Frame Technology and FrameMaker

David J. Murray

***Abstract*— Frame Technology became a leading provider of technical publishing software within two years of its founding in 1986. Frame went public in February 1992, and by the end of that year, its flagship software, FrameMaker, ran on Apple Macintosh, Microsoft Windows, and all of the leading Unix workstations of that era. It was used by thousands of high tech companies around the globe. The company was acquired by Adobe Systems in 1995 at a valuation of approximately $500 million. This article is a personal remembrance by the founder who cowrote FrameMaker 1.0 and lead Frame's product design and product strategy through its IPO. It tells Frame's founding story and describes the key product and architectural features that contributed to its success. The article also briefly describes Frame's early funding model, business and sales strategies, and culture.**

***Index Terms*— desktop publishing, office automation, text processing, Frame Technology, FrameMaker, personal computing, publishing technologies, technical publishing software**

I. Introduction

Frame Technology was founded in April 1986, by Charles Corfield, Steven Kirsch, Vickie Blackslee, and myself. The four of us had no significant experience in the publishing technology industry, and we started with no outside capital. Within two years, Frame had become a dominant player in the technical publishing world. By the time we went public in February 1992 at a valuation of $146 million, our prospectus stated that we had 100,000 users at thousands of companies around the globe, 297 employees with offices on four continents, and prior calendar year revenues of more than $41 million [1]. Our software ran on Unix platforms, Apple Macintosh, and later in 1992, Microsoft Windows. In 1995 Frame was acquired by Adobe Systems at a valuation of approximately $500 million [2], and Adobe continues to develop and sell FrameMaker to this day.

II. Introducing the Founders

Frame's founders brought complementary skills and backgrounds to the company, which served it well in the first few years. This section introduces each of us, emphasizing the aspects of our backgrounds germane to the early success of the company.

A. Charles Corfield

The story of Frame's history must begin by acknowledging the seminal contribution of founder Charles "Nick" Corfield, who is truly the father of Frame's flagship product, FrameMaker. While Frame's success as a company was achieved by a diverse and remarkable team, it would not have been possible without work that Charles did prior to the founding of the company when he wrote a prototype program called /etc/publisher, which served as the foundation for the development of FrameMaker.

Charles was raised in England and got his undergraduate degree in mathematics and physics at St. John's College at the University of Cambridge in 1982. He then came to Columbia University to pursue a doctorate in astrophysics. By all accounts, Charles was a brilliant mathematician, having won top honors at Cambridge, which begs the question of why he abandoned his studies to write a software program. I have heard several reasons given over the years by others that I believe are fiction. When we first met, Charles told me he realized while at Columbia that a career as an astrophysicist would not provide him with the means to pursue his many passions, so he abandoned his studies and wrote /etc/publisher with the goal of "getting rich enough to pursue my dreams, and starting a software company seems like a plausible path to that goal."

According to Charles, his search for a compelling application originally drew him to hypertext (years before the web and HTML), but he recognized that a prerequisite to making hypertext useful would be the ability to create interesting, rich-content documents that could be linked together. So he settled on first writing an editing and publishing tool that excelled at integrating text and graphics, with the intent to add hypertext linking later.

At that time, Sun Microsystems was a leading vendor of Unix workstations, and they had a program, run by John Gage, under which academics could get a loaner in order to write a program that Gage's office thought had merit. According to Charles, he applied under that Sun Catalyst program and was amazed to find a Sun 2 development system delivered to his New York apartment shortly thereafter. In a fortuitous oversite, the hard drive still contained the work of its prior user, including the source code for the new SunView windowing system. That material provided Charles with invaluable sample code while he programmed /etc/publisher.

B. Steven T. Kirsch and Vickie Blakeslee

Frame's founding CEO was Steven Kirsch, who earned undergraduate and master's degrees in electrical engineering and computer science at the Massachusetts Institute of Technology (MIT). While there, his frustration with the unreliable roller-ball-based computer mouse used on Lisp machines inspired him to invent and patent an early optical mouse. In 1982 he founded Mouse Systems to bring his optical mouse to market, and Sun Microsystems was one of Steve's original equipment manufacturer (OEM) customers.

Another founding member of Mouse was Vickie Blakeslee, who came to the company from the University of California, Berkeley Computer Science Department, where she was the department's administrative manager. She was a perfect partner for Steve. While he was brilliant and exuberant, he

D. J. Murray cofounded Frame Technology in 1986 (ClearEyePhoto.com).

also changed direction at lightning speed and was not always interested in the minutia of running a company. Vickie provided just the right balance, also being incredibly smart, super organized, efficient, and methodical.

C. David J. Murray

My interest in computer software development began in the mid-1970s when I was working on a doctorate in music composition at the University of Illinois, Urbana-Champaign. My path from Illinois to Frame included four key experiences that would make crucial contributions to my work at Frame.

At Illinois I learned about graphical interface design from my work with the Plato computer-assisted learning system [3]. Plato was a time-sharing system supporting thousands of simultaneous users via graphics-capable touch screen terminals. I designed and implemented many Plato programs, including an early computer music composition program (PLACOMP) that allowed composers to enter both coded compositional algorithms and traditional graphical music notation on the Plato screen. Personal and statistical feedback on my programs' usability from hundreds of users helped me understand how people interact with graphical interfaces.

In 1978, while working on the final stages of my doctoral program, I became a music theory instructor at the California State University, Northridge (CSUN). By that time, I was more interested in writing software than teaching music, so I switched careers and became a staff programmer at the CSUN computing center. Part of my job required that I consult volumes of technical documentation to help students use programs I barely knew. This experience exposed me to industry norms for the formatting of technical documentation from a range of industry leading vendors.

In 1980, I was a programmer at System Development Corporation in Santa Monica, California, helping to develop a file server called OFISFile that was eventually marketed by Burroughs [4]. It supported files from market-leading word processors, including Wang, Xerox, Vyadec, NBI, and IBM. My job was to create a universal file format to store documents within OFISFile and to write translators that converted documents between each word processor's native format and the universal one, in either direction. I attended training classes from each vendor and worked with expert users to develop a detailed understanding of how each word processor worked and how its features were typically used to format documents as well as its document model, file encoding, and user interface design.

In late 1983 and 1984, at Telos Software Products in Santa Monica, I was the lead designer and coauthor, with Matt Jacobs and Howard Metcalfe, of an early award-winning program for Macintosh called FileVision [5]. FileVision combined a vector-oriented drawing program with a flat file database supporting multiple data types. Every vector object in the drawing could have an associated record in the database. The product shipped with a sample file that was a drawing of the United States. States were represented by polygons that had associated database records (type State), and cities were represented by user-definable icons having associated records (type City). You could do all the straightforward things this design suggests: select a city or state object in the drawing and bring up its corresponding record, show or hide objects based on their record type or data, and automatically control the appearance of the graphics based on their underlying data. In response to a database query, the picture would redraw with matching objects highlighted and nonmatching objects grayed out. Steve Jobs liked the associated visual effect so much that he frequently demonstrated it in live presentations and featured it as the ending clip of a 1985 national television commercial.

In a feature we considered self-evident, every record had a system-defined field that determined what happened when the user double clicked on the associated graphic object, with one option being to open another FileVision picture file. For example, double clicking on the Florida polygon in the US map would open a more detailed drawing of Florida. Because of this feature, as far as I know, FileVision is the earliest commercial program for a personal computer that let nonprogramming end users create what we now call "hypertext applications" [6]. I demoed FileVision to Bill Atikinson in 1984, and he later told me that it inspired him to write HyperCard.

FileVision was the AFIPS 1986 Outstanding Software Product of the Year [7] and InfoWorld's software product of the year. It also won praise and awards in magazines such as MacWorld and Byte. This visibility led to my meeting and working with Steve Kirsch, and it supported my early working relationship with Charles Corfield. In late 1985, Steve recruited me to Mouse Systems as director of application software development, but the CEO who had been installed at Mouse by its investors never supported the software projects Steve and I proposed. After a few months at Mouse, I was on the lookout for something else to do.

III. Coming Together of the Founders

In early March of 1986, in the context of that growing frustration at Mouse, Steve and I fatefully visited John Gage at Sun to discuss some product ideas, but he insisted on giving us a demo of /etc/publisher, which he called "a delightful little program by a genius in New York." What he showed us was a very Mac-like program that was an intriguing blend of MacWrite and MacDraw, but it had the full power of a Sun workstation behind it. It was clear to us that this was a potentially life-changing moment.

Figure 1. Frame Technology founders in about 1987. From left to right, David J. Murray, Charles Corfield, Vickie Blakeslee, and Steve Kirsch. (Courtesy of Adobe Corporation)

Steve reached out to Charles in New York that evening. The three of us had a few long phone calls, and then Steve flew Charles to San Jose to discuss us all working together. After meeting for a few days, we agreed to start a company together, with the goal of bringing a document authoring and publishing tool to the Sun Workstation market based on /etc/publisher. Figure 1 shows the four founders.

By the start of April, we had rented a three-bedroom house near mine in Morgan Hill, California, and Charles arrived with his loaner workstation a few days later. He used part of the house as his residence, and the remainder became what we playfully called “Frame Labs.” Steve, Vickie, and I turned in our resignations at Mouse Systems on April Fools’ Day and turned our attention to Frame full time by mid-April.

The complementary skills and experience we each brought to Frame were clear to us, even during that initial week of discussions in March. Charles was a brilliant thinker and coder, a driven, hard worker, and yet a fun and well-balanced person. Steve, also a brilliant engineer, had demonstrated significant business and entrepreneurial skills and experience. I brought strong programming skills and software development management experience to the team, plus domain-specific knowledge that directly related to our product. This included detailed knowledge of how industry-leading word processors worked, broad knowledge of the formatting requirements for technical documents from a range of leading technology vendors, experience designing and implementing a commercial vector drawing program, experience and some industry credibility in the nascent area of hypertext applications, and experience designing GUIs in general and Mac programs specifically. And Vickie was completely in her element taking care of all the nuts and bolts of starting a company: leasing administrative and sales office space in San Jose; setting up systems, policies, procedures, and payroll; and taking care of everything else that a company needs to run well on a day-to-day basis.

Having worked in numerous dysfunctional software development environments prior to Frame, I was determined to not repeat the mistakes I’d seen. I’d just finished reading Richard Feynman’s book Surely You’re Joking, Mr. Feynman! [8] and had been struck by his description of how decisions were made at Los Alamos. Here I paraphrase what he described:

- Everyone had an opportunity to propose solutions to problems being faced.
- Everyone got to discuss the pros and cons.
- A single, trusted decision maker would decide which of the proposed solutions to adopt based on the merits.
- Everyone would then get behind that decision and move forward, doing their best to help it succeed. No looking back. No second guessing. And no in-fighting.

I thought this was brilliant and that it would work well for us at Frame. So I wrote a memo to the other three founders in which I proposed a similar model, assigning each of us final decision making authority in the areas that made the most sense based on our backgrounds and roles. We would all seek out and welcome input from each other and freely discussed pros and cons. But each of us would have decision-making authority in our respective areas. Charles controlled the code’s architecture and technical design and whether any proposed functionality was practical to implement. I controlled the product feature set, the document model as understood by the end user, and the user interface. Steve controlled all of the business, sales, and financial decisions, and Vickie controlled all of the company operational decisions.

IV. Origins of the Company and Product Names

Steve Kirsch chose the company name based on his experience in naming Mouse Systems. Steve had originally named that company Rodent Associates, which seemed clever at the time, but led to nothing but confusion when he called on customers, so he changed it to Mouse Systems. But even that name was problematic: people would hear “mouth” instead of “mouse” or think of Mickey. This time, Steve vowed, his company would have a solid sounding name that was easy to say and spell and that would be taken seriously. “Frame” fit the bill.

Steve chose the word frame because of his enthusiasm for an early outline processor called Framework. Although the frames in /etc/publisher shared only a few similarities with those in Framework, Steve seized on the similarity and hoped that our product would add outline processing too. This did not happen until native Standard Generalized Markup Language (SGML) editing was added to Structured FrameMaker five years later.

We all agreed that we needed a new product name, and that came to a head in May as we prepared for a demo at Apple. Charles noticed a framing store in downtown Morgan Hill called The Frame Maker and decided that would work, at least temporarily. I always thought the name was too similar to

PageMaker and put too much emphasis on frames, but we never got around to changing it, and it worked well enough.

V. Frame's Product Strategy and Target Markets

Because we started the company with no outside funding, our initial product strategy was highly focused and pragmatic: turn Charles' /etc/publisher into something that we could sell as quickly as possible. The target market was "engineering users of Sun workstations." The required features and important use cases were driven by the types of documents those users created.

Although /etc/publisher was an impressive prototype, it required a lot of work before it could be considered a real product. For example, it was not capable of producing printed output, had numerous user interaction rough edges, and supported only a single master page. Charles and I spent six intense months modifying and enhancing the original code, quality-assurance testing it with significant help from Steve Kirsch, and writing end-user documentation for what we called prototype FrameMaker (also called FrameMaker 0.6), which Steve started selling around October of that year.

We actually presented FrameMaker 0.6 to customers as a presale of FrameMaker 1.0, with the benefit that our risk-taking customers would have use of the prototype and beta versions in advance of 1.0's release and also have the ability to influence the design and feature set of version 1.0. Our first paying customer was an engineering services group at John Deere in Moline, Illinois. Version 0.6 was surprisingly stable and usable, and we were able to sell a few hundred licenses between its release and the release of version 1.0 in March 1987. This allowed the company to become cash flow positive in less than a year.

Even though our initial target market was narrow, we had a fairly clear idea from the outset of how we would broaden it over the next few years: We'd enhance the functionality so that the product could also be used by technical publishing professionals to create complex, high-quality printed and online (hypertext-based) technical publications per Charles' original goal while at Columbia. We also saw an opportunity to bring the same functionality to other platforms: first to other Unix workstations and then to the Mac and PCs, when the processing power of those platforms caught up to the level of mid-1980s workstations. Key aspects of our multiplatform strategy were that we would fit in and behave like a native application on each platform and that we would support transparent file sharing across platforms.

Our initial target market was ripe for the taking. At the time, engineers using Sun workstations lacked inexpensive and simple tools for creating richly formatted work documents, such as design specifications, technical reports, software tool documentation, and memos, even without the inclusion of graphics. Most Sun users wrote these types of documents in vi or Emacs, and they formatted them by embedding troff or TeX markup. Or they sent plain text documents to their internal tech pubs departments, who used expensive dedicated technical publishing tools like Interleaf TPS to turn those documents into publications.

In mid-1986 Interleaf was the market-leading vendor of technical publishing software that ran on standard workstations and printers. Their primary product, TPS 3.0, ran on Sun, Apollo, Dec, and IBM RT workstations, and they had over a dozen sales offices worldwide. When digital publishing technology pioneer and Atex cofounder Charles Ying visited us at Frame Labs in late 1986, he politely watched our demo and then used a colorful analogy involving an ant and an elephant to characterize our chances of competing with Interleaf.

Still, we were not discouraged. We knew that Ying's metaphor was more apt that he intended. Interleaf TPS was big and expensive. According to the July 7, 1986 Seybold Report on Publishing Systems, a single-user Interleaf "starter system" cost $29,000 and a typical four-workstation system cost $140,000. These prices were required, in part, by their architecture and therefore difficult for them to reduce and still make money. Interleaf used its own proprietary windowing system, UI toolkit, screen fonts, and print rasterization software. Installing it on a stock workstation required hours of onsite work by an Interleaf tech, and it made the workstation unsuitable for running other engineering applications.

Frame, on the other hand, was built using Sun's native windowing system, and it used standard LaserWriter screen fonts and our built-in PostScript generator. This made it trivial for an end user to install Frame on the workstation they already used, with no support from us, and to run it alongside their other applications, allowing them to conveniently document their work. We could easily charge just $2,500 per license and make money, and our customers could adopt FrameMaker widely with no additional hardware costs.

VI. Key FrameMaker Features and Attributes

The defining aspect of FrameMaker was that it was designed and priced to be an all-in-one authoring and publishing tool. This allowed it to be used in a workflow where both content and layout/design could easily be modified across a document's entire life cycle. It allowed both single users and teams to design, author, and publish complex documents ranging from flyers and newsletters to long technical documents using both ad hoc and formal structured document designs. Most other document authoring and publishing tools of the time required workflows, where content creation and document layout were done sequentially, in separate tools, and often by different people.

In the case of most desktop publishing and word processing products, this separation was driven by feature limitations. The word processors were good for textual content creation, but they could not handle complex page layouts or graphics integration. The page layout programs, on the other hand, were not very good for content creation. Interleaf was driven largely by the cost and architectural issues already mentioned. In all cases, the resulting workflow was full of friction. It made the inclusion of things like cross references and diagrams

awkward for authors, and it made late content changes expensive, as they might “break” the layout and related issues like section numbering, tables of contents, cross-reference page numbers, and so on. FrameMaker allowed for a workflow that completely eliminated these types of friction.

When it was released in late 1986, FrameMaker supported an unusually flexible and powerful approach to the integration of text and graphics. Other tools used two rather different models to support this integration:

- *Automatic layout model*. In this case, the backbone of each document is a single, main stream of paragraph-oriented text, like the text that forms a traditional novel, that is paginated automatically based on document-level properties like page size, margin sizes, and numbers of columns. Graphics were accommodated (if at all) by attaching or anchoring them into the text at various points and letting the system take care of layout using more or less complex layout properties and rules. This model was used by products like Interleaf and ArborText in the mid-1980s and by major word processors as they added graphics support over time. The model was ill suited to handle complex, irregular text and graphics layouts that were typically handcrafted by graphic artists.
- *Manual desktop publishing model*. First widely associated with Aldus PageMaker and then Ventura, Quark, and many others, with this approach text and graphics page elements could be placed anywhere on the page by an end user, like a virtual paste up board, with the special ability to automatically hyphenate, justify, and paginate text within linked rectangular text-column objects. This model excelled at supporting handcrafted layouts, but it struggled to support the regular types of layouts at which word processors excelled. To be fair, desktop publishing programs like PageMaker and Ventura included features to help automate the layout of longer, regularly formatted documents too. But few people used those products as long-document authoring environments.

FrameMaker was adept at working like a traditional word processor with embedded graphic frames and like a desktop publishing system for the creation of handcrafted irregular layouts, or even as a mix of the two. It accomplished this using a document model that was based on objects placed on pages. Users could directly manipulate those objects to create arbitrary handcrafted layouts. But FrameMaker also had a set of property-driven behaviors that allowed it to automatically mimic the interactive and layout behavior of traditional word processors and long-document formatters. Using those properties, it could automatically create pages prepopulated with objects to hold text as it was entered into a document, paginate that text as needed to lay it out properly, delete unneeded empty trailing pages automatically, and automatically adjust the size of pages and the layout of the columns on those pages in response to the user modifying the document’s page size or resetting the column count and margins at a global level.

Within the context of this flexible layout model, FrameMaker supported a powerful set of important technical-documentation-oriented features:

- Real-time automatic numbering of headings (chapter, section, subsection, etc.), figure titles, table titles, and any other textual entity requiring automatic numbering.
- Real-time automatic cross references.
- An embedded interactive equation editor.
- Anchored frames for holding graphics that were automatically paginated along with the text, based on a variety of automated placement options.
- Sophisticated typographic and line layout capabilities. This was generally referred to at the time as hyphenation and justification (H&J), but it also included control over letter spacing, automatic kerning, and other layout attributes.
- Automatically laid out tables, including sophisticated automatic pagination properties and automatic repeated headers and footers for multipage tables. Each cell in a FrameMaker table could support the full range of features associated with all other FrameMaker text flows except pagination, including the embedding of graphics and equations, automatic numbering, cross references, and hypertext links.
- Multilingual documents, supporting word range level language-specific hyphenation and spell checking.
- Support for long documents stored across multiple files, while still supporting shared formatting catalogs and cross-file automatic numbering and cross referencing.
- Automatic generation of tables of contents, indexes, and other types of front and back matter.
- The ability to generate different presentations of the same document, such as print and online versions, from a single source.
- Support for automatic and manually created hypertext linking within and across documents when viewed in Frame’s online viewing program FrameViewer (a precursor to today’s web browsers).
- Native SGML support in later versions of the product.

The mechanisms that supported these features can best be understood through the detailed description of FrameMaker’s document model, which is described in detail in a web extra page.

A. Cross-Platform Binary File Sharing

By the end of 1992, Frame’s products ran on more than 25 different types of Unix workstations, Macs, and Windows PCs, all of which could transparently share binary document files on networked file systems, which was a unique and important competitive advantage at the time. Two things made this a challenge.

First, system-level binary file read/write routines called from programs like FrameMaker, written in C, were not able to automatically handle differences in big-endian and little-endian CPU architectures. These differences mattered in shops that ran workstations from different vendors. They even mattered in pure Sun Microsystems shops because Sun offered

workstations based around both Motorola and Intel CPUs. For performance reasons, Frame wrote its binary files using whichever endian format was native to the platform on which it was running. But when opening binary files, it was able to detect when the file had been written in the "other endian" format and swap things around if necessary.

The architecture that supported all of this endian magic was known as DRF format. The public meaning of that acronym was Directly Readable File format, but we really chose it because it was designed and implemented by Frame's official first employee (after the four founders), David R. Fuchs, a brilliant software developer who had been Donald Knuth's right-hand man on the TeX project at Stanford before he joined Frame. To avoid the potential confusion from having two David's in the company, people referred to us by our email names, calling me DJM (said Di-jum) and him DRF (said Derf).

The second challenge presented by cross-platform file sharing involved font handling. Different platforms often had different fonts available, especially for screen display. FrameMaker did not transcend this problem elegantly, as was later achieved by Adobe in the PDF format. But we were able to manage the issue by temporarily substituting available fonts for missing ones. This meant quick edits made to a document on a system with missing fonts did not damage the document and its formatting when those documents were subsequently used on computers where the intended fonts were present.

B. Maker Interchange Format

FrameMaker could export, open, and import documents represented in an ASCII-based, human-readable file format called Maker Interchange Format (.mif). I designed MIF to use a syntax similar to SGML, but it was significantly less complicated to parse. It was never intended to be written directly by end users (so I did not include any of SGML's markup minimization capabilities) because we did not want to license a full-blow SGML parser and because I had little time to implement the MIF reader in our rush to get FrameMaker 1.0 released.

One important use of MIF was to enable customers to use FrameMaker (and its companion FrameViewer) for automated database publishing applications. Customers would write custom code that generated MIF documents based on information they pulled from databases. While a MIF file could fully represent all of the information contained in a FrameMaker binary file, it was not required to do so in order to be readable.

A typical database publishing application would be implemented by customers as follows: Using the Frame API, the program would tell FrameMaker to open a binary FrameMaker template document created by a designer using interactive FrameMaker, containing the desired page layouts and tagged text styles. It would then generate a MIF file that only specified the body text, formatted by including character, paragraph, and table tags within that text's markup, but none of the MIF specification for the actual formatting attributes associated with those tags. Next, it would use the API to cause FrameMaker to import the MIF file into the previously opened template. FrameMaker/Viewer would use the tags in the MIF to properly format the incoming text and tables based on the corresponding formats it found in the template document. The resulting document could then be displayed as a locked hypertext document using FrameViewer or printed using FrameMaker, via the API.

C. Licensing Technology

Another key feature that contributed to our competitive success was our license server technology. Rather than requiring each workstation at a company to have its own license, customers could purchase a license for "N simultaneous users." This let part-time users essentially share a license, while reserving some licenses for full-time users. This allowed everyone to take advantage of the product, without excessive cost for part-time use.

Increasing the number of simultaneous users could be accomplished over the phone and through email. We implemented all of this via a centralized license server system that I believe was a first in our market.

D. Portability Architecture

The initial version of FrameMaker was written for Unix/SunView, but our initial product strategy was to make FrameMaker available across a variety of platforms. Even before FrameMaker 1.0 shipped, we began a project to redesign the code with portability in mind. The new architecture separated the code into what we internally referred to as the Core and the Fields (in honor of founder Charles Corfield—you may note a pattern emerging here), one Field for each OS/windowing system. The design requirement was that all the non-platform-specific functionality in the product be implemented in a completely sharable, device-independent set of C libraries, which communicated with the operating environment not via direct OS-specific calls, but through a virtual machine interface that we devised and that provided abstracted services for file I/O, networking, windowing, font services, graphics, a UI toolkit, and so forth.

The primary work required to port FrameMaker to a new OS/window system was to implement the virtual machine in that environment and compile. This of course is not a novel idea, but it is easier said than done. The detailed design of the Core/Field architecture and the rewriting of FrameMaker 1.0 to use it under SunView was led by our second nonfounder employee, Ken Keller. Ken was a PhD computer scientist from UC Berkeley, where he did groundbreaking work in integrated circuit CAD systems design and graphics editors. Before joining Frame, he cofounded Cadence and helped to build it into one of the leading CAD software companies.

Our portability architecture, known formally and publicly as Device Independent Maker (DIM), was highly successful. A single developer named Greg Cockroft implement the Field virtual machine for the NeXT system and had FrameMaker running on it in under a month. The initial implementation of the Field for X Windows, written by Don Bennett under an OEM deal with Tektronix that was signed in November 1987,

took a little more than four months to develop, factoring in the holidays. It was released in April 1988 and served as the basis for a string of X Windows releases for workstations from other companies.

The period between our founding in 1986 and our IPO saw an explosion of competing windowing systems for Unix workstations, such as SunView, OpenLook, XWindows, NeWS, DecWindows, and NextStep/DisplayPostscript. By the end of 1992, there were versions of FrameMaker that ran under all of those Unix windowing systems as well as MacOS and Windows 3.0. All those versions shared the same Core code modules that implemented Frame's application-specific document-processing features. At the same time, FrameMaker looked and behaved like a native application in each of those environments, from the way it used windows and presented menus and dialogs, to its support for inter-application data exchange through system-specific clipboard mechanisms. This made it fairly easy for new Frame users on a specific platform to learn the program. At the same time, users who already understood FrameMaker's document model could quickly switch from one platform to another and still feel at home in the program.

This architecture made it practical for us to release new versions of FrameMaker across all platforms more or less at once because the new document-processing functionality would be implemented within the Core, using preexisting virtual interfaces wherever possible. When not possible, the virtual machine would be expanded and then updated for each platform.

VII. Beyond Document-Processing Features

FrameMaker's technical success certainly relied on these features, but it also relied on the product's performance, capacity, reliability, and usability. Our design standards required that FrameMaker respond more or less instantaneously to interactive editing and online viewing, even in documents containing hundreds of pages filled with complex automated content. We accomplished this performance, capacity, and reliability in part through clever coding and rigorous testing and in part through strategic design. Our multifile book feature made it easy, efficient, and sensible for users to break up large documents into smaller linked files, typically at chapter or section boundaries, while still allowing easy cross-section browsing, editing, and mass format and content updates.

On the usability front, new and casual users appreciated FrameMaker's use of their platform's native mouse-based GUI conventions. But more importantly, we carefully streamlined the product for expert users. Here are just a few examples:

- Every command could be invoked by an easy to remember and comfortable to type three-key sequence, which could be combined with other input in easy to record macros.
- All formatting styles (property sheets) could be applied in a few keystrokes, without touching the mouse.
- Paragraph styles could optionally apply a different style to the next paragraph created when pressing the Return key.
- Erroneously transposed characters could be fixed with a single keystroke.

Taken individually, each of these features, and the many other accelerators we implemented, sound trivial. However, they combined to create a user experience that our daily users valued and were unwilling to give up when competing companies tried to win their business. The most common story we heard from customers switching to FrameMaker from Interleaf was how much less tedious it felt to use FrameMaker. I never detected a serious attempt on Interleaf's part to address that weakness. Customers switching from Word, PageMaker, and many other Mac and PC products would talk about how those products worked fine for short documents, but they became unreliable and sluggish when documents exceed more than a few dozen pages.

And finally, FrameMaker's success owed a great deal to the fact that we used it to produce all of our own documentation at Frame, from internal engineering specs, to marketing and sales materials, to end user product manuals and training materials. If something did not work well, everyone who wrote documents at the company was impacted. And when a developer did a great job implementing something or fixing a problem, everyone knew and thanked them for their efforts.

VIII. Frame's Funding Over Time

Frame was initially funded with a $100,000 loan from cofounder Steve Kirsch. Because we were able to start preselling FrameMaker 1.0 within six months by letting customers use FrameMaker 0.6, we became cash-flow positive within our first year, and we continued to finance operations with revenue from product and maintenance sales into 1987 [1].

In June 1987, we had the good fortune to raise $1 million in operating cash without giving up any equity, and with very little cost or effort, through a source code licensing deal with Toshiba in Japan [9]. At that time, Toshiba had an OEM relationship with Sun that allowed it to sell Sun workstations under the Toshiba label. Toshiba had developed a Japanese version of the Sun operating system and was investing millions in developing Japanese versions of important SunView applications. Our source code licensing deal with them was masterfully negotiated by Steve Kirsch and Vickie Blakslee (with input from the other founders, our incredible outside council Steve Wurzberg, and several board members): Toshiba paid us $1 million up front, as a nonrefundable advance against future royalties that they would owe us if and when they started selling Japanese FrameMaker. They received the source code to FrameMaker 1.0 and about a week of developer training, plus a year of support for their developers, which they rarely used. There were no other source code or development strings, so our development of subsequent versions and new features was not complicated by the Toshiba deal. Their license only allowed them to make a

version of Japanese FrameMaker that could be sold in Japan on Toshiba-labelled Sun Workstations, a market we had no intention to enter on our own. Toshiba's version of FrameMaker was fairly successful in Japan, generating additional royalties for us in excess of $5 million over the next few years.

The success of the venture with Toshiba served as a model that we repeated with Digital Equipment Corporation and Wang in the US and with Matsui in Japan. Each of these deals brought in around $1 million up front, for a source code licensing deal that only allowed the licensee to port the product to their own proprietary platforms, which were markets we never would have entered on our own.

In May 1988, we raised $3.1 million in Series A venture funding from Menlo Ventures and Hambrecht & Quist. In March 1992 we raised approximately another $10 million in Series B funding primarily from those same sources. In early 1992, right before our IPO, those outside investors owned approximately 20% of Frame, the founders owned around 54%, and the remaining 25% was owned by Frame employees and directors [1].

IX. Frame's Marketing, Sales, and Distribution

In our first few years, Frame primarily used a telemarketing sales model, with some in-person direct sales for major accounts. This approach was aided by the fact that Sun's salesforce often used FrameMaker in their demos and because of the positive exposure we received in the press, especially in the Seybold Report on Publishing System. It was also possible because end users could easily install our software themselves and because our licensing technology let us offer free trial periods.

By the time of our IPO, we had developed a complex multichannel marketing, sales, and distribution strategy. In North America we distributed our Unix products through a direct sales force as well as through value-added resellers (VARs), resellers, and OEMs; we distributed our Macintosh product through distributors for sale to resellers. In Europe we sold our products through distributors, OEMs, and VARs. And in the Pacific Rim, we relied on OEM relationships [1].

Our North American field sales representatives were located in 15 locations across the continent including Atlanta, Baton Rouge, Boston, Chicago, Cleveland, Denver, Dallas, Los Angeles, New York, Orlando, Ontario Canada, San Jose, Seattle, and Washington DC.

We had approximately 230 Unix VARs including Highland Digital, Workgroup Technologies, and Software Associates. Our distributors, resellers, and dealers included Egghead Discount Software, CompUSA, Corporate Software, Ingram Micro, Merisel, and Access Graphics.

Frame's OEMs fell into two broad categories: standard and customized. Our standard OEMs were licensed to sell Frame-branded versions of FrameMaker that we adapted to run on their hardware, typically with some cobranding. These OEMs were responsible for all marketing, sales, and end user support of their versions of FrameMaker, and they paid per copy royalties. Most of these deals involved nonrefundable prepaid royalties to cover our upfront costs. Standard OEMs included Altos Computer Systems, Bull HN Information Systems, Data General, Intergraph Corporation, MIPS Computer Systems, Motorola, Sequent Computer Systems, Silicon Graphics, Sony, and several more.

Our customized OEMs purchased limited source code licenses and added proprietary enhancements. These OEMs included Toshiba, DEC, Siemens, and Mitsui & Co.

Our core customer base consisted of engineers, scientists, and technical publishing professionals at corporations across a range of industries, government, research organizations, and universities. See the web extras for a partial list.

Frame's annual revenues between our founding and IPO were as follows: $3.4 million, $8.8 million, $16 million, $25.4 million, and $41.7 million for the calendar years 1987 through 1991, respectively. For those same periods, profits (or losses) were $500,000, $1.2 million, ($300,000), $300,000, and $2 million. While those third and fourth years were more disappointing than we had planned, they reflected our general strategy in the first four years to invest as much revenue as possible back into the company to develop products and to expand our sales and marketing resources [1].

X. Frame's Internal Growth and Culture

Frame faced significant hiring challenges and growing pains as we brought on more and more employees to execute the development, marketing, and sales strategies described earlier. From 1986 through 1991, employee counts at the end of each year leading up to our IPO were as follows: 10, 37, 80, 140, 191, and 297, which represents annual growth rates of 370%, 210%, 175%, 136%, and 150%, respectively. (Employee counts are drawn from the internal Frame newsletter, called Frame Flyer, and information in the 1992 Frame Technology Corporation Prospectus [1].)

In the first few years, there was little turnover, as most employees were passionate about their jobs and the work was fun, for the most part. On the development side, I wrote detailed initial specifications for all new features. Then I worked closely with each developer to modify those specs based on their input while ensuring an overall coherence to the product's operation. The result was that each developer felt a strong sense of ownership over the functionality they implemented. Although Steve, Charles, and I often frustrated our employees with our willingness to instantly change plans in response to market and customer input, we encouraged open feedback and got earfuls in response.

I think our leadership style was tolerated largely because employees could see that we carefully considered their input and did our best to treat everyone with respect and fairness, even if we did not always do what they wanted. Lauren Benton, who was Frame's in-house recruiter and then director of Human Resources from 1987 to early 1990 recently recalled:

> Frame was an exciting, fast-paced, fluid environment that was a meritocracy. We hired, promoted, and rewarded those that performed (achieved company goals and objectives). I enjoyed my role in growing the company and supporting managers in leading their teams.

I still hear from early Frame employees, who reminisce fondly about the company's early days. Our internal monthly newsletters, written by frontline employees without much oversight by senior management, reflected an irreverent and good-natured tone. A typical example was this fictitious job posting in the October 1988 Frame Flyer: "Job Title: Wife for CEO Steve Kirsch… Purpose: Manage Steve. Key Functions: Get him dressed in the morning… Impact of Position: Steve would stop driving us all crazy."

But even in those early days, all was not rosy. For a brief period, the sales team was infiltrated by a group of Scientologists who used covert and unethical tactics to coerce others to convert or leave. An early vice president and an outside director spread a coordinated set of lies that briefly created chaos at the executive level. Their goal was to force the founders to accept a lowball buyout offer they were secretly behind. And our engineering team leaders were often very frustrated by what they described as our "whack a mole" approach to software development.

In the summer of 1988, we had the good fortune to hire Interleaf's vice president of North American sales, Steve Klann, as our VP of sales. Steve brought several of his key team members with him, including one of Interleaf's top salespersons, Claudia Ferrini, and Interleaf's top technical sales specialist and product evangelist extraordinaire, Max Hoffman. While Frame had been competing successfully against Interleaf on many fronts, Steve and company added quite a boost. Steve was the architect of the multichannel sales strategy described earlier. Max was widely recognized in the industry as an expert on Interleaf's products, so when he spoke about the relative merits of FrameMaker, people listened. He also provided my product design team and me with valuable insights into FrameMaker's competitive strengths and weaknesses that had significant influence on the product's evolution.

By the end of 1989, there was a general consensus that we had grown to the point where we needed to bring in more experienced upper management, and in February 1990, Paul Robichaux took over as CEO. The transition from Steve Kirsch to Paul was highly disruptive and, at the time, seemed nearly fatal. In his first three weeks Paul laid off 9% of the staff and brought in a new CFO and vice president of engineering (who lasted less than three months). Steve Klann, Lauren Benton, and several other key players were gone by the end of the year. But to his credit, Paul got things back on track and lead Frame to a successful IPO two years later.

Under both Steve and Paul, diversity was a key contributor to Frame's culture and early success. I do not recall an explicit mandate to seek diversity in our hiring practices, but it nevertheless happened across several dimensions, including gender, race, and sexual orientation. This enriched us in every way you can imagine. Teams produced better results because of the different perspectives brought by their diverse members, and the social fabric of the company was refreshingly open, supportive, and welcoming because of it. While our first two engineering hires, David Fuchs and Ken Keller, were highly talented men, the next two, Sandra Sundberg and Sharon Lam, were highly talented women who served as key members of the engineering team for years. That hiring pattern continued, and at the time of our IPO, all levels of the organization included a fairly diverse mix of genders, ethnic backgrounds, and sexual orientations, even at the VP level. Our senior executive team included two women (Patricia House and Vickie Blakeslee), an African-American (Drew Newton), and a gay man (William Edwards). I regret that there is not space in this article to write more about these people and the many other individuals who contributed to Frame's success.

XI. Frame Post IPO

Frame's performance after our IPO was rocky. In 1992 revenue rose to $79.9 million (from $41.7 million in 1991) and net income was $7.2 million. But in 1993 revenue fell to $59.8 million, and the company suffered a stunning net operating loss of $32 million. The stock fell from a first quarter high of $18.81 per share to a second quarter low of $5.25 per share.

In the summer of 1993, George Klaus was brought in to replace Paul as CEO. George was a rock solid, smart, and dependable leader who successfully put Frame back on its feet. In 1994 revenue rebounded to $77.8 million, and profits grew to $11.8 million. By the end of the second quarter in 1995 the company was on track to reach annual revenues over $90 million, and the stock had reached a high of $31.75 per share, setting the stage for our merger with Adobe that fall at a valuation of almost $500 million [2].

Acknowledgments

I wish to thank Frank Romano at the Museum of Printing in Haverhill, Massachusetts, for access to its collection of Seybold Reports and the numerous early employees whom I interviewed in preparation for this article.

References

[1] "Frame Technology Corporation Prospectus," Frame Technology, Feb. 12, 1992.

[2] "Frame Technology Corporation Proxy Statement/Adobe Systems Incorporated Prospectus," Adobe Systems, Sept. 22, 1995.

[3] D. L. Bitzer and R. Johnson, "Plato: A computer-based system used in the engineering of education," *Proc. IEEE*, vol. 56, no. 6, 1971, pp. 960–968.

[4] K. B. Noble, "Burroughs expands product line for use in automated offices," *New York Times*, vol. 130, no. 44989, June 24, 1981, p. D5; https://www.nytimes.com/1981/06/24/business/burroughs-expands-product-line-for-use-in-automated-offices.html.

[5] "FileVision," Macintosh Garden, http://macintoshgarden.org/apps/filevision.

[6] J. Conklin, “Hypertext,” *Encyclopedia of Microcomputers*, vol. 8, A. Kent and J. G. Williams, eds., New York, NY: CRC Press, 1991, p. 402.
[7] “Update,” *Computer*, vol. 19, no. 8, Aug. 1986, p. 86.
[8] R. P. Feynman, *Surely You’re Joking, Mr. Feynman*! New York, NY: W. W. Norton & Company, 1985.
[9] “Japanese Frame Maker [sic] from Toshiba,” *Seybold Report on Publishing System*, vol. 16, no. 20, June 29, 1987, p. 44.

David J. Murray cofounded Frame Technology in 1986 with Charles Corfield, Steven Kirsch, and Vickie Blackslee. At Frame, he was lead designer and coauthor of Frame's products, a director, and a vice president. Prior to founding Frame, in 1984 he designed and coauthored the award-winning Macintosh program, FileVision. Following Adobe’s acquisition of Frame in 1995, Murray cofounded the internet mapping company Geographic Services Corporation in 1997. From 2000 to 2008, he held various positions at Propel Software, including vice president of engineering, COO, and interim CEO. Murray has a master’s degree in music theory and composition from the University of Connecticut. Since 2008 he has been a commercial photographer. Contact him through ClearEyePhoto.com.