

# Task Analysis by Problem Solving (TAPS): Uncovering Expert Knowledge to Develop High-Quality Instructional Materials and Training

Richard Catrambone  
Georgia Institute of Technology  
rc7@prism.gatech.edu

## ABSTRACT

*Task analysis is a method of identifying knowledge and procedures for performing a task or solving a problem. There are a variety of cognitive task analysis methods, but they often require significant overhead to learn and rely on having a subject matter expert (SME) defining how one should solve problems. This can be problematic because expertise often blinds people to the knowledge that novices in a domain need in order to solve problems or carry out tasks. I present a task analysis approach, Task Analysis by Problem Solving (TAPS), which focuses on uncovering procedural and domain knowledge that a learner needs to acquire. In TAPS the SME performs tasks or solves problems while justifying each step to the knowledge extraction expert (KEE) who is a domain novice. The KEE develops notes based on the solution procedures and justifications provided by the SME. Eventually the KEE uses the notes to solve problems provided by the domain expert; the notes get revised throughout all phases. The KEE reaches the point where, using the notes, he can solve all problems given by the SME. The notes can then guide the development of training and assessment materials. Due to the preceding process the materials are much more likely to cover the necessary details compared to materials that are developed through an SME's intuitions about what should be included.*

**KEYWORDS:** Task analysis, instructional design, training, assessment, training technology.

## 1. INTRODUCTION

When instructional designers or educational researchers create training or instructional materials, they must determine both the learning objectives and the specific knowledge (i.e., requisite procedural and declarative knowledge) to be taught. Task analysis (TA) is a method for identifying this knowledge. Task analysis is used, for

example, in the design of training and procedures [11], interface design [9], and general system design and redesign.

A completed TA can be used to develop learning and training materials. These materials might be designed with the goal of, for example, educating a specific population or assessing the effects of different instructional designs. Because the TA identifies the requisite knowledge for a learner, experimenters testing instructional designs can ensure that each design format contains the requisite information, and thus more carefully assess the presentation of that information via the different designs. Similarly, instructional designers can use the TA output as a guide to ensure they cover all the necessary information, or to create assessment instruments.

## 2. TASK ANALYSIS BY PROBLEM SOLVING (TAPS)

In this paper I describe Task Analysis by Problem Solving (TAPS), a task analysis method I have developed in which experts perform tasks while justifying their actions. Other TA methods share at least one of the following limitations: they do not directly address problem solving, they specify multiple, highly structured procedures that require significant training prior to beginning a task analysis, they do not focus on novice performance, or they do focus on novice performance but have the SME directly define and organize the knowledge. In contrast, TAPS focuses on problem solving (by both the SME and the analyst), does not use multiple, structured procedures that require prior training, and distills expert knowledge for novices.

TAPS is most appropriate for domains with tasks that emphasize solving problems or carrying out procedures. The aim of TAPS is to uncover or “rediscover” the procedures and knowledge that an expert uses to solve problems or carry out tasks. TAPS has been successfully applied in a variety of domains --- albeit most with an

academic bent --- in order to develop instructional materials in experimental settings (e.g., [1, 2, 3, 4]). The domains have included probability, physics, computer science, and chemistry. Additionally, this method has been used to guide the design of training for using software programs such as the Command Post of the Future (CPOF) [5].

In TAPS a subject-matter expert (SME) and a domain novice, who is also a knowledge extraction expert (KEE), work together. The SME identifies a set of typical problems or tasks that a learner should be able to solve or perform if the learner “understands” the part of the domain in question. Once these problems are selected, the knowledge extraction sessions may begin.

Initially, the SME solves a subset of the problems that he has identified while the KEE takes notes and makes the SME justify his steps. Later, the KEE attempts to solve some of these old problems and when he fails, gets help from the SME. Eventually the KEE attempts to solve novel problems and when he fails, gets help from the SME. Throughout this process the KEE continuously updates and reorganizes his notes; this reorganization allows the KEE to develop solution procedures that are independent of specific examples. When the KEE can solve all the old and new problems using just the notes and without the help of the SME, then these notes represent a complete TA document.

It can not be overstated the importance of the having the SME solve problems and then later for the KEE to solve problems. Solving problems helps “focus” the SME and later, serves as an excellent test of the notes when the KEE solves problems. The SME does not give a lecture nor explain how one would do the tasks. The SME’s explanations are filtered, clarified, shaped, and refined through questioning by a domain novice (the KEE).

TAPS, like other task analysis approaches, is part science and part art and an experienced KEE is important. For instance, how often to interrupt the SME depends on, among other things, whether the SME is getting annoyed with interruptions and whether the problem solving “flow” is disrupted. The issue of how “low level” to get in the task analysis document is also a judgment call based on the likely target audience (of materials developed using the TA document as a guide).

Because of the emphasis on problem solving, TAPS is suited for extracting procedural and declarative knowledge. The SME provides “just in time” theory. That is, conceptual explanations and domain theory are given, but only as they are needed in order to solve the particular problem. The aim of TAPS is to identify procedural knowledge and tightly tie it to the relevant

theory. I am not claiming domain theory does not matter; rather, I am claiming that the theory needs to be tied tightly (typically as justification) to the procedural details in order to create instructional materials that will be most useful to a learner.

### 3. OVERVIEW OF TAPS

The TAPS approach is summarized in Table 1. The goal of TAPS is to identify the procedural knowledge (and related declarative and strategic knowledge) that a novice must know in order to solve problems in the domain.

While the SME solves the problems, he talks about what he is doing. Although it sounds similar, this is different than a talk-aloud protocol. The goal is not to veridically record everything that the SME says he is doing. Rather, the KEE is requiring the SME to justify the steps he is doing. This can feel unnatural for an expert. An expert likely has proceduralized the strategic decision process and problem solving steps such that decisions are automatic or “obvious” and series of steps are combined. This strategic knowledge and combined steps often no longer require conscious thought of how to execute them and experts often do not verbalize all these steps (e.g., [15]). Experts typically forget what it is like to be a novice and the goal of TAPS is to help the expert re-identify the procedural and declarative details and decisions that he no longer thinks about.

In my experience, if the SME is left to freely explain, he will omit important procedural information and add unnecessary declarative information. To limit this oversight, the KEE does not just write down everything the expert says, but rather explores the SME’s statements as deeply as the KEE feels is needed to uncover the underlying steps and reasoning. The intent is to get the SME to explain the why for each step. The why represents theory. It is embedded into the procedural information in a “just in time” fashion.

This method of working with SMEs is motivated by empirical research on expertise in skill acquisition. Experts differ from novices in their domain knowledge and how they solve problems in their domain. For instance, experts have more robust conceptual schemas that involve multiple levels of superordinate and subordinate concepts, procedural knowledge, and conditions of applicability [7]. Additionally, experts are more likely to reason from domain principles [12], use long-term memory chunking, use a strategy of working-forward from the problem givens [7, 15], and have proceduralized subroutines. Because of the differences in base knowledge and problem representation, experts approach problems differently than novices, and thus their explanations about how they solve the problem are

typically incomplete relative to what novices need to know.

It is often difficult for experts to recognize the ways in which their reasoning and problem-solving within the domain is "compiled". In other words, they exhibit an expert blind spot [13, 14]. This expert blind spot is hypothesized to significantly affect how one decides to present content knowledge, and can result in emphasizing domain principles and structure that are primary and sufficient for an expert, but insufficient for a novice [14].

**Table 1. Summary of TAPS**

- Subject matter expert (SME) identifies set of problems/tasks that SME thinks learners/users should be able to solve/do if they "understand" the domain
- SME solves the problems (does the tasks)
- Knowledge extraction expert (KEE) requires SME to justify each step
  - takes detailed notes on the steps and the WHY for steps
  - theory of domain comes "just in time" and tied to procedural information
- KEE edits notes
  - reorganizes them
  - extracts procedures, decision rules, etc from examples
- SME solves problems again
  - KEE edits notes to fill gaps, fix inconsistencies, etc
- KEE attempts to solve old and new problems using notes
  - Reaches impasses, gets help from DE, revises notes until all problems can be solved by KEE

TAPS addresses the expert blind spot by employing the KEE who is a domain novice. The KEE's questions, mistakes, and misunderstandings that arise while working with the SME and solving problems are representative of issues likely encountered by other novices. This increases the likelihood that the TA (and corresponding instructional materials) will be at the appropriate level for a domain novice.

### **3. DISCUSSION OF SOME KEY FEATURES OF TAPS**

#### **3.1 SME and KEE**

The person doing the task analysis, the KEE, should be a relative domain novice. At least one domain expert is needed; I recommend using at least two SMEs when possible.

#### **3.2 Problem Selection**

To begin, the KEE asks the SME to select a set of problems or tasks that an individual who has sufficiently learned the targeted area of the domain should be able to solve or carry out; these tasks should vary in difficulty.

#### **3.3 SME Solves Initial Problems**

During the initial sessions the SME should solve one (or more) problems while talking through what he is doing and why. Generally, during the first problem the KEE is concentrating on taking notes. Any questions asked by the KEE are primarily for clarification of the procedure, rather than explanation. The KEE will likely be rapidly typing what the SME is doing and why (to the extent the SME is explaining why). As the SME solves the next problem (and the next) the KEE will ask more pointed questions in order to obtain the justifications (theory) for the steps. The KEE's notes from these initial problems should be as complete as possible and can be edited later. At this point, the KEE's notes will be tightly tied to the specific examples and their solutions.

While observing the SME, especially after the SME has completed the first problem or task, the KEE should interrupt to ask the SME to explicate assumptions he is making and theory-motivated reasons for the steps. Sometimes the theory will be nothing more than "this is the way the software is written" (for a TA on a particular piece of software) while other times it might be a fundamental domain principle or a convention in the domain. The KEE should look for inconsistencies in the rules and reasons that the SME is giving. For instance, it might be the case the SME initially says "if X then Y" but with further problems and questioning it turns out that it should have been "if X then: if Z1 then Y1; else if Z2 then Y2".

How much the KEE interrupts the SME also depends on the SME's clarity. If the KEE does not understand what the SME is doing, then the KEE should stop the SME and ask. It is also useful for the KEE to offer statements and restatements to the SME that can serve as a means of

checking the KEE's comprehension, and these might suggest a repackaging to the SME that he did not think of.

In addition to asking the SME for clarifications, the KEE will ask for justifications for the procedures the SME uses. Asking *why* is one of the most important questions. Questioning and rephrasing the SME's statements should not be overused, but is a useful tool if the KEE wants to make sure he understands something.

For the first session or two the KEE should keep the notes and questions tightly tied to the examples. The KEE should not attempt to force out domain principles other than those needed to solve that particular problem. This makes sense for a few reasons. One, the KEE presumably does not understand the domain well and given that his job is to record details, trying to grasp domain principles and high-level procedural goals is too hard while taking notes. Two, given the importance of making the SME justify what he is doing, frequent reference to domain principles will make things too complicated. Three, pushing on domain principles and high-level goals during the first couple of examples will slow the SME too much and disrupt his train of thought and possibly drop the SME into lecture mode. The KEE should try to keep the problem solving going, not let the problem solving grind to a halt by delving into generalities and high-level discussions. The KEE should keep the SME from lecturing and ensure that the SME provides domain theory only when needed to justify a particular step.

In the beginning sessions, the KEE often asks the SME to repeat terminology, defines symbols and abbreviations, and clarifies how to use this nomenclature. Essentially, the KEE is trying to understand how to use the jargon, conventions, and symbols in the domain. Likewise, the KEE often asks for definitions of concepts that the SME did not define when starting out; for the SME these terms are obvious, but they are not to the KEE.

Limiting the bounds of the TA document does sometimes require judgment calls about what to expect learners to know beforehand. The KEE can assume something about the learner's prior knowledge – for example using a dialog box, typing, or basic domain knowledge. The KEE does not necessarily write these assumptions in the task analysis document, but it is often there implicitly (and might be included in a final report, for example).

### **3.4 Focus SME on Solving Problems and Not "Lecturing"**

Knowledge extracted this way will focus on procedural information. Conceptual and theoretical information will also be included to justify why certain steps are carried out and to provide definitions of items in the domain as

needed. However, the KEE should try to keep the SME centered on the specific problem that he is solving, rather than general explanations about how to solve "problems like this one". The KEE must keep the SME from slipping into teaching mode and not let the SME structure the domain for the KEE. If SMEs could always do this well on their own then one would not need to perform a task analysis in order to improve instructional materials and instructors!

### **3.5 KEE Revises TA Notes**

After each task analysis session, the KEE goes through his notes without the SME present and condenses, fills in, reorganizes, and adds questions to be addressed by the SME in the next session. The KEE's aim is to identify and exploit redundancy in the process of separating procedures from the specific problems, and to identify relationships that he might not have been noticed during the TA session. The TA document is a living document in that it is repeatedly modified and improved during and after each TA session.

While there are no set categories of information to be included in the TA document, prior TA efforts have suggested that a good deal of the identified knowledge can be considered goals, subgoals, rules, subroutines, examples, definitions, and assumptions. The KEE should not overly formalize the TA document, however. Instead the KEE should concentrate on making it a complete, usable document. The strength of this TA method lies in getting the SME to explicate what he is doing and why; it is not as important how the KEE labels things. That said, the KEE should format the TA document to show hierarchical relationships (e.g., using bullets, indentations, and white space). The TA document is a text heavy document; there are no figures other than if a figure is needed to illustrate a portion of the problem or task. Table 2 shows a sample of a TA document that was produced while conducting a task analysis of the Command Post of the Future [5]. Figure 1 shows the relevant screen shot to which the text refers (Figure 1 is not part of the task analysis; it is presented here for clarity).

Over multiple session-revision cycles, the TA document will evolve. Initially, the problem solving procedures in the document will be tightly tied to the individual examples carried out by the SME. After several sessions (and subsequent revisions of the TA document), however, the procedures and steps should become less tied to the specific examples. The result is akin to a collection of small procedures (and conditions for using them) that can be flexibly applied to solve problems or carry out tasks in the part of the domain under analysis.

**Table 2. Sample Text from Task Analysis of Command Post of the Future (CPOF)**

- Create a Unit
- Get Unit/Event Palette from Frame Dispenser
- Click on Unit tab (if not already the selected tab)
- Drag any Unit to desktop
- Use BACKSPACE key to remove "untitled" and give it name (e.g., 2BCT)
- Type grid coordinates into grid coordinate field
- Select type of Unit (e.g., infantry)
- Use drop down boxes to select features (e.g., blue, friendly, brigade)
- Drag the window to the 2D Map anywhere
  - It might disappear if grid location for the Unit is not on current Map, but if the user goes to Map with that grid location on it, the Unit will be displayed there
- Can make several of these and Clone each on to Map

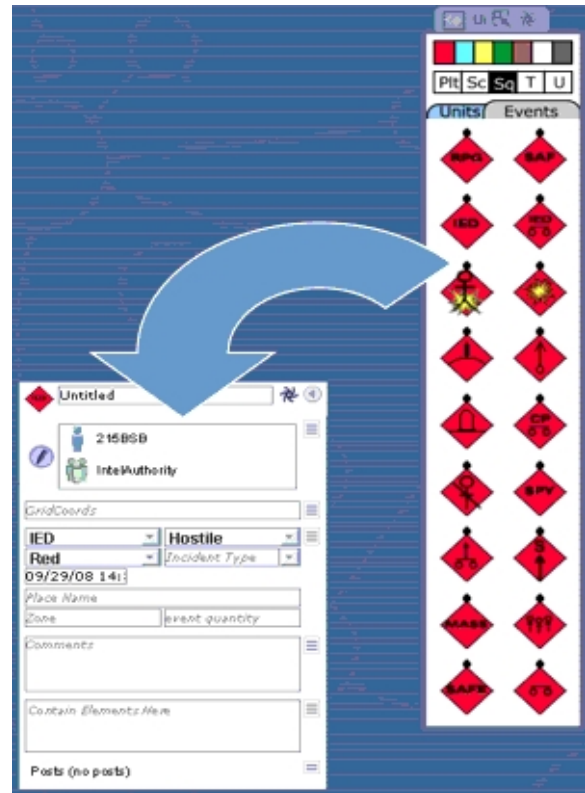
### 3.6 KEE Solves Problems

After the SME has solved the set of problems, the KEE will solve some problems using the TA document. This is not a test of the KEE's knowledge but rather a test of the notes. The KEE should first solve the same problems the SME solved. This serves as an initial test of the notes and helps reveal missing details and mistakes. It is crucial that the SME be available to help the KEE when asked. It is inevitable that the KEE will run into impasses and will need help.

The KEE will revise the TA document as he works on problems including new ones supplied by the SME. The TA document will also be edited after each of these sessions. The KEE should continue revisions until he can use the TA document to solve all problems provided by the SME. At this point the TA document is considered complete. This is not a formal proof, but a good heuristic for assessing completeness of the TA.

### 3.7 Sharing TA Document with SME

At this point the TA document can be shared with the SME for comments. However, the SME's comments must be taken cautiously. That is, the SME is likely to have opinions about the domain theory and about pedagogy; such topics are not central to the TA document because the TA document itself is not an instructional device. The most useful SME comments will concern fact correctness and nomenclature.



**Figure 1. Command Post of the Future (CPOF) Unit/Event "Palette"**

### 3.8 Challenges Working with SMEs

Getting the SME to explain his reasoning is a critical part of TAPS. Often the expert has a well-formed end goal and works both backward from the goal and forward from the givens when mapping out the method. In moving forward and backward the SME might encounter decision points that determine the solution path. Because the expert might do this processing implicitly and thus fail to vocalize the steps, getting the SME to clearly and accurately explain this processing requires persistent and effective prompting by the KEE.

After the SME has begun solving a problem, he might make a critical decision after which every successive step is based on former steps and decisions. It is important for the KEE to force the SME to go slowly, to retrace his steps. The SME might think that it is obvious which steps to do, but a novice will not. Often the SME will not explicate these decision points or end goals. Instead, the SME might use phrases such as "so I can tell that" or "what has to happen next is".

Another technique for extracting justifications is for the KEE to ask the SME to make connections between his problem solving procedure in the current problem and the

procedure used in the preceding problem. The KEE might ask, "What are the cues you are using to select this approach, rather than another approach?" or "How do you know when to use this [theorem, procedure, method]?"

The SME might have difficulty explicating his reasoning or his heuristics, but it is the KEE's job to try to guide him to be able to. Nevertheless, even an experienced KEE working with a motivated SME might be unable to get the SME to verbalize his reasoning. One solution is for the KEE to review the notes and then attempt to formulate a rationale or strategy himself. Moreover, when the KEE begins solving problems, he might be able to formulate the rationale. The KEE can later ask the SME about this rationale to determine if it is correct, and revise it if not. Additionally, using multiple SMEs sometimes circumvents the problem of the SME not being able to explain his reasoning. Remember, the goal is to identify the relevant procedures and conceptual knowledge, not necessarily the SME's particular approach or idiosyncrasies.

A task analysis of a electronic circuit simulation system for training troubleshooting provides an example of an expert not mentioning an important detail that the KEE discovered only by solving problems. In the simulation, the voltmeter's red lead must be dragged to the right side of a fuse when testing current flow. The SME had not brought up the right/left issue during any of the previous three sessions, probably because it had become automated. Further, the KEE had not asked whether there was a difference between the two sides of the fuse because he did not realize that the left side of the fuse was an option until attempting the task himself. As a result, when the KEE attempted to solve a test problems he failed. This illustrates why it is important for the KEE to solve problems; expertise can blind the SME to important procedural information, and the KEE might not recognize subtle decisions and steps until he attempts to solve problems.

Another example is from the CPOF software (see Table 2 and Figure 1). The SME did a few tasks that involved dragging "unit" icons (e.g., to represent friendly or hostile forces) from the "Unit/Event Palette" onto a map. Sometimes though the unit that was dragged did not match the goal of the task. Only by questioning the SME, and doing tasks himself, did the KEE discover that in practice it is easier/faster to drag a random unit onto a map and then select options from the unit menu to turn the unit into the desired type of unit. One always has to make adjustments to a unit when it is dragged onto a map, so there is no particular reason to drag the "right" unit initially.

## 4. THE TA DOCUMENT

The KEE should write all that seems important and, after a session has ended, when reviewing the notes, decide what should be included, excluded, or condensed. Using the mantra of "a TA document contains the information needed to solve problems in a domain" it is not necessary to preserve everything; the goal is not to collect a protocol of all that the SME says and does. The KEE's initial notes will contain a variety of information (comments by the SME that turn out to not be useful, notes to oneself, etc.) that is ultimately removed, revised, or embedded somewhere else in the TA document. Using this reasoning, the TA document does not typically contain information about a mistake (or "typical mistakes"). Warning about mistakes is more of an instructional design issue.

When reviewing the document the KEE might find mistakes or incompleteness; these can be fixed or flagged for follow-up with the SME. During revisions, the KEE will identify and prepare questions for the next session. Often it is useful to have the SME do the problem that generated the question again, as a way to check for consistency in solution procedure, and as a means to ask questions while the expert is solving problems.

The KEE should let the needs of the project determine whether he imposes a classification system on the TA document. If so, the KEE should develop the classification system only after several session-revision cycles. I do not recommend making an *a priori* classification system and then "fitting" the TA document into this organization, nor letting the SME develop the classification system. This reorganization should be done after TA sessions, never during the session; when taking notes the KEE should not worry about any classification scheme.

After repeated sessions and revisions of the document the KEE will become more familiar with the domain and example-independent subprocedures will emerge. These example-independent subprocedures are collections of subprocedures that learners can combine and arrange in order to solve problems in that domain. These subprocedures emerge, in part, because the KEE has repeatedly forced the SME to justify his decisions and solution steps for a set of problems. Across problems the decision rules increase and therefore the documented procedures become independent of a specific example's solution procedure. Additionally, when the KEE begins to solve problems himself, commonalities among problems will become apparent and guide identification of example-independent subprocedures.

When these subprocedures are used in larger procedures they may be identified by name rather than being completely restated. Thus, larger subprocedures might include references to multiple subprocedures, as well as notes about how to combine and use the subprocedures. With many sessions and revisions the TA document will become less tied to specific solution paths.

## 5. USING THE TA DOCUMENT TO CREATE INSTRUCTIONAL AND ASSESSMENT MATERIALS

Once the TA document is completed, it can serve as the base for designing training and instructional exercises, for answering questions about what information should be presented, and for guiding development of knowledge assessment measures. Although the TA specifies the content to be conveyed, how this information is conveyed or assessed is a separate design decision. To illustrate how the TA document is a first step in designing instructional materials, and how those instructional materials can then be manipulated to test instructional design decisions, I briefly review two of research programs that were predicated upon using TAPS.

Designing instruction based on knowledge identified in the TA allows researchers to increase the likelihood that the instructional conditions convey the same information through different delivery mechanisms (e.g., animations or static diagrams). Thus, conclusions about differences between instructional designs are based upon holding the content constant, while varying only the presentation or type of interaction.

I have used TAPS in several studies of teaching novices about computer science concepts, such as stacks [6] and binomial heaps [10]. In both studies our emphasis was on testing different ways of using algorithm animations to improve learning. Catrambone and Seay [6] compared still frames to animations, using either TA-designed materials or standard materials. Gane and Catrambone [10] tested different methods for students to interact with the animations by manipulating learner involvement (choose examples to view versus assigned examples to view) and scenario (study the animations versus use the animations to answer homework questions). In both studies, we used TAPS to reveal information necessary for novices, and using that information, to design the learning materials, the instructional text, animations, and examples. Further, we used the TA to design assessment materials (i.e., isomorphic, near transfer, and far transfer items).

Catrambone and Seay [6] compared the new TA-based text to a conventional text, borrowed from a popular

textbook [8]. As an additional manipulation, Catrambone and Seay designed learning aids (static diagrams and animations) based on their task analysis. Compared to the conventional text, the TA text improved scores on both near and far transfer items. Additionally, the conventional text, when supplemented with animations that were based on the TA, improved scores. These findings suggest the TA was effective in identifying critical information, and that this information could be implemented by authoring either (1) new text or (2) learning aids to supplement the existing text [6].

## 6. CONCLUSIONS

TAPS has been useful in identifying the knowledge for learning materials and is particularly useful for revealing procedural and conceptual knowledge a learner needs to know. Compared to other popular cognitive task analysis methods, TAPS is focused on problem solving and does not use highly structured methods. Instead, the KEE follows general heuristics for how to interact with the SME that flow from the TAPS approach (e.g., keep the SME focused on solving the specific problem, make SME justify steps, revise notes between sessions, etc.).

These heuristics are based on experience using TAPS and cognitive theories of problem solving such as the subgoal-learning model [4]. TAPS is not tied to a cognitive architecture and does not demand formalized conventions for representing the extracted knowledge. This allows for flexibility across multiple domains, reduces the time to learn to use the method, and makes it a powerful tool to improve instruction and assessment.

## ACKNOWLEDGEMENTS

I thank my graduate students and colleagues and the many subject matter experts I have worked with across a large variety of projects.

## REFERENCES

- [1] R. Catrambone. "Improving examples to improve transfer to novel problems." *Memory & Cognition*, 22, pp. 606-615, 1994.
- [2] R. Catrambone. "Aiding subgoal learning: Effects on transfer." *Journal of Educational Psychology*, 87, pp. 5-17, 1995.
- [3] R. Catrambone. "Generalizing solution procedures learned from examples." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22, pp. 1020-1031, 1996.
- [4] R. Catrambone. "The subgoal learning model: Creating better examples so that students can solve novel

- problems." *Journal of Experimental Psychology: General*, 127, No. 4, pp. 355-376, 1998.
- [5] R. Catrambone, R.L. Wampler, and M.L. Bink. "Determining a Critical-Skill Hierarchy for Command Post of the Future (CPOF)." *ARI Research Report 1906*, Alexandria, VA: U.S. Army Research Institute for the Behavioral and Social Sciences, 2009.
- [6] R. Catrambone and A.F. Seay. "Using animation to help students learn computer algorithms." *Human Factors*, 44, No. 3, pp. 495-511, 2002.
- [7] M.T.H. Chi, R. Glaser, and E. Rees. "Expertise in problem solving". In R. S. Sternberg, (Ed.), *ADVANCES IN THE PSYCHOLOGY OF HUMAN INTELLIGENCE*, Vol. 1. Erlbaum, Hillsdale, NJ, pp. 1-75, 1982.
- [8] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *INTRODUCTION TO ALGORITHMS*. MIT Press, Cambridge, MA, 1990.
- [9] D. Diaper (Ed). *KNOWLEDGE ELICITATION: PRINCIPLE, TECHNIQUES AND APPLICATIONS*. Springer-Verlag, New York, 1989.
- [10] B.D. Gane AND R. Catrambone. "Give learners questions to answer while they watch animated examples." In S. A. Barab, K. E. Hay, and D. T. Hickey (Eds.), *Proceedings of the Seventh International Conference on Learning Sciences - ICLS '06*, pp. 922-923, Erlbaum, Mahwah, NJ, 2006.
- [11] B. Kirwan and L.K. Ainsworth. *A GUIDE TO TASK ANALYSIS*. Taylor & Francis, London, 1992.
- [12] J.H. Larkin. "Processing Information for Effective Problem Solving". *Engineering Education*, 70, No. 3, pp. 285-288, 1979.
- [13] M.J. Nathan and K.R. Koedinger. "An investigation of teachers' beliefs of students' algebra development." *Cognition & Instruction*, 18, pp. 209-237, 2000.
- [14] M.J. Nathan and A. Petrosino. "Expert blind spot among preservice teachers." *American Educational Research Journal*, 40, pp. 905-928, 2003.
- [15] D.P. Simon and H.A. Simon. "Individual differences in solving physics problems." In R. Siegler (Ed.), *CHILDREN'S THINKING: WHAT DEVELOPS?* Erlbaum, Hillsdale, NJ, 1978.