WHAT MAKES AND DOESN'T MAKE A 'KILLER APP' IN CIVIL ENGINEERING: A RETROSPECTIVE EVALUATION

Steven J. Fenves

Manufacturing Engineering Laboratory, National Institute of Standards and Technology, Gaithersburg MD, USA
Civil and Environmental Engineering Emeritus, Carnegie Mellon University, Pittsburgh PA, USA

1 INTRODUCTION

Every software developer, from the individual amateur to the largest enterprise, dreams of giving rise to a "killer application (commonly shortened to killer app) that is so useful or desirable that it proves the value of some underlying technology"[1]. Whether the 'killer app' provides financial benefits either to the developer or the hardware platform vendor is beside the point. The important thing is the professional or social component: a true 'killer app' radically alters some form of human activity, either by creating an activity that did not exist before, or by improving the performance of an activity so dramatically that its practitioners view it as a revolutionary change. The first set of 'killer apps' so named, the early spreadsheet programs VisiCalc and Lotus 1-2-3, certainly revolutionized finance, accounting, engineering and many other professional disciplines. These programs, in fact, engendered the pursuit of the dream referred to above.

The title of the talk is not "How to create a killer app in Civil Engineering" but "What makes and doesn't make a killer app in Civil Engineering." Forecasting is always a tough art. Given the wide range of human activities, it is even tougher to predict what tool will radically alter one such activity. Retrospective appraisal is much easier: you just need to evaluate what happened and attempt to trace from causes to consequences. Furthermore, because of the rarity of 'killer apps' generally, and in civil engineering particularly, it is not possible to treat the subject in any generic way; it can only be treated by evaluating examples and attempting to generalize from them.

This talk will examine two themes that took up a major share of my professional concerns. One quickly developed into a 'killer app', perhaps not of the same scope as spreadsheets, but making a significant change in one aspect of civil engineering[2] practice. The other, which actually occupied a much larger part of my working life, never led to any kind of substantial change in the tools engineers use. I, and a few of my students, have often wondered about this disparity. This talk gives me the opportunity for making a comparative evaluation between them.

2 THEME 1: STRUCTURAL ANALYSIS

Setting #1: Fall 1961, University of Illinois.

I had defended my Ph. D. dissertation just before the semester began and was wondering what to do next. Experimental structural dynamics, the topic of my dissertation, held no interest. I had written a number of specialized frame analysis programs as a consultant and even written a proposal to IBM for a general analysis program but there was no response. Then I got hold of Professor Charles Miller's report on COGO (1). That was it! I wrote Professor Miller, saying that I wanted to provide for structural engineers what he had provided to surveyors: a general problem-solving capability for the domain, invoked by terms that a professional in the field would use in giving instructions to a colleague[3]. In short order, Charlie Miller wrote back inviting me to MIT for the 1962 – 1963 academic year. With the concurrence of my department head, Professor Nathan M. Newmark, I accepted the invitation and started to formulate ideas.

The basis

Structural analysis is used to determine the internal forces and displacements in a structure due to applied loads or other environmental effects by the application of equilibrium, continuity and constitutive laws. The field was inaugurated by Galileo and has expanded ever since. Analy-

[1] Definition from Wikipedia, another "killer app'

[2] Both of the themes covered deal with structural engineering design activities. I follow US practice of treating structural engineering as part of the broader profession of civil engineering.

[3] Younger people need to be reminded that in that period, and long after, commands, choices and decisions were invariably communicated to programs coded as integers. Looking at the echo-print of the output, a user could not determine what commands, etc., he or she used without consulting the program's user manual. For example, in one of the most popular finite element analysis programs, well into the 1980's, the user specified elastic analysis by setting the input parameter ISOLV to 1.

sis has always been one of the most labor- and skill-intensive aspects of structural design. Structural engineers have compensated for this fact in many ways: designing structures that were easy to analyze, e. g., statically determinate beams and trusses that could be analyzed by laws of equilibrium alone; embracing simplified analysis methods, particularly graphic statics and Williot-Mohr diagrams for determining forces and displacements, respectively, in trusses; specialization, e. g., consulting firms specializing in movable, arch or suspension bridges; and developing approximate and iterative solution methods, such as moment distribution[4].

The majority of existing matrix analysis programs at the time was geared to very large problems in aerospace and defense that required extremely large computing resources. Thus, the NASA frame analysis program could analyze very large structures for that time, of the order of hundreds of nodes, using an IBM 7090 and twelve tape drives for temporary storage. The trouble was that in order to solve a three-bar truss the program still needed all twelve tape drives. Very general matrix analysis programs were also becoming available (3, 4). The chokepoint was the amount of input: every matrix element had to be precomputed manually and keypunched separately. This was judged to be the appropriate level of human - computer interaction at the time[5].

Smaller programs for the small computers of the time were emerging, but they tended to be highly specialized[6]. Many of these faithfully implemented existing methods such as the slope-deflection method (2). These programs also tended to contain idiosyncratic provisions particular to the design office they originated from[7]; consequently, program exchange among the civil engineering firms pioneering computer use at the time was more myth than practice.

The run-up

As I prepared for our move to Cambridge, Massachusetts, with my wife Norma and three small children, I thought that I had identified all the principal requirements for the tool I was to build:

1. a problem-oriented textual input language patterned after COGO
2. flexibility in problem size, so that the solution of small problems would not be penalized by the program's capability of solving very large problems
3. complete generality in handling various framed structure types, e. g., frames or trusses and
4. complete generality of methods.

It did not occur to me that I may not have had all the knowledge and tools at my disposal for accomplishing my objectives. I soon found out that my knowledge was sorely lacking, but I also found the source of the knowledge I needed. The IBM sales representative at the University of Illinois arranged for me a faculty summer internship at the IBM Development Laboratory in Poughkeepsie, New York, on the way to Cambridge. There I fell under the tutelage of Dr. Frank H. Branin, Jr., an outstanding mathematician as well as the author of ETABS, an early general-purpose analyzer for RLC electrical networks. From the first half of his expertise I learned the topological formulation of network analysis. Matching the nodal method of network analysis to the stiffness method of structural analysis (then also called the displacement method) and thereby generalizing it took less than a week. Matching the mesh method to the flexibility (force) method took considerably longer, until I realized the conceptual analogy between the spanning tree of a network and the statically determinate primary structure[8] (5). From the other half of Frank's vast expertise, I learned dynamic memory allocation and many other tools for developing large-scale programs.

The process

On arrival to MIT in September 1962, I quickly realized that Charlie Miller was so engaged in his new position as department head that any active collaboration with him was out of the question. However, I met the partners Charlie had selected for me. We found that we were very compatible and we quickly made the task allocations that held up for the year. I laid out the processing of the problem-oriented input[9], the overall process for the stiffness method, the provisions for the five structure types we were to support (planar trusses, frames and grids and space trusses and frames) and the dynamic memory allocation scheme. Bob Logcher, a recent Ph. D. who was as anxious to break new grounds as I was, undertook several of the major system tasks, including the decoding of the command language and the management of backup storage. Ken Reinschmidt, an IBM graduate fellow in the Computer Center eager to move up from being a user consultant, undertook the bulk of the equation forming, solution and back-substitution tasks within the dynamic memory allocation environment. Leon Wang, another graduate student, wrote the member-related routines. At mid-year, we were joined by another fresh Ph. D., Sam P.

[4] When I was an undergraduate in Civil Engineering in the early 1950's, our instructors repeatedly emphasized that our employability in the field was going to be determined solely by the speed with which we could do moment distribution.

[5] Then called man-machine interaction; as late as 1967, a distinguished colleague insisted that entering the 36 elements of a 6x6 transformation matrix from local to global coordinates was the proper way for the user to define the relationship between the two coordinate systems.

[6] The Illinois Highway Department, for example, had separate programs for three- and four-span continuous girder bridges; the department did not build other types of continuous bridges.

[7] The program I wrote in 1957 for the analysis and detailing of reinforced concrete bridge piers set the maximum spacing of stirrups in the columns to 12 inches, even in areas of low shear. My supervisor insisted that this was necessary so that ironworkers tying the reinforcing bars into cages could climb up on the stirrups they had tied previously. With the adoption of pre-tied rebar cages erected as units, this heuristic provision is no longer needed.

[8] Younger people will wonder about the emphasis on the flexibility method, now almost totally ignored – the explanation will follow shortly.

[9] Using the despised integer codes until Bob Logcher was ready to map the user commands into them.

Mauch, who quickly became the chief debugger[10]. My wife Norma's contribution was the program's name: she thought that the word STRESS had the proper ring to it; mapping STRuctural Engineering System Solver to the acronym was an evening's effort.

We progressed remarkably fast, given the two to three debug opportunities a day available at that time and the fact that Charlie had asked me to teach a course both semesters I was at MIT. Getting the dynamic memory allocation working gave us the most satisfaction[11]. The overall program turned out to be quite different from COGO: instead of being a command language with individually executable commands (e. g., LOCATE POINT, ADJUST TRAVERSE), it was a data description language (e. g., JOINT COORDINATES, MEMBER PROPERTIES) with one "executable" command, SOLVE, at the end.

We felt that we had satisfied the first three requirements I have set for myself. The fourth, complete generality of methods was another story. The original STRESS had a command, METHOD STIFFNESS, and I had expected that many other methods (flexibility, moment distribution, column analogy, method of joints, etc., etc.) would be added in time. Chris Holley, my mentor at MIT, convinced us that many of these methods were crutches of the past and did not need to be perpetuated. Thus, stiffness was the only method implemented and the command itself was soon dropped[12].

We got many things right, but also some things wrong and there were many things we just didn't know about. One bad decision was the choice of the notorious "β angle" for fully defining the orientation of the member cross-section in space. A significant flaw was that the program assumed that only one member connected any two joints, but did not check for this. This condition could be detected, however, by the presence of large residuals loads at joints that were supposed to be free to displace[13]. Bandwidth or infill minimization, frontal solving, etc., etc. were all unknown at the time.

A glimpse of the future

In the spring of 1963 MIT embarked on Project MAC, an early experiment in timesharing. Bob Logcher and I went to see what this was about. When we asked what Project MAC was going to be used for, Fred Corbato, the project director, said it was for rapid debugging of programs that could then run in the background. Bob and I looked at each other as we walked out. In a week, Bob had an interactive version of STRESS running[14]. That was our first glimpse of the "second computer revolution" to come.

The outcome

My family and I returned to Illinois in the summer but I was back in the fall to complete the editing of the manuals (6, 7) and to conduct a week-long workshop on STRESS. The reaction of the practitioners attending was most positive and the program's fame began to spread rapidly. Bill LeMessurier, a prominent Boston structural engineer[15], described STRESS as a brilliant but patient graduate student unstintingly executing any analysis task you posed.

The following year STRESS achieved the Wikipedia definition of a 'killer app.' IBM was marketing a new small computer, the 1120, to the engineering community. A number of engineering firms affiliated with the CEPA (Civil Engineering Program Applications) user group said that they would buy or lease the computer only if it supported STRESS. Thus was the IBM 1120 STRESS born[16] and thus was a first generation of structural engineers introduced to matrix structural analysis.

The legacy

By 1963, the MIT Civil Engineering Department had embarked on the ambitious ICES (Integrated Civil Engineering System) mega-project (8). Under Bob Logcher's leadership, STRESS metamorphosed into STRUDL (STRuctural Design Language), with many notable additions, expansions and generalizations (9). Some of these did not survive beyond the MIT environment, among them an elegant PRELIMINARY ANALYSIS capability[17]. Eventually, STRUDL became just another finite element analysis tool, and the geometric modeling and graphic user interface capabilities of these tools have rendered the problem-oriented textual input unnecessary.

Of the many experiences related to STRESS, I will briefly relate two.

Around 1964, I received a request from Fazlur Khan to assist the team at the architectural firm of Skidmore, Owings and Merrill (SOM) in the analysis of the structure that became the John Hancock Tower in Chicago. In his writings, Fazlur always referred to my role as assisting in

[10] Sam was so impatient with the slow FORTRAN compilers of the day that he preferred to work with binary patches directly. Unfortunately, he sometimes neglected to update the FORTRAN source code to reflect the patches. For many years, the most commonly used version of STRESS produced erroneous results for sloping members connected to a support by a hinge and carrying member loads. Although it was Sam who produced the original formulation (21), he never transcribed one of his correction patches into the source code.

[11] This was at least six years before the first paged memory computer became available.

[12] Everywhere but at the University of Illinois, where the version of STRESS implemented in the Civil Engineering Systems Lab had a rule: "if userid = sjf then a missing METHOD STIFFNESS command is a fatal error, else ignore its absence."

[13] The detection was made easy by my decision that when the user requested PRINT REACTIONS, a full backsubstitution was made and residuals at all joints were printed out.

[14] In fact, the interactive program was much simpler than the batch one because the latter continued checking the input after a fatal error was encountered in order to maximize the value gained from each run.

[15] Subsequently Bill gained considerable notoriety with his design and retrofit of the CityCorp building in New York.

[16] IBM, of course, took the easy way out and adopted the program written the previous summer by an MIT undergraduate, Dick Goodman. On the small computer, dynamic allocation was out: there was room only for three 6x6 matrices, all else was done by bookkeeping. Unfortunately, the 1120 version propagated the bug created by Sam Mauch's impatience.

[17] Eventually, the maintenance and updating of STRUDL devolved from MIT to Georgia Tech, under the leadership of Leroy Emkin.

the design (10). I did no such thing. By the time I had modified my version of STRESS to accept two-thirds of the height of one quadrant of the building, Fazlur's team had proceeded through the design using a graded series of increasingly more detailed two- and three-dimensional models, so that the STRESS run was just a final verification of the team's design process. Approximately 20 members out of the more than 900 had to be resized as a result of the STRESS space frame analysis. What made the event memorable was that a prominent structural engineer gave a speech in Chicago saying "Steve Fenves' program says the tower will stand up, I say it will not." I knew that that engineer only had a space truss program to back up his claim, but I could not get the SOM partners to put up $600 to rerun the analysis as a space truss and verify the engineer's claim[18].

In the early 1970's, I received a letter from the Florida state engineering licensing board. On the Professional Engineer exam, candidates were presented with a one-story one-bay frame and told to analyze it. One candidate wrote out the STRESS input and stated that the results of this provided the answer sought. The board wanted my opinion on whether the candidate should be passed. I wrote back that it depended on what they wanted to test for: if they were only after concepts, the candidate's reply sufficed; if they wanted to test the candidate proficiency in producing the results and, possibly, evaluating them for reasonableness, then the candidate failed. I never received a reply from the board.

In retrospect, I take a great deal of pride in what we accomplished and what our work fostered, even though we did not receive any financial rewards from it. On the other hand, it is sobering to reflect that my most significant contribution to the profession was made when I was 32 years old.

3 THEME 2: STANDARDS PROCESSING

Setting #2: Summer 1965, National University of Mexico.

My family and I had gone to Mexico for the summer for me to teach a couple courses at the National University and to complete the manuscript of *Computer Methods in Civil Engineering* (11). In preparing the book I could include many notable names in the chapters on numerical methods: Newton (1642 – 1727), Lagrange (1736 – 1813), Gauss (1777 – 1855), etc., but there was a dearth of precedents to present in the chapter on logical methods I wanted to include. Then I got hold of a paper on tabular decision logic, commonly referred to as decision tables (12). That was it! It was a good first step towards modeling the often complex chain of reasoning in both engineering and information processing. In my first paper on the topic, I illustrated decision tables by modeling a small set of provisions from the AISC Specification (13).

[18] As an indication of the rapid progression of the state-of-art, a few years later STRUDL could analyze the entire Sears Tower as a unit without recourse to symmetry and superposition.

The basis

"Design according to the code" is a misunderstood, yet frequently maligned process. Buildings must provide for the life and safety of their occupants; since antiquity building codes have been used as society's policing mechanism to that purpose[19]. The term "building code" itself is misunderstood, particularly in the U. S., where the enactment and enforcement of the legally binding building codes are the responsibility of individual jurisdictions. In the U. S., since the early part of the 20th century a three-tier structure has evolved to assist the municipalities in this task. First, standards development organizations, often sponsored by professional societies, develop design standards or design specifications for common materials or building types[20]. Second, model code organizations evaluate, edit and compile this information in the form of model codes ready for adoption. Third, jurisdictions adopt all or parts of the model codes, with or without local amendments.

That is the legal aspect. The professional aspect, embodied in the design standards and design specifications, is much more interesting. In the dispersed and discontinuous industry that civil engineers serve, there is little memory within individual organizations. Here, the standards and specifications serve essentially as the "collective memory" for the profession as a whole of what worked in the past and, even more significantly, what has not worked[21]. The standards and specifications are also the primary outlet of research in structural behavior. This role was most convincingly argued by Chester P. Siess, a former chairman of the American Concrete Institute building code committee, where he characterized research, standards and practice as three node of a graph, with the major flow of knowledge going from research into the standards and hence affecting practice[22] (14).

Thus, "design according to the code" means ascertaining that (1) the requirements and limitations presented in the design standards and specifications are applicable to the structure or part being designed and (2) the applicable provisions are satisfied by the design. The first, interesting, part of this process is to select key requirements and convert the inequalities to assignments of values to key parameters. The second, manifestly dull, part is the processing of all other applicable requirements to ascertain that they too are satisfied. The costly part is the discovery that some requirements are violated, indicating that the key requirements have not been properly chosen and that the structure or part of it has to be redesigned, reanalyzed and rechecked. The first part requires experience and expertise. The second part is much more routine, but in

[19] The earliest building code, part of the 282 laws promulgated by Hammurabi (*ca.* 1810 BCE – 1750 BCE), prescribed harsh punishments to builders if their buildings' collapse caused deaths.

[20] The initial design standards were vehemently denounced by the leading engineers of the period as infringements on the designers' professional responsibility to their clients.

[21] After every major disaster, natural or man-made, every affected standards development organization empanels a group of experts to identify flaws in their standards and recommend remedies.

[22] Following ACI practice, Siess uses the term building code to mean design specification.

terms of total billable hours in a design office it is likely to be of the same order of magnitude as analysis.

By the mid-1960's, there were many small programs for the design and partial standards conformance checking of structural components. They all tended to be highly idiosyncratic to the mode of operation of particular design offices. Furthermore, they all relied on individual interpretation of the governing standard's provisions. Most importantly, none of the standards development organizations had made any accommodations for the trend of interpreting and coding the standards provisions into computer programs.

The run-up

Unlike in the first scenario, I had no immediate plans for further work in standards processing. However, my paper came to the attention of Theodore (Ted) Higgins, the Director for Research of the American Institute of Steel Construction (AISC). Ted invited me to attend a meeting where several software vendors made proposals for an "electronic version" of the then forthcoming 7th Edition of the AISC Specification. For me it was "déjà vu all over:" every potential vendor firm proposed to provide its own engineering interpretation of the standard's provisions and to hard-code that interpretation into one or more computer programs. There was no way for either AISC or the user community to ascertain that the programs thus crafted correctly and fully implemented the Specification's provisions.

After the meeting I made a counterproposal to AISC to: (1) have us formally represent the AISC Specification provisions in the form of decision tables; (2) have the Committee on Specifications authenticate that the representation is complete and correct; and then (3) let anyone who wishes to do so, whether user or software vendor, code programs from the formal representation. Ted Higgins was greatly concerned that our analysis would reveal overlaps, contradictions and omissions in the Specification, but he saw the potential and was sufficiently intrigued to have AISC fund us for the first step.

The process

The project staff consisted of my senior colleague, Ed Gaylord[23], two graduate students and me. Our routine was that the students would draft one or more decision tables covering a provision, I would check these for logic and consistency, and then we would jointly go to see Ed. Our questions to him were invariably of the form: "Why is the following combination of conditions not covered?" "Because we don't do that!" was Ed's immediate answer. Then we asked whether that was because that combination was: (1) physically not realizable; (2) bad practice to be discouraged; or (3) to be proscribed by the Specification. Ed would provide the reasoning and we would fill in the tables accordingly. Thus, slowly, we elucidated and completed the individual decision tables and thereby significantly clarified the contents and scope of the Specification. Ted Higgins' concerns turned out to be unfounded. We found very few overlaps, inconsistencies and overt omissions; however, the boundaries of what was covered and what was not were frequently not spelled out precisely.

Linkages between tables representing related provisions turned out to be cumbersome, therefore we recast all the tables so that each table produced values for only one variable, like a function, and we linked the tables into a network, lower-level tables producing values for ingredients to higher-level tables. Because of the way that the organization of the Specification evolved since its first edition in 1924[24], the linkages jumped haphazardly from section to section. The recursive program we wrote that executed the network frequently had to recurse six and at times even eight levels deep[25] (15). Thus it became clear that organization of the Specification was as much an issue as the completeness of the individual provisions.

The outcome

The initial acceptance came about through a lucky coincidence. When I was asked to present our work to the Committee on Specifications, I brought along transparencies of several of the tables. First I had to sit through a committee meeting, where the committee debated how a certain provision would be expanded. As I was given the floor, I projected the decision table for the provision in question, which I just happened to have with me, on the screen. I showed how the committee's discussion resulted in one rule being split in two with an added condition, producing two different actions. Then I asked whether they were sure that this was the only rule affected by the new condition. "Oh, yeah!" came the exclamation. The committee went back into session and made another change, previously overlooked. Our funding for a second study was assured. But the idea of the committee authenticating our representation never got off the ground, for two reasons. One, we did not do a complete job: a lot of textual, descriptive material in the Specification did not lend itself to a decision table format and we simply ignored it. Two, we presented to the committee a new representation that was very different from the text they were used to, and they were not ready to act on it.

A number of colleagues produced similar decision table formulations for other standards. There is evidence that a number of software developers used our formulation in developing their applications[26]. The follow-up study looked at ways that the Specification as a whole may be

[23] Ed was a professor of the old school: a popular teacher, my mentor when I started teaching, the author of a widely used steel design textbook and a member of the AISC Committee on Specifications; but he had not done research previously.

[24] In manual processing, this would be equivalent to interrupting the processing of a provision, leaving a bookmark and proceeding to process the provision specifying a missing ingredient only to interrupt that, leaving a second bookmark, etc., until one can return to the last bookmark, resume processing there, etc., till at the end one could ascertain whether the initial provision is satisfied.

[25] The first edition carried an endorsement from Herbert Hoover, then Secretary of Commerce.

[26] There is also anecdotal evidence that after some lectures in the graduate steel design course the students would check out our report from the library in order to understand what the lecture was about.

organized in a more logical fashion (16). Thus emerged the eventual three-tier model of the Specification: (1) trees of descriptors defining the organization; (2) data networks for relating and sequencing interrelated requirements and determinations; and (3) decision tables for expressing individual requirements and determinations (17, 18, 19).

The reorganization study recommended that a reorganization of the Specification be undertaken only in conjunction with a major change in the design philosophy embodied in the Specification. Such a change came about a few years later with the introduction of the Load and Resistance Factor Design (LRFD) philosophy and the accompanying rationalization of component behavior models. We worked with the LRFD development team lead by Ted Galambos, providing draft decision tables and suggesting draft organizations. Upon completion, Galambos' study was turned over to the AISC Committee on Specifications, which charged nine existing Task Committees with converting the study into nine chapters of a new edition of the Specification. When the draft chapters came in, there was pandemonium: the organization and style of presentation of each chapter, reflecting the long history of each Task Committee, was radically different. Eventually, the committee appointed me as editor and, after many iterations, a new, logically and consistently organized and expressed LRFD Specification emerged[27].

There was one more attempt at an electronic version that could be sanctioned by AISC, with Bill McGuire, another senior member of the Committee on Specifications, as the domain expert (20). The program development tools had improved considerably. We produced an interactive version to be used by the committee for editing and tracing the consequences of proposed changes, a tutorial version with affected conditions, rules and actions highlighted as the reasoning progressed, and a production version generator which could compile user-selected sets of decision tables into C^{++} code. Unfortunately, legal issues over intellectual property prevented deployment of the tools. Thus, the Committee on Specifications did not have to make a decision on whether it would authenticate our representation[28].

The legacy

No 'killer app' emerged from this work, not even a conventional application. The legacy is very small. True, a well-crafted Specification emerged and survived several major revisions. I am still chairing the Editorial Task Committee of the AISC Committee on Specifications, and just this year I received the AISC Lifetime Achievement Award.

[27] With thanks to Kincho Law for producing the various iterations on an unreliable daisy-wheel printer.

[28] My sense, unsubstantiated, is that many members of the Committee would have extensively exercised the tutorial version, but that the Committee as a whole would still not have provided an endorsement.

4 COMPARISON AND CONCLUSION

I have often wondered about the reasons for the discrepancy between the two narratives above, but I still do not have a clear answer. As I said, the amount of time devoted to the two processes in a design office is of the same order of magnitude. However, one used to be the reserved province of highly esteemed analysts. Analysis programs that followed STRESS have democratized the profession, largely reducing the specialization factor among offices as well as among engineers within these offices, while at the same time enabling engineers to model structural behavior much more extensively than the linear behavior model in the original STRESS formulation[29]. The second process was and still is much more dispersed within structural engineering practice, and there are far fewer identifiable experts.

A second factor may be related to the structural engineers' education. From their sophomore year on, students know the difference between design and analysis. Furthermore, by their junior year most of them also "know" that they don't like analysis. Even though analysis courses have completely changed from teaching approximate methods to the current formal ones, students avoid doing either kind of analysis as much as possible and are only pleased to relegate this work to a machine. In contrast, design courses tend to concentrate on global measures of structural behavior and may cover standards processing only in a tangential fashion. It may be that students don't get a chance to discover that they don't like standards processing.

Finally, I believe that the distinction has to do with the formalism of the subject or, more precisely, the degree to which the subject can be formalized. The early days of computer analysis gave rise to the discipline of computational mechanics, just as the early days of computer graphics gave rise to computational geometry and the early file manipulation programs to database theory. The processing of standards, by contrast, is rooted in the highly empirical world of observations of what has worked and what has not. Even with the current move towards performance-based standards, the subject does not have the formalism or rigor that has, so far, made it a discipline[30].

To the dreamers who wish to parlay their engineering research into the next 'killer app' I have a simple advice: unless your research creates or contributes to a solid science-based discipline, you have no reason to expect that it will result in a 'killer app.' In this respect, engineering seems to be different from data processing and entertainment, where such a precondition does not seem to be present.

[29] The best accolade I ever received came from a senior colleague some years ago, when suddenly at a formal dinner he asked me what I would like to see written on my epitaph. I said that I had no idea, and asked him what epitaph he would write. He responded "You have made the cost of analysis immaterial in making design decisions." I gratefully accepted that.

[30] I am an Honorary Member of the Executive Council of the International Association for Computational Mechanics; there is no comparable organization or honor in standards processing.

ACKNOWLEDGEMENTS

I wish to a thank Yoram Reich and E. (Sub) Subrahmanian for the fruitful discussions on the subject and their critical comments on early drafts of this paper.

REFERENCES

(1) Miller, Charles L., COGO, Department of Civil Engineering Report, MIT, Cambridge, MA, 1961.
(2) Barneson, R., Rigid-Frame Analysis with a Digital Computer, Proceedings 1st Conference on Electronic Computation, American Society of Civil Engineers, New York NY, November 1958, pp. 267 – 278.
(3) Clough, R. W., Structural Analysis by Means of a Matrix Algebra Program, Proceedings 1st Conference on Electronic Computation, American Society of Civil Engineers, New York NY, November 1958, pp. 109 – 132.
(4) Shore, S., The Elements of Matrix Structural Analysis, Proceedings 2nd Conference on Electronic Computation, American Society of Civil Engineers, New York NY, September 1960, pp. 145 – 164.
(5) Fenves, S. J. and Branin, F. H. Jr., Network-Topological Formulation of Structural Analysis**,** Journal of the Structural Division*,* American Society of Civil Engineers*,* Vol. 89, No. ST4, August, 1963, pp. 483-514.
(6) Fenves, S. J., Logcher, R. D., Mauch, S. P. and Reinschmidt, K. F., STRESS - A User's Manual, MIT Press, Cambridge, MA, 1964.
(7) Fenves, S. J., Logcher, R. D. and Mauch, S. P., STRESS - A Reference Manual, MIT Press, Cambridge, MA, 1965.
(8) Miller, C. L., Man-Machine Communication in Civil Engineering, Journal of the Structural Division, American Society of Civil Engineers, New York NY, Vol. 89, No. ST4, August 1963, pp. 5 – 30.
(9) Logcher, R. D. and Sturman, G. M., STRUDL – A Computer System for Structural Design, Journal of the Structural Division, American Society of Civil Engineers, New York NY, Vol. 89, No. ST4, August 1963, pp. 191 – 212.
(10) Khan, F. R., Iyengar, S. H. and Colaco, J. P., Computer Design of 100-Story John Hancock Center, Journal of the Structural Division, American Society of Civil Engineers, Vol. 92, No. ST6, December 1966, pp. 55 – 74.
(11) Fenves, S. J., Computer Methods in Civil Engineering, Prentice-Hall, Englewood Cliffs, NJ, 1967.
(12) Montalbano, M., Tables, Flow Charts and Program Logic, IBM Systems Journal, White Plains NY, Vol. 1, No. 1, September 1962.
(13) Fenves, S. J., Tabular Decision Logic for Structural Design, Journal of the Structural Division*,* American Society of Civil Engineers, New York NY, Vol. 92, No. ST6, December, 1966, pp. 473-490.
(14) Siess, C. P., Research, Building Codes, and Engineering Practice, Journal of the American Concrete Institute, Detroit MI, Vol. 31, No. 11, May, 1960, pp. 1105 – 1122.
(15) Goel, S. K. and Fenves, S. J., Computer-Aided Processing of Design Specifications, Journal of the Structural Division*,* American Society of Civil Engineers*,* New York NY, Vol. 97, No. ST1, January, 1971, pp. 463-479.
(16) Nyman, D. J. and Fenves, S. J., Organization Model for Design Specifications, American Society of Civil Engineers, New York NY, Vol. 101, No. ST4, April, 1975, pp. 697-716.
(17) Fenves, S. J. and Wright, R.N., The Representation and Use of Design Specifications, Technical Note 940, National Bureau of Standards, Washington, DC, 1977.
(18) Wright, R. N., S. J. Fenves and Harris, J. R., Modeling of Standards: Technical Aids for their Formulation, Expression and Use, Technical Report NBSIR 80-1979, National Bureau of Standards, Washington, DC, March 1980.
(19) Garrett, J.H., Jr. and Fenves, S. J., A Knowledge-Based Standards Processor for Structural Component Design, Engineering with Computers, Springer Verlag, New York NY, Vol. 2, No. 4, 1987, pp. 219-238.
(20) Ackroyd, M. J., Fenves, S. J. and McGuire, W., Computerized LRFD Specification, Proceedings AISC National Engineering Conference, American Institute for Steel Construction, Chicago IL, June, 1988.
(21) Mauch, S. P. and Fenves, S. J., Releases and Constraints in Structural Networks, Journal of the Structural Division*,* American Society of Civil Engineers, New York NY, Vol. 93, No. ST5, October, 1967, pp. 401-417.