In P. Romero, J. Good, E. Acosta Chaparro & S. Bryant (Eds). Proc. PPIG 17 Pages 131 - 144

The Psychology of Invention in Computer Science

Ronald J. Leach, Caprice A. Ayers

Department of Systems and Computer Science
Howard University
Washington, DC 20059
rleach@howard.edu, cayers@howard.edu

Abstract.

Much of the existing work on the psychology of programmers has been experimental. The purpose of this initial study is to help provide some avenues for future experimental research by addressing creativity in computer scientists. We studied several writings and published interviews with a number of prominent computer scientists in an effort to understand the nature of creativity in computer science and to develop a set of research questions to be answered about the thought processes of programmers. The intent was to identify similar factors of their experiences that may have contributed to their success. Parallels to some existing work on the nature of creativity in mathematics and architecture are also made. The paper concludes with a discussion of how the study of common experiences of creative research computer scientists can be extended to study the creative processes of programmers.

Introduction

Creativity in fields such as astronomy, music, mathematics, and engineering has long been studied. In the nineteenth and early twentieth centuries, it was common for scientists and mathematicians to publish expository papers, educating the masses and discussing creativity in their domains. Computer science derived initially from a combination of mathematical and electrical engineering principles, so one might expect similar patterns of creativity. While the majority of computer scientists have devoted their attention to solving complex problems and introducing new technology, few have written on the thought process behind their creations. We describe some of the relatively sparse literature here.

Research plans of most people active in the field of analyzing programmer behavior include one or more of the following: case study, comparative analysis, and controlled experiment. Often, the process is iterated, with additional studies and experiments as new information and analysis suggest other research questions to be examined.

This paper sits earlier in the research process, gathering information, some of which is anecdotal, about the thought process of computer scientists who have made significant contributions to software. The paper is directed toward helping researchers

develop new hypotheses on creativity of both leaders and followers in the computer science and software engineering communities.

We reviewed several papers and anecdotal work of several prominent computer scientists for details on how they reached many of the conclusions proposed in their work. The intent was to identify similar factors of their experiences, including education, environment, and research activity that may have contributed to their success. In short, we attempt to understand the creative process of computer scientists.

The choice of references discussed here was highly idiosyncratic, as any paper in this area must be. For example, the existance of an article entitled "Who Studies Creativity and How Do We Know?" by Beghetto, Plucker, and.MaKinste illustrates the compartmentalization of some of this research. [1]

The paper is organized as follows. After the introductory section, we discuss some early popular works on mathematical creativity. (One of the books discussed there, by Jacques Hadamard, was entitled *The Psychology of Invention in the Mathematical Field*; this title was the motivation for the title of this paper, although this preliminary work would hardly be considered psychological by modern standards.) We then provide excerpts from a series of interviews collected in the book *Out of their Minds*. A separate section is devoted to an in-depth discussion of a book by Tracy Kidder, which reported on the development of relevant software for a minicomputer. This is followed by comments on the creative processes of a few other computer scientists. After a brief essay on the creative design process in the unrelated field of architecture, the paper concludes with a brief analysis of creativity in computer science and makes some suggestions for further, experimentally based research on the creative process in computer science.

Mathematical Creativity

The mathematician Henri Poincaré published his personal experiences of discovery and invention as well as his extensive research. Poincaré discussed his subjective view of creating mathematics for a paper published in the journal *L'enseignement mathématique* as part of an investigation of mental habits and methods of mathematicians [10].

He began by providing a series of questions posed at the process of invention including: How does it happen that there are people who do not understand mathematics? How is error possible in mathematics? He answers these questions by providing sample scenarios to consider, in addition to his own experiences. His most pertinent insights were the process of illumination and the reasoning that invention is a combination of discernment and choice.

According to Poincaré, an illumination is "a manifest sign of long, unconscious prior work" that appears after a period of rest. It is his opinion that the bulk of creativity occurs in the subconscious, where the mind "chooses" which combinations are fruitful. This is followed by a period of conscious work when it is necessary to "shape the results of this inspiration, to deduce from them the immediate consequences, to arrange them, to word the demonstrations, but above all is verification necessary." [10]

He described being hit by the door of a bus when the solution to a mathematical problem became clear, just as he was about to board.

Poincaré suggested that conscious work always precedes all fruitful unconscious labor, where *good combinations* are derived from a finite number of entities. He spends the remainder of the paper combining his previous theories to sustain this thought.

In *The Psychology of Invention in the Mathematical Field*, Jacques Hadamard analyzed Poincaré's theory of mathematical creation [4]. He accompanied his critique with examples and with opposing theories. Hadamard's primary observations were the following:

- Unconsciousness
- The unconscious and discovery
- The preparation stage (Logic and Chance)
- Later conscious work
- Discovery as a synthesis (the help of signs)
- Different kinds of mathematical minds
- Paradoxical cases of intuition

Hadamard opened with a discussion of unconsciousness and the layers therein. He stressed the functionality of the unconscious in the role of invention. He quoted Desdouits' *Theorie de l'Invention* in the following excerpt: "…mind thinks either by analogy or by habit; thus, mind jumps over intermediaries." With regard to the layers, Hadamard delves into the distinctions between sub-consciousness and unconsciousness, offering examples to his reasoning.

Next he attemped to illustrate the relationship between the unconscious and discovery. Discovery, or invention, "takes place by combining ideas" [4, p. 29]. The goal was to show that several steps, including the following, are undertaken as a prerequisite for the process of conscious discovery.

- Combinations are sifted through and created during unconsciousness.
- To invent is to choose, supporting Poincaré's idea that 'fruitful' combinations are selected or chosen.
- Esthetics in invention — feeling which are most useful and pulling them to the forefront are critical. Hadamard notes that this work is not seen or observed by us, but must take place in order for the results to be visible. "The privileged unconscious phenomena, those susceptible of becoming conscious…directly or indirectly affect…our emotional sensibility" [4, p. 31].

Hadamard discussed Poincaré's conclusion of choice, including coming back to the unconscious and other views on *incubation*, which is said to precede *illumination*. The discussion continued with the analysis of how scientists experience illumination after a period of rest, possibly due to fatigue, dubbed the *rest-hypothesis*. He also gives Poincaré's version of this theory, called the *forgetting-hypothesis,* as well as views of the opposition.

Lynn Arthur Steen, a former president of the Mathematical Association of America, has some interesting perspectives on the teaching of mathematics. [3] In a paper written for the mathematical education community, he states. "In fact, as

contemporary neuroscience reveals, the brain is less like a computer to be programmed or a disk to be filled than like an ecosystem to be nourished." [14]

Interviews with Some Prominent Computer Scientists

D. Shasha and C. A. Lazere wrote an interesting book entitled *Out of Their Minds* [12]. This book contains a collection of interviews with experts in several areas of computer science: linguistics, algorithmics, software engineering, and artificial intelligence; only interviews with software researchers are described here. We summarize each interview and include some quotes relevant to the study of creativity.

John Backus
Quote: "We didn't know what we wanted and how to do it. It just sort of grew. The first struggle was over what the language would look like. Then how to parse expressions—it was a big problem and what we did looks astonishingly clumsy now..."
Motivation: imprecision/ inefficiency of current choices for scaling
Inspiration for field: He visited a IBM Computer Center in NY; studied math courses at Columbia University; and worked on IBM's SSEC for three years.
Invention: Fortran, Backus-Naur form, floating point arithmetic
Note: In 1953, Backus wanted to design a programming language (for IBM 704), which already had a floating-point capability, in contrast to Von Neumann's "scaling factor" technique. The problem was that "scaling factor" required the programmer to know a lot about the problem to prevent numbers from overflowing. Backus devised Fortran as a solution.

John McCarthy
Quote: "If you want the computer to have general intelligence, the outer structure has to be commonsense knowledge and reasoning."
Motivation: resolve issues of *deduction*—reasoning computers with the ability to perform commonsense reasoning, creating a new kind of mathematical logic.
Inspiration for field: grew up with confidence in technology as good for humanity due to belief in communism; advanced at mathematics.
Invention: LISP (LISt Processing), principles of Artificial Intelligence.
Note: In 1949, he first attempted to model his idea while a Ph.D. candidate at Princeton where John Von Neumann encouraged him. He later approached Claude Shannon, creator of mathematical theory of communication theory about collecting papers on the subject of artificial intelligence, which had been first dubbed 'automata studies.'

Michael O. Rabin
Quote: "We should give up the attempt to derive results and answers with complete certainty."
Motivation: He was challenged in junior high school to solve a geometry problem and became interested in the theory of establishing truth about lines and circles.

Inspiration for field: The work of Alan Turing—technology has a mathematical, logical idea as a foundation, which led to his interest in logic and computability.
Invention: He expanded the work of Turing with his colleague Dana Scott by creating theory of nondeterministic finite-state machines.
Note: He also worked with John McCarthy, who was at the time battling challenges of Fortran, after being asked the question of the two spies, which involved secrecy of passwords. His solution came to be known as a *one-way function*, which translate into computational procedures. They are easy to compute in one direction, but hard to compute in the other. The computation of large-digit prime numbers led to Rabin's theory of randomization with the possibility of error.

Donald E. Knuth
Quote: "Computer programming is an art form, like the creation of poetry or music." "It's not true that necessity is the only mother or father of invention…'Oh man, I have a unique background that might let me solve it—it's my destiny, my responsibility.'"
Motivation: His desire to understand compiling. Love of grammar and crossword puzzles.
Inspiration for field: Define the semantics for Backus-Naur form syntax.
Inventions: A program to compute high school players' statistics. Knuth-Morris-Pratt algorithm for text matching, computer languages for digital typography including TEX and METAFONT, contributions to LR(k) parsers, attribute grammars.

Alan C. Kay
Quotes: "All understanding begins with our not accepting the world as it appears." "Computer programming is a bit like a Gregorian Chant—a one-line melody changing state within larger scale sections. Parallel programming is more like polyphony."
Motivation: Combination of physiologists and artists for parents attributed to thinking out of the box.
Inspiration for field: "As We May Think", An article by Vannevar Bush that appeared in the *Atlantic Monthly* in 1995. The article speculated about a device that would enable browsing through microfiche library—Memex. It looked for a single building block that would permit a simple, powerful style of programming. In 1961 an unknown programmer discovered that he could transport data files and procedures by bundling them together—this formed the basis for Kay's ideas about objects and how to structure them.
Inventions: Flex machine, Dynabook prototype, Smalltalk, object-orientation paradigm.
Notes: He had a form of illumination where the biological analogy was the influence for his ideas. "The big flash was to see this as biological cells. I'm not sure where that flash came from but it didn't happen when I looked at Sketchpad. Simula didn't send messages either."

Edsger W. Dijkstra
Quote: "I asked my mother, a mathematician, whether mathematics was a difficult topic. She said to be sure to learn all the formulas and be sure you know them. The

second thing to remember is if you need more than five lines to prove something, then you're on the wrong track."
Motivation: His parents were both scientists (chemist, mathematician).
Inspiration for field: He began programming extensively at the Mathematical Centre in Amsterdam. He was asked to demonstrate the powers of the ARMAC for a conference in 1956. He thought about the problem of determining the shortest route between two points on a railroad map and how that concept could be applied to computing. Later, he wanted to synchronize sequential processes in order to "reason about them."
Invention: Shortest-Path algorithm, application of semaphores to enforce mutual exclusion, Dining Philosophers problem.
Notes: One Sunday morning while eating breakfast, he thought of how trains were controlled as a solution to the self-posed shortest path problem. He did not publish his ideas at the time because of the lack of enthusiasm for algorithms within the mathematical community. Again, he applied the reasoning used to signal trains (semaphores) to the concept of enforcing mutual exclusion for access to a critical section. The Dining Philosophers problem posed the question of how to prevent starvation, deadlock, and lack of fairness as they apply to shared resources.

Robert E. Tarjan
Quote: "I visualize structures, graphs, data structures. It seems to come easier than a lot of other things." "A good idea has a way of becoming simpler and solving problems other than for which it was intended."
Motivation: reading columns in *Scientific American* sparked interest in math. He worked with IBM cardpunch collators at a state hospital while still in junior high school.
Inspiration for field: He desired to answer the question: "Can a graph be laid flat?" The solution would ensure that for certain applications edges would not overlap while connections are preserved. A graph that can satisfy these criteria is called planar.
Invention: Planarity testing, applying *depth-first search*, efficient network flows, union-find and Amortization.
Notes: He worked with John Hopcroft to successfully lay a graph out on a plan directly. Their algorithm applied depth-first searching with the inclusion of planarity testing and required a five-step process.

Leslie Lamport
Quote: "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."
Motivation: His father's unrealized goal of being a doctor along with his curiosity in math and a fierce need to see proofs to theories or else disprove their validity. A visit to IBM and working for Con Edison started interest in Computer Science.
Inspiration for field: Einstein's work on speed of light in physics. Lamport wanted to show "the relative order of events in a distributed system can depend on the observer." Also, wanted to change the notion that pure math was superior to applied math within the mathematical community

Invention: Bakery Algorithm (alternative to Dijkstra and Knuth's solution to mutual exclusion); Lamport Clocks for synchronization in distributed systems without a global clock.

Stephen Cook
Quote: "The idea that there won't be an algorithm to solve it—this is something fundamental that won't ever change—that idea appeals to me."
Motivation: His father was a chemist and professor at University of Buffalo, NY. He was interested in math and chess as a child. He had the opportunity to work with Wilson Greatbatch, inventor of the implantable pacemaker, on transistors in Clarence, NY shop as a teenager. He took a 1-credit programming course under Bernard Galler at the University of Michigan.
Inspiration for field: Hao Wang, Professor in Applied Science, was very influential to him while he worked on his Ph.D. at Harvard. Wang fostered his interest in automatic theorem proving. Another influence was Rabin's work on complexity theory and *uncomputability* results explored in his 1959paper on the subject.
Invention: NP-complete problem categorization
Notes: He worked with Leonid Levin to categorize algorithms that required an infinite amount of time to complete, but that can be easily verified.

Leonid Levin
Quote: "Sometimes it is good that some things are impossible. I am happy there are many things that nobody can do to me."
Motivation: His father was professor and mother was a industrial architect. They both encouraged his early interest in science and math. At one point in time as young man, he rewrote the Periodic Table of Elements in order to structure it to his liking. Several meetings and discussions with Kolmogorov, a well-known Russian scientist, while he was still in high school, led to problems being posed to him that he would later encounter in computer science.
Inspiration for field: His desire to outline a formal relationship between *perebor* problems and Kolmogorov's work on information theory, randomness in particular.
Invention: NP-complete problem categorization
Notes: Levin was unaware of Cook's work for years because Russian libraries and institutions did not receive relevant publications and economic issues hindered private accessibility. There was a very weak response by the Russian community to Levin's accomplishment. "Some people were interested. I didn't know if this was a genuine interest or sympathy to my political troubles…"

Edward A. Feigenbaum
Quote: "There are three important things that go into building a knowledge-based system: knowledge, knowledge, knowledge. The competence of a system is primarily a function of what the system knows as opposed to how well it reasons."
Motivation: His stepfather was an accountant and office manager in a bakery and would take him to Hayden Planetarium and Museum of Natural History every month when he was a child. He excelled in science and was intrigued by the

electromechanical calculator used at his stepfather's job. Later, his parents encouraged him to study Electrical Engineering instead of pure Science.
Inspiration for field: Early graduate school interest in Artificial Intelligence. "I was intrigued by the vision of a highly intelligent, maybe super-intelligent artifact." Newell and Simon's work with the *Logic Theorist*, a "thinking machine" that modeled human problem solving was inspirational. It provided the basis for his doctoral thesis, in which he would model a limited kind of memorization with similar goals (EPAM.) He wanted to present a computer model of memory organization that would explain a subject's incorrect, as well as correct behavior, what he dubbed *discrimination net.*
Inventions: Elementary Perceiver and Memorizer (EPAM); Dendral; Knowledge Principle.
Notes: In 1965, John McCarthy convinced Feigenbaum to join him at Stanford. "I decided I didn't want to be a psychologist and study human behavior using a computer. I wanted to build the computer artifacts. I moved from Berkeley to Stanford to be a part of this new department."

Douglas B. Lenat
Quotes: "How many people have in their lives a 2 to 10 percent chance of dramatically affecting the way the world works? When one of those chances comes along, you should take it." "The AI goal is a kind of *mental* amplification so we can be smarter, be more creative, solve harder problems, forget less, be reminded of more."
Motivation: As a sixth grader, he discovered Isaac Asimov's books on physics and biology and his interest grew from there.
Inspiration for field: While at the University of Pennsylvania he took a course taught by John W. Carr III that introduced him to artificial intelligence. "[It was] like being back doing astronomy right after the invention of the telescope. The subject was inherently fascinating, but it had two other really interesting properties."
Invention: Automated Mathematician (AM), Eurisko, and Cyc
Notes: AM's main contribution was to embody the notion of learning heuristics. Eurisko's goals were to discover new heuristics and create journal-caliber mathematics; its principal success was to complete some innovative circuit designs.

The Soul of a New Machine

In *The Soul of a New Machine*, Tracy Kidder combined objective summaries of employee experiences, including first-hand accounts of incidents, with quotes from colleagues and public history regarding working for Data General Corporation [5]. This book followed the pattern of most of Kidder's work – embedding himself in an organization for a year while a major project was completed.

Kidder discussed Data General's motivation to develop a 32-bit minicomputer to compete with the popular VAX product line. Although the term was not used, the environment was a precursor to the market-responsive, crisis-driven, concurrent-engineering paradigm popular in the 1990s and early 2000s. Comments on (portions

of) the management team included: "… expected you to be on his secret wavelength, and if you weren't he'd be disappointed in you…he didn't have time to explain" and "some of the engineers closest to … suspected that if he weren't given a crisis to deal with once in a while, he would create one."

In our discussion of Kdder's observations, we will not give names of specific individuals. Instead, we will categorize some of their behaviors as relative to creativity in the software domain.

Some of the MicroKids, as the new recruits were dubbed, liked the brash fervor in the air; others cited the relaxed atmosphere and flexible work schedule. One senior engineer said, "Engineers want to produce something. I didn't go to school for six years just to get a paycheck."

As a child, one team leader was desperate to find out how the telephone worked. While in the third grade, he tapped his family's phone after he discovered the wires running across the basement ceiling. His parents tolerated his curiosity, mostly because his father, an engineer, designed refrigerators for Westinghouse and understood his intrigue. He was addicted to all night programming sessions, and used some interactive AI-based games to encourage his staff to do the same.

Another team leader never stopped suggesting to people at the Westborough location that their talents had been slighted. "Let's show 'em what we can do." He developed the phrase "flying upside down" to describe taking large risks. Other notable quotes were "Well I guess a good strategy is one that no one else understands," "not everything worth doing is worth doing well," and "trust is risk, and risk avoidance is the name of the game in business."

The Microteam, a team with four members who were responsible for microcode, liked to attach their name to almost everything they touched. For example, they called the conference room a Micropit, and employee recognitions were called Microawards. One member of the team, whose father was an engineer, organized the microcode for Eagle and drew up the rules of grammar for the code, which had been partitioned into *microverbs*. Each microverb and the rules for their application were held in a book.

The Microteam called this book their "bible;" the hardware designers compared it to a "Sears Roebuck catalog" and called it the "Microteam's wishbook." This sort of denigration is typical of the tension between designers reusing software components and programmers who wish to creatie new ones.

Views of Some Other Computer Scientists

We give a few comments of some other prominent computer scientists on the nature of creativity. The selection of the individuals from whose work comments were taken was highly idiosyncratic.

Daniel Berry

His publication, "The Importance of Ignorance in Requirements Engineering," offers the premise that ignorance of a particular curriculum or subject is positive and helpful in creating efficient requirements documentation. [2] The results are from an experiment conducted at the author's school, where the participants included the

author and a graduate student. Berry acted as the requirements analyst and the student acted as a client. He cites three methodologies he used to determine the requirements necessary:

- Abstract data typing
- Strong typing
- Jewish Motherhood

Leon Osterweil

A major theme of Osterweil's work was to review processes and use the results to improve the development and evolution of software products.[9] He cites that one of our major problems is our inability to understand "our most central and difficult problems in terms of the process description, instantiation, and execution paradigm.

In response to the problem domain, the author suggests an approach of creating software process descriptions to guide the development of key processes. This issue is how to fully describe what is different between what he is proposing and what has been done to date—process programming. He suggested a straightforward type of process for testing application software that highlights key aspects of testing and hiding lower-level details.

He concludes that a key advantage of his approach would allow managers to efficiently communicate steps in the product's development or evolution goals to workers, customers, and other stakeholders. He also discusses the need for knowledge management and the ability to identify the need to acquire or retain it.

In a personal conversation with one of the authors, he described his experience with an "aha." He indicated that the revelation of the fundamental insight occurred while walking around the West End of London looking at store windows during a sabbatical leave.

Stephen Morse

Much of the information on him comes from his web site, stephenmorse.org, including an article from HeritageQuest magazine by Barbara Krasner-Khait that is summarized here [6]. As a child, he developed an interest in mathematics and electricity. When he was in junior high school, he asked his mother which career would combine both and she replied "electrical engineer." He obtained three degrees in this field and was a major contributor to the Intel 80806 chip architecture. Some of his professional software research was in the area of weak dependence for compilers.

He is best-known to the non-computer science world, however, for his applied work in the field of computerized genealogy. Like many Americans wishing to study their immigrant ancestors, he started to use the Ellis Island database as soon as it became available. The limitations of the database interface and the slow response time were frustrating to him (and most other users). Perhaps the most serious problem, however, was the inability to do anything more than a pure name search, which was highly inadequate when searching databases in which there might be many name variants. In Morse's own words, "I couldn't do any more on my form than what their engine did, but it quickly became apparent that there were two omissions in what they supported – the ability to search by town and Soundex. This was confirmed by many

e-mails requesting these features and ultimately led to my writing my own search engine." [6]

In response to these problems, he wrote a set of program scripts to develop a front-end interface to the Ellis Island database that used phonetic and linguistic techniques, strategies to treat certain data entry errors, and multiple optimized queries to search for people, ships, and places. He later extended his work to include several commercial and free databases. He also developed a set of software tools to allow genealogical researchers to provide their own interfaces and search engines to compiled data on-line.

A View from a Different Discipline

In "The Design Process," Ellen Shoshkes commented on design creativity in the field of architecture. "Design is decision making, and the key to successful projects lies not only in the final form but in the process leading up to it." [13, p, 8]

Some other comments are also quite familiar to the software industry:

- "Projects are becoming larger and more complex." Robert Gutman
- "Learning from case studies is important. (NB – there is no mention of experiments or comparative studies.
- Clients play an important role – you are only as good as your clients are
- Clients must select right architect.
- Competitions are often used instead of requests for proposals (RFPs).
- Trends and styles may vary, and end product must be evaluated against building performance or user's needs.
- Architects often have to undergo public reviews of intermediate steps

The jury system often used to evaluate architectural design has some features of both design reviews and apprenticeship, but is different from each in fundamental ways. We intend to pursue this point in future research.

Analysis

Most scholars are excellent problem solvers, but have spent little time understanding how they have reached their solutions – computer scientists are no exception. Little of the published information studied in this paper about the computer scientists selected offered a different perspective of how psychology and computer science can correlate and produce creative results. This, of course, may be due to the very limited number of writings reviewed for this paper. However, the difficulty may lie deeper. The writings excerpted from *Out of Their Minds* attempt to detail the ideas of the subjects by recalling situations and instances, as well as providing motivation for their work.

Contrast this with the writings of Poincaré and Hadamard. Poincaré discussed illumination and discernment. Although, his view might be limited based upon the premise that one cannot analyze each thought while having that thought, he provides

insight into his own thought process and introduces identifiable stimuli sometimes noted by other scientists. Hadamard's comparative analysis promotes further discussion based on his approach of simultaneously supporting and opposing Poincaré's ideas in a constructive manner.

Note, however, that Poincaré and Hadamard were mathematicians, working in fields where much of the important research is typically done by a single researcher, although he or she may have been highly influenced by discussions with other mathematicians, or even influenced by a G ttingen effect, where being in the proximity of a large number of active world-class researchers appears to force the subconscious to work more often, thereby causing great increases in research activity.

We posit that this pattern applies to theoretical computer science research, but not to the building or creation of systems, which are inherently team-based. We note, however, that there is considerable consensus in the research community that the term "creativity" refers to the production of work that is both novel and appropriate. [8] Since almost no one now creates modern software systems from scratch, the pattern in question might be paraphrased as "Can individuals exhibit creativity in systems designed by others?"

The interviews with Backus, McCarthy, Rabin, Knuth, Kay, Dijkstra, Tarjan, Lamport, Cook, Levin, Lenat, and Feigenbaum show a general pattern of early involvement with creativity. They often had parents trained in, or highly motivated by, the sciences (McCarthy, Rabin, Kay, Dijlstra, Tarjan, Lamport, Cook, Levin. Lenat, Feigenbaum) and had serious technical opportunities while in their early and middle teens. Two talked about reading compelling popularized articles and books – by Vannevar Bush and Isaac Asimov. Several mentioned coming under the tutelage of inspiring professors.

The book *The Soul of a New Machine* contains interesting insights on the state of mind of employees of one company during the most crucial periods of development. This allows promotion of many assumptions regarding the creativity of its staff, including how environmental stimuli can encourage or adversely affect product development.

The comments on the creative process on their own work by Berry, Osterweil, and Morse pose additional insights. The relevant work of Berry and Morse described here is largely the result of individual effort, while Osterweil's work is highly motivated by process issues.

It is clear that the need to further understand the creative process as it pertains to computer science could, itself, be beneficial to the programming community. The goal is to answer questions such as: Can the low-level describing of a process be equated to or used to model how humans think, create, or modify their creations?

An Agenda for Future Work

This highly preliminary work on the nature of creative work of computer scientists needs considerable refinement before it can be used directly to help set the research agenda for the study of the psychology programmer behavior. Nevertheless, some issues worthy of study can be identified. Each of these issues should be studied in a

series of case studies, comparative studies and experiments (controlled and “natural”) in order to advance our understanding of the field.

- Are there any common patterns discernable in the educational and environmental backgrounds of successful programmers?
- Are there any differences between the thought processes of the computer scientists who envision novel systems (such as the ones described here, or the first spreadsheet, the first multi-tasking operating system, the World-Wide-Web, or the page-indexing algorithms of Google) and the programmers who implement them?
- Are there any differences between the thought processes of, say, the lead software architects of systems such as the ones described above and other software engineers?
- Are there major differences in the thought processes of the most productive programmers, who may be more productive than their colleagues by a factor of 10 or more, and the average colleague in their organization?
- To what degree is the so-called “Not Invented Here” syndrome that often makes implementation of a systematic policy of software reuse that involves extensive searching of libraries and repositories for potentially reusable components and architectures difficult actually a function of otherwise productive programmers resisting limitations on their creativity? (The issue often comes up in the reuse community, but the authors are unaware of any study that compares the performance of average programmers and designers from the perspective of reuse with the performance of exceptional ones.)
- Are the thought processes of these super-productive programmers closer to those of creative computer scientists or to those of the typical programmer in the field?
- Are there differences between the creative processes of highly creative theoretical and experimental computer scientists?
- Are there sufficient similarities between the design process in, say, architecture and computer programming to improve programming education and practice? If so, can the jury system used in architectural education be useful for training programmers?

Two Caveats

Thomas Morton observed "Historians often reason from the internal evidence...but (in science and technology] a parallelism between two accounts cannot reliably be used to infer that one influenced another (or even that they were influenced by a common source.)" It is easier to attribute every invention to one person or organization rather than have to untangle the unwieldy web of the way things happen. If the same idea crops up in two places, it is easiest to assume that one must have taken it from the other. [7] Jef Raskin expands upon jumping to conclusions about the creation of several well-known computer artifacts. [11]

There is also a great danger in depending on self-analysis as a guide to understanding the creative process. The science fiction writer Isaac Asimov once described going to a lecture where one of his short stories was discussed. The lecturer claimed that Asimov’s story had a secondary theme that was motivated by a particular

underlying idea of Asimov. After the lecture, Asimov sought out the lecturer, introduced himself and stated that he had had no such motivation when writing the story. The lecturer retorted "How would you understand what the motivation was for the story? You are only the author"

Acknowledgement

This research was partially supported by the National Science Foundation under grant number 0324818.

References

[1] Beghetto, Ronald A., Plucker, Jonathan A., and James G. MaKinste, "Who Studies Creativity and How Do We Know?," Creativity Research Journal, Vol. 13, No. 3&4, 2001, Pages 351-357.

[2] Berry, Daniel M., "The Importance of Ignorance in Requirements Engineering," Journal of Systems and Software, Vol. 28:1 pp179-184, Elsevier Science, Inc., 1995.

[3] Education Commission of the States, *Brain Research Implications for Education*, Vol. 15, No. 1, winter 1997.

[4] Hadamard, Jacques, *The Psychology of Invention in the Mathematical Field*, Dover Publications, Inc., 1954, with permissions from Princeton University Press, 1945.

[5] Kidder, Tracy, *The Soul of a New Machine*, Atlantic Monthly Press, New York, 1981.

[6] Krasner-Khait, Barbara, "Morse's Code," *HeritageQuest Magazine*, June, 2004.

[7] Morton, Thomas, American Scientist, Vol. 82, pp 182-5.

[8] Lubart, T.I., "Models of the creative process: past, present and future," *Creativity Research Journal,* Vol. 13, No. 3/4, 2000, pp. 295-303

[9] Osterweil, Leon, "Software Processes are Software Too," ACM 0270-5257/87/0300/0002, 1987.

[10] Poincaré, Henri, "Mathematical Creation," in Newman, James R., *The World of Mathematics.,* Vol. 4 pp 2041-2050, Simon and Schuster Publications, 1956.

[11] Raskin, Jef, "The Genesis and History of the Macintosh Project," available on–line at www.digibarn.com/collections/business-docs/raskin-mac-genesis/index.html.

[12] Shasha, D. and Lazere, C., *Out Of Their Minds*, Copernicus, an imprint of Springer-Verlag New York, Inc., 1998.

[13] Shoshkes, Ellen, *The Design Process*, Whitney Library of Design, Watson-Guptill Publications, New York, 1989.

[14] Steen, Lynn Arthur, Twenty Questions about Mathematical Reasoning, in *Developing Mathematical Reasoning in Grades K-12*. Lee Stiff, Editor. Reston, VA: National Council of Teachers of Mathematics, 1999, pp. 270-285.