Notes For “Notes On “Notes” ”

Richard P. Gabriel

Outside the window,
next door,
a shovel scrapes along the surface
of concrete and I'm guessing
something sloppy is happening.

Richard P. Gabriel, lines from *Clinical Locution*

[*This short note is a prolegomenon to a longer, more thorough essay on my experiences trying to reproduce Christopher Alexander's Indian Village design results as presented in "Notes on the Synthesis of Form" and "The Determination of Components for an Indian Village" -rpg*]

I first heard of Christopher Alexander around 1990; I started with “The Timeless Way of Building,” dipped into “A Pattern Language,” absorbed “The Oregon Experiment,” moved to “A Foreshadowing of 21st Century Art: The Color and Geometry of Very Early Turkish Carpets,” then I got pre-publication photocopies in 1996 of the four volumes of “The Nature of Order.” In the early 1990s, Alexander started to become popular with software developers, and I wrote a series of essays about his work for my column in *The Journal of Object-Oriented Programming*; these essays turned into a book, and Christopher Alexander wrote its Foreword. Later I supervised his work on the Gatemaker program.[1]

“Notes” was among the last books of Alexander that I read—around 2015. Unlike some computer scientists who loved his concept of misfits and his algorithmic approach to design modularity, I considered this formalism non-Alexandrian, and therefore a distraction. But so many people talked about this book that I felt I needed to read it to be a complete Alexandrian scholar.

I was intrigued by the idea hinted at that a program written around 1960 could solve as complex a problem as the Indian Village redesign / rebuild—the “Worked Example” reported in the Appendix to “Notes.” The essential problem was to take a set of design “requirements” (141 of them), a set of interactions among them (about 1400 of them), and partition them into groups that represent coherent design subtasks (more or less) or ‘components.’ Alexander's approach was to create a ‘goodness’ measure that would determine (numerically) how good a partition was. Then the idea was to ***generate*** disjoint partitions and ***test*** them using this measure—computer scientists call this algorithmic search technique “generate-and-test.” The “Notes” Appendix included a pretty decomposition of the problem.

I tried to reproduce Alexander's results. I was immediately confused by the many clerical-like errors in the raw data supplied in the Appendix and the odd mathematical approach he took

[1] https://www.youtube.com/watch?v=o8b7ZBWGmu4

to creating his goodness measure. The clerical errors[2] and sketchy definitions of terms made interpreting the apparently straightforward goodness measure difficult. Moreover, "Notes" did not contain a direct statement that the program hinted at actually produced the presented decomposition.

The references in "Notes" mention two research reports that seemed to promise explanations: I call them "HIDECS 2"[3] and "HIDECS 3."[4] I was unable to obtain them until long after the start of my investigation.

The problem to be solved is essentially the problem of ***cohesion and coupling,*** a pair of technical characteristics of programs in modern software modularity: it is important to gather together programming "concerns" that strongly belong together (*cohesion*) while isolating less strongly binding concerns (*coupling*). One way to think about it is that the members of a family do a lot of things together (*cohesion*) while members of distinct families don't do as much together (*coupling*). These modern concepts of cohesion and coupling were (likely) not available to Alexander in 1959 in this exact form—that is, with these names.

In addition to trying to decipher Alexander's approach, I tried several now-classical algorithms: K-Means clustering, Silhouette clustering, Karger's algorithm, and several of my own devising. For generate-and-test I used dynamic programming, greedy algorithms, simulated annealing, genetic programming, and some simple hill-climbing techniques. None worked well enough to come close to reproducing the decomposition in "Notes."

After many failed tries at reproducing the results in "Notes" I finally obtained the two HIDECS reports as well as a version of the HIDECS 2 program transliterated into Python.[5] At the same time, I obtained a paper entitled "The Determination of Components for an Indian Village,"[6] in which Alexander shows a slightly different goodness measure from the one in "Notes" and states directly that "minimization according to this function has been programmed for the IBM 7090. It is this function which gave the decomposition of the village problem that follows." The decomposition that followed was exactly the one in "Notes."

Of the 50 errors in the interactions table, 30 involve requirement 33: "Fertile land to be used to best advantage." The errors are that some requirements list asymmetric interactions. That is, whenever we see a statement like "33 interacts with 56," we (and Alexander's algorithm) expect to see "56 interacts with 33." The key to Alexander's mathematical analysis of complex decomposition problems and the goodness measure he creates is counting the number of links between sets of requirements. Before I had the source code for his program,

[2] There are either 1383 interactions or 1433, depending on how you treat the errors. In the HIDECS 2 report, Alexander makes it clear he used 1383, regarding the 50 others as essentially cardpunch errors.

[3] Christopher Alexander and Marvin Manheim, "HIDECS 2: A Computer Program for the Hierarchical Decomposition of a Set with an Associated Graph," M.I.T. Civil Engineering Systems Laboratory Publication No. 160 (Cambridge, Mass., 1962).

[4] Christopher Alexander, "HIDECS 3: Four Computer Programs for the Hierarchical Decomposition of Systems Which Have an Associated Linear Graph," M.I.T. Civil Engineering Systems Laboratory Research Report R63-27 (Cambridge, Mass., 1963).

[5] https://gitlab.com/Zenbagailu/hidecs-2-python

[6] In Christopher Jones, Conference on Design Method, Pergamon, 1963. https://beautiful.software

these errors made it hard to understand his analysis and therefore his goodness measure. Such errors are not uncommon in his early papers. While doing a close reading of an earlier paper of his—"A Result in Visual Aesthetics"[7]—I noticed a handful of similarly careless and sloppy statements and data.[8] When something like this happens in fiction, it's called an "unreliable narrator."

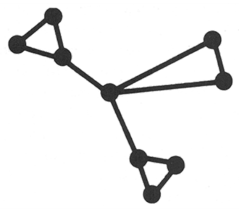
The two HIDECS reports describe five different programs, each using a different approach to partitioning a design problem. After receiving the new material I coded up my own versions of most of them, but none of them produced exactly the decomposition in "Notes." However, that was not the interesting conclusion.

The program called HIDECS 2 was designed to separate components into clusters with minimal information transfer between them, meaning that the number of interaction links across cluster boundaries is small. Alexander was trying to solve the coupling part of the cohesion / coupling problem. Using the family analogy, he was trying to identify families in a population by finding clusters of people where each cluster doesn't do much with the other clusters.

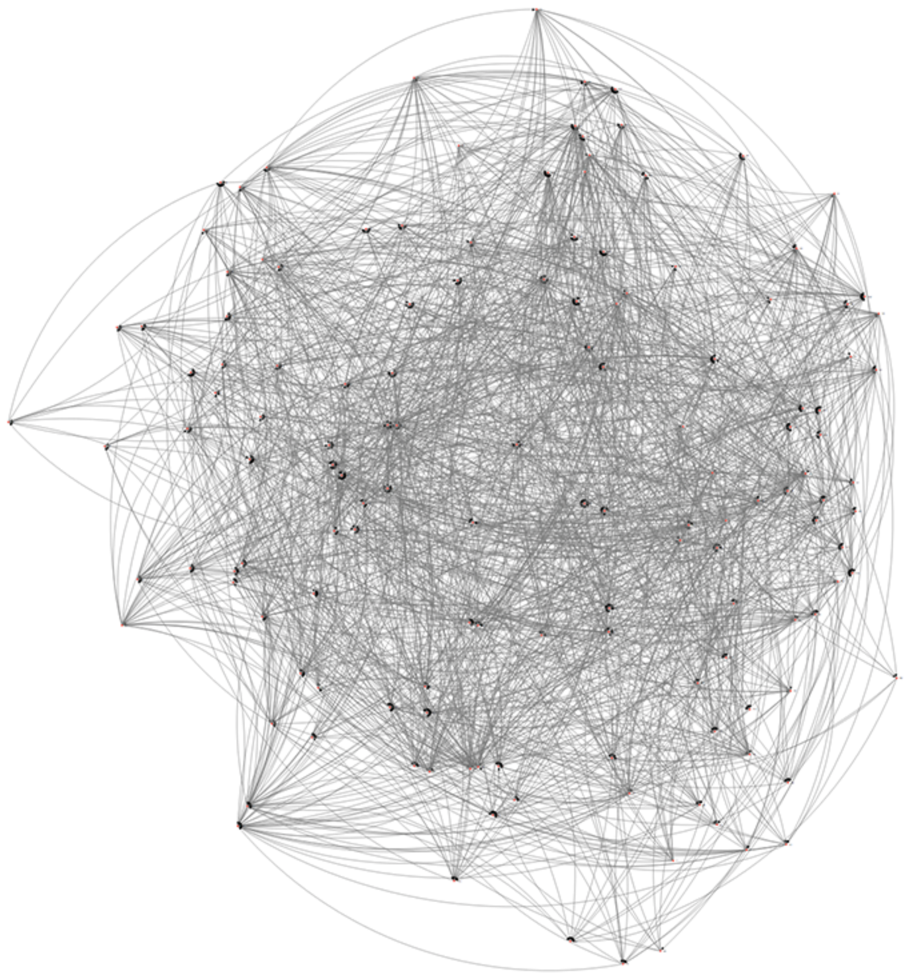
In the HIDECS reports Alexander calls the design requirements "vertices" or "misfit variables" and the interactions between them "links." HIDECS 2 proceeds by splitting the set of all the vertices into two disjoint subsets (*partitions*) using a random selection process that produces two subsets of, typically, unequal size. Next the program systematically tries moving single vertices from one subset to the other, one at a time, measuring the goodness of partition at each step, and selecting the best.[9] This yields a binary partition of the set of vertices into disjoint subsets; the program moves ahead by doing the same process on the two partitions separately. The result is a binary tree: each node in the tree has exactly two subtrees below it. Computer scientists describe this strategy as a "top-down algorithm." Note also that the goodness measure needs to measure the goodness of a partition of only two sets.

In my early investigations I had discovered that trying to find clusters by looking for weak coupling did not work well when the interactions were dense, such as in the Indian Village problem. I also tried looking at cohesion as well as cohesion / coupling combined. In the main body of "Notes," Alexander shows what he calls "a typical graph" as part of his description of how to decompose design problems using a program. Here is that typical graph:

[7] Christopher Alexander, "A Result in Visual Aesthetics," British Journal of Psychology, Volume 51, Issue 4, November 1960.
[8] Richard Gabriel, "Notes on "A Result in Visual Aesthetics," https://dreamsongs.com/Files/Aesthetics.pdf.
[9] Being a randomized algorithm, HIDECS 2 runs these steps (random partition followed by hillclimbing) a number of times, choosing the best partition. My computer and version of this program can run these steps hundreds of times more than his could in a tolerable amount of time.



Every program I wrote and every program in the HIDECS reports can decompose this. By way of contrast, here is a visualization of the network of interactions for the Indian Village problem:



Once one starts to look for strongly cohesive clusters instead of loosely coupled ones in a dense network of interactions, overlap naturally occurs. I know that Alexander noticed this too. First, because playing with Alexander's earliest program and seeing it not do a good job or not doing a consistent job would lead anyone with curiosity to try alternatives. Second, because he said so:

> *HIDECS 2 has three important weaknesses:*
>
> *1. The fact that the decomposition is made in a series of binary steps leads to certain 'mistakes,' since the holistic relatedness of system and subsystems is not properly taken into account.*
>
> *2. The fact that the decomposition criterion INFO* [the goodness measure] *is based on very stringent assumptions about the nature of the system G(M,L). Namely, that the elements of M are binary variables, that the two variable correlations are very small, and that the many variable correlations vanish altogether. These assumptions make it hard to find systems in the real world which the formalism of HIDECS 2 can adequately represent.*
>
> *3. The fact that the subsets of elements which make the most natural subsystems of a system are not always disjoint, but often overlap.* [HIDECS 3]

In the HIDECS 3 report, Alexander addresses these flaws. He describes four programs. The first flaw is that by going top down, HIDECS 2 never looks at the total, fine-grained partition presented by the leaves of the binary tree. The approach in the first HIDECS 3 program is to start with a partition of the vertices into sets of single elements—for the Indian Village problem, this is 141 sets. The program systematically tries combining pairs of partitions, measuring the goodness of the entire partition; to do this, Alexander extended the HIDECS 2 measure. This produces a decomposition into disjoint sets, not a tree.

Alexander then observes a flaw with this program: a vertex with many links to a single other vertex in the same potential partition might be pulled into a different partition because it also has many single links to the vertices there. Returning to the family analogy, someone with many friends in another family might be considered a member of that family and not of their real family. The second program proceeds by starting with a partition into single-vertex sets and then systematically tries moving one vertex at a time from the set it happens to be in to each of the other sets, one at a time. Alexander also created a new goodness measure that looks only at cohesion—that is, to how strongly each vertex is linked to other vertices in the same potential partition. The algorithms using the earlier goodness measures try to minimize those measures—that is, minimize coupling; this algorithm tries to maximize the goodness measure—that is, maximize cohesion. Keep in mind he likely did not have available the named concepts of cohesion and coupling.

Once the first move was made to working with cohesion, Alexander moved more strongly in that direction. In 1957 a pair of researchers came up with an improvement to one of the first *clique-detection* algorithms: they were Frank Harary and Ian Ross; Alexander adopted this algorithm (by direct reference to their paper[10]) for the third and fourth programs in the series of four in the HIDECS 3 collection. The essential idea is that a partition is very strong when each vertex interacts with every other one—this is the definition of a clique. For example, if there are three vertices, each interacts with the other two; if there are four, each interacts with the other three. The third and fourth of the programs in HIDECS 3 are variations on

[10] Frank Harary and Ian C. Ross, "A Procedure for Clique Detection Using the Group Matrix," Sociometry, Vol. 20, No. 3 (Sept 1957).

this. In his typical graph, one can see three strongly interacting triangles of vertices; these are cliques.

Alexander noticed such tight cohesions in the HIDECS 2 paper and program. While partitioning a set into subsets, when the program notices such ***complete graphs,*** it does not try to subdivide them.

The Harary & Ross algorithm has flaws, as reported by Harary in his 1969 text, "Graph Theory." Instead of using that algorithm, I used a more modern one, the Tomita variant of the Bron-Kerbosch algorithm. In 1967, Edward Bierstone and Allen Bernholtz developed a semi-lattice recomposition program described in their report "HIDECS-RECOMP PROCEDURE."[11] I implemented that as well, and it can be used to visualize the various decompositions that start from clique detection.

Once I had all the bits of source code I needed to understand what the HIDECS programs were doing, my interest in improving the results faded, as I suspect it did for Alexander. It became clear that the original program, HIDECS 2, being a randomized algorithm, could spit out a different partition each time it ran, but that there was a limit to how well they would measure out according to the goodness measure. Moreover, as far as I know, Alexander never reported a complete partitioning of the Indian Village problem, and what he did report did not conform to what the program would produce. Namely, Alexander presented a decomposition of the full problem into four sets, each likely the union of two that were produced. This makes it a little difficult to judge how well his original program does compared to my modern version, which I wanted to do for the sorts of problems he described.

The basis for comparison was to use my recoding of his program running on modern hardware to try to reproduce results he recorded. Alexander wrote in the HIDECS 2 report:

> *…the program requires as input…LATIS, the number of starting sets for the hill-climbing algorithm to be chosen from the lattice…. The larger the value of LATIS selected, the more likely that the sampling procedure will discover the optimal TSET—but as the sample size increases, so does the amount of computer time used.* [HIDECS 2]

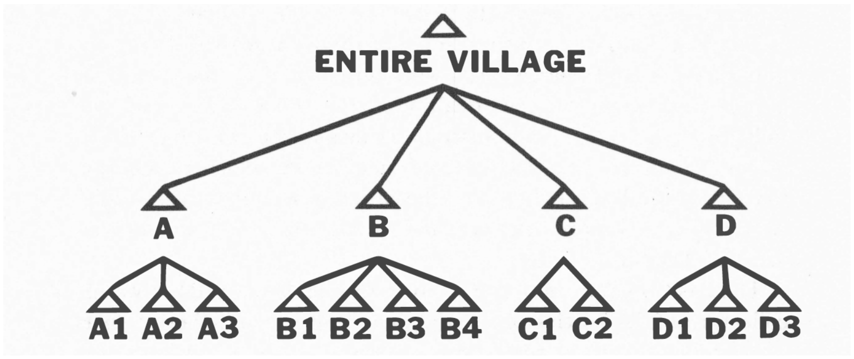
My program running on my computer can support values for LATIS 50–500 times larger than his could for a given expected duration of computation. For the goodness measure I decided to use the one he described in "Determination of Components," which is not quite the same as the one in the HIDECS 2 report, but it preserves ordering—if G_D is the measure in "Determination of Components" and G_H is the measure in HIDECS 2, then

$$G_D(\pi_1) < G_D(\pi_2) \text{ if and only if } G_H(\pi_1) < G_H(\pi_2)$$

[11] Bierstone, E & Bernholtz, A, "HIDECS-RECOMP PROCEDURE," Department of Civil Engineering, MIT, Cambridge, Massachusetts, March 1967.

where π_1 and π_2 are two partitions. G_D is the measure that produced the decomposition of the Indian Village problem as reported in both "Determination of Components" and "Notes."

In general the results for the Indian Village were that his program found partitions with worse goodness measures than mine. The only directly stated example of a partition into exactly two sets is the partition of C into C1 and C2:



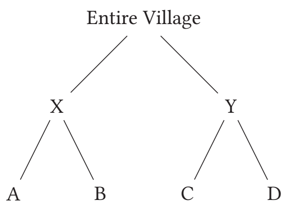
The goodness measures for Alexander's partition and the one my program produced using 250 times more starting sets is as follows, where smaller is better (−91 is better than −89):

	C
CA	−89.60
rpg	−91.60

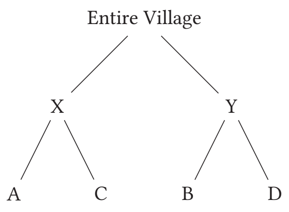
I discovered one extraordinary anomaly while looking at the top two levels of decomposition in "Notes."

I wanted to see how well Alexander's program did partitioning the Entire Village—the hardest partition of all. Alexander presents a partition of the whole problem into four sets. A, B, C, D. As noted, his program actually produces a binary partition of the whole problem (X,Y), and then each of those was further partitioned into two, yielding four. But, of the four shown, which two came from the same initial partition? That is, the Indian Village must have been partitioned in two sets, X and Y; does X=A+B, X=A+C, or X=A+D?[12] Here are the possibilities:

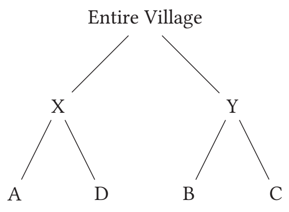
[12] "A+B" means the vertices of A and B are combined into a single set (set union). And note that if X=A+B, then Y=C+D, etc.



(a) Option 1



(b) Option 2



(c) Option 3

We know what vertices are in A because we know what vertices are in A1, A2, and A3: they are listed on page 151 of "Notes":

Group	Elements
A1	7, 53, 57, 59, 60, 72, 125, 126, 128
A2	31, 34, 36, 52, 54, 80, 94, 106, 136
A3	37, 38, 50, 55, 77, 91, 103

Similarly for B, C, and D. Therefore we know what vertices would be in X if X=A+B and in Y if Y=C+D, for example.

To find out which two came from the same initial partition, I tried all possible pairings—that is, I tried Options 1, 2, and 3—and the pairing that produced the best goodness for X and Y using the goodness measure is Option 1. For concreteness, here are the raw values (smaller numbers are better, so −645 is better than −562):

Option	Goodness
Option 1	−645.04
Option 2	−434.40
Option 3	−562.65

I guessed Option 1 was what Alexander's program did. Then I tried running my version of HIDECS 2 on the Entire Village; its result at the first level, X_1 and Y_1, measured out to −655.12—clearly better than all the options derived from Alexander's partition into four sets. I expected that if my program took that X_1, it would produce A_1 and B_1 that would measure out better than Alexander's A and B; and taking that Y_1, it would produce C_1 and D_1 that would also measure out better. This was naïve: the resulting partitions from my program were not much like Alexander's; it proved problematic to come up with an apples / apples comparison.[13]

While trying to figure out how to proceed, I ran an exhaustive pairwise computation of the goodness measure on Alexander's A, B, C, and D:

Pairs	**Goodness**
A & B:	−197.83
A & C:	−257.00
A & D:	−197.98
B & C:	−341.70
B & D:	−345.84
C & D:	−297.75

From this table I guessed that Alexander's program partitioned the Entire Village into X=A+C and Y=B+D. This is the worst of the three options. When I used those for starting points and derived my versions of A_2, B_2, C_2, and D_2, they were exactly the same as Alexander's.

Stated bluntly: the overall best partition (for A, B, C, D) is not necessarily obtained by doing the best job starting at the top and working down to get the best X and Y, followed by getting the best A & B from X and the best C & D from Y. This is possibly what Alexander meant in the first of his three observed weaknesses of HIDECS 2 as discussed in the HIDECS 3 report: "the holistic relatedness of system and subsystems is not properly taken into account."

Alexander's other HIDECS programs produced single levels of partition; some produced partitions with overlaps. In general the results shed confusing light on the Indian Village problem, and I believe this was how it seemed to Alexander.

During my investigations I was struck by the cold abstractness of the problem statement: 141 vertices and ~1400 links binding them together. However, these requirements came from real people and state real issues. Alexander writes:

> *All these misfit variables are stated here in their positive form; that is, as needs or requirements which must be satisfied positively in a properly functioning village. They are, however, all derived*

[13] My program partitioned X_1 into pairs with goodness −320.53 and Y_1 into pairs with goodness −173.39.

> *from statements about potential misfits: each one represents some aspect of the village which could go wrong, and is therefore a misfit variable*…. [“Notes”]

Moreover, the vertices are broken into 13 groups: *Religion and Caste; Social Forces; Agriculture; Animal Husbandry; Employment; Water; Material Welfare; Transportation; Forests and Soils; Education; Health; Implementation; Regional, Political, and National Development*; here is a selection from each group:

- 7. Cattle treated as sacred, and vegetarian attitude.
- 23. Men’s groups chatting, smoking, even late at night
- 36. Protection of crops from thieves, cattle, goats, monkeys, etc.
- 53. Upgrading of cattle.
- 65. Diversification of villages’ economic base—not all occupations agricultural.
- 67. Drinking water to be good, sweet.
- 79. Provision of cool breeze.
- 98. Daily produce requires cheap and constant (monsoon) access to market.
- 106. Young trees need protection from goats, etc.
- 112. Access to a secondary school.
- 125. Prevent malnutrition.
- 129. Factions refuse to cooperate or agree.
- 133. Social integration with neighboring villages.

In “Notes” Alexander writes:

> *Above all, the designer must resist the temptation to summarize the contents of the tree in terms of well-known verbal concepts. He must not expect to be able to see for every S some verbal paradigm like “This one deals with the acoustic aspects of the form.” If he tries to do that, he denies the whole purpose of the analysis, by allowing verbal preconceptions to interfere with the pattern which the program shows him. The effect of the design program is that each set of requirements draws his attention to just one major physical and functional issue, rather than to some verbal or preconceived issue. It thereby forces him to consolidate the physical ideas present in his mind as seedlings, and to make physical order out of them.*

While trying to reproduce the decomposition in “Notes,” I entertained the hypothesis that Alexander made it by hand, and that he looked at the realities expressed in the requirement statements. Some of my speculative, pre-HIDECS-informed programs took into account the 13 groups or various other groupings of them based on what they meant. And in fact, when Alexander describes his decomposition, he spins a story of how they are connected. Here is the start of one such:

> *The sacredness of cattle (7) tends to make people unwilling to control them, so they wander everywhere eating and destroying crops, unless they are carefully controlled. Similarly, the need to upgrade cattle (53) calls for a control which keeps cows out of contact with roaming scrub bulls; and further calls for some sort of center where a pedigree bull might be kept (even if only for visits); and a center where scrub bulls can be castrated. Cattle diseases (57) are mainly transferred from foot to*

> *foot, through the dirt—this can be prevented if the cattle regularly pass through a hoof bath of disinfecting permanganate….*

What can we learn from these investigations? Christopher Alexander's journey was of slowly dawning insights not a grand aha! He created flawed software that hinted at approaches to decomposition instead of reliably solving the problem. Although he did not have the concepts of cohesion and coupling as they are now known, he navigated the waters between them. He was not shy about using techniques and algorithms invented by others: some randomized algorithms already existed and were generally known in the late 1950s;[14] clique detection algorithms were known and Alexander acknowledges using one. Alexander and Manheim were not inept programmers—the HIDECS programs were written in assembly language and exhibited a sophisticated use of so-called "bumming" techniques.[15]

The HIDECS reports are not part of Alexander's formal publications; they are technical or research reports internal to a research organization—they are not very different from lab notes. It isn't fair to criticize based on deep analysis of such ephemeral materials.

However, it is fair to note the progression of thought from these very early investigations to those near the end of a career. Imagine the mind that progressed as follows:

> *The tree of sets this decomposition gives is, within the terms of this book, a complete structural description of the design problem defined by M; and it therefore serves as a program for the synthesis of a form which solves this problem….*
>
> *The organization of any complex physical object is hierarchical. It is true that, if we wish, we may dismiss this observation as an hallucination caused by the way the human brain, being disposed to see in terms of articulations and hierarchies, perceives the world. On the whole, though, there are good reasons to believe in the hierarchical subdivision of the world as an objective feature of reality.* [Notes]

That is originally from the early 1960s. Next from "A City is not a Tree"[16]:

> *For the human mind, the tree is the easiest vehicle for complex thoughts. But the city is not, cannot and must not be a tree. The city is a receptacle for life. If the receptacle severs the overlap of the strands of life within it, because it is a tree, it will be like a bowl full of razor blades on edge, ready to cut up whatever is entrusted to it. In such a receptacle life will be cut to pieces. If we make cities which are trees, they will cut our life within to pieces.* [ACINAT]

Around the same time, in an essay:[17]

[14] Simulated annealing, which I used in this investigation, was invented in 1953 by Nicholas Metropolis.
[15] Bum: "to make highly efficient, either in time or space, often at the expense of clarity."
[16] Christopher Alexander, "A City is Not a Tree," Architectural Forum, Vol 122, No 1, April 1965.
[17] Christopher Alexander, "On Value," Concrete 1965.

> *Myself, as some of you know, originally a mathematician, I spent several years, in the early sixties, trying to define a view of design, allied with science, in which values were also let in by the back door. I too played with operations research, linear programming, all the fascinating toys, which mathematics and science have to offer us, and tried to see how these things can give us a new view of design, what to design, and how to design.*
>
> *Finally, however, I recognized that this view is essentially not productive, and that for mathematical and scientific reasons, if you like, it was essential to find a theory in which value and fact are one, in which we recognize that here is a central value, approachable through feeling, and approachable by loss of self, which is deeply connected to facts, and forms a single indivisible world picture, within which productive results can be obtained.* [ON VALUE]

Then in "The Nature of Order, Book 4":[18]

> *The I, that blazing one, is something which I reach only to the extent that I experience, and make manifest, my feeling. What feeling, exactly? What exactly am I aiming for in a building, in a column, in a room? How do I define it for myself, so that I can feel it clearly, so that it stands as a beacon to steer me in what I do every day?…*
>
> *What I aim for is, most concretely, sadness. I try to make the volume of the building so that it carries in it all feeling. To reach this feeling, I try to make the building so that it carries my eternal sadness. It comes, as nearly as I can in a building, to the point of tears.* [NoO]

We see in the early mind what the mind became. When we read the backstories in the HIDECS reports and read carefully the words in his formal publications, we learn that the reality of the computer and the poverty of programming languages were stern teachers, teaching Alexander that cold abstraction requires a warm human hand and experienced (tear-filled) eyes, that machines can be partners for exploration, and that a city is not a tree.

[18] Christopher Alexander, "The Nature of Order, The Luminous Ground," Center for Environmental Structure, Berkeley, CA, USA, 2004; Volume 4.