An Overview of the Task Knowledge Structures Model of Task Analysis

Tool/Model Paper—Final

Submitted by Beth Jo Woodell

Syracuse University School of Education

IDE 712 - Analysis for Human Performance Technology Decisions

Dr. Rob S. Pusch

April 4, 2023

Origin and Background of the Task Knowledge Structures Model

People have been thinking about how people interact with computers pretty much since the first computer, ENIAC, was introduced to the public in 1946. Fitzpatrick (2018) mentions that since the mid-20th century, scholars have been pondering the concerns, methods, and interdisciplinary connections between humans and computers—human-computer interaction, or HCI—and how they evolve with each innovation a new generation of computing devices brings. When describing, analyzing, and modeling the knowledge and skills a human user must develop in order to interact with an inanimate system or device, the metaphor of a computer as a brain made out of semiconductors (or the brain being a computer made of meat, if you prefer the computer to have the upper hand) comes in handy. Thinking about the need to develop a framework for explaining how humans interact with computers from a more cognitive and less behaviorist perspective led Card, Moran, & Newell (1983) to propose the GOMS (Goals, Operators, Methods, Selection) model for analyzing these interactions. GOMS was originally used on text editing tasks; language processing, being unique to *Homo sapiens*, is an ideal setting for applying the information processing concepts in cognitive psychology that GOMS newly offered at the time.

GOMS and a few other models, including Task Knowledge Structures (TKS), which I discuss in this paper, arose in the early 80's as cognitive task analysis became of interest. TKS, according to Jonassen, Tessmer, & Hannum (2009), differed from GOMS and other cognitive task models in that GOMS concerned itself with extremely detailed, even micro-level user interactions with the computer. TKS doesn't drill down that deep. It concerns itself more with

representative, generic tasks, and as such is useful for characterizing tasks of various levels of complexity.

"Various levels of complexity" is an understatement when it comes to the types of performance analysis assignments I am likely to encounter when I return to the consulting world. This is what first attracted me to (TKS), first identified by Johnson, Johnson & Wilson (1995). Though the majority of instruction I will design and develop will be for training enterprise resource planning (ERP) programs such as SAP, the TKS model is useful for analyzing and cataloging tasks with multiple levels of complexity in other industries also.

Purpose and Function of the TKS Model

Because TKS arose as a concrete representation of cognitive schemata, and because it is assumed that such schemata are hierarchical in nature, it is logical to employ TKS to represent task knowledge hierarchically. Any complex task can be broken down into a set of simpler tasks, which in turn can be further broken down into even simpler tasks. At some point, a level of simplicity is reached where we can't break down the task any further, or to try to do so would be impractical or not useful. Successive breakdowns of each of the tasks that roll up to the specified goal yield a hierarchy of what needs to be done to accomplish a task. Identifying all of the tasks and subtasks and placing them in their correct positions in the hierarchy gives us a visual model of the goal and what needs to be taught in order to achieve it.

Due to this hierarchical nature, TKS would work to analyze complex tasks in any topic that involves executing simple tasks to create a complex object or result, and then executing another series of tasks to create an object or result of even greater complexity. Some non-computer areas I can see leveraging the TKS method include machine assembly (assemble parts,

then assemble the assembled parts to create larger parts, then assemble those “super-parts” to create a motor or engine), apparel construction, or even medical or dental surgery.

Not every procedure that comprises a sequence of tasks is amenable to TKS analysis. For example, a short sequence of simple tasks, such as the steps needed to brew a cup of coffee, might not warrant a TKS analysis. Incorporating the brewed coffee in a more complex culinary dish, such as baking a mocha almond cake, might be an occasion for TKS (bake the cake has a series of steps; making the frosting has a series of steps, including brewing the coffee; and assembling the cake and garnishing it have series of steps as well).

Assumptions, Advantages, Disadvantages

In all the examples I list above, there is a tangible result at the end of all task execution. For example, in SAP, the result of a hierarchy of steps the user executes might be a business document such as a purchase order, sales contract, or invoice. In the case of machine assembly, the result might be a properly built engine or motor; and in the case of the cooking steps, the end result is hopefully a great-tasting cake. TKS assumes that the activities involved in a complex process are oriented toward a specific goal (Johnson, Johnson, &Wilson, 1995). The goal provides a structure for the user to learn, understand, and recall the components of the task, including the tools, steps, and the environment where the tasks are performed. In the case of SAP, the environment may be a specific server that the user logs on to in order to create the document. If the user knows they have to generate a purchase order, that should trigger recall of which server they need to log on to in order to do that. A visual hierarchy, such as the organizational diagram in Figure 1 below, may or may not be necessary. The user, if a master of the task structure, already possesses the hierarchy—in their mind. If a visual is desired, however, one can be created fairly easily.

The Task Knowledge Structures Model provides several advantages in instructional design, when implemented in an appropriate scenario.

- It identifies user and task requirements by providing a supporting methodology for designers to follow. Having to think about tasks that come both upstream of a given task and downstream of a task gives the instructional designer a framework for what to include or exclude in a course design.
- TKS focuses on work tasks—what people do, not just what they know or understand. Other Critical Task Analysis (CTA) methods focus on minute details of user interactions, which may be key for those methods but not necessarily for TKS.
- Because constructing a hierarchy of tasks forces the designer to think of both antecedents and successors to tasks, TKS provides a more comprehensive method for analyzing tasks.
- Because the TKS process naturally identifies all preceding and following tasks in a structure, a kind of road map for rapid prototyping of instructional materials is created practically automatically. The designer, after going through the process, knows exactly what tasks to include in a given course and, by extension, what not to include. Once again using an example from SAP, often I am called upon to design and develop training courses that involve identification and proper use of various data types such as customer and supplier account numbers, general ledger account numbers, and product or part numbers. However, in no case is it incumbent on the users learning such courses to create those data from scratch. An Accounts Payable clerk doesn't have to and in fact shouldn't create a new supplier account every time they want to pay an invoice. Therefore, creating the supplier account isn't part of the clerk's task hierarchy, and I don't have to include

that task in the design of an Accounts Payable training course. (I do include that task in a different training course, with a different goal and a different audience.)

TKS does have some disadvantages and constraints.

- Because TKS is an empirical method, it isn't the best to analyze those learning scenarios where non-task learning, or strictly cognitive learning, is involved.
- It doesn't work well with task analysis techniques which are not concerned with knowledge, such as ability profiling (Fleishman et al., 1984) and other techniques which have an evaluative role in assessing the complexity of task performance but have no explicit method of task analysis.
- TKS is not meant to be an evaluative tool.

Structure and Use of the Tool

A high-level summary of the steps involved in conducting Knowledge Analysis of Tasks (Johnson & Johnson, 1991) includes:

1. Collect information about the task using data-gathering and knowledge acquisition techniques.
2. Identify the knowledge components used in task performance.
3. Identify representative, central, and generic properties of tasks.
4. Construct the task model.
5. Communicate the TKS to skilled performers.
6. Skilled performers validate the structures or change them.

A complete list of the steps involved in conducting a TKS analysis is in Appendix A.

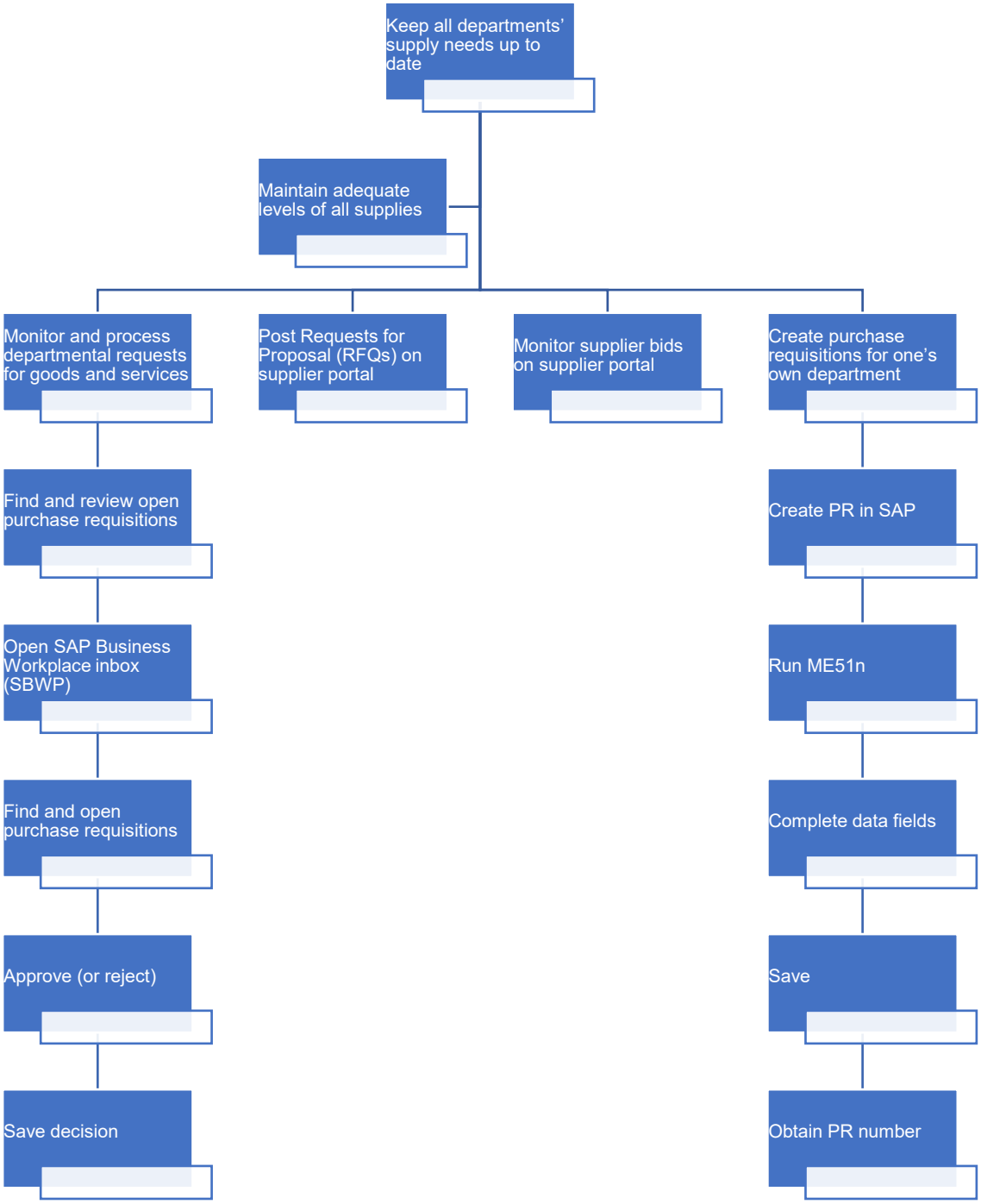

Figure 1 Task Knowledge Structures applied to a portion of the supply chain process in SAP.

An Example of TKS in Practice

Johnson and Hyde (2003) present a complex yet interesting application of TKS in a then-underrecorded setting, namely that of collaborative goal achievement. Until that time, most research had revolved around individual task performance. Johnson and Hyde applied the basics of TKS to a decidedly un-tech task: solving a jigsaw puzzle! The article begins by discussing basic features of and external factors affecting group collaboration, as well as the mechanics of collaboration. They then justify their selection of the problem, since solving a jigsaw puzzle isn't exactly a prime example of HCI. They note that certain elements of the solution process, such as declarative knowledge of the objects and their relationships, high-level planning, and different applications of knowledge emerging at different stages of the puzzle solution, mirror similar solution processes in HCI. They note that "…that there are significant HCI considerations in constructing a tool to support individual and collaborative construction of jigsaws in terms of presentational, structural, navigational, shared resource and accessibility issues." (p. 347) And conveniently, this type of puzzle has only one correct solution, which is given to the participants initially (it's right on the box!), so there is no confusion about what the top-level goal is and therefore no opportunity for conflict of opinion on the goal.

Johnson and Hyde then discuss the methodology of the study, including participants (two male graduate students in computer science), setting, and puzzle features (two dinosaur puzzles of 120 pieces each, 292 mm square). The subjects were videotaped solving one puzzle individually, and some baseline features of their approaches established. The subjects then collaborated on the same puzzle plus a new puzzle. Next, after discussing the results of the two solution sessions, the authors describe the TKS model in some detail, with some new (at the time) notions concerning collaboration and its influence on knowledge structures, apparently a poorly studied topic up to that point. It turned out that the TKS they derived from the students' solution activities was quite different from what they normally saw with individual subjects. Knowledge structures were higher-order, more general, and as such applicable to a greater variety of tasks. The constructs were different enough that the authors termed them Fundamental Knowledge Structures (FKS, p. 356).

Johnson and Hyde constructed several hierarchies for the TKSs displayed by both subjects, including a structure displaying the collaboration on the second puzzle the subjects solved. An example is given in Figure 2 below. They list eleven features of the problem that TKS models could explain at the time, plus nine additional features TKS did not capture adequately, for example what the subjects do or do not do in the absence of a given behavior, or what they do in an adversarial situation.

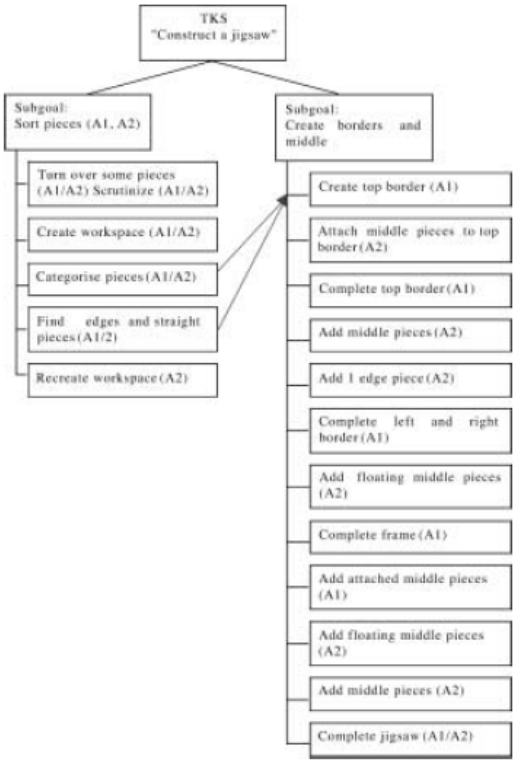

Figure 2 TKS diagram for two subjects solving a jigsaw puzzle.

In the discussion of the findings, Johnson and Hyde note two components of constructing a task analysis model for collaborative settings: the declarative/procedural knowledge, and the knowledge about how to work in a group. They propose an FKS for collaboration based on these observations. They then discuss the factors involved in an FKS model that are similar to and dissimilar from the TKS model in use at the time.

This was a complex subject and contained information, such as the postulation of the FKS model, that I was not expecting to see. Some of the conclusions might be beyond my current comprehension. Still, it was interesting to see a TKS model applied in a novel and non-computer-related learning environment. It might also be interesting to try to apply the FKS model to a learning scenario I encounter in the future.

References

Card, S. K., Moran, T P., & Newell, A. (1980). Computer text-editing: An information processing analysis of a routine cognitive skill. Cognitive Psychology, 12, 32-74. In Jonassen, D.H., Tessmer, M., & Hannum, W. H. (2009). *Task analysis methods for instructional design*. New York, NY: Routledge.

Fitzpatrick, G. (2018). A Short History of Human Computer Interaction: A People-Centred Perspective. In Proceedings of the 2018 ACM SIGUCCS Annual Conference (SIGUCCS '18). Association for Computing Machinery, New York, NY, USA, 3. https://doi.org/10.1145/3235715.3242569

- Fleishman, E.A., Quaintance, M.K., & Broedling, L.A. (1984). Taxonomies of Human Performance: The Description of Human Tasks.

GOMS (February 14, 2023). In *Wikipedia*. https://en.wikipedia.org/wiki/GOMS

Johnson, H., & Hyde, J. (2003). Towards modeling individual and collaborative construction of jigsaws using task knowledge structures (TKS). ACM Transactions on Computer-Human Interaction, 10(4), 339-387. https://doi.org/10.1145/966930.966934

Johnson, H. & Johnson, P. (1991) Task knowledge structures: Psychological basis and integration into system design. Acta Psychologica, 78, 3-26. In Jonassen, D.H., Tessmer, M., & Hannum, W. H. (2009). *Task analysis methods for instructional design*. New York, NY: Routledge.

Johnson, P., Johnson, H., & Wilson, S. (1995) Rapid Prototyping of User Interfaces Driven by Task Models. In J. M. Carroll (Ed.) Scenario-based design for human

computer interaction (pp. 209-246). John Wiley & Sons, Inc.

Johnson, P., Johnson, H., Waddington, R. & Shouls, A. (1988). Task-related knowledge structures: analysis, modelling and application. People and Computers IV. 35-62.

Johnson, P., Johnson, H., Waddington, R. & Shouls, A. (2001). Task-related knowledge structures: analysis, modelling and application.

Markopoulos, P, Wilson, S., & Johnson, P. (1994). Representation and use of task knowledge in a user interface design environment. Computers and Digital Techniques, IEE Proceedings -. 141. 79 - 84. 10.1049/ip-cdt:19949996.

Appendix A

Jonassen, Tessmer, & Hannum (2009) list the detailed steps involved in performing a knowledge analysis of tasks:

1. Collect information about the task using data-gathering and knowledge acquisition techniques.

1.1. Observe skilled performers (more than one for each task being analyzed) in their workplace settings

1.1.1. Record their actions.

1.1.2. Record the tools and equipment they use while performing.

1.2 Interview skilled performers in their work context.

1.2.1 Performers describe activities they engage in, including the objects (tools, models, signs they use).

1.2.2 Performers demonstrate and think aloud the procedures they use for each of their activities, including the technical aspects of their performance.

1.2.3 Performers describe their performances retrospectively (abstracted replay of performance). Video tape their performance and have the performer describe the actions, assumptions, and decisions while replaying the video.

1.2.4 Generate frequency counts of how often a task component is used or referred to across tasks.

1.3 Repeat 1.1 and 1.2 until you have a full understanding of the task.

2. Identify knowledge components used in task performance.

2.1 Identify goals and subgoals. Use one or more of the following techniques

2.1.1 Ask questions in interview about goals and sub goals of task.

2.1.2 Analyze manuals or textbooks which decompose task.

2.1.3 Construct a tree or hierarchical diagram of goals connected to subgoals.

2.1.4 Identify phases of task from observations or think-alouds.

2.2 Identify procedural knowledge. Use one or more of the following techniques:

2.2.1 Ask questions in interview how user performs task. Ask "what do you do if...". Ask about strategies used to perform subtasks.

2.2.2 Use think-alouds or abstracted replays of performance.

2.2.3 Use card sort, where designers lists each action involved in performance and performer organizes into proper sequence.

2.3 Identify object-action pairs. Use one or more of the following techniques:

2.3.1 Identify objects of actions in instruction manuals.

2.3.2 Question performer in abstracted replay about each object used and the action performed on it.

2.3.3 Ask performer to list all of the objects involved in task and the actions performed on them.

2.3.4 Observe performance and note all objects.

3. Identify representative, central, and generic properties of tasks.

3.1 For each object-action identified in 2.3, describe how critical or central to the task it is.

3.1.1 Construct two separate lists, one of the actions and one of the objects that have been identified in some way by the task performer.

3.1.2 Note frequency of action and object in the list and remove all repetitions from list.

3.1.3 Have performers rate the importance of each subtask; or have performer rank order cards with each subtask listed on them.

3.1.4 Choose generic actions and objects by identifying the most frequently occurring items or by grouping like terms by asking independent judges to "group together the actions (objects) which go together, or are the same kind of action (object)." Identify a generic label or term for identifying the action or object.

3.2 For each procedure identified in 2.2, describe how critical or central to the task it is.

3.2.1 List all of the procedures and the goals for which they are used.

3.2.2 Note the frequency of each procedure in a subgoal and across subgoals.

3.2.3 Note the frequency of each subgoal across task instances and performers.

3.2.4 Have performers rate the importance of each subtask; or have performer rank order cards with each subtask listed on them.

3.2.5 Reduce the lists to comprehensive and non-repetitive lists with each procedure and subtask appearing only once.

4. Construct task model (TKS).

4.1 Construct goal structure as hierarchy of goals that may be performed simultaneously (goal structure does not necessarily imply sequence of performance).

4.1.1 Describe subgoals required to fulfill goal of task.

4.1.2 Describe subgoals required to complete those subgoals. Repeat until all subgoals are identified. The bottom level subgoals represent procedures of actions that must be taken in order to fulfill the sub goal at the next higher level.

4.1.3 Construct hierarchy diagram of goals and subgoals.

4.2 Describe procedures required to fulfill bottom subgoals.

4.2.1 List sequence of actions involved in the procedure.

4.3 Describe objects used on the performance of the task being analyzed.

4.3.1 For each object, describe object properties or characteristics.

4.3.2 For each object, describe prototypical example.

5. Communicate the TKS to skilled performers.

6. Skilled performers validate the structures or change them.