HIERARCHICAL TASK ANALYSIS (HTA)

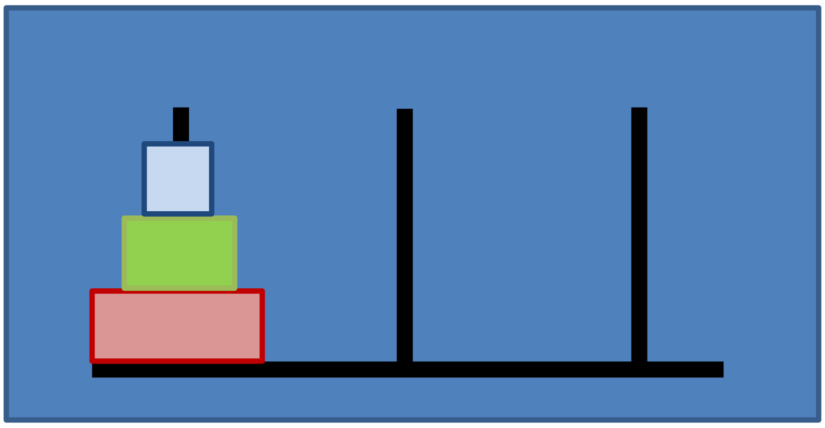
Introduction

There are dozens of methods to support 'task analysis'. Some of these are detailed in HCI textbooks. I contend that there is, in HCI, a mistaken view that task analysis methods come from time-and-motion studies, and that the aim is to decompose work activity into 'one-best-way' of working. Task analysis has been dismissed as be overly reductionistic (decomposing human behaviour into a purely physical tasks) and overly prescriptive (imposing a single way of working). Both of these views are wrong. In this handout, I show how task analysis can be used to explore strategies that people for interacting with technology, and to show how this can help analysts reflect on the cognitive activity performed when people choose and use these strategies.

Test-Operate-Test-Exit (TOTE)

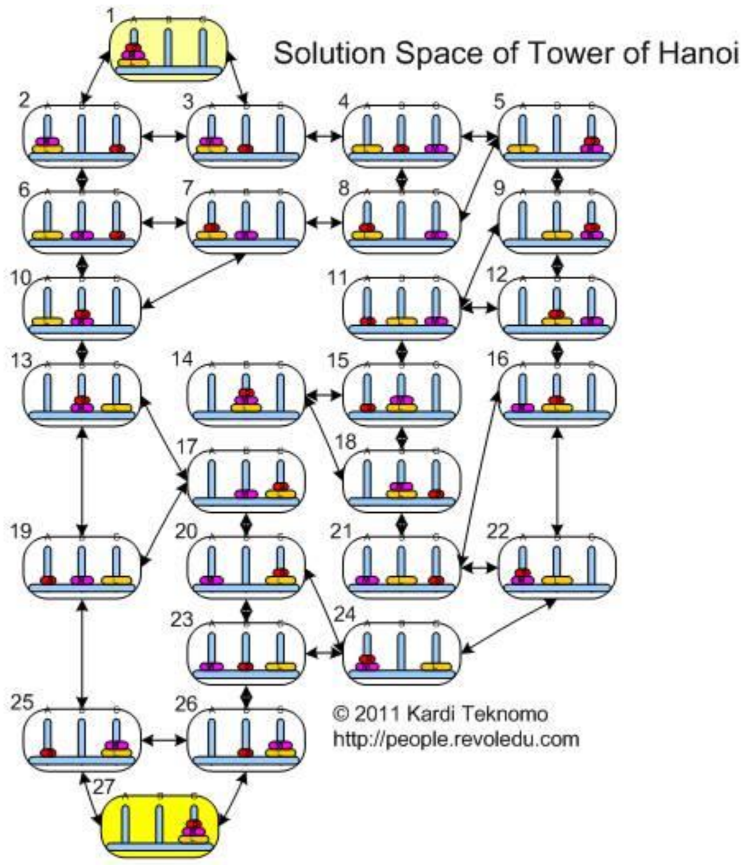
Three of the founders of modern Cognitive Psychology, George A. Miller (who proposed that working memory has a limit of 7±2 – a proposal that will be critically discussed in this module), Eugene Galanter and Karl Pribram produced an influential textbook called *Plans and Structure of Behavior* in 1960. In this book, they proposed a simple model that could be used to model cognitive activity. This was called T-O-T-E, or Test-Operate-Test-Exit. In this model, a 'goal' is subjected to a 'test' to check whether it has been achieved. If not then you 'operate' (or do an action) and then repeat the 'test'. You keep doing this until the 'goal' is achieved, and then you 'exit' (or stopping doing this). This simply idea was surprisingly influential both in cognitive psychology and in computer models of problem solving, particularly ones that used production systems. Of course, what T-O-T-E doesn't tell you is how you get the 'goal' in the first place or how you decide what to 'operate'. But, compared with the Stimulus-Response models of Behaviourism, it wasn't a bad starting point for a theory of cognition.

What T-O-T-E requires is that the 'goal' is thoroughly specified; you have to know when to 'exit' and when to 'test', and that there is a clear and obvious mapping between the 'goal' and the actions that you can 'operate'. As an illustration, imagine that you have a well-defined problem like the Tower of Hanoi. The 'goal' is to move the stack of discs from one peg to another while obeying two rules: only move one disc at a time, do not put a large disc on top of a small disc.



From this we have not one 'goal' but two…move the stack of discs, make sure that small discs go on top of big discs (and not vice versa), and one operation which is 'move one disc from a peg to another peg'. For TOTE, we would set the goal as move the discs from peg 1 to peg 3. Our test would ask 'are the discs of peg 3?' and 'is a big disc on top of a small disc?' The answer to both of these tests is no, so we move to 'operate' and move the top disc to peg 2 or peg 3. And then we keep repeating the steps, moving discs from peg to peg until we have solved the problem.

This might appear trivial but it captures some essential elements of task analysis. The first is the need for a clear definition of the 'goal'. If it is not possible to define a goal in clear and unambiguous terms, then you either need to try harder or accept that task analysis is probably not the right method. For a 'goal' to be clearly defined, there needs to be an unambiguous measure of success so that the action can finish. So, 'use a cashpoint' is not a goal in this sense, but 'withdraw £20 from a cashpoint' is. The second is that activity often involves several goals. Some of these are subgoals of a high-level goal, and others compete with each other. A good task analysis should be able to reflect the hierarchy of goals and subgoals, and be able to account for goal competition. Third, the decision of when to 'operate' and (in more complicated activity) which action to operate, is dependent on the current state of the problem which is defined as the result of the 'test' activity. This requires a clear definition of each operation that could be performed and clear criteria against which to test. So, returning to the Tower of Hanoi, we might elaborate the operations from the single statement of 'move one disc from one peg to another' into two statements, 'move one disc from its current peg to a peg to the left', 'move one disc from its current peg to a peg to the right'. We could continue adding to the list of operations until we have exhaustively described all possible moves and this would produce a solution space, like this:



With a problem such as the Tower of Hanoi we can write a programme that would allow a computer to solve this problem, and that could learn to solve problems that are similar to this, e.g., with the discs starting on a different peg or with more than three discs or more than three pegs. In effect, the purpose of task analysis is to describe efficient ways in which a solution space like this can be navigated by a person.

Hierarchical Task Analysis (HTA)

HTA was first outlined by John Annett and Keith Duncan in 1967. It is both a method of goal decomposition and a method of constraint definition. People sometimes forget the second point, but without it you are merely describing rather than analysing tasks. HTA generates two outputs: a diagram that illustrates the goal decomposition (which can also be presented in the form of a table) and a set of 'plans'. In HTA, a plan describes the conditions under which a set of operations will be performed. It is the plans that are the most valuable output of HTA because it is these that allow the analyst to reflect on the cognitive activity of the person – and then to decide whether there needs to a change in the information provided to support this activity.

Hierarchical Goal Decomposition

HTA involves decomposition of goals into subgoals. Each goal / subgoal is defined in terms of three components:

- Task Statements are defined as <Verb> <Object>. This should be as unambiguous as possible. Thus, 'Move disk' or 'Place disk on peg' would be a viable task descriptions. If there was more than one door, then you would need to include a further definition of the object. Task definition are presented in the HTA diagram and are typically numbered, so that in a plan, it is the task number that is included rather than the description.
- Performance Standards indicate any criteria that need to be met for successful completion of the task. These could be defined by time, e.g., within 5s, or physical parameter, e.g., to 100^o.
- Conditions indicate where the task is performed and what materials or tools might be assumed to be present.

Normally I use Post-It notes (stickies) to record each Subgoal and this allows them to be move around. It is rare to have a complete decomposition at first attempt, and as you inspect and check the hierarchy you find that you need to move or add subgoals. I also tend not to begin with the 'goal' as a formal concept until I have worked through the hierarchy a few times. This is because the overall performance objective that the goal describes might change as the analysis develops. I also find that use Post-It notes and the idea of goal decomposition is a useful form of semi-structured interview with Subject Matter Experts.

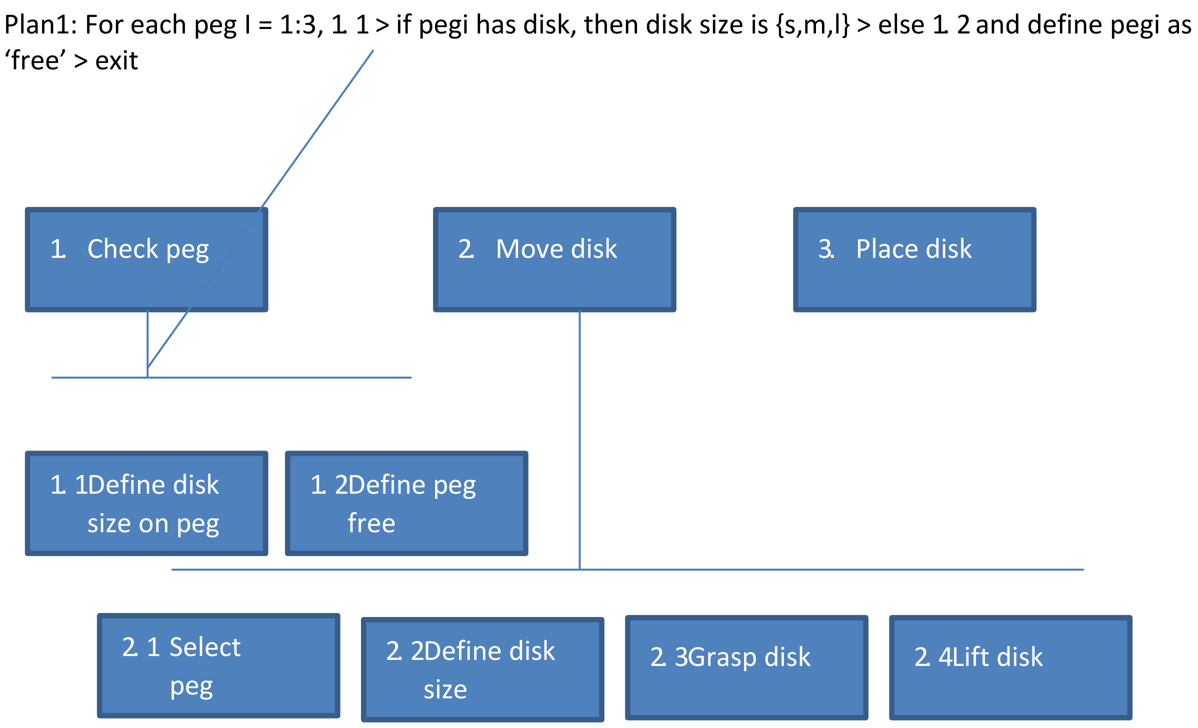
While the aim is to produce a goal decomposition, I typically begin by looking at the objects that are available to the person performing the task and what these might typically be used for. This allows me to work bottom-up from object to possible goals associated with these. So, if I am looking at the use of a ticket vending machine I begin by listing the objects that might be involved in this activity, such as ticket (printed by machine, indicating destination…), money (to put into machine, to pay for ticket…), touchscreen (to select destination, to select ticket type…), coin slot (to insert money (coins)), note-feed (to insert money (notes)), collection tray (to collect tickets or change…). This is not an exhaustive list but allows me to look around the machine (or environment) to determine what my task description needs to contain. It also begins to define the Conditions under which the subgoals are achieved and the Performance Standards that relate to these.

Next, I look at the operations that can be performed with these objects and write these combinations onto Post-it notes. As part of the semi-structured interview process, I would ask the Subject-Matter Expert to define these combinations. From these I begin to work out ways in which objects and operations combine to support goals.

Often goals are similar in different contexts, and it can be useful to have an initial idea of how to approach their description. Omerod and Shepherd (2004) suggested that this could be achieved through the use of templates.

Subgoal Template	Unit-task	Context
ACT: operation as part of sequence	A1: Activate	To make a component operational, e.g., to switch from off to on
	A2: Adjust	To regulate the activity of a component, e.g., increase or decrease
	A3: Deactivate	To make a component non-operational, e.g., to switch from on to off
EXCHANGE: to share information with another agent	E1: Enter	To record a datum to a specific location
	E2: Extract	To obtain a datum from a specific location
NAVIGATE: to search for a specific location	N1: Locate	To find the specific location of a datum
	N2: Move	To go to a specific location
	N3: Explore	To browse a set of locations and / or data
MONITOR: to continually check the state of components to changes	M1: Detect	To routinely compare the current state of components against a target state
	M2: Anticipate	To compare current (or pending) state against target state to determine readiness for known action
	M3: Transition	To routinely compare current state to target state in order to determine rate of change

The definitions of unit-task in this table can be useful to help select the <verb> part of the Task statement. Other useful sources include Task classifications, such as the one in Appendix A. As an example, consider the subgoal of 'Select Destination' on the ticket vending machine. This requires a number of operations to complete, e.g., you need to: know the name of the destination (which you can obtain from checking a map of the train network, or from checking the invitation you have been sent, or from asking the person you are travelling with, or recalling from your memory of a previous journey to the destination etc.), make sure that the machine is in the correct mode or state to allow you to select a destination, know the initial letter of the destination, select the initial letter, check the list of destination options, select additional letters until your desired destination is on the list, select the destination… Notice that the selection of destination in this example follows a TOTE model of activity in which you *operate* 'type letter' until the 'destination in list' *test* is passed.


Plan1: For each peg I = 1:3, 1. 1 > if pegi has disk, then disk size is {s,m,l} > else 1. 2 and define pegi as 'free' > exit
1. Check peg
2. Move disk
3. Place disk
1. 1Define disk size on peg
1. 2Define peg free
2. 1 Select peg
2. 2Define disk size
2. 3Grasp disk
2. 4Lift disk

A partially completed HTA for solving the 'Tower of Hanoi' problem

For the Tower of Hanoi problem, a top-level Goal might 'Solve the problem in a few moves as possible'. This would require a subgoal that defines (and keeps track of the problem's rules and constraints, and the number of moves taken). In this figure above, I have introduced some variables that will also need to be tracked ('free' and 'disk size'). As 'move' would involve checking the peg, move a disk, place the disk. Drilling down further, checking the peg involves (in this description) scanning the pegs (1 to 3) to check if a peg has a disk on it or not. Assuming there is a disk on one of the pegs, this would define the peg that is used in 2.1. The 'move disk' description is, of course, not complete here. However, I would include this in a high-level plan, Plan 0 which would cycle between check peg and move disk (so that when a disk with lifted, you would go to check peg, task 1, and then have a further decision as to whether the lifted disk is smaller than the disk on a peg – if it is then place the disk onto the peg, if it isn't the place the disk on an empty peg.

As this extract from the HTA illustrates, each subgoal can be further decomposed. While there is meant to be a 'stopping rule[1]' to determine when a sufficient level of detail has been reached, few people make use of this. Rather, there are three heuristics that are applied: 1. Never have a single subgoal below a higher one; if you cannot have a branch with two subgoals then stop; 2. Never describe beyond 5 subgoals (some people suggest 8 or 10 as the limit, but my feeling is that anything more than 3 is probably getting a bit messy and more than 5 is becoming unduly complex); 3. Stop decomposition when the subgoals below this involve micro-actions, i.e., if the subgoal is 'release disk', this is a good point to stop (because 'open fingers' etc. feels overly detailed)…unless the analysis explicitly focuses on micro-actions.

[1] A Stopping rule is often defined in terms of the likelihood that further decomposition will have utility. This *could* be quantified as the Probability of poor performance of the task x the Cost arising from that poor performance. Very few people bother with this formulation (it is not so easy to define the numbers in the first place).

A second point to be made regarding this extract is that the subgoals below ‘define destination’ do not *all* need to be performed; rather, the choice of which of these to pursue depends on the relations between the conditions of the person and the environment. These relations are defined by Plans.

Plans in HTA

In HTA, a plan is really a TOTE unit. So, rather than write ‘move disc A from peg 1 to peg 2’, a plan would include the criteria that need to be tested. The criteria could be phrased as two questions: is the peg occupied? If so, if the peg occupied by a disc smaller than disc A? So, now the plan is elaborated to ask (assuming that you have already decided to move disc A – and this decision would be made at a higher level in the hierarchy): Is peg 2 occupied? If no, move disc A to peg 2; if yes, is peg 2 occupied by a disc smaller than disc A? Is no, move disc A to peg 2; if yes, exit. You can readily see how this form of plan can be represented as a decision flowchart or as production rules (If…then…). The reason why this is potentially useful is that it forces you to determine the conditions and criteria used to defined the choice of an operation, and thinking about these gets you thinking about how the person is making decisions as they perform the task. This is one of several reasons why I disagree with people who claim that HTA is only about physical tasks.

At the simplest level, plans only define a Sequence of operations. These could be:

- Linear (1>2>3>exit)
- Non-linear (1/2/3>exit)
- Simultaneous (1+2+3>exit).)
- Branching (If X then 1, else 2>3>exit)
- Cyclical (repeat (until condition): 1>2>3>exit)
- Selection (If(condition)select 1:2:3:4>exit)
- Waiting-for (Do 1, until t = 2m > exit; Do 1, until x > exit)

Returning to the HTA extract, we could write:

```
px.1: if destination known, x.1.4,
        else if map of network is to hand, x.1.1,
                else if invitation to hand, x.1.2,
                        else x.1.3,
                                exit
```

The order of these operations is *not* defined in the HTA but by the conditions (defined by the ‘if’ statements) in the plan. This offers some flexibility for the analyst in that the same diagram can be read in different ways, depending on how the plans are defined. This means that it is possible to create different plans, depending on the information that is available in the environment or on the subgoal’s measure of success.

References

Berliner, C., Angell, D. and Shearer, J. (1964) Behaviors, measures and instruments for performance evaluation in simulated environments, *Symposium and Workshop on the Quantification of Human Performance*, Albuquerque, NM

Kirwan, B. and Ainsworth, L. (Eds.) (1992). *A guide to task analysis*. Taylor and Francis

Miller, G.A., Galanter, E. and Pribram, K.H. (1960) *Plans and the Structure of Behavior*, New York: Henry Holt.

Ormerod, T.C. and Sheperd, A. (2004) Using task analysis for information requirements specification: the sub-gaol template (SGT) method, In D. Diaper and N.A. Stanton (eds.) *The Handbook of Tasks Analysis for Human-Computer Interaction*, Mahwah, NJ: LEA, 347-365.

Shepherd, A. (2000) Hiearchical Task Analysis, London: CRC Press

Stanton, N.A.; Salmon, P.M.; Walker, G.H.; Baber, C.; Jenkins, D.P. (2005). *Human factors methods: a practical guide for engineering and design*. Aldershot, UK: Ashgate

APPENDIX A: Classification of Tasks (from Berliner et al., 1964)

Process	Activity	Example
Perceptual	Search for / retrieve information	Detect; Inspect; Observe; Read; Receive; Scan; Survey
	Identify object, action, event	Locate; Identify; Discriminate
Mediational	Process information	Calculate; Categorize; Compute; Encode; Interpolate; Itemize; Tabulate; Transfer
	Solve problem / Decide	Analyze; Choose; Compare; Estimate; Predict; Plan
Communication		Advise; Answer; Direct; Indicate; Inform; Instruct; Request; Transmit
Motor	Perform discrete task	Activate; Close; Connect; Disconnect; Hold; Join; Lower; Move; Open; Press; Raise; Set
	Perform continuous task	Align; Regulate; Synchronize; Steer; Track; Transport