

Using task analysis for information requirements specification: The SGT method

Thomas C. Ormerod, Psychology Department, Lancaster University, LA1 4YF, UK.

t.ormerod@lancaster.ac.uk

Andrew Shepherd, Synergy Consultants.

andrewshepherd@symos.co.uk

Chapter 16 in D. Diaper & N. Stanton (Eds.) (in press) *The Handbook of Task Analysis for Human-Computer Interaction*. London: Lawrence Erlbaum Associates

Overview

Designers increasingly recognise the need for user-centred approaches to system specification, yet the last thing they want is for yet another design method to be imposed upon them. To receive widespread use, task analysis methods must be an integrated and primary part of the design process. This chapter describes the Sub-Goal-Template (SGT) method for producing requirements specifications from task analyses. The SGT method captures a set of information requirements for each user task, and documents the commonalities and differences among tasks and task sequences under constraints of context and limits of performance. Critically, this information is specified in a form that can be used directly by clients and designers as a specification of information requirements, and even as tender and evaluation documents. A brief overview of the SGT approach to hierarchical task analysis is illustrated with an example showing the design of supervisory control systems for a simplified railway network.

Introduction

Articles on task analysis often begin with a plaintiff cry that systems developers fail to consider the human user sufficiently early in the design process, focussing instead upon the specification of functional requirements. We began most of the papers we have written to date on task analysis in a similar vein. Systems design methods emphasise the configuration of elements to process materials and information, but these design methods do not take full account of the tasks that must be carried out to supervise or control processing elements. This argument still needs to be made: Designers do not always recognise the need for user-centred methods such as task analysis, believing that systems design methods are what design support is all about. Without understanding the tasks that human beings are required to carry out, many design decisions risk creating design errors. If people are involved in setting up equipment it is important to know this because it can affect where equipment is sited and the environmental protection that must be provided for personnel. Equally, it is important to establish, early on, how many people will be involved in operating a system because the general manner of their control, where they work and how they will communicate with each other constitute design constraints that will affect the choice of system elements, site and access. Indeed, recognising the contribution that an operating team will make can affect engineering considerations such as the use of automation as well as how that operating team will interact with the system.

Despite recognising the continued importance of task analysis in the design process as a means of identifying design constraints, in this paper we want to rehearse the

converse argument: Designers are *right* to be suspicious of task analysis methods that concentrate upon human-computer interactions at the expense of the engineering of the system. In reality, systems designers increasingly recognise the need for user-centred approaches to system specification, but they are also concerned to preserve control over specifying the functionality of systems. This is understandable and essential to ensure that systems meet their objectives. Designers must constrain the manner in which systems are used in order to preserve security, safety, productivity and other commercial and functional benefits. Consider the following contexts:

- In service sector organisations (e.g., telephone sales) careful procedures must be followed to protect customer identity information and to ensure transactions are completed securely.
- In the supervision and control of physical plant, it is necessary to preserve safe operating conditions, while at the same time optimising productivity and maintaining plant and equipment in a serviceable condition.
- The supervision and control of transportation systems must allow safe yet commercially practicable operation. Aircraft movement is constrained by rules for maintaining aircraft separation as well as by principles governing payload and flight. The movement of trains is constrained by signalling rules as well as timetabling.
- The practices and procedures adopted in the health services to provide care and treatment for patients must be dictated by nursing and medical knowledge to ensure the well-being of the patient and legal constraints to protect the interests of staff and their employers.
- Designers of computer-based user interfaces need to record the constraints upon how tools and equipment are controlled via the interface - pumps should not be started before they are primed, missiles should not be fired before the target is located and fixed, and so on.

In each of these contexts, there are critical user-centred issues: operators need appropriate training, expertise, instruction, feedback, supervision, workload, communication channels, and so forth. However, there are also physical, temporal, regulatory and other properties of practical and safe system operation that must be captured and understood. These are not issues about the capabilities or otherwise of the user: they are issues about the use of the system, and rightly they should remain the central focus of the designer. Thus, the systems designer must be clear about operational constraints as well as the human factors of interacting with systems. These should be the twin aims of task analysis in interactive systems design. The modest extent to which task analysis methods have been adopted by the systems design world is well-documented (e.g., Dillon, Sweeney & Maguire, 1993; Lansdale & Ormerod, 1994; Diaper, 2001). In our view, this slow uptake can be explained, in part, by the failure of task analysis methods to deliver both operational and interactive requirements.

The widespread use of task analysis methods is further inhibited by the fact that designers frequently find them to be non-intuitive and jargon-riddled, and to carry heavy learning overheads. Many task analysis and other human-centred methods do not deliver a usable product to the designer - they may alert the designer to important issues, but their outputs are not clear signposts to implementation. Consequently, designers end up

working with poor quality outputs of poorly conducted task analyses (cf. Parker, 2001), while carefully crafted methods remain the rarefied tools of the researchers who developed them. Our philosophy behind developing the SGT (Sub-Goal Template) method was to provide a single approach to capturing systems operation and user information requirements (for detailed descriptions of the SGT method, see Shepherd, 1993; Ormerod, Richardson & Shepherd, 1998; Ormerod, 2000). At the same time, we set out to minimise the need for learning, jargon and special tools, while providing designers and clients with recognisably useful output.

In developing the SGT method, we were mindful of the need to leave designers in control of the design process. Task analysis has been criticised (e.g., Benyon, 1992) for over-constraining the solution options that a designer can consider. In part, this view stems from the (in our opinion, incorrect) assumption that task analyses should model only existing user-system interactions. Where this is the case, then the use of a task analysis method condemns the designer to work within the constraints of the existing interaction method. This need not happen if the analyst is alert to the fact that neutrally stated operations may, in principle, be redescribed using different methods that use different technologies. For example to ‘transfer luggage from one airplane to another’ may be achieved by the baggage handler making lots of journeys on foot, by loading a truck or by installing a conveyor belt. To explore this task at a lower level of detail, it is necessary to focus on one technology rather than another, for each has implications for what the baggage handler will be required to do. If the analyst can remain alert to the distinction between the functional requirements of the task, i.e. to transfer the luggage and the method by which a transfer is made, then it is possible to consider different ways of doing things.

Most task analysis methods stress the need for the analysis process to be iterative and reflexive, and we concur with this view. Task analysis should be a process of information gathering, organisation, negotiation and revision. However, the analyst needs stopping rules. On the one hand, stopping rules should provide an unambiguous decision procedure that enables the analyst to determine when a task has been analysed to a point at which its structure has been decomposed, documented and contextualised in sufficient depth that nothing important remains unreported to the designer. On the other hand, stopping rules should force an analysis to cease before it starts dictating *how* tasks should be implemented by the designer. Maintaining a clear separation between task analysis and design is extremely difficult. We believe that this separation needs to be addressed, and doing so provided a key motivation for the SGT method.

In the remainder of this chapter, we describe the SGT method, and how it provides designers with information requirements at an appropriate level of analysis to allow creative design. First, we examine how the scheme inherits the processes and products of hierarchical task analysis (HTA), recognising both the strengths of HTA and also its perceived weakness in delivering to designers a directly usable product. We then outline the SGT method itself, and finally describe a brief example of applying the SGT method to the design of supervisory control systems.

Hierarchical task analysis

HTA was developed by John Annett and Keith Duncan in the 1960s as a general method for examining complex tasks (e.g. Annett and Duncan, 1967; Annett et al., 1971;

Duncan, 1972; Shepherd 2001). A more complete account of HTA is provided elsewhere in this volume, so here we focus upon key concepts that are inherited by the SGT method.

Operations

HTA focused the analyst's attention on the industrial or commercial goal to which the operator's effort in executing the task was directed. Its principal unit of description was the operation. An operation is a statement that briefly describes the operator's current objective, expressed as an imperative. For example, the operation to correct a fault in a washing machine might be described as 'repair washing machine'. This form of description is versatile and can be used to describe broader activities, such as 'maintain domestic appliances' or more focused activities such as 'replace inlet filter from the cold water feed'. This facility to describe activities at different levels of detail enables the analyst to adjust the grain of description for different parts of the task. Thus a task can be decomposed into its components, which can in turn be further decomposed.

An operation is stated in a way that is neutral with respect to its solution (Duncan, 1972, used the phrase 'psychologically celibate'). It is important to distinguish between an operation and the behaviour that enables it to be carried out. As far as a system is concerned, the precise strategy by which an operator carries out an operation is immaterial provided it is successful and does not introduce unacceptable consequences. Performance strategy could be guided by a job-aid, recalled as a consequence of repetition during training, recalled from a recently successful encounter in similar circumstances, developed through the operator's analysis of a current situation, or prompted by the affordances of a display.

Operational behaviour, however it is executed, is goal directed. The operator selects actions from those available at the task interface in order to realise the goal. Actions are regulated by information gleaned from the system, and can be described using Annett and Duncan's input-action-feedback (I-A-F) classification. This information will provide detail about the system (input), indicating when action is necessary and guiding choices between appropriate actions (action), as well as indicating when action may cease (feedback). Using this classification, the analyst identifies which aspects are likely to be at the root of performance problems and so guides the generation of hypotheses to overcome these problems.

Stopping Analysis

The analyst needs principles to guide where there is benefit in proceeding with further redescription. For this purpose, Annett and Duncan suggested a stopping rule where the analyst estimates the probability of inadequate performance (P) and the cost of inadequate performance (C); their stopping rule was that analysis should be taken no further than a level of redescription where the product of P and C was acceptable to the client. To estimate C, the analyst establishes what is of value in the system, including the commercial consequences of failure (including production costs and consequences), and risk to the health and safety of personnel and the public. Costs can be offset by benefits. So, for example, commercial benefit could include anticipated long term benefits to offset immediate investment for which a return has not yet been realised. Incurred costs could also provide opportunities for staff to obtain experience and gain expertise that will provide a return in the longer term. The cost factor is an amalgam derived from the analyst's enquiries and should reflect the system and not the analyst's values. To estimate

P, the analyst can use reliability data or performance shaping factors. The analyst would consider the task demands, the personnel employed to take responsibility for the task, the resources available to support the task and the environment in which the task takes place. The $P \times C$ rule is a heuristic rather than a prescriptive arithmetic rule. Thus, if *C* is unacceptably high, then further analysis is warranted no matter how low *P* is (assuming that it will not be a perfect zero). If *C* is quite high it might still be worthwhile exploring the operation in greater detail, even where *P* is low. If *P* and *C* are both low, then there would be little benefit in exploring the operation because it would not substantially affect overall performance.

Plans

Another important component of HTA is the plan. If an operation is decomposed into component operations, then there needs to be an organisational system to indicate the conditions under which each component is carried out in order to complete the superordinate operation. For example, painting a window-frame entails ‘prepare window-frame’, ‘apply primer’, ‘apply undercoat’, ‘apply gloss-paint’. These operations must be done in an appropriate order – sanding down and washing must precede applying the different coats of paint, each of which should also occur only when previous coats have dried properly. Plans are crucial to HTA. They capture many of the subtleties of tasks which cannot be explained merely by reference to operations. Shepherd (2001) illustrates numerous plans, showing how different types of plan combine to account for complex performance. Thus, some plans deal with sequences of activity, some with cycles of activity and some with choices. By combining these in an HTA hierarchy results in apparently complex arrangements that are better understood by seeing how they result from the interaction of simpler plans.

Utility and Transparency of HTA

Annett and Duncan’s basic ideas were informed by principles from human factors, systems thinking and operational practices. The HTA method employs natural language to express operations and task conditions in plans. Consequently, system designers may engage their clients directly in agreeing and expressing what must be done. Equally, the client or the client’s agent can validate an analysis to confirm that it properly reflects the tasks to be carried out.

By adopting the unit of an operation, HTA maintains neutrality in task descriptions. An operational description indicates what must be achieved and not how it is achieved. The designer is at liberty to propose different technologies that might engage the human operator in physical work, mental work, or no work at all. In providing a human factors or resources solution to a performance problem, many solutions may be evaluated and selected on the basis of cost, consistency with other solutions, and utility. Thus, human performance problems may be solved by improving the environment, personnel selection, training, interface design or a combination of these and other methods.

The stopping rule, or more pertinently the principle that the hierarchical description may be developed to different levels of detail in different places according to the respective importance of these places, means that the values of the system, and not simply those of human factors, are taken into proper account in the human factors contribution to the design process. It means that human factors cannot dictate the route

which the human operator's contribution will take unless it can be first demonstrated that the issue addressed is important.

HTA and Interface Design

HTA has been shown to be capable of providing useful descriptions of a variety of tasks in many contexts. Equally, it has been shown to aid human factors and resources decision-making, including the design of teams and jobs, operating procedures, selection methods, interface design, and training, as well as reliability assessment and quantification. For many of these solutions, HTA provides a structure which facilitates the application of other human factors methods and principles (Shepherd, 2001). In addressing the issue of *interface design* we are seeking a method that preserves the benefits of HTA with regard to thoroughness, flexibility of description, utility and transparency. Interface technologies support communication between operators and systems. Typically, the operator elicits information about the system's status in order to make decisions about how best to exercise control over the system, then expresses these decisions in a way that will affect the required changes. For this purpose, those aspects of HTA concerned with stopping rules need to be reviewed. We should distinguish between existing systems where a redesign of an interface is envisaged and new systems where an interfacing method has yet to be prescribed.

In an existing system the decision to revise an interfacing method may be taken following a completed task analysis, including an estimation of the risk entailed using the existing interface technology. The analyst must establish how the operator uses the existing technology. The decision to change may involve judgements about the suitability or otherwise of improved training, job-aids, recruitment procedures or other organisational factors. If redesign of an interface is judged essential, then it will be necessary to back-track up the task hierarchy to a place where the functional requirement of interaction is expressed in broad detail, before the level at which the interfacing technology was assumed. This level is the starting point from which a new technology or approach can be devised. If interface design is considered for a new system, then the analysis should proceed as far as this point of functional specification and no further, since there would then be a need to make assumptions about how interfacing is to be accomplished. At this stage a method for interface design can be considered, and it is at this point that the SGT method is utilised.

Thus, neutrality in terms of design hypothesis is maintained within HTA by two features. First, the analyst preserves operational neutrality in task descriptions, thereby enabling different types of solution to be proffered. Second, HTA moves from general operational statements towards specific methods where devices for delivering solutions are assumed. This means that an analysis can, in principle, be developed in a number of ways so long as different technological solutions can be entertained. If one such development proves unsatisfactory, then the analyst may revert to the common, neutral, point and consider another.

The SGT method

The task of information requirements specification consists of specifying the reception and transmission of information (including exercising control) that the operator requires to carry out tasks in order to operate a system. The sub-goal template (SGT)

scheme was devised to provide a nomenclature for classifying and organising the stereotypical tasks that operators undertake, so that the information necessary for each task could be collated across the analysis to give a requirements specification for systems designers. Our aim was to retain the simplicity and utility of HTA, by providing a scheme that captures the vast majority of operations in as few categories and hierarchical levels as possible (i.e., to minimise the breadth and depth of task classes). In general, we adopt the view shared by others in the task analysis community (e.g., Payne & Green, 1986) that recognising consistent structures in any task environment is good design practice. The resulting SGT scheme comprises a categorisation of operator task by sub-goal, given an overall goal to control a system in pursuit of a desired outcome.

Insert Figure 1 about here

Insert Table 1 about here

The SGT method terminates in the assignment of SGTs to each task element in a task analysis, at which point information requirements can be assigned. To get to that stage requires the analyst and engineer or software developer to undertake a process of ongoing negotiation. Figure 1 shows a schematic description of this process, illustrated as a task analysis of a generic process-control operation (to maintain a system). The main stages of this generic process-control operation are explained in Table 1. This shows maintaining plant to be described in terms of two main activities, ‘monitoring plant’ and ‘dealing with significant perturbations’. The analysis consists of two kinds of activity: task decomposition, where tasks are broken down into their constituent sub-tasks, and task redescription, where tasks are reconceptualised using the concepts and notation of the SGT method.

Information Handling Operations (IHOs)

When an analyst works with an engineer or a systems developer to specify the control of a plant or software application, they deal with two types of operation: functional operations concerning the states that have to be attained for the system to work and informational operations which specify information and control interventions necessary to attain these states. Our starting point for the SGT method is the recognition that all systems, whether they are process plants or PC applications software, will at some point require information to be handled outside the system being controlled – indeed, that is generally the reason why people are employed. These are information handling operations (IHOs). A task can be redescription as an IHO if it provides the point where system activity requires an information-based intervention.

IHOs involve the same three classes of activity: receiving information, evaluating information, and acting on information. For example, when a red wavy line appears under the text being typing, the system is engages the user in an information handling

operation. The user has to receive information (that there is a problem with a word located above the wavy line), evaluate it (the word-processor does not recognise the spelling of “SGT”, but it is an acronym that we wish to retain), and act (to instruct the spell-checker to learn or ignore the acronym). A ‘receive’ component might involve an operation to obtain information about the status of the system (including categorical states, trends, prognoses), to receive a communication (usually a command to execute a specified action, achieve a specified state, or adjust current operating goals), or to retrieve data (about targets, previous states pertinent to current control etc.). In effect, information can come from the system, from another operator or from within the operator themselves. An ‘evaluate’ component might require the operator to monitor a system, diagnose a fault, plan a sequence of activities, or manage and oversee a change to a system. An ‘act’ component may require an operator to carry out action to change the state of the system, to issue a communication (usually to another person to execute a specified action, achieve a specified state, or adjust current operating goals), or to record data for future use.

Once the analyst has decomposed a task goal sufficiently to recognise that a sub-task is an IHO, the method requires that decomposition ceases, albeit temporarily. The reason for stopping the decomposition process here is that it is the first point at which the neutrality of task analysis is challenged. There are potentially an infinite number of ways in which an information handling operation might be conducted. For example, the designers of word-processing software might have decided to delay the indication of spelling errors until the user has invoked a spell-check command, or they might have decided to provide automatic correction of errors. The way in which an information handling operation is carried out is a *strategic* decision. The task analyst cannot simply decompose it - the strategy must be negotiated with the client, engineer, software developer, and so on. Often, the strategy will involve elements that are not supported by interface design. For example, in a power plant, the sub-task “rectify fault” might be dealt with by requiring the operator to diagnose and adjust the system, or by triggering an automatic mandatory shutdown. The task analyst cannot decide what the appropriate strategy will be in advance of negotiation. Once the analyst has identified the designer or client’s (current) chosen strategy for achieving an IHO, then each of the ‘receive’, ‘evaluate’ and ‘act’ components can be further decomposed into their constituent tasks.

Sub-goals

Insert Table 2 about here

To provide an unambiguous stopping point at which the analysis can be handed over to the interface designer, there needs to be a specification of the lowest level of operator task below which there should be no further decomposition. The initial SGT scheme (Shepherd, 1993) adapted the structure of process-control activities outlined by Annett & Duncan (1967). At any particular point in time, a process controller may have a sub-goal to change the state of a system (by action), to observe the state of a system (by monitoring), to communicate information with or about a system (by communication), or to repair a system (by diagnosis). We subsequently extended the scheme to provide a nomenclature for classifying the stereotypical tasks that arise in the operation of other

kinds of interactive system (Ormerod, 2000). This extension involved the addition of two sub-goals, to exchange information with a system (by entering or extracting data), and to navigate through a system (by moving, locating or browsing). These additional sub-goals broadened the remit of the scheme to account, for example, for office information systems, ubiquitous computing devices, and so forth. However, the extensions came at the cost of coherence and overlap in the scheme. One of our goals here is to present a revision to the scheme. In particular, the original Communication sub-goal of Shepherd (1993) is merged within the Exchange sub-goal. Furthermore, we recognised that the 'diagnosis' set of sub-goals included in the original scheme must be understood at a strategic level, and have dropped this SGT from the scheme. The revised classification scheme is shown in Table 2.

At a lower level, each sub-goal is differentiated by 'task elements', that is, specific variants of the sub-goal. The reason for this lower level of specification is to capture the point that operators may set up and execute the same sub-goal for a number of different purposes, depending upon the stage and conditions of the activity they are accomplishing. So, for example, the Monitoring sub-goal is sub-divided into three specific types of monitoring element. At any particular point, the purpose of monitoring may be to check that system is functioning as the operator expects (to detect deviance), to check a system prior to further action (to anticipate change) or to inspect rate of change from one state to another (to inspect transition).

The point of the task element level of categorisation is to provide designers with a way of supporting the basic operations that an operator must accomplish. For example, traditional process-control displays have presented the operator with a bank of indicators, alarms and controllers (or their computer-based equivalents), one set for each component (tank, valve, pipe, etc.) in the plant. The shift from pneumatic to computer-based control raised a problem regarding the monitoring of complex processes: how can an operator monitor a large system when they can only view a small fraction of it at any one time on a computer monitor? Relying on automated alarms is insufficient - when one device goes wrong in a heat exchange loop, for example, every alarm will go off on every device as the interdependencies among components work their way through the system. Supporting monitoring by allocating displays and alarms to every device is a recipe for information overload, of the kind implicated in one of the most widely publicised accidents in recent year, namely the Three Mile Island emergency (Rubinstein and Mason, 1979). Task elements specify the tasks that operators must carry out in order to ensure the proper functioning of a plant. They are intended to aid the designer in specifying precisely how to support the activities of monitoring by appropriate information without overloading the operator.

One obvious question is whether the SGTs and task elements that we have specified are complete, or whether there remain further SGTs or task elements to be discovered. We are open to the possibility that others may be found, though we would seek strong grounds for expanding the scheme, since we aim to keep it as simple as possible. The only justification that we would accept for expanding the scheme would be if new SGTs or task elements were found that had unique *information requirements*, a topic to which we now turn.

Information requirements

As well as providing a taxonomy of task types, the SGT method captures the basic information requirements of each SGT element, and it is at this point that we believe we provide the designer with precisely the information they need to develop a detailed systems design specification. Determining the information requirements of a specific SGT element seems, at first, problematic, because an operator needs to conduct a number of discrete activities in order to execute an SGT-level operation. Take, for example, the SGT 'A2 - adjust'. This task element might itself be decomposed into the following sequence of activities:

- Locate adjustment device
- Review operation of adjustment device
- Review adjustment to be made
- Execute adjustment
- Monitor system response
- Deal with deficiencies

This set of activities is, prototypically, the 'template' of operations needed to execute the 'adjust' sub-goal (hence the name 'sub-goal templates'). However, given that each of these operations is itself an SGT element, one could end up recursively decomposing operations ad infinitum (indeed, the 'deal with deficiencies' operation will itself entail an adjustment operation). In order to provide an absolute boundary to the process of operational decomposition, we have converted each of the stereotypical sequences of operations that are necessary to execute an SGT into a list of the minimum information that would be needed if these operations were to be undertaken. So, for example, the 'A2-adjust' task element requires information on:

- Pre-conditions, action points, and order (to locate device, review operation and execute adjustment)
- Current, alternative & target states (to review adjustment to be made);
- Rate of state change, plus outcomes and dependencies (to monitor system response);
- Halting and recovery indicators (to deal with deficiencies).

In developing the SGT method, we adopted the following rule: task elements are added to the scheme only where they possess one or more unique information requirements. This rule follows from the belief that we should provide the minimum necessary task nomenclature. Information requirements, broadly defined, comprise the data that an agent (be it human or automated, single or multiple) needs in order to accomplish a task safely and efficiently. Some information requirements are common to the task elements of an SGT. So, for example, all the task elements of the Action SGT require a specification of action points and the order in which they must be sequenced; the current, alternative & target states; necessary pre-conditions, outcomes and dependencies; and halting, reversal, and recovery information. The 'A2 - adjustment' task element, however, necessarily requires information on the rate at which a system changes in response to the ongoing adjustment. It is an essential information requirement that distinguishes it from other actions where such information is optional (and might, therefore, be unnecessary or even distracting).

The information requirements of each task element are neutral as to the chosen method of implementation. For example, one way of supporting the information requirement to provide halting and recovery indicators would be some kind of 'Undo'

facility. Alternative methods might include some kind of shutoff or ‘panic button’, an automatic limiter, and so on. The nature of the solutions provided for each information requirement must, in our view, remain the preserve of the designer, but the information requirement offers a cue to their provision. The resultant design should then be subject to task analysis to confirm that it is appropriate and will support reliable performance.

Sequencing elements

Insert Table 3 about here

The inclusion of plans in HTA recognises the importance of indicating the conditions under which component operations are carried out in order to satisfactorily complete a task goal. Including sequencing elements in the SGT method was, to some design researchers, controversial, because it mixes data and control flow in the same specification (e.g., Benyon, 1992). However, as Ormerod (2000) argues, sequences of operation dictate information requirements as much as operations themselves. For example, if the designer is developing an interface to start up a plant, then it is essential that the same interface supports the parallel task of monitoring the state of the plant during start-up.

The SGT method capitalises upon the fact that there are, in principle, only four ways in which operations can be sequenced relative to each other:

- Operations may follow each other in a strict and mandatory sequence, the outcomes of earlier operations providing the operating conditions for later operations. For example, “prepare”, “start-up” “maintain” and “shut-down” operations will always appear in a strict linear order regardless of intervening operations or conditions;
- Operations can be contingent upon the outcomes of earlier operations or contextual factors, and so arise in a sequence only if the necessary pre-conditions exist. For example, operations involving adjustment will typically be contingent upon other operational or contextual outcomes;
- Operations may require parallel performance. For example, system control actions generally must be performed alongside monitoring activities;
- Operations may not have a pre-determined order, but may instead be undertaken as and when the operator wishes to do so. For example, routine documentation updates (e.g., personnel logs) may take place whenever the operator has time to do them - their inclusion in a sequence of operations may be mandatory, but the precise point at which they occur may be determined on-line rather than in advance.

Table 3 shows the notation used to capture these sequencing elements. There are two points to observe about the notation. First, the S2 ‘contingent sequence’ element makes explicit the (usually implicit) sequencing constraint that there are always two states and/or outcomes that must be considered by the designer: what the operator tasks are if the pre-conditions do arise and what the operator tasks are if the pre-conditions do not arise. For example, a common contingency might be “If the alarm sounds then call the supervisor”. An “if not” condition may simply give rise to the operational statement “continue” (e.g., “If an alarm does NOT sound, then continue to monitor the process”). Nonetheless, failure to make such equilibrium-maintaining information explicit can lead

to problematic gaps in the information requirements specification. Second, the notation requires that each sequence is nested within a procedural flow of control. Nesting of sequence elements may, in some instances, reverse the order of tasks as they are naturally specified in a task decomposition. Table 7 (part of the railway network case study described below) contains an example of such a reversal. The need to ‘deal with problems’ may arise at any time during the operations of ‘ensuring railway conditions’ and ‘power up’. Thus, the plan for sequencing these three operations begins with an ‘S3 - Do in parallel’ sequencing element that makes explicit the need to deal with problems throughout the operational sequence.

Example: The supervisory control of a rail network

System supervisory tasks arise when human operators are employed to maintain conditions in a complex system. There are many such tasks, including the supervision of transport systems such as railway networks and air traffic control, and the supervision of automated production units such as flexible manufacturing systems, and process plants concerned with chemical manufacture, petroleum refining and power generation and so on. In each of these cases, we encounter staff responsible for controlling operations, usually through symbolic representation of the elements under control. Control staff must oversee automated systems to make sure that they are operating according to design and that no problems have arisen. Staff may be required to supplement automatic control, to alter operating rates and conditions, to control intermediate stages of system start-up and shut-down, or to deal with problems. There are several reasons why such remote, symbolic control is required. Reasons include operating from a position of safety away from risky physical environments, overseeing a large number of elements distributed remotely, and collating information from different sources in one place.

Supervisory systems have a wide range of technologies, but share many characteristics. These commonalities include responding to events that are not wholly under the control of the operator. For example, the movement of aircraft is a consequence of pilot decisions; movements must be monitored by the air-traffic controller through radar and voice communication with flight crew and controlled by issuing instructions which may or may not be wholly complied with. Chemical reactions generally comply with the laws of chemistry and physics, but their precise effects may vary according to processing conditions. A doctor may prescribe a course of treatment, but the patient’s response needs careful monitoring to establish the precise effects of treatment. Generally in such systems, controllers must obtain information, make judgements about how target conditions can best be reached or re-established, act on these judgements, then monitor the effects of their actions and adjust where necessary. The requirement for appropriate information, action controls and feedback is paramount.

Insert Table 4 about here

A simplified railway control task illustrates the application of the SGT method to supervisory tasks. The railway example shows how the SGT method presents explicit challenges for the designer in providing dialogue features to support the task. Table 4

describes the main operations involved in this task. In HTA, this is described as ‘Supervise Railway system’, a global goal which can be decomposed into 3 main operations governed by a plan. Note that each of the resulting sub-tasks can be further decomposed by the analyst. None of them necessarily require an information-based intervention for their execution.

Insert Table 5 about here

The activities involved in the sub-task ‘1 - Provide services’ are decomposed further in Table 5. In many system supervisory tasks, there are initial stages where the controller makes adjustments to operating targets: here these adjustments are concerned with amending the timetable. The controller then makes system checks to establish any impediments to progressing with the start-up. The decomposition of the ‘Provide services’ sub-task reveals a series of activities that are unambiguously IHOs - it is inconceivable that any of these supervisory activities could be conducted without information being received, evaluated and acted upon. It might be an automatic system that conducts these activities (e.g., some kind of intelligent agent software that automatically adjusts the timetable according to received information regarding deviations from normal running), though in the context of rail network supervision, the involvement of a human operator at this level is likely if not mandatory, on grounds of ensuring safe operation. Regardless of involvement of a human operator, the activities are IHOs, and the client, engineer or software developer needs to work with the analyst to further understand the strategies that might be considered to execute each IHO.

Insert Table 6 about here

Table 6 shows a strategic decomposition of the IHO ‘Revise timetables and rotas according to information’. This task is undertaken to review the timetable according to more recent events. The assignment of ‘Monitor to detect deviance’ SGTs to each of the monitoring sub-goals, and their associated information requirements (see Table 2) draws the designer’s attention to the types of information that must be provided to support this strategy. However, treating these task elements as monitoring activities is only one possible strategy for revising timetables and rotas. An alternative strategy might be one in which operators are trained to undertake regular sampling of information sources and to communicate these to another operator or device. This strategic approach would then be coded as a sequence of information exchange activities (‘E2- Extract’ followed by ‘E1 - Enter’ SGT elements). The point of the strategic decomposition is to make explicit the point at which the emerging design becomes committed to specific methods for implementing information handling operations, and to encourage a review of alternative methods. It also provides the designer with the impetus to consider more than one strategy, thereby avoiding the problems of satisficing (Simon, 1981) associated with poor quality design processes.

Interface design may be affected by many aspects of work design, such as assigning communication responsibilities to other personnel. Often a controller's decision making can be substantially simplified by requiring a colleague to be more explicit in his/her encoding of events. This is particularly the case in industrial environments where people must interpret the implications of information recorded in logs without any clear format. For example, the person entering the information could be made responsible for recording the clear implications for operating, so that less ambiguity would ensue. All of these contextual factors require explication, and a high quality HTA should capture them. However, the stage of strategic decomposition provides a semi-formal point at which the significance of such contextual constraints is evaluated for every strategic proposal.

Table 7 shows a similar examination of the strategic decomposition of 'Make section of track available'. The operations associated with ensuring acceptable conditions, under the strategy embodied in this analysis, require confirmation from engineers and other staff that satisfactory states pertain. It would be possible to install a signal to be sent by the engineers concerned, but, for reasons of safety, it is advisable always to supplement this with a direct communication and for the controller to enter this value on the control screen, together with the authority given. Thus, even when conditions allow for automation of an IHO, there are often sound operating grounds for retaining human intervention.

Conclusions

In this chapter, we have presented a revision of the SGT method, an extension of hierarchical task analysis developed for the purpose of supporting the development of interfaces. The method involves the decomposition of tasks to a point where information handling operations (IHOs) are recognised, that is, operations in which information-based intervention becomes mandatory for continued system operation. At this point in the analysis, the method necessitates the elicitation, through negotiation with clients, designers and so forth, of one or more strategies for implementing each IHO. These strategies are then further decomposed until a point is reached where operations can be redescribed as sub-goal templates (SGTs), that is, stereotypical operation categories that capture the vast majority of operator activities regardless of overall goal, technology or performance context. Each SGT (and at a lower level, each task element, that is, specific type of SGT) has associated with it a set of unique information requirements, that is, the information that, regardless of chosen implementation method, must be provided to allow the activity to be performed at the interface.

One of the functions of this chapter has been to update the SGT method so that it is no longer geared specifically to process control domains. There are important differences in the development of process-control and other interactive systems that need to be recognised by designers. Perhaps most significant among these is the apparent non-deterministic nature of many process-control systems. Systems that embody complex processes such as heat-exchangers have a nasty tendency to behave in an unpredictable and unstable fashion, making the role of the operator in monitoring and adjusting the system a critical one. Despite this important distinction between process-control and other interactive systems, we believe there is much to be gained by delivering SGTs as a single coherent method of task analysis. Moreover, the embedding of hidden but

proactive devices within information systems such as web browsers (e.g., avatars and other forms of interface agent) means that interactive systems are becoming, at least in perceived interaction style, increasingly non-deterministic. Thus, the concerns of process-control domains increasingly apply to other domains of interactive systems development.

We have endeavoured to provide a method that has the qualities of simplicity, flexibility, explicitness and utility. Good design provides individual operations support through interface design, as well as addressing collaboration, training, job design, personnel selection, fault diagnosis, maintenance, and health and safety issues. Separating these perspectives into independent design issues is a bad thing. Yet, the complexity of an integrated understanding is very challenging. Good design processes involve many perspectives and contributors, all of whom have to be able to share an understanding of methods, tools and their outputs. Therefore, good design support is also minimal design support. The beauty of HTA is its simplicity and naturalness, and we have endeavoured to continue this tradition in the SGT method. Task analysis will always be a skilled activity, but the SGT method endeavours to facilitate the acquisition and execution of analytic skills by providing a clear *modus operandi* for pursuit of task analysis within an ongoing design process with minimal learning overheads and an explicit statement of outcomes for designers.

At the same time, good design is creative – designers should not be told what to do by task analysts. Good task analyses do not, as some have argued, condemn designers to the limitations of previous approaches to implementing operations, but good task analyses are not always so easy to conduct. The SGT method endeavours to ensure good practice in task analysis, recognising that task analysis is a process of negotiation between analyst and engineer (or client, or designer, or user group), not a process of capturing an absolute standard for implementing a set of operations. We believe that there is a, currently unrecognised, danger of trying too hard to drive the design of systems by user-centred considerations. A danger of focussing too much on user operations is that it may undermine the need to drive design by a clear functional specification of objectives and constraints (see Vicente, 1999 for a similar argument regarding the importance of focussing upon the functional status of plant in designing support for operator tasks). The SGT method is explicit about the points at which it is neutral to implementation and the points at which it must lose its neutrality. The key to successful application of the method is the recognition that, regardless of the domain of application, the engineering of the system should drive the design of human-system interactions, and not the converse. Only then will designers take on board a complete assessment of the needs of human users in the specification of complex interactive systems.

References

- Annett, J. and Duncan. K. D. (1967) Task Analysis and Training Design. Occupational Psychology, 41, pp. 211-21.
- Annett, J., Duncan, K. D., Stammers R. B. and Gray. M. J. (1971) Task Analysis. London: HMSO.
- Benyon, D. 1992, The role of task analysis in systems design, Interacting with Computers, 4(1), 102-123.

- Diaper, D. (2001). Task analysis for knowledge descriptions: A requiem for a method. Behaviour and Information Technology, 20, pp.199-212.
- Dillon, A., Sweeney, M., & Maguire, M. (1993). A survey of usability engineering with the European IT industry: Current practice and needs. In J.L. Alty, D. Diaper & S. Guest (Eds.), People and Computers VIII, pp. 81-94, Cambridge: Cambridge University Press.
- Duncan, K.D. (1972). Strategies for analysis of the task. In Strategies for Programmed Instruction: An Educational Technology (Ed. J Hartley). London: Butterworth.
- Duncan, K.D. (1974) Analytical Techniques in Training Design. In Edwards, E. and Leeds, F. P. (eds) The Human Operator and Process Control. London: Taylor and Francis, pp. 283-320.
- Lansdale, M.W. & Ormerod, T.C. (1994). Understanding interfaces: A Handbook of Human-Computer Interaction. London: Academic Press.
- Ormerod, T. C. (2000). Using Task Analysis as a Primary Design Method: The SGT Approach. In J. M. C. Schraagen, S.F. Chipman, & V.L. Shalin (Eds.), Cognitive Task Analysis Mahwah, NJ: Lawrence Erlbaum Associates, Inc. pp. 181-200.
- Ormerod, T. C., Richardson, J., & Shepherd, A. (1998). Enhancing the usability of a task analysis method: A notation and environment for requirements specification. Ergonomics, 41, 1642-1663.
- Parker, K. (2001). An approach to requirements analysis for decision support systems International Journal of Human-Computer Studies, Vol. 55, No. 4, pp. 423-433
- Payne, S. J. and T. R. G. Green (1986). "Task-Action Grammars: A Model of the Mental Representation of Task Languages." Human-Computer Interaction 2: 93-133.
- Rubinstein, E. and Mason, J.F. (1979). An analysis of Three Mile Island. The accident that shouldn't have happened. IEEE Spectrum, November, 1979, pp. 33-42.
- Shepherd, A. (2001). Hierarchical Task Analysis. London: Taylor and Francis.
- Shepherd, A. (1993). An approach to information requirements specification for process control tasks. Ergonomics, 36 807 - 819.
- Simon, H. A. (1981). The sciences of the artificial, second edition. Cambridge MA, MIT Press.
- Vicente, K. J. (1999). "Wanted: psychologically relevant, device- and event-dependent work analysis techniques." Interacting with Computers, 11: 237-254.

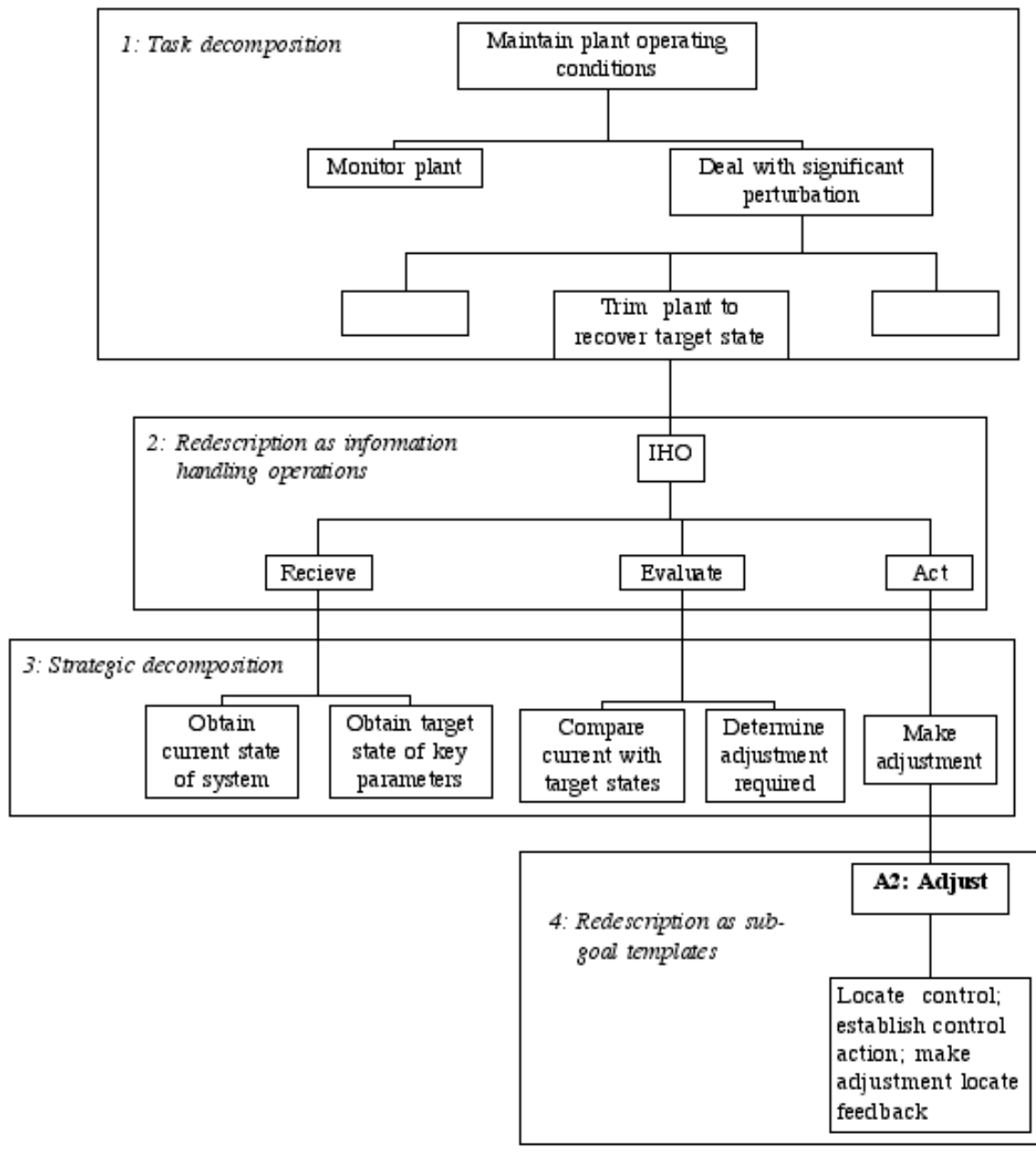


Figure 1. The hierarchy of information handling operations, showing 1) task decomposition, 2) redescription as information handling operations (IHOs), 3) strategic decomposition, and 4) redescription as sub-goal templates with associated information requirements.

Table 1. The relationship between monitoring and fault handling tasks.

Tasks	Explanations
0 Maintain plant operating conditions <i>plan 0: Throughout — 1. If perturbation is judged significant — 2.</i>	This is a common requirement in all system supervisory tasks.
1 Monitor plant <i>plan 1: Do 1 then, if there is a deviation from target, do 2.</i>	
1.1. Monitor specific parameters <i>plan 1.1: Do 1 & 2. Then do 3.</i>	
1.1.1. Obtain current state of key parameters	These 3 activities are the essence of obtaining information in monitoring tasks.
1.1.2. Obtain target state of key parameters	
1.1.3. Compare current with target states	
1.2. Determine whether perturbations are significant	Some perturbations result from temporary blips and will recover if left.
2 Deal with significant perturbation <i>plan 2: Do 1. Then, according to plan, 2, 3, 4, 5 & 6.</i>	Diagnosis and, at least one other activity are required to deal with the problem, finishing with 8.
2.1. Diagnose and plan appropriate remedial action	This may involve identifying a cause, but primarily it is concerned with determining how to proceed to maintain the system state.
2.2. Trim plant to recover target state	It may be sufficient to 'fine-tune' the system.
2.3. Put standby equipment on line	It may be sufficient to 'switch to standby equipment.
2.4. Compensate for off-specification conditions	Production may be optimised by adjusting running rates elsewhere.
2.5. Isolate problem	The problem may be isolated such that other aspects may proceed in the best manner possible.
2.6. Shut-down system	A complete shut-down may be required to make the system safe for repair.
2.7. Rectify fault	Usually repair or replacement.
2.8. Recover target operating state	When a repair is complete, operating conditions must be recovered.

Table 2. Sub-goal templates, specific task elements, and information requirements (IRs)

SGTs	Task elements	Context for assigning SGT and task element	Information requirements
Act		Perform as part of a procedure or subsequent to a decision made about changing the system	Action points and order; Current, alternative & target states; pre-conditions, outcomes, dependencies; halting, recovery indicators.
	A1 activate	Make sub-unit operational - switch from 'off' to 'on'	Temporal/stage progression, outcome activation level
	A2 adjust	Regulate the rate of operation of a unit maintaining 'on' state	Rate of state change
	A3 deactivate	Make sub-unit non-operational - switch from 'on' to 'off'	Cessation descriptor
Exchange		To fulfil a recording requirement. To obtain or deliver operating value	Indication of item to be exchanged; Channel for confirmation
	E1 Enter	Record a value in a specified location	Information range (continuous, discrete)
	E2 Extract	Obtain a value of a specified parameter	Location of record for storage and retrieval; prompt for operator
Navigate		To move to an informational state for exchange, action or monitoring	System/state structure, current relative location
	N1 locate	Find the location of a target value or control	Target information, end location relative to start
	N2 move	Go to a given location and search it	Target location, directional descriptor
	N3 explore	Browse through a set of locations and values	Current/next/previous item categories
Monitor		To be aware of system states that determine need for navigation, exchange and action	Relevant items to monitor; record of when actions were taken; elapsed time from action to the present
	M1 monitor to detect deviance	Routinely compare system state against target state to determine need for action	Normal parameters for comparison
	M2 monitor to anticipate cue	Compare system state against target state to determine readiness for known action	Anticipated level
	M3 monitor transition	Routinely compare rate of change during state transition	Template against which to compare observed parameters

Table 3. Sequencing elements used to construct plans containing SGT elements.

Code	Type	Syntax
S1	Fixed sequence	S1 - Do X....
S2	Contingent sequence	S2 - If (c) then do X.... If not (c) then do Y....
S3	Parallel sequence	S3 Do together X.... Y....
S4	Free sequence	S4 In any order do X.... Y....

Table 4: Decomposition of main railway control tasks, showing their sequencing as two parallel activities (S3 sequence) and one nested contingent activity (S2 sequence).

Tasks		Explanation
0	Supervise Railway system	
☐	<i>plan 0: S3 Do together 1 and 2; S2: If problems are identified through system monitoring or if incident is notified do3, else continue.</i>	Most railways operate on a daily cycle, with services commencing in the early hours, and then increase in frequency to offer a full timetable. Towards the end of the day, the service is reduced according to the timetable. As track clears, engineering hours are scheduled.
☐	1 Provide Services	Controllers bring trains into service at the start of the day and introduce further trains according to the timetable. As trains are no longer required, they are removed from service.
☐	2 Maintain Best Service	When trains are introduced into service, they must be controlled to comply with the timetable, without risk to services, equipment, staff or the public.
☐☐	3 Deal with System Problems	If problems arise, they must be dealt with promptly and effectively in order to maintain safety and service standards.

Table 5: Decomposition of activities involved in ‘Provide services’ and their subsequent redescription as information handling operations (IHOs).

	Tasks	Explanation
1	Provide Services	
	<i>plan 1: S1 At start of day/shift - 1.1. S2, If start of day - 1.2. If not - continue; S1 Throughout day - 1.3. S2 If timetable alerts -1.4 If not - continue; S1 At end of day- 1.5.</i>	Service provision entails some activities at the start of the day, then a series of activities according to the timetable.
	<u>Sub-task: IHOs:</u>	
	1.1. Revise timetables and rotas according to information	Various documents must be read to establish any last minute changes to the timetable.
	1.2. Carry out overall system start-up checks	Ensure that all systems are operational and that there are no significant resource deficiencies that would prevent services from starting.
	1.3. Monitor timetable	The timetable must be monitored throughout the day, because it will prompt many of the necessary control interventions.
	1.4. Make section of track available	When prompted by the timetable, steps are taken to ensure that track sections may be safely used.
	1.5. Clear line section at end of day	When the last train has passed through a section of track at the end of the day, ensure that the track can be released for safe engineering work.

Table 6: Strategic decomposition and assignment of SGTs to ‘Revise timetables and rotas according to information’.

Task element	Explanation	SGT
1.1.Revise timetables and rotas according to pertinent information <i>plan S3Do together 1.1.1 and 1.1.2; S2 If necessary do 1.1. 3. If not - continue.</i>	Timetables and rotas are typically published in advance, but should be revised at the start of each day.	
1.1.1. Monitor sources of information <i>plan S4: In any order 1.1.1: do 1.1.1.1 -> 1.1.1.3.</i>		
1.1.1.1. Inspect log book 1	Logged information from previous shifts may affect current activities.	M1
1.1.1.2. Inspect traffic 2 circulars and line notices	Special timetabling modifications may be prompted and communicated through traffic circulars	M1
1.1.1.3. Examine nightly 3 engineering notices & safety arrangements	Work during engineering hours may affect the availability of sections of track, thereby interfering with the timetable.	M1
1.1.2. Evaluate information and plan revisions		
1.1.3. Make adjustments to control instructions <i>plan S4 In any order 1.1.3: do 1.1.3.1 -> 1.1.3.3.</i>		
1.1.3.1. Make changes 1 to timetable	The timetable must be revised according to planned revisions.	A2
1.1.3.2. Make changes 2 to staff rota	Staff rotas must be updated.	A2
□ 1.1.3.3. Make changes 3 to working practices	Special arrangements may need to be communicated regarding safe operating.	A2

Table 7: Strategic decomposition and assignment of SGTs to the sub-task ‘Make section of track available’.

	Task element	Explanation	SGT
	1.3. Make section of track available		
<input type="checkbox"/>	<i>plan 1.3: S3 Do together 1.3.3 and S1 On timetable prompts do 1.3.1; S1 When conditions are acceptable for starting up the section do 1.3.2.</i>		
<input type="checkbox"/>	1.3.1. Ensure railway conditions are acceptable for section start-up		
<input type="checkbox"/>	<i>plan S4 In any order do 1.3.1: 1.3.1.1 -> 1.3.1.3.</i>		
<input type="checkbox"/>	1.3.1.1. Establish whether engineers are clear from overnight possession	Safety practices require controllers to confirm that engineering work has been completed.	E2
<input type="checkbox"/>	1.3.1.2. Establish whether sub-station equipment is available	There must be confirmation that the section of track can be powered up.	E2
<input type="checkbox"/>	1.3.1.3. Establish whether overnight signals maintenance has been cleared and tested	Signalling systems must be shown to be fully operational. This may mean that a formal test procedure is implemented.	E2
<input type="checkbox"/>	1.3.2. Power up section of line		
<input type="checkbox"/>	<i>plan Do 1.3.2.1 then .1.3.2.2 1.3.2: then 1.3.2.3.</i>		
<input type="checkbox"/>	1.3.2.1. Receive line clear/line safe message	A communication will be sent from the track to indicate that the section of line is clear. The controller must await this cue for further action.	M2
<input type="checkbox"/>	1.3.2.2. Reset tunnel telephone lines		A1
<input type="checkbox"/>	1.3.2.3. Recharge current rail section		A1
<input type="checkbox"/>	1.3.3. Deal with problems <input type="checkbox"/>	Problems arising from operations will be dealt with by direct communication with engineering staff.	