Chapter 11
Methodology I: Task Analysis

Abstract Task analysis (TA) is a useful tool for describing and understanding how people perform particular tasks. Task analyses can be used for several purposes ranging from describing behavior to helping decide how to allocate tasks to a team. There are several methods of TA that can be used to describe the user's tasks at different levels of abstraction. We describe some of the most commonly used methods and illustrate the use of TA with some example applications of TA. TA is widely used but when using TA there are considerations to keep in mind such as the fact that many approaches require an initial interface or specification, and that many do not include context multiple users or ranges of users. These considerations help describe where and when TA can be successfully applied and where TA will be extended in the future.

11.1 Introduction

You should by now be familiar with the idea that when we are designing user-centered systems we are thinking about particular people doing particular tasks in a particular context. So far in this book we have looked in some detail at people-related aspects, and at some of the context-related aspects. In this chapter we focus more on task-related issues.

To understand what people do when they perform a task we use a technique called task analysis (TA). TA provides a way to describe the users' tasks and subtasks, the structure and hierarchy of these tasks, and the knowledge they already have or need to acquire to perform the tasks. Using these descriptions it becomes possible to predict how long users will take to learn a task, and how long they will take to perform a task. In this chapter we will mostly focus on the sorts of tasks that are nowadays carried out using some form of computer-based system.

TA can be used in most stages of system development. Before carrying out a TA, however, Diaper (2004) suggests that you first identify the stages of

F. E. Ritter et al., *Foundations for Designing User-Centered Systems*,
DOI: 10.1007/978-1-4471-5134-0_11, © Springer-Verlag London 2014

development when TA will be used, and then ask four interrelated questions for each of the identified stages:

1. What knowledge do you want the TA to provide? Frequently this is not an easy question to answer and your answer may need to be refined as you start to learn more and uncover interesting issues.
2. What format should the output of the TA take? The output from a TA may not map directly onto software engineering representations, so the contribution of a TA may sometimes be indirect, e.g., building models of the world through TA can improve the models that are used in software engineering.
3. What data can be collected? Data can be collected using several methods such as observing performance, interviews, document analysis, training program analysis, and analysis of existing systems.
4. Which method should be used? The choice should be based on knowledge about the input data and the desired output data, rather than familiarity with a particular method.

The results of a TA can be prescriptive or descriptive, depending on the aims of the analyst. Prescriptive analyses show how the user should carry out the task, and hence are associated with normative behavior. Prescriptive analyses can be used to help create safety procedures, and standard operating procedures, which prescribe how the user should carry out particular tasks. In contrast, descriptive analyses show how users really carry out the task, and are hence associated with actual behavior. Descriptive analyses are usually more situation specific than prescriptive analyses. It is important to be able to distinguish between the two, however. A prescriptive analysis of car driving, for example, would not include anything about driving above the speed limit; a descriptive analysis would be more likely to include details about people driving faster than the speed limit (perhaps only for particular sections of the road network).

When thinking about the different types of TA that are available, it is important to remember that there may be some slight differences in the way they use terms such as *goals*, *tasks*, and *actions*. In general, goals are achieved by carrying out tasks. These tasks are usually performed as a series of actions on some object.

In the rest of the chapter we first provide some examples of how task analysis can be used before describing four approaches to TA. The different approaches should be seen as complementary, as each has its own particular strengths and weaknesses. Although the approaches are similar in several respects, such as the need to apply them systematically, they also differ in many respects, such as the types of development activities they can support, and the level of detail that they can provide. For each of the different TA approaches we provide examples of how they can be used, and note some of their strengths and weaknesses. We finish by noting some of the limitations of TA.

Table 11.1 Areas where TA can be applied, adapted from Kirwan and Ainsworth (1992)

1. Person specification
2. Staffing and job organization
3. Skills and knowledge acquisition
4. Allocation of function
5. Performance assurance
6. Task and interface design

11.2 The Uses of Task Analysis

TA can be used for many different purposes. Kirwan and Ainsworth (1992), for example, identify six areas where TA can be applied, which are noted in Table 11.1.

It should be noted that these areas are not all independent and sometimes overlap. Here we focus our discussions on the last three items, because these are the ones that are most closely associated with the process of designing user-centered systems.

11.2.1 Allocation of Function

The traditional way of allocating functions to users has been to use a Fitts' (1951) list approach, based on whether the associated tasks are better carried out by people or by computers. In the original formulation there was often only one person, and one machine to think about. Nowadays, however, there may be several people, working together as a team, and several computer-based systems to consider. A task analysis helps you identify the knowledge that is needed to carry out the functions, and so can be used to inform how the functions are allocated. At the highest level it also lets you lay out a list of all the tasks that will need to be allocated. As part of Cognitive Work Analysis, Vicente (1999) has developed an approach to take large descriptions of tasks and use the complete set of tasks across a team of users to allocate sets of tasks to users that make sense to the users based on the large set rather than traditional job descriptions or existing interfaces.

11.2.2 Performance Assurance

The way that systems are used often changes over time. This is due partly to the way the system evolves, partly to the way people learn to use the system over time, and partly to changes in the context in which the system is deployed. The net effect is that there will often be a change in the tasks that the user carries out. It is

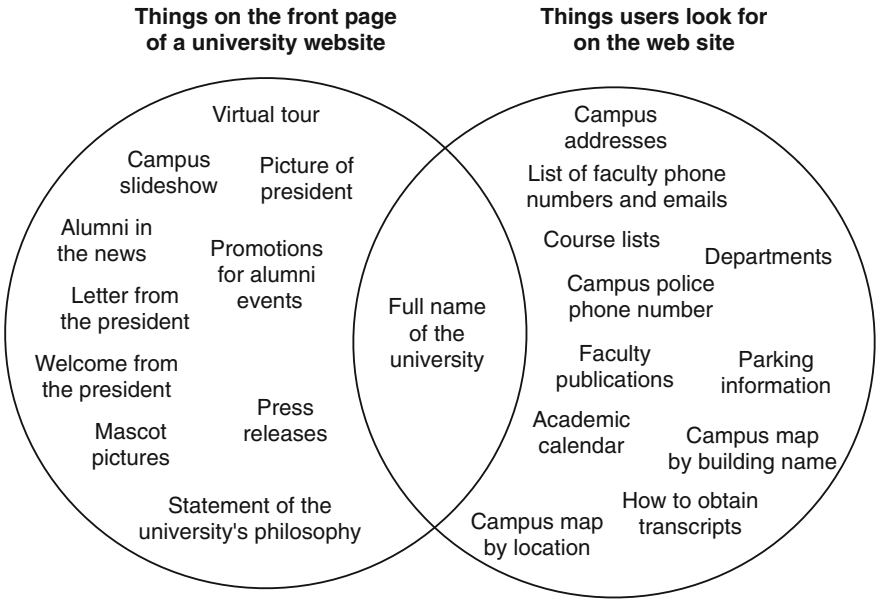

Fig. 11.1 The potential mismatch between university department web sites built without a thoughtful task analysis and what users want from university department web sites. Inspired by a cartoon from xkcd.com

therefore important to continually check that the system still supports the user in carrying out those tasks that have changed. When there are large numbers of tasks, this can be both particularly daunting and important.

For example, university department web sites have a large number of tasks or information content types that users look for. Figure 11.1 notes the mismatch that has happened on some sites. The list for a department or college consists of over 100 types of information (Ritter et al. 2002). Sites can use that list to investigate whether a site supports all the anticipated information needs of their users. Ritter and colleagues' list has been used to make specific suggestions about how these sites should be updated, including putting up more information, and distributing the updating across members of the college because it is too much for a single person to update.

Another use of TA is to find out more about how a task is done. When you compare the way that people are supposed to perform tasks, and the way they actually carry them out, there are often several noticeable differences. Creating a TA can help you to understand what the user is doing because you have a better understanding of the task, and often discussion based on a TA with users to update the TA provides further insights into why they are behaving in a particular way, for example, because of their particular knowledge or the context in which they perform the task.

11.2.3 Task and Interface Design

TA can be used in several ways to guide task and interface design as listed below.

1. To predict user task performance. These predictions of time, errors, or workload can be used to compute whether an interface can be used to perform a time-sensitive task (e.g., interact with a video game), or how many copies of an interface are needed to support a number of users (e.g., the minimum and the optimal number of ATMs or ticket machines in an airport), or when a system needs to be speeded up to keep pace with the user. This approach has been and is being used to guide the requirements of how many sailors are needed to run a ship (Chipman and Kieras 2004).
2. To suggest where users will make mistakes. Furthermore, we can take two types of task analysis and look at their relationship—for example, is the relationship between goals/subgoals and the chronology of actions simple or complex? Are similar tasks similar? Are common safe tasks easy to do and expensive or dangerous tasks harder to do? Where there is a complex relationship between the users' goal structure and the interface's action structure, then an interface may be hard to use and may lead to more errors than it should or could. This approach has been applied to tasks in aviation. Models are built of how ATC interacts with pilots (Freed and Remington 1998) or how ground control navigates planes (Byrne and Kirlik 2005). The results suggest ways to avoid errors and ways to mitigate the effects of errors.
3. To understand the relationship between old and new versions of a system. Given a task analysis of how people currently do a task, or would conceive of doing the task, we can ask how much that process and knowledge overlaps with the formal analysis of how it should be done in the new system. Increased similarity of how to use both systems can help with efficiency of learning, use, and comfort, and can lead to greater acceptance of the new system. The original work in this area (Bovair et al. 1990; Kieras and Polson 1985) noted the knowledge to use an initial editor, and then compared that to the knowledge to use a new editor. The results provide several suggestions, including: that ignoring old knowledge is easy; that modifying existing knowledge is, of course, slower; and that learning new knowledge slows the user down the most.
4. To compare different designs. A task analysis of two interfaces can predict which interface will be faster and easier to use, as long as the two designs are reasonably distinctive. In most cases, however, the comparisons are accurate and useful. Gray et al. (1992), for example, used TA to compare the interface for a new system that was to be used by telephone toll operators—the people who used to respond when you dialed 0—against the old system's interface. The new interface was predicted to be slower, and an empirical test confirmed this. It also showed that the effect of the difference in performance would cost the telephone company about $2 million per year.

   TA has been used to investigate the design of menus using real and simulated cell phones (St. Amant et al. 2004, 2007). In this project a user model was

created to perform a set of five tasks using a particular cell phone. The tasks were to adjust the ringer volume, access the tip calculator, view all contacts, view the date, and access the web browser. The resulting model was used to predict how long it would take to perform the five tasks with a cell phone and its initial menu structure. When compared with data from five practiced users the predictions turned out to be fairly accurate, but also showed where there was room for improvement in the design of the menus. The model suggested that reordering the menus to put the most often used menu items first would save about 30% of the total interaction time.

5. To create user manuals. The results of the TA show in detail the steps that are involved in carrying out a task. User manuals can be created by translating this into text and elaborating, where appropriate. Basing the manual on a task analysis helps to make the manual more accurate, and more acceptable to the users, particularly if the TA was used to analyze how users really did the tasks.

Closely related to the idea of user manuals is the notion of training people to do the task. In safety critical systems, for example, you want all the users to do the same task in (more or less) the same way. A TA can be used as the basis for developing standard operating procedures (which are used in domains other than safety critical systems too). In addition, because the TA shows the knowledge that is required to carry out particular tasks, it becomes possible to compare what the users know with what they need to know. Where any gaps are identified, this suggests where training could be useful to increase the users' knowledge to the appropriate levels.

11.3 Hierarchical Task Analysis

Hierarchical Task Analysis (HTA) is probably the most widely used method in human factors and ergonomics. An initial HTA is often a precursor to the use of other methods, largely because it provides details about task activity. The precise level of detail depends on the depth of the analysis, which is determined by the original purpose of the analysis. Like most methods of task analysis, HTA takes a decompositional approach.

11.3.1 HTA Components

HTA involves decomposing goals into subgoals (Annett 2005), although it is often described in terms of decomposing tasks into subtasks. The order and structure of these goals and subgoals is represented visually. The analysis is usually presented either as a hierarchical graph structure, or in a tabular textual format. Each goal (and subsequent subgoals) will be accompanied by a plan. These plans, which

describe how goals and subgoals are achieved, cover aspects such as the (possibly conditional) sequencing of subgoals, the need for concurrency, and how cued actions can direct performance.

11.3.2 Example Application of HTA

Table 11.2 shows a simple HTA for making an online purchase. It is not complete, but illustrates the major steps, and would be useful when talking about how to perform the task, and how to help users perform the task. It is complete enough that it could be used to start a design. The plan at the bottom of the table is an example, high-level plan. The plan—plans are usually numbered, with plan 0 being the plan associated with the top level goal—suggests that 1 must be completed before 2. There would also be plans for the subgoals, and sub-subgoals as necessary. There might also be other plans for different users and uses.

Examining the HTA in Table 11.2, we can start to identify some suggestions for design. If you wanted to provide your users with the option of browsing the listing before logging in, for example, you would have to change the plan to allow task 2 to happen before task 1, as well as afterwards, so the revised plan would look something like:

Plan 0: 1 or 2, then 2 as needed, then 1 if needed, then 3.

Figure 11.2 shows the HTA (this time drawn as a graph) for getting a file from a web-based email system. A plan for downloading the first file would include all the sub-tasks (1.1–1.4). A plan to download a second file could skip sub-tasks 1.1 and 1.2.

The analysis in Fig. 11.2 can also suggest changes to the design of the way that the task is performed. Sub-task 1.2.2, for example, could be made more general, allowing that users might move and click, and that CR might be accepted as a way to move between fields. Further, it may be possible to skip some sub-tasks in particular circumstances. If the browser remembers the user's details then sub-task 1.2 could be skipped. Similarly, if the mail client downloads the files automatically (like many mail clients), sub-task 1.4 can be skipped. The details about skipping sub-tasks would normally be described in the associated plans.

Performing an HTA can be a time-consuming process involving many choices. Although it yields a useful representational description, the technique does not specify how to go about gathering the task information that you wish to represent.

There is no hard and fast rule about the lowest level at which you should terminate the analysis. In general you should consider stopping at a natural level, where further analysis would not contribute any more to your understanding of the situation. Usually it is fairly obvious when to do this. More formally, you can consider using the so-called stopping rule. For this you need to assess the probability, P, of inadequate performance and the cost, C, of inadequate performance if

Table 11.2 An example hierarchical task analysis for making an online purchase

0 Perform online purchase
1. Login
  1.1. Select login screen
  1.2. Enter ID
  1.3. Enter password
2. Choose object
  2.1. Browse listing
  2.2. Select item
3. Pay
  3.1. Choose pay screen
  3.2. Select payment method
  3.3. Enter payment method details
Plan 0: 1, then 2 as many times as needed, then 3

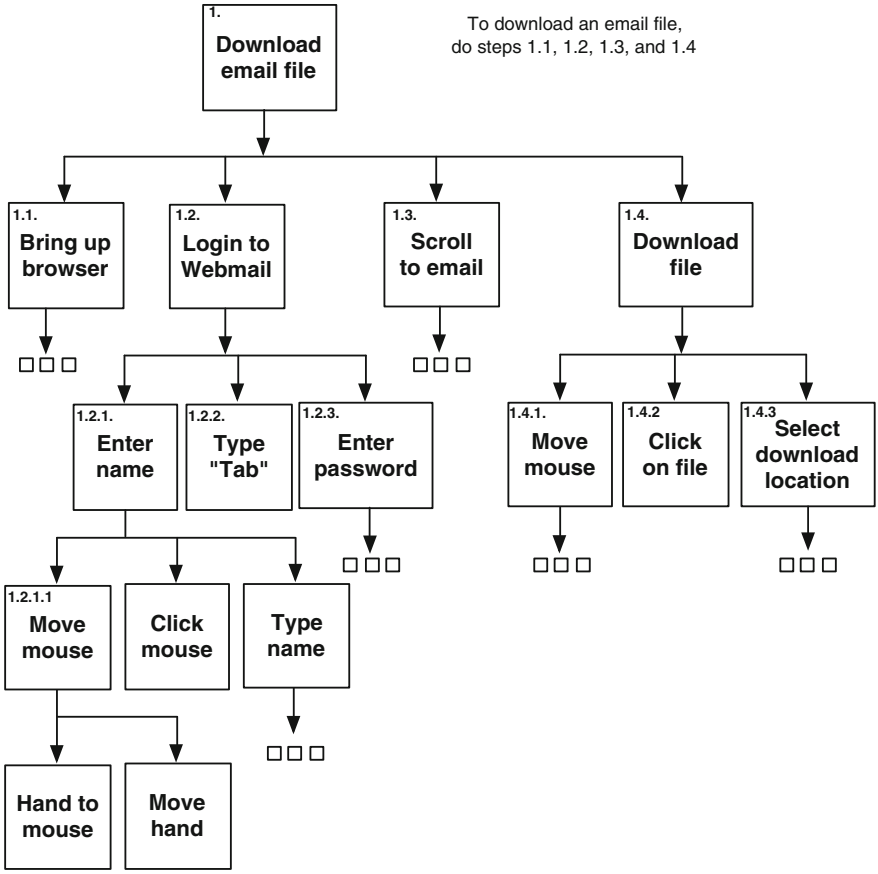

Fig. 11.2 Hierarchical task analysis of downloading an email attachment. (Further subtasks shown as three *small boxes*.)

you did not expand the hierarchy down a level. You then multiply these together to get the stopping value: if this is acceptable, then you terminate the analysis (Kirwan and Ainsworth 1992).

11.3.3 Summary

The main advantage of HTA is that, when carried out correctly by a skilled analyst, it can provide useful information to inform design relatively quickly. It does not require the interface or the interface details to be fully specified, so it can be used during the early stages of system development. Once created, the analysis can be extended, using appropriate methods to provide further details on the cognitive aspects of the task. HTA's main disadvantage is that the way in which its results are usually presented do not map readily onto the representations used by software engineers. Either the task analyst or the system developer will probably have to carry out further work to translate the results of the HTA into something that can be used during development.

11.4 Cognitive Task Analysis

As the nature of work in many domains moved from mostly manual control to mostly monitoring and supervisory control—sometimes described as a change from doing to thinking—it became increasingly important to take account of the cognitive aspects of work. Rather than pulling levers, opening and closing valves, the users now spent most of their time tracking the behaviour of a computer-based system on a display screen, making decisions and interacting with the system using the relevant input devices (e.g., in aviation many aircraft now employ a joystick device rather than a control yoke to help fly it).

Cognitive Task Analysis (CTA; e.g., see Schraagen et al. 2000 for an overview) extends more traditional task analysis techniques to facilitate the collection of information about the mental processes that underpin observable task performance. It should be noted that this does not mean that CTA and HTA are mutually exclusive: normally they should be used to complement one another.

11.4.1 CTA Components

A CTA will usually comprise the application of a set of methods (e.g., Seamster et al. 1997 discuss a range of methods that can be used in the aviation domain). One of the first steps in carrying out a CTA is therefore to choose the methods and tools that are appropriate to the particular situation that is being studied. The methods that can be used include:

1. Interviews
2. Concept (or card) sorting
3. Verbal reports
4. Cognitive walkthroughs
5. GOMS (described in more detail below).

The choice of methods is influenced by factors such as the objectives of the CTA, the available resources, the types of tasks involved, and the skills of the person doing the CTA.

Once the data for a CTA has been collected, the results from the various methods have to be compiled and interpreted. The results of the CTA should provide details about the cognitive aspects of how tasks are performed, as well as details of the information and knowledge needed to perform those tasks and how that information and knowledge are used.

11.4.2 Example Application of CTA

Baxter et al. (2005) carried out a case study of work in a neonatal intensive care unit where a new decision support system (Fuzzy Logic Respiratory Neonatal Care Expert, FLORENCE) was being developed. FLORENCE was intended to help clinical staff make decisions about changes to the mechanical ventilator that is used to treat respiratory distress syndrome (RDS) in premature babies. It was important that the introduction of FLORENCE did not adversely affect the dependability of the delivery of care in the unit, and that it was acceptable to users. A CTA was therefore carried out using lightweight rich pictures analysis (Monk 1998), the critical decision method (CDM; Klein et al. 1989), and observation.

Task analysis methods, in general, pay little attention to the context in which the tasks are carried out; rich pictures analysis provides a way to take account of that context, although it is quite novel for it to be included as part of a CTA. In this case study, a rich pictures analysis was carried out to capture details about the context in which FLORENCE was to be deployed. Data was collected using semi-structured interviews to identify the roles, responsibilities, and concerns of eight members of staff working in the unit, including the staff that provide administrative support in the unit. The analysis also showed how staff communicate and interact with members of staff who work in other parts of the hospital, such as the test laboratories. The importance of communication within the unit was also identified, since it forms the basis for clinical care of the babies. Furthermore, the central role of the various patient records was highlighted: FLORENCE would have to find some way to produce appropriate records.

The CDM was used to analyze how clinical staff make decisions about any changes that need to be made to the settings of the mechanical ventilators used to treat the babies. It is critical that these decisions are correct and timely, otherwise they can seriously adversely affect the babies' health. The basic steps of the CDM

were followed for the incidents selected by each of the eight participants, and a critical cue inventory—highlighting the cues that clinical staff paid attention to—and situation assessment records—which describe how the incident unfolded in terms of cues, goals, expectations, and decisions—were developed for each incident. These highlighted aspects such as the importance of having a distinctive alarm for FLORENCE because the unit already has several alarms sounding on a fairly regular basis, and the hierarchy of decision making which flattens when dealing with acute incidents.

Finally, observation was used to capture details of how clinical staff work in situ to treat babies with RDS. RDS is self-regulating, and usually lasts around 72 h. The treatment of two babies was recorded using timed note taking—the unit is a high stress situation, so video recording was not a viable option—over a 2-h period. The first baby was moved off the unit after 1 day; the second baby was observed on three consecutive days all at the same time of day. The results showed how often staff interacted with the equipment around the babies' cots under normal conditions and when an alarm sounded.

The results of the CTA were analyzed and used to inform the development of FLORENCE. These included decisions about the interface (e.g., the need for instructions about changes to ventilator settings to be precise and unambiguous, such as "increase PIP by 2–16"), as well as training (e.g., staff need to learn how to prioritize their response to FLORENCE's alarm) and more general physical ergonomics concerns (e.g., space is needed around the baby's cot to accommodate the PC running FLORENCE).

11.4.3 Summary

As its name suggests, CTA is particularly suited to analyzing the cognitive processes and products that people use when performing tasks. It is not so good at dealing with the work context in which performance takes place (although using rich pictures analysis does provide at least a partial solution, e.g., see Baxter et al. 2005). Carrying out a CTA can be quite time consuming, particularly when the selected methods require access to experts. CTA also requires that the analyst(s) are competent in carrying out a range of methods. Furthermore, it requires a high level of skill to be able to interpret the results of several methods in a holistic way so that they can be presented to the system developers.

11.5 GOMS

GOMS is an acronym for Goals, Operations, Methods, and Selection rules (Card et al. 1980, 1983). It is a method that can be used as part of a CTA, but we include it in its own right because of its close relationship with the field of HCI.

The focus of GOMS is on specifying the details of error-free, expert behavior, and using the resulting specification to predict learnability, usability, and task execution times, allowing that there may be multiple strategies available for similar tasks. GOMS tends to be used when designing interactive technology like phones and GPS systems, where milliseconds can matter, where consistency in the users' knowledge and in the interface matters, and where operators may be expert but not trained to do things in a single way.

The four GOMS techniques:

- GOMS: sometimes referred to as CMN-GOMS, using its authors' initials to distinguish it from the generic name
- NGOMSL: Natural GOMS Language
- CPM-GOMS: Cognitive Perceptual Motor GOMS, which supports concurrent actions
- The Keystroke Level Model (KLM) which is described in more detail below

are sometimes described as a family (John and Kieras 1996a). They vary in how easy they are to apply and use.

11.5.1 GOMS Components

The components of GOMS are listed in Table 11.3. With GOMS models, the cognitive structure is represented in the concept of starting from Goals, and also from the explicit use of Selection rules to indicate how Methods—comprising subgoals and Operators—are chosen to accomplish those Goals.

11.5.2 Example Application of GOMS

Table 11.4 provides an example GOMS model for editing text. This model consists of two parts. The main task (which comes second in the table) is to perform all the edits. This is a cyclical task, and represents starting with a marked up manuscript and then performing the edits marked on the paper (or highlighted in the electronic file). Each of the edits are *unit tasks* which define a single edit. A full model would have multiple types of unit tasks for editing. The first part of the model notes a declarative memory chunk about the Cut command, a unit task, including how to find it and what the keystroke accelerator is. A more complete model would have further unit tasks defined.

The method for editing a document is shown as a looping algorithm of working through a set of tasks to edit a document. This algorithm represents how people work through editing a document with changes to be made in the document, stopping when there are no more edits. This remains a relatively common task, for

Table 11.3 The components of GOMS

Goals are the desired states of affairs. They are brought about by the application of methods and operators
Operators are elementary perceptual, motor, or cognitive actions. Intended either to change the world (a normal key-press) or to change our knowledge about the world (e.g., reading). In practice, operators are really those subgoals whose methods of solution we have decided not to analyze any further. The choice of appropriate operators is critical to a GOMS analysis. Some critics note that there are no clear guidelines for doing so, although it is often a fairly straightforward decision
Methods describe procedures for achieving goals. They contain a sequence of subgoals and/or operators, with conditions potentially attached to each part of the method. These conditions relate to the current task environment (e.g., repeat until no tasks left)
Selection rules augment the basic control structure of the model. Where multiple methods are available the selection rules indicate how to choose between the methods. These include, for example, when scrolling a document: if you want to scroll a long distance, use the search function; if you want to scroll a short distance, use the scroll bar; and if you want to scroll a very short distance, use the arrow keys

Table 11.4 Example GOMS model for text editing (reprinted from St. Amant et al. 2005). This model is written using NGOMSL notation

```
LTM [long term memory] item: Cut Command [a unit task]
        Name is Cut.
        Containing Menu is Edit.
        Menu Item Label is Cut.
        Accelerator Key is “Control-X”.
[[There would be more types of commands here in a complete model.]]

Method for goal: Edit Document
        Step. Store First under <current task name>.
        Step. Check for done.
        Decide: If <current task name> is None, Then
                Delete <current task>;
                Delete <current task name>;
                Return with goal accomplished.
        Step. Get task item whose
                Name is <current task name>
                and store under <current task>.
        Step. Accomplish goal: Perform Unit task.
        Step. Store Next of <current task>
                under <current task name>;
                Goto Check for done.
```

example, how an author right now is working through this chapter making changes to improve the text and how students revise lab reports.

The loop has several types of substeps. These steps can store state (e.g., what the current task is); they can perform conditionals (if); and they can subgoal (Perform unit task). Within the unit tasks (not fully shown), information can be obtained from the display and mental and physical actions performed. GOMS manuals provide a full list of the actions (e.g., Kieras 1999) and when to include mental operators to retrieve task structure, but in practice these lists can be extended to include unusual devices such as joysticks or different types of manual controls (such as different knobs or toggle switches). The analysis in Table 11.4 does not include all the details of all the subtasks that can be performed, and not all the details of how to do a unit task.

Table 11.5 shows a model for dialing a cell phone. In this example the tasks are slightly smaller and the interaction and memory aspects are more prominent.

11.5.3 Summary

GOMS models sometimes appear structurally quite similar to other hierarchical task analyses of goals/subgoals. The main difference is that GOMS has formalized its components to a greater level of detail. It is assumed that the methods are known before the GOMS analysis is carried out, and that they are not developed during task performance. The particular strengths of GOMS lie in assessing how to make interfaces easier to use by comparing alternative models, or methods, for a particular task.

Many people question the utility of GOMS analyses, arguing that there is no such thing as error-free, expert performance. This limitation can be somewhat ameliorated by incorporating the error correction strategies in the analysis. Probably the biggest limitation of GOMS, however, is that the interface has to be implemented or at least described in enough detail to be able to identify the actions for a particular task. There is also a lack of a clear specification of what can be used in Selection rules—there is an implication that it should be the current task context, but real behavior, for example, undoubtedly allows choices to be based on previous selections. It is also difficult to do a vanilla GOMS analysis when multitasking or task interleaving is involved. CPM-GOMS was developed to address this, but at the expense of making it more difficult to create models.

11.6 The Keystroke Level Model

A simplified version of GOMS is the keystroke level model (KLM) of Card et al. (1983). Like GOMS, it is a method that can be used as part of a CTA, but we include it here in its own right because of its close relationship with the field of HCI.

Table 11.5 Example GOMS model, written using NGOMSL notation, for telephone dialing taken from St. Amant et al. (2005)

```
Task item: T1          [[ tasks will go from T1 to T11 to dial (814) 865-3528 ]]
        Name is T1.
        Digit is "8".
        Next is T2.
Visual object: first digit
        Content is "8".
Task item: T2
        Name is T2.
        Digit is "1".
        Next is T3.
Visual object: second digit
        Content is "8".
. . .
Task item: T11
        Name is T11.
        Digit is "8".
        Next is NONE.

Visual object: second digit
        Content is "1".

Method for goal: Dial Number
        Step. Store T1 under <current task name>.
        Step Check for done.
                Decide: If <current task name> is T11, Then
                Delete <current task>;
                Delete <current task name>;
                Return with goal accomplished.
        Step. Get task item whose Name is <current task name>
                and store under <current task>.
        Step. Accomplish goal: Dial Digit.
        Step. Store Next of <current task> under <current task name>.
        Goto Check for done.

Method for goal: Dial Digit
        Step. Look for object whose Content is Digit of <current task>
                and store under <target>.
        Step. Point to <target>; Delete <target>.
        Step. Click mouse button.
        Step. Verify "Correct digit pressed".
        Step. Return with goal accomplished.
```

The KLM is usually applied where there is just one person interacting with a computer-based system. It is a fast and approximate way to compute how long users will take to perform a unit task. A unit task is essentially a small cognitively manageable task. Large tasks can be decomposed into several unit tasks, although

not all tasks will have a unit task substructure. The time to do the whole task is calculated by simply adding up the calculated times for all the component unit tasks. It should be noted that you also need to include the time it takes to acquire the task, i.e., the time it takes for the user to work out what to do. The KLM does not provide a way of calculating this but Card et al. (1980) suggest that the time needed for each unit task when manipulating a manuscript is 2–3 s if the instructions are written down, 5–30 s if it is a routine design situation, and even longer for creative composition situations.

To calculate the time for a unit action involves analyzing the operations into their elemental perceptual, motor, and cognitive actions (e.g., keystrokes). Then by adding together the times for these individual actions it can be possible to make time predictions for expert, error-free performance of that task.

11.6.1 Description of KLM Components

The execution of a unit-task requires operators of (basically) four types:

1. Keystrokes (K): unit time based on typing speeds (0.08–1.2 s/keystroke, mouse click, or button press)
2. Pointing (P): moving mouse to target (clicking is a keystroke) (approximately 1.1 s, but Fitts' law can also be used)
3. Homing (H(mouse) or H(keyboard)): moving hand to/from mouse and keyboard (approximately 0.4 s)
4. Drawing (D): dragging mouse in straight-line segments (0.9 n + 0.16 l where n = number of segments and l = length of segments).

To these should be added some number of mental operators (M or Mop, 1.35 s). Well practiced tasks with a simpler structure will require less mental operators. Getting the number of mental operators correct is a difficulty in using this method, but if it is done consistently across applications it is less of a problem. Finally, the system's response time (Sys), if it limits the user's task performance, also has to be included. This can be estimated, or, again, if the estimate can be assumed to be consistent across interfaces, is less important than you might fear.

The number of mental operators is computed using a set of rules—basically between all operators, except those linked through knowledge or skill (e.g., the keystrokes in a single word, or a set such as point and click a mouse).

Where there are selection rules governing the choice of methods then it is up to the analyst to decide whether to go for best or worst case time predictions.

The time to perform these operators as noted above can be done in an approximate style. There are several ways to improve the predictions of KLM models.

(1) Keystroke times vary. Card et al. (1983) include tables that provide different times for different keystrokes. These tables do not differentiate different typist speeds, but do show that different keys take different times. This suggests that

Table 11.6 Example application of the KLM to a simple editing task

Method
1. Delete 3rd clause. H[mouse] K[mouse down] P K[mouse up] M K[D]
2. Insert it in front of 1st clause. P M K[l] K[ESC]
3. Replace ": o" with "O". P M 2 K[SHIFT R] H[keyboard] 2 K[O ESC]
4. Replace "T" by ": t". H[mouse] K[mouse down] M K[R] H[keyboard] 4 K[: SPACE t ESC]
5. Delete 3rd clause. H[mouse] P K[mouse] M K[D]
6. Insert it in front of 2nd clause. K M K[l] K[ESC]
7. Find next task. M K[F]
Time Predictions
T*execute* = [23 t_K + 4 t_P + 5 t_H] + 7 t_M
= 22 (0.15) + 4 (1.1) + 5 (0.4)] + 7(1.35)
= 19.15 s

Note that in this example the same average time is used for each keystroke

interfaces should choose keys that are fast (and presumably easy to type). This suggestion includes choosing words that use both hands for touch typists and avoiding punctuation characters (which are slower to press).

(2) The gross predictions of mouse moves and drags can be made more accurate by using Fitts' law to make predictions based on the target size and distance. When the initial location of the mouse is not known, it can be approximated.

(3) The system response time can be computed using tools to analyze key press times. The response times are still important in devices such as ATMs, and in systems which may be required to present the same data simultaneously on different continents, within a multinational company, for example.

11.6.2 Example Application of the KLM

Table 11.6 shows an example KLM analysis worked out for a simple text editing task of cutting a phrase (Step 1), inserting it (Step 2), changing the case of two letters (Steps 3 and 4), and taking up the next task (Step 5). The analysis suggests that the time to perform this task is not so long (24 s). Thus, if this edit takes longer, it is not the interface that is restraining the user's progress, it is the cognition to come up with the letters or to think about other changes. It is the writing, essentially, to come up with the changes that takes so long!

11.6.3 Summary

Although closely related to GOMS, note how keystroke-level analysis is closer to time-and-motion (chronological) analysis than goal/subgoal analysis. It assumes a simple world: concentration on one task at a time, a fully specified interface, no

interleaving of goals, no interruptions, a single method, error-free, expert performance, and so on. Indeed, KLMs can be developed without worrying about goals and selection rules. These limitations make the KLM easier to use, but they also highlight the types of tasks that cannot easily be modeled using the KLM and where it will not give accurate results.

Quite a considerable effort has gone into trying to make KLMs more usable—particularly by building computer tools to apply them (Beard et al. 1996; Nichols and Ritter 1995; Williams 2000) and to extend them to new types of interfaces such as mobile GPS systems (Pettitt et al. 2007). These tools provide a language for defining KLM and GOMS models and computing the task time based on the analyses, although sometimes it may be just as easy to use a spreadsheet.

11.7 Considerations When Choosing a TA Method

There are some general issues that you need to be aware of when deciding whether to carry out a TA to inform system design, as they will influence how successful the TA is. Superficially it appears quite an easy technique to use, but to carry out a TA that produces useful results that can inform system design is an acquired skill. It requires skills in collecting the appropriate data or task specifications, analyzing that material, and then interpreting it.

It may seem obvious, but to perform a TA, you need to have access to a detailed enough task description. This can either be written (documentation) or captured through interviews and observation. If you are using interviews and observation, you need to make sure that you can get access to the appropriate people in a timely manner. For new or unimplemented interfaces you can often base your analyses on existing interfaces and extrapolations. If you are analyzing an unbuilt interface you should note your assumptions very clearly.

Different types of TA place emphasis on different aspects of task performance. You need to be aware of this. If you are interested in the work context, for example, then HTA will help; if you are more interested in the cognitive aspects of the task, you will be better off using some form of CTA. The different methods are not mutually exclusive.

In general, TA cannot easily simultaneously represent users with different levels of knowledge, skills, and abilities. Novice users and the way a task is learned, for example, are not well represented by most of these techniques. TA techniques, when applied sympathetically, can help support novice users, but this has to be done by the analyst. Most methods can deal with some aspects of task learning, such as how much knowledge has to be learned in order to do the task. To model the time course of learning you may need to develop a more detailed cognitive model (e.g., see Paik et al. 2010; Ritter and Bibby 2008).

Most methods of TA are strongly goal-oriented. These methods cannot generally be applied to situations that have little structure or only very abstract goals, such as play, conversation, and team building. All TA should be done with an

understanding of the general context in relation to work and activities. Methods like activity theory address these issues (e.g., see Bertelsen and Bødker 2003 for an introductory review). Activity theory is a very high level method that analyses the types of user activities, generally using a very informal representation. It uses these descriptions to suggest how to design the context in which the activities take place. For example, identifying children's activities (play activities, painting, block stacking, and doll-house play, for example), would inform the design of a kindergarten room. Similar analyses make strong suggestions about the design of research labs (collaboration spaces, mailing rooms, teaching rooms) and interfaces where goals are less important than activities such as drawing, designing, interacting, team-building, exploring, or enjoying.

Most TA methods do not note and are not sensitive to context, including physical context such as lighting and heating, social norms such as doing what helps others in the environment, and not using others space or resources. It is especially common in work places to discover that the physical or social context produces changes in the way people do tasks from how they would ideally do them, or from the way they are supposed to do them. Casey (1998, 2006) provides numerous examples where not understanding the context on task analyses and on design led to problems, such as where other people are when the user is doing a task (not in the path of the rocket!), or the effect of weather on how to do tasks and which tasks to do on a ship.

11.8 Summary

Task analysis in all its various guises is a very useful technique, and usually fairly easy to perform when designing systems, which may help explain why it is so widely used. The results of a task analysis can be used to help designers create easier to use interfaces by highlighting the structure of tasks, and the time it takes to perform tasks, as well as making it easier to compare alternative methods for carrying out those tasks. Task analysis can also be used to represent the trade-offs in designs, helping designers to make informed decisions. TA can also be used directly to improve interfaces, and Table 11.7 notes a few common ways in which TA can help improve interfaces.

There are many different uses for task analysis, and the method that you choose will depend to some extent on how you intend to use the results. Choosing any one is better than none, that you should choose one that is easy for you to use because you are a user as well. The choice will also be affected by the context in which you will use the method, the limitations, and possibly the experience of the analyst(s) (several of the methods work at similar levels of abstraction, so they can be used somewhat interchangeably).

Using task analyses to guide system design has a good return on investment (RoI), ranging from 7 to 20 to 100 in one survey (Booher and Minninger 2003) to 1,000 in another more focused HCI study (Nielsen and Phillips 1993). Given the

Table 11.7 Ways to use TA to improve interfaces

1. Modify the interface to support all the tasks in the TA
2. Modify the interface to not support tasks not required by the task analysis
3. Modify the interface to do steps for the user where choices are not required
4. Modify the interface to use fewer actions
5. Modify the interface to use a simpler task structure
6. Modify the interface to provide more regular knowledge/interactions to perform related tasks
7. Modify the interface to not require holding state variables in the task analysis or user's head
8. Modify the interface to make common tasks faster
9. Modify the interface to make expensive tasks slower/harder to initiate
10. Teach any alternative methods through and in the interface

multiple ways that task analyses can be used and the savings they suggest, these numbers are quite plausible.

The various forms of task analysis can be ordered by how much detail they include. A preliminary classification can be into declarative and procedural representations. The declarative representations merely denote the actions or tasks the users will perform. The simplest of these are HTA and CTA. The KLM is a slightly more complex example of this because it attributes time to each action.

Procedural task descriptions, such as advanced versions of GOMS (John and Kieras 1996b) and models in cognitive architectures (noted in the final chapter), can include enough procedural knowledge of how to perform the task such that they can also perform the task in some way, perhaps with a simplified version of the interface. Declarative representations of behavior, like the KLM, for example, will note that the result of a mathematical task is "a number," whereas a fully procedural task analysis will be able to compute the actual number. This chapter discusses several examples of each type. There is a wide range of approaches, and descriptions of them are available in surveys (Adams et al. 2012; Beevis 1999; Schraagen et al. 2000), as well as in manuals and tutorials on these other TA approaches.

As you use TA in a wider range of situations, you should become more skilled in applying it. For specific situations you may want to look at some of the more specialized forms of task analysis. For example, if you have many novice users, you might create a situation where the user has to read the documentation to find the commands, which means that your design should ensure that the knowledge of how to use the interface is findable or transmitted to the novice through instructions, conventions, and other design elements. When you need to do more complex analyses, for example, where error recovery or multi-tasking is important, you may need to consult further material to find the most appropriate technique, as well as investigate supplementary techniques such as knowledge elicitation (e.g., Ericsson and Simon 1993; Kirwan and Ainsworth 1992; Schraagen et al. 2000; Shadbolt 2005; Shadbolt and Burton 1995).

The results of a task analysis on their own will not tell you whether a particular system or interface is acceptable or not. It just shows how well the system or interface will work for particular tasks for particular people (in a particular context). Decisions about modifications to the system or interface will often require consideration of other factors, such as the resources that are available for further development, other systems or interfaces (to cater for consistency and interoperability, for example), and assumptions about the target users. Task analysis cannot suggest whether novices should be served preferentially by the system, or whether experts should be supported in an interface design, but TA can help compute the costs and represent the trade-offs involved in making such decisions.

11.9 Other Resources

The Handbook of task analysis for human-computer interaction, edited by Dan Diaper and Neville Stanton (2003), although 10 years old, still provides a fairly comprehensive overview of TA. It makes a useful reference book, rather than being something that you should read from cover to cover. A more general, but still useful overview of the many types of TA and their application is Kirwan and Ainsworth's *A Guide to Task Analysis* (1992) even though it is now over 20 years old.

Clayton Lewis and John Rieman's shareware book, *Task-Centered User Interface Design* (1994), www.hcibib.org/tcuid/ provides a useful expansion of the material in this chapter, particularly on the role of task analysis in system design.

Shepherd's book, *Hierarchical Task Analysis* (2004), provides a detailed discussion of HTA. It illustrates how it can be applied to a wide range of tasks, and how the results can generate a wide range of benefits.

For a practical guide to using CTA it is worth looking at *Working Minds: A practitioner's guide to Cognitive Task Analysis* by Crandall et al. (2006). For details about the sorts of methods that can be used in a CTA, it is worth consulting *Applied Cognitive Task Analysis in aviation* by Thomas Seamster et al. (1997), and *Human factors methods* by Neville Stanton et al. (2005).

David Kieras has papers, manuals, and tools covering the KLM, GOMS, and task analyses available via his web site: www.ai.eecs.umich.edu/people/kieras/kieras.html. You can also find several KLM calculators online. With these you simply enter the KLM operators that are used to do the task, and the total time is automatically calculated for you.

Kim Vicente's book *Cognitive Work Analysis* (1999) describes a broader, integrated framework which includes models of the work domain, control tasks, strategies social-organizational factors, and worker competencies. This framework builds on the body of work called cognitive (systems) engineering which focuses on the analysis, design, and evaluation of complex socio-technical systems.

The book's web site has additional information on task analysis including some tools.

11.10 Exercises

11.1 Create a table showing each type of task analysis. Note (a) the focus of the approach (work, the task, what the user knows, etc.); (b) what they are typically used for, for example, what predictions they can provide; (c) an example snippet of analysis; (d) how time intensive you believe they will be (easy to do, hard to do); and (e) the advantages and disadvantages of each type.

11.2 Choose an existing interface and list the tasks it can or should be used for. Perform a task analysis. From this task analysis make five suggestions how it could be improved. Interfaces to consider include word processors, spreadsheets, online libraries, and email clients.

11.3 Write a short report on tasks you or a study partner do at your desk, and the quality of the desk and work area to support those tasks. Compare it with the non-work related tasks that you do indulge in (e.g., play with a pencil, line up your toy cars, doodle, drink coffee) at your desk. Make suggestions for improving the workspace in question. Then, do the same thing for your computer display.

11.4 List ten ways that the KLM or GOMS methods could be made more accurate. You should be able to do this using this book. For example, the section on reading makes several suggestions for making these models more accurate with respect to adjusting for users' reading speed and the material being read.

11.5 Consider a smartphone or tablet computer, either a specific one or a composite one. Devise a trade-off function between a set of features (which you specify) and weight. Describe how to choose a reasonable point on that curve. Use a task analysis that provides estimates of task time to represent this design trade-off.

11.6 Use a task analysis methodology to examine two related online systems, such as bookstores. You should choose three to five important tasks, and explain why these are important tasks for this analysis. How do their assumptions differ as represented in the task analysis? For example, does one assume that you log in before browsing, or does one assume that you read recommendations before purchasing?

11.7 Use a task analysis methodology that you choose to calculate how long it takes to perform the following tasks using a spreadsheet program, using a calculator, and using your cell phone: (a) a tip on a $13.28 restaurant bill; (b) your salary raise as a percentage from $8.50 to $8.85/h (c) your hourly rate if you work 40 h per week and 48 weeks per year and make $45,000 per year, and (d) your hourly rate from the previous problem, but starting from scratch, if you had a 3% raise. Comment on at what point in time you should switch between these three tools.

11.8 Use a task analysis methodology to calculate how long it takes to select and to delete an automatic signature to your email, using your signature rather than the one in Sect. 11.8. Is it faster to insert the signature manually when you need it, or to included it automatically and delete it when you do not need it?

Draw a curve representing the time to use both strategies as the frequency of need to sign an email changes from 0 to 100% in 10% increments.

11.9 In 2003 the University of Denver (noted in the Chronicle of Higher Education, 19 December, p. A35) decided to require all applicants to be interviewed by college officials. The question on the interviewers' minds, according to the assistant vice chancellor for enrollment management, is "Will the kid graduate?" The retention rate has gone from 82 to 86% because (or despite) the college starting using the system 2 years ago.

(a) Based on your experience as a college student, come up with a task analysis of what you have to do to be (i) successful in college and (ii) graduate. Include multiple strategies and allow tasks you need to terminate or avoid. (This integrates results from several chapters.) Do keep in mind that GPA is an important but surely not the only measure of success in college, and you might note other successes besides graduation.

(b) Discuss how you would interview someone to find out whether they can do these tasks.

(c) Can you provide any alternative hypotheses about why retention might have gone up besides the interview process?

References

Adams, A. E., Rogers, W. A., & Fisk, A. D. (2012). Choosing the right task analysis tool. *Ergonomics in Design: The Quarterly of Human Factors Applications, 20*(4), 4–10.

Annett, J. (2005). Hierarchical task analysis (HTA). In N. Stanton, A. Hedge, K. Brookhuis, E. Salas & H. Hendrick (Eds.), *Handbook of human factors and ergonomics methods* (pp. 33-31–33-37). Boca Raton, FL: CRC Press.

Baxter, G. D., Monk, A. F., Tan, K., Dear, P. R. F., & Newell, S. J. (2005). Using cognitive task analysis to facilitate the integration of decision support systems into the neonatal intensive care unit. *Artificial Intelligence in Medicine, 35*, 243–257.

Beard, D. V., Smith, D. K., & Denelsbeck, K. M. (1996). Quick and dirty GOMS: A case study of computed tomography interpretation. *Human-Computer Interaction*, *11*, 157–180.

Beevis, D. (Ed.). (1999). *Analysis techniques for human-machine systems design: A report produced under the auspices of NATO Defence Research Group Panel 8*. Wright-Patterson Air Force Base, OH: Crew Systems Ergonomics/Human Systems Technology Information Analysis Center.

Bertelsen, O. W., & Bødker, S. (2003). Activity theory. In J. M. Carroll (Ed.), *HCI models, theories and frameworks: Toward a multi-disciplinary science*. San Francisco, CA: Morgan Kaufmann.

Booher, H. R., & Minninger, J. (2003). Human systems integration in Army systems acquisition. In H. R. Booher (Ed.), *Handbook of human systems integration* (pp. 663–698). Hoboken, NJ: Wiley.

Bovair, S., Kieras, D. E., & Polson, P. G. (1990). The acquisition and performance of text-editing skill: A cognitive complexity analysis. *Human-Computer Interaction, 5*, 1–48.

Byrne, M. D., & Kirlik, A. (2005). Using computational cognitive modeling to diagnose possible sources of aviation error. *International Journal of Aviation Psychology, 15*(2), 135–155.

Card, S. K., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the ACM, 23*(7), 396–410.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.

Casey, S. M. (1998). *Set phasers on stun: And other true tales of design, technology, and human error*. Santa Barbara, CA: Aegean.

Casey, S. M. (2006). *The Atomic Chef: And other true tales of design, technology, and human error*. Santa Barbara, CA: Aegean.

Chipman, S. F., & Kieras, D. E. (2004). Operator centered design of ship systems. In *Engineering the Total Ship Symposium*. NIST, Gaithersburg, MD. American Society of Naval Engineers. Retrieved March 10, 2014, from http://handle.dtic.mil/100.2/ADA422107

Crandall, B., Klein, G., & Hoffman, R. R. (2006). *Working minds: A practitioner's guide to cognitive task analysis*. Cambridge, MA: MIT Press.

Diaper, D. (2004). Understanding task analysis. In D. Diaper & N. Stanton (Eds.), *The handbook of task analysis for human-computer interaction* (pp. 5–47). Mahwah, NJ: LEA.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data* (2nd ed.). Cambridge, MA: MIT Press.

Fitts, P. M. (1951). *Human engineering for an effective air navigation and traffic control system*. Washington, DC: National Research Council.

Freed, M., & Remington, R. (1998). A conceptual framework for predicting error in complex human-machine environments. In *Proceedings of the 20th Annual Conference of the Cognitive Science Society* (pp. 356–361). Mahwah, NJ: Erlbaum.

Gray, W. D., John, B. E., & Atwood, M. E. (1992). The precis of project ernestine or an overview of a validation of GOMS. In *Proceedings of the CHI'92 Conference on Human Factors in Computer Systems*. New York, NY: ACM Press.

John, B. E., & Kieras, D. E. (1996a). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction, 3*(4), 320–351.

John, B. E., & Kieras, D. E. (1996b). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction, 3*(4), 287–319.

Kieras, D. E. (1999). A guide to GOMS model usability evaluation using GOMSL and GLEAN3: AI Lab, University of Michigan. Available from www.ftp.eecs.umich.edu/people/kieras

Kieras, D. E., & Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies, 22*, 365–394.

Kirwan, B., & Ainsworth, L. K. (1992). *A guide to task analysis*. London, UK: Taylor & Francis.

Klein, G., Calderwood, R., & MacGregor, D. (1989). Critical decision method for eliciting knowledge. *IEEE Transactions on Systems, Man, and Cybernetics, 19*, 462–472.

Monk, A. F. (1998). Lightweight techniques to encourage innovative user interface design. In L. Wood (Ed.), *User interface design: Bridging the gap between user requirements and design* (pp. 109–129). Boca Raton, FL: CRC Press.

Nichols, S., & Ritter, F. E. (1995). A theoretically motivated tool for automatically generating command aliases. In *Proceedings of the CHI'95 Conference on Human Factors in Computer Systems* (pp. 393–400). New York, NY: ACM.

Nielsen, J., & Phillips, V. L. (1993). Estimating the relative usability of two interfaces: Heuristic, formal, and empirical methods compared. In *Proceedings of InterCHI '93* (pp. 214–221). New York, NY: ACM.

Paik, J., Kim, J. W., Ritter, F. E., Morgan, J. H., Haynes, S. R., & Cohen, M. A. (2010). Building large learning models with Herbal. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of ICCM: 2010- Tenth International Conference on Cognitive Modeling* (pp. 187–191).

Pettitt, M., Burnett, G., & Stevens, A. (2007). An extended keystroke level model (KLM) for predicting the visual demand of in-vehicle information systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1515–1524). ACM.

Ritter, F. E., & Bibby, P. A. (2008). Modeling how, when, and what learning happens in a diagrammatic reasoning task. *Cognitive Science, 32*, 862–892.

Ritter, F. E., Freed, A. R., & Haskett, O. L. (2002). *Discovering user information needs: The case of university department websites* (Tech. Report No. 2002-3): Applied Cognitive Science Lab, School of Information Sciences and Technology, Penn State. www.acs.ist.psu.edu/acs-lab/reports/ritterFH02.pdf

Schraagen, J. M., Chipman, S. F., & Shalin, V. L. (Eds.). (2000). *Cognitive task analysis*. Mahwah, NJ: Erlbaum.

Seamster, T. L., Redding, R. E., & Kaempf, G. L. (1997). *Applied cognitive task analysis in aviation*. Aldershot, UK: Avebury Aviation.

Shadbolt, N. R. (2005). Eliciting expertise. In J. R. Wilson & E. Corlett (Eds.), *Evaluation of human work* (3rd Edition, pp. 185–218). London: Taylor and Francis.

Shadbolt, N. R., & Burton, A. M. (1995). Knowledge elicitation: A systematic approach. In J. R. Wilson & E. N. Corlett (Eds.), *Evaluation of human work: A practical ergonomics methodology* (pp. 406–440). London: Taylor and Francis.

St. Amant, R., Freed, A. R., & Ritter, F. E. (2005). Specifying ACT-R models of user interaction with a GOMS language. *Cognitive Systems Research*, *6*(1), 71–88.

St. Amant, R., Horton, T. E., & Ritter, F. E. (2004). Model-based evaluation of cell phone menu interaction. In *Proceedings of the CHI'04 Conference on Human Factors in Computer Systems* (pp. 343–350). New York, NY: ACM.

St. Amant, R., Horton, T. E., & Ritter, F. E. (2007). Model-based evaluation of expert cell phone menu interaction. *ACM Transactions on Computer-Human Interaction, 14*(1), 24.

Vicente, K. (1999). *Cognitive work analysis*. Mahwah, NJ: Erlbaum.