A Formal Method for Integrated System Design:

How to Incorporate Cognitive Engineering Principles in a Systems Engineering Method

Michael Feary
MS262-4
NASA Ames
Research Center,
Moffett Field,
CA, USA
94035-1000

Lance Sherry
Honeywell
Inc./Rand Corp.
1200 S.Hayes St
Arlington, VA,
USA
22202-5050

Everett Palmer
MS262-4
NASA Ames
Research
Center,Moffett
Field,CA, USA
94035-1000

Peter Polson
Department of
Psychology
Campus Box
345
University of
Colorado

Summary

This paper describes a formal methodology (referred to as the Situation – Goal – Behavior (SGB) Methodology) that has been used in the design of software logic based avionics systems. It is expanded to incorporate some cognitive engineering guidelines and principles for use in design of complex systems. The SGB method is referred to as *formal* because the tool associated with the method has algorithms that are used to assist requirements specification. The purpose of this paper is to describe the application of the method as a framework for integrating the requirements for interface design, training, and procedure design within the system engineering method. This paper first describes the method, how it can be used, and then moves on to case studies which have shown the method to be useful for incorporating information requirements in the design process.

The Situation – Goal – Behavior (SGB) Method

The Situation – Goal – Behavior (SGB) method is a means of organizing and providing a common framework for all of the necessary groups involved in the design process of software based systems. The SGB method is similar to many decision-making behavior modeling environments, but with a different representation and approach to the problem of organizing decision-making software. The primary focus of this paper is to integrate the use of cognitive engineering guidelines and principles within a systems engineering method early in the design cycle.

The main benefit of the SGB table format from a cognitive engineering perspective is the ability to use the table as a type of description language for discussing design implementation across the engineering, training, operational (flight standards, etc.), interface, and procedure design groups. This early integration is necessary for the development of systems which are easier to train, learn and operate. (Billings, 1996). The ultimate goal of the use of the methodology would be to have a design change implemented based on a training requirement, for the purposes of reducing training, while early in the design process.

Proper task specification and the communication of the goals of the system in automated systems is crucial in complex, automated system development (Rasmussen, 1994). It is, therefore, the intent of the SGB method to involve the users in specifying the mission, or the tasks that the system will need to accomplish. If the task specification is successful and complete, the design engineering task of making the system function properly should be straightforward. Additionally, the training and interface development should be task-oriented as well, if it were possible to put the task specification in a framework that is accessible to the operational community (users, training developers, interface developers, etc.) in the design cycle the result should be a more “user-centered” system.

How the SGB Method Works

The SGB method uses a tabular representation of a systems engineering model which represents the decision logic of an automated system (Table 1). The table is composed of Situation Inputs and States, Situations, Goals, Behaviors, and Behavior Outputs.

In a typical design cycle for a large complex system, the customer requirements for a new system would ideally be presented by the customer or a representative of the customer from the operational community (users, training, and procedure development representatives), to provide the desired Goals and Behaviors for the system. As the Goals and Behaviors for the system are defined, the engineering design team can provide the inputs required by the system to determine the situation and perform the correct behavior. The *formal* systems engineering portion of the method includes some algorithms to ensure that all possible combinations of inputs are covered, and that no two combinations are the same, which would lead to a non-deterministic system. Typically, this process will iterate several times, with engineers asking the customer what behavior they might prefer for situations that were not defined originally. To initially demonstrate the method and the use of the table, an example SGB table for a very simple digital wristwatch which has one button and an 8 character wide display is provided (Table 2).

Table 1. SGB Table. The X-axis of the table shows the Situation, Goal and Behavior descriptors, and the Y-axis shows the Situation input and input-state names.

		Goal 1		Goal 2
		Situation A	Situation B	Situation C
Input I	state a			
	state b			
Input II	state a			
	state b			
	state c			
		Behavior Description 1		
Output	state a			
	State b			

Table 2. Example of SGB table usage

		Display Date		Display Time	
		Change to date mode	Stay in date mode	Change to time mode	Stay in time mode
Button	Pushed	true		true	
	Not pushed		true		true
Current goal	Time mode	true			true
	Date mode		true	true	
		Display Date		Display Time	
Display	Date	true			
	Time			True	
Calculate	p:Date	date			
	m:Time			time	

Situation Inputs and States
Inputs and Input states are developed by the design engineers to try to meet the Goals specified by the customer. The Inputs may represent a detected action by the user, an input that is automatically detected by the system via a sensor, the result of calculation elsewhere in the system or other means. Input states are used to quantify the input to enable the automation to interpret the input value. In Table 1, Inputs I and II are represented by two and three states respectively, however in actual tables there is no limit to the number of states per input. An example of this will be shown in the following section.

How Inputs are Defined. As a cognitive engineering requirement, these inputs should be meaningful to the operational community. This is easily met in a simple system, such as watch, by using the button label. However, it can be a challenge with a larger, complex system with multiple sensor inputs, and results of calculations elsewhere in the system which have no physical attributes to provide inherently operationally meaningful names.

If the inputs can be given operationally meaningful names, then the interface design groups can design feedback to the user on the accurate goal of the system. The example in Table 2 is a simple illustrative example and because it does not have much software decision-making logic, the input names are intuitive and operationally meaningful. However in larger, more complex systems, inputs may not have names that are easy to understand and operational, especially if the input refers to a portion of computer code, for which names may be arbitrary. It is important to make the inputs operationally meaningful so that the situations resulting from the combinations of inputs are operationally meaningful. If the situations are operationally meaningful, users should be able to answer the questions “Why is it doing that?”, and “What is it going to do next?”. Unfortunately, in many of today’s systems, these questions are asked all too frequently.

Another requirement is to identify the inputs which are visible to the operator. Ideally, all inputs would be visible to the operator, but the inputs which are not must be justified and agreed upon by the different groups. Additionally, the situations which are defined by the inputs which are not visible to the operator will have to be accommodated. These situations will naturally lead to confusion, because there is a difference between the information the operator is using and the information the system is using.

Formal analyses (e.g. model checkers) can perform an evaluation of the situations defined in the system that the user can see (via controls and displays) versus those situations which the user cannot see because they are based on system information which is not displayed or requested by the user (operator).

Situations

Situations are a description of the combination of inputs intended to make the table readable and easily understood by the customer. In Table 1, Situation A can be seen as a description of the combination of Inputs I and II. The Situation descriptions can be very valuable to the training, interface and procedure development communities later as they define the tasks of the system, and as the system is reorganized. During the design cycle the situation description becomes the foundation for organizing the hierarchy of goals.

How Situations are Defined. The Situations in the table provide a description of each of the events that can occur in the system. In the watch example, the system needs to be able to cope with at least two situations to display the date. First, if the system is currently displaying the time, then pushing the button will display the date, but the system also needs to know the current state so that pushing the button changes the situation, otherwise pushing the button would cause the watch to always switch to date mode or always switch to time mode.

Goals

Goals refer to the mission objectives that the customer will be responsible for. In complex systems, goal descriptions are a hierarchical grouping of the situations by the operational community and are representative of the task to be accomplished. In this way, Goals are used to organize what the design team thinks is most important for the user to know.

How Goals are Defined. Goals or Behaviors are generally the first requirement given by a customer. In the example in Table 2, the customer for this simple wristwatch has specified two goals and behaviors, Display Date and Display Time. In this simple example, the Goal and the Behavior are the same, however the usefulness of the Goal representation becomes evident in the design of larger and more complex systems, when decisions need to be made about what priority should be given to different system behaviors. The interaction can be seen between the Goal and the Behavior in the form of an ”m:” preceding the “Time” behavior output. This “m:” refers to a sub-mission, or lower level table, which describes functions that the design team has decided do not need to be immediately visible to the operator. In this case the “m:” refers to a table which describes how to set the time on the watch.

Behaviors

The Behaviors refer to how the system will perform to meet to associated Goal. The customer defines the required behavior for the system, and the engineering portion of the design team uses the outputs and output states in a similar fashion to the inputs and input states to meet the performance objectives. The Behaviors are natural language descriptions that are written by the operational group of the design team and should closely match the goal.

How Behaviors are Defined. The behaviors are used in conjunction with the Goals to organize the system around cognitive engineering principles. When the desired behaviors have been specified by the customer, the design team can set about organizing the behaviors based on importance and level of interaction required by the operator. Those behaviors which the operator can have little effect on in the system are candidates for being made less visible and those behaviors which the operator needs to know about to operate the system correctly need to be made more visible. This is accomplished through the use of hierarchy with multiple tables, as shown in Table 2 with the “m:” preceding a behavior output. In the method the highest level tables are designed to be the most visible. This process needs to be carried out with the training and interface design representatives.

The table does not automatically provide the best information requirements for a particular system. These decisions require operational experts within the design team. What the tool does is allow the different members of the design team to communicate with each other in a format that forces them to be precise about what they intend. Later design decisions and environmental requirements may force modifications to the information requirements, but having the decisions documented in this format allows the changes to be made easily. Once all of the information requirements for the display have been identified, the interface design team can apply other human factors guidelines to aid in the design of the display. An example of this is the concept of *label following* which has shown that users will tend to respond to a task by using an interface object that most closely resembles the task, even if it is not the correct action. (Franzke, 1995).

Another cognitive engineering practice is to match the input devices to the behaviors to avoid functional overload. A functionally overloaded input device is one for which the behavior of the input is dependent upon the state of the system at the time the input device is used. This functional overload may easily become cognitive overload and error prone for the operator, if that operator has poor feedback, is improperly trained, has memory lapses from fatigue, etc. (Sherry et al., 2001)

Behavior Outputs

Combinations of behavior outputs are used by the engineers to meet the requested behavior by the customer, and give feedback to the user.

How Behavior Outputs are Defined. From a systems engineering perspective, the output states can be given different classifications within the SGB tool, examples of these include assignment to another table (shown by the “m:” preceding a Behavior Output) or, assignment of primitive functions (e.g. an equation) where the table stops and leads to a calculation that can be made by the automation. The primitive function is represented by the “p:” in the Date behavior output in Table 2. In this case, it refers to the watch calculating the date.

Summary of Cognitive Engineering Guidelines and principles using the method

- Develop, identify and organize Goals for the level of interaction
- Develop operationally meaningful inputs
- Identify/ensure inputs are visible to the operator
- Match the Goals/behaviors with display design
- Match the input devices to the behaviors to avoid functional overload

Application to aviation automation

Aircraft automation, particularly the automation surrounding vertical flight guidance, has been cited as an area of training difficulty and a source of confusion during operation. A number of incidents have been attributed to a lack of crew understanding of what the automation was doing.

As part of a NASA program, the pieces of the SGB method were evaluated using aircraft automation on a modern, commercial transport category aircraft very similar to the Boeing MD-11 aircraft. To determine where the method should be focused, a survey was distributed to Boeing MD-11 line pilots to assess where pilots thought they were having difficulty, and where they would like the most help with the automation. (Feary et al., 1998) More than 75% of the pilots surveyed felt that aspects of the Vertical Flight Guidance system were trained inadequately, including the FMS Speed Logic, PROF (Vertical Navigation Mode), and the interpretation of the Flight Mode Annunciator (FMA). The SGB Methodology was used to design a new interface, procedures and training material for the Vertical Flight Guidance system. These changes were evaluated experimentally (Feary et al., 1998), and examples from these experiments should demonstrate the utility of the method for some applications.

Table 3 shows a small fraction of the Vertical Guidance for a modern commercial air transport. (Honeywell Cockpit Pilot's Guide, 1994). The Goals, Situations, inputs, input states, and behaviors are representative of those used in the larger table, although the inputs have been reduced to correspond with the different behaviors depicted. Specifically, Table 3 shows 3 behaviors used in the descent phase of flight.

Before discussing the meaning of Table 3, some background domain information needs to be defined. The input state Reference altitude refers to the altitude specified by the pilot using a knob on the Autopilot control panel, generally referred to as the Mode Control Panel (MCP).

Starting with the input column, the input *VG Type* refers to the level of automation being used.

Vertical Navigation (VNAV)/ Profile(Prof) mode refers to the names of the most highly automated modes in current aircraft. In *VNAV/Prof* (the names differ for different aircraft manufacturers), speed and altitude targets are specified in the Flight management Computer (FMC) and flown automatically. Airmass – VNAV/Prof refers to slightly less automated mode for which the speed targets are automatically generated by the FMC, but the altitude targets are not. Finally the AFS mode refers to one of the least automated modes, for which the pilot selects both speed and altitude targets.

Table 3. Example of SGB table of a vertical guidance system. In the table, “1” refers to a true state.

Goals		Airmass	Descent	Late	Descent	Descent Path	Overspeed
Inputs	Situations/ Input States	Aircraft is Descending (without both Prof and FMS Speeds)	Aircraft is descendin g early of D/A Path and Prof/FMS speed engaged	A/C is level late of the D/A Path level at the ref. Alt and the ref. alt	Aircraft is descending late of D/A Path and Prof/FMS speed engaged	Aircraft exceeds speed tolerance while descending on D/A path	Aircraft is level with a speed that exceeds the speed tolerance when ref. Alt is lowered and a/c captures D/A path
VG Type Altitude	VNAV /Prof			1	1	1	1
	Airmass – VNAV/Prof	1	1				
	Airmass - AFS						
Aircraft Altitude	Above distance Referenced D/A path			1	1		
	below distance Referenced D/A path						
Aircraft Speed	Overspeed for D/A path					1	1
	Within speed tolerance for D/A path	1	1	1	1		
Aircraft Altitude	Within D/A Path capture region						
	Not Within D/A Path capture region	1	1	1	1		
Reference Altitude	Has not changed						
	Has changed		1	1			1
Behaviors		Airmass Descent to the D/A path D/A path speed	Referenced recapture using the descent profile	Airmass Descent the D/A the late profile	Referenced to recapture path using descent speed	Airmass Descent D/A path path descent	Referenced around the at the D/A speed profile
Altitude Target	M:Climb/Cruise						
	M:Descent/App roach	Descent/ Altitude	Approach Target	Descent/ Altitude	ApproachTar get	Descent/ Altitude	ApproachTarg et
Speed Target	M:Late descent			Late Speed	Descent Target		
	M: Descent/Approach					Descent/ Speed	Approach Target
	M: Airmass Descent	Airmass Speed	Descent Target				
	P: engine-out						
Speed/ Altitude Control Mode	P: THRUST |HOLD						
	P: PITCH ||IDLE	PITCH |	| IDLE	PITCH|	| IDLE	PITCH |	| IDLE
	M: D/A DESCENT SPEED/Altitude						

The Speed/Altitude Control Mode refers to how the automation will control the airplane to meet the guidance requirements. For example an aircraft may pitch for a certain rate of climb or descent, or the aircraft may pitch for a particular airspeed. In another, unique case the automation may pitch the aircraft to remain on fixed earth-referenced path referred to as the Descent/Approach(D/A). The D/A path takes into account Air Traffic Control (ATC) speed and altitude restrictions and computes a descent path such that the engines should remain at idle thrust, resulting in the most fuel efficient descent possible should the aircraft stay on the path.

The behaviors depicted in the table exist to deal with exception cases which would cause an aircraft to deviate from the D/A path. The Airmass Descent case accounts for scenarios during which the FMC has not computed a D/A path, this may happen if the pilot does not enter a destination in the FMC, or for the short time while the FMC is computing the path. The Late descent behavior accounts for a scenario for which the aircraft is not allowed to descend until passing the optimum descent path (due to ATC constraints), or a case in which the aircraft has too much energy to stay on the optimum descent path (an unforecast tailwind would cause this). The Path Descent Overspeed case accounts for a scenario similar to the Late Descent case, however in this case the aircraft the aircraft starts on the path, but once reaching a certain speed threshold faster than the assigned speed target, the aircraft flies away from the D/A path at a predetermined speed.

The behaviors shown in Table 3 are all different, however the display in many modern commercial aircraft are very similar for the different behaviors, and in at least one they are exactly the same presentation to the pilot with little to no displays available to differentiate between them.

Case Study 1: The Behavior based Flight Mode Annunciator (FMA)

The number of distinguishable display events should be directly related to the number of different Goals/Behaviors of the system. This is required for users to be able to answer the question “What is it doing now?”, which is a question frequently asked by users of complex, time-critical and highly dynamic systems (e.g. aircraft automation) today.

That simple guideline was used to evaluate an automation display of a modern transport category airplane, and several cases of non-compliance were found. To evaluate the impact of this non-compliance on one automation display, referred to as the FMA, Feary et al., (1998) conducted a study with a modified display in a certified simulator with actual airline pilots who were current and very experienced in the type of aircraft being evaluated.

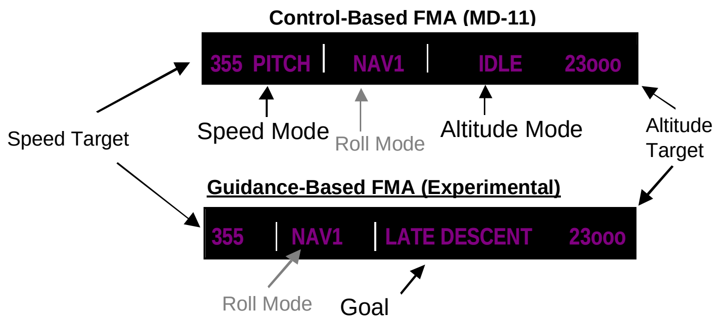

Figure 1. Control vs. Guidance-Based FMA

The experiment involved a change in the organization and wording of the Flight Mode Annunciator display, as it is seen in the cockpit of a modern, transport category aircraft. To provide more conceptual system information on the display, the new display annunciated the Goals of the system. These Goals were based on the Behaviors in the SGB representation. The current display was compared to the new display (Fig.2) and, combined with a training package, distributed to participants in both conditions (to equalize the group). The displays were tested by 27 current line pilots, with at least 1 year of experience on the MD-11 (an airplane with the same display and vertical guidance to the experimental display), flying a short flight in a Fixed Base Simulator. The pilots in the new display condition showed significantly less errors ($p>.03$) for the experimental scenario.

Case Study 2: Modification of a functionally overloaded Control Device

An example of functional overloading was observed on the MCP of a modern, transport aircraft. The current MCP for the aircraft under investigation uses one button *PROF/VNAV* to engage a higher level of automation, and depending upon the situation of the automation at the time, pushing the button may result in one of at least 13 different behaviors. It is interesting to note that in at least one aircraft, the FMA displays the letters PROF to refer to descending on the D/A path, while at the same time using a button labeled PROF to refer to changing between more and less automated modes of flight. By clearly defining the behaviors in the system, the complete SGB tables of the vertical guidance system (not presented here) show this contradiction clearly. Additionally, using the table as a starting point for further analysis, it can be seen that descending on the D/A path is fundamentally different than any of the other modes in that it has a fixed earth-reference point. A proposed solution is shown in Figure 2, resulting in the simple addition of a DES PATH button to refer to the unique mode, and some modification of the logic to restrict automatic switching to other descent modes. (Sherry et al., 2001)

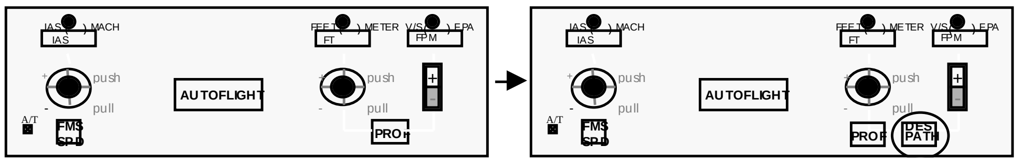

Figure 2. Comparison of modern Autopilot Control Panel with MCP designed according to the cognitive engineering principles. A separate input device (DES PATH button) provides the option to arm the capture and tracking of the FMS optimum path. This button has only one behavior – to capture and track the path.

Current and Future Work

At the time of writing an initiative has begun to link the Situation Goal – Behavior - Method to task analytic techniques to evaluate usage and procedure techniques. This work will consist of building a model of a pilot's conception of a portion of the Vertical Guidance System and making a formal comparison with Situation – Goal – Behavior model of the same system. It is hoped that the task analytic model will reveal areas of possible pilot error, and be able to predict areas of the system for which operators lack conceptual understanding.

New versions of the tool are being developed and evaluated, and parts of the tool are publicly available. Requests can be sent to mfeary@mail.arc.nasa.gov or Lance.Sherry@cas.honeywell.com.

References

Billings, C., 1997, *Human-Centered Aviation Automation*, Mahwah, NJ, Lawrence Erlbaum Associates.

Feary, M., Sherry, L., Palmer, E., and Polson, P (1998) Aiding Vertical Guidance Understanding. NASA Technical Memorandum 1998-112217. NASA Ames Research Center, Moffett Field, CA, USA.

Franzke, M. (1995), Turning research into practice: characteristics of display-based interaction, in Proc. CHI'95 Human Factors in Computer Systems, ACM, pp. 421-428

Honeywell (1994). *MD-11 Cockpit Pilot's Guide* (Revision 1). Honeywell Document C29-1101-01-01. Phoenix, AZ., USA

Irving, S., Polson, P., and Irving, J. (1994) A GOMS Analysis of the Advanced Automated Cockpit. In Proceedings of CHI 94 conference. Boston, MA.

NTSB (1999) Safety Recommendation. A-99-39 through 44.

Rasmussen, J. (1994) *Cognitive Systems Engineering.* New York, USA: John Wiley and Sons, Inc.

Sherry, L., Feary, M., Polson, P., Mumaw, R., and Palmer, E. (2001) A Cognitive Engineering Analysis of the Vertical Navigation (VNAV) Function. NASA Technical Memorandum 2001-210915. NASA Ames Research Center, Moffett Field, CA, USA.

Vicente, K. (1999) *Cognitive Work Analysis.* Lawrence Erlbaum Associates.Mahwah, NJ.