (motion down the road)."

Controlled natural language is a compromise between a formal language and natural language [Nyberg and Mitamura 96; Fuchs et al 96; Fuchs et al 06]. Controlled languages have been used effectively in a variety of contexts, notably for technical documentation manuals in order to facilitate machine translation and readability. [Blythe and Gil 04] used a controlled English interface generated from ontological models to interactively reformulate user instruction to be understandable by the system and to eliminate ambiguity in the user utterances. A core process model can be the basis for a controlled English interface [Fuchs et al 06; Clark et al 05].

4.3 Helping Humans Improve How They Teach

Another challenging research area is to guide humans to provide more complete and correct instruction that facilitates the system's learning task.

Some approaches have been proposed for guiding a human teacher to generate more complete instruction. This is known as guided instruction. [Mahling and Croft 88] show that forms are a very successful means of eliciting information about effects, which they found are often missing from instructions. [Van Merrienboer 97] proposes the use of process worksheets to guide students through complex tasks. Worked examples and process worksheets are also effective techniques for guided instruction. [Young 99] studied the production of automatic instructions from formal representations of procedures. They found that including too much detail leads to lack of flexibility in accomplishing subtasks, while providing insufficient detail results on omitting preference information to discern across choices for underspecified steps. The challenge is in finding the right balance between making the instruction more complete while leaving enough flexibility to the human teacher.

Other approaches are based on structuring the interactions with the teacher based on a shared declarative model of process properties. A shared process model is essentially a general ontology of process properties (input requirements, conditions, subtasks, etc), often known as an upper model or upper ontology. This shared model cannot be based on the requirements of the system only, but rather needs to take into account the cognitive aspects of human instructional practices. For example, [Mahling and Croft 88] propose a framework that accounts for their human task recall studies that could be used as such an ontology. [Mahling and Croft 88] combined their framework with the use of pre-populated forms, where the forms elicit effects from users that would otherwise naturally provide poor instruction

about effects of actions. Tutorial instruction of procedures, like other forms of scientific and technical expositions, exhibits goal-oriented hierarchical structure [Britt and Larson 03]. Such goal decomposition hierarchies have been found to be useful for guiding users to specify procedures [Gil and Melz 96] as well as for providing explanations [Swartout et al 91].

4.4 Adjusting People's Attitude towards Teaching Computers

The interaction that human teachers have when instructing computers versus instructing other humans turns out to be different. In general, humans interact with computers differently than with other humans, and this applies to instruction as well [Herberg et al 08]. Other studies report that existing learning algorithms do not conform to the kinds of instruction that humans provide [Thomaz and Breazeal 08a; Thomaz and Breazeal 08b; Thomaz and Cakmak 09], making it hard to adapt them for learning from tutorial instruction. User studies where humans teach procedures to robots have found that speech prosody (tone of voice and rhythm) and affect (emotions and attitudes) are used to convey some forms of instruction.

A great challenge is that people do not have a good model of the abilities of computer systems. This is important because humans customize their instruction in response to the learner's competence level. [Kim et al 09] found that a student that had made mistakes before received more detailed instruction than one that had not made prior mistakes. Human teachers also expect learners to become more competent over time [Butko and Movellan 07]. The challenge for the system is to convey its abilities in terms of what kinds of instruction it is able to learn from and what it understands from the human teacher.

4.5 Learning from Instruction in Combination with Other Teaching Modalities

Instruction can be of many kinds [Webber et al 95], including general policies regarding acceptable behaviors, advice on how to proceed, suggestions for preferences, analogies, requests, and tutorial descriptions. Instruction can be given before, during, or after a procedure takes place. During the execution of a procedure, the instruction may be given to overcome a specific failure.

Instruction can have didactic or dialogue style [Chi et al 01]. In a didactic style, the teacher provides an expository presentation of the entire instruction set. In a dialogue style, the teacher may respond to prompts from the student, provide explanations for specific aspects of a problem the student is solving, or provide feedback based on the student's actions.

Tutorial instruction can be combined with other forms of instruction, such as using examples or demonstrations. They can also be presented in the context of general situations or scenarios.

Instruction may be accompanied of diagrams or pictures, or consist solely of text [Larkin and Simon 87; Koedinger and Anderson 92; Chandrasekaran et al 95].

The particular focus of this article is tutorial instruction of procedures that are recurrent and routine, given before execution with a didactic-style, and provided in natural language. A major research challenge is to combine this modality of teaching with all these other modalities, such as learning from tutorial instruction that integrates text and diagrams, learning from tutorial instruction combined with examples or demonstrations, or providing tutorial instruction when the student makes mistakes while practicing a procedure that the student has learned.

5. CONCLUSIONS

Developing systems that can learn procedures from tutorial instruction provided by end users has been a long-standing research goal. This would enable large numbers of people to develop complex applications without having to learn to program. In this paper, we surveyed work from the literature that highlights the challenges in learning from tutorial instruction, illustrating them with examples in many domains. We discussed two major characteristics of tutorial lessons affect the difficulty of learning procedures from human teachers. First, procedures can be very complex and involve many different types of interrelated information: 1) situating the instruction in the context of relevant objects and their properties, 2) describing the steps involved, 3) specifying the organization of the procedure in terms of relationships among steps and substeps, and 4) conveying control structures. Second, human tutorial instruction is naturally plagued with omissions, oversights, unintentional inconsistencies, errors, and simply poor design. We also discussed how the complexity of the learning task increases with the complexity of the procedure in terms of the information to be conveyed and in terms of the amount of omissions and errors that occur in each of these dimensions.

This article provides a framework to situate the research to date on addressing these challenges. To understand the state of the art, a survey would need to cover research in very different areas spanning several decades, including end user programming (e.g., [Lieberman et al 05; Kelleher and Pausch 05; Myers et al 04; Ko et al 06]), intelligent user interfaces (e.g., [Allen et al 07; Gil et al 12]), knowledge capture (e.g., [Clark et al 05]), machine learning (e.g., [Walker et al 11; Gil et al 11; Huffman and Laird 95]), natural language (e.g., [Fuchs et al 06; Webber et al 95]), and robotics (e.g., [Gold and Scassellati 07; Gold et al 07; Kim and Scassellati 07]).

Despite the challenges highlighted in this article, humans can learn from such imperfect instruction because they have strategies to work around those imperfections. Clearly, the student's task is easier or harder depending on the degree and nature of the imperfections in the instruction. We should design systems that exhibit the same kind of resilience. They will be more accessible to human teachers if they are equipped to learn from the kinds of instruction that human teachers typically provide.

REFERENCES

ADAMS, S. "The Future of End User Programming?" *International Conference on Software Engineering*, 2008.

ALLEN, J. F., CHAMBERS, N., FERGUSON, G., GALESCU, L., JUNG, H., SWIFT, M., AND TAYSOM, W. "PLOW: A collaborative task learning agent." Named Best Paper, *National Conference on Artificial Intelligence (AAAI)*, Vancouver, BC, 2007.

ALTERMAN, R., ZITO-WOLF, R., AND T. CARPENTER. "Interaction, Comprehension, and Instruction Usage." *Journal of the Learning Sciences*, 1(3-4), 1991.

ANDERSON, C. "The Long Tail: Why the Future of Business is Selling Less of More." *Hyperion*, 2008.

ANDROUTSOPOULOS, I., RITCHIE, G. D., AND P. THANISCH. "Natural Language Interfaces to Databases – An introduction." *Natural Language Engineering*, 1(1), 1995.

BAKER, COLIN L., FILLMORE, CHARLES J., AND J. B. LOWE. "The Berkeley FrameNet Project." In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics* - Volume 1 (ACL), Vol. 1. Association for Computational Linguistics, pp 86-90. Stroudsburg, PA, USA, 1998.

BLYTHE, J. AND GIL, Y. "Incremental Formalization of Document Annotations through Ontology-Based Paraphrasing." *Proceedings of the Thirteenth International World Wide Web Conference*, (WWW), New York, NY, May 17-22, 2004.

BOEMH B. W. ABTS, C., BROWN, W., CHULANI, S., CLARK, B. K., HOROWITZ, E., MADACHY, R., REIFER, D. J., AND B. STEECE. "Software Cost Estimation with COCOMO II". Prentice Hall, 2000.

BRITT, M. A., AND A. A. LARSON. "Constructing Representations of Arguments", *Journal of Memory and Language*, 48, 2003.

BROOKS, F. "No Silver Bullet: Essence and Accidents of Software Engineering." *IEEE Computer*, Vol 20, 1987.

BROWN, P. AND J. GOULD. "How People Create Spreadsheets." *ACM transactions on office information systems*, 5(3), 1987.

BUGMANN, G., LAURIA, S., KYRIACOU, T., KLEIN, E., BOS, J., AND K. COVENTRY. "Using Verbal Instructions for Route Learning: Instruction Analysis." *Proceedings of the Conference Towards Autonomous Robots* (TIMR), Manchester, UK, 2001.

BUTKO N. J. AND J. MOVELLAN, 2007. "Learning to Learn," *Proceedings of the Sixth International Conference on Development and Learning* (ICDL), 2007.

CASTELLI, V., BERGMAN, L.D., LAU, T., AND OBLINGER, D. "Sheepdog, parallel collaborative programming-by-demonstration." *Knowledge-Based Systems*, 23(2): 94-109, 2010.

CHEN, J. AND WELD, D.S. "Recovering from errors during programming by demonstration". *Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI)*, Canary Islands, Spain, January 2008.

CHI, M. T. H., SILER, S. A., JEONG, H., YAMAUCHI, T., AND R. G. HAUSMANN. "Learning from human tutoring," *Cognitive Science*, 25(4), 2001.

CHI, M. AND K. VANLEHN. "Eliminating the Gap between the High and Low Students through Meta-Cognitive Strategy Instruction." *Proceedings of the 9th International Conference on Intelligent Tutoring Systems,* 2008.

CHANDRASEKARAN, B., GLASGOW, J., AND N. H. NARAYANAN (EDS). "Diagramatic Reasoning". *AAAI Press*, 1995.

CLARK, P., HARRISON, P., JENKINS, T., THOMPSON, J. AND R. WOJCIK. "Acquiring and Using World Knowledge using a Restricted Subset of English." *Proceedings of the 18th International Conference of the Florida Artificial Intelligence Research Society (FLAIRS)*, 2005.

CLARK, P., THOMPSON, J., BARKER, K., PORTER, B., CHAUD-HRI, V., RODRIGUEZ, A., THOMERE, J., MISHRA, S., GIL, Y., HAYES, P., REICHHERZER, T. "Knowledge Entry as the Graphical Assembly of Components," *Proceedings of the International Conference on Knowledge Capture (K-CAP)*, 2001.

CLARK, R. E., FELDON. D. F., VAN MERRIENBOER, J. J. G., YATES, K., AND EARLY, S. "Cognitive Task Analysis", *In Handbook of Research on Educational Communication and Technology*. Lawrence Erlbaum, 2004.

DI EUGENIO, B. "An Action Representation Formalism to Interpret Natural Language Instructions." *Computational Intelligence*, Vol. 14(1), 1998.

DI EUGENIO, B. AND B. L. WEBBER. "Pragmatic Overloading in Natural Language Instructions." *International Journal of Expert Systems*, Vol. 9(1), 1996.

DIXON, P. "Plans and directions for complex tasks." *Journal of Verbal Learning and Verbal Behavior*, 21, 1982.

DIXON, P. "The processing of organizational and component step information in written directions." *Journal of Memory and Language*, 26, 24-35, 1987.

DIXON, P., FARIES, J., AND GABRYS, G. "The role of explicit action statements in understanding and using written directions." *Journal of Memory and Language*, 27, 1988.

DONIN J., BRACEWELL, R.J., FREDERICKSEN, C., AND DILLINGER, M. "Students' Strategies for Writing Instructions." *Written Communication*, 9(2), 1992.

EYLON, B., AND F. REIF, "Effects of Knowledge Organization on Task Performance." *Cognition and Instruction*, 1(1), 1984.

FELDMAN, J. "Minimization of Boolean Complexity in Human Concept Learning." *Nature,* Vol 407, 2000.

FILLMORE, C. J. "The case for case". *In Universals in Linguistic Theory*, Emmon Bach and Robert T. Harms (Eds), Holt, Rinehart and Winston, Inc., 1968.

FRITZ, C. AND GIL, Y. "A Formal Framework for Combining Natural Instruction and Demonstration for End-User Programming." *Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI)*, Palo Alto, CA, February 2011.

FUCHS, N. E. AND R. SCHWITTER. "Attempto Controlled English (ACE)." *Proceedings of the First International Workshop on Controlled Language Applications* (CLAW), 1996.

FUCHS, N. E., KALJURAND, K. AND G. SCHNEIDER. 2006. "Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces." *Proceedings of the International Conference of the Florida Artificial Intelligence Research Society* (FLAIRS), 2006.

FURNAS, G. W., LANDAUER, T. K., GOMEZ, L. M., AND S. T. DUMAIS. "The Vocabulary Problem in Human-System Communication." *Communications of the ACM*, 30(11), 1987.

GALOTTI, K. M AND W. F. GANONG. "What Non-Programmers Know about Programming: Natural Language Procedure Specification." *International Journal of Man-Machine Studies*, Vol 22, 1985.

GIL, Y. AND MELZ, E. "Explicit Representations of Problem-Solving Strategies to Support Knowledge Acquisition." *Proceedings of the Thirteen National Conference on Artificial Intelligence* (AAAI), 1996.

GIL, Y., RATNAKAR, V., AND FRITZ, C. "TellMe: Learning Procedures from Tutorial Instruction." Gil, Y.; Ratnakar, V.; and Fritz, C. *Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI)*, Palo Alto, CA, 2011.

GIL, Y., RATNAKAR, V., CHKLOVSKI, T., GROTH, P., AND VRANDECIC, D. "Capturing Common Knowledge about Tasks: Intelligent Assistance for To Do Lists." *ACM Transactions on Interactive Intelligent Systems*, 2(3), 2012.

GOLD, K. AND B. SCASSELLATI. "A robot that uses existing vocabulary to infer non-visual word meanings from observation." *Proceedings of the Twenty-Second Annual Meeting of the Association for the Advancement of Artificial Intelligence* (AAAI), 2007.

GOLD, K., DONIEC M. AND B. SCASSELLATI. Learning Grounded Semantics With Word Trees: Prepositions and Pronouns. *Proceedings of the 6th IEEE International Conference on Development and Learning* (ICDL 2007), London, England, July 2007.

GREEN, T. R. G. AND M. PETRE. "Usability analysis of visual programming environments: a 'cognitive dimensions' framework." *Journal of Visual Languages and Computing,* Vol 7, 1996.

GREENE, S., DEVLIN, S., CANNATA, P., AND L. GOMEZ. "No IFs, ANDs, and ORs: A Study of Database Querying." *International Journal of Man-Machine Studies*, Vol 32, 1990.

GRICE, H. P. "Logic and Conversation." *In Syntax and Semantics III: Speech Acts*, P. Cole and J. Morgan (Eds), Academic Press, New York, NY, 1975.

HERBERG, J. S., SAYLOR, M. M., RATANASWASD, P., LEVIN, D. T., WILKES, D. M. "Audience-contingent variation in action demonstrations for humans and computers." *Cognitive Science*, 32, 2008.

HOC J. M. "Do We Really Have Conditional Statements in Our Brains?" *In Studying the Novice Programmer,* E. Soloway and J. C. Spohrer (Eds), Lawrence Erlbaum Associates, 1989.

HUFFMAN, S. AND J. LAIRD. "Flexibly Instructable Agents." *Journal of Artificial Intelligence Research*, 3, 1995.

HUO, J. AND W. COWAN. "Comprehending Boolean Queries." *Proceedings of the 5th ACM Symposium on Applied Perception in Graphics and Visualization*, Los Angeles, CA, 2008.

JOHN, B. E. AND D. E. KIERAS "Using GOMS for User Interface Design and Evaluation: Which Technique?" *ACM Transactions on Computer-Human Interaction*, 3(4), 1996.

KAY, A. "Computer Software." *Scientific American*, 251(3), 1984.

KELLEHER, C. AND R. PAUSCH. "Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers." *ACM Computing Surveys*, 37(2), 2005.

KIERAS, D. E. AND S. BOVAIR. "The role of a mental model in learning to operate a device." *Cognitive Science*, 8(3), 1984.

KIM E. AND B. SCASSELLATI. "Learning to refine behavior using prosodic feedback." *Proceedings of the 6th IEEE International Conference on Development and Learning (ICDL)*, London, England, July 2007.

KIM, E.S., LEYZBERG, D., TSUI, K.M. AND B. SCASSELLATI. "How People Talk When Teaching a Robot." *Proceedings of the 4th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2009.*

KIM, J. AND Y. GIL. "User Studies of an Interdependency-Based Interface for Acquiring Problem-Solving Knowledge." *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, New Orleans, LA, January 9-12, 2000.

KIRSCHNER, P. A., SWELLER, J. AND R. E. CLARK. "Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching." *Educational Psychologist*, 41(2), 2006.

KO, A.J., MYERS, B.A., AND CHAU, D.H. "A Linguistic Analysis of How People Describe Software Problems." *Proceedings of the Visual Languages and Human-Centric Computing Conference (VL/HCC)*, 2006.

KO, A.J., ABRAHAM R., BECKWITH L., BLACKWELL A., BURNETT M.M., ERWIG M., SCAFFIDI C., LAWRENCE J., LIEBERMAN H., MYERS B.A., ROSSON M.B., ROTHERMEL G., SHAW M. AND WIEDENBECK S. "The State of the Art in End-User Software Engineering." *ACM Computing Surveys*, 43(3), 2011.

KOEDINGER, K. AND J. ANDERSON. "Abstract Planning and Perceptual Chunks: Elements of Expertise in Geometry." *Cognitive Science*, 14(4), 1992.

KOSSEIM, L. AND G. LAPALME. "Choosing Rhetorical Relations in Instructional Texts: The Case of Effects and Guidances". *Proceedings of The Fifth European Workshop on Natural Language Generation*, 1995.

KOSSEIM, L. AND G. LAPALME. "Choosing Rhetorical Structures to Plan Instructional Texts". *Computational Intelligence*, 16(3), 2000.

LARKIN, J. AND H. A. SIMON. "Why a Diagram is Worth 10,000 Words." *Cognitive Science*, Vol 11, 1987.

LAU, T., DREWS, C. AND J. NICHOLS. "Interpreting Written How-To Instructions". *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, Pasadena, CA, July 2009.

LAURIA, S., BUGMANN, G., KYRIACOU, T., AND E. KLEIN. "Mobile Robot Programming Using Natural Language." *Robotics and Autonomous Systems*, Volume 38, 2002.

LEE, M. D., AND M. J. DRY. "Decision Making and Confidence Given Uncertain Advice", *Cognitive Science*, 30, 2006.

LEFEVRE, J. AND P. DIXON. "Do Written Instructions Need Examples?" *Cognition and Instruction*, 3(1), 1986.

LI, I., NICHOLS, J., LAU, T.A., DREWS, C., AND CYPHER, A. "Here's what I did: Sharing and reusing web activity with ActionShot." *Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI)*, 2010.

LIEBERMAN, H., PATERNO, F., AND, V. WULF (EDS). "*End-User Development*." Kluwer/Springer, 2005.

LINDEN, K. V. "Generating Precondition Expressions in Instructional Text." *Proceedings of the 32nd annual meeting on Association for Computational Linguistics,* 1994.

LINDEN, K. V. AND B. DI EUGENIO. "A Corpus Study of Negative Imperatives in Natural Language Instructions." *Proceedings of the 16th International Conference on Computational Linguistics* (COLING), 1996.

LINDEN, K. V. AND J. H. MARTIN. "Expressing Rethorical Relations in Instructional Text: A Case Study of the Purpose Relation". *Computational Linguistics*, Vol 21, 1995.

MAHLING, D.E. AND W.B. CROFT. "Relating human knowledge of tasks to the requirements of plan libraries." *International Journal of Human-Computer Studies*, Vol. 31, 1988.

MAHLING, D.E. AND W.B. CROFT. "Acquisition and Support of Goal-Based Tasks." *Knowledge Acquisition*, Vol. 5, 1993.

MANN, W. C. AND S. A. THOMPSON. "Rhetorical Structure Theory: Toward a Functional Theory of Text Organization." *Text*, 8(3), 1988.

MILLER, L. A. "Programming by Non-Programmers." *International Journal of Man-Machine Studies*, Vol 6, 1974.

MILLER, M. L. "A structured planning and debugging environment for elementary programming." *International Journal of Man-Machine Studies*, Vol 11, 1979.

MILLER, L. A. "Natural language programming: Styles, strategies, and contrasts", *IBM Systems Journal*, 20(2), 1981.

MYERS, B. A., PANE, J. F., AND A. KO. "Natural Programming Languages and Environments." *Communications of the ACM*, 47(9), 2004.

NARDI, B. A. "A Small Matter of Programming: Perspectives on End User Computing." *MIT Press*, Cambridge, MA, 1995.

NYBERG, E. AND T. MITAMURA. "Controlled Language and Knowledge-Based Machine Translation: Principles and Practice." *Proceedings of the First International Workshop on Controlled Language Applications (CLAW)*, 1996.

ONORATO, L. A. AND R. W. SCHVANEVELDT. "Programmer—Nonprogrammer Differences in Specifying Procedures to People and Computers." *Journal of Systems and Software*, Vol 7, 1987.

PANE, J. AND B. MYERS. "The Influence of the Psychology of Programming on a Language Design." *Proceedings of the 12th Annual Meeting of the Psychology of Programmers Interest Group*, 2000.

PANE, J. F. AND B. A. MYERS. "Tabular and Textual Methods for Selecting Objects from a Group." *Proceedings of IEEE International Symposium on Visual Languages*, Seattle WA, 2000.

PIETRAS, C. M. AND B. G. COURY. "The Development of Cognitive Models of Planning for Use in the Design of Project Management Systems." *International Journal of Human-Computer Studies*, Vol 40, 1994.

RESNICK, MITCHEL, JOHN MALONEY, ANDRÉS MONROY-HERNÁNDEZ, NATALIE RUSK, EVELYN EASTMOND, KAREN BRENNAN, AMON MILLNER, ERIC ROSENBAUM, JAY SILVER, BRIAN SILVERMAN, YASMIN KAFAI. "Scratch: Programming for All." *Communications of the ACM*, 11, 2009.

SCAFFIDI, C., SHAW, M., AND MYERS, B. "Estimating the Numbers of End User Programmers." *Proceedings of the 2005 Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2005.

SCHWARTZ, D. L. AND J. D. BRANSFORD. "A Time for Telling." Cognition and Instruction, 16(4), 1998.

SLOTTA, J. D. AND M. T. H. CHI. "Helping Students Understand Challenging Topics in Science Through Ontology Training." *Cognition and Instruction*, 24(2), 2006.

SMITH, E. AND L. GOODMAN. "Understanding Written Instructions: The Role of an Explanatory Schema." *Cognition and Instruction*, 1(4), 1984.

SOLOWAY, E., BONAR, J., AND K. EHRLICH. "Cognitive Strategies and Looping Constructs: An Empirical Study." *Communications of the ACM*, 26(11), 1983.

STEEHOUDER, M., KARREMAN, J., AND N. UMMELEN. "Making Sense of Step-by-Step Procedures". *Proceedings of IEEE Professional Communication Society International Professional Communication Conference and Proceedings of the 18th Annual ACM International Conference on Computer Documentation: technology & teamwork*, 2000.

SWARTOUT, W. R., PARIS, C., AND MOORE, J. "Explanations in Knowledge Systems: Design for Explainable Expert Systems." *IEEE Expert* 6(3), 1991.

THOMAZ, A. L. AND C. BREAZEAL. "Teachable robots: Understanding human teaching behavior to build more effective robot learners." *Artificial Intelligence*, 172:716-737, 2008.

THOMAZ, A. L. AND C. BREAZEAL. "Experiments in Socially Guided Exploration: Lessons learned in building robots that learn with and without human teachers." *Connection Science*, Special Issue on Social Learning in Embodied Agents, 20(2&3), 2008.

THOMAZ A. L. AND M. CAKMAK. "Learning about Objects with Human Teachers." *Proceedings of the International Conference on Human Robot Interaction* (HRI), 2009.

VAN LEHN, K., GRAESSER, A. C., JACKSON, G. T., JORDAN, P., OLNEY, A., AND C. P. ROSE. "When Are Tutorial Dialogues More Effective Than Reading?" *Cognitive Science*, 31, 2007.

VAN MERRIËNBOER, J.J.G. *Training Complex Cognitive Skills: A Four-Component Instructional Design Model for Technical Training*. Educational Technology Pubns 1997.

VAN MERRIËNBOER, J.J.G., KIRSCHNER, P.A., & KESTER, L. "Taking the load of a learners' mind: Instructional design for complex learning." *Educational Psychologist*, 38(1), 2003.

WALKER, T., KUNAPULI, G., LARSEN, N., PAGE, D. AND J. SHAVLIK. "Integrating Knowledge Capture and Supervised Learning through a Human-Computer Interface." *Proceedings of 6th International Conference on Knowledge Capture (KCAP)*, Banff, Canada, 2011.

WEBBER, B. L., BADLER, N., DI EUGENIO, B., GEIB, C. W., LEVISON, L. AND M. MOORE. "Instructions, Intentions and Expectations." *Artificial Intelligence*, 73(1-2), 1995.

WEILAND, W. AND B. SHNEIDERMAN. "A graphical query interface based on aggregation/generalization hierarchies." *Information Systems*, 18(4), 1993.

WINOGRAD T. AND F. FLORES. "Understanding Computers and Cognition: A New Foundation for Design." *Ablex Publishing*, Norwood, NJ.

WRIGHT, P. AND A. J. HULL. "How People Give Verbal Instructions." *Applied Cognitive Psychology*, Vol 4, 1990.

YOUNG, D. AND B. SHNEIDERMAN. "A Graphical Filter/Flow Representation of Boolean Queries: A Prototype Implementation and Evaluation." *Journal of American Society for Information Science*, 44(6), 1993.

YOUNG, M. "Using Grice's Maxim of Quantity to Select the Content of Plan Descriptions", *Artificial Intelligence*, 115(2), 1999.