And then the phone rang…

J.P. Gunderson and L.F. Gunderson

Gamma Two, Inc.
1733 York St
Denver, CO USA 80206-1212
{jgunders, lgunders}@gamma-two.com

Abstract[1]

There are two critical interfaces that must be crossed when deploying any technological product. These are the hand-off from research to development and the hand-off from development to commercialization. These interfaces are frequently problematic, and are especially problematic when the technology being deployed is based on artificial intelligence.

This paper will present our analysis of the problems and lessons learned from the deployment of an artificial intelligence based financial application that was developed and commercialized over the last two years, and is currently being used to manage a private hedge fund. During the process we encountered several bumps in the road, and two, in particular, were due the fact that we were deploying an artificial intelligence based system.

At the research to development interface, we began to appreciate the difference between an experimental A.I., with its quirks and oddities; and a development tool that must be a reliable and testable product. During this transition we encountered the problem of being able to confirm that the software, after optimizations and modifications, was functionally equivalent to the original software. The necessitated the development of repeatable 'intelligence tests' that could be automated to confirm that no functional changes occurred.

At the development to commercialization interface we became painfully aware of just how messy the real world is, and the impact of this messiness on an adaptive intelligent system. Suddenly, we were called upon (by the paying clients) to improve the reliability, robustness, and capabilities of a high-reliability, real-time intelligent system. This necessitated the extension of the intelligence in the system to monitor its own performance and adapt to changing conditions in its environment – in order to get to the state where the originally designed intelligent system could begin its work. This is similar to the biological mechanisms of having one intelligent system - the brain, protected by a completely independent intelligence – the immune system.

Overall we learned several key lessons which we have incorporated into our development process; lessons which are designed to smooth the transitions along the path from a research project to a commercially deployed artificial intelligence application. Since our business depends on efficiently producing commercial intelligent applications, these lessons are key to our business survival. However, we believe that they are of value to any group that is developing intelligent software and hardware.

Background

In late 2002, we were approached by a small financial brokerage house. They had identified a unique market inefficiency that they knew they could exploit for their customers. However, the detecting the inefficiency required a trained observer to spot specific patterns in the price changes of stocks, and it was sufficiently rare that a human observer could not find sufficient opportunities to supply the fund.

We were asked the questions "Can a software application be developed that could detect the pattern?" and "Could it be efficient enough to reliably supply a large number of opportunities on a daily basis?" If we could answer 'yes' to both these questions, the brokerage house would be interested in being the end user of the system, once it was deployed.

Over the course of the next two years, we answered both of these questions, and learned what we think are significant lessons about the process of research, development, and deployment of artificial intelligence systems into the 'real world.' Nils Nilsson, in a recent AAAI article (Nilsson 2005) suggests that the true calling of artificial intelligence research should be to deploy systems into work environments.

One of the most significant issues we uncovered was that each of the three phases (Research, Development, and Deployment) required its own set of assumptions, and that the transition from one phase to the next involved removing the dependencies on the assumptions needed for the early phase, and instituting assumptions needed by the later phase. While this process is not unique to deploying artificial intelligence software, we found that there was a synergistic effect that significant increased the problems and that deployed artificial intelligence solutions had additional hurdles to overcome. In the end, we developed one AI system to 'do the job', and developed a second AI system to allow the first system to work in the 'real world.'

The Problem

In this case study, the brokers were able to look at specific graphs of stock performance and from the shapes of the curves (See Figure 1) over time, determine specific investment opportunities. Unfortunately, only a limited number of the brokers could reliably see the opportunities, and any given broker could only examine a small number of graphs on a given day.

Our task was to:

1. See if it was possible to encode the visual cues used by the brokers into a pattern recognition problem that could be analyzed by computer; and
2. Do the pattern recognition quickly and reliably so as to provide a reasonable number of investment opportunities each trading day.

To complicate the problem, the frequency of the key opportunities is quite low (on the order of 1 in 500,000), and the critical data does not become available until fairly late in the day. But the list of target opportunities was needed before market open on the following day. This added a critical upper bound on the execution time of the pattern recognition algorithm, provided we could develop it.

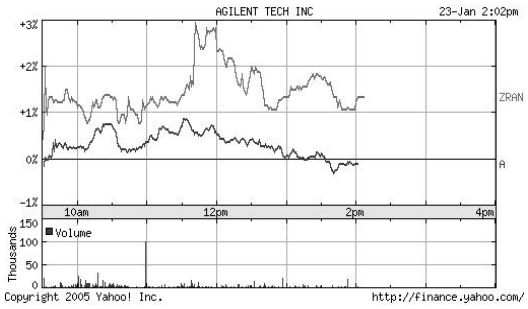

Figure 1 - A sample intra-day price chart for two stocks (Yahoo finance 2006).

Phase I - Research

During the initial research phase, it is common to make some simplifying design decisions. These may be a necessary to effectively limit the solution space to one that can be searched, or it may be required by the nature of the problem (Baldwin and Clark, 2000).

Typically the requirements used during the research space include:

1. The environment in which the search occurs is known;
2. The environment in which the search occurs is stable;
3. Non-essential variability is minimized; and
4. A satisficing solution is being sought.

These working rules allow researchers to focus on the core aspects of the problem and determine if a solution is possible, before investing in issues such as performance, optimality, robustness, and computational burden.

The impact of the environment in which problem solving is to occur has been well documented. Newell and Simon devote much of their analysis of problem solving to the inter-relationship of the problem and the environment (Newell and Simon, 1972).

Given the interactive nature of the problem and environment, the traditional first step is to control the environment so that all affective aspects are known. These will be the laboratory conditions under which the research will be done.

The financial analysts work in a very complex and volatile environment. During the research phase, it is necessary to have a stable environment so that comparisons can be made between one potential solution and another. Comparing the results of two runs when the underlying hardware has been swapped out between the runs doesn't tell the researcher much about the software changes.

During the research phase the key question typically takes one of two forms. Either the researchers are asking "Can this be done at all?" or the question is "Can this be done in a time effective manner?" The former is the case when one is investigating an 'open problem,' the latter obtains when there is a known solution, but the execution time for any but toy problems may exceed the heat death of the universe.

One traditional approach to research involves controlling the changes in the variables, to assess the effect of each independent variable on the outcome.

As discussed above, during the research phase, the primary question is on of can it be done, either at all, or in a time effective manner. However, the focus is not usually on optimality, maximum efficiency, or reliability.

Case Study

We worked with the brokers on a daily basis to determine what they were looking for in the graphs, and then attempted to develop metrics that captured the key aspects as numeric transforms on the raw stock data. The process elicitation phase took approximately 3 months, by which time we were reliably matching their expert's opinion of the patterns. We had met condition 1; we could detect the patterns from the available data streams. By August of 2003, we had a functioning prototype, and we could begin the next phase of the journey.

Interface I

Up to this point there have been no significant differences between the research into any open problem, and a problem that involves an artificial intelligence solution. However, as we move into Development, changes in the underlying

criteria set begin to impact the artificial intelligence aspects of the solution.

During the Research phase, we simply needed to see if the problem could be solved, and we produced a prototype that functioned under 'laboratory conditions.' In addition it needed to meet the real time deadlines of a production system.

During development we need to produce a fully functional, reliable system that has all the bells and whistles needed by the customer. We were perfectly happy staring a black screens with columns of number scrolling past, launched by painstaking manual entry of dozens of command line parameters, with re-compiles required to change some of the more complex settings. Somehow the customer had a different picture in mind.

Phase II – Development

In the development phase, we needed to re-visit the criteria that were selected, and adjust them for the next phase. There were three key changes to the working criteria: first we could no longer artificially minimize the variability; second, the system now had to meet real time deadlines. Finally, the kind of hand cleaning of data sources, and manual data acquisition had to be replaced by automated data preprocessing.

This gave us the following requirements (or design rules in the Baldwin and Clarke model) for the development phase:

1. The environment is known;
2. The environment is stable;
3. A satisficing solution is being sought;
4. The system must run in an autonomous, hands-free manner as much as possible; and
5. The system must produce its output to meet real time deadlines.

The first change was the relaxation of the controlled variability criterion. During research, the focus is on producing a solution that works under ideal, repeatable conditions. Moving to development normally means that the system must work in more 'real world' conditions. This would necessitate the system being far more robust to the normal changes in the environment, and to the types of natural errors that occur in real world data.

Criterion 4 addressed a common change from a research environment to a development environment: that more of the system must be automated. During the research phase, it is common to work repeatedly with a canonical data set, which has been cleaned, formatted, and possibly, pre-processed in other ways. One of the major tasks in development is to produce automated systems that will provide the cleaned, formatted, and pre-processed data in an ongoing operation.

Finally, with the addition of the real time requirement, we addressed the second question originally proposed: "Can we produce enough opportunities, in a timely manner?" This is compounded by the criterion 4, since there is added computational burden associated with the data pre-processing stage.

Case Study – Development

The changes in the design rules led to major re-writing of the entire package. Since the system needed to acquire its own data after the close of market on each trading day, an entire system for downloading and updating the data set was built. Surprisingly, much of the financial data from the three major US stock exchanges is hand processed. This is due, in part, to the historical development of the markets and how there information is structured. The data is designed to be used and analyzed by humans, and not by computers. One of the more interesting issues was that there is no 'key' (in the computer science sense) for a given stock. The ticker symbols are changed to reflect short term conditions and the names change. As a result, the same stock may be referred to by three or four different symbols over the course of a month or two of trading, and the same ticker may refer to two different stocks over the course of a year or two. There is a large business niche consisting of nothing but providing clean computer processable trading data for stock analysis programs, most of which are maintained by groups of people in front of computers, solving the 'correspondence problem' several hundred times a night. We chose to purchase clean data from one such company, and integrate their data files into our system, their people would hand clean the data and have it ready for download by 5 hours after market close.

Once we had achieved 'clean' data, we had to address the real time needs of the system. The prototype that came out of the research phase had a highly accurate pattern matching engine, which required a few tens of milliseconds to correctly detect a target opportunity. Unfortunately, we needed to scan through tens of millions of candidates to find a hundred or so targets. This gave us a system that would take about 200 hours to process a daily report. Since the clients needed to have the reports by market open, and we did not have the data available until 17 hours before market open, we needed to reduce our processing time by a factor of 10 to 15.

This was accomplished by a combination of code optimization (replacing data structures chosen for ease of use with structures chosen for speed of access), and by the introduction of pre-filters to the pattern matching engine. By October of 2004, we had a system that was reliably delivering the targets approximately 2 hours before market open.

To address the concern that the pattern matching engine was too fragile, we ran a back test on the system. In this case we loaded all the stock market data for the last three years (beginning in October of 2001, and ending in September of 2004). The system then went through the study period day by day, and determined which targets would have been opened, and which would have been closed. As a side note, this required building a new engine that detected closing conditions on the targets. At this point we had built a very specialized emulation of a stock broker, which would tell you what to buy, when to buy, and when to sell. Over the three year back test, the emulated broker

significantly out-performed the markets, and most of the large mutual funds. After the back test was provided to the brokers, they felt confident that they could use the system to manage their own fund profitably, and they requested a deployable version of the emulated broker system.

Interface II

Theoretically, the transition from development to Deployment was simply a case of bundling up the system, installing it onto the client's machine, and doing a little monitoring to make sure everything was running smoothly. During development, we had instituted complete, automated regression testing and both the unit and systems level. We had correctly specified the hardware and network environments, and had live tested the complete package for 90 days. We knew that it met all of the design criteria we had detailed above, it was fast enough, it was accurate, and it was reliable, under the development assumptions.

Phase III – Deployment

In analyzing the deployment environment, there were very few differences from the development environment. Admittedly, we had no control over the deployment environment, since we were releasing our system into the outside world, but we had done extensive testing over the last six months, so we had a good model of the world. All we needed to do was sit back, and do a little monitoring…

Unique aspects of Artificial Intelligence Applications

So far the process we have described is fundamentally the same regardless of the type of product. However, we believe that there are aspects that are more influential when developing software systems in general and artificial intelligence applications in particular. One of the distinguishing characteristics of an artificial intelligence system is that is supposed to do something intelligent. In our case, it is doing something that people do, but it is doing it faster and more thoroughly. In fact, it is doing something that not all people can do. It is also doing something that people disagree about. In the case of our pattern analysis, one expert might say that X is an example, but her colleague may disagree. This leads to a unique aspect of deploying artificial intelligence applications: trust.

If a researcher has developed an algorithm to calculate a solution to a well defined, well known, problem it is only a matter of demonstrating that the algorithm produced the correct result. When the problem is in the realm of a judgment call, or if the solution space is so large that the 'correct' solution is not well known, demonstrating correctness is more problematic. These are exactly the types of problems that require artificial intelligence solutions. In these cases, it is usually impossible to 'prove' that the results are correct; rather the system must be 'trusted' to produce the right answer.

There have been a number of conferences and symposia that have addressed both the issue of trust and the wider social issues associated with allowing artificial intelligence applications to make decisions and execute those decisions in the real world (see Kortenkamp and Freed, 2003, Gunderson and Martin, 2004, Shapiro *et al*, 2005 for a selection of papers). During the research and development phases of the process, the trust issue is not significant, however, once the system is deployed, it becomes paramount. We had just released an artificial intelligence application that was going to make recommendations for where and when to invest people's money. We had built up some level of trust during the research phase, when each decision made by the system was analyzed by the brokers. We built up more trust during the final 90 day live testing, when the brokers received a report every morning, and looked over the targets presented by the system. But now it was going live.

The requirements we had for the deployed systems were

1. The environment is known;
2. The environment is stable;
3. A satisficing solution is being sought;
4. The system must be reliable, and not depend on operator input;
5. The system must produce its output to meet real time deadlines;
6. The system must be robust; and
7. The system must be trustworthy.

And then the phone rang…

The world is not a kind place. Some of the events that the system had to deal with included:

- The phone rang when we were out of town (of course) at a conference. The report had not been delivered that morning. The operating system had been set up to automatically update itself at 03:00. This update required a reboot, in the middle of the run. We turned off the automatic updates.
- The report was missing half of the data from the previous days trading. The company providing the cleaned data had a staffing shortage, and did not get their files uploaded until three hours before market open. We added a monitor system to validate data set size.
- After a week of incomplete data, we switched to a direct download system. The monitor program was improved to detect failures and switch to the alternate sources.
- The format of the download site was changed, requiring a recoding of the download system. In addition, the monitor program was given the

ability to text message for help if things went unrecoverably wrong.

- The DSL modem had a hardware failure, and the monitor program was unable to email a text message for help. We installed a second, redundant system using a separate power substation, and a different ISP. The secondary monitor program designed to detect failure of the first system and deliver duplicate report.

At the end of the first year of deployment, the original pattern matching code represented less than one third of the code base, and the monitor represented the other two thirds. We had designed and developed an intelligent wrapper for our intelligent system, whose sole job was to deal with the complexities of a dynamic, uncertain environment and achieve the goal of keeping the core system happy.

And then the phone rang again….

In August of 2005, the brokers called up to say that they weren't happy with the system. It was running reliably, within the real time deadlines, but they didn't trust it any more. It wasn't giving them good target opportunities, or they said, maybe they weren't using them properly. In any event, in spite of the 3 year back test that indicated the system would provide a significant return on investment, they were losing money. They no longer trusted the system.
We re-ran the back testing (Note: due to the significant size of the data set, it was necessary to do statistical sampling of the historical data) and the results were consistent with the previous runs. We then ran the back testing on the previous 12 months of data, and discovered that the deployment was based on an invalid assumption: the environment was not stable. Beginning in August of 2004, the underlying distributions of the stock data underwent a statistically significant change. As a result, the emulated broker was no longer in tune with the realities of the daily trading.
In addition, we found that the brokers had depended on the stability of the environment. Therefore, we had coded this dependence into the system. We needed to reanalyze the requirements of both the clients and the code. So, after another month of meetings, we recoded the system to reflect the changed environment. We also instituted a regular meeting to gauge the 'trust-level' of the brokers. The next step is to expand the monitor program to give it a hypothesis checking capability, in effect allowing the system itself to ask "How confident am I that the environment is within my design parameters?"

What is intelligence anyway?

One of the defining characteristics of an intelligent system, according to many researchers, is the ability to learn and adapt to changes in its environment. If the environment is not stable, the system must be capable of detecting changes in the environment, and if it intelligent, it must be capable of adapting to those changed to achieve its goals. The design criterion reflecting a stable deployment environment was flawed. Since we were deploying an intelligent system, that assumption impeded the design and development to meet a key component of our system.

Conclusions

In trying to assess "what went wrong and why," it is important to remember that some things go wrong due to procedural or structural problems, while others are due to a bad roll of the dice. Separating these can be difficult and in some cases impossible. We will try to err on the side of caution. We have identified five guidelines that we are taking out of this experience, and will use in the next deployed intelligent system project.

Different design rules are in effect at different stages
During the three stages of Research, Development, and Deployment, it is important to make different assumptions about the requirements. In part, these assumptions facilitate solving the unique problems encountered at each stage. However, during the transition from one stage to another, these design rules must be amended for the new stage. Failure to use the correct design rules will result in a stage deliverable that does not work in its intended environment.

Several design rules have direct impact on Artificial Intelligence applications
Artificial intelligence solutions are typically applied to problems only when there is no effective traditional algorithmic answer. Yes, it is always possible to deploy a complex probabilistic planning algorithm to find the shortest path through a graph, but it is rarely needed, efficient, or the most effective solution. Frequently, the problem to be solved has no 'well known' answer, and the first phase must stringently restrict the working environment to enable the finding of a solution. However, the final deployed solution cannot rely on this artificial environment, or it will fail in the real world.

Managing these assumptions materially affects the success of the final deployment
It is not simply that design rules get relaxed as the project moves through the phases. The needs of the deployed system are neither a subset nor a superset of the development design rules, the same holds for the relationship between the development rules and the research rules. Ignoring these transitions results in either a) a restriction that does not apply in the deployment environment is needed by the deployed system, or b) a restriction that does apply in the deployed environment is ignored by the system. In either case, the deployed artificial intelligence system will not be well engineered for its job.

Artificial intelligence applications need to be trustable
If the problem has a straight forward, well defined algorithmic solution, it probably does not require the use of artificial intelligence solutions. The corollary of this is that there can always be disagreement about whether the system came up with the correct solution, or came up with a

solution in the correct way. In evaluating the 'performance' of intelligent systems (rats) in 1939, Egon Brunswik summed it up with:

> "In the natural environment of a living being, cues, means or pathways to a goal are usually neither absolutely reliable nor absolutely wrong." (Brunswick 1939)

When such systems are deployed into the working world, it is likely that someone's performance evaluation, job, or life may depend on the functioning of the intelligent system. Since the system is providing a judgment call about the problem, there may be no way to prove the correctness of the intelligent system. The people relying on the system have to trust it, and trust is very fragile. During the research, development, and deployment of the intelligent system, a critical goal is to enable the eventual clients to develop trust in the system. Failure to do so can easily result in a perfectly functional intelligent system that simply does not get used – a failed system.

The world is not a stable place

During the first two stages of the process, it is necessary to limit the variability of the environment compared to the 'real world' simply to solve the problem. This causes a direct conflict with the requirements of the final stage because the real world environment is by definition not so limited. There appear to be two main approaches to addressing this conflict. First, in a more traditional waterfall model, detail in advance every possible thing that can affect the performance, and design the system to handle each case and all combinations. The second approach is to iteratively develop a more and more complex system that can handle more and more of the variability. The problem with the first approach is that it may be impossible to come up with the complete catalog of failures, and the process may significantly impact the cost, performance, and development time of the system. This suggests that during the development stage, the system should be made increasingly tolerant to the irreducible uncertainty of the deployment environment. This can be thought of building an immune system by exposing the intelligent system to the outside world a little at a time. Regardless of how it is done, the very 'simplifying assumptions' that made the intelligent system possible will have to be undone to make it deployable.

More AI may be needed to deal with the world than is needed to solve the core problem

The analogy of an immune system is another key aspect of achieving a successful deployable system. Several researchers have argued that living organisms have two different brains (Coutinho, 1994); one is the central nervous system, the 'seat of intelligence.' The second is its immune system. Recently, several artificial intelligence researchers have begun investigating immunocomputing (See Tarakanov, *et al* 2003 for example) as a tool for developing intelligent systems.

Dealing with the variability of the real world is a challenge for all intelligent systems deployed into the outside world. In our development, we found the need to pre-process the outside world and handle the complex changes and failure modes that were discovered over almost two years of deployment. These infrequent events cannot be discovered or even planned for during the development stage, rather the system must be designed to discover them during the deployment. We have had to build a second brain that is much larger and more complex to detect, analyze, and respond to the vagaries of a real world deployment of an otherwise fragile artificial intelligence. Perhaps the most valuable lesson learned from this project was that solving the original 'hard problem' was child's play compared to building a reliable, robust, and trustable intelligent system that has to go out and work in the real world. For now, we are waiting for the phone to ring again.

References

Baldwin, C. Y. and Clark, K. B., *Design Rules, Volume I The Power of Modularity*, MIT Press, Cambridge MA 2000. Ch. 2.

Brunswik, E. 1939. Probability as a determiner of rat behavior, "Journal of Experimental Psychology, Vol. 25, pp 175-197.

Coutino, A., Immunology: the heritage of the past, *Letters of the Institute Pasteur of Paris*, Vol. 8, 1994, pp 26-29.

Gunderson, J. P. and Martin, C., eds. 2004. Interaction between Humans and Autonomous Systems over Extended Operation. AAAI Technical Report SS-04-03.

Kortenkamp, D. and Freed, M. eds. 2003. Human Interaction with Autonomous Systems in Complex Environments, AAAI Technical Report SS-03-04.

Newell, A. and Simon, H. A., *Human Problem Solving*, Prentice Hall, Englewood Cliffs, NJ, 1972, Ch. 3.

Nilsson, N. J., 2005. "Human-Level Artificial Intelligence? Be Serious!, *AI Magazine*, Vol. 26, Num. 4, pp. 68-75.

Shapiro, D., Berry, P., Gersh, J., and Schurr, N. eds., 2005. Persistent Assistants: Living and Working with AI. AAAI Technical Report SS-05-05.

Tarakanov, A. O., Skormin, V. A. and Sokolova, S. P., *Immunocomputing Principles and Applications* Springer-Verlag, New York, 2003.

Yahoo Finance, 2006. Image captured on 01/23/2005 from *http://finance.yahoo.com/q/bc?t=1d&s=A&l=on&z=m&q=l&c=ZRAN*.