Alan Kay: Transforming the Computer into a Communication Medium

Susan B. Barnes
Rochester Institute of Technology

Alan Kay is referred to as the ''father of the personal computer'' because his 1969 doctoral thesis described an early prototype of personal computing. Kay's ideas contributed to the transformation of the computer from a calculating machine to a communication medium. This article focuses on Kay's vision for personal computing.

Alan Kay's visionary ideas about computers were instrumental in transforming the computer from an office machine into a mainstream communication medium. Throughout his career at the legendary Xerox PARC (Palo Alto Research Center), he encouraged his colleagues to design small notebook-size computers. However, in the early 1970s, the technology was not yet available to transform Kay's vision into a reality. As a result, Kay focused on software design and the creation of programs easy enough for children to use. This led to the development of overlapping windows, and the object-oriented software program called Smalltalk. In addition, Kay and his team advanced the design of graphical user interfaces (GUIs) that were invented in the 1960s. Their GUI design concepts were later popularized by Apple Computer's Macintosh and Microsoft's Windows software. Kay's research goal was to make a functional prototype of a computer that had everything but the right size. Today, many computer users interact with notebook computers that incorporate Kay's design concepts.

In this article, which I wrote in cooperation with Alan Kay, I explore the background influences that led up to Kay's innovative design concepts.

Early years

Alan Curtis Kay (see Figure 1) was born in 1940 in Springfield, Massachusetts. His father was a physiologist who designed arm and leg prostheses, and his mother was a musician who taught him music. Kay grew up in an environment of art, literature, and science. He could read by the age of three, and by the time he started school, Kay had already read several hundred books. In 1950, Kay's family moved to New York City where he attended Brooklyn Technical High School. After high school, he went to college but before completing his undergraduate degree, Kay joined the US Air Force in 1962.

While in the air force, Kay discovered computers. He passed an aptitude test to become an IBM 1401 programmer and was given a one-week IBM programming course. He gained further experience by working with a variety of computers including the Burroughs B500. In the air force, he also learned approaches to data abstraction. Programmers had to figure out how to transport data files on computers that had no standard operating system. In one instance, an unknown programmer decided to divide a file into three parts. Part three was the actual data: the second part contained Burroughs 220 procedures plus instructions on how to access records and fields to copy and update part three data, and the first part included an array of relative pointers into entry points of the part two procedures. The data bundle, including its procedures, could be directly used at a new site. This led to the idea that a program could use procedures without knowing how the data were represented,[1] a concept that later contributed to Kay's development of object-oriented programming languages.

After leaving the air force, Kay worked his way through college, programming weather data retrieval systems for the National Center for Atmospheric Research. While doing this work, he became interested in simulation and in how one machine could replicate another machine, an idea that also influenced his later work.

Kay finished his undergraduate degree work at the University of Colorado, where he helped debug a CDC 6600 machine. As an undergraduate, he read Gordon Moore's 1965 article,

 Published by the IEEE Computer Society 1058-6180/07/$25.00

which predicted the future of integrated silicon on chips that would exponentially improve in density and cost over many years. This prediction became known as Moore's law and it forecast a future of smaller computers.

University of Utah

After completing his undergraduate degree in mathematics and molecular biology, Kay went on to earn his doctorate from the computer science program, headed by Dave Evans, at the University of Utah. Utah's computer science program was financed by the Department of Defense's Advanced Research Projects Agency (ARPA) run by J.C.R. Licklider and later by Robert Taylor. Licklider's vision of ''man-machine symbiosis'' (human–computer interaction) had influenced the interactive computer projects that ARPA funded. In 1968, Licklider and Taylor published an article titled ''The Computer as a Communication Device.''[2] The article described conceptual ideas for the design of computer networks that would support human communication, and it also influenced ARPA research. Kay was one of the many graduate students attending ARPA-sponsored conferences that contributed to ARPA research and projects, such as time-sharing and the early Arpanet.

During his graduate work, Kay was influenced by several major technological developments, including Sketchpad, Simulation Language (Simula), flat-screen displays, Logo, and Engelbart's interactive computing system. Evans made all new students read Ivan Sutherland's Sketchpad thesis. Sketchpad was the first interactive computer system, and the first to use computer graphics along with computer-aided design (CAD). The system's unique software program took a drawing (an instance), checked it against the indicated constraints, and then attempted to change the drawing to obey the rules of the constraints. Master drawings could produce instance drawings and be controlled by constraints. Information was graphically and interactively displayed to the viewer. More important, Sketchpad provided direct visual feedback to the computer user by graphically displaying information in zooming windows. Users virtually ''sketched'' their designs on an electronic screen that represented a 1/3 square mile.

In addition to Sketchpad, Kay encountered Simula, developed in 1965 by Kristen Nygaard and Ole-Johan Dah1 to develop computer models for production line and manufacturing systems.[3] The program was first written as an extension of Algol 60, a popular European programming language, and was later expanded into a full-scale general-purpose language. Simula supported discrete-event simulation construction and introduced object-oriented programming concepts, including classes, objects, inheritance, and dynamic binding. Kay used a biological analogy between ideas in Sketchpad and Simula: First, cells (instances) conform to basic ''master'' behaviors. Second, cells are autonomous and communicate with each other by sending messages. Third, cells become different parts of the organism, depending on the context. These ideas later became central features of the Smalltalk program design, especially the concept of communicating through the exchange of messages.

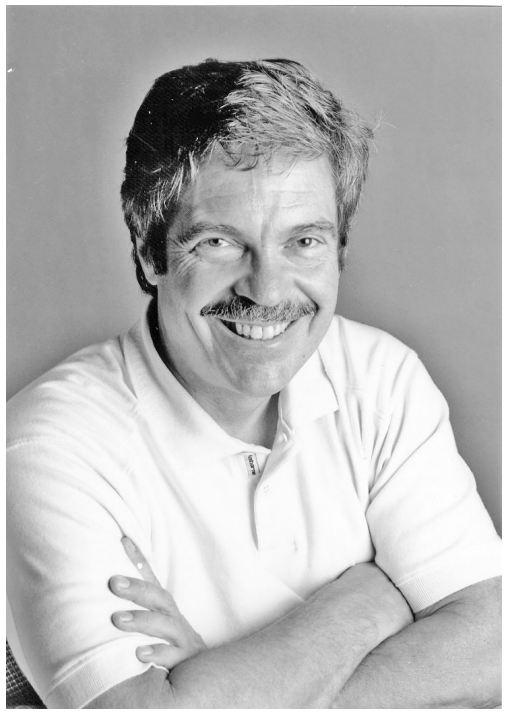

Figure 1. ''The best way to predict the future is to invent it.''—Alan Kay. (Courtesy of Alan Kay.)

The University of Utah's Evans believed that graduate students should become involved with actual computing projects, and he helped his students obtain consulting jobs. He introduced Kay to hardware designer Edward Cheadle, who was employed at a local aerospace company. Cheadle was working on the design of a ''little machine'' and the two teamed up. They began building the FLEX computer (see Figure 2) that would provide sharp graphics and windowing features. Kay described FLEX as ''the first personal computer

Figure 2. Drawing by Alan Kay of FLEX machine, c. 1968. (Courtesy of Alan Kay.)

to directly support a graphics and simulation-oriented language.''[4] In addition to Simula, FLEX was influenced by previous work done by others, including Wesley Clark's LINC (*L*aboratory *In*strument *C*omputer), a small computer that weighed several hundred pounds; the Rand Corporation's JOSS (Johnniac Open-Shop System); Douglas Engelbart's interactive Augmentation System; and Seymour Papert's LOGO project.

While working on the FLEX machine, Kay witnessed a demonstration by Engelbart.[5] In early 1967, Engelbart visited the University of Utah and showed films demonstrating the online System (known as NLS) with its metaprogramming language. NLS was designed as an interactive tool to augment human intellect and the collaborative decision-making process. The system included graphics, hypertext, and a mouse for inputting commands. Engelbart and his team had built an entire conceptual world that enabled users to navigate through ''thought vectors in concept space.'' Inspired by Engelbart's vision of collaborative computing, Kay wanted to design systems to be used as personal, dynamic media by children. Interacting with this new medium would require a keyboard and stylus.

The design for the FLEX's interface was influenced by JOSS and the *Gra*phic *I*nput *L*anguage (GRAIL).[6] JOSS was developed as an Open-Shop system to support 12 personal communication stations that could be simultaneously accessed. In the initial design, users connected to the system via typewriter terminals. Later, the GRAIL project was initiated to investigate new techniques to improve human–computer communication. This led to the development of the Rand Tablet, which supported real-time computer recognition of hand-printed text, and it could produce precise boxes from loosely drawn ones. Programmers used GRAIL and the Rand Tablet to make flowcharts. The system used direct manipulation (some modeless), which made it a friendly user interface. Modeless computing lets users open several different types of programs at the same time and move back and forth between them. For instance, a window containing a drawing program can be open at the same time as a window containing a word processing program. While at PARC, Kay later viewed modelessness as a way to enable users to move between functions without explicitly exiting any.

As a PhD student, Kay began to appreciate the work of media guru Marshall McLuhan. After reading *Understanding Media*, Kay realized that the most interesting thing about any medium is what you have to become to communicate with it. Anyone who wants to receive a message must first internalize the medium so it can be subtracted out to leave the message behind. People become different types of thinkers after they internalize a medium. As a result, Kay began to think about the computer as a communications medium rather than as a tool. Accordingly, a high-resolution display screen was needed for this new medium that would approximate the quality of a printed page. If computer screen quality were similar to book quality, computers could be introduced into schools. A technology that could produce this type of image quality was a flat-panel display screen.

While attending an ARPA graduate student meeting in summer 1968, Kay saw a one-inch square piece of glass with neon gas that could make individual spots light up on command. It was the first flat-panel display screen. Recalling Moore's law, Kay began to calculate how long it would take to put a computer on the back of flat-panel screen technology and estimated that it would be sometime in the late 1970s or early 1980s.

At this point, Kay's concept about computers began to evolve as he envisioned millions of personal machines and users, instead of thousands of institutional mainframes. Personal machines would need to be designed as extensional systems in which end users could tailor and direct the construction of their tools. Shortly after this realization, Kay visited one of Seymour Papert's early Logo tests, which would have a bearing on his thought process concerning personal machines and users. Between 1968 and 1969, Papert worked with 7th-grade students at the Muzzey Junior High School in Lexington, Massachusetts.

In the Logo project, students used the computer as a learning environment. Prior to working on Logo, Papert had spent five years

working with educational psychologist Jean Piaget. Piaget began a revolution in learning theory with his observations on how children learn. His research indicated that learning is not simply something adults impose on children through education. In contrast, learning is deeply embedded in the ways in which children are innately equipped to react to their natural environments. Applying Piaget's ideas to learning, Papert understood that the computer's representational capabilities and responsiveness could be used to construct simulated ''microworlds'' for children to explore and learn mathematical principles.[7] Children would explore the computer-simulated worlds in a manner similar to how they explored the natural environment, while simultaneously learning the language of math (see Figure 3).

When Kay visited Papert in 1968, he saw children writing computer programs that generated poetry, translated English into Pig Latin, and created arithmetic environments. He then became interested in the analogy between print literacy and the Logo program. Prior to visiting Papert, Kay believed that you needed to be an adult to be a programmer. After seeing children using Logo, Kay decided that computer languages should be developed on a level for children to understand. Children should be able to learn to read and write with this new medium. Kay described this as follows:

> The ability to ''read'' a medium means you can access materials and tools created by others. The ability to ''write'' in a medium means you can generate materials and tools for others. You must have both to be literate. In print writing, the tools you generate are rhetorical; they demonstrate and convince. In computer writing, the tools you generate are processes; they simulate and decide.
>
> If the computer is only a vehicle, perhaps you can wait until high school to give ''driver's ed'' on it—but if it's a medium, then it must be extended all the way into the world of the child.[8]

At this point, Kay envisioned a Kuhnian paradigm shift in which the computer would become a revolutionary new communication medium with a social impact similar to the invention of the printing press. This new medium would be designed as a dynamic process for representing, communicating, and animating messages with words, images, and sounds. Kay's use of the term *dynamic simulations* referred to the state of images presented on a computer screen and the ability to change the information display. Dynamic simulations had the potential to surpass the book as a form of communication and annihilate the passive boredom of television.

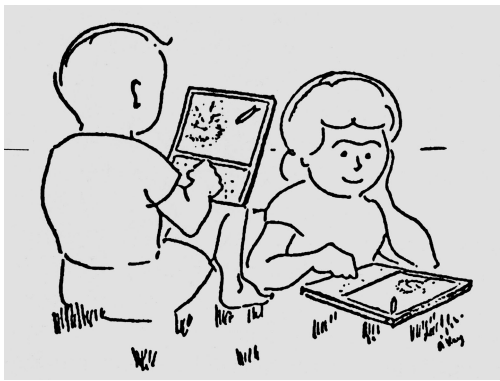

Figure 3. Drawings by Alan Kay of children with small computers. Created after the visit to Seymour Papert. (Courtesy of Alan Kay.)

Inspired by this realization, Kay made a cardboard mock-up of his vision for a tablet-style personal computer with a flat-panel screen, keyboard, and stylus. The keyboard and stylus created a balance between the low-speed tactile freedom offered by a stylus and the faster keyboard input. Meanwhile, he continued working on FLEX.

Kay's thesis was called the ''Reactive Engine,'' and the FLEX computer was completed in 1969. FLEX was designed with a tablet as a pointing device, a high-resolution display for text and animated graphics, multiple windows, and a user interface. Although much of the hardware and software was considered successful in terms of computer science research, the design lacked the expressive range to engage ordinary users. The use of multiple windows was not simple enough for a wide array of users. At that time, state-of-the-art technology could not capture Kay's vision for dynamic personal computing.

Kay's experiences with FLEX, flat-screen displays, GRAIL, McLuhan, and Papert had come together into a vision of what personal computing should be. Kay's dissertation included illustrations of both complex diagrams of functions and line drawings of single-user machines to be developed at a future date. This new machine would be a personal, dynamic medium of communication that children could use.

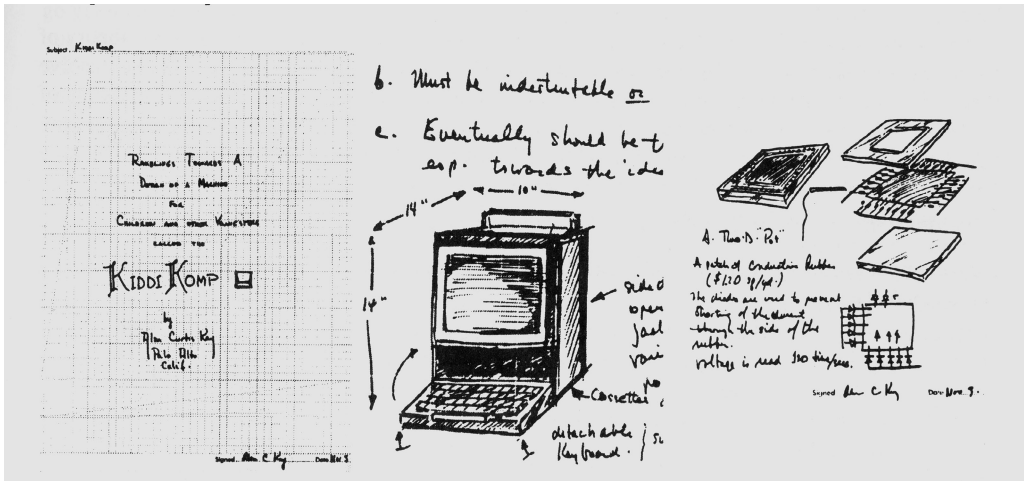

Figure 4. Drawings by Alan Kay of KiddiKomputers. (Courtesy of Alan Kay.)

Small computers

After graduation from the University of Utah, Kay spent two years as a researcher at the Stanford Artificial Intelligence Laboratory, where he began thinking about book-sized computers that children could use. Kay wanted to make the smallest possible prenotebook computer that would enable him to learn how children would use this new technology. During the course of his research, he discovered Planner, a programmable logic system developed by Carl Hewitt. At the AI Lab, he wrote several programming languages based on a combination of the pattern-matching schemes of Planner and FLEX. However, Kay spent most of his thinking time working on ideas for notebook "KiddieKomputers." In 1970, he made sketches of KiddieKomps, which were later incorporated into design ideas for the Alto computer, developed in 1973 at Xerox PARC (see Figure 4).

Xerox PARC

In 1970, Xerox established Xerox PARC to pursue long-term research to develop "the office of the future" and hired Robert Taylor to recruit talented computer scientists from across the US. Prior to joining Xerox PARC, Taylor realized that telecommunication networks were more than the sending and receiving of information from one point to another.[2] In other words, Taylor discovered principles of communication as a process rather than communications as a device: First, communicators are active participants and play a central role in the network communication process. Second, communication is a mutually reinforcing process, which involves creativity. Third, the digital computer is a flexible, interactive medium that can be used to support cooperative human communication. Finally, Taylor understood the idea of common frameworks or the use of mental models in computer-based communication exchanges. Mental models are the mental images of people, objects, and the environment that form as people interact with others. For instance, individuals create mental images of people they meet through electronic mail. In human–computer interaction, the desktop is the mental model that makes it easier for people to operate a computer.

Taylor hired the computer scientists who built the first interactive systems, the first augmentation systems, and the first packet-switching computer networks. In 1971, Kay switched roles at Xerox PARC, from consultant to employee. He established the Learning Research Group, which was interested in all aspects of communication and the manipulation of knowledge. Central to the group's research was the design, development, and implementation of dynamic media. Computers are dynamic media because electronic documents can easily be edited, compared, and juxtaposed. Kay hired people who shared his interest in interactive computing and notebook-size computers, which he called Dynabooks. The Learning Research Group established goals to

- provide examples of how Dynabooks could be used across subject areas;
- study how Dynabooks could help to expand an individual's visual and auditory skills;

- provide access to children so they could spend time probing the system and searching for personal ways to understand daily processes; and
- study children's unanticipated uses of the Dynabook and Smalltalk.[9]

At PARC, Kay was a visionary force in helping to develop the tools to transform computers into a major new communication medium. He has been described as PARC's ''self-defined futurist-in-residence'' who blurted out the unofficial PARC credo: ''The best way to predict the future is to invent it!''[10] The following excerpt is from a 1971 PARC memo that Kay wrote:

> In the 1990s there will be millions of personal computers. They will be the size of notebooks of today, have high-resolution flat-screen reflexive displays, weigh less than ten pounds, have ten to twenty times the computing and storage capacity of an *Alto*. Let's call them *Dynabooks*.[11]

The Dynabook was described as a powerful and portable electronic device the size of a ''notebook and destined to become just as ubiquitous. The Dynabook would carry an encyclopedia of information inside its circuits and plug into readily available networks containing the sum of human knowledge.''[12] Today, the Dynabook is viewed as the visionary prototype of laptop and notebook-sized computers. The name, *Dynabook*, was influenced by the writings of Marshall McLuhan who described the profound cultural impact of the Gutenberg printing press.[13] By naming this new medium the Dynabook, Kay wanted to suggest that the Dynabook would have a similar cultural impact as Gutenberg technology.

Computers have the ability to simulate the details of any descriptive model, which means that the computer itself can be *all other media*. It is a *metamedium* that can respond to a user's queries and experiments. Computers can involve the user in a two-way conversation. A second key characteristic of a personal computer would be the size of the medium. Inspired by Moore's law, Kay envisioned this new medium as physically the size of a three-ring notebook with a touch-sensitive liquid-crystal screen and keyboard for inputting information. Dynabooks would need a programming language and interface that children could understand.

Graphical interfaces

In addition to encouraging PARC researchers to develop small computers, Kay and his team created graphical interfaces and the Smalltalk programming language. During his early years at PARC, Kay was interested in developing a totally visual computer language. The work of psychologists Piaget and Jerome Bruner showed that one of the dominant thinking modes of children is visual imagery. Therefore, Kay's team incorporated visual icons into the system's interface to make the machine easier for children to operate. Children could first manipulate things on the screen with their hands, and then use abstract symbols.

Kay's theory for designing interfaces was primarily based on Bruner's educational theories. In his book *Toward a Theory of Instruction*, Bruner examined the cognitive processing children use for activities such as problem solving, conceptualizing, and thinking. Bruner's work was built on Piaget's pioneering efforts. But Piaget and Bruner differed in their approach to learning mentalities. Piaget believed that children and adults have a single mentality that progressed through a series of stages until the child reaches maturity. He identified these as the action stage, visual stage, and logical stage. Children begin understanding the world in a doing stage, in which thinking is action. For instance, children dig holes and grasp objects. Consequences of actions have no meaning for the child because he or she just acts. In the second stage, the child's understanding of the world is dominated by visual information. A taller glass has more water in it than a smaller and wider glass because tall thin glasses ''look'' larger. Around the age of 11 or 12, the child begins to enter a facts and logic stage that is removed from the visual environment. A child's single mentality progresses from a moving to a logic stage.

In contrast, Bruner argued for the existence of multiple mentalities, which suggested a model for computer interface design. Bruner studied children by examining how the child stores and retrieves information. From his studies, Bruner suggested that there are probably three ways in which people translate experience into a model of the world. The first stage Bruner called *enactive*, or learning through action. In the enactive stage, representation is based on learning a set of responses, which become habitual. The second stage, *iconic*, uses a system of representation that is dependent on visual and other sensory organization of experience. Finally, in the

third *symbolic* stage, representation is established through words or language.

However, there is a major difference between the first two stages of representation and the third symbolic stage. In symbolic representation, the words used to represent the world are arbitrary. The arbitrary meanings assigned to words introduce syntax and grammar into the process of perception. For example, language introduces lawful syntactic transformations that make it easy to make declarative propositions about reality. Bruner stated:

> We observe an event and encode it—the dog bit the man. From this utterance we can travel to a range of possible recodings—did the dog bite the man or did he not? If he had not, what would have happened and so on? Grammar also permits us an orderly way of stating hypothetical propositions that may have nothing to do with reality.[14]

According to Bruner, the ability to view the world through language and the transformation of sentences creates a concept of reality that is not possible with actions or images. However, before children learn to use symbolic language, they first progress through the earlier stages. The child moves from an understanding of the world that is based on enactive and visual representation to a worldview that is abstracted from reality. Building on Bruner's work on representation that describes the enactive, iconic, and symbolic learning mentalities, Kay created a model for computer interface design. The model was described by using the slogan "*Doing* with *Images* makes *Symbols*" (see Figure 5). Kay described this method as follows:

> Now, we agree with the evidence that the human cognitive facilities are made up of a doing mentality, an image mentality, and a symbolic mentality, then any user interface we construct should at least cater to the mechanisms that seem to be there. But how? One approach is to realize that no single mentality offers a complete answer to the entire range of thinking and problem solving.[15]

DOING	mouse	enactive	Know where you are, manipulate
with IMAGES	icons, windows	iconic	recognize, compare, configure, concrete
makes SYMBOLS [computer language]	Smalltalk	symbolic	tie together long chains or reasoning, abstract

Figure 5. Kay's DOING with IMAGES makes SYMBOLS model.[16] (Courtesy of Alan Kay.)

From Kay's perspective, the process of thinking and problem-solving utilizes several different mentalities. He argued that the most important creative work in disciplines, such as science and music, are done in the initial two mentalities, doing (enactive) and image (iconic). The creative problem-solving process used in these disciplines is not at all linked to the symbolic mentality. Furthermore, he argued that there is a basic difference between using the iconic and symbolic mentalities, because the visual system is interested in everything in a scene. Eyes dart all over the scene and change directions. People using the iconic mentality tend not to get blocked because there is always something new and interesting that appears and distracts. In contrast, the job of the symbolic system is to stay within a context and make indirect connections. As a result, it is difficult for people using iconic mentality to get anything accomplished because they get distracted. Conversely, people who use the symbolic mentality are good at finishing things because they can focus for long periods of time on a single context, but they have a hard time being creative because they tend to get blocked. Because none of the mentalities are supremely useful to the exclusion of the others, the best interface strategy would be to force a synergy between them.

To create a synergy, Kay began to incorporate the three different levels of representation into an interface design. The mouse would be a form of enactive representation to actively navigate and manipulate text and display icons on a computer screen. Icons and windows were incorporated into the design as a level of iconic representation. Icons, the mouse, and the Smalltalk programming language were developed to handle the different levels of representation.

Smalltalk

The Smalltalk programming language was developed as the result of a bet. Originally, Kay thought that Smalltalk would be an iconic programming language and would take two years to develop. However, after a hallway discussion with programmers Ted Kaehler and Dan Ingalls, the idea changed. They challenged Kay to define a new computer programming language on a single page. For two

weeks, Kay worked from four in the morning until eight at night to solve the problem. The end result was the outline for Smalltalk, which Ingalls then implemented.

Smalltalk was further developed by the Learning Research Group (see Figure 6). Members included Kay, Ingalls, Diana Merry, Kaehler, Adele Goldberg, and Chris Jeffers. Various interns and summer students also worked with the group. Team members came from a variety of backgrounds—Bob Taylor referred to them as ''the lunatic fringe.'' Goldberg, Kay's co-leader of the group, was an educational specialist and Merry a former secretary. Ingalls, who took on the role of transforming many of Kay's ideas into reality, eventually became a main designer of the Smalltalk programming system.

The Smalltalk system, designed to be an integrated programming environment, consisted of a programming language, debugger, object-oriented virtual memory, an editor, screen management, and user interface facilities. Everything in the system was accessed as an object that could be manipulated by the programs within the system. The Smalltalk system had 100 to 200 classes including streams, processes, files, rectangles, and splines.

Smalltalk was selected as an innocuous name for the program because other programming systems were often named after Greek gods, such as Zeus, Odin, and Thor. Over the next 10 years, Kay and his team programmed over 80 variations of Smalltalk, which featured visual feedback and accessibility to novice users. Kay stated:

> The object-oriented nature of Smalltalk was very suggestive. For example, object-oriented means that the object knows what it can do. In the abstract symbolic arena, it means we should first write the object's name (or whatever will fetch it) and then follow with a message it can understand that asks it to do something.[16]

In Smalltalk, an object is an abstraction of a computer capability. For example, objects can represent numbers, lists, text strings, dictionaries, spatial locations, areas, text editors, processes, compilers, and debuggers, plus all the other components the system requires. Once users understand objects and messages, the entire system becomes accessible. Thus, computer users could now identify an iconic object displayed on the computer screen and learn the types of messages that apply to the object.

The Learning Group Members:

Alan Kay, Head	Adele Goldberg
Dan Ingalls	Chris Jeffers
Ted Kaehler	Diana Merry
Dave Robson	John Shoch
Dick Shoup	Steve Weyer

Students Included:

Tom Horsley	Barbara Deutsch
Steve Saunders	Steve Purcell
David C. Smith	Bob Shur
Radia Perlman	

Child Interns Included:

Marian Goldeen (age 13)	Dennis Burke (age 12)
Bruce Horn (age 15)	Susan Hammet (age 12)
Kathy Mansfield (age 12)	Lisa Jack (age 12)
Steve Putz (age 15)	

Support from Other PARC Groups Included:

Dave Boggs	Patrick Baudelaire
Larry Clark	Bill English
Peter Deutsch	Bob Metcalfe
Butler Lampson	Alvy Ray Smith
Bob Sproull	Larry Tesler
Chuck Thacker	

Figure 6. The Research Learning Group and Outside Support. (Courtesy of Alan Kay.)

Smalltalk was the first dynamic object-oriented programming language (OOPL). *Object-oriented* refers to software design based on objects that know what they can do. First, the user identifies the object by its name or the command that will fetch it. This is followed by a message the object can understand. For example, calculation objects understand mathematical messages. In the object-oriented model, the object is first selected and then the user can be provided with a menu of the commands the object can perform. The object comes first and the action second.

The Smalltalk programming language was developed first, and it was then used to build an operating system and graphical interface. With Smalltalk, the Xerox researchers built an entire programming environment that included editors, debuggers, and compilers. In turn, they used these tools to implement several large-scale applications, such as paint, music, and animation systems.

Early versions of Smalltalk were tested with school children from a local junior high school. Kay believed that a personal computer's content was its ability to create dynamic models, which includes the interactive tools. To become literate in this new medium, children would need to create these tools

themselves. In the test groups, children were taught programming by working from examples of more advanced programs. For instance, the children were taught how to animate a simple box. Soon the children were creating paint, music, illustration, and animation tools. Over a four-year period, Kay and his team invited more than 250 children aged 6 to 15 and 50 adults to try versions of Smalltalk with its interface (see Figure 7). The team then used the visitor's suggestions and projects to improve the design.

These experimental projects included programs for home accounts, information, drawing, painting, music synthesis, writing, and games. The team, however, encountered the following problem: in what order and depth should programming be taught to children? Early successes could not be easily extended to larger groups. To solve this problem, Goldberg developed a series of exercises and design templates to teach children how to program in Smalltalk. Using the templates, "the children could look at a situation they wanted to simulate, and decompose it into classes and messages without having to worry just how a method would work."[17] Planning could be done in English, and the notes could later guide the writing of the code. But, the team was still not happy with the results.

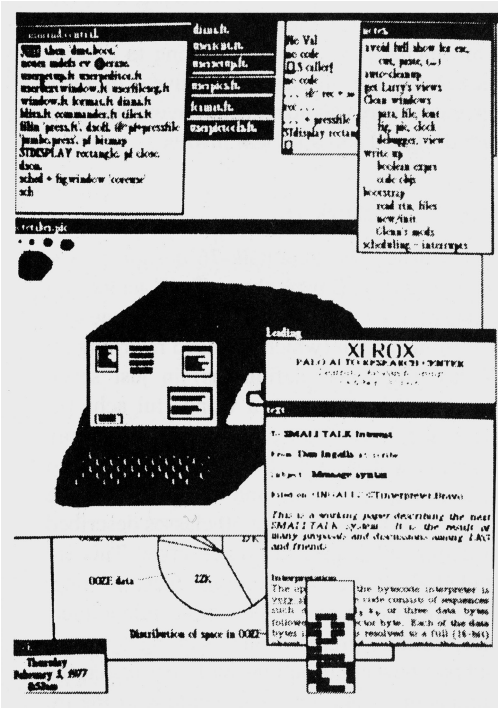

Figure 7. Ingalls' and Kay's Smalltalk-72 Interface. (Courtesy of Alan Kay.)

By this point, Kay and his team had developed the first graphical interface using a sophisticated object-oriented programming language. A direct relationship exists between the philosophy of object-oriented programming and the way people interact with a GUI. In GUI design, a user first selects a visual object on the computer screen with a mouse. The user then chooses an action from a list of command menus to make the object do something. For example, the user can select an icon of a document with the mouse, and then go to a command menu and choose "open." The document will then open and appear on the computer screen. In addition to icons, the team integrated modeless interaction into their designs.

In the 1970s, the Smalltalk object-oriented programming environment seemed fanciful. However, two aspects of the system—its GUI and object-oriented programming—now define the essence of contemporary personal computing.

Alto: An interim Dynabook

Xerox's corporate vision was to invent technologies for electronic offices. The electronic office concept included capturing, viewing, storing, retrieving, processing, and communicating information. However, the vision of the Computer Science Laboratory at Xerox PARC was much larger than office technologies.

Smalltalk ran on the Alto computer, which was designed as a desktop computer with a specially constructed monitor. It was developed as an interim Dynabook: a self-contained system that was essentially a small box in which disk memory could be inserted. Each disk had 1,500-page equivalents of storage. In contrast to ordinary terminals then available, the PARC researchers developed a system in which documents looked as if they were professionally typeset. The incorporation of typeset documents and graphic images fit with the Dynabook that Kay envisioned. The Alto also had a mouse and the popular desktop environment of icons, folders, and documents.

The Alto (see Figure 8) was the first distributed personal computer system. Distributed personal computing systems are based on workstations, interconnected local networks that provided high bandwidth communication, and network servers that provide shared capabilities.[18] Inherent in the Alto's design was the vision that computers could be used to help people think and communicate. In just

over three months, Chuck Thacker developed the brilliant design and realization of the Alto. Thacker did most of the work himself and was assisted by Kay, Taylor, and Butler Lampson, who designed Engelbart's operating system.

During 1971 and early 1972, researchers working in the Computer Science Laboratory spent most of their time developing a set of hardware and software facilities to support future work. Hardware characteristics were determined by the needs of the software systems. Lampson, Kay, and Taylor were involved in the machine's design criteria. Their visions were implemented by Thacker, with help from Ed McCreight, who worked on the memory disk. The researchers wanted a system that could supply an entire work environment for its user, including communications capability, text and graphics manipulation, and computing. Moreover, the system should be a personal system and communications device.

> It was hard for people to believe that an entire computer is required to meet the needs of one person. The prevailing attitude was that machines are fast and people are slow; hence the merits of time-sharing, which allows one fast machine to serve many of the slow people. … When the machine is required to play the game on the human's terms, presenting a pageful of attractively (or even legibly) formatted text, graphs, or pictures in a fraction of a second in which the human can pick out a significant pattern, it is the other way around: people are fast, and machines are slow.[19]

In contrast to earlier computer systems, the Alto attempted to emulate many of the characteristics of paper. The Alto keyboard resembled a typewriter with eight uncommitted keys that software could use as function or option keys. Information was also input with a five-finger keyset, which Engelbart had successfully used in his NLS system developed in the mid-1960s. However, the use of this device required training, and it never became popularly accepted. In contrast, the mouse became widely used by the PARC researchers and later by personal computer users. At PARC, Engelbart's analog mouse was converted into a digital device for use with the Alto.

Original machines contained 128,000 bytes of main storage with a 2.5-million-byte cartridge disk. The main memory was later expanded to 512,000 bytes. Alto's processor was designed to be flexible and expandable. It was microcoded to enable the researchers to experiment with new instruction sets and I/O

Figure 8. Xerox Alto II Workstation. (Courtesy of Computer History Museum.)

devices. According to Thacker, a key difference between Alto and other minicomputers of its time was

> that the microprogrammed processor was shared between the emulation of a target instruction set and the servicing of up to fifteen additional fixed-priority tasks, most of which were associated with the machine's input-output devices. Task switching occurred rapidly, typically every few microseconds.[20]

The Massachusetts Institute of Technology Lincoln Laboratory's TX-2 had previously used this technique successfully.

The Alto was designed to be a computing system that could be adaptable by programming various tasks. However, it was developed to let users interact with the system as well as to program it. The system was programmed in a variety of languages including BCPL (Basic Combined Programming Language), Mesa, and Smalltalk. Each language had its own instruction set and operating system. The entire system had a common file system format on the disk along with network protocols established in the BCPL operating system. Documentation for the Alto's various programs and subroutines was collected into a reference manual. The Smalltalk develop-

ment team, however, was a driving force behind the development of a consistent programming and interface style.

The Alto was connected to an 8.5 × 11 in. display screen. Although a television set is an analog technology, the electron beam and picture elements can be used in digital applications. Computers can be programmed to turn on or off picture elements displayed on the screen. The screen positions can then be stored in the computer's memory and saved for later use. This technique is called bit-mapping because there is a one-to-one correspondence between the bits in the computer's memory and the picture elements displayed on the screen. Additionally, the Alto's developers used a raster scanner rather than lower-cost calligraphic techniques that were available at the time. Display resolution was 606 pixels horizontally by 808 pixels vertically. The monitor looked like a television screen turned sideways to support the display of a full electronic page of text. The display image itself was refreshed directly form the main memory in a method called bit-mapped raster scan.

Because of the limited amount of information the screen could display, the PARC researchers began to think about the screen in terms of a physical desktop. Kay realized that people pile pages on top of each other on a desk; he thought that similarly, they could place windows atop each other on the screen. The idea of overlapping windows, originally developed in 1971, was applied to the screen display. In fall 1974, Ingalls created a program called BitBlit, short for *bit boundary block transfer*, as a way to shift rectangles on the Alto's bitmapped display from one location to another in a single operation. BitBlit consolidated rectangular drawings into efficient pieces of microcode. Kay and his team worked BitBlit operations into the user interface design. Different programs were placed in multiple windows, which were effortlessly shuffled on the screen like pieces of paper. The bit-mapped display refreshed the window that popped to the top, but it let the user see other windows in the background. Using dynamic overlapping windows, the researchers wanted to be able to do animation in 2-1/2 dimensions and refresh objects to the screen in different orders.

Printing

In addition to designing a system that was easy for users to operate, communication was an important goal for the Alto designers. Printers were developed to share information in a paper format, and networking was designed for sharing electronic documents and resources. Designing practical printers that could make high-quality hard copy of the images created with an Alto was a challenge. The first imager developed was an asynchronous low resolution (200 dots per inch) device that drove the Xerox Graphics Printer. A second imager was developed to store thousands of typeset pages that could be printed at a rate of 40 pages per minute in a quality close to xerographic copies. Additionally, Alto documentation could be stored in the system and printed on demand.

Although their goal was to develop personal computing, the researchers did not want to lose time-sharing systems' capability to share information and physical resources. To meet this need, Xerox PARC developed Ethernet as the hardware base along with PUP (PARC Universal Packet), which preceded TCP/IP (Transmission Control Protocol/Internet Protocol). PUP's creation enabled the researchers to create a country-wide internet before the actual Internet was started. In 1976, PARC researchers had Altos on their desks, laser printers for publishing documents, and electronic mail. All of these communication tools were central to the PARC researchers' work activities.

NoteTaker

By 1976, Kay realized that Smalltalk did not meet his goals for designing a computer language that children could use to read and write. The future for Smalltalk was in revamping the program for adult operators, and Ingalls transformed it into a full-service programming language (see Figure 9).

Smalltalk-76 also marked the end of the effort of trying to create a language that children could use. Smalltalk-76 was 200 times faster than Smalltalk-72. Ingalls described Smalltalk-76 as follows:

> Communication is the metaphor for processing in the Smalltalk language. The communication metaphor supports the principle of modularity, since any attempt to examine or alter the state of an object is sent as a message to that object, and the sender need never know about internal representation. Every object in Smalltalk is created as an instance of some class. The class holds the detailed representation of its instances, the messages to which they can respond, and the methods for computing the appropriate responses. The class is the natural unit of modularity, as it describes all the external messages understood

by its instance, as well as all the internal details about methods for computing responses to messages and representation of data in the instances.[21]

As Ingalls further developed Smalltalk, Kay shifted his attention toward a new project called NoteTaker, another notebook-sized computer. The central idea was to take a percentage of the Alto's functionality and put it into a compact portable or "luggable" machine. The NoteTaker was designed and built by Doug Fairbairn. Its design included a custom-built touch-sensitive display screen that eliminated the mouse; it also featured stereo audio speakers with a built-in microphone, 128,000 bytes of main memory with a rechargeable battery, and an Ethernet port. NoteTaker ran the Smalltalk multiprocessor architecture that was closer to the Dynabook design. The NoteTaker design approach was later picked up by Osborne Computer.

The actual NoteTaker was a plump attaché case that looked like the first generation of "luggable" computers, which were created six years later. The case flipped open with the screen and disk drive set in the larger half to face the user when the box was laid flat on a table. The smaller section contained the keyboard, which connected to the larger half by a flexible cable. NoteTaker was so difficult to carry that Larry Tesler and Fairbairn built a rolling cart for it. The cart enabled them to slide the computer under an airline seat. It has been reported that Fairbairn was the first person to use a personal computer on an airplane because he turned on the battery-powered NoteTaker during a flight from California to Rochester, New York.[10] Ten NoteTakers were built and team members flew around the US trying to get Xerox managers interested in the product. Xerox, however, never produced NoteTakers.

After PARC

In 1979, Steve Jobs viewed a demonstration of the PARC technologies and applied many of the concepts to the development of Apple's computers. The following year, Kay took a sabbatical from PARC, leaving Adele Goldberg in charge of the group. Goldberg was instrumental in creating the first commercially successful version of Smalltalk (Smalltalk-80), later marketed by a Xerox offshoot called ParcPlace. For almost 30 years, Smalltalk has been the language of choice for high-complexity applications, such as Texas Instruments' semiconductor manufacturing system.

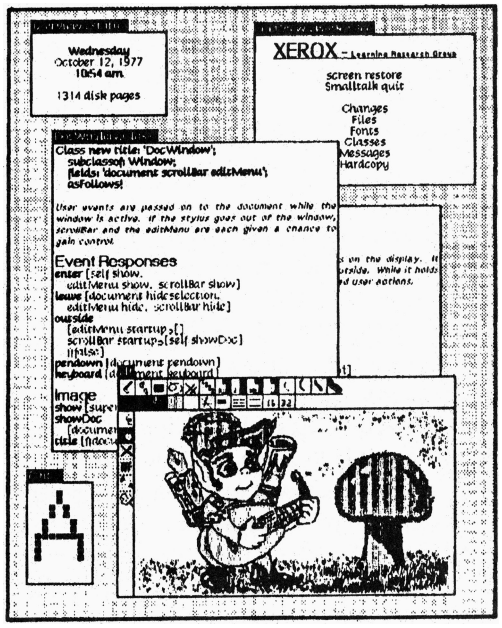

Figure 9. Smalltalk-76 user interface displaying a variety of applications, including a clock, font editor, painting editor, iconic menus, and overlapping window interface. (Courtesy of Alan Kay.)

Kay and the PARC researchers never fully developed a portable Dynabook, but they did create the underlying technologies that paved the way for personal and portable computing. Kay left PARC in the early 1980s to move to Los Angeles, take time off, and take organ lessons. In 1983, he worked as chief scientist for a year at Atari before joining Apple Computer. Introduced in 1992, Apple's PowerBook turned the Macintosh into a notebook-sized computer that, ultimately, turned Kay's Dynabook concept into a reality, except PowerBooks did not weigh two pounds. While at Apple, Kay assembled a team, including Dan Ingalls, Ted Kaehler, John Maloney, and Scott Wallace, to develop Squeak (http://www.squeak.org). Squeak is an open-source-code Smalltalk language that is available through the Internet. One purpose of the Squeak project is to create an educational platform that can be programmed by children and anyone not technically inclined.

In 1997, Kay moved his team to Disney's Imagineering division and continued to work on the Squeak project. Five years later, Kay set up Viewpoints Research Institute (http://www.viewpointsresearch.org), a nonprofit organization dedicated to supporting media in teaching and developing interactive constructive educational environments for "children of all ages."

Today, Smalltalk-80 is not designed to be accessible to children. But, Kay's Squeak e-toys program based on Smalltalk is for children, and Squeak is currently being used by children in the classroom.

Kay's largest contribution to computer science is his commitment to turning the computer into a dynamic personal medium that supports creative thought. Additionally, he continues to explore ways in which computers can be accessible to children.

In 2004, the ACM honored Alan Kay with the prestigious Turing Award for his pioneer work on personal computers. That same year, he also won the Charles Stark Draper Prize from the National Academy of Engineering for the development of the Alto, and the Kyoto Prize laureate in advance technology from the Inamori Foundation. Currently, Kay is the president of Viewpoints Research Institute. His deep interest in children and education still remain a catalyst for his ideas and a source of personal inspiration.

Acknowledgments

I would like to thank Bruce Austin and the anonymous reviewers of this article for their comments. In addition, I thank Alan Kay and Kim Rose for their suggestions and help with the development of this article.

References and notes

1. D. Shasha and C. Lazere, *Out of Their Minds*, Springer-Verlag, 1995, pp. 38-54.
2. J.C.R. Licklider and R. Taylor, ''The Computer as a Communication Device,'' *Science and Technology*, April 1968. Also available at http://www.ece.rice.edu/~dhj/History/licklider.pdf and http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-061.html.
3. K. Nygaad and O. Dahl, ''The Development of the Simula Languages,'' *History of Programming Languages*, R.L. Wexelblat, ed., Academic Press, 1981, pp. 439-480.
4. A.C. Kay, ''Microelectronics and the Personal Computer,'' *Scientific Am.*, Sept 1977, pp. 231-244.
5. S.B. Barnes, ''Douglas Carl Engelbart: Developing the Underlying Concepts for Contemporary Computing,'' *IEEE Annals of the History of Computing*, vol. 19, no. 3, 1997, pp. 16-26.
6. C.L. Baker, ''JOSS—Johnniac Open-Shop System,'' *History of Programming Languages*, R.L. Wexelblat, ed., Academic Press, 1981, pp. 495-507.
7. S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*, MIT Press, 1980.
8. A.C. Kay, ''User Interface: A Personal View,'' *The Art of Human-Computer Interface Design*, B. Laurel, ed., Addison-Wesley, 1990, pp. 191-207.
9. A. Kay and A. Goldberg, ''Personal Dynamic Media,'' *Multimedia: From Wagner to Virtual Reality*, R. Packer, and K. Jordan, eds., W.W. Norton & Co., 1977, pp. 173-184.
10. M.A. Hiltzik, *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age*, HarperCollins, 1999, p. 122.
11. A. Kay, ''The Early History of Smalltalk,'' *Programming Languages*, T.J. Bergin, and R.G. Gibson, eds., ACM Press, 1996, p. 551.
12. F. Rose, ''Pied Piper of the Computer,'' *The New York Times*, 8 Nov 1987, p. 60.
13. M. McLuhan, *The Gutenberg Galaxy*, Univ. Toronto Press, 1962.
14. J. Bruner, *Towards a Theory of Instruction*, Harvard Univ. Press, 1966.
15. A. Kay, ''User Interface: A Personal View,'' 1990, p. 195.
16. A. Kay, ''User Interface: A Personal View,'' 1990, p. 197.
17. A.C. Kay, ''The Early History of Smalltalk,'' *ACM SIGPLAN Notices*, vol. 38, no. 3, Mar 1993.
18. B.W. Lampson, ''Personal Distributed Computing: The Alto and Ethernet Software,'' *A History of Personal Workstations*, A. Goldberg, ed., Addison-Wesley, 1988, pp. 293-333.
19. C.P. Thacker, ''Personal Distributed Computing: The Alto and Ethernet Hardware,'' *A History of Personal Workstations*, A. Goldberg, ed., Addison-Wesley, 1988, pp. 267-288.
20. Ibid., p. 273.
21. D.H.H. Ingalls, ''The Smalltalk-76 Programming System Design and Implementation,'' *Conf. Record of the Fifth Ann. ACM Symp. on Principles of Programming Languages*, ACM Press, 1978, p. 2. http://users.ipa.net/~dwighth/smalltalk/St76/Smalltalk76ProgrammingSystem.html.

Susan B. Barnes is the associate director of the Social Computing Lab and a professor in the Department of Communication at the Rochester Institute of Technology. She is the author of *Online Connections* (Hampton, 2001) and *Computer-Mediated Communication: Human-to-Human Communication Across the Internet* (Allyn & Bacon, 2003).

Readers may contact Barnes about this article at sbbgpt@rit.edu.