APPLE'S NEWTON: HANDYWIDGET IN SIGHT

Two years ago (June 1990) we wrote an issue about HandyWidget -- an intelligent handheld pc on a stick that could manage your communications, your calendar, pretty much your life. At that time, it was a product spec for the future rather than a sneak preview of anything real. We had vague hopes that someone would build a product like it -- perhaps an extension of what is now Perspective from PenSoft for PenPoint, or whatever was causing Apple’s Larry Tesler to smile mysteriously.

With the announcement of the Newton technology -- not a machine -- last week, HandyWidget now has a name, even though it is still some time off. While the initial target machine is a small palmtop with wireless communications and handwriting recognition, that is only one body; Newton technology could be ported to almost any hardware. Newton’s soul is an object-oriented operating system imbued with intelligent AI. Its modular architecture can accommodate multiple recognizers and contexts which can recognize what a user is trying to accomplish and complete much of it for him.

Aside from its behavior, which we covered at length in 1990 and will describe briefly here, Newton holds real promise for a new third-party software industry focused on content-based applications. This is not putting books into CD-ROM with an index and hypertext, but rather providing unique combinations of situation-specific smarts and data -- knowledge of geography and customer locations, say, combined with the ability to dial into Easy Sabre to plan and book a trip -- subject to the user’s approval. This promises a huge market of smallish, high-priced niches rather than the horizontal cut-throat markets for release 6.0 of yet another spreadsheet.

Current concerns

Will Newton take market share from IBM, Compaq and the Mac? asks the press. Initially, its under-$1000, handheld implementation is more likely to take share from the makers of cappuccino machines. If Apple and its competitors do their jobs right, this new class of machines won’t steal market share from pcs; it will create a new market.

It won’t replace cappuccino machines functionally, but it may replace them in the budgets of that segment of the population that likes gadgets and wants to be ahead of the pack.

In fact, says Apple’s Tesler, people who already have pcs tend to notice Newton’s deficiencies -- limited storage, limited functionality, small screen. People who don’t already have computers...
tend to notice the things it can do -- store names and addresses, dial the phone, keep track of things. It's like automatic cash machines: People who have personal secretaries to do their banking for them may have complained about their inadequacies, but the far larger population of people who have to wait on line at banks recognized their utility immediately. Both ATMs and Newtons (will) make life more convenient for normal folks.

The goal of Newton is not just to offer the user tools, but to do the work -- within limits, of course. Anything that the user or a third-party developer can express explicitly, it can do. The challenge for third-party developers over the next few years will be defining and expressing (in code and data structures) a wide-variety of such tasks to automate.

The difference between a tool and an assistant is that a tool helps you do the work, while an assistant does it for you.

Natural market segmentation

In the short term, of course, Apple will naturally enough target the initial embodiment of the technology at a market niche of early adopters, and there won't be much application software to begin with. The trick will be to generate enough excitement about the machine to create the prospect of an attractive installed base -- and enough understanding of the technology to lure the right kind of developers. At Lotus, for example, the appropriate group would more likely be the Marketplace business (now regrouped elsewhere) rather than the spreadsheet people. But just as an upstart (Aldus) had more to do with the Mac's success than did any established software vendor, so will new players come to the fore through and for Newton.

Apple expects the hardware to become a commodity. It has recognized this with its licensing policy, trying to retain some control but broadening its reach out beyond its own grasp. Instead, Apple will attempt to retain more of the software value-added. Won't third-party developers resent and fear Apple if it competes with them -- especially if it competes more successfully than Claris? Perhaps, but third-party developers resent Microsoft, and Microsoft is still doing okay. In the end, developers look at the appeal of the platform and its installed base; the behavior of the platform vendor is secondary. Moreover, the basic premise of Newton is a much more software-value-intensive business.

Natural niches

Yes, Apple itself will supply the basic note-taker, calculator and other modules (although there will surely be competitors as well). The Newton aftermarket will be a market of niches. As noted, one of the business benefits of Newton is that it will naturally foster market segmentation since so much of its value is in its specific knowledge. More importantly, the fundamental role of "applications" will change. This will be a different market from the current one where a single package can hold 50 percent or more of broad, billion-dollar markets such as spreadsheets or databases.

The fundamental feature making Newton a personal product will be not its form factor but software consisting of thousands of fragments and niches:

**Release 1.0**

15 June 1992
more knowledge and less generality. The current vision of such a market is mostly just data-based (or discrete applications, such as HyperCard stacks), but there are other richer forms of industry knowledge: rules of thumb such as how to follow up on a press release (don't call!), how to schedule a deposition, manage an ice-cream franchise, evict a tenant or open a new sales territory. Or consider the context-specific intelligence in MapInfo's geographical programming language MapBasic. MapInfo's value-added is not just mapping and geographical data, but the kind of geographical intelligence a trip planner might use: Which clients' offices lie along the route between Denver and Aspen? What's the closest airport to Bratislava with international flights? Do we have a sales office near this potential new client?

Thus there will be, say, five map-interpretation tools and hundreds of map vendors, selling everything from Palo Alto street layouts with buildings for sale highlighted and attractiveness computed (using your own weights and criteria with the application's data), to census/marketing data for specialty ice cream franchises, Paris metro maps (Apple's example) with path-finding tools, or maps of Moscow with a filter/highlighter so that Juan and Alice can pick out swimming pools and computer companies. There will be 10 or 20 drawing tools with specific recognizers and capabilities, and hundreds of packages of objects and icons for those packages to use.

Brains and brawn

There will be room for modules for hundreds of industry sectors, geographical modules for planning trips to various areas (complete with updated hotel lists, airline schedules, etc.) and so on. Applications and data will come in useful combinations, although the data (like human knowledge) should be transferable across applications.

That's especially enticing if you look at where pricing is going in the broad horizontal markets worth billions in revenues and little in profits. Although price points will be somewhat lower, these niches should provide more revenues overall and more profits per revenue dollar, since customers will buy more units (however you wish to measure units) -- for each task, for each location, for each segment of a large body of knowledge.

Remember that wonderful phone company ad that showed two address books? The one on the left, shiny and new, costs $10, or maybe $15 in the leather version. The one on the right, battered and dog-eared with little slips of paper sticking out, is, well, priceless. It contains the user's life. How you show that with a Newton, without the dog ears, we leave to BBDO to figure out.

On-the-job training

Market segments of one, of course, are the ideal. Much of Newton's value-added will ultimately be the user's own, the result of years of use and training. (You can simply transfer the knowledge as you upgrade to more powerful machines over time.) Fortunately, Newton will be useful even as it learns on the job by scanning faxes, detecting patterns in the user's calendar and applying situation-specific intelligence to the user's activities.

Release 1.0 15 June 1992
Caveats

When the hoopla is over and the first machines are delivered, Newton will no doubt be criticized for its physical frailties: short battery life, a price higher than Apple (and consumers) hoped, limited capacity. But we repeat: The essence of Newton is its intelligence, not its physical form factor, which will change over time. It's an operating environment, not a machine, and its technology should find its way onto larger, more powerful platforms. Apple is now joining Microsoft as a software vendor, although it has the baggage of a large hardware company both to support it financially and restrict it culturally.

We read all the coverage and were somewhat amazed: All this ink and no real explanation of why the product bears mention. "Pocket organizers will be ubiquitous... Apple..." While we do think the product is exciting, it is not well understood. The major argument for announcing it early, that developers should know about it, doesn't seem to hold. The company is being coy about the development environment, even as it's hazy on the physical specs. With luck that will change quickly, so that the Newton won't merely look like sand in Apple's competitors' eyes. Nonetheless, we present a developer-oriented perspective below.

Perpetually upgradable

Unlike most tools, which can top out fairly easily, Newton should be perpetually upgradable. In the near term, consider it just like a person. You can always add a new module not just with more data but with more smarts -- better rules for inferencing, more common sense, better pattern recognition, up-to-date insights into industry conditions or world affairs.

In the long term, Newton may top out in smarts, but you can always add more data modules -- even beyond where most humans would reach their limits. Beyond that? Well, there's a question. Can we make a machine smarter than we are? First you have to define "smart."

For practical purposes, Newton currently lies at the dumb side of the chart shown across. But unlike the database on the left with lots of records in it, Newton has the ability to learn more -- and to handle more facts. Over on the right is a smart, sophisticated person who knows how the world works -- at least in a particular industry and a particular country. She can learn more, too, but she'll never be able to remember reliably as much simple information as even the dumbest database with a big storage system.

CYC: A CONTEXT FOR NEWTON

Also on the chart is Cyc, a research project sponsored by MCC of Austin and led by Doug Lenat, who has worked at Stanford since 1978 and at MCC since 1984 (see Release 1.0, 3-86). Cyc is now almost eight years old. Through an MCC partner program, Apple is one of the project's sponsors, along with Microsoft, Bellcore, Boeing Computer Services, DEC, Kodak and Interval Research (see Release 1.0, 4-92). Three years ago Lenat established Cyc-West, a Palo Alto pseudopod of the project where he spends a quarter of his time.

Release 1.0

15 June 1992
Cyc-West is about to take up physical residence in space next to Interval's. Although there's no formal joint research program, Interval and Cyc-West expect to cross-fertilize over pizza and Cokes.

Call Cyc Newton's Mycroft, a mysterious smarter older brother who stands in the background, deferring to the publicity-seeking Sherlock Holmes. Although Newton is not directly associated with the Cyc project (which Apple monitors via its Advanced Technology Group), its position on the chart may help to clarify where Newton is headed. One of these days we will write a full description of Cyc, but here's a brief explanation:

All positions approximate and variable!

This is the kind of illustration it would be so easy to produce on a Newton. The basic idea is to show increasing smarts along the horizontal axis, and increasing knowledge along the vertical. You can add more data to a dumb machine or to a smart one, although its utility is of course greater in a smart system. A database has lots of data but is pretty dumb; a person is quite smart, but can't retain as much information as a fully loaded Cyc. Much of Cyc's knowledge is expandable; it knows about birds or invoices in general, so it can make conclusions about a particular one. By contrast, a database contains more bits, but can't generalize too well.
The idea behind Cyc (both short for enCYClopedia and a play on psyche and cyclic knowledge) is to fill up an inference engine with general knowledge and common-sense rules. That could take almost forever, of course (which is why Newton is perpetually upgradable). But in theory, at least, when Cyc is about 5 percent full, it will be able to start learning by itself. What could it learn from? Should it go sit in a library? Well, to make a useful Cyc, you might hook it up to a Newton and start feeding it a particular person's e-mail and calendar items.

Or to make a Newton useful, you could hook it up to Cyc. Newton's immediate limitations, of course, have more to do with calculation speed and memory than knowledge. The system has fairly severe limits on what it can hold, although it also will have wireless access to larger databases and back-end inference and other engines (such as Cyc itself, in extremis).

Complement to expertise

The basic idea is of a huge knowledge base. It's not an encyclopedia, but rather a complement -- the real-world knowledge you would need to interpret an article in an encyclopedia. It's easy enough to fill up a database with hundreds of megabytes of say, customer data -- much more, for example, than any person could remember. Yet in our limited brains, we somehow manage to keep enough data and content-oriented generalizations (about, say, how people act, how objects behave, how things are related) to make sense of individual bits of data. For example, if you need to go somewhere, you know whether to drive, walk, or make flight reservations. If you get a letter, you interpret it in context: Is it a personal invitation, a press release, a request for a product, a complaint? What do you know that makes you recognize when a letter is not meant for you?

Rather than a database with lots of instances, Cyc is closer to a giant entity-relationship model or a declarative object-oriented database which contains not so much the data itself as information about the classes. Thus it does not know how much an individual is paid, but it might know the appropriate salary range for engineer, that people are paid periodically, and what it means to get paid (in the context of an economic model). It might also know that people get paid more in California than in Ohio -- or Slovenia. The data can get plugged in for a specific case or a specific company; Cyc provides a way of interpreting them, so that a human or an application can communicate with the data more easily.

A fundamental idea in Cyc is that of contexts -- domains in which sets of rules hold true (introduced into Cyc by R.V. Guha). Since the rules and relationships expressed in Cyc are themselves objects, they too can be manipulated and classified and organized into subsets. Thus, expected behavior is different at home and at work, in an airport lounge or on the beach (despite the famous question: Would your mother let you do that at home?).

Thus, Cyc doesn't require global consistency, notes Lenat. "When we get an error, we don't fall apart. Maybe we have to create another context, or we left out an assumption from an existing context." Thus Cyc has an unusual (for now) architecture. It exists in one basic version, with about 100 "laggingly consistent" implementations at MCC and various partner sites (mostly academic or industrial research) throughout the country. "Anything one knows the others eventually know," says Lenat.

Release 1.0

15 June 1992
The system practices replication in a way somewhat similar to what Lotus Notes does, except that Cyc is smarter at finding and resolving inconsistencies -- or accommodating them through multiple contexts. Longer run, there are likely to be Cyces with different "life experiences" and bodies of knowledge or expertise, but Lenat's vision is of a single reference model, since after all storage is cheap and there are few trade-offs to having all the knowledge everywhere. It's this kind of flexible, forgiving, catholic approach that could easily embrace a multitude of Newtons with discrete, different subsets of a broader knowledge base (contexts).

Coming of age

So far, says Lenat, Cyc is learning through a form of "brain surgery" -- where programmers and domain experts operate and put knowledge in wholesale. But it is now getting close to an educational model, where you have teachers who tell it things or better yet, where it can learn by itself from the same kinds of materials humans learn from. Already, the knowledge enterers are not programmers, but normal folks with a normal-folks approach to mundane areas such as physical devices found in homes, human behavior, economics and manufacturing operations.

Cyc is now in its eighth year and contains 200 megabytes of "knowledge" (data and rules) -- plus about 100 megabytes of operating environment. It is written in pure Common LISP and is implemented on a variety of machines; Microsoft has it on a DEC 5000 while Apple has it on a Mac IIfx. Newton, by contrast, is implemented in C and C++ and runs on the ARM 610 RISC chip (for now). It will eventually offer its own development tools and scripting language, but Apple is vague on that right now. One candidate is Dylan, a so-far-experimental LISP-like language under development at Apple's Advanced Technology Group. Its main advantage over LISP is that it is a language, not an environment, and can work easily in standard operating environments (or in Newton's).

NEWTON'S PRETTY FACE: DRAW WHAT I MEAN (DWINDRAW?)

But useful intelligence is hard to achieve in a baby, and Newton is far from reaching Cyc's level, while Cyc is far from reaching Lenat's goal of an easily teachable system. (Cyc's funding is guaranteed through 1995, and sponsors are already starting to develop products to use it as a back-end.)

Thus Newton will need to appeal to its early customers on other grounds than raw intelligence. One is its superior graphics intelligence, given that by mid-1992 everyone will be sending and receiving faxes, deciphering handwriting (to a greater or lesser degree) and managing calendars.

Apple had the benefit of seeing what GO et al. were doing, and realized that Newton's graphics capabilities could provide a useful differentiation in the short term -- until its other capabilities kick in. Newton goes a step beyond recognition of shapes to recognition of graphics properties such as symmetry and alignment. It can guess that certain objects are supposed to be the same size, shape, etc. It can recognize groupings, etc. In spirit, it's a combination of Ashlar's Vellum (see Release 1.0, 12-89) and your standard raster-to-vector drawing tool. For example, if you draw two similar-sized boxes, it will render them as duplicates (unless you turn the intelligence off). If you draw a crooked box with equal sides, it will

Release 1.0 15 June 1992
render it as a rhombus. It can also automatically smooth curves, line up objects horizontally, vertically or concentrically, and recognize reflections (mirror images). If you draw a circle with a few oval things sticking out of it, it will recognize and render an idealized daisy, with petals evenly spaced. (At last! Now we can draw a perfect five-point star.)

Although it's a single module, the graphics intelligence provides a model of how Newton will work. From the user's inexplicit promptings -- some Ink on the screen -- it tries to divine the user's intentions. Certain clues give it a context -- graphics or text, for example. But contexts go beyond that. A landscape architect might have a context -- and a recognizer -- that could identify trees, shrubbery, flagstones, birdbaths and the like. Let it recognize the names of some people in your company and recognize a box, and suddenly you're in the org chart context, ready to move boxes around on the page and create the underlying data structures.

That's why Apple is so proud of its multiple-recognizer architecture -- not because recognizing handwriting itself is so important (although it is), but because the multiple recognizers make it deeply smarter. The "multiple-recognizer" architecture is in fact a multiple-context architecture, which can manage the appropriate situation-intelligent modules that will be supplied by third parties or built in-house to implement corporate procedures and policies.

Inkfree

Like most of the pen systems, Newton can capture, compress and manipulate uninterpreted ink (see Release 1.0, 8-91). However, this is a capability it de-emphasizes (vs. graphics intelligence) precisely because it believes it has something better to offer. The point of Ink is that the user (or recipient) is usually better at interpreting it than a machine could be. However, if the machine can interpret it, then that's probably even better, since the machine can do more with it -- searching it, classifying it, and doing all the other things that require symbolic intelligence. And of course, it can also store the meaning in a smaller amount of bits. For a handheld machine, that's no minor consideration.

KNOWLEDGE OF THE PAST

It's no fluke that Apple should come up with Newton. The idea goes all the way back to the Dynabook, dreamed up over 20 years ago by Alan Kay -- then at Xerox PARC, now an Apple Fellow. It wasn't implementable in those days; it just may be now. It started its present incarnation some time ago under Steve Sakoman, who had run Apple hardware development in the late Eighties. In 1988, burned out on the Mac, he started a small skunkworks to work on a dynabook. This was just about the time Sculley's Knowledge Navigator video came out to general acclaim, and so the project found favor within Apple.

---------------

1 Personally, we found it excessively anthropomorphic. We don't like our agents to smile and crack jokes; we prefer them to know their place. Besides, we thought the resources could be better spent -- as they have been with Newton, which has no cutes.
Sakoman ran the group as a product development effort, but it had no deadlines. Instead, it had exciting research agendas and lots of vision. In 1990 the company reorganized and Jean-Louis Gassee, Sakoman's sponsor, left. Sakoman left too, abandoning the project (and is now working with Gassee at Be Labs). In January 1990, Sculley sent Larry Tesler, Apple's head of Advanced Technology, to assess the project and fix it or fold it. In the course of three months, says Tesler, "we kept the vision but redefined the product. We brought in marketing people. And we looked for someone to run it." No one turned up, and, says Tesler, "Suddenly I woke up and realized I was more attached to it than I recognized. It was an intense three months!" He handed Advanced Technology over to his deputy, David Nagel, and took on the job. Tesler, who had always loved to talk about his work, went silent.

A year ago, the company realized that its edge was almost all in the software, and took on Sharp as its (first) manufacturing partner. Of course, it will also manufacture some models itself.

Advanced technology applied

Meanwhile, Newton's informal relations with Apple's Advanced Technology Group remain strong. Although the Newton group and MCC/Cyc have no direct connection, ATG provides an "impedance match" between the two groups.

Also relevant is Apple's Casper, which is a voice recognizer, but shows Apple has worked with the issues of grammars and trees. When you're in a certain context, the machine expects to hear certain commands in Casper. Likewise, the Newton can look for certain inputs, and then can load in the proper context/recognizer. (Although the user may feel that the system has "no modes," there are certainly the equivalent of modes under the covers.) Although it sounds simple, Casper is evidence of Apple's experience in how to build unambiguous command structures, discriminate between commands and user objects being acted on, and so forth.

Another relevant piece of Apple research is Eager (see Release 1.0, 5-91). This is a system that works within the HyperCard environment and recognizes user actions -- sequences of commands acting on objects. The trick is to define the objects properly so that you can recognize the user's patterns, making them explicit. For example, is the user highlighting words that appear in the company product database, or calling clients near Aspen? (Load in the geographic intelligence!) Does she always send out thank-you letters after sales-call appointments? Note that patterns may occur in a single session, or over time.

You can't give an engineer a typewriter and turn him into a secretary. Nor can you give a secretary your contacts database and turn her into an expert on your life. There's a lot more involved... For example, when Stewart Alsop hired Nanette Buckhout, John Sculley's personal secretary some time ago, he acquired not just her native intelligence, which is considerable, but also her familiarity with a community and the people in it.

To recognize the objects and the actions, you need a class structure that classifies them. You also need frames, or active objects, where each slot is another object and can be acted on and is also a member of a class. This rich structure underlies Newton's intelligence.

Release 1.0 15 June 1992
MODEL FOR THE NINETIES

Manufacturing resource planning (MRP) was a hot market in the Eighties. It continues to be important today, but it is past its era of greatest growth. One company, Aurum Software of Santa Clara, is betting it can repeat a success along the same lines with a new function: customer resource planning. With the very name of CRP, Aurum founder David Buchanan acknowledges his model and pays homage to ASK Computer Systems, as recently described in ASK founder Sandy Kurtzig's book, "CEO: Building a $400 Million Company from the Ground Up," Norton, 1991.

One of the greater Silicon Valley successes of the Eighties was ASK Computer Systems, a founded-in-the-bedroom start-up that became the premier supplier of MRP systems. Founder Sandy Kurtzig started it in kinder, gentler days when it was easier to make and recover from mistakes, basically relying on customer/reseller contracts both to fund products and guide design. Once the firm reached critical mass, Kurtzig moved from strength to strength. A hot start-up in the heyday of venture capital, ASK funded its own growth with almost no outside capital, and finally went public in 1981 through L.F. Rothschild Unterberg Towbin. On the road show, Kurtzig didn't just tell analysts about her great company; she urged them to press their other start-ups to use the ASK system in their own manufacturing operations. Securities analysts took the hint, and soon use of ASK was considered the mark of a savvy company. Overall, ASK had a solid, customer-refined product, and marketed it brilliantly to the hottest sector of the US economy.2

One man who read Kurtzig's book with great interest was Aurum's Buchanan, a district manager in the Eighties for Cincom, a database company that did quite well in the Seventies. Traveling around supporting customers, he had noticed two things: His customers felt increasingly uncomfortable depending on a single database vendor (especially if it happened to be Cincom). Secondly, although Cincom was used mostly in financial and manufacturing applications, these companies were all struggling to support their customers, using mostly paper and other manual methods.

Juan, Jaun, Joan and Alice, Alissa and Elise

Customers were strewn all over the place, like missing inventory. They popped up in salespeople's databases here and there, were logged as phone calls in customer support, were stored as correspondence in marketing files that should have been passed on to QA or product design, and appeared five or six times under slightly different pseudonyms in the lead file. Customers got a different story from each department they called and had no consistent identity across vendor departments -- even though the first commandment of customer support is "recognize thy customer."

2 There has been some history since, to be sure. Kurtzig brought in professional management and left the company, which subsequently fell on hard times. Kurtzig (who had sold all her stock), wrote the book about her adventures, and then was inveigled back to rescue the company, which she is now doing -- not by vainly trying to revive a flagging product and repeat past successes, but by bringing in new products.
At his next job, with network vendor Proteon, Buchanan lost Boeing as customer to a competitor because of poor customer support. His own company just wasn't following up all right, and as a single salesperson, he couldn't coordinate all the activities that should have been performed for this customer. Customer support was everyone's job and no one's responsibility.

Customers are a resource to be managed. They can have value added to them, just like products, with calls, support and other investments. They can be re-used, from one division to another. They can be cultivated. And like products, they can be mismanaged, lost, used inefficiently, processed out of sequence and otherwise spoiled. Just as MRP is not automation but management of a physical process, CRP is not automation of customer support -- you don't have automatons calling the customers or robots fixing the machines -- but management of a physical process -- calling, talking, visiting, writing -- carried out by humans. Yes, CRP can interface to telephones and dialing systems, and word-processors and mailing tools, but in the end it organizes the process so that the people (like the machines on the factory floor) can do their jobs optimally.

Same plot, new cast and props

All this, Buchanan recognized, was an opportunity, and he and his wife Susan founded Aurum in 1990 to exploit it. If the Eighties were the decade of manufacturing and production line rationalization, the Nineties would be the decade of customer support, customer coddling, customer cherishing. MRP was the function of procuring, caring for, allocating and tracking materials and finally turning them into products. But businesses really get rich from customers, not from products.

Funded by his wife, the top salesperson for database/4GL vendor Unify, Buchanan started to build a set of database-independent customer support tools, using Unify's 4GL. Like Kurtzig, Buchanan let his customers do a lot of the work. His wife (now evp at Aurum) called him up one day and said, "Triad Systems needs a support-call tracking system. Can you do a demo for them?" He went to see Triad on Thursday, and promised a demo the following Monday. He started describing his (vapor)system; they interrupted him with some ideas, and by the time he left he had a complete product spec in the rough. He spent all weekend working and (testimony to the power of the Unify tools) he had a working prototype in time for the Monday meeting. That is the basis of the company's SupportTrak, which manages customer support from customer calls and problem analysis to spare-parts tracking.

The company repeated the process for each of its next two products (although the demos were a little solider each time), with customers Sony Communications (QualityTrak, focused on resolving quality problems) and Unify itself (TeleTrak, for lead-tracking and telemarketing; and SalesTrak, which manages accounts and prospects, monitors forecasts and revenues by account, and basically automates as much of each salesperson's paperwork and process as possible -- and thereby ensures that it gets done!)

Release 1.0 15 June 1992
RESOURCES & PHONE NUMBERS

Larry Tesler, Apple Computer, (408) 974-2219; fax, (408) 974-1794
Sandy Kurtzig, ASK Computer Systems, (415) 969-4422; fax, (415) 962-1974
Dave Buchanan, Peter Mork, Aurum Software, (408) 562-6370; fax, 562-6195
Doug Lenat, Cyc-West, (415) 323-6799
David Liddle, Interval Research, (415) 424-0722; fax, (415) 903-2274
Laszlo Bardos, MapInfo, (518) 274-6000; fax, (518) 274-0510
Mike Levinthal, Yogen Dalal, Mayfield Software Partners, (415) 854-5560; fax, (415) 854-5712
Doug Lenat, MCC (Austin), (512) 343-0978, 338-3436; fax, (512) 338-3600

COMING SOON

- Information publishing.
- Parallel computing.
- Commercialization of the internet.
- Infrastructure for groupware.
- Pen stuff.
- Constraint-based reasoning.
- Text tools.
- And much more... (If you know of any good examples of the categories listed above, please let us know.)

--------------------------

Release 1.0 is published 12 times a year by EDventure Holdings, 375 Park Ave., New York, NY 10152; (212) 758-3434. It covers pcs, software, CASE, groupware, text management, connectivity, artificial intelligence, intellectual property law. A companion publication, Rel-EAST, covers emerging technology markets in Central Europe and the Soviet units. Editor: Esther Dyson; publisher: Daphne Kis; circulation & fulfillment manager: Robyn Sturm; executive secretary: Denise DuBois; editorial & marketing communications consultant: William M. Kutik. Copyright 1992, EDventure Holdings Inc. All rights reserved. No material in this publication may be reproduced without written permission; however, we gladly arrange for reprints or bulk purchases. Subscriptions cost $495 per year, $575 overseas.

Release 1.0 15 June 1992
**Release 1.0 Calendar**

<table>
<thead>
<tr>
<th>Date</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 1-4</td>
<td>*GroupWare '92 - San Jose. Sponsor: Lotus Development and SRI. Keynote by Jim Manzi. Speakers include Philippe Kahn, Ray Noorda, Irene Greif, Thomas Malone and Esther Dyson. Call David Coleman, (415) 282-9151 or (800) 247-0262.</td>
</tr>
<tr>
<td>September 22-24</td>
<td>*UNIX EXPO - New York City. Sponsored by Bruno Blenheim. Call Annie Scully, (201) 346-1400 or (800) 829-3976.</td>
</tr>
<tr>
<td>October 11-14</td>
<td>*Third annual East-West High-Tech Forum - Prague. Sponsored by EDventure Holdings. Contact: Daphne Kis, 1 (212) 758-3434; fax, 1 (212) 832-1720; e-mail, MCI 511-3763 or CompuServe 75140,761.</td>
</tr>
</tbody>
</table>

Please let us know about any other events we should include. -- Denise DuBois
Please enter my subscription to Release 1.0 at the rate of $495 per year in the U.S. and Canada. Overseas subscriptions are $575, airmail postage included. Payment must be enclosed. Multiple-copy rates available on request. Satisfaction guaranteed or your money back.

Name ____________________________________________
Title ___________________________________________
Company _________________________________________
Address _________________________________________
City________________________________ State________ Zip________
Telephone _______________________________________

☐ Check enclosed.
☐ Charge my
  ☐ American Express  ☐ MasterCard  ☐ Visa
Card Number ___________________________ Expiration Date ___________________________
Name on Card ___________________________ Signature ___________________________

☐ Please send me information on your multiple-copy rate.

Please fill in the information above and send to:

EDVENTURE HOLDINGS INC.
375 PARK AVENUE, SUITE 2503
NEW YORK, NEW YORK 10152

If you have any questions, please call us at (212) 758-3434.

Daphne Kis
Publisher