100 days:
the riskview.com experience

Drs. Tom Love and
John Wooten

Shoulders Corporation
January 1988

1. Go or no go?

In March of 1997, Tom Love was consulting for Infinity Financial Technologies in Mountain View, California. One of the projects that he was asked to investigate was called Falcon. The purpose of the project was to build an Internet-based equity risk management system to be offered without charge for professional investors. The software technology chosen was Java.

Because of Infinity's traditional focus on the professional investment market, the company had underestimated the appeal of such a product for the serious individual investor. So instead of having a few hundred or few thousand users the product could have a few hundred thousand users. Oh what a difference a few hundred thousand users can make!

The recommendation was "feed it or kill it". If the software and its supporting operational infrastructure were designed and sized to support 1000 users, but 100,000 showed up, many users would be very disappointed with their ability to access the site and its performance. Furthermore, the way you design and deliver an application to an investment professional at Fidelity is very different from what the Peoria investment club might expect.

Infinity decided to "feed it", if Tom would agree to serve as program manager for the project. We had 107 calendar days until an important conference was scheduled where this application was to be launched. There were two problems – we had no specification and there was no development team! But we did have 107 days and a partially working prototype, developed by a very smart and hard working Infinity engineer.

2. Planning

There are a couple of critical points in any development project. The first is when the decision is made to start. At that point there is some level of understanding of what needs to be built and what resources and time will be required to deliver the project. Of course we should do everything we can to make these understandings more concrete and mutually understood. But the fact is that great uncertainty exists. Executives must trust their development team and the development team must trust management.

A second critical point is when the specifications are frozen. More about that later.

2.1 management commitment

Senior management has to be constantly involved in high intensity projects. Approval and purchasing cycles have to be optimized. Decision cycles have to be short and definitive. Reviews have to be frequent. A five day delay in making a key decision is intolerable, for example.

2.2 staffing the team

Infinity management decided this project should be done with external contractors for two reasons. There were no Java trained designers or developers available within the company. Secondly, there was the possibility that the project would be spun off into a separate business or cancelled.

Management also decided to hire all contractors, except the Program Manager on an hourly basis. While this certainly made the long hours and the many weeks more palatable, I think many people over estimated its true importance. A motivated team will do what is required for a hundred days to complete an important project.

Borrowing a technique from Wall Street firms, we selected very smart individuals for this project. The academic credentials were impeccable. Five members of the team had doctorates (2 from Stanford; one from MIT); another team member was within a year of completing a Ph.D. at Stanford. Another had an MBA from Harvard. One team member was to be a senior in computer science from Rice, but had an unusual knowledge of computing, acquired developing complex computer games.

Team members were also carefully evaluated based upon their communication ability. How well did they communicate verbally and in writing? How well did we think they would work within a high performance, team environment? How well would they be able to communicate to both commercial partners and end users?

We also looked carefully at the mathematics background of each developer and tester. Ph.D.'s in Physics or Engineering could handle the risk analytics. BS computer scientists might not have been able to, for example.

Working with the team, but not counted as full time, was an extremely knowledgeable person with nearly 20 years of experience at J. P. Morgan. Till Guldimann, the Vice Chairman of Infinity, conceived of this project and convinced Dow Jones to participate in the first place. His ideas, knowledge, support and direct day to day involvement was crucial to the success of this project. His total inflexibility regarding schedule was a virtue, in retrospect.

100 day projects are demanding. If people live too far away, have too many outside interests, or have complicating family problems, they cannot be a full participant. One team member had to be replaced for this reason three weeks into the project.

2.3 environment

What you are about to read will be challenged by more than 80% of the developers who read this. By contrast, 80% of the executives will love the idea.

Seven of the eight full time members of the project team were located in a totally open space about 15 by 30 feet. The exception was the product marketing person who had an office 100 feet away. There were four feet high partitions behind the desks, but no walls, no doors and no dividers. The program manager worked in exactly the same space as the junior tester. The only difference was that he had the smallest screen and the slowest computer.

Adjacent to this team space was a 12 x 12 enclosed office with a speaker phone for multiperson phone calls and small meetings. The room was furnished like a living room – sofa, comfortable arm chairs, and coffee table. A white board was also in the room. As always, the white board was too small.

Other more conventional conference rooms with tables and white boards were nearby and available by reservation.

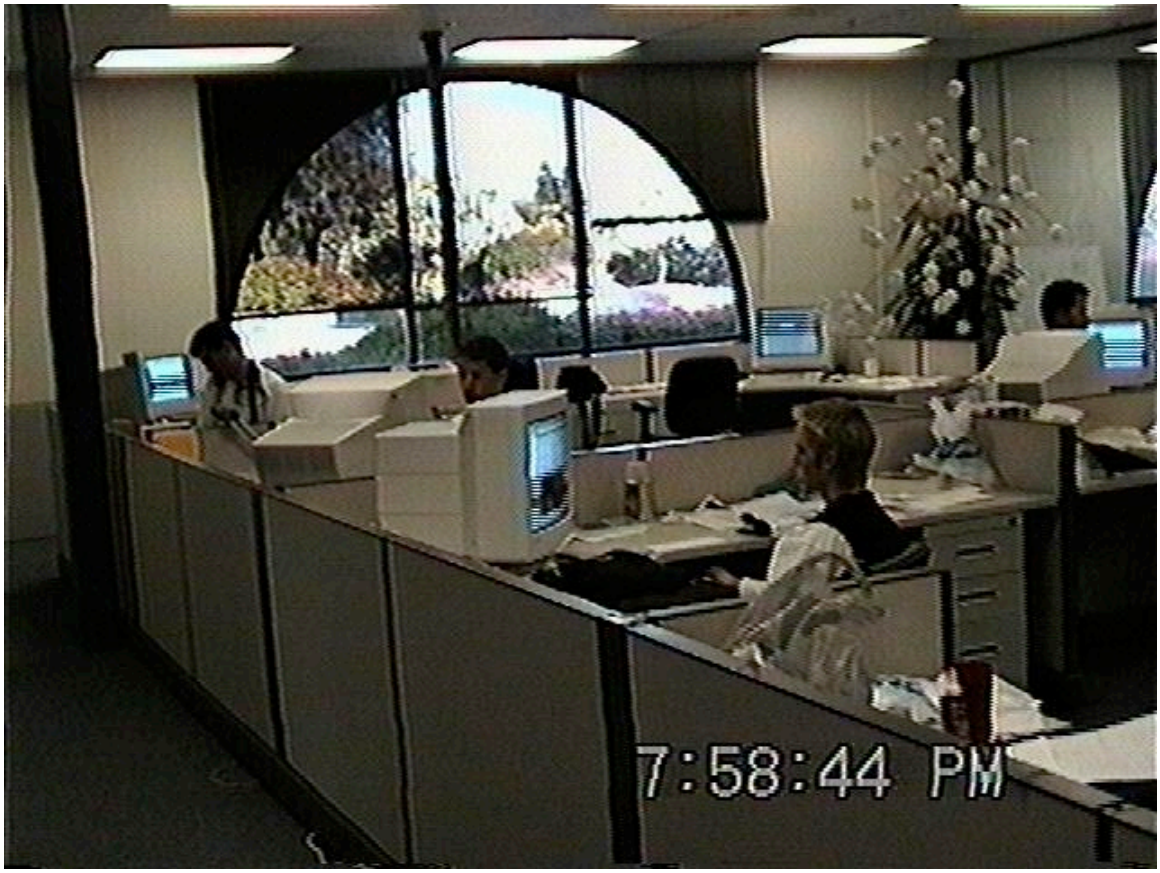


Photo of the physical office environment.

Amusingly, the team was located in the accounting department and across the building from the other software developers. The accountants had never seen a team work so long and so hard. I think they were truly impressed to see what it takes to get a significant product built in a short period of time. They also were very willing to serve as beta testers of the product. The company controller spent a weekend thoroughly exercising the product and won a prize for reporting more errors than anyone else in the company. They wanted the project to succeed.

Each developer had a fast workstation (Pentium 200mHz) with a large screen (20”), fast Ethernet, local printer and a comfortable chair. The software environment was Symantec’s Visual Café Pro with Clear Case configuration control running on a Sun workstation elsewhere in the building. Lotus Notes was used to maintain a bug database and as the project library. Netscape Navigator was available to everyone as well as a dedicated T1 communications line.[1]

Team lunches and dinners were commonplace. Almost every day several members of the team went out for a quick lunch and informal discussions. About twice a week either a lunch or a dinner was brought in for the team, typically organized by marketing.

---

[1] Later in the project this early decision proved critical. It allowed for efficient movement of software releases to three different sites almost as though they were workstations located on our local network.

The project team consisted of a program manager with full commercial responsibility for the project (TL), one architect/developer (JW), two developers, one tester, two people doing testing and documentation, and one product marketing person.

3. System Engineering

Java is a powerful programming language, but its users must keep in mind that network resident applets are actually real time, distributed systems. Building an applet sounds easy; building a real time distributed system is known to be tough.

Designers and implementers of real time distributed systems have learned some important lessons over the years. These include:

1. Do a system engineering analysis (or simulation model) as early in the project as possible to evaluate speeds and feeds.

2. Develop or preferably acquire a test harness capable of reproducing any possible error, including subtle timing or network errors.

3. Design the system from small modular units, which are thoroughly tested before being assembled into larger units.

4. Wherever possible build testing capabilities into all aspects of the system.

In the first week of the Falcon project, we did a system engineering analysis and concluded that the planned operational environment for the application was completely inadequate given the revised estimate of number of users. We also concluded that with the best of environments, downloading such a large applet would be slow. Interestingly this was viewed as a virtue. The idea was that a totally free applet with a significant amount of data would be made available to the public. If users liked what they saw, a commercial application could be quickly constructed and fed with a more comprehensive dataset for which users would pay a monthly fee.

Based upon this work we met with Dow Jones to see if they could support the new requirements, which could consume many T1 lines to efficiently transmit data and large applets to customers. They could not. We had a problem!

Two paths were pursued. One was to look for a third partner capable of hosting such a site with adequate communication capabilities. A second was to begin discussions with professional hosting services capable of supporting such a site under contract. We chose option number one with IBM as the hosting partner, but we had a fall back position that could have been implemented with a week's notice.

4. Early & Frequent Milestones

With a new team, programming language, and development tools we needed to quickly determine if the team could accomplish what it estimated it could. The first such milestone was to complete the previously developed prototype within the first two weeks. This forced the new developers to understand in detail what had been done already and the degree of complexity of this application.

A second milestone involved the entire team. By the fourth week we almost met the following set of milestones:

1. Completed the prototype so that it could be used with real data.

2. Designed the application, including the names of all classes and many methods.

3. Developed a written requirements specification which included a description of the target user community, its size and sophistication and the major behavior to be delivered in this application.

4. An initial mockup of the documentation for the prototype.

5. A complete user level specification of the external interface for the system (screen mockups).

6. A detailed project plan with agreed upon milestones including interaction with external partners.

7. A comprehensive test plan for the project.

The documentation milestone was solidly missed. We had underestimated how much knowledge was required to successfully describe and document this type of application. We chose a professional technical writer with some familiarity with the financial industry.

This staffing mistake was almost fatal. We tried for several weeks to help a very conscientious and smart person to learn what was needed so that she could write what was required. It did not work. Eight weeks into the project we had to replace the technical writer.

The time cushion had been used. Within two days, we found a graduate student from Stanford who was about to complete a Ph.D. in Statistics. We found him by searching through web pages for students and faculty. He was perfect for the job and did an amazing job in a short period. He was not only able to do an excellent job of documentation, but he also contributed significantly to the testing efforts, specifically by doing an independent validation of the analytics within the application.

5. Product Management

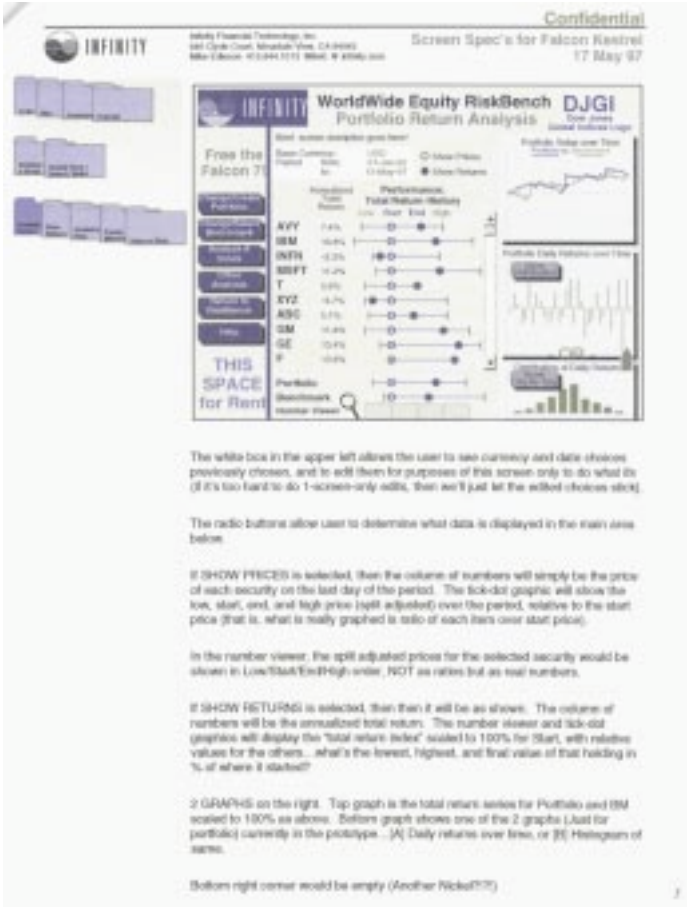
As early as possible, write a one page description of the product. Review this description with all involved corporate executives until they are willing to agree to the spec and sign it. This will take longer and be more challenging than you ever expected.

The tradition in the software business is to write a long and involved requirements document. This document gets followed by an even longer design document – often extending to hundred of pages.

For one team working in one room designing and building one product in 100 days, all wasted effort must be avoided. For riskview.com we used PowerPoint to define the user interface as slides. We used the notes feature of PowerPoint to describe the dynamic behavior of each user screen.

In a lessons learned session after the project, we felt a need for more detailed specifications of the exact behavior of each button press or allowable user action. Effectively designers were asking for use cases to be written based upon the screen designs. Containing such an activity so that it does not become an endless activity is always difficult.

A crucial step in developing a good system design is to understand the results of each anticipated user action down to the smallest detail. A method for recording this is to use a “story board” approach similar to what the Disney studio might use to lay out a cartoon. Each action should then be reviewed by potential users and by content experts. We did not do this for riskview.com and it was a mistake. For example, while tax rate was an essential part of our calculations, nowhere in the original design was there a place to enter the tax rate! Carefully going through the screen mockups and checking the anticipated analytics would have brought this omission to light.



An example specification developed by the Product Manager using PowerPoint and its Notes feature. The similarity of this spec to the actual product is striking.

Open environments don't tolerate goofing off well. Goofing off has to be done elsewhere. Senior executives can be particularly distracting, especially to employees of the company. Their comments, attitudes, and anecdotes are listened to attentively -- sometimes impacting a very tight schedule.

A pizza and/or a beer can be a very good investment. Mostly it is a show that management cares about the project. Fridays are especially good days for a little "celebration".

6. Partner away risks

When doing a project to assist high net worth individuals and professional investors manage the risk of their international equity portfolio, it is prudent to think about managing the risk within the project as well. All too many projects never spend an hour identifying the risks in a project and taking a proactive approach to dealing with that risk. The idea of hedging project risks seems to rarely occur to even experienced project managers.

An active strategy in riskview.com was to think about the inherent risks and think of ways to reduce, transfer or eliminate those risks. An important risk that was identified early in the project was the operational risk of hosting a web site capable of supporting the number of users that we expect and of operating that site reliably on a 7 day a week, 24 hour a day basis.

A three way partnership of Infinity, IBM and Dow Jones.

Convincing IBM's Banking, Finance and Securities business unit to partner with us on this project eliminated this commercial and operational risk. It of course added some risk associated with dealing with such a large company on such a short schedule. Many a person told us, we could never get IBM to move as fast as required, including numerous IBMers. This project is now being used as an important counterexample within IBM. The company is different and is now capable of moving quickly for important strategic opportunities.

7. Design

A few bookcases could be filled with books written about software design. A few bookshelves have been written about object oriented software design. Most are written by people who have spent more time thinking about design methods, than actually building systems.

We did design for riskview.com. It was done very quickly and efficiently. No elaborate method or CASE tool was either used or seriously considered – we didn't have the time. We went to the supply cabinet, got a stack of large post-it notes and spent time in a small conference room.

The objective was to figure out the minimum number of classes required to support the defined functionality, as described in the user specification. We explicitly tried to find objects that we could "draw a picture of".

Some time was spent thinking about some rather different approaches to designing this application. In particular some approaches that anticipated a menagerie of future requirements were rejected in favor of satisfying the here and now requirements. We wanted a design that was simple and that could be built in the remaining 85 days.

7.1 Physical object model

After the design stabilized, we built a physical model of the inheritance tree of classes within the application. The model was built using Styrofoam balls, wooden dowels and file folder labels then hung from the ceiling between the testers and the developers.

Photo of object model.

This model may represent the least expensive CASE tool on the market – materials cost $20.35. It was more useful than any CASE tools I have ever used.

We used it in three important ways.

1. It fixed the number of classes that we would deliver in the finished application and we did not allow new ones to be added, unless existing ones could be removed.

2. It was a very useful way to publicly document which classes had been code reviewed (blue ribbons) and tested (green ribbons).

3. It helped everyone understand what was being built and how much time and effort it takes to do testing, documentation and code reviews.

We delivered 66 classes on day 100.

8. Implementation

Coding is real work no matter how you slice it. Java eliminates certain classes of errors, but many errors remain. In particular, errors resulting from a misunderstanding of user requirements can be quite persistent.

We also did not take the time to deal with data errors properly. The project architect had the idea early in the project to take the time to create a test database against which all development could be done. We did not accept this suggestion because the interface to the real database was in some sense 'working'.

Data problems were a constant menace. Many misunderstanding about analytics were masked by data errors far too long in the project. Equity data for the past 5 _ years from 29 countries will have errors. The application we built made it especially easy to detect errors that had persisted in the database for years.

9. Project Hygiene

The project hygiene techniques described below are very similar to those practiced at Microsoft and described in *Microsoft Secrets*.

Every line of code was configuration managed from day one.

At 6:00 AM the tester and a senior developer did a complete build of the system before other developers arrived between 7:00 and 8:00 AM. Usually one or more developers was still working at 10:00 PM. Every team member worked several hours in the office every weekend. The weekend before the launch, everyone was present for many hours, both days.

Beginning about three weeks into the project, all errors with requirements, functional specifications or code were recorded in a problem reporting system. This system was used throughout the project.

About a month before the final release, we began to have daily conference calls with everyone concerned. As the project peaked, there were as many as twenty people involved in these calls from New Jersey, Georgia, New York, Illinois, Colorado and California. During the final two weeks, these calls happened daily.

The attempt was to keep the call short, but we also wanted to solve small problems as quickly as possible. Complex problems were assigned to individuals or teams to resolve and report the solution the following day. The usual call took twenty to thirty minutes.

10. Test Plan

The tester for riskview.com was a very capable young man who had exactly the right training for testing a complex product – playing and building adventure games!

His first assignment was to think how to solve an admitted infinite problem in a short, finite time. The goal was to achieve the highest possible quality in the available time. He developed an excellent albeit aggressive test plan based upon the Quality chapter of *Object Lessons* (Love, 1993).

The test plan included considerations of code review, regression testing, unit testing of each class, user interface testing and performance testing. The plan anticipated the acquisition of powerful specialized testing tools.

Significant time was spent evaluating newly available tools for testing Java. They did not work. We had to resort to a far more manual testing regime than desired. Hopefully, by the time you read this the tools for testing Java programs will be mature enough to use. If not, start early and build some simple tools to at least allow regression testing on a class by class basis.

11. Management

High intensity projects require full time management. The more knowledge the manager the better. A requirement is that the project manager is capable of reading every line of code written by the team. If programmers can learn new languages, managers must do the same.

Once upon a time, a project manager asked how he could be expected to learn six programming languages, since that was the number being used on his project. He was told there were two choices – reduce the variety of languages on the project or learn a lot of languages.

Once it becomes obvious what partners can and can't do; accept the reality and don't waste too much time escalating problems to helpless executives. Never expect too much from your big, powerful partners.

Feature and function must be frozen early and not be allowed to change. New ideas should be recorded for subsequent development cycles. Creeping featurism is known to be the most likely source of slipped schedules in the software business. In fact, developers are more responsible for this than marketing or company executives. You cannot let this happen. Fixing the number of classes that you will deliver is one of the best ways to insure that that feature creep does not happen.

Through daily meeting, constant email and public posting of key project documents, everyone should know exactly what is expected of them each hour of each day. Management should constantly be wandering around talking individually to the developers to be sure that there are not stuck and need help.

There were no doors; communications were open; and everyone on the project freely discussed errors, stupid blunders, mistakes, and problems. This concept, egoless programming, first described in Jerry Weinberg's, *The Psychology of Computer Programming,* should be practiced on every project. By discussing and openly admitting errors, senior project members made it clear to the less experienced members that mistakes were acceptable, so long as they were "interesting new mistakes".

12. External, invariant commitment

A fact that contributed to the success of this project was that the delivery date was fixed and was not changeable. If we could have changed the date we would have. The date on which the announcement would be made and when we must do a demonstration was fixed from day one – July 23rd, exactly 100 days after the first new team member joined the team.

Challenging as it is, we recommend such an external, invariant commitment for delivery. It focuses the mind.

13. Marketing

The success of a product is only partially attributable to technical work. At least as important is the quality of the marketing activity. Young companies often fail to adequately invest in this activity. Some over invest in marketing inadequate products.

Sometimes a development team can be fortunate enough to have some unique "barrier to entry". More often the development team thinks they have uniqueness when all they actually have is lack of knowledge of competitive products.

With the Internet, competitive analysis has become a lot easier. Any development team that doesn't spend some time every day surfing the web to understand what the competition is doing is making a serious, possibly fatal mistake. One very good way to manage this activity is to have each member of the development team take responsibility for a particular competitor. Each week, each member of the

development team should send email to the rest of the team describing interesting announcements, analysts reports on the company, customer testimonials or interesting notes in a user chat group.

Ideally a development project should have a very competent person doing competitive analysis on a full time basis. At Infinity we had just such a person. We also had access to a marvelous data base developed by the Wall Street Journal which had never before been made available to the public – a major barrier to entry. The product manager at Infinity read widely, carefully monitored new developments on the Internet, and talked with academic, commercial and technology suppliers constantly. He also had the ability to define what needed to be built and the self control to keep that definition relatively constant during the project.

Product marketing involves defining the product, communicating its defining features and functions, identifying its differentiating characteristics, and working with the rest of marketing to do the organization, promotion and rollout of the product. Keeping all these people "on message" is hard and time consuming. If you are working with partners, it is even harder.

A press release looks so simple. But a good press release, involving multiple partners, can take weeks to prepare. On riskview.com we had four companies involved in the initial press release. We made the mistake of assuming that if the PR person within a partner company had approved a press release, that executives from that company had approved the release as well. Be sure that the most senior executive from each of the partner companies sees the final release before it is released. It is not enough to email it or fax it and assume that it is received. You must call and verify that it has been received, reviewed and agreed. Expect you will get last minute surprises.

The very best PR firm will seem barely adequate.

If you are building a high visibility project, budget professional public relations from the start. For short duration projects, PR activities need to start before coding starts. Significant time will be required "training" the PR representatives. More time will be spent deciding what the PR plan will be. Then the PR firm goes to work trying to interest reporters, schedule meetings, prepare for interviews, and do damage control after the interview. If PR includes TV or radio, the challenge is even greater. Providing video clips of the product in action can itself be quite time consuming and expensive. Dealing with executive egos to decide which exec goes on TV and which ones watch might take even longer.

New products are often introduced at trade shows. A really hot product should produce lots of traffic at the company's booth. It is critical that everyone "manning the booth" be adequately trained. This training should involve

- learning how to demonstrate the product quickly and professionally
- memorizing answers to "50 questions" – the most likely questions about the product, the company, the price and the competition
- knowing which questions should be referred to someone else in the booth or in the company
- learning how to spot an important customer and be sure that the right person in the booth is dealing with such a person
- knowing exactly how to get contact information with each person visiting the booth – taking business cards, filling out forms on the computer, or swiping a special card provided by the conference.

The real action in a tradeshow does not happen at the booth. It should be happening in the executive suite nearby. The booth crew has the job of finding important prospects, referring them to the executives in

the suite and scheduling appropriate appointments. Never underestimate the administrative support required to support this scheduling and meeting management.

Expect the unexpected. We had the remarkable experience of discovering that the fancy new monitors and the hot new computer supplied by IBM were not compatible at the screen resolution that we required. A frenetic set of phone calls allowed us to solve the problem exactly 5 minutes before the senior editor for the Wall Street Journal walked into the suite for a demonstration. The demo went flawlessly. The results were published the next day.

THE WALL STREET JOURNAL

MONEY & INVESTING

Internet Site Offers Help To Investors

Risk-Analysis Service on the Web

Article published in the Wall Street Journal on July 23, 1997 – day 100!

14. Celebrating

High performance teams work hard and should party hard. Plan and budget for a few parties in any 100 day project.

Find something to celebrate during the first month of a project. The quality of the party should improve as the importance of the deliverables increase. Parties don't have to be held in the evening or on the weekend. They can be interesting lunches, decompression walks, special pastries in the morning or special speakers.

Then find something to celebrate almost every week. Plan a serious party a week or so after the official release of the product. Invite spouses or significant others. Senior executives of the company must be present – it is even better if they are hosting the party in their home.

Hiring a professional photographer to photograph the team is important. We did not do that for this project. It was a mistake. We did take several team photographs and a little videotape, however.

For future projects, project scrapbooks should be started when the project starts. Let everyone contribute photographs and quips. It will be an important contribution to the company's culture as the product begins to build momentum in the marketplace and the profits start accumulating.

15. Summary

With an excellent team, a supportive management, and committed partners, amazing products can be produced in an incredibly short period of time. We look forward to duplicating this experience with other companies.[2]

[2] See www.mba.com for the second 100 day success story managed by the authors of this paper.