COMMAND AND CONTROL IN AXIOMATIC DESIGN THEORY: ITS ROLE AND PLACEMENT IN THE SYSTEM ARCHITECTURE

Jason D. Hintersteiner, Derrick Tate

Department of Mechanical Engineering
Massachusetts Institute of Technology
77 Massachusetts Avenue, Room 31-261
Cambridge, MA 02139, USA
jdhinter@mit.edu, dtate@alum.mit.edu

ABSTRACT

This paper describes how system command and control may be integrated into a system architecture. There are two key questions addressed: First, how does command and control fit within the decomposition—on which branches and at what levels are its functional requirements (FRs) and design parameters (DPs)? Second, what are the functions of command and control algorithms at different levels of the design hierarchy—what are their inputs and outputs? Our conclusions illustrate that the decisions about hardware components are made as part of the processes to which they belong; thus, the hardware components (DPs) that are controlled are distributed among several branches of the design hierarchy. Additionally, each module within a process has local software algorithms associated with it for its control. At higher levels of the design hierarchy, there are other algorithms responsible for coordinating the modules below it. Thus, software control algorithms exist at multiple levels of the design hierarchy. By properly structuring the design hierarchy in this fashion, design choices can be made about system command and control in a non-iterative manner and remain consistent throughout multiple levels of the design hierarchy.

KEYWORDS

Axiomatic Design, Command and Control Algorithms, System Design

1 Introduction

This paper details the incorporation of command and control within a system architecture. Here, system architecture is defined as a complete description of the functions and components of a system, and it is captured in axiomatic design as sets of functional requirements (FRs), design parameters (DPs), and design matrices (DMs) in a hierarchical arrangement. In particular, it is the authors' view that only by better understanding system command and control can systems be designed which both satisfy the independence axiom and maintain consistency in the design decisions throughout the design hierarchy. While systems can be conceptualized to maintain functional independence at the highest level of system design, it is the design choices made at lower levels that ultimately determine whether or not this vision is realized. By describing the components of command and control requirements and how they fit into a system design from the axiomatic design perspective, we hope to provide designers with the tools to make their lower-level design decisions consistent with their higher-level ones.

Command and control algorithms (CCAs) are found at multiple levels of the design hierarchy, and their FRs and DPs are distributed among different branches. The role of CCAs at different levels of the design hierarchy is discussed in section 3. At higher levels of the design hierarchy, the command aspect of the CCAs is emphasized: sequencing of FRs and scheduling of DPs is performed. The control aspect of command and control, where signals are sent to control specific hardware components, is important at lower levels of the design hierarchy. This aspect is discussed in detail in section 4.

2 Axiomatic Design

Design is defined as the development and selection of a means (design parameters, or DPs) to satisfy objectives (functional requirements, or FRs), subject to constraints. Axiomatic design provides a framework for describing design objects which is consistent for all types of design problems and at all levels of detail. Thus, different designers can quickly understand the relationships between the intended functions of an object and the means by which they are achieved. Additionally, the design axioms provide a rational means for evaluating the quality of proposed designs, and the design process which is used guides designers to consider alternatives at all levels of detail and to makes choices between these alternatives more explicit.

The main concepts of axiomatic design include the following: (1) *domains*, which separate the functional and physical parts of the design; (2) *hierarchies*, which categorizes the progress of a design in the functional and physical domains from a systemic level to more detailed levels; (3) *zigzagging*, which indicates that the decisions made at one level of the hierarchy affect the problem statement at lower levels; and (4) *design axioms*, which dictate that the independence of the functional requirements must be maintained and that the information content (i.e. cost, complexity, etc.) must be minimized, in order to generate a design of good quality. For a more thorough explanation of axiomatic design theory, the reader is referred to [1; 2].

3 System-level branches of the design hierarchy

One of the fundamental assumptions in the development of a system architecture is that the system can be modeled as a series of interacting inputs and outputs. In a general sense, the input-output transformations at the top level of the system design can be broken down into several categories: process functions, transport functions, command and control functions, and support and integration functions, as shown in equation 1. Command and control algorithms, therefore, can be viewed as the embedded logic that controls and coordinates such input-output operations.

$$\begin{bmatrix} \text{Process 1} \\ \text{Process 2} \\ \dots \\ \text{Process } u \\ \text{Provide Throughput} \\ \text{Transport 1} \\ \text{Transport 2} \\ \dots \\ \text{Transport } v \\ \text{Command System} \\ \text{Implement Control} \\ \text{Mech. Integrate Modules} \end{bmatrix} = \begin{bmatrix} X & O & \dots & O & O & O & \dots & O & O & O & O & O \\ O & X & \dots & O & O & O & \dots & O & O & O & O & O \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ O & O & \dots & X & O & O & O & \dots & O & O & O & O \\ X & X & \dots & X & X & O & O & \dots & O & O & O & O \\ ? & ? & \dots & ? & X & X & O & \dots & O & O & O & O \\ ? & ? & \dots & ? & X & ? & X & \dots & O & O & O & O \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ ? & ? & \dots & ? & X & ? & ? & \dots & X & O & O & O \\ X & X & \dots & X & X & X & X & \dots & X & X & O & O \\ X & X & \dots & X & X & X & X & \dots & X & X & X & O \\ X & X & \dots & X & X & X & X & \dots & X & X & X & X \end{bmatrix} \begin{bmatrix} \text{Process Module 1} \\ \text{Process Module 2} \\ \dots \\ \text{Process Module } u \\ \text{Number of Process Modules} \\ \text{Transport Module 1} \\ \text{Transport Module 2} \\ \dots \\ \text{Transport Module } v \\ \text{Command and Control Algorithm} \\ \text{Computer Hardware Configuration} \\ \text{Structural Framework} \end{bmatrix} \quad (1)$$

The nature of a CCA depends on the level being considered in the design hierarchy. The higher-level CCAs (type I) deal with sequencing and scheduling, and the lowest-level CCAs (type III) deal with hardware control. Mid-level CCAs (type II) may perform both sequencing and scheduling as well as hardware control; however, such a CCA is responsible for scheduling DPs at its level and for sequencing sub-FRs at the next, lower level. In summary, the three types of functions that can be performed by a CCA include sequencing of FRs, scheduling of DPs, and control of hardware parameter values. These three types of functions performed by CCAs are summarized in Table 1.

Figure 1 illustrates how the CCAs are represented in a system architecture. Here, planarization step A1 is composed of two specific hardware motions (A11 and A12), but is one of three steps involved in the whole polishing recipe, which dictates that the three steps are implemented in a particular sequence. At an even higher level of control, the command and control algorithm is responsible for selecting the right process recipe which determines that these sub-FRs (that is, steps A1, A2, and A3) make up the process.

In the case where there are multiple sub-modules at one level of the decomposition, one command and control algorithm may be used for this whole set. That is, the algorithms may be integrated into one unit. Furthermore, the level of physical embodiment is not necessarily uniform at any given level of the design decomposition. For

example, one FR may be decomposed into a set of sub-FRs, where some of the DPs are hardware components while others are modules with *their own* sub-processes.

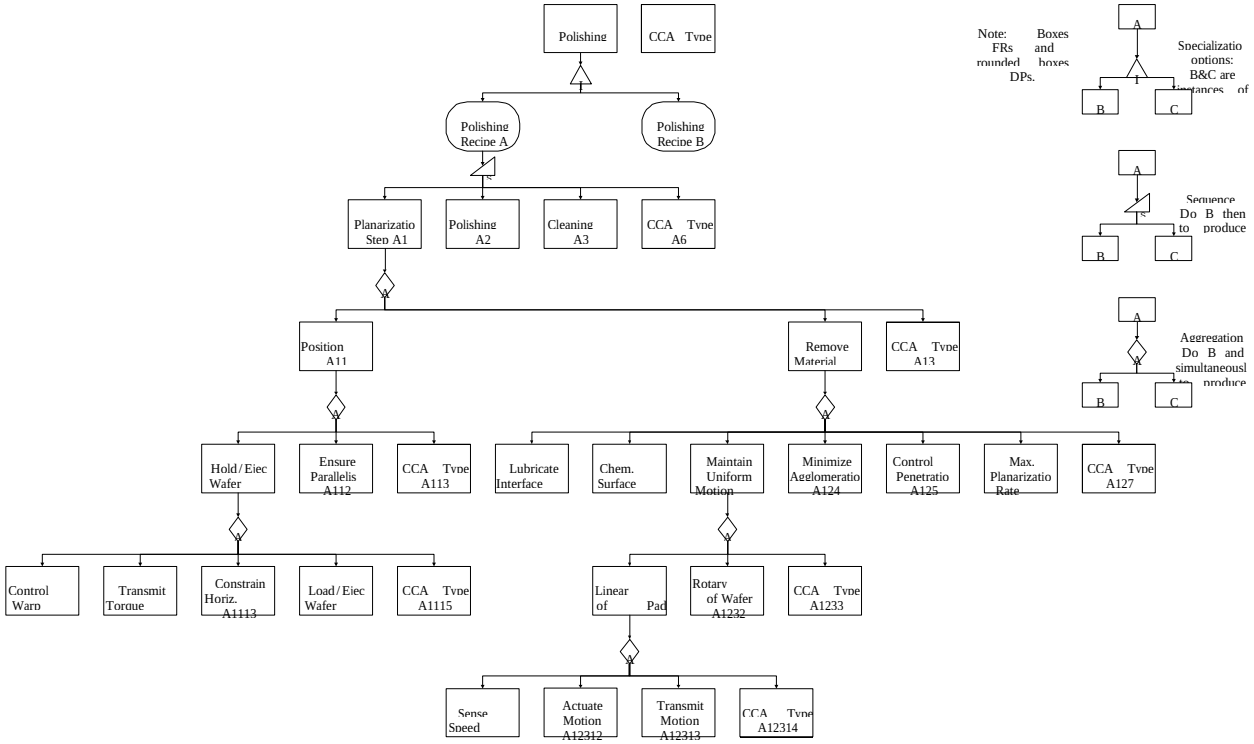

Figure 1. Types of functions performed by the command and control algorithm (CCA)

Table 1. Types of functions performed by the Command and Control Algorithm (CCA)

Type	Input Characteristics	Output Characteristics
I: Selection of recipes (determine FR sequence)	operand output, rate	series of FRs to be performed ("recipe" sequence)
II: Scheduling of DPs (assign times to DPs)	series of FRs to be performed ("recipe" sequence)	schedule of DPs (assigned DPs), FR values over time (desired output)
III: Hardware control (control param. values)	FR values over time (desired output of hardware)	signal to control hardware (DP value over time, $q(t)$)

4 System control

The system architecture representation of the components for a controller has not previously been well defined in axiomatic design. This is because, unlike the hardware components that make up process and transport modules, the different components required for controlling a mechanical system are interspersed throughout the design hierarchy. Furthermore, the order in which decisions are made for the design and implementation of the controller is independent of the order of operations utilized by the controller during normal operation.

In this section, the information flow model for a hardware controller that controls parameter values for specific pieces of hardware is analyzed to determine the appropriate order in which design decisions are made. Next, the controller is examined from the perspective of the system architecture. While the sequence of design decisions must, and indeed does, remain the same, the different control components are included in different branches of the system architecture, since the system architecture is intended to reflect the functional hierarchy of the system. The key results that are shown include the following:

- Actuators and sensors are designed or selected as a part of the process / transport module to which they belong.

- To coordinate all of the motions of a system, a hierarchical structure of control algorithms is defined.
- At a high level in the system architecture, the necessary components for implementing the CCAs must be specified. This includes the choice of computer hardware and software as well as all of the necessary interfaces (that is, hardware, software, user, and network).

Figure 2 shows a schematic diagram of the flow of information through a generic motion controller. Data are acquired from sensors, amplified and read into a computer through A/D or other sensor-specific boards (represented by SP in the diagram), and processed through one or more algorithms. The output of the algorithms are command signals for the actuators, which are written out through D/A, motion controller, or other actuator-specific boards, amplified as appropriate, and transmitted to the actuators.

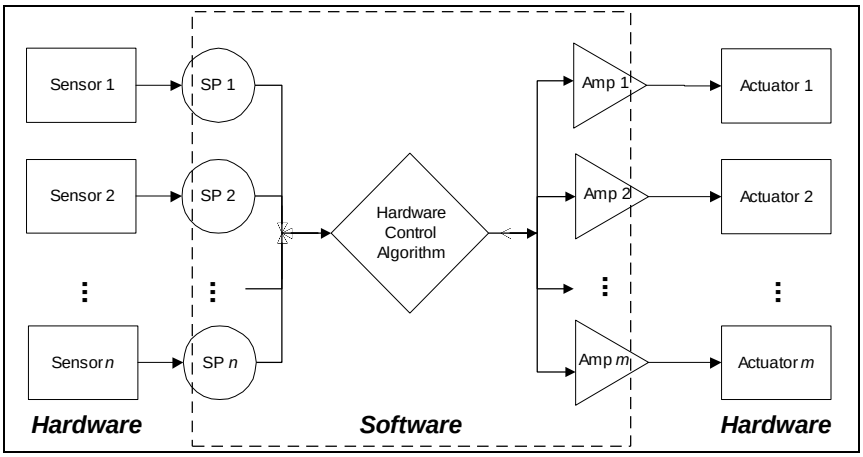

Figure 2. Information flow diagram for a generic controller.

While the information flow model of a controller is relatively intuitive to understand, it does not reflect the order in which these components are designed or selected within the overall system architecture. The actuators required for creating individual motions and the sensors required for characterizing the results of those motions are included in the design of the process or transport module of which they are a part. Only after this hardware is specified can hardware control algorithms be devised for creating the appropriate output signals from the given input parameters. Once these algorithms are developed, decisions can be made as to the appropriate computer hardware and software required for implementing the controller, and finally the appropriate user and network interfaces to enable the controller to accept input from the user or from other systems can be selected.

The controller design matrix that shows these dependencies is shown in equation 2. This may be considered a "lifting out" of command components from the system architecture. Note that the implementation of the computer hardware which runs the algorithms is designed last, so that the computer hardware can meet the needs of the control scheme. The design shows a decoupled matrix, and thus specifies that design decisions are made in a non-iterative, sequential order.

$$\begin{bmatrix} \text{Generate motion } i \\ \text{Sense parameter } j \\ \text{Control action } k \\ \text{Process input } j \\ \text{Transmit output } i \\ \text{Coordinate actions} \\ \text{Implement control} \end{bmatrix} = \begin{bmatrix} X & O & O & O & O & O & O \\ ? & X & O & O & O & O & O \\ X & X & X & O & O & O & O \\ ? & X & X & X & O & O & O \\ X & ? & X & O & X & O & O \\ O & O & X & O & O & X & O \\ O & O & X & X & X & X & X \end{bmatrix} \begin{bmatrix} \text{Actuator } i \\ \text{Sensor } j \\ \text{CCA (3) } k \\ \text{Sig. Proc. } j \\ \text{Amplifier } i \\ \text{CCA (1 \& 2)} \\ \text{Computer Hardware} \end{bmatrix} \quad (2)$$

While the design matrix in equation 2 for a generic controller reflects the relationships between the various controller elements in terms of making design decisions, it does not adequately capture the location of these elements in the system architecture. In general, several different controllers operate in parallel within a process or transport module, and coordinating communication and inputs between these different controllers must be considered. In addition, the system architecture must also account for the design of the computers and the necessary hardware interface boards required for implementing these controllers.

In order to include these controllers within the system architecture, further examination is required to determine where each of the components of a controller is determined in the design of the overall system. This examination can be broken up into three major areas: design of the hardware, specification of the command and control algorithms, and implementation of the controller.

- Hardware: During the design of the hardware for a process or transport module, functional decomposition progresses to the point where a particular motion needs to be generated as a sub-FR for performing the process or transport function. The generic DP that satisfies this FR is a "motion generator". This DP can be decomposed into functions for providing motion, sensing relevant parameters, and correlating input and output signals, as shown by equation 3.

$$\begin{bmatrix} \text{Provide motion (1..m)} \\ \text{Transmit motion (1..m)} \\ \text{Sense parameters (1..n)} \\ \text{Correlate input \& output data} \end{bmatrix} = \begin{bmatrix} X & O & O & O \\ X & X & O & O \\ ? & ? & X & O \\ X & X & X & X \end{bmatrix} \begin{bmatrix} \text{Actuator (1..m)} \\ \text{Transmission (1..m)} \\ \text{Sensor (1..n)} \\ \text{H/W Control Algorithm} \end{bmatrix} \quad (3)$$

- Algorithms: Once all of the individual motion generators have been designed for a process or transport module, algorithms must be defined to coordinate the motions in order to accomplish the task of that module. For example, a 3 axis robot used to transport parts between two manufacturing cells should be able to generate each axis motion independently. However, an algorithm is required to coordinate these motions so that the robot can move the part between two points while meeting constraints, such as not impacting with anything in the environment or providing the motion within a certain amount of time. Within the framework of the system architecture, a hierarchical structure for these control algorithms emerges from the fact that there are different command and control issues at different levels of the design hierarchy, as discussed above. Within the scope of a process or transport module, module command and control algorithms are required *at each level* of the design hierarchy to coordinate all of the individual hardware control algorithms, so that the module will work as one unit. These module command and control algorithms handle the command and control functions at each level of the design hierarchy, and are implemented in the CCAs.
- Implementation: The system architecture must also include the elements required for implementing the control system. The decomposition of the computer hardware includes a specification of the hardware and software platforms, signal interface boards, and network interfaces to other systems.

5 Conclusions

This paper has demonstrated how command and control issues may be represented in a system architecture. Specifically, it has been shown that the components (DPs) required for control are spread out multiple branches of the design hierarchy where active control is required. On the lowest levels, specific actuator and sensor equipment is correlated with a hardware control algorithm to coordinate low-level inputs and outputs. On higher levels of the hierarchy, command algorithms exist to schedule and coordinate all of the lower level command and control algorithms. In this manner, a hierarchy of command and control algorithms spans each branch of the system architecture where active control is required.

Current work in this area includes applying this technique in the generation of system architectures for new and existing systems, as well as developing generic scheduling algorithms to allocate DPs in real-time to meet the needs of time-varying FRs.

REFERENCES

1. Suh, N. P. (1990). *The Principles of Design*, Oxford University Press, New York. '0-19-504345-6 0-19-504345-6
2. Suh, N. P. (1998). *Axiomatic Design: Advances and Applications*. Work in progress.