

Three Design Methodologies, their Associated Organizational Structures and their Relationship to Various Fields

Joel Moses
MIT
March 2004

Abstract

We discuss three methodologies for the design of large scale engineering systems. The classical methodology is hierarchical decomposition, where one breaks a problem down into parts and keeps doing so until the parts are small enough to actually solve, and then combines such solutions. A second approach, which is becoming increasingly popular in software engineering, is the use of a network such as the Internet that contains modules or components, where one attempts to integrate components that were largely designed by others into a system that solves the entire problem. The least popular methodology, we claim, is that of layered design, another hierarchical methodology. Numerous existing systems are in fact layered. For example, the systems underlying personal computers are composed of several layers. Yet, layering is not used as an alternative standard methodology for the design of new applications. Our major goal in this paper is to discuss reasons why this situation exists.

There are advantages and disadvantages to each of the methodologies. Hierarchical decomposition is extremely popular, partly because it is general – it can be applied in most situations. On the other hand, in large scale systems that are undergoing continual change, systems designed using hierarchical decomposition may readily become overly complex, and thus difficult to modify further. Layered systems, on the other hand, can be quite flexible and cope with change very well. The disadvantages of this methodology include the added cost for entering and leaving each layer and the difficulty of determining the right one or two abstractions that split a problem into two or three layers. The network based approach is fine when the network contains the right components, but often the integration of disparate components is not easy, partly because they were not designed to be integrated easily, and eventually the loss of control in this approach may become a serious problem, especially as the system's overall function changes.

We have found that these methodologies and the structures related to them, especially tree structures and layered structures, occur in a variety of other fields. For example, we find them in artificial intelligence, software engineering, systems engineering, management, religion, philosophy, biology, sociology and mathematics. We discuss the relationship of the methodologies and their structures to cultural values, and thereby attempt to explain why layering is underutilized in the U.S.

Introduction

We first consider three methodologies for the design of large scale engineering systems. The classical methodology is that of hierarchical decomposition. A more recently used approach, especially in software systems, is reliance on a network of components. We shall call this the network methodology. Finally, we shall consider layered design, which we believe is insufficiently utilized as a methodology. A key goal in this paper is to explain why layered design is underutilized in the U.S.

The hierarchical decomposition methodology has been used in systems engineering or software engineering since the inception of these fields in the 1950's and 1960's. It and the other methodologies are, however, based on old approaches. We claim that these approaches appear, for example, in mathematics, AI, organization theory, philosophy, religion, and sociology. The reason for the recurring appearance of the approaches is that many fields, such as political theory, organizational theory, and even mathematics, rely on design, problem solving and organizational structures, just as engineering systems do. There are key advantages to the methodologies and their related structures, and these advantages will usually show up regardless of the context. The disadvantages will, of course, also tend to show up in each field. Let us first explain the three methodologies and their associated structures in a bit more detail.

Hierarchical decomposition relies on the notion of breaking problems up into parts. One does so until the parts are simple enough to be solved. This creates a tree structure of parts and solutions. Now one integrates the solutions and thereby derives a solution to the original problem. Hierarchical decomposition is a general approach, since one can easily imagine breaking a large scale problem into parts. Its key weakness as a methodology for the design of systems is due to a key weakness of tree structures, namely that the resulting system is relatively inflexible. That is, if the function of the system needs to change, and function usually does need to change in large scale engineering systems, then it often becomes increasingly difficult to correspondingly change the system due to an increase in its complexity, unless one was extremely careful in the integration step of the approach so that, for example, the eventual structure looks like a layered system.

The network methodology assumes a network that contains significant components of the solution to a problem. This approach has become popular, at least in software engineering, due to the Internet. The advantage is, of course, that one may be able to rely on others for the solution of large parts of a problem. The disadvantage arises when one realizes that one's ability to rely on others is limited, especially as the requirements for a system change during its lifetime. A further disadvantage arises when one realizes that what looked like an appropriate solution to a subproblem is not quite what is needed, and we are back to relying on others to work with us to change their solution. Relying on large scale components that are obtained from a network sometimes leads to a layered solution.

Examples of layered systems include the landline telephone system that uses different hardware/software layers for local, regional and national calls [1]. Personal computers are

layered, with one layer being the microprocessor, and other layers containing the system and application software. In a layered system one does not have a single parent to each part, as in hierarchical decomposition (prior to the integration phase). Instead all members of the immediately higher layer can be parents to a given part. Layering as a methodology is sometimes mentioned in software engineering, but we do not believe that it is really used as such. That is, a conscious attempt to create layered solutions to large scale applications is not used very much. Software engineers frequently mentioned a failed attempt to create a computer-communication standard as a seven layer model by the International Standards Organization as a reason for not proceeding further along this line. The likelihood of some loss in performance in layered systems is another reason given for not using this approach to design and architecture. The difficulty of conceiving one or two ideas that break a problem into two or three layers is yet another reason that is given. We believe the actual reasons for the lack of use of layering as a design or problem solving methodology, especially in the U.S., are different from these reasons and are quite deep.

The analysis of the methodologies, especially those of hierarchical decomposition and layering, will proceed in an autobiographical fashion in the next section, as my understanding of the issues underlying these approaches and their connections to other fields deepened over the years. While we find significant similarities to the design methodologies and their structures in many fields, we are cognizant of the fact that are differences that arise in differing fields, and over long time periods.

The Methodologies and their Relationship to Fields Other than Engineering Systems

Artificial Intelligence

When I became a graduate student in Artificial Intelligence in 1963 I had little idea of the differing approaches to problem solving and design. I was taught then that a key example of an AI system was a theorem proving program in mathematical logic due to Allen Newell and Herbert Simon [2]. This theorem prover worked by creating a tree structure of subproblems. That is, it used hierarchical decomposition. A later system by Newell and Simon yielded the General Problem Solver, which used a similar idea, but on a much more general set of problems [3]. I rebelled against this approach by 1965, arguing that I did not see how such approaches, which tended to increase the number of subproblems exponentially, could solve very difficult problems. Computers in those days were relatively slow, and had relatively small memories. The hope among AI researchers at that time was that increasingly faster and bigger computers would permit us to solve hard problems. But an exponential rate of growth is difficult to beat when one has to rely on an approach that grows at a lower exponential rate, even one that is as large as Moore's Law for integrated circuits (i.e., doubling every 18-24 months). Deep Blue, the special IBM supercomputer, that beat the world champion chess player has provided an exception to this rule.

My approach to problem solving, presented in part in my doctoral thesis in 1967 [4], was to know enough about a problem and how to represent it in a structured fashion so that

the solution becomes much easier. This approach was later called the Knowledge Based Systems approach. It was admitted by many in the following years that AI systems needed to know a great deal about the area in which they were trying to solve problems (calculus integration problems in my case), rather than having them start from scratch with axioms and rules of deduction each time. What was not appreciated about my approach was that the knowledge needed to be structured, if possible in a layered fashion. It was obvious to me that a layered structure was needed, but I did not understand then why others in AI did not appreciate this. In particular the Rule-Based Expert Systems approach to AI later on equated itself with the Knowledge Based Systems approach, but featured a non-structured approach to the set of rules[5].

Software Engineering, in particular the methodology called Structured Programming, was born as a result of a conference in 1968 [6]. It looked to me like a variant of hierarchical decomposition that I saw in AI. Later on I realized that systems engineering also largely relies on hierarchical decomposition. As a result I did not pay much attention to these fields for many years.

**Hierarchical Decomposition: Classical Artificial Intelligence (heuristic tree search)
Layered Design: Knowledge Based Systems**

Organizational Structures

My next encounter with hierarchical decomposition and layering occurred when I became an academic administrator in 1978. I decided that it would be wise to read books on organization theory. I discovered that a key textbook was by Herbert Simon and his CMU colleague J.G. March [7]. The structure it advocated and analyzed was a tree structure for human organizations. I recalled Simon giving lectures at MIT in 1968, which became chapters in his famous monograph “The Sciences of the Artificial [8].” It now became clear to me that not only was Simon using the same structure and methodology in AI and management, but that he also used hierarchical decomposition and tree structures in several other fields, such as economics. One might say that his Nobel Prize in economics was based on recognizing a weakness of hierarchical decomposition. Economists assumed that men made rational decisions by evaluating a decision tree of possible outcomes. Simon coined the term “limited rationality” by pointing out that it was not in general possible to evaluate all decision trees since these grew exponentially. Thus one had to make some simplifications that led to a limited form of rationality.

I now thought about what could be an alternative structure to the classical tree structured organization. Is there a layered approach to organization, for example? There is indeed. Note that universities were not organized into departments until the 19th century. The old structure used assistant, associate and full professors in a layered fashion. Thus full professors would act as a unit to make decisions for the university. A vestigial form of this structure still exists in American universities. The structure was inherited from the Catholic Church, where the priests, bishops and archbishops form three layers. The cardinals among the archbishops form the College of Cardinals and vote for a new Pope, for example. I asked myself where the Catholic Church obtained this idea of

organization. My answer, based on my readings as an undergraduate, was Plato. Before I discuss Plato I note that I later realized that there are many organizations in the U.S. besides the Church and universities that rely on layering as a basic organizational structure. These are large partnerships, such as law firms. The variant on the concept of obtaining tenure in these organizations is becoming a partner. There are usually three layers in these organizations as well, associates, junior partners and senior partners.

Hierarchical Decomposition: Tree structured hierarchies

Layered Design: Layered organizations, large partnerships

Greek Philosophy

Plato's "Republic" discusses an ideal society, called the Just Society, run by a philosopher-king [9]. The highest layer of the Just Society was that of guardians. It was also called the gold layer. The middle layer was silver and the lowest layer was bronze. The guardians formed a cooperative, communistic layer. I recall that Americans tended to deemphasize this part of the "Republic," and emphasized instead the importance of education in it. I also recalled that Plato did not particularly like the relatively flat and partially democratic society of Athens.

I also recalled that Plato's student, Aristotle, was unhappy with some of his teacher's ideas. He thought that Plato and Socrates were vague, for example, in their use of abstract concepts, such as justice. This led me to think about how Aristotle approached problem solving and organization. Aristotle, it turns out, loved tree structures. He used tree structures to describe how Greek city-states ought to be managed in his "Politics [10]." He also invented a form of logic, which led him naturally to thinking about proving statements using a tree structure of subproblems

So philosophy has at least two of the methodologies and modes of thought in it. One reason for the differences in their modes of thought is that Plato's family was part of the Athenian aristocracy, whereas Aristotle's was in the middle class, even though Alexander the Great was Aristotle's pupil. A second reason for the difference in their views is likely to be the difference in their personalities. Unfortunately each of these great philosophers felt that his view was best, whereas my developing view was that what is best depends critically on the circumstances in which the system operated, and the changes with which it had to deal, and that the answer could change over time. Did this difference of approach between Plato and Aristotle occur elsewhere in philosophy? Before answering this question I delved into the Bible. Did some people not say that the Bible has answers to everything? Sure enough, one can find different organizational and problem solving approaches in the Pentateuch.

Hierarchical Decomposition: Aristotle

Layered Design: Plato

The Bible

A famous story in Exodus is that of Jethro, Moses' father-in-law. Jethro goes to visit Moses at the foot of Mount Sinai. He sees Moses spending all his time rendering judgment in disputes. He tells Moses that this is not good. He should appoint judges for every thousand, judges for every hundred, fifty and ten. Thus only the most difficult cases would end up being judged by Moses. Moses said fine to all this. I believe that this is the first time that a tree structured organization is described in writing.

Why did Moses give in so easily? One answer is that the tree structured hierarchy is an obviously good approach to organizing a judicial system. Why then did Moses not think of it himself? The reason, I claim, is that his basic approach to problem solving and organization was different from that of a smart and normal person such as Jethro. Moses did create a structure in the Bible about which he cared a great deal. This is the religious structure of the Israelites described in Numbers. This structure had Aaron as the High Priest. Aaron's family became priests at the highest layer of the hierarchy, the rest of the tribe of Levi was in the middle layer, and the rest of the Israelites were in the lowest layer. The Priests acted as a unit in this hierarchy. Israelites did not deal with a particular priest at all times at the Temple, but dealt with whichever priest happened to be there at a given time. This structure anticipates aspects of the Just Society by Plato by hundreds of years. Plato's society, however, did not rely on the accident of birth as the way to determine where one is placed in the structure. He relied on education and testing, and while this is theoretically quite interesting, Plato fell into a trap of creating a society where wives were shared so that the husbands could not favor their own children during the testing process. The lesson here is to be wary of theorists. They sometimes go too far.

A beautiful example in the Bible of layering predates Jethro and Moses. In the first chapter of Genesis, God says "Let us make man in our own image" He also let him have dominion over living things. Who is "us"? Many commentators say that God is speaking to the angels. Angels compose the top layer of a hierarchy, where men and women are in the middle layer, and other living things are at the lowest layer. Since only Adam and Eve were in the middle layer at that point, the first law in the Bible is "Be fruitful and multiply."

The concerns in the Garden of Eden are over the diminishing differences between mankind and the angels. Once Adam and Eve ate from the fruit of the tree of knowledge of good and evil, they must be prevented from eating from the fruit of the tree of life lest there be little difference between them and the angels. The structure implicit here is almost purely algebraic in character. I shall return to abstract algebra as a way of describing the structure of systems later.

Hierarchical Decomposition: Jethro
Layered Design: Moses

Philosophy since the Middle Ages

Let me return to philosophy. Aristotelian writings were lost to the Western world for hundreds of years until the 12th and 13th centuries. At the turn of the millennium

Christianity was influenced by neo-platonic thought. It should not be surprising that Western societies at the time were usually layered, with kings at the top of the hierarchy, an upper class made up of land-owning barons, a middle class of merchants and other specialists, such as knights, and a lower class, largely of serfs. St. Thomas Aquinas attempted to integrate the Aristotelian mode of thought with that of the Church in the 13th Century [11]. One way in which the conflict between Platonic and Aristotelian modes of thought showed up later in the Middle Ages was in the Nominalist/Realist controversy. Plato emphasized abstraction. To him abstractions, such as that of an ideal chair, were quite real. Aristotle, or so the Church fathers thought, would think of names for objects as just names, and hence that position was called the Nominalist position. The Nominalist position shows up later in mathematical logic, for example.

The Aristotelian mode of thought grew in importance in the Western world over the following hundreds of years, especially as a result of the Renaissance, the rise of the Protestant middle class, the growth of democracies, and the rise of science. This was not uniformly so in the West, since Catholic regions tended to hold neo-platonic views of problem solving and the structure of society, at least to some degree. In particular, one sees a difference in position between Britain and Germany. David Hume, early in the 18th Century, talked of the mind as being an empty slate, with learning after birth accounting for all our personal knowledge [12]. The German philosopher Kant criticized Hume's position in "Critique of Pure Reason [13]" Kant talked of built-in categories in the mind, something of which Plato would likely approve. This difference in point of view between British and German philosophers shows up in the 20th Century, when Bertrand Russell criticizes Kant for being vague, an accusation similar to that made by Aristotle against Plato and Socrates.

Hierarchical Decomposition: British Analytic Philosophy **Layered Design: German idealism**

Mathematics

As a student I studied mathematics as well as computer science and artificial intelligence. Thus I recognized an affinity between Aristotle and Mathematical Logic. Was there a similar affinity in mathematics to Platonic thought? Plato viewed himself as a geometer, but I think that were he alive today, he would likely consider himself to be an algebraist or an algebraic geometer. Plato was an abstract thinker, and abstract algebra provides a language for mathematical abstractions, such as rational functions. Algebra, I believe, is a good way of describing layered systems, whereas logic is a good way of thinking about hierarchical reductionism and its tree structures. Abstract algebra was largely developed in Germany in the second half of the 19th Century, partly by Riemann and the Noethers, father and daughter. While Germany also played a key role in the development of logic, Lord Russell and other British thinkers also played an important role in its development. British analytic philosophy is based in large part on logic. I think this split between logic and algebra hints at the split between Hume and Kant as well as between Aristotle and Plato.

Hierarchical Decomposition: Logic
Layered Design: Abstract algebra

Models of the Human Mind and Body

In 1985 I began a faculty seminar that has been going on for nearly twenty years. Our initial interest was in understanding learning algorithms using neural networks [14]. My interest in this approach was piqued because I saw in it a three-layered architecture for pattern recognition. Minsky and Papert [15] showed in 1969 that a single perceptron could not recognize connected figures, and they conjectured that two layer perceptrons would do no better. Based on my historical analysis I guessed that three layers would make a significant difference. I was not altogether thrilled that the neural network researchers first published that it was so. These three layered systems are, however, still quite limited in their problem solving power, especially in areas involving natural language. Nevertheless, I find it interesting that the cerebral cortex has six layers.

The body uses a variety of structures, in addition to the layering in the brain. A very common structure is the tree structure that is used, for example, in arteries and veins. My guess is that tree structures are used, although they are quite inflexible, as we shall see later, because the body can replace arteries and veins at a microscopic level relatively easily. This is not so in most engineering systems.

Manufacturing

While I was a department head at MIT in the 1980's there was a crisis in American manufacturing. It was realized that the Japanese were able to manufacture products, such as automobiles, at a lower cost and higher quality than their American competitors. They were also able to introduce new car models much more easily than their American counterparts. This issue led to me to read literature related to the organization of firms in Japan. I eventually read about their national culture [16,17]. What I saw there was a greater reliance on layered systems than I saw in the U.S. There was also greater reliance on cooperation and teamwork within firms. This led me to think about national cultural values and their relation to the methodologies and associated structures.

National Cultures

The early European settlers in the U.S. rejected certain European ways, especially feudalism. Feudalism relies on a layered structure of society in which, contrary to Plato's society, one is placed in the layer to which one was born. The alternative layered structure, the one where one can rise up the ranks, did not take hold in the U.S. except for special cases, such as large partnerships, the Church and universities. Thus the culture in the U.S. differs from that of several of its major trading partners in certain aspects associated with layered systems, such as cooperation. The U.S. culture emphasizes

competition. Tree structures are excellent vehicles for conveying the value of competition. They are also excellent vehicles for a highly individualistic society, since one can assign individuals to manage a subtree, and have them compete with other individuals at the same level of the organization.

Attitudes toward engineering and engineers vary from country to country. For example, the attitude toward engineers is quite poor in England, where engineers are viewed as operators of engines. England has, of course, had a significant influence on the U.S. On the other hand, the attitude toward engineering is quite positive in Japan and Germany. One reason may be that in contrast to the England, Germany, Italy and Japan were not united as modern nation-states until 1860-1870. Thus their memory of a feudal past, at least in some regions of these countries, is quite good. For example, my father, who was born in Germany, became an apprentice to one of his father's competitors in 1920 with a written agreement lasting five years.

It was not surprising to me to see layered solutions to engineering systems arise in Japan. I am thinking, in particular, of platform-based designs, such as are used in automobiles. A very interesting example of the use of platforms is the Sony Walkman family of products. The Walkman family uses a new chip every couple of years, with each chip being associated with dozens of different packages, each resulting in a different Walkman product. The key advantage of platforms is the reduced cost of designing and producing a variety of ostensibly different products. A disadvantage is that each product is not fully optimized, but in many cases the overall savings for the family of products outweighs the financial implications of a possible loss in performance.

Americans learned from the Japanese, partly through successful books, such as *The Machine that Changed the World* [18]. We learned the importance of teamwork in teams composed of members from different parts of the company. This was not an easy task, since the American firms were not initially organized to cooperate among the different divisions and functions in the firm. What made this transition possible were the recession in the early 90's and the threat that unless the firms changed their methods of operation, the employees would lose their jobs.

The interesting change in recent years is that trade between nations with different cultural values, and different approaches to design and organization, has led to a convergence of approaches in various parts of the world. I believe that much more is to be learned, both about the implications of cultural differences, and about how convergence can be increased even further. I do not believe that there is an ideal long term solution to structural issues in large scale systems, and thus it is important to understand the advantages and disadvantages of alternative approaches.

Hierarchical Decomposition: U.S. culture
Layered Design: Japanese culture

The Network Methodology and Tribalism

The network or grid-based methodology did not play an important role in my thinking until the 1990's. With the rise of the Internet, increased use of software and even hardware components obtained via the net became at least a possibility, and in many cases a reality. Given my interest in finding analogies to the problem solving and organizational methodologies, I looked for instances where network-like approaches were used in the past. I claim to have found it in the behavior of tribes. Imagine that you are a merchant doing business in different countries thousands of years ago. Whom would you trust to do business with? For one, you might trust someone who was a member of your extended family or was born in your geographic area. Italians, for example, tend to ask which town you or your parents came from, in order to develop some commonality, if not trust. Piore and Sabel [19] discuss how families in certain towns in northern Italy work together on new product development and manufacturing. Another trust relationship might be based on common religion. Jews used to be successful merchants for millennia because there were Jews dispersed over many countries, and they would trust to deal with each other, at least initially. I consider both situations as examples of tribalism.

One of the surprises in my career in Computer Science was the popularity of open software systems in the past two decades. My colleague, Richard Stallman, is largely responsible for this movement. The development of GNU/LINUX relies on the work of many programmers around the world. I think that the relationship of these programmers to each other is comparable to that of a tribe. What we are now seeing is that large firms, such as IBM, using open systems as a strategic tool. Whereas much of the tribal approaches in the past led to relatively non-hierarchical firms, the recent open systems approaches can be used to create hierarchies, in particular layered hierarchies.

Layered systems based on components from the net have obvious advantages and disadvantages. Their obvious advantage is that net-based components reduce the work needed to develop key parts of a system. Disadvantages arise when the requirements for the system change, as is likely to occur often in most large scale systems. Now one has to trust the component developer to help make the appropriate changes. Similarly, when the component developer changes their part of the system, it will likely force the recipients to change theirs. Another set of issues involves the fact that one usually does not fully appreciate the actual behavior of a large component, even when it and other components all adhere to a single standard.

In the sections below we discuss three issues that arise in the methodologies and their related structures.

Cooperation and Competition in the Methodologies

One of the issues that arise in the various organizational structures is the differing role of cooperation and competition in them. Hierarchical decomposition lends itself to competition between the different parts of the tree structure. In a society that emphasizes competition, such as the U.S., tree structured organization and problem solving approaches will be favored. Layered systems lend themselves to cooperation, since each

layer can be thought of as a whole. Societies that emphasize cooperation, such as the Japanese society, will tend to use layered ways of designing systems and organizing firms more so than the U.S. would. The network methodology will likely be an intermediate approach, using cooperation between components, and potentially some competition within components.

Hierarchical Decomposition: Competition favored

Layered Design: Cooperation favored

Network-based Design: Variable

The Role of Trust

Trust relationships can be very important in the design and management of systems. Tree structured organizations do not usually rely heavily on trust. Instead, bosses are assumed to check on the work of their employees. This attitude changed somewhat as a result of the manufacturing crisis in the 1980's when, for example, teams were created whose members came from different parts of the firm.

Layered systems and societies can have very complex trust relationships. For example, one may assume that individuals at the highest layer in a society will honor their agreements. Japanese claim to have multiple levels of trust, with the highest being the willingness to have you marry their daughter. It is not surprising that the relationship between Toyota and its Japanese suppliers is largely based on trust. The suppliers will trust Toyota not to lower its payments, even in recessionary times, to such a level that the suppliers will be forced to go out of business. Toyota can trust the suppliers to do their utmost to meet the changing specifications of the products, as well as keep lowering their costs to them.

Hierarchical Decomposition: Relatively low trust

Layered Design: High trust

Network-based Design: Variable to high trust

The Role of Flexibility

Flexibility is a key system property in large scale engineering systems. Such systems undergo numerous changes during their lifetime, and the ease by which these changes can be implemented, namely the system's flexibility, is thus a key property. We claim that tree structured systems and organizations tend to be relatively inflexible for many classes of changes. What is apparently easy to do with a tree structure is to combine systems under a master node, or delete a subsystem. As a result, U.S. firms tend to merge and sell divisions to each other with relative ease. What has been more difficult was creating teams from different parts of the organizations that would work together quickly and effectively. While teams have been formed quite readily in the U.S. since the 1990's, one wonders how much more effective they might be if the organization of which they are a part were organized to promote cooperation and foster flexibility.

In [1] we define flexibility as the number of paths in a system divided by the number of nodes. This definition is related to the intuitive notion that the more internal choice points there are in a system the easier it will be to implement changes in its overall behavior, and if need be its architecture. As a result of this definition it becomes clear that a tree structure, which has just one path per final node, is relatively inflexible. A layered structure, especially one with three or more layers, will be quite flexible. A networked structure is extremely flexible, although it may be difficult to control, since in this structure control is usually distributed.

We note that one could and often does implement changes in tree structured systems by violating the tree paradigm and creating nearly horizontal connections. This approach results in greatly increasing the complexity of the system. Eventually such systems may become so complex that they cannot be changed further.

Now consider the rate of change that a system must undergo. If the rate is relatively low, then a tree structure may be fine. I believe that this explains why GM's organization was so effective between 1920 and 1970. GM was sufficiently large relative to its U.S. competitors so as to be able to control the real rate of change in automobiles models. Even when it introduced new models each year, the underlying rate of change was low enough so as not to create too much difficulty for GM. The Japanese automobile manufacturers were able to compete effectively with GM and other American manufacturers in the past thirty years on the basis of quality, cost and, for us especially, shorter time to market, thereby designing cars that are likely to be of interest to the consumer. We claim that at least some of these properties are a direct result of the Japanese organizational structure and their societal values.

Hierarchical Decomposition: Low flexibility

Layered Design: Medium flexibility

Network-based Design: Potentially very high flexibility

Summary

We discuss three methodologies for the design of large scale engineering systems and their associated organizational structures. These approaches are hierarchical decomposition and its associated tree structure, layered design, and network-based design. We then describe how we found references to these methodologies and structures, especially the first two, in a surprising variety of fields – artificial intelligence, software and systems engineering, management, religion, philosophy, sociology, biology and mathematics.

In comparing the different methodologies we emphasize issues, such as cooperation versus competition. We also emphasize the ease of making changes in systems, namely the system's flexibility. Our definition of flexibility leads to the conclusion that tree

structured organizations are inherently quite inflexible, and thus cannot handle well medium or high rates of change in the functionality of a system.

Acknowledgements

This paper is based in part on a much longer essay entitled “Organization and Ideology” written while the author was on sabbatical at the Harvard Business School during the academic year 1989-1990. My colleagues at the Engineering Systems Division at MIT and in the Moses Seminar have had to endure various expositions of this point of view, and I wish to thank them all.

Appendix

The Power of Abstract Algebra and Layering: The Case of Indefinite Integration

This Appendix that gives a brief explanation of how the modern algorithms for indefinite integration work. These approaches rely on abstract algebra and algebraic geometry. The methodology and structure used in this algorithm has had a major influence on my thinking about design methodologies and organizational structures. In particular, the algorithm uses a layered structure. I have found it relatively difficult to explain the value of abstract algebra as an approach for modeling large scale complex systems [], and I hope that this appendix will help fill some of the need. It is unfortunate that engineers have not used abstract algebra as a way of describing the structure of systems. Nor have algebraic geometers done much to explain their approaches to problem solving and structure.

As every calculus student knows, integration problems are solved in the textbooks with a variety of heuristics, such as integration by parts. There are some problems for which no integral in closed form exists. Some teachers have been known to ask students to solve such problems, especially over a weekend. It would appear that determining whether integrals exist in closed form at all must be an insuperably difficult problem. The very best mathematicians believed this until recent decades. Actually, general algorithms for integrating many types of calculus problems were known in the 19th century, but the hard classes of problems involving algebraic functions, such as roots of polynomials, were not solved until thirty years ago.

A key idea in the modern theory of integration, due, in part, to Robert Risch [20,21], is the representation of the integrand in a layered structure, called a tower in abstract algebra. Such a representation separates the functions composing the integrand sufficiently to make it relatively easy to determine whether an integral exists. The basic functions in the calculus are the rational functions that are ratios of polynomials. These form a structure called a *field* in abstract algebra because one can add, subtract, multiply as well as divide them and still get rational functions. It has long been known that the exponential function, e^x , and the logarithmic function, $\log(x)$, are not rational functions. In fact, these functions are transcendental over the rational functions since no polynomial with rational functions as coefficients has as a root either of these functions. Now if one lets $y=e^x$ and $z = \log(x)$, then one can manipulate (i.e., add, subtract, multiply, divide) any function involving these two transcendental functions and rational functions in x as if one were manipulating a rational function in three variables, x , y , and z . One need not fear of making any errors (such as generating a 0 without recognizing it) as a result of the transformations. Moreover, one can think of rational functions in x and e^x as a rational function in y with coefficients which are rational functions in x . This equivalence makes clear the layered relationship between x and y . That is, the set of polynomials in y with

coefficients in x is an abstraction over the set of polynomials in x only. For example, the following equation would be equivalent if y replaced e^x :

$$2x e^{2x} + (x^2 - 3) e^x + 7x - 4$$

$$2x y^2 + (x^2 - 3) y + 7x - 4$$

The latter equation is, of course, easier to manipulate. One also needs to be able to differentiate new variables, such as y . This is relatively easy to do by introducing a substitution, such as $y' = y$, if y replaced e^x .

The general integration process, for problems that do not involve algebraic functions, is then to create a pyramid or tower of abstractions in which the integrand lies. Each new abstraction beyond the rational functions will be a logarithm or exponential (we exclude in this discussion the case of algebraic functions, such as roots of polynomials). Once we have generated the appropriate tower we can solve the original integration problem by going down the layers and creating progressively simpler integration subproblems, until we get to the layer involving only rational functions. Those subproblems involving rational functions are either soluble, and we have generated an integral, or are not soluble, and no integral exists in closed form in terms of the elementary functions of the calculus. By the way, problems involving trigonometric problems are solved in this algorithm by substituting their complex exponential equivalents.

Some examples will clarify the general approach. Consider the following integration problem:

$$\int x e^x dx$$

The first thing to do is to determine the appropriate tower or pyramid of extensions to the rational functions in x . Since we only have e^x as an additional function to be considered, and we know that e^x is transcendental over the rational functions, we create the pyramid $R(x, e^x)$. That is, we create a pyramid $R(x, y)$ where $y=e^x$. The integral, if it exists in closed form, must lie in such a pyramid. (In general, one must allow for new logarithms, but this cannot occur in our case.) In fact, we know more about the integral. Given that the last extension in the pyramid is an exponential, the integral must be a multiple of that exponential. This multiple is not a constant multiple but a function, $A(x)$ say, that is in the pyramid, but in the layers of the pyramid below the exponential layer. In our case, $A(x)$ must be among the rational functions in x . If we differentiate this proposed form of the integral we get the following equation:

$$x e^x = A'(x) e^x + A(x) e^x$$

Dividing the exponential out of both sides, we get

$$x = A'(x) + A(x)$$

Recall that $A(x)$ is a rational function, that is, a ratio of polynomials. If A had a nontrivial denominator, then the degree of that denominator in x would increase in A' , and such a term could not be cancelled by the rest of the equation. Therefore, A must be a polynomial in x of degree n , say. To determine a bound for n , note that the degree of the left-hand-side in x is 1. Therefore the degree of the right-hand-side must be 1 also. This is then a bound for the degree of $A(x)$. That is, $A(x)$ is linear in x . Let $A(x) = ax + b$, where a and b are constants. Substituting this value in the equation we obtain

$$x = a + (a x + b)$$

$$x = a x + (a + b)$$

Solving the equation above for a and b we get $a=1$, $b=-1$. That is, the integral is $(x - 1) e^x$.

We could have obtained the same result with less effort by integrating by parts, but the advantage of the general method is that far harder problems can be tackled in essentially the same way. Consider the well-known problem that is not integrable in closed form in terms of the usual elementary functions of the calculus:

$$\int e^{x^2} dx$$

A similar analysis to the one given above shows that the integral, if it exists in closed form, must be in $R(x, e^{x^2})$. Once again the integral must be a multiple of the exponential, say $A(x)$, where A is a rational function in x . Differentiating we get the following equation:

$$e^{x^2} = (A'(x) + 2 x A(x)) e^{x^2}$$

Dividing both sides by the exponential term, we get

$$1 = A' + 2 x A$$

A similar analysis to that above shows that A cannot have a nontrivial denominator, and must thus be a polynomial again. Let us say its degree is n . The degree of the left-hand-side of the equation above is 0. Therefore the degree of the highest term on the right must also be 0. But this is not possible because if A is any nonzero polynomial the term $2 x A$ would have degree at least 1 in x . If A is 0, then the right-hand-side is 0 and clearly does not equal the left. So we have a contradiction, which proves that the problem is not integrable in closed form in terms of the elementary functions of the calculus.

Historically, arguments of nonintegrability were quite complex, and were usually outside the scope of the calculus texts. Now such arguments are quite natural and

implicitly made in algebra systems, such as Macsyma and Mathematica[22]. Moreover, when the result exists in closed form in terms of the usual elementary functions of the calculus, the algorithm will find it.

References

- 1) J Moses, "The Anatomy of Large Scale Systems," ESD_WP_2003_01.25, esd.mit.edu, 2002
- 2) A. Newell, and H. A. Simon, "The logic theory machine: A complex information processing system" *IRE Trans. Inf. Theory*, 1956, IT-2:61-79
- 3) A. Newell, J. C. Shaw, and H. A. Simon, "Report on a general problem solving program," *Proceedings of the International Conference on Information Processing*. UNESCO, Paris, 1960, pp. 256-64
- 4) J. Moses, *Symbolic Integration*, MAC-TR-47, Project MAC, MIT, 1967
- 5) E. Feigenbaum and P. McCorduck, *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*, Addison-Wesley, 1983
- 6) O.J. Dahl, E.W. Dijkstra, and C.A.R. Hoare, *Structured Programming*, Academic Press, 1972
- 7) J. G. March and H.A. Simon, *Organizations*, Wiley, 1958
- 8) H.A. Simon, *Sciences of the Artificial*, Third Edition, MIT Press, 1996
- 9) Plato, *The Republic*, trans. by D. Lee, Penguin Books, 1955
- 10) Aristotle, *The Politics*, trans. T.A. Sinclair, Penguin Books, 1951
- 11) St. Thomas Aquinas, *Summa Theologica*, 1920
- 12) D. Hume, *An Inquiry Concerning Human Understanding*, Oxford, 1993
- 13) I. Kant, *Critique of Pure Reason*, Hackett, 1996
- 14) D.E. Rumelhart, J.L. McClelland and the PDP Research Group, *Parallel Distributed Processing*, Vols. 1 and 2, MIT Press, 1986
- 15) M.L. Minsky and S. Papert, *Perceptrons*, Expanded Edition, MIT Press, 1988
- 16) G.C. Lodge and E.F. Vogel, eds, *Ideology and National Competitiveness: An Analysis of Nine Countries*, Harvard Business School Press, 1987
- 17) G. Hofstede, *Cultures and Organizations: Software of the Mind*, McGraw-Hill, 1997
- 18) J. P. Womack, D. T. Jones, and D. Roos, *The Machine that Changed the World: The Story of Lean Production*, HarperCollins, 1991
- 19) M. J. Piore and C. F. Sabel, *The Second Industrial Divide: Possibility for Prosperity*, Basic Books, 1984
- 20) R. H. Risch, "The Problem of Integration in Finite Terms", *Trans. AMS* 139 1969, pp. 167—189
- 21) E.R. Kolchin, *Differential Algebra and Algebraic Groups*, Academic Press, 1973
- 22) S. Wolfram, *The Mathematica Book*, Fifth Edition, Wolfram Research, 1993