An interview with John Walker - Part 1

On Thursday of last week (September 4th, 2008, for those people accessing this post sometime in the distant future :-), I had the great honour of meeting with John Walker, one of Autodesk's founders and its former CEO.

I've known for some time that John lives in the Neuchâtel area, and it turns out he lives just 10 minutes' drive from my home. I decided quite recently to get in touch with John and suggest I interview him for this blog, and I was delighted when he accepted. Not only was John very generous with his time - we talked for four hours - he also shared many fascinating historical tidbits and forward-looking insights into the CAD and software industries.

For those of you who are not aware, John authored/compiled a fascinating book on Autodesk's early history, The Autodesk File, which is just one of the interesting resources available on John's Fourmilab web-site. I strongly recommend anyone interested in Autodesk's business and origins read this book: it provided me, at least, with many valuable insights into the history and culture of the company at which I've chosen to spend so many years of my working life.

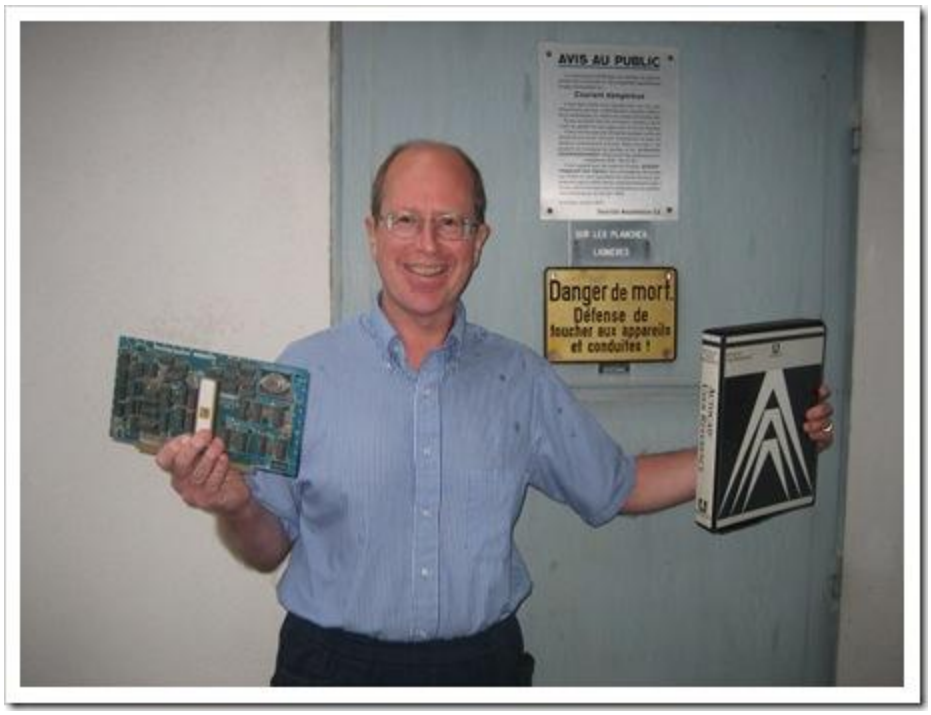
I've just spent three days transcribing the recording I made of the interview. To improve the flow I've edited the order of some of the questions and answers - to say I'm an inexperienced interviewer would be a gross understatement - but I've done my best to leave John's words untouched. I must stress that John's words are his own, and do not necessarily reflect the opinion of current Autodesk management and employees. Whether that's a good or a bad thing I'll leave for you to decide. :-)

At the end of the transcription effort I found myself with 18,000 words of text to publish. Needless to say I haven't put that in a single post, so I'm serializing over a number of weeks: it'll be a regular Friday feature until it's all published, which will also give me the chance to continue posting technical content over the coming weeks. Here are the planned episodes - I'll make sure these become links as the posts come out:

- Part 1 - Autodesk's early history
- Part 2 - AutoCAD's architecture & APIs
- Part 3 - Autodesk's eco-system and strategy
- Part 4 - Past and future opportunities

One last word on the conventions I've used in the below text: my own words - which you'll be thankful I've kept to a minimum - are in **bold**, John's are in normal text. I've enclosed editorial comments *[as italic text in square brackets]*.

And now for the interview itself... enjoy! :-)



Arrival at Fourmilab

[John welcomes me by showing some AutoCAD artifacts...]

AutoCAD 1.3 manual from 1983

At that time there was no customization, except that you could make your own menu. And, surprisingly enough - as you could put multiple commands on one menu line - some people were able to make simple applications. That's what really gave us the... well I think we already knew from the outset that we wanted to make it programmable and extensible, but it was this tremendous cleverness of using something we didn't even think of as an application medium that put the priority on getting there.

AutoCAD 2.1 prototype hardcover manual [which John is holding in the above photo, along with the Marinchip 9900, AutoCAD's first hardware platform]

This is a prototype of the first hardcover manual. For AutoCAD 2.1. If you look at page 200, this is the Variables and Expressions feature, the pre-cursor to AutoLISP. In fact, all of LISP was there, it was just turned off. Primarily to save memory, but also because we didn't have the mechanism to reach in and retrieve entities or create entities and do anything like that. It just didn't make sense to put in a programming language that didn't have access to the drawing database. And so this was kind of half-baked. The people by this time that were doing advanced menu macros were just screaming for ways to turn modes on and off at that point without pumping in commands, so this was just a stop-gap.

I think it was the 2.13 *[*]* release - which was a disastrous release - but I believe that's where full AutoLISP came in. That was really the worst release of AutoCAD ever made - at least during my tenure - from a quality stand-point. There was a horrific bug - in x86 assembly language - and most of this stuff dates from the age of 16-bit machines... when you transfer from one program to another you have to load the segment register. It's two instructions and you have to do it in a particular order and in the AutoLISP code I got it wrong. It's a timing thing - one time in a million you'd get an interrupt between those two instructions and everything would go black. Of course on

a computer, one in a million happens about every hour. So we finally figured that out, but that was really tough. We also had, as I recall, some driver problems, and we changed some stuff in the drivers, so that one - I think it was 2.15 we finally called it when we got it right.

[A follow-up comment from John: It's a little more complicated than this. Variables and Expressions was included with 2.1 in May of 1985. This was the release in which the hardware lock was introduced on international versions of AutoCAD, but the domestic version remained unlocked. AutoLISP was introduced in version 2.18 in January 1986, but did not provide access to the entity database until version 2.5 in June 1986. It was in version 2.5 that the hardware lock was introduced in domestic versions.]*

AutoShade manual

This was perhaps our second *[**]* product. It went through a long, bouncing-around history. We were going to acquire a rendering engine from another company and build an interface around it. That went around and around and around. Their rendering engine was much better than AutoShade in terms of what it could do, but it was just never getting done. So I went off, with one other guy, and built this thing - very, very simple. And actually, I don't know, some of the... at the point we put the Z-buffer shade command in AutoCAD, there really wasn't any need for this anymore. And at that point we were working with the Yost group doing 3D Studio. That was a real ray-tracer engine, and there really wasn't any need for AutoShade anymore.

*[** A follow-up comment from John: Well, actually not. In fact, AE/CADD, CAD/Camera, and AutoSketch all preceded AutoShade.]*

It was actually really embarrassing - we had AutoShade well before we had anything other than the pure extrusion 3D, and so we could make all these beautiful pictures, but you couldn't model them in AutoCAD! And so we jammed in this 3DFACE command, to try to plug that hole.

[Then starts the interview proper...]

Autodesk's early history

How did Autodesk became a CAD company?

The short answer is: COMDEX 1982. Up until that point we were really focused on the product Autodesk [*an office automation system for small computers which only later became the name of the company*] that was going to be the killer product. AutoCAD was kind-of this... it was interesting because we were pretty sure we could get it to market quickly. It and AutoScreen (a screen-based text editor), we were pretty sure we could get those done for COMDEX. It was a real push, but it could be done. Autodesk was a lot more of an ambitious thing. We could show a prototype - in fact we had a prototype - but getting the whole thing done would take a lot of work.

Secondly, and this came back to bite us a few years later, AutoCAD was licensed from Mike Riddle, who was also a founder of the company. Unlike everyone else who joined the company, he did not contribute his software unencumbered: he had a royalty deal associated with it. And that royalty deal was - given the price we envisioned for AutoCAD originally - so onerous that it was a very low profit product for us. Now, when you start out with $59,030, I think it was, and you're spending a thousand or two a month, anything that got money to come in, even at a low profit margin, was attractive. Because that would mean - since everyone was moonlighting - people could quit their jobs and start working full-time. Which would drastically increase the amount of work we could get done.

So it's not like we weren't interested in it. But we had the same opinion as the rest of the industry did. That it's a niche product, it's got nowhere near the market - I mean just compare the number of architects with the number of people that write documents - that's pretty obvious. It's

inherently an expensive product - even if we gave it away it would be expensive, because at that epoch graphics boards with a reasonable resolution and digitizers were very high-end products, you just couldn't buy them cheaply. Most computers didn't have mice - there wasn't any graphic pointing device at all, so it was a battle even to be able to get a pointer on the screen. So that meant the hardware configuration was going to be expensive, and that again was going to reduce the size of your market.

Well, we got this AutoCAD 1.0 done, and in fact, I believe we actually sold a copy of the CP/M version - because we had it on the CP/M, the IBM PC and the Victor 9000 before COMDEX. But I think it was to one of our dealers: I'm not sure you can really call that a sale. But we took it to COMDEX, which at that point was on this exponential growth explosion. 1982 was the last year it all fitted into the Las Vegas convention center. After that they started expanding out to other venues and so forth. We were in the convention center, but were about as far from the action as you could get. Way up against the wall, near - as Barnum used to say - the Egress. Well if you were heading for the egress you would have to pass by our booth. A 10 by 10 foot booth, and I think we had four machines: a CP/M machine - and CP/M at that point had the nicest graphics displays available, so even though today you'd think of it as an archaic 8-bit machine, in fact they'd got the megahertz right up high enough, it was really just as fast as the IBM PC, and most people don't know that it ran in single precision, so the floating point was a lot faster. Really the two were quite comparable, but it had this gorgeous 512 by 512 SCION MicroAngelo display, and not only was that twice the resolution of the IBM PC in CGA mode, it had its own computer onboard and did the vectorization itself. So it could draw lines and circles and things - you just gave it the command, and so that meant you didn't have to set every pixel on the screen, which made it faster still. So that was the machine we did our demos on, basically. But we had it there on the IBM PC and the Victor 9000 and then we had another machine that was running our AutoScreen text editor.

From the day this show opened until the day it closed - which I think was Wednesday through Friday - the booth was absolutely jammed, you couldn't get in there. There were lines of people waiting to see it. Historically, at COMDEX '82 the IBM PC had come out earlier that year, and every computer manufacturer - or would be computer manufacturer - was furiously re-engineering their machines, because they were mostly CP/M machines before that, to make their response to the IBM PC. Better graphics, larger disk, more memory, apples flying out of the top, whatever. COMDEX '82 was when they all launched their products. They were all looking for a killer app. Something that could not only sell their machine but could distinguish their machine - and all the great stuff they've done to it - from the IBM PC. So we had Compaq, Texas Instruments, NEC and Wang, and a dozen others you've probably never heard of, coming by our booth saying "what would it take to get this application on our machine? This is the thing we want to demo, because we have better graphics, we can run faster, etc." And we, of course, being impecunious at the time, said "well, send us a machine and we'll make it run on your machine".

So not long after COMDEX all these machines started arriving and we realized "gee, this is going to be a lot of work". Especially when everybody that saw AutoCAD at COMDEX asked "can you add these twelve features?". Obviously the same people were doing these drivers and the features, so that was a bit of a trade-off.

Driving back - and this is how we got there, in a big station-wagon we rented in San Francisco - there were three of us that made the drive... we'd said we'll do five products and count on one being a hit. We now knew which one was the hit - there just wasn't any question. We maybe sold four or five copies of AutoScreen after that. When we got back, the phone was ringing off the hook with prospective dealers saying "how do I sign up to be a dealer?" or manufacturers saying "OK, how do we get you the machine? When can we have this ready?". From that point on there really wasn't any question that AutoCAD was going to be the product.

Well from the business standpoint - and I have to say I don't think we thought about the business standpoint for the next few months, because we were just trying to keep our heads above the water - we still had this royalty problem. It was then that it occurred to us that while the royalty affected the base product, it did not cover additional features that were sold at an extra price. That gave us a tremendous incentive to add features, and that was the birth of what was called the ADE *[AutoCAD Dimensioning Extension, which was later renamed Advanced Drafting Extension]* packages. Because the ADE packages were not subject to the royalty. One of the first things I did when I got back was to write this really idiot-stupid dimensioning package. It was done in a weekend - that's how simple it was. We sold it for $500 - and that was a $500 revenue that came straight to us and wasn't subject to royalties.

Suddenly the product was actually - with dimensioning, I don't know the numbers off the top of my head - it became maybe twice as profitable as it was before and that's why you had this progression of ADE-1, ADE-2, ADE-3 *[ADE-1 was the original dimensioning feature]*. We hardly ever sold anything but the highest level of the product, because the dealers obviously would demo it at that level, and the customer wasn't going to buy something that didn't have all the neat features that the dealer had shown them.

So that's really how it came to be AutoCAD, and from the weekend of COMDEX there really wasn't any doubt. We did continue to work on Autodesk for a while, but it was just clear now we had a product that people wanted, we had machines coming in that people could work on, we had dealers wanting to sign up. We even had, not long after that, writers contacting us that wanted to do review articles for magazines, and Autodesk was - at best - maybe 6 months to a year of being ready. We just didn't have time to work on it.

In any case, it was clear: we'd priced this thing at $1,000, and people were willing to send us $1,000. Now I don't think we could have got that for Autodesk. So that was another consideration. Autodesk would probably have been a product in the $100-300 niche that word-processors were going for, at the time. And now it would have been very cool to have what was essentially a Macintosh desktop running in text mode on the IBM PC. I still think that would have been a killer product if we'd gotten it done sometime in 1983, but after that point it was a question of, there was one guy who was working on it and he was really needed to work on AutoCAD.

So from that point on we really became the AutoCAD company. Now you asked, how did we become a CAD company? Well that's another interesting story. Because we never thought of ourselves as a CAD company, even then. We were a software company that had this CAD product. And if we ever had time to do anything else we might do other, completely unrelated software products.

Along comes 1985 and it's time to do our Initial Public Offering. Well, there had been a whole raft of PC company - not so much software company, but PC company - stock offerings in '82 & '83. When everybody was jumping in against the IBM PC and the products were being launched. And most of those stocks performed disastrously. It wasn't even so much the companies: there was a heck of a recession going on, at the time, and the stocks were way down. I used to say, at the epoch, if you wanted to clear a room of investors you only needed to say two things: one is "PC" and the other is "retail sales". Because the dealers were collapsing all over the place, too.

So there was no way - we probably could have made the stock offering, but at half the price - if we were perceived as a PC company or a PC software company. We positioned ourselves in the prospectus as a CAD company - "this is the low end of the CAD market" - because CAD was glamorous at the time, workstation CAD: Computervision, Intergraph, CADAM... and that was the hot thing, and that was what investors wanted to see. We were "the first CAD company with a mass market" - that was really the pitch. Interestingly enough, we did our secondary stock offering in 1987, and if you read that prospectus, we're a PC software company. Because Microsoft had

gone public, and Microsoft had taken off in a big way, and PC software was hot. And so we didn't want to be seen as this niche CAD company. And that's what they call marketing. :-)

Autodesk - the company - was one of the most successful high-tech startups of all time, if you consider its lack of debt and external (Venture Capital) investment. Do you think that kind of success is possible to do, these days, without VC?

No! :-)

How did you manage it?

In one word, luck. We were at the right place, at the right time, at a unique juncture in the software business. When I say "no", I don't mean that it's not possible for a couple of guys with a great idea to make billions of dollars - look at Google - but not in the industry we were in.

The difference is, we arrived just at the time that the IBM PC had turned the whole PC business upside down. Suddenly you had this completely new machine, which could not run any of the existing software written for the Z-80 and CP/M, you had 15-20 manufacturers jumping in the market with both feet, Japanese companies bringing in boatloads of these PCs with absolutely no software to run on it, other than CP/M or MS-DOS. And quickly-ported and ghastly versions of WordStar and so forth, which were mostly hack-ported from CP/M and kinda worked, maybe. And there was this huge vacuum awaiting software. If you read the proposals from '82, that was the whole concept of the company: we are at a juncture here. There are millions of these machines coming in that need software. They are not going to sell unless they have software on them. And we - fifteen mainframe programmers - know how to build big software packages. And further, we're familiar with the applications that run on mainframes, and these machines in a couple of years are going to have the power of mainframes and will need mainframe-style software.

It was that game-changing moment. And in that kind of a moment, you don't have to have the production values that are assumed for an entry-level product today. Look at the investment that went, say, into Google Chrome, and they didn't even write all of it - they didn't write the rendering engine in it. And it's getting panned by a lot of people who are saying it's not ready yet.

Back then, at the point we launched AutoCAD at COMDEX, that was the work of three people over about six or seven months. The manual was written by one guy - the original manual, which I don't have - didn't have any illustrations in it. It's the manual of a CAD program, and it didn't have any pictures in it, because we didn't have any way to make them then. AutoCAD didn't work well enough, hardly, to get the pictures in the manual! It was printed out on a daisywheel printer; that's how the master was made. And so you didn't need Venture Capital when you had people - who, at that point, were pretty senior people in their regular jobs - moonlighting and able to throw something like that together. Well, today the investment that goes into the icons on the screen is greater than what went into AutoCAD. Everything is this monster thing.

And also you're on this outrageous and despicable treadmill of Microsoft changing the rules every 6 months, and just the investments it takes to keep up with their disastrous Operating System, if you can call it that, is something that requires probably a company staff larger than Autodesk had in '85 after we'd gone public.

So the game has just become so unbelievably complex and the standards so high.

Also, it was possible, back then, to make noise and be heard without a huge production and advertising budget. All we had to do was take it to a trade show: that cost us about $3,000 - which is a lot when you've got $60,000 in the bank - but it wasn't like the $50M roll-out they do with some of these things today.

And that provided an opportunity. I think you need only look at all the niche businesses that have started up by people putting their hobby on eBay and now they have fifty people on staff and are

selling around the world. In Glenn Reynolds' book, an Army of Davids, which is entirely about the fact that, for example, what does it take to make Hollywood-quality film? Well, a camcorder and a copy of Adobe Premier! So independent film-making is really independent, because it's a question of "do you have the talent?" as opposed to "do you have the budget?". That's the kind of game-changing thing that happened to us, at that point for the PC software industry.

But I've said many times, what was responsible for our success? Luck. And what particular piece of luck? The IBM PC AT coming out when it did. Prior to the AT, it was just so painfully slow. And all of the stuff that we added would have just been laughable. I remember when we first got hatching working, and I did a demo of a complex shape for one of the venture capitalists - one of the probably hundred venture capitalists that visited us and decided not to invest. It took 45 minutes - this was not really impressive. "Er, gee - couldn't you actually draw it with a pencil faster than that?", people would remark!

But if the AT had come out, say, a year before we'd started, all of our competitors would have been on it, before we had anything to show, at all. And they were better, at that time. If it had come out later, we would have been out of business because we were trying to sell something that wasn't terribly practical. I mean it wasn't even a kind of "bleeding edge" early adopter, it was like a "bleeding stump" early adopter that you had to be to be an early adopter of AutoCAD on the original PC, at least without a nice graphics card and a bunch of stuff that ran the price way up. If you had the add-ons it was fine, but if you had a stock IBM PC, it was just bad.

If the AT had been late, we would have run out of money, probably. I don't know - maybe, maybe not. And that was something we didn't even anticipate, when we started. It just happened at the right time, for us. The AT, and then the explosion of the Compaq 386 was the thing: that changed the game again. I remember when we got one in from Compaq, just to verify that everything worked on it - it should, and it did, in fact - but it was, like, six times faster than anything I'd ever seen before.

That may have been what made applications feasible. Because of course, when you look at the origins of AutoLISP, we fell into exactly the same trap that every scripting language developer falls into. We said "well, it doesn't have to be fast, because it's pumping commands in! Executing the program is nothing compared with the time it's going to take for AutoCAD to do these commands." Of course not realizing that people are going to do things hundreds of times more complicated than you ever anticipated. And then it actually does become a bottleneck.

But again, suddenly, the fact that the machines were speeding up meant that we didn't have to address that problem as early as we probably would have had to, otherwise. And that was a dog-gone good thing, because until we got out of the MS-DOS 640K restriction, there wouldn't have been any way - it just wouldn't have fitted in the can.

Autodesk was very much a multi-platform software company at the time.

Until after I left. It was a multi-platform company when we showed AutoCAD - we had it running on three different machines with two different microprocessors: CP/M 80, Victor 9000, and the IBM PC. And, I might mention, in two different programming languages: the CP/M version was in PL/I and the x86 was in C. As a result of our glibly saying "oh, send us a machine and we'll get it to run", I think we were on something like sixteen or seventeen different machines by the end of 1983. In fact, there was a picture - at 150 Shoreline we'd rented this big, open space on the second floor, and that was our porting and QA lab. We actually once collected all the machines together, in a picture known as the Great Train Wreck - I don't know if there's a copy still in existence, but we put it out in one of the newsletters, and so forth, to show all the platforms it ran on. And by that time it was over twenty. Now, by that time they were all x86 microprocessor machines that had the same C code and different drivers.

The Sun was the first UNIX - and that was the 68000 architecture, at the time - and that was probably, I don't know, we probably started working on it in '86 and it wasn't done until '87 or '88, I'd have to go back and look at that. But that was our first leap. We actually had tried to do a UNIX port well before that, on a machine, a 3B2 from AT&T, or something like that, which was a little odd, as they didn't have graphics, at the time. But they were going to have graphics later, and we figured at least we can get over the hump of making the code run on UNIX, by then. And, as sometimes happens, it was given to this guy who never got anything finished and it just sat there for a long time. But it was really the Sun that made the difference.

Not long after Sun we had Apollo, which was another 68000-based machine, then I believe next we had MicroVAX. We never really sold very much on those platforms, but they had a tremendous influence on AutoCAD, in two ways: one, putting your code on a second machine finds hundreds of problems in the code, particularly when it's a machine with a different byte-order, different microprocessor, and all of those kind of things. Secondly, it gave us a preview of the future where we had unlimited - or what to us was unlimited - memory. At that point we may have been spending 50% of our time fighting with 640K problems. That was really just consuming a vast amount of our resources, and even though we knew we couldn't go away from the PC platform, we could see this was going to be fixed: people were talking about DOS Extenders, Microsoft was talking about a true 386 mode, OS/2 was coming on the horizon. The Microsoft-IBM alliance on OS/2 - that was going to solve our problems. And working on the UNIX machines showed us what that was going to be like. So we understood, both that things were going to get a lot easier from a technical standpoint, and the bar was going to be set a lot higher in terms of graphical user-interfaces, user expectations, and just the feature-level that people want on these platforms.

We did Silicon Graphics - that one came pretty late, in '91, I think - and, of course, Macintosh, which I did originally. We had this big project - there was always this Macintosh project of some kind - and they were trying to port it to the native Macintosh development tools. And, oh my golly, if you think 640K is bad, talk about 32K segments on the original Macintosh architecture! AutoCAD was just simply not written for that bizarre way that they set things up on the Macintosh. You basically have to re-write the whole program to do that. And so these Mac people were like "well, we'll just re-write the whole program! No problem!". "Erm, well, guys, you're talking about, we're up to maybe 10 man-years in this thing, now, and that's erm a bit of a job."

And so I suddenly realized, "wait a minute - I've got this Sun machine, here. It's got the same microprocessor as the Macintosh II. I can compile code on the Sun that will run in native 68020 mode, and it'll run on the Macintosh", I demonstrated that. So I had this thing called "the Titanic and the tugboat", where the Titanic was AutoCAD compiled on the Sun without any user-interface and the tugboat was this little thing on the Macintosh that had all the user-interface written in Mac code and just called the Titanic every time a command came in. And it worked! And we shipped it! It ran, and it was successful enough that we hired a bunch of Mac guys to clean it up. So what did they do? Well, this was awful. They started re-writing AutoCAD to fit it into the Macintosh development tools. So fortunately that didn't last too long.

Then it was only after Carol Bartz arrived that the winnowing of platforms started. I think we may even still have had the Macintosh version, then. Maybe, maybe not. But we definitely had Sun, HP Apollo, and SGI, still. MicroVAX had basically petered out, as DEC had abandoned the MicroVAX. Somebody was working on a NeXT - I don't think that ever got done. But the radical focusing on Windows/Windows NT was really something of the Bartz era. And let me say - that was something I was 100% vehemently for, at the epoch. If you read the remarks that I made at the introduction of the first AutoCAD for Windows, I was onboard with that.

You saw that as the future at the time.

I saw that as the future - one thing I didn't see was just how wretched it was going to be. I assumed that with the resources that Microsoft had, they could at least do as good a job as Sun

and Apollo and Silicon Graphics. Which, admittedly, wasn't perfect, but it wasn't this junk that we've endured now for two decades. I guess the other thing is, we were already on OS/2 at that point. We'd done the OS/2 port, and OS/2 was done right. That was done by people that knew what the heck they were doing. So we assumed "OK, for business reasons there's been a falling out between IBM and Microsoft, but Microsoft knows how to do it, because they've been working on OS/2 for the last three to four years." So how can they mess it up? It's going to be better than OS/2, and that's what Microsoft were saying, "it'll be better than OS/2". And besides, even if they mess it up, there'll still be OS/2. People are not going to use an inferior product. Well, I was wrong! People did use an inferior product. And they still do. And I don't think anybody anticipated that, at the time. Because you could say "yes, sure, it's junk, but look where it was two years ago", if you compared Windows 1.0 to Windows 3.1, well it had come a long way. And obviously they had the processor power, they had the memory coming, the microprocessors that can handle large address spaces, and OS/2 had already done all those things. So it's just astonishing, and one of the most depressing things we've seen in the industry, that we are where we are today.

Now, did Autodesk make the right decision, to focus on the NT platform? Absolutely! Where would your market-share be, if you'd gone to the Macintosh or Sun, or something like that? Did the workstation vendors blow it, by not concentrating on marketing AutoCAD and taking out Windows, when they could have? Yep, they sure did!

Scott McNealy of Sun *[Microsystems]*, always used to talk about the insurmountable opportunity, and the opportunity is only insurmountable if you don't want to try to do it. I sat in the Sun conference room and pitched to him and John Doerr. This was when they had the Sun 386, which didn't cost that much more to make than the PC. It was done by a rogue group in Boston, basically. This thing was an out-of-the-box solution that just runs AutoCAD fantastically better than any PC platform at the time. And that's the PC with the add-on graphics, with the add-on monitor, with the add-on memory extender, with the add-on DOS extender package loaded in, and all this junk you had to throw together just to get workstation - minimal workstation - capabilities. You unpacked the Sun workstation, you put in the AutoCAD install disk, you're in business - you're running AutoCAD! I said, "what you ought to do, you ought to get this thing in every one of our retail dealers. Because the dealers understand - this thing is cheaper than what they're selling now. It's not cheaper than the bare PC, but it's cheaper than the PCs they're selling with all the junk that they have to add-on to get this kind of capability." And McNealy's response was "well, we couldn't handle the support calls. We couldn't handle inexperienced users. We couldn't handle people who don't understand how to do UNIX system administration." To which I said, "Scott, get yourself an AutoCAD PC and look at how difficult it is to get that set up, and get everything working and configured and so forth. And secondly, I think the users are a lot smarter than you may think." But, of course, they had this dream of a next generation that was going to be really easy to use and then they could roll that out, but it wasn't ready yet. Well, you have to do it when you have to do it, sometimes. And if you have to hire a hundred people to read from support scripts, because there's nothing else you can do, well I guess we found out that works. It's not very nice, but it can be done. And I think they threw away a huge opportunity, there.

The other influence, though, to get back to your original question, the influence that Sun had on Autodesk, was that we moved our development from the PC platform to Sun. All the people that were doing the feature development of AutoCAD started to work on Suns. The quality of the tools was just so much higher that their productivity probably tripled. Now there was additional work, obviously - it's a risk: you develop on one machine, you target another, you're going to have problems - but the productivity gain from having developers working three times faster far more than compensated for the QA problems that we had, eventually making the PC version. Besides, it was just around that time that the native 640K version basically went away, and we now had essentially unlimited memory with the DOS extender package on the 386 machines.

All of these industry analysts - when I say the word "analyst", I'm thinking the word "idiot" - were saying that the 286 machines would be around for at least 10 years and that Microsoft and IBM are right in continuing to keep Windows on the 286 architecture. Well, all I can tell you is that in the AutoCAD market, they evaporated like the morning dew on Mercury. About a year after the 386 came out there wasn't anybody running AutoCAD on the 286, except somebody that had a legacy machine, but there certainly wasn't any new user - and there wasn't anybody interested in an upgrade of AutoCAD - that ran on the 286.

So that really let us get onto a platform that eliminated all the horrific memory problems. And then that made everything else possible, like the C-language ADS application system, and so forth.

[Tune in next week for** Part 2 - AutoCAD's architecture & APIs**.]

An interview with John Walker - Part 2

This is the second part of the series documenting my recent interview with John Walker. Where the first part looked at the early history of the company, this part focuses more on the architectural evolution of AutoCAD and its APIs. I have not linked to terms previously linked to in the first part of the series, which should also be referred to for typographical conventions, if they're not clear.

AutoCAD's architecture & APIs

We've talked a little about Variables and Expressions, already. What was behind the decision to use Lisp as a language?

Well, that's interesting. I would say, as much as anything, opportunity. We had this menu macro language, and that clearly had gone as far as it could go. We needed something that would provide access to the database. We needed something that could actually do algorithmic operations needed in writing the program. And it had to fit in 64K. It had to fit in a 64K segment, basically, and it had to fit - with that 64K segment - in a 640K IBM PC environment. Along with all the rest of AutoCAD, which at that point was hideously overlaid just to squeeze AutoCAD - alone - into that environment. Now a whole programming language had to go in there.

That really ruled out about 90% of the programming languages that you could think of doing. So the question was, do we roll our own little, interpretive programming language, think of it as like mini-Perl, or something like that, or do we use an existing language. We were all, from the standpoint of compatibility, all for an existing language. And here was this thing called XLISP that was written by a fellow named David Betz who was, at the time, an employee of Digital Equipment Corporation in Massachusetts. And it was this tiny little PC LISP interpreter. Pure interpreter. And because he was an employee of Digital Equipment, their policy at the time was that, if you did any work on your own, you could either give it to them or you could put it in the public domain. He put it in the public domain. So that meant we were free to use it without any royalties or encumbrance.

Now XLISP was actually a different dialect of Lisp: it was more towards the MACLISP dialect - the MIT Lisp - than Common Lisp. It was missing a lot of very important things including, for example, floating point numbers. I don't think it had any trig functions. So there were just huge gaps. But the core of LISP is four functions - you can theoretically, given those four or five functions, write the whole of the rest of it in Lisp, if you have to. That functionality was there, and a good part of the rest of what you expect. And it fit in 64K! So here was something, here was a complete programming language that by a simple matter of programming we could extend to do anything we want. The architecture of it was such that there wasn't any problem putting in floating point arithmetic, it was just a matter of setting it out and doing it. And when we were done, we could actually converge upon Common Lisp, which was not only a respectable programming language, it was - in the mid-80s - the hot thing. That's where all the AI money was going in. "AI is the wave of the future, Lisp is the language of AI!" There was a certain marketing sizzle to choosing Lisp, as well.

Now it's interesting - and we didn't even know this, at the time, that's how isolated we were from the UNIX community, because, again, we went directly from mainframes to PCs, we skipped UNIX, in there - but around that time Richard Stallman was looking for a programming language for

Emacs, and he ended up with Lisp also. And so Lisp ended up being the programming language of both the most widely used text editor on the UNIX environment and AutoCAD. And, I don't know, he wasn't faced with the memory restrictions we were - because he was working on VAXes, at the time - but we both came up with essentially the same answer, completely independent of one another.

Lisp is very much still in use today, of course.

Sure - if you want a little glue for something, absolutely. Even after we had ADS I did probably half to three-quarters of the little things I did with AutoCAD in AutoLISP. It's just easier - you don't have to compile a program.

And there's a swing back in that direction from companies such as Microsoft, who've got an explorative, interpretive interface to F#, for instance, and Functional Programming is definitely having an overall resurgence. Do you track programming language evolution?

From time to time. I've never really been one of these programming language either fanatics or groupies or trend-followers. I mean all of these languages were invented in the 1950s. There's nothing really new. It's just every decade there's another fad and something comes to the fore. And these things were invented in the 1950s because they were all good for various things, and none of them are good for everything - that was tried, that was PL/I. And let me tell you, that didn't work. And because of the universality of computation, anything you can do in one, you can do in all the other ones. Unless there's something deliberately omitted, or something like that.

So it's just a question of what's the best tool for the job. When someone talks to me about Lisp vs. functional programming, vs. Prolog, vs. declarative languages, to me that's just like asking "what's better - a hammer or a screwdriver?" Well, do you need to turn a screw or pound on something? I would say that other than fixing legacy code, I do about 95% of all my new software development in Perl. And Perl is about seventeen times slower than C. Who cares? You've got a 3GHz machine and the thing runs for a second! What's the difference whether it runs in a seventeenth of a second or a second? And I think that's one of the reasons that people are going back to the interpretive environments, because they are, in the most part, more friendly to the developer and certainly a lot easier to diagnose when something goes wrong in the field. And with the speed of machines we have and unlimited memory, the advantage that you had from native code just isn't there anymore.

Besides, who's running native code anymore? The x86 processor doesn't run the code you put in memory. It takes that x86 code and disassembles it into a very long instruction word architecture, which is the way the microprocessor really works. So even if you're writing in assembly language, that's not the code you're really running. So what's the difference between that and just-in-time byte-code compilation in a Java or C# implementation? That's just one more level.

I have a floating point benchmark that I update. Every time a new language comes out, I put it on there. And do you know what's the fastest language from my floating point benchmark, right now? Running on an x86 architecture, running a benchmark that does exactly the same thing and produces exactly the same answers. It does a ray-trace design of a refractor telescope. It does this 100,000 times, and you measure how long it takes. Well, it's Visual Basic .NET. Visual Basic, with its byte-code interpreter, is actually faster than C. And if C is 1, Java is 1.027. So, again, around 3% on a 3GHz machine - who's going to see that? Java has a problem with its startup time, because a typical Java program pulls in all kinds of stuff from the disk. And that's probably where it's losing the 3%, if I ran it longer then it might come close to approaching parity, but there are rules.

And none of these differences are big - the big differences are when you get to the Perls and Pythons and the dynamic languages. But even GNU Lisp is only seven times slower than C.

When Lisp was introduced, was the focus on user-level customization, professional development, or both?

I would say, really, both. I think what we had found was the developers - who prior to that, using menu macros, had done amazing things - had all started as users. They were simply users who found, "oh gee - look!" And, as I would describe it, they would always say "no, I'm not a programmer." What are they doing? They're programming - they just don't know it, because programming is this arcane thing that these people do at Microsoft, or something. But no - they're programming, and they just get sucked into the dark side, basically, by... first it's a menu macro, then it's a little variables and expressions, then it's AutoLISP... before you know it they're building big applications.

So there was that desire. We never even remotely considered making the development module something we would sell or license to official developers, or something like that. Everybody was going to have it, it was going to be in every product. Because we expected most of our developers were people who started out as users and built something interesting. We'd seen that hundreds of times. The whole UNIVAC community had a thriving, giveaway, third-party software market for things that UNIVAC mainframe users had developed for their own use, and said, "well, we're a government agency, we don't sell software, here it is."

And you might not know this, but DBase II, the Ashton-Tate database system, was a copy, a reimplementation of a package developed at the Jet Propulsion Laboratory called JPLDIS. In FORTRAN on UNIVAC mainframes. Everybody used JPLDIS. UNIVAC never expected Jet Propulsion Laboratory to develop a database system that everybody used. But suddenly you had people buying UNIVAC machines to run JPLDIS.

So there was that component, that we wanted to provide something to everybody, so our application developers would spontaneously appear, from users who found themselves creating applications. But also from the people - particularly Rick Janaczek and Dennis Neely - who were responsible for what was first called AE/CADD and later became AutoCAD AEC Architectural, which was entirely a tablet of menu macros, at the epoch, we had a set of specific requirements for what they needed to go beyond what they had. And so those requirements went into what we put in. Not really so much for Variables & Expressions but the full AutoLISP. The ability, particularly, to reach in and look at the entities in the database and get the attributes and things.

Were they part of Autodesk?

No. Dennis Neely was an architect, who continued to do architecture, Rick Janaczek was a consultant who Dennis Neely brought in when he discovered it had gone beyond what he knew how to do. They had their own business relationship. Eventually, we licensed their product through a royalty deal, and these were two of the first - well, they were not the first add-ons to AutoCAD, but they were the first ones that anybody bought. And it was originally called AE/CADD, and then they made one called Mechanical - which was not for mechanical engineering but for the mechanical aspects of architecture.

Ah yes, that's right - there were AutoCAD AEC Architectural and AutoCAD AEC Mechanical...

Which confused everybody. But they were giving this - Rick, being the programmer - was telling us, "this is what I need, this is what I can't do with these bloody macros that we've got here. I need a way to get values out of entities, I need a way to tell how long a line is." So that's where it became obvious that we needed to let the thing - at least, in read-only fashion - reach into the database.

So you had external input guiding the evolution of the API-set.

Exactly, as with the feature-set of AutoCAD we had the users providing guidance there. One thing we always did, and this persisted, really, - and you may still do it - but certainly until Carol arrived, we'd always have line developers go out and do demos at trade-shows. Because you'd come back from trade-shows and have forty, fifty feature requests that were things that you would never have understood why it was important until the guy that needed it stood there and showed you on a copy of AutoCAD what it was that they couldn't do. And I just can't imagine a software company not doing that, because it's $100M market research for free. And it costs you a week - or three or four days - of your developer's time to get it.

And it was really the same thing. Once, particularly, with the guidance - because Rick had done far more than anybody else, and he was also in California, in the San Francisco area, so he was always at our office, pounding on the table and saying, "I need this" and "that works", believe me - but once AutoLISP was out there it was, again, the very same thing. Suddenly we got a flood of "well, I can't do this" and "you need to give me access to that" and "how do I turn this off". And so I've always found... people criticize Microsoft for getting some shoddy thing out there and improve it, but the advantage of getting the shoddy thing out there is that you start to get the feedback, of what it really needs to do as opposed to what you thought it needed to do.

I spent a year and a half developing this web application *[The Hacker's Diet Online]* and I put it out there, and I had a beta for a couple of months, and opened it up on the 2nd of July last year. And from Day 1 it had this little web feedback form, "bug"/"request for feature", and I think I'm over my hundredth "request for feature" implemented in the application today. Almost all of them were "darn, why didn't I think of that?"

What were the drivers behind the introduction of ADS *[the AutoCAD Development System]*?

I would say as large a motivator as any other was the fact that application developers would not make the investment in making large applications if they had to give away their source code. And with AutoLISP, when you sold the application you gave away your source code. We had an obfuscator - the Kelvinator - but there were de-Kelvinators out there, and so forth. And we had really reached the point, both that the AutoLISP applications were getting so big that there was actually a problem, because you couldn't really segment an AutoCAD application all the easily, I mean you could have things chain-load and so forth, but it was just maintaining code that big. And particularly if you're not doing things the Lisp way. I mean some of the very largest programs ever written have been written in Lisp, but they aren't written procedurally the way C people write things. What we had was a bunch of developers who were thinking BASIC or C and writing it in Lisp, and after a while it gets pretty unwieldy for that kind of thing. Obviously with ADS you got a huge performance benefit - not as much as you might think, if you're doing a lot of computation, obviously, it was enormous, things that were designing things parametrically, for example, for that kind of stuff it was of enormous benefit - but still, the interface between ADS and AutoCAD was the AutoLISP interface. ADS was essentially the code that AutoLISP used to talk to AutoCAD with an API put on the other end. Everything, at least initially, went through the same pipe that AutoLISP went through. So the only place that you got a real speed-up was on the pure C side, for computation. When it was talking to AutoCAD it didn't run any faster than AutoLISP did.

That changed, as it evolved. The biggest motivation - and the biggest reason for its adoption - was the ability to make binary applications that could not be reverse-engineered.

Were you involved in Release 13, at all?

R12 was the last release that I had any involvement with.

So you weren't involved in the decision to rearchitect the product during that release?

I was railing at the executive level for the fact that that needed to be done, and hearing that - this would have been like '91-'92 - we wouldn't have a fully-integrated Windows API version that

looked like a Windows application until Release 15 - or what was called Release 15, at that point - which might be three or four years out. And I said, "that is death for the company". And they said, "well, it can't be done". And I said, "you have $160M in the bank - you can do a lot of things with $160M, if the future of the company is at stake." And I think eventually the message got home, that they did need to do a massive rearchitecture of the program. But that was really after management was changed and people who thought about what needs to be done, rather than what can be done, were in charge.

AutoCAD evolved with AutoLISP and ADS, and at some point we needed to be able to make modules that could be loaded directly inside AutoCAD that had the performance profile of DLLs, and also allowed us to extend AutoCAD without having direct source access, both internally and externally. That I always see as a significant decision. Although R13 was a difficult release, especially from a quality perspective, it needed to happen because AutoCAD would otherwise have died long ago.

It's always difficult... everybody makes this mistake, when building software, and they make it over and over and over again, serially for their whole careers. They start the program and they say "it's absolutely important that this application be completely open, that we expose everything that we do inside to the outside. That we create no barrier between our native code and the developer. That we give the developer every power to extend the program that we have. " And they go in saying that, and why do they go in saying that? Because they knew that the last program they worked on, they had to go and rearchitect the whole thing when they suddenly realized that they'd need it to be open.

And they start developing the thing, and it's open, and then, nobody's using that openness yet, and so it closes. And so you put this thing in, but you can't ship it in time, if this is all done outside or exposed. And eventually you get to that horrible realization that it's happened again, and that you have an architecture where there's a whole bunch of stuff going on inside there that you can't do from outside, and then you pay many times the cost that it would have been to do it right the first time and expose that.

I think if you'd sat down and talked with the people developing AutoCAD in 1982 and asked them, "how are you going to architect the program?" They would have all said, "it's got to be completely open." But, doggone it, first, you've got a 640K limit, second, you've got to get to market, and third, you've got users and dealers who are clamoring for features, they say "I could sell four times more", or "I would buy this if it had X", and you don't delay the product for three or four months to go and build it that way. And that's just the reality of software development. And I think that in the entire history of the software industry, there has not been a single, successful product that did not get used for things vastly larger and more complicated than its developers ever imagined it would be used for. And it's difficult to architect something for something that you can't imagine. That problem gets you, as well.

But again, I did a project in the late '80s, completely forgotten, except on my web-site, called ATLAST - we had to rename it from ATLAS because of a trademark problem. That was basically a Forth-based, of all things, API. So that you could build your program out of little chunks and glue them together, with the language. But the whole concept was - first of all, it ran as fast as native C code, so there was no hit at all in doing it this way as opposed to coding it in native C - but it meant that if you built your own application this way, you couldn't avoid exposing your entire functionality to developers. Because it was there, your own application used the same little bits with the glue code, as your developers had. From a strategic standpoint, if you didn't want to expose something, you could put a wall up, but it meant that technologically there was no barrier that separated you from the developers. Thank goodness we never did that, because Forth was not a memory-safe language.

The only thing that ATLAST ever got used for was a little project that I threw together in about a week, a week and a half, and it was just to get the goat of Jim Meadlock at Intergraph. I wanted to really irritate him. Intergraph had just launched this massive campaign - this was during the object-oriented fad - "Intergraph is an object-oriented CAD system." I can't imagine, really, how they justified it. I never really saw their argument. I think it was just like, I don't know, selling this 1950s Cadillac "with Nuclear Power!", or something like that. There wasn't anything object-oriented about it.

So I built this thing with ADS and ATLAST, that lets you have entities in AutoCAD that actually had methods. And you could send messages to them, and it had the big four of object-oriented: it had messages, it had inheritance, it had polymorphism... it was all there. And so I called this ClassWar, which stood for Class Language Application Support System Within AutoCAD, Really! I demoed this thing at one of our shows and said "this is an object-oriented CAD system!" And I never did get a reaction from Meadlock.

[Tune in next week for Part 3 - Autodesk's eco-system and strategy.]

An interview with John Walker - Part 3

This is the third part of the series documenting my recent interview with John Walker. Where the first part looked at the early history of the company and the second looked at the architectural evolution of AutoCAD and its APIs, this part focuses on Autodesk's business strategy and focus on its eco-system. I have not linked to terms previously linked to in the first part of the series, which should also be referred to for typographical conventions, if they're not clear.

Autodesk's eco-system and strategy

We've talked about AutoCAD's early architecture. How did this relate to Autodesk's business strategy?

From the outset the architecture of AutoCAD was intended to be as open as possible. We intended to leverage every partner that wanted to work with us, in any capacity whatsoever, to use their resources. This was something that we had a lot of trouble explaining to the financial community and to the traditional CAD analysts. Because they would say, "how can you possibly compete with a company like Intergraph? They have a training staff that's larger than your whole company. They have a field support staff that's larger than your whole company. etc. etc." And I said, "because I believe in the market. Because I believe that if you create the market, our dealers will become our field support staff, universities will become our training department." And, indeed, all of those things not only happened, they had already happened by the time I was trying to explain this to them, those things were already underway. You will have every community college teaching this product, and they're not going to teach the Intergraph product because they don't have $150,000 for an Intergraph workstation to teach it. But they've already got the PC: all they need to do is to put this disk in it. And in terms of documentation and training, we've got authors already writing books about this product. For themselves - we're not paying them. I believe in the market solving things. A lot of people laugh at that, but hey, it worked for us.

You mention the importance of the market and having a healthy ecosystem. How important has it been to Autodesk's success?

Absolutely central. I don't think the company or the product would have been viable, if there weren't that community of dealers and developers, of educators and writers of books. One huge contributor to AutoCAD's success was the book *Inside AutoCAD*. I think I have a first edition, somewhere. Because that was what you would now call the missing manual. The AutoCAD manual told how it worked, this told how to do things with it. And that book created a whole publishing company, basically.

So we really had... the number of people that were contributing to the success of AutoCAD and Autodesk was much greater than the number of Autodesk employees. And, even better, we didn't have to pay them. That was really the strategy. As I said, we were explaining that to venture capitalists in '83, when they were visiting us. In exactly the form I've presented it here, except it was more in the future then than it is now. They simply didn't get it. These are guys who, their job title includes capitalist, but they didn't understand capitalism! I mean, the whole idea is that you create a market, and people will pop up spontaneously to fill the needs of the market, because they want to get rich doing so. And that's exactly what happened.

When we got back from COMDEX, as I said, we were deluged by dealers calling... "how can I sell this thing?" Why were they interested in selling it? Because it cost $1,000! That's why they were interested in selling it. A heck of a lot better than selling a WordStar for $300. And what's more, they sold a big graphics monitor, and a card and a digitizer and a plotter - that's a $10,000 sale. As opposed to $2,500. That's why they were interested in it, not because of the technological capability.

And what did we tell them? "You want to be a dealer? You get 40% discount. You have to qualify to be a dealer. To qualify to be a dealer, buy more than one." That was our policy. We had a couple of hundred dealers, before too long, as what's more we had a lot of money coming in, because they had to buy two copies at $600 a piece. So that was $1,200 for every dealer we signed up.

You were also surprised by the quality of some of the dealers, who wouldn't have been eligible if you'd put in place rigorous certification from the beginning.

That's right. We had dealers who would have never been considered for any kind of certification. Some of our most successful dealers were practicing architects who bought one, and suddenly people would visit their shop and see it, and "well, I can set one of these up for you." And before long, their architecture business was gone and they had this thriving business selling and servicing AutoCAD. I think we had a community college professor who found himself in the same situation. He started teaching it, and suddenly his students were out needing to buy systems and they were starting to work for people. And he developed a business selling these systems.

One other contributor to Autodesk that's frequently forgotten, is that, among our founders, we did have four Europeans. So we were operating in Sweden, Switzerland and the UK from day one. And in fact some of our very first sales were there. And through much of Autodesk's history, through the '80s, about 40% of our business was in Europe. And 40% is your profit margin - if we didn't have that business, we would have been a marginally-profitable company.

A lot of US companies took a long time to get there. And we really knew from the start - because my little, piddling, ridiculous hardware company got about 40% of its business from Europe, and that was about my profit margin. So we knew that we needed the localized versions, that we needed translated manuals, that we needed to have a presence on the ground in order to address that market.

We were much slower out of the block with Asia. Just getting the first localized Japanese version was a real challenge. That was, of course, the "go, go, Japan #1" era, lifetime employment, you couldn't hire a programmer in Japan, you couldn't get them. An American company that's only three years old? Uh-uh. We ended up with a whole staff in California of Japanese Americans. So we'd get these machines from Japan. And of course these Japanese manufacturers were just tripping over themselves to get AutoCAD on their machines. But, of course, they came in and the machine speaks Japanese and all the manuals are in Japanese. And at that point the DOS for Japan - this was all pre-Windows - was a pretty weird creature. There was a lot of stuff that was different with it, with Kanji keyboards and data-entry and so forth. So that was a lot of work getting there. But once we finally got the work done it was a great success.

Although the Japanese market was - there really wasn't the Japanese equivalent of the IBM PC, except for the IBM PC - every big Japanese company had its own PC, which was nothing like anybody else's, and if you wanted to sell to Hitachi, you had to put it on Hitachi's machine, and same for Sony, same for Toshiba, etc. And, again, so you had this platform explosion, each to address essentially one customer.

But we had, really... we spent probably the better part of three or four years flapping around in Japan, listening to these analysts who told you, "the Japanese economy is entirely top-down, and you have to get into the big, integrated companies, and sell to the top, and they make a company-wide decision when they choose the product. And that's what you have to do." So we pursued

them, we were doing all these ports, we were doing everything, and then we hired a new manager for Japan who was a Japanese national but he'd lived about half of his life in the US, and he basically said "you know that's all bull----, your market in Japan is going to be small business, just like it is in the US. There is a huge sector there. Because the huge companies, they already have a CAD lab with workstations, and things like that in it. They can afford the very highest-end CAD systems. You may get some AutoCADs out in their field offices, and so forth, and for things that you're not doing on the workstation, and we can get there eventually, but we need dealers, we need local support people, we need exactly the model you've got in the US." And so we said "go ahead." And up it went.

AutoCAD used to have a hardware lock outside the US...

In the early '90s, we had no locks in the US, and locks everywhere outside the US, and that created quite a bit of tension. I would always ask the regional managers, for Europe and Asia, "can you estimate for me the difference in revenue if we did not have the hardware lock on AutoCAD in your market. Universally the answer would be 'none'. Because there were so many cracks going around that anyone that wanted it could get it to run without the hardware lock. And there was simply no person that was willing to use it illegally who was unable to, that was the fundamental reality. And I said "well, why don't we get rid of it? Because it'll eliminate this tension that we have here." Particularly with copies flowing out of the US, with unlocked copies to Europe, which the European dealers were just enraged about. They wanted us to lock it in the US, and consider the Indian market, which speaks English. I mean, good grief. We had zero penetration there, everybody was running illegal copies from the US.

It would also, at that point, knock $30-40 off our manufacturing costs for every copy with that thing that we were buying, and so our profits would go up. And they said that you could never do it. And I said "why?" And they said "the dealers - the dealers would walk away from us in disgust." The dealers would perceive it as we're abandoning them to the pirates. Even though actually it would probably have had no impact on their sales.

And this wasn't just idle speculation: because at this point, most of the other major applications were not locked, and they were selling fine in Europe. So it wasn't like Microsoft wasn't selling any Microsoft Office in Europe.

The hardware lock was introduced briefly in the US at one point, I believe.

In 2.5 it was introduced around Easter, and pulled around Thanksgiving.

And it was then nearly put back in?

About three years later they were going to try to put it back in, and that's when I went to war. And the amazing thing, the astonishing thing about it is that, almost without exception, the people who were going to put it back in were the same people who had lived through having it in and taking it out the last time! It really is amazing how the human mind can flush unpleasant experiences, so you're set up to have another one.

Now it's certainly true: I designed the original hardware lock - I'm guilty as charged - and I can show you it, the prototype downstairs [*John did, indeed, show me the first, prototype dongle for AutoCAD at the end of the interview*]. And it wasn't very good, actually, it had a lot of problems. It created compatibility problems with a number of plotters, for example. And it stumbled into a bug in the serial port chip that a lot of PCs had. It was a fundamental bug in that chip, and the hardware lock found it. So it wasn't just that people didn't like the hardware lock, it's that they put the hardware lock in and - you've got a PC, you've got one serial port, you put your hardware lock on and you put your plotter on the other end - well, the plotter would plot for an hour and then go "BRRREEEEEEE" and start drawing to the moon, or something. Quite understandably people were irritated by that. And they perceived it was the hardware lock that caused the problem.

Actually it wasn't the hardware lock that caused the problem, it was this serial chip but the adaptation of our serial port driver code to accommodate the hardware lock stumbled into this bug in the National Semiconductor chip. So true - if we'd not had the hardware lock,we wouldn't have changed the code, we would have continued to skate past the hole in the ice. So you can say the hardware lock caused it but in that case it wasn't actually the hardware lock. But it was just a marketing disaster to us.

And again, part of my reason in saying it wouldn't affect our sales is when we put it on in the US our sales didn't go up, and when we took it off in the US our sales didn't go down. So it really didn't have any impact.

What tenets most contributed to Autodesk's early success? From your perspective - with the understanding you haven't had much contact with the company in recent years - do you believe them to continue today?

When I attribute so much of the initial success of the company to luck, it's hard to say that's something that has continued in the long-term. In a way it has, because Autodesk - like every other established PC software company - has benefited from the fact that Moore's Law has continued, all this time. And so you've never really reached a ceiling in the platform, in terms of the size of the applications you can develop or the power that you need to run them. And that's a piece of luck that has come along, and it may not continue forever.

When I say we were lucky, I mean that if we hadn't been, at that time - at that unique time in the market - with the unique product that we stumbled upon as much as anything... it started out as an application that ran on my Marinchip machine. If I hadn't had the Marinchip machine, the application would have never been written or it would have been written for something else. I'd have never known of it. Never thought of it, basically. Do you call that luck? Yeah, I call that luck.

And from that point, I think it was really just a question of doing my favorite three words, "whatever it takes", and being driven by the market. Listening to what the dealers were saying, listening to what the users were saying, listening to the feedback we got from the big CAD analysts, when they finally started to notice us. I think there were a lot of times when it was tempting to go off on some kind of technological crusade, to try to do something really ambitious and neglect a whole bunch of really boring stuff that just needed to be slogged out, that people were asking for.

I remember, this was probably '87-'88, in that epoch, we started a thing where we would invite the CAD managers of big companies to come and have a day at Autodesk and see all of our developing technologies. And they would fly down at their own expense, and they thought they were really getting something. What we were doing was picking their brains, obviously, for this whole time, because - again - free market research with some very eminent and distinguished people. And I remember sitting at a round table, and having the CAD manager of Boeing - now, to me it was an honor to be in the room with the CAD manager of Boeing, right, not to mention three or four other guys like that - and he said, "well, you've asked me what do I think you should do. And I'll answer, I'll tell you what not to do. Don't try to build CATIA: because we've already got CATIA, and it doesn't do enough for us, and it takes us three years to train an operator to do what it does now, and we've got people developing on it and you can spend your life there and you'll never catch up, because they're moving faster than you move. What you have to do is to do what suits your users, in your sector. And your users are not CATIA users. They're completely different communities." And I really took that to heart, because we were thinking, "big CAD, Numerical Control, Finite Element Analysis integration".

And if I hadn't heard that from his mouth, we might have done some really stupid things. That, again, wouldn't have wrecked AutoCAD, but they were just missed opportunities: with opportunity cost you're spending your resources in places that you're never really going to be successful.

Are you at all surprised about AutoCAD's continued success?

Yes and no. I think by the late '80s, I had really just one worry, two worries, maybe: Autodesk is going to do something really stupid and screw it up - either a disastrous update release or something that destroys the developer community or wrecks the dealer channel or something like that - and secondly, Microsoft. I thought Microsoft was going to come after us. Just look at their history: every major application sector, they've launched a product against it. And they did eventually: Visio. And they blew it. I actually - in March of '93 - I met Gates and I asked him, "why didn't you come after us?" And he said, "at the point that we really could have, I didn't think it would ever get to be a half-billion dollar business. I never believed that it could be as big as Office." And, actually, at that point AutoCAD was not as big as Office, but it was as big as Excel or Word, taken independently. And that was big enough to interest him. But he just never thought that it would be there. He thought - I guess I think he probably thought that...

Well, AutoCAD, to my knowledge, did something in the software industry that has never, ever happened with any other product. It sold more than a million copies at more than $1,000 a copy. And that never happened. You've got expensive software but a tiny market, cheap-large market. That's what it really took to hit that revenue figure, and I think that Gates probably having seen the prices of his own products erode, over time, as the market became larger, assumed that by the time we got to the millionth unit, we'd be down in the $200-500 price range. And that not only didn't happen, but the price went up, over time. And that created that unique profile. And that profile, I think, cost Autodesk a great deal in the '80s and '90s, because it created an impossible standard. Whenever Autodesk was looking at doing something new, it had to be another AutoCAD. Another Microsoft Word wasn't good enough, another DBase II wasn't good enough. And there aren't any - nobody's ever found another AutoCAD.

It was an absolutely unique circumstance, at the time. And I think - as much as anything - created by the fact that it took such an expensive hardware configuration to run it on, that people didn't look so much to the cost of the software. They weren't just looking for a box to run on their existing PC. And that did create problems, but really, those two things not having happened, it doesn't surprise me because the fact is, when you get to an 80-90% market share, other than messing up or having a competitor with unlimited funds come after you, you tend to stay there. A Wall Street analyst whose name I forget, who was one of the first... he basically made his reputation on having predicted that Lotus was going to be a huge success before anybody else. And he said - and this was like in the '80s, when people were talking about the big CAD companies coming after us, and they were all readying their PC packages - and he said, "they're all going to fail, you're going to succeed, and you're going to succeed because they all under-estimate the power of the franchise and of an installed base. That you have. They don't have community colleges teaching their products, they don't have shops with ten stations that are buying three or four more. They'll do fine cannibalizing their high-end systems, selling low-end systems to their existing customers, but they're not going to hit you." And I've always kept that in mind, and I think that when you hear about... the risks that you run are the ones that are a completely different thing. That the world has changed and you haven't. It's Windows is here and you're five years from being on Windows. It's Google launches Google CAD, and you have nothing to respond to it. Those are the things that are the really big risks, now.

[Tune in next week for Part 4 - Past and future opportunities.]

An interview with John Walker - Part 4

This is the fourth and final part of the series documenting my recent interview with John Walker. Where the first three parts looked at the early history of the company, the architectural evolution of AutoCAD and its APIs and Autodesk's strategy and eco-system, this part takes a look at past and future business opportunities for the company. I have not linked to terms previously linked to in the first part of the series, which should also be referred to for typographical conventions, if they're not clear.

Past and future opportunities

What technology trends do you monitor in the industry?

Well, I don't even really try to do that, since I'm almost always consistently wrong. I just kind of wait for the technology to roll over me like a wave, and then try to get up and surf on it. I think the next big trend is the one that you haven't seen yet. Because if you've seen it then it's the current big trend.

I've always taken this view that was so discredited, the George Gilder view, that was some completely discredited during the dot-bomb: that there really isn't any technological limit to how much bandwidth we can deliver. And how much we can connect. There isn't any technological limit - at all - to how much computing power you can put on the web, as long as you're willing to have it broken up into little pieces that talk to one another. That's not visionary - that's here. Because Amazon and Google both have platforms that do that. And on the 10th of September, the LHC is going to start taking data and sending it out to the grid. And doing their computing - of their unbelievable data-stream, terabytes coming out of that thing - storing and distributing it all over the world and doing their processing wherever there's computing power available to do it.

So that's here, now. What are the applications for that? Well, the Large Hadron Collider is an application, but that's kind of a one-off - you're not going to have the PC collider, or something like that. But what are the applications that you can't do today, but that you can do with, say, 300,000 machines of super-computer capability? Well, I can name one: spam! That's how it's being done today, it's being done by the botnets, running on corrupted Windows machines. So there's one. But what's one that - well, that's one you could make money off of, actually - but what's another one that you can make money from that's legal and ethical, and so forth? Particularly for the Autodesk niche, what can you do - I would say probably not so much on the CAD side, anymore, because you've got more compute power than you need for CAD, right now - but what can you do in the video- and movie- production environment, with a million times more computing power? Well, there are probably a lot of things you can do, suddenly. If you could imagine rendering Jurassic Park 3 in real-time, that's interesting.

Have you monitored the products Autodesk has developed or acquired over time?

I've never even seen them. I know them only from what's written up about them in the annual reports, which is basically the only communication I have with Autodesk, anymore. I think they've - certainly from a business and financial stand-point - done superbly. It's something that Autodesk absolutely needed to do. At the point that I basically got disgusted and walked away I was saying "you guys, you've got too much money, you need to do acquisitions. Because you need to do acquisitions in things that have a growth in front of them. Because when you've got 95% of the

market, where's your growth? You're not going to grow any faster than the general economy." And they were originally very tepid in doing these little acquisitions. I think Discreet was the first big acquisition, and that worked beautifully. I mean I didn't see the messiness from the inside, I just looked at the financials, but it worked beautifully from the financial standpoint.

I think they've done very well with that. That's an industry that was - very much like AutoCAD - poised to profit from every increase in compute and graphics capability. And that was something that was - again - at the right time.

But again, it's a lot harder than it used to be. When we were sitting there in 1982, how difficult was it to think of products to put on the PC? Well, about as hard as looking at the products that we'd already worked on on mainframes. And making a forecast as to when that capability would be present on the PC. That's all you had to do. There were dozens of product categories that ran on mainframes that you could basically port to a PC. And plotting out Moore's Law you could be confident that by the time you were done, you're going to have the compute power that you needed to run that. Well, if you were wrong, that was a problem, but we never were wrong! In fact, most estimates were actually less than we got in the '80s and '90s.

And today, you can't do that anymore. Because the cutting edge is on the PC platform.

You're plotting out what can be taken from the PC and put in the cloud.

Maybe. Or what can you do with the cloud that you cannot do on the PC. But you've got to think of that. You don't have a place that you can look and say "well, people are doing this with supercomputers". No, because the cloud is far more powerful than a supercomputer, and it's a different kind of beast. And so that's something that's much more of a challenge. I have an almost unlimited optimism in technological progress but I have an even greater optimism in the ability of creative humans to think up things to use all of that technological progress to do. Those applications are going to come, but I have no idea what they're going to be.

Who would have imagined Twitter? I mean, come on!

I'm curious - do you use Twitter?

No.

Do you have an interest in those technologies?

I'm interested in anti-social networking. I'm interested in protecting private data and one's own history in this environment of unprecedented disclosure.

I remember you developed an early, encrypted VoIP tool.

I spent half of the 1990s doing that. That was Speak Freely. And that was launched in 1996 and I finally drove a stake into it on January 15th, 2004. It's the first one that really worked, basically. And it not only worked, it was cross-platform from Linux and Windows. It was a bit of a reach. It would run on a 50MHz 486 with a 14.4K modem Internet connection. That wasn't easy. And it ran on Windows 3.1, too. By, say '98-'99, at the point the Internet - the consumer Internet - was exploding, at that point the infrastructure was such that it just worked. There just wasn't any problem with it and people weren't even using the most restrictive compression modes or any of the redundancy and recovery and so forth. People just routinely used it. There was a software company in Berkeley, California that had a bunch of programmers working in Moscow, and they would have their weekly teleconferences on Speak Freely, it was just a routine thing and it just worked.

It got killed by NAT. When the great broadband pile-on happened, suddenly people who had unrestricted inbound-outbound connections were behind this NAT-box, and they could just not initiate connections to the outside. So that meant that the only way that you could make it work

was with a central server, and nobody had the bandwidth to - I mean there were a million downloads of Speak Freely - and nobody had the bandwidth to set up a server to handle that kind of traffic.

What do you consider the most promising Autodesk products - that were under development or made it to market - that you believe should have been successful, but were not? I'm thinking along the lines of products such as HyperChem or projects such as Xanadu, which I remember reading about during my Computer Science studies.

Well, I'll give you the number one, and that's probably one you didn't expect to hear, and that's AutoSketch. Which we developed in '85-'86. Autodesk spent a year and a half on that product - I don't know how much money, particularly when you count QA people working on the product, and so forth, and spent absolutely nothing to promote that product in the United States. I mean zero. After it was shipped it was maybe six years until the next release came out, there was no-one doing any development work on it. And how was the product received in the market? Well, in Europe it got about a 90% market-share in about 18 months, because the people in Europe did promote it. So I don't really believe that there's a cultural difference that meant it wouldn't have had the same results in the US. But the Autodesk dealers didn't want to sell it, because they thought it would cannibalize sales of AutoCAD. The marketing organization didn't want to put it through the mass-market channel, because they were afraid the dealers would show up with pitch-forks and torches, if they thought we were shifting to the mass distributors. And it was only $79.95, I mean we only got twenty bucks, or something, from it.

So they ignored it. And they consequently had to spend $10M buying Generic CADD, which should never even have existed. Which they then ran into the ground and obliterated, through - may I use the word - incompetence. Incompetence and lack of interest. I mean, Pete O'Dell went up to Generic CADD, and he was a fire-brand, I mean he was doing everything in the world: the development, porting things, application developers, developing a new distributor channel, he was getting it into like Home Depot, with kiosks, and so forth. Coming up with products for individuals who wanted to redesign their kitchen or bathroom. And the Autodesk organization just strangled the resources, didn't give him anything he needed, even though he was generating profit and contributing profit to the bottom-line. And just ran it into the ground. And so twice - we had two products, well a product and a whole product family - in the low-end sector and it was just a classic "California blinders", so we had to do it a third time with LT! Having blown the opportunity, and particularly blown the opportunity to own 90% of the market from the outset, the way we did with AutoCAD. There should have been at least four releases of AutoSketch in those six years.

And the thing about AutoSketch - and I was the principal developer of AutoSketch, so I have both some knowledge and emotion in this - was that it was deliberately designed not to cannibalize AutoCAD. It was architected not to cannibalize AutoCAD. You could add all kinds of things to AutoSketch, but because its interface - AutoCAD was basically a command-line interface, this was a pull-down menu/dialog box, albeit MVC text-based - and AutoCAD was inherently programmable, AutoSketch was inherently interactive. So it just wasn't going to happen, you weren't going to see people abandoning AutoCAD for AutoSketch, whatever you put in AutoSketch, because it's just a different beast.

And that was just thrown away. And there wasn't any button that I, or anybody else who cared about it, could push that would change that marketing and sales organization, which was generating so much money just sitting there and writing orders from dealers selling AutoCAD, that it didn't want to do anything else that wasn't as profitable as that. And it didn't want to do anything at all that could possibly imperil that. And that was just lost. And that, I think, was probably - at least during the epoch I was there, which is all I can talk about - the biggest strategic failure of Autodesk in its core market.

Now the things that we did in the '80s, those were all opportunities that really had nothing to do with AutoCAD, in a large part. Those were attempts to get in the next rising big market, so that we could ride that one up and not simply grow as fast as the general economy.

Well, I think Xanadu... it's really very simple: if you had visited the Autodesk lab in the summer of 1991, and asked for a demo of Tapestry and Xanadu, but you had in your head what you have now, you'd say, "oh, that's a web browser!" So we basically had the web - Xanadu - and the browser - Tapestry - running as a prototype in 1991-2. And killed it in 1993. Nobody saw the Internet exploding the way that it did. And the fact is, of course, that the Internet - mass-market Internet - is a prerequisite to having that technology work.

Now I've always taken the approach that I think Bill Gates takes, which is: if I see something that I just know is eventually going to be an unboundedly large market, I'm willing to lose a modest amount of money for however long it takes for that thing to arrive. Because when it arrives, I want to be the one there with the product. And I think you could see that in the '70s and '80s with the CD-ROM. That he organized the first CD-ROM conference, he made all of his products available on CD-ROM, he put CD-ROM drivers in Windows. Nobody had CD-ROM drives! It took years before that really happened. But when it happened, he was ready. Now that wasn't a huge money-maker for him, but it was in the sense that it enabled him to deliver products that were vastly larger than 320K floppies, or something like that. So that was strategically important to him. I counseled Carol, "you can afford this. This is going to happen. And you want to be in a position when it happens to be the person that everybody comes to for the technology." And that was not the decision that she made.

Now there were a lot of problems with those projects, and there were a lot of problems with the people on those projects. And, whether, in fact, if you had gone on funding them at the level we funded them, they would have ever have gotten anything done, that would even have been compatible with the first release of Mosaic, is not clear. Because remember, really, Berners-Lee was already working on the web in '92 and '93, so it was going to happen pretty soon, anyway, and at the point that the web appeared, the growth was so instantaneous that Xanadu would have just been forgotten instantly.

Although, of course, you could say that it was a lot better than the web, you don't have any broken links in Xanadu, for one thing. But whether it would have really... because it was, in some ways, more centralized. Whether it would have been able to fit the model by which the web grew is not clear.

But then there was also AMIX, the AMerican Information eXchange, which was killed the same day as Xanadu. If you'd seen a demo of that, you'd have said, "oh, eBay!".

So there's a couple.

HyperChem was more a problem with the very crusty developers that were responsible for the package. They just didn't fit with Autodesk, and the way that we'd licensed the product was such that they could essentially pull the plug on us. And they essentially did: we were not going to get any new releases and development was not going to go in the direction that we thought it was going to. And we told them, "if you go this way you're going to have a very specialized, very high-price product. It's never going to have the mass impact." But they were not willing to lower themselves to the commercial hucksterism that we used to promote AutoCAD to get there.

And that's fine: it's their product, it's their work, it's their company. I think they missed an opportunity by doing that. Would it have ever been a great, mass money-making product? No. I mean, it would have been a very prestigious product, and it would have established Autodesk... it would have been a substantial amount of revenue, I think, if it had been done right. It would have established Autodesk in a completely different sector, which was when all the analysts were

saying, "you're a one-product company, you're a one-industry company!" OK, well architects, and mechanical engineers and... chemists. Now is that different enough for you? Thanks, right.

And it was also a superbly Windows-integrated program, and they were just on top of getting something like that to work in Windows 3.1. So I think we would have been able to suck a great deal of Windows expertise that would have helped us with AutoCAD. And that wasn't a reason to do it, but it was another benefit that you might get.

Hewlett-Packard doesn't make caesium atomic clocks primarily because they make so much money selling like eight of them a century to the Bureau of Standards. They make them because knowing the company that makes the most accurate clocks in the world also makes your PC is a benefit on the marketing side. I think perhaps HyperChem would have helped in that way, too.

Other than for the prestige, do you see it at odds for a volume software company such as Autodesk to invest in - whether buying or developing - low-volume, niche technology?

I think the first thing to do is look for the niche that becomes the next volume market. That's where - again - you play the game with Moore's Law. The thing you want is the niche market that's constrained by the fact that the hardware that it takes is too expensive. Because wait a few years and the hardware it takes won't be expensive. Now that doesn't say that every niche is a potential mass market. But remember, that there was a time, in the 1960s, when there were maybe eight computers in the world that could play video computer games. And that's a substantially bigger market, now.

So they do exist. And I think a lot of people would have said: professional-quality video-editing - back in the days of Avid - or the Avid hardware... workstation - that's never going to be a mass market, because even if it were free, the skills that it would take to do that are so specialized that it'll never be a mass market. Well, wrong!

There is a cultural and commercial problem, though. Because when you acquire a company you also acquire their sales force. That's used to writing $100,000 orders, and suddenly you're talking about $1,000 orders and they're not interested, thank you.

Well that's a very famous quote from Jim Meadlock, when he was speaking... it's the only time I actually met him in person, we were both at - I think it was - the Silverado conference in 1986 and I believe I had given my presentation about how were going to simply take over the whole CAD business, because within a year, every PC was going to have what you guys now call workstation capability. Who's going to need the workstations when we have that? The workstation people can go higher, than what they've got now, but we can do that on the PC. And Meadlock got up and did his presentation, and I think that's when he started flogging the object-oriented stuff, but one of the Wall Street CAD analysts got up and said, "Jim, what do you think about this AutoCAD phenomenon? The guy before said he's going to take your whole market away." And Meadlock said, "when I figure out how to make money selling CAD for $1,500, I'll do it." And I wanted to get up and say, "well, ask me! We're doing it!" :-)

I loved that speech from the Silverado conference, where you got up and started the speech by pointing at a photo of a 1960s computer and said "remember when computers looked like computers?". And then you finished the speech with the comment that in twenty years' time, someone would point at a picture of an IBM AT and say "remember when computers looked like computers?" That, for me, painted a very clear image of technological progress. Did you end up getting the chance to say that again in 2006?

In a way, though, has there been that much progress? I mean for a lot of that stuff, people really do, it looks a lot better but it doesn't even seem much faster to me. An AutoCAD, sure, but word-processing, the stuff that you did back then in '82...

Then that's not the right device - maybe you should point to, say, "remember when mobile phones looked like mobile phones?"

Right! Like bricks with huge battery packs.

Maybe the real innovation has really shifted somewhere else.

Nobody saw the Internet 18 months before it happened. It's like the classic example. You can read all the science fiction from the 1920s through the 1960s! The 1960s! And they had asteroid mining, and they had faster than light travel, and they had terraforming Mars, and nobody saw the PC. And in the same way you can read through almost all the science fiction through the 1980s, and nobody saw the Internet. A couple did, John Brunner, and obviously Ted Nelson, if you call his stuff science fiction and I think we can now can call it science fiction.

But I don't think anyone imagined that it would happen, and yet the funny thing was, the people that built the Internet in the 1970s, they all knew it was going to happen. They were believers. They didn't necessarily know when but they knew what. And thank goodness, because if that hadn't made some of the design decisions they made, back in those days, we would have just had horrific problems getting to where we are now.

[Portions of this interview have been elided both at the discretion of the interviewee and of the interviewer.]

[So ends my transcription of the interview. I'd once again like to extend my sincere thanks to John for sharing so many valuable anecdotes and insights. While he may attribute much of Autodesk's early success to luck, I think it's clear that John's leadership was just as significant a factor.]