

Enterprise Infrastructure Monitoring
using SL-GMS Enterprise RTView[®]

A Build vs. Buy Analysis

*SL White Paper
June 2003*



Dynamic Graphics for Real-Time Data Display

Copyright

Copyright © 2003 Sherrill Lubinski Corporation. All Rights Reserved
June 3, 2003

Trademarks

SL, SL-GMS, GMS, SL Corporation and the SL logo are trademarks or registered trademarks of Sherrill-Lubinski Corporation in the United States and other countries.

TIBCO™, Hewlett Packard Open View, Tivoli NetView, and Computer Associates UniCenter are registered trademarks of their respective companies in the United States and other countries. They are mentioned in this document for identification purposes only.

Table of Contents

Abstract.....	1
Enterprise Infrastructure Monitoring.....	1
Business Value	2
The Build versus Buy Question.....	3
SL-GMS Enterprise RTView	3
Build vs Buy, the Tradeoffs.....	4
1) The Cost of Development Resources.....	4
2) Total Time for Implementation.....	5
3) Quality of the Resulting Solution	7
4) Performance of the Resulting Solution.....	7
5) Protection of Investment for the Future	8
A Build It Yourself Scenario	9
A Buy and Build Scenario	12
Conclusion.....	13

Abstract

In today's large scale business IT departments, a new class of problem has emerged, which is not solved by traditional infrastructure management products. The need to monitor and control complex integrated applications and huge data flows is increasingly important as dependencies on the systems grow. The IT department can either build all of the tools it needs, or make use of a product such as **SL-GMS Enterprise RTView™** as part of the solution. The critical decision to build versus buy a third party product is examined here.

Enterprise Infrastructure Monitoring

Most large business operations today are dependent on a complex hardware and software infrastructure, often involving custom applications developed specifically for that business. Such systems are built on top of a range of disparate hardware systems, off-the-shelf application software, and middleware used to integrate the applications. The daily operation and management of these systems presents numerous challenges to the IT personnel tasked with maintaining their smooth operation and responding to problems that arise.

Many excellent solutions already exist for managing the hardware and operating system software at the base of these large business operations. Hewlett-Packard's OpenView, Tivoli's NetView, and CA UniCenter, among others, have been in use for many years and do this job well. Hardware failures, operating system crashes, and other system errors can be trapped and corrected quickly using such systems.

However, in recent years, business solutions have increasingly been built to include sophisticated applications, such as database, ERP, and CRM systems. A whole new category of solution provider, the EAI (Enterprise Applications Integration) vendor, has emerged providing "middleware" software used to integrate these independent components. Today, many large business systems make extensive use of "messaging" to communicate large amounts of data between different applications to get the job done.

The increasing complexity of such systems has produced a new class of problem. With huge volumes of data being passed between applications, bottlenecks can occur as one application cannot keep up with another. Often, the relationships between the applications can be quite complex, so it is difficult for anyone to figure out what went wrong when the system crashes. Fingers are pointed, and IT people spend hours poring over lengthy reports and log files to try to determine the source of the problem.

Compounding the problem is the fact that the operation of a large system may involve numerous products each with its own console and reporting mechanism. One customer complained that he had 16 different dashboards, one for each of the systems he was using. Management often would like to have a centralized view of operations so that they can go to a single location in order to view a summary of all business operation health and status information, such as throughput and volume statistics, and to drill down to more detail if needed.

Existing infrastructure management tools may be good, but are not enough. Something must be done to provide for the monitoring and control of the processes that make up a complex business solution. This expanded problem can be referred to as Enterprise Infrastructure Monitoring.

Business Value

Enterprise Infrastructure Monitoring becomes increasingly important as a business becomes more dependent on the smooth operation of its internal IT systems.

The scope of this problem and its effect on the profitability of an organization can be enormous, as failures or bottlenecks can have immediate and costly implications. Inability to respond to customers in a timely manner not only reduces profits, but can also result in loss of customers to competitors. It is obvious that something must be done to mitigate this vulnerability.

The middleware that is often used to integrate applications sometimes comes with tools to allow the user to see what is going on in the system. However, the quality of the information is often poor and limited. The vendors of these systems specialize in the function of their applications, the middleware, not in providing user-friendly graphical interfaces. The lack of visibility into the behavior of this critical part of one's business operation is a common source of problems.

Similarly, the business applications that are developed internally by IT departments are designed to solve specific business problems. High level monitoring and control of these applications is often an afterthought. The IT expertise applied to these problems typically involves processing data inputs or fulfillment of purchase orders, for example. It is only after the system is in production and large volumes are being processed that weaknesses begin to appear. One company discovered that a simple network outage due to overloaded queues cost them \$600,000 in lost revenue because customer trade orders could not be fulfilled in a timely manner. Had they been aware of the increasing queue depths ahead of time, the problem could have been avoided.

What will you do if you do not invest the resources to build the necessary application process monitoring tools? You will probably do things exactly as they are done now, on edge at all times, waiting for the next mysterious system failure to occur and then scrambling to try to determine the source of the problem, while revenue is lost minute by minute. After hours or days go by, the problem is finally solved, and you go back to business as usual, until the next problem shows up.

There is no question that a need exists for tools that help to visualize the relationships between various application processes along with analyzing the flow of data between components. How could it not be important to your business to improve the efficiency of these systems and minimize their downtime?. Your competitors are certainly doing it. But what exactly should be done?

Once the decision is made to address this problem once and for all, IT departments in large organizations face the daunting task of trying to determine how to provide the necessary solutions. The next step is to determine whether to build something internally or to purchase a product like SL-GMS Enterprise RTView (SL Corporation's Graphical Modeling System for Enterprise Real Time View) ... or both.

The Build versus Buy Question

Often, the decision about how to proceed with the development of a large scale enterprise monitoring system is characterized as a "build versus buy" decision.

The applications involved in a large business solution are customized, developed in house to deal with the specific details of that business. It is unlikely that a third-party product will be found that provides a complete solution to the many problems that accompany these custom applications.

Therefore, the decision is better described as one of "build it all" vs "buy some of it, and build the rest."

Specifically, you can put to good use a product such as SL-GMS Enterprise RTView (ERTV). While you may still employ many people in the development process, instead of starting at square one you could be starting at square seven, and they will be well on the way to completing all of the required functionality.

The question is not whether to buy something from a vendor or to build it yourself. This is an oversimplification of the issue. Rather the question becomes ... how much do you buy from a vendor and how much do you build yourself? This is a much more practical way to view the problem, recognizing that no one solution exists, yet there may be products available that can greatly facilitate the development of the necessary system for your business.

There are no products out there that do exactly this. Nor does ERTV purport to do this. Because of the custom nature of almost all application software and integrated middleware solutions, there is simply no solution other than to employ the services of one's IT department to build or assemble custom monitoring systems to oversee the health and functioning of all the custom applications developed over many years.

SL-GMS Enterprise RTView

SL Corporation has a 20-year history of providing graphical user interface and data visualization software for mission-critical applications in process control, network management, traffic operations, and aerospace.

SL-GMS Enterprise RTView provides a rich set of graphical components, along with a framework that includes data access and manipulation features that enable users to create customized dashboards for the visualization of business operation data with no programming

required. ERTV has evolved over the years and is used in many applications in process control, network management, power plant controls systems, traffic management systems and others.

The purpose of this paper, however, is not to present a technical description of SL-GMS Enterprise RTView. Rather, it is to examine the tradeoffs involved in buying a starting point monitoring solution versus building it all yourself.

SL products are specifically designed to provide general solutions, rather than solutions specific to a given vendor. While ERTV has been adapted well to the TIBCO environment, it also is designed to attach to and operate with data from other sources as well. Any infrastructure monitoring solution from a specific middleware vendor is going to have both the weaknesses associated with this function not being their area of expertise, and also the fact that it will be tied only to their system.

Build vs Buy, the Tradeoffs

A business decision to build something yourself versus buy from a third party vendor cannot be viewed as black or white. As discussed above, something must be done ... building it or buying it will result in some improvement in the situation. The question is, which of these will result in the best improvement?

There is a big difference between deploying your developers to build a large monitoring system from scratch, versus deploying them to build atop an existing, feature-rich product, like SL-GMS Enterprise RTView.

Numerous tradeoffs exist in making this decision. Keeping in mind that we are characterizing this decision as “build all of it yourself” vs. “buy some from a third party, and build the rest”, we will examine each of the following tradeoffs :

- 1) The Cost of Development Resources
- 2) Total Time for Implementation
- 3) Quality of the Resulting Solution
- 4) Performance of the Resulting Solution
- 5) Protection of Investment for the Future

1) The Cost of Development Resources

It would be overly simplistic to say that the cost of the third party product is to be weighed against the cost of the personnel that would be deployed to develop the equivalent solution. Sometimes the costs may appear to be similar, but there are a number of other important considerations:

Build:

IT employees assigned to develop an enterprise infrastructure monitoring system from scratch are starting at “square one.” Developing even the simplest of graphical interfaces for the front end often requires research and development of new skills.

These employees may be very good at what they currently do, but they are probably not familiar with all the issues associated with building a sophisticated monitoring system. They may be able to do it eventually, but there is inefficiency inherent in this approach. The total cost in this situation is not only in the employees' time, but also the training in lower-level development skills that would not be necessary if they were building atop a product like ERTV.

Sometimes, especially during slow economic times, a company will attribute a lower cost to deployment of their own employees on an internal development project. Rationalizing that they are "already paying them", the company assumes that the cost is zero. However, even in slow times, it is rare to find a situation where these employees are truly "doing nothing." If their efforts are applied to the development of a new monitoring system there is certainly a cost associated with it.

The total cost of building the solution involves not only the employees' time, but also the intangible cost of the inefficiency of using developers without the necessary skills on such a project.

Buy:

Of course, there is a cost associated with purchasing a third party product and learning to use it. There is also a cost associated with the accompanying higher-level development, although it is likely to be far less than that of developing the entire system.

Employees assigned to working with SL-GMS Enterprise RTView will be starting at a much higher level since many of the components necessary for the solution are already in place. With proper training, in a short amount of time they are typically able to produce powerful and effective solutions for the monitoring problems. In addition, the expertise of these employees will be used more effectively as it is applied at a higher level directly to the problems that the monitoring system must solve. They have the knowledge to effectively make use of a product like ERTV.

2) Total Time for Implementation

It is obvious that the implementation of a monitoring system built on top of a solid foundation like SL-GMS Enterprise RTView will take much less time than one built from scratch. However, there are other, even more important, aspects to consider about the development schedule:

Build:

Since an internal do-it-yourself development project begins at square one, it can often take several months before even the simplest features become available. At that point, the real work begins, as more complex features can take many months to develop. Even after a year, the system may have no more capability than what would have been obtained out of the box using the ERTV product.

Developers, even good ones, are notorious for underestimating the time that it will take to implement a new project, particularly in an area with which they are not so familiar.

This is due not to a shortcoming in their programming skills, but rather a failure to anticipate the degree to which unexpected problems and surprises will delay their efforts. It can often take days or weeks to finally uncover why that one object causes the Netscape browser to crash, for example. Such unexpected problems can result in huge delays in the project.

Often, the people called on to work on internal development projects are some of the best developers in the group. While this may seem like a good thing, the fact that they are so good means that others in the group will constantly be calling on them for help in resolving some emergency situation that arises. While these developers may be officially tasked on the new project, the fact is they will undoubtedly be interrupted many times, sometimes for lengthy periods, as duty calls. This is another situation that can wreak havoc on the schedule for development of the monitoring system.

All the while this development is going on and delays are mounting, there continue to be application process failures and bottlenecks causing untold cost to the organization, as the old way of doing things is perpetuated and reinforced.

Buy:

Typically, within one month of installation, there can be seen an immediate and measurable benefit to the company in using SL-GMS Enterprise RTView. Since the product is fully tested and has been deployed in many different environments, it is likely that smooth installation will be followed by a period of rapid progress in development of useful monitoring displays and applications.

By starting with a highly advanced base of features, including a graphics display builder, built-in data access methods, and a powerful and flexible historian, developers take far less time to produce effective systems. Even with the inevitable day to day interruptions, they are able to construct and deploy applications quickly and easily. Surprises are minimized, as most common issues have already been addressed by SL Corporation, given its long experience in building such systems.

As the IT department becomes more adept at using the product, the solutions produced become even more effective and useful. Developers are spending their time advancing the solution they built in the early stages, far outpacing the benefits that might come from developing it oneself.

Instead of taking a year to re-invent functionality that already exists in the SL products, the IT group is quickly producing advanced solutions based on a proven solid foundation. This is a result that is vastly more valuable to the organization. Significant benefits in reduction of downtime and improvement in overall system performance are seen in a very short time.

3) Quality of the Resulting Solution

The quality of the solution produced solely by internal development can be reasonably good. However, there are some important issues to consider:

Build:

The fact is that internal developers will probably be “reinventing the wheel” by developing a basic application monitoring capability from scratch. As part of the project, it will be necessary to develop a basic foundation for managing multiple dynamic graphics displays, handling the interface to data variables to drive these displays, and important ancillary functions, like historical data archival and retrieval.

While none of these things are terribly complicated on the surface, it is common knowledge that the first implementation of anything is nowhere near as good as the second and third time you do the same thing. Thus, the implementation made by internal developers is not likely to be as good as one developed by a third party with a great deal of experience in the area and multiple revisions of its product.

Additionally, some of the best features in a product like that provided by SL Corporation come about through repeated exposure to many different customer use cases. This exposure can only occur over a long period of time. It is unlikely that within a single company environment, there will be enough exposure and development response to result in a feature set nearly as rich as that provided by SL.

Buy:

A solution produced using the SL-GMS Enterprise RTView software as a foundation is likely to be much higher quality by many measures.

SL Corporation has had over 20 years of experience building high performance graphical systems for monitoring and control of sophisticated applications. During this time, techniques have been developed and honed over many revisions of the product to manage large numbers of graphics objects and data references, to filter data in various ways, and to perform functions on raw data. Exposure to many different types of problems has resulted in a very rich feature set, designed to graphically visualize data in many different ways. Even long time competitors of SL cannot match the richness of functionality available, let alone a small internal group developing from square one.

4) Performance of the Resulting Solution

While the internal IT group may be capable of developing a reasonably good process monitoring application, there are a number of areas related to performance of the resulting application that need to be considered:

Build:

Typically, an internal build-it-yourself development project starts out simple. In the zeal to demonstrate something working, a quick implementation is often made, using the most obvious data structures. String compares are often done to find named objects,

large tables of data are retained in memory to simplify coding, object replication is ignored, and memory leakage and growth is not considered a problem in the early stages.

It is only after the system starts to be used on a larger scale that significant performance issues emerge. At this point, so much code may have been written using the simpler data structures that it becomes impossible to change without significant effort. Ugly patches and workarounds get applied, which ultimately do not work very well and result in a maintenance nightmare.

The result seen in most cases is a system that performs slowly, gobbles up memory, and often crashes. In time, it becomes very difficult to add new features. Overall, the monitoring system itself becomes a burden and may never actually be used.

Many man months can be spent correcting the results of designs that were well intended, often by very good programmers. Usually, these people have a great deal of expertise in specific business domains, but are asked to develop applications where they do not have the experience. Costs grow rapidly in correcting problems introduced by this approach.

Buy:

In contrast, a monitoring solution built on top of SL-GMS Enterprise RTView is likely to perform significantly better.

The algorithms and data structures used in SL-GMS applications have been improved over many versions of the product. They have been specifically designed to manage large numbers of objects and data variables and are optimized to provide best possible performance. Memory usage has been extensively analyzed to avoid leakage and minimize consumption.

In the Java environment, a technique involving automatic source code generation is used to obtain the best possible rendering performance. An ERTV based application can easily be deployed as a stand-alone application or as an applet running in any number of standard browsers, like IE or Netscape.

What all this means for the IT department is that little or no time needs to be spent reworking code in order to achieve good performance. It can be assumed that the best possible performance is already available by building on top of the existing ERTV system.

5) Protection of Investment for the Future

Every few years, there occurs some significant change in the computing environment. This is one of the issues most overlooked when considering how to develop an internal application process monitoring system.

Build:

The internal development group is often consumed by the demands placed on it for designing and implementing the monitoring system in the first place. It is typically not an area of expertise, so there is a need to do research and develop new skills. Usually, it takes all available resources to come up with a reasonably good solution matching the current requirements.

In this process, it is highly unlikely that much thought will be given to planning for ways to build the system so that it can be ported to the next generation computing environment which may emerge a few years from now. That next generation always seems so far off ... yet in reality it is often much closer than one thinks. If this issue is not addressed up front, it is likely that in a few years all the work that was done to internally develop a useful monitoring system will have to be thrown away and a new project started from scratch. We see this all the time.

Buy:

Over 20 years, SL Corporation has progressively adapted its core SL-GMS graphics capabilities to at least six different hardware and software environments. Ten years ago, UNIX/Motif applications were all the rage. Then everyone jumped onto Microsoft with its MFC interface libraries. At each step of the way, SL provided upward compatibility for its customers who were using earlier versions. While not 100% compatible at all times, this approach saved vast amounts of effort for customers who needed to rapidly make their products available in the newer environments.

Currently, most user interface applications are being developed using the Java language, fully supported by the SL-GMS product line. However, the Microsoft .NET environment is coming on very strong, and is gaining a great deal of acceptance. While it is not guaranteed, it is likely that there will be a need to migrate some of the applications currently developed in Java to the new environment.

SL Corporation already has a .NET version of parts of its product line, and is continuing development to make the transition for customers to this new environment when it becomes necessary. At the present time, most SL customers develop in the Java language, safe in the assurance that, as the environment changes, they will have as smooth a transition as possible.

While this issue may not seem terribly important right now, many years of experience reveals this to be one of the most costly overlooked issues of all, when you think for the longer term. Protecting your investment against obsolescence must be a priority.

Let's now examine a typical scenario for each of the two approaches.

A Build It Yourself Scenario

Company A sets out to build a GUI front end to monitor application infrastructure.

An initial estimate is provided by the program manager and developers for resources needed to complete the project:

“In 2-3 months you can have something working. Then we'll enhance it from there. Estimate one person, with a little help from another half-time.”

After one month, some initial prototype may be working – perhaps a screen or two can be viewed. It's always the easy stuff that gets done first. It might even look pretty good, and the developers say that it's 80% complete, with just a little more programming needed.

total cost = 1.5 man months

The next month, they discover they are beginning to get bogged down as they try to run the system in environments other than their own machines. They download new Java versions, and discover problems with the JDBC drivers. The applet version won't work on certain browsers, like Netscape. They spend several days looking at web posts, trying to figure out why something won't come up the first time. Two people are now involved full time.

total cost = 3.5 man months now

At the end of the third month, they are still bogged down in systems issues, and the pressure is on to get the job done faster, so that its completion will not be delayed. The initial 2-3 man month estimate is extended to 4 months. Development continues, but due to pressure to get it done, shortcuts are taken. Some functions work, but when demonstrated to management, it doesn't work all the way. Developers say they will need an extra 2-4 weeks to make it work completely. The completion date is extended to 6 months.

total cost to date = 5.5 man months

During the next 3 months, development continues, but the project is still not complete. In order to try to finish the project, an additional developer is added to project. Testers start to work on the project too. Now 3 people have been working on it for an additional 3 months (9 man months).

total cost = 14.5 man months

During this last period, problems are encountered with performance. If more than two or three people access the system concurrently, everything bogs down. Sometimes the system crashes after 45 minutes because it runs out of memory. The developers say they need tools to help them solve the memory problems. Another

3 or 4 months go by while these problems are addressed. The functionality is still not complete. During this period the system becomes partially workable, but major holes in the system remain, requiring 6 more man months to develop. Another 6 man months are used up in this phase.

total cost = 20.5 man months

Now the project is 9 months old. Most functionality works, but a long list of enhancements and problems remain.

New users of the system find additional requirements. The original design may not adequately have made room for these enhancements. E.g. the system has been designed with specific names and variable conventions based on an initial set of requirements. Now the developers realize that a slight error was made. It was fine as long as the requirements were what they were. But these new requirements prompt some good design discussions, and it is realized that a major rewrite of the system is needed in order to support a new request from management. It won't take too long, maybe a month, but some other requests will have to be put on hold while the architecture is redesigned.

Meanwhile two of your best people have been working on this problem for 9 months, and have developed a fairly workable system, but the cost has been significant.

At this point, some of your people begin to discuss the fact that maybe this system should have been written in .NET because this is becoming a more popular development environment.

Everybody scratches their heads, and comes to the conclusion that there is no way the system can be migrated without a major and complete rewrite, because so much code is Java-specific.

A year later, it is clear that something must be done. A couple years later, management is looking to a complete rewrite of the system in a new language and environment. And the cycle begins again.

A Buy and Build Scenario

Company B sets out to purchase a system designed for graphical display of key business process metrics and to adapt it to their purposes.

It would be presumptuous to paint a completely rosy picture of the alternative, using a product like SL-GMS Enterprise RTView. Obviously, there are pitfalls here too. However, due to many years of experience, not only building systems, but also dealing with changing requirements, and environments, means that it is much more likely that the design for the system will have the depth to allow for growth and expansion. A typical scenario might be:

Initial phase.

A developer is assigned to survey the market, to compare available products and make a recommendation. The SL GMS Enterprise RTView product is recommended due to its track record, functionality, flexibility and cost. The software is licensed and installed, and a proof of concept project initiated. Some initial problems are encountered as version issues are resolved, and the requirements are better understood.

total cost = 1.5 man months + software license fee

Second phase.

Further analysis of requirements and prototyping ensues. Training of a designated ERTV expert and a backup developer in use of the tools is conducted. The primary developer spends full time and the backup developer half time, and they are able to create dozens of immediately useful displays during this two-month phase.

total staff cost to date = 4.5 man months

Third phase.

During the next 4-6 months, customer personnel have developed expertise in creating new applications and displays using the SL-GMS Enterprise RTView system. Features that were unthought of at the beginning of the project have now been implemented. The company is seeing significant return on their investment in the software and the development, as the advanced features built on top of the stable ERTV base are providing highly useful real-time information to the company.

Developers are able to respond quickly to every new request that comes up, as the system facilitates rapid development of new displays and monitoring applications.

New requirements come in from management, and developers discover that, wow, ERTV already has that capability! If it doesn't, contact with SL reveals that SL has done something very similar for another customer and can provide a working example in a few days that should illustrate how to implement this new feature in

their system. A few weeks later, the new function is fully implemented and functioning. No rewrite is necessary.

total staff cost = 12 man months

Over the next couple of years, a new language or environment comes along. Management wants to investigate deployment in this new environment. Developers are happy to discover that SL has been developing for years in the new environment, and has a product that is largely compatible with the Java product. In fact, all screens developed in the past 3-4 years are readily deployed in the new environment. Some programming changes are needed, but they are minor. The effort is estimated at one tenth of the effort of a complete rework.

The result is a more quickly deployed, less costly, more reliable and adaptable Enterprise Infrastructure Monitoring solution that can easily evolve to meet new requirements for years to come.

Conclusion

The scenarios described above are obviously exaggerations. Chances are that company A will not do as badly as described. Typically the company has some pretty good people, the job does get done and things tend to work out.

Likewise, buying a product like SL-GMS Enterprise RTView is no panacea. There will be problems, integration issues, etc. and realistically it is likely there will be some things the system won't be able to do and there is a learning curve to be able to fully exploit the system's capabilities and to realize its full benefits.

No complex decision is ever black and white. The decision-maker's job is to weigh the various tradeoffs and make an intelligent decision based on likely scenarios.

This often means avoiding the unknown risks of a "go it alone" strategy in favor of known costs to license a proven base system and the expertise that comes with it. You want to avoid making the same mistakes that have been made so many times by others. The build-your-own strategy is so fraught with potential problems that one would have to go in this direction very, very carefully.

Let's say that you have some good people who could get the job done and that you expect that you can manage to succeed in building your own monitoring system. The questions you still must ask are:

- What is the actual cost to the business of not making these solutions available sooner? Who can say how much money will be lost in the interim by doing things the same way as before?

- Could you actually do the project better, if you were to use a proven tool like ERTV? Yes maybe your apps will be pretty good, but what if your competition is designing an "excellent" solution and basing it on existing product, so that in six months they will be that far ahead of you ... is it worth that risk? How can you measure the financial cost of that risk?
- ... and finally, what happens when the system that may take years to develop eventually becomes obsolete (which it will)? Will your engineers have the time and foresight to build in mechanisms that guarantee upward compatibility to the next environment that comes along? Maybe, but probably not to the degree that a product from SL Corporation would, with over twenty years of experience doing precisely that ... anticipating obsolescence and building in techniques to handle it.

One must assume that the SL-GMS family of products deliver value far in excess of their cost since, for two decades, companies just like yours have selected and successfully deployed infrastructure monitoring solutions for their complex and mission-critical applications. And during that time, the product has been continually enhanced to keep pace with changing technology. With this track record, it is highly unlikely that your project will go sour when based on such a solid foundation.

Given this assumption, there remains only one issue ... do the benefits of building a system on top of SL-GMS Enterprise RTView outweigh the benefits of trying to build it internally?

Based on the analysis given in the last few pages, the answer is a resounding "Yes!" Building it yourself will take longer and probably cost more in the end. It is also not likely to be as complete or flexible and will most likely not be built to address technology changes that are sure to emerge.

The SL-GMS products are designed with your immediate success and your future success in mind. Building and maintaining enterprise-class high performance, real time graphical monitoring systems is our business and we do it well. By building on the SL foundation, you can concentrate on applying your expertise to the real business problems.

Contact Us

For more information on SL Corporation itself, and the SL-GMS Enterprise RTView product, visit the www.sl.com web site, or contact SL directly:

SL Corporation
240 Tamal Vista Blvd.
Corte Madera, CA 94925

USA email: info@sl.com
tel: 800.548.6881 (inside US)
tel: 415.927.8400
fax: 415.927.8401