Building Systems from Parts in the Real World

Everything's Possible, But Nothing is Easy

Roy R Weil

Michael Baker Corporation, 4301 Dutch Ridge Road, Beaver, PA 15009

WeilR@acm.org

Key words: Component reuse, software integration, component packaging, configuration management

Abstract: As manager of the software support group for a full-service civil engineering firm, my major challenge is in composing project-specific software solutions from pre-existing parts. Too often, the available parts are not quite right for the task; or the parts work individually but make incompatible assumptions about interaction, representation, or other aspects of integration; or a solution works on prototypes but does not scale up for production. For seven example cases, I describe the architectural integration problems and what we did about them. None of the specific examples presents a major challenge – my real problem is that each such example must currently be solved as a special case. The architecture research community could help me most by developing general methods and tools to help me identify and resolve these integration problems systematically and routinely.

1. INTRODUCTION

Baker Engineering is a full service civil engineering firm. We design and build structures such as highways, bridges, buildings, airports, and subdivisions. We have a division that operates and maintains clients' facilities. We also have a Geographic Information Systems division that creates geographic databases for clients. Some of our services are software intensive. We use engineering design analysis tools, geographic information systems, and Intranets both in-house for our engineering tasks and as products that we deliver to clients.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35563-4_35](https://doi.org/10.1007/978-0-387-35563-4_35)

The software is bought, provided by clients or government agencies, and developed internally to various degrees. We often adapt software and packages from previous projects. We prefer to create projects from existing components whenever possible, resorting to new code development only as necessary. The various departmental groups within Baker typically develop these projects with consultation from our small software group. When we know in advance that the software aspect of the project is problematical and/or requires extensive internal development, our software group may do the primary software development and then turn over the project to a departmental group. These groups typically do not have a person who is dedicated to software development. They typically do have an engineer or technician who, as a sideline to his production responsibilities, provides immediate computer support to his department.

I manage the software development staff that provides software consulting services to the various other units of Baker. We usually take on projects requiring one to three man-months of effort, although we may have one or two longer-range projects going on at any one time.

My biggest ongoing problem is the difficulty of adapting existing software to new projects, creating the glue that makes separate software components work together, and scaling/hardening prototypes for production use. For example,

- Projects often create or acquire interactive tools to prototype the production phase. When the project goes into production, we must process large batches of data – an operation best done in batch. But the interactive tools sometimes don't provide enough program hooks to run in batch.
- Projects adopt tools that provide good immediate functionality but have proprietary representations. As time passes, we need to add functionality, but available components that perform the new functions can't handle the proprietary representation.
- Professional engineers (i.e. non-software types like civil, mechanical) who are not programmers may develop or adapt applications that are good for personal use and later the company wants to expand the use to other individuals or larger projects. This may involve simply adding a better user interface and more robust error detection and handling, or it may require the replacement of some components with more general or scalable ones, or it may involve re-implementing the concepts in some language/application that will scale to a larger volume or more users.
- Cost considerations sometimes force us to assemble solutions from parts or even write fresh code, even when technically adequate solutions already exists. Often the critical cost is not the initial cost of the

application, but the cost of deploying its viewers, browsers or other client components to all potential users.

In the body of this paper I describe several specific examples. Each of these can be solved in isolation; the solution is often straightforward. Indeed, the solution is often included in the example. The major challenge I face as the manager of the software development group is not solving any particular problem, but the sheer volume of problems like these and the drain it places on our software development resources.

The important point behind these examples – the pervasive architectural issue – is that these problems come along as regularly as rush-hour traffic, and each one consumes resources, sometimes significant resources, for its special-case solution. Further, we generally spend more resources in discovering the incompatibilities and shortfalls, than in fixing them. The consequence is that we don't realize the promised benefits of component reuse, and we get less done at more cost than we should. Even worse, there are projects we would like to do but do not even try, because we can see how the petty aggravating integration details will more than offset the benefits.

2. EXAMPLES

2.1 LIDAR: Improved aerial mapping

2.1.1 Problem background

Lidar is an aerial mapping technology for producing high precision contour maps to be used in engineering design. A specially equipped airplane uses prototype technology for high-precision data acquisition. Software developed by an external research organization does the initial conversion from data acquisition to xyz coordinates. We (Baker) had to take this software and turn it into a production product that would efficiently process over 30 Gigabytes of data per county scanned. Plans are in place to process over 50 counties in the next year.

The airplane acquires data in the form of scan lines at 3-meter spacing. Points along the scan line are 3-meters apart. This gives us a grid of points 3-meters by 3-meters. Each point consisting of the distance from the airplane to the ground, the GPS (Global Positioning Satellite) location of the aircraft, and the roll/pitch/yaw information for the aircraft. The data must be converted to topographic maps by the process:

- Do internal calibration and correction of the data acquisition system
- Calculate the $\langle x, y, z \rangle$ location of each spot on the ground through spherical trigonometry

- Transform the $\langle x, y, z \rangle$ location for geographic projection
 - Produce a contour map
 - For quality control, produce difference maps for overlapping areas; also compare the contour maps to known low-precision maps
 - Merge adjacent flights where the data overlaps
- We use a mixture of pre-built and custom software to accomplish this.

2.1.2 Computing obstacles and resolutions

Contouring package format incompatibility. The program was originally developed on a Sun workstation. The developer created $\langle x, y, z \rangle$ output ASCII files that were formatted for a graphical display package. The display language provides commands to specify scale, viewpoint and sun angle. It allowed comments with a # sign in column 1. The contouring package wanted the data in $\langle y, x, z \rangle$ order and comments with a semicolon in column 1. The initial workaround was a Perl script that did the format conversion. A better one was a parameter to the conversion program that told it which format to output. This of course required us to rework the program to find all the places that output data.

Difference package worked in interactive mode, but failed in batch. Some of the flights covered the same ground area. In particular one of the early flight was flown perpendicular to most of the other flights. By checking this flight against later flights we could verify that the data acquisition system did not drift over time. We do this check by calculating the difference between the two generated ground surfaces. The difference program worked fine in the interactive environment, but produced no output when I tried to run it in batch. Since I was planning to do about 8,000 of these differences, this was unacceptable. I sent data to the vendor and they were able to reproduce the problem. It took three weeks to get the fix back. They sent me the patch, but it was for the wrong package. The software we use is structured in 12 to 15 sub-packages each with specific functionality. You buy as few or as many of the sub-packages as you need. There is overlap in the functionality. The difference routine was in three of the sub-packages. They sent me the patch for the only one of the three that I had not bought. The workaround was to get an evaluation serial number for the unpurchased package. Hopefully they will issue the update to the package that I have purchased before the evaluation period runs out.

Display package couldn't handle volume. Because of the high density of the data, (a point every 3 meters) the contouring package produces lines that are very ragged, rather than the smooth lines that are on a typical contour map. This makes the display files 20-50 times larger than usual, and they exceed the allowable file size of the display package. The workaround

was to process the data in sections. Two programs were written: one to split the data into reasonable-sized chunks, and the second to tell the display package to display the multiple files required to cover the specified area.

Coordination with our partner company. The development was a joint venture with an external research organization. Each group had separate responsibilities. We had a difference of interpretation with the partner company. The original data met the specifications but showed a systematic error. In our opinion, the specification was not quite right, because it allowed systematic, correctable errors. The specs called for an error of plus or minus one foot. The data showed a systematic error of plus 6 inches on every tenth scan line. There was a physical explanation for the error. One of the pieces of the apparatus was a spinning ten-faceted mirror. Since it was the partner company's responsibility to produce the $\langle x, y, z \rangle$ data, I thought it was their job to determine the exact magnitude of the error, and produce the 10 adjustment factors. Their response was that the data was within specs and therefore they were not going to do anything about it. Since I knew the eventual clients of the data would not accept data with this kind of error, I had to modify the partner's program to accept the adjustment factors and to determine the values.

Dropping information too soon in the process. To make the problem of removing the systematic error a little more interesting the data acquisition hardware reported out the rotation angle of the ten-faceted mirror in a 17-bit integer. Since the original programmer did not care which facet was being processed, he simply extracted the low order sixteen bits (2 bytes) from the data stream to obtain the angle of the mirror for each data point. This meant that when I needed to determine which facet was being processed, I had to determine where in the data stream the 17 bit was located and merge it with the other 16 bits before determining the facet number. Obviously a better interface between the modules would have made this task simpler.

Scaling up the data sizes. When the program was in the developmental stage the length of a data collection stream was relatively small, less than 2 miles of data. At a sample rate of 30 points per second, this produced 1300 points and the programmer declared his arrays and structures at 2,000. The production environment uses longer flight lines. There turns out to be a system constraint of about 35 miles. These limits were nicely `"#defined"` in C so I upped the limit from 2,000 to 32,000 and got "program exceptions" for some medium size cases. It turns out when you do a `malloc` of a double precision variable (8 bytes) for 2,000, the answer (16,000) is less than 16,384 so storing the `malloc` calculation in a small integer works. I had to find all the `mallocs` (there turned out to be 23 of them) and change the calculated variable from an integer to a `long`. And, of course, after I had set the limit to 32,000, 5 flight lines out of about 400 exceeded that limit. This

required increasing the limit above 32,768 to process those lines. I had to revise the program so that all array accesses were done with long integers.

Miscommunication of operator procedures. To process a flight line of data requires 7 related files. To provide for the batch processing and to reduce the chance for operator error, I devised a file-naming scheme in which the first 5 characters of the file name established a relationship. This scheme apparently was not conveyed correctly to the person producing the files. They did not observe the naming rules. This resulted in a group of files with the wrong names, which therefore could not be processed automatically. Since the data was burned onto a write once CD after being created it was not correctable by a one time renaming of the files. To resolve the problem I wrote a rename script that the operators must execute each time they copy files from the CDs

2.2 GeoMedia Web

2.2.1 Problem background

One of our long-standing utility clients was interested in a web server for displaying geographic data. He wanted to ask questions like “where are all my open work orders” and see a display on a map with all the selected buildings filled in red. He also wanted to click a telephone line on a map and have the system trace it back to the central office. The drawing files were in Intergraph CAD (Computer Aided Design) file format. Intergraph has a product called GeoMedia Web, which will take a one of their CAD files, or a portion there of, and reformat it to a standard web format (.CGM) for which there exist plug-ins and ActiveX controls.

Another of our clients, with our help, had populated such a database and had developed the queries and Perl scripts to do those kinds of queries. They were willing to allow us to use a small sample of the data from the existing web site and demonstrate it to the other client. So the task was to take a working web site, move it to another server, and reduce the size of the data set.

2.2.2 Computing obstacles and resolutions

This task turned out to be an adventure in incompatible assumptions and hidden configuration files with hardwired information.

We installed GeoMedia Web on the new server. We copied the existing web directories from the working server and attempted to access the home page on the new server. We got a “404 page not found” error message.

Different web servers assume different default names when you request the home page. In one case it was index.htm, in the other it was default.htm

We next discovered that the GeoMedia Web program had some internal configuration files that it used to let itself know what “projects” were available. It kept these files within its own directory structure, rather than in the Web’s directory structure. We had to go back to the original web and obtain this file, and then modify it to contain only the demo project, not the other five projects that were on the original site.

Another of the configuration files contained the name of the ODBC entry that pointed to the database. On the new machine the ODBC name was different and the original user id and password did not exist.

Still another configuration file contained a list of file names within the project. These names were fully qualified and therefore had the wrong drive letter and upper-level directories for the new web servers file location.

Once the cause of each problem and the associated configuration file was identified, it was straightforward and easy to fix. However, to discover the cause of the problem took some time. In general, the error messages just indicated that something did not work or was missing. It took some sleuthing, debug tracing and debug prints to find out exactly what was wrong. Finding and resolving these problems could have been easier if there had been a configuration description and updating tools.

2.3 GIS version control

2.3.1 Problem background

Intergraph offers a suite of programs for working with GIS (Geographic Information System) data. It includes a basic nucleus package, an administrator package, a digitizing package, several kinds of analysis packages, etc. In all, there are about 12 to 15 specialized packages, each of which does a collection of related tasks. The specialized packages are sold separately. These packages use a database to store some of the configuration and attribute information.

One of the things Intergraph did right some years ago was to develop an interface between the packages and the various database engines. They created an internal definition of SQL, which all the packages use. This interface package processes the internal SQL and transforms it into acceptable SQL for the given database engines that is attached at your site. Microsoft users will recognize the ODBC model. Intergraph calls it RIS (Relational Interface System).

2.3.2 Computing obstacle and resolution

I recently needed to use another one of these packages. I downloaded and installed the new package. When I ran it, I immediately got an invalid database message. Since the database in question was working with the other packages I knew that was not the problem.

Upon investigation I discovered that Intergraph was in the process of upgrading the packages to use Microsoft's ODBC database interface rather than RIS. The working packages were the updated versions, which I had associated with a Microsoft Access database using the ODBC interface. The non-working package had not been updated and would only work through the RIS interface.

To accomplish my project I needed to be able to run the different GIS packages that were at different version levels. The solution was to transform my Microsoft Access database to an Oracle database and attach the Oracle database that would work in both version levels.

2.4 Upgrade version compatibility problem

2.4.1 Problem background

I was working with a commercial package that had a number of command-line processing commands, each of which had 6 to 10 parameters. These were expressed in the Unix command line style of minus sign, followed by a single letter identifying the parameter, followed by a space, and then the parameter value:

```
task -I input file -O output file -L level
```

We had put together an extensive series of scripts to process some data. This collection contained over 10,000 lines of script. We have been using these scripts in a production environment for over 12 years now.

2.4.2 Computing obstacle and resolution

As you can imagine, over the 12 year life of the system the underlying commercial package has gone thorough a five major revisions and many minor revisions. Each revision has brought changes in the parameters and parameter letters. The changes generally made the letters more consistent amongst the various commands, but sometimes additional functionality added more parameters. To compound matters, the scripts were being run in

several locations, on many different machines, which made it infeasible to do a simultaneous switch among versions.

At first, every time a new version appeared, we would install it on a development station and proceed to test the scripts. When an incompatibility was found, tests were inserted into the scripts to determine which revision was running. Based on the revision number different sections of the script were executed. After a while the scripts got pretty ugly looking, and it became hard for maintainers to follow.

After the third major upgrade the software/scripts were stable enough that we did not install the next two upgrades. So we are now running on packages that are two versions (about 6 years) back. This gives us grief when problems do occur because we can not get any support from the vendor. In addition the hardware maintenance cost on the older machines is expensive compared to the purchase cost of today's machines. At some point the maintenance costs are going to force us to redevelop all that software.

The overall problem is that the vendor's architecture did not provide a migration path between versions. Nor did it provide a reasonable way to run multiple versions of the same software on multiple machines with shared scripts.

2.5 Image library

2.5.1 Problem background

The Corporate Communications Department wanted to make available a collection of public relation images for projects to use. These were photographs of past projects, generally of high quality. These photos would be used in marketing brochures and in proposals to other clients.

One of the primarily software selection criteria was the mandate to allow as many people as possible within the corporation to have viewing, selection and download capability of these images.

We considered several packages. Most give us the required functionality of filing an image, assigning keywords to that image, performing a search on the keywords, displaying thumbnails of the search results, and downloading selected images. Of those that met the requirements the costs ranged about \$2-5K for a server and \$20-30 per client plus the cost of deployment. For the corporate communications application we estimated a need for 100-200 clients. Other departments could also use a similar system for the storage and deliver of drawing details, inspection report photos and progress report photos.

Instead of purchasing one of these systems, we decided to implement a package in-house using a web-based approach. This was almost purely a

cost decision, since some of the packages did exactly what we wanted. The difficulty of maintaining client viewers on several hundred workstations was also a deterrent.

2.5.2 Computing obstacle and resolution

The in-house application was built from scratch. It uses Microsoft's active server page technology to access a database that contains information about the images. The images are stored in a directory on the web server. The system consists of about eight or nine different web pages that are populated on demand from the database.

One of the web pages consists of a display of a number of images that had been selected by a search request. To make the system work at acceptable speeds this display had to show a thumbnail view of the images. The thumbnail would be a 96 by 96 pixel image that would download and display very quickly. By clicking on this image the user could obtain a larger, more detail picture.

We found three free or cheap applications that would convert a directory of high resolution images to a set of thumbnails. However, all were interactive. An operator had to select/display the images, invoke a command from a pull-down menu and perhaps enter new file names and/or directory. One of these applications would do some commands via a DDE interface. Given a DDE interface it is possible to write code that will call the application in a batch environment. However, the application did not expose the functionality for the process that we were executing.

Since we were dealing initially with only 500 to 600 images, we decided to grunt it out. We used the macro facility of PhotoShop to do the re-sizing and naming work. The operator loads the image, invokes the macro, and then goes on to the next image.

As a side note: After the 600 conversions to thumbnails were completed, another application became available that allowed the creation of the thumbnails in a batch environment.

2.6 Resume library

2.6.1 Problem background

The Document Services Department maintains a collection of resumes of the professional staff to be used in writing proposals. The resumes are formatted electronically in WordPerfect, using a house style, including logos. They are available on a shared server. Proposal-writers who want to incorporate a resume into proposals could attach to the server, copy the

relevant WordPerfect file to their hard drive and make the necessary changes. In general, these changes would be textual, not format. The changes might include deleting paragraphs that were not relevant to the proposal and adding paragraphs to emphasize relevant experience. Once the proposal is submitted the modified version is discarded.

The task was to make this collection of documents more widely available, easier to search and easier to obtain. The obvious choose was to make the data available on the internal Intranet.

2.6.2 Computing obstacle and resolution

No available WordPerfect-to-HTML converter works well enough. Everything we could find butchered at least the house formatting and sometimes everything else as well.

We needed a full-text search. Sometimes a proposal author wanted to include a resume of an individual with a particular expertise that was not commonly exploited by the company. In such a case knowledge of who had that experience would not be available from mental memory.

Although the resumes are loaded on a central server, the updating is done throughout the company by the various departments and individuals.

To avoid the problems with multiple copies of the same information, the resumes were kept on a single shared server. Typically the remote offices would not be attached to this server, so they would have to go through the procedures to attach to the server before they could access the data.

We examined several alternatives:

There exist web browser plug-ins (KeyView, QuickView) that have the capability to display the Word Perfect format. However at \$30-50/client plus installation, the plug-in route is expensive, both in initial outlay and in distribution/installation/update costs.

PDF format is another alternative. The user would view the PDF formatted document, but download the Word Perfect version for the proposal. Version control is the main reason that this was rejected. Updating is done rather frequently and by many people in the WordPerfect formats. Getting the updater to create the PDF file whenever a change is made is problematical, and would require the Adobe Distiller at many desktops. Automatically invoking the PDF converter would have to be programmed.

Another package (Net-it Central) converts any printable document into its own proprietary format (.jdoc), which is displayable by a browser using a Java applet. This package will work in a scheduled batch mode, converting any documents that have changed since the last time the package was run. This batch process creates a table of contents for the documents. However, the resulting table of contents isn't wholly satisfactory. Net-it Central does

provide a template facility to modify the header/footer/table of contents of the resultant pages that it is creating. This solution adopts yet another document format, and yet another user interface for controlling a display. The buttons for moving around the document are separate controls, not integrated into the browser buttons or scroll bar. It is yet another interface for the user to know and use.

Any of the above solutions will work. None of them, however, would give us an increase in functionality or availability over using the shared server great enough to warrant the expense and aggravation of the change. As a result, we decided not change existing procedures.

2.7 Database communication

2.7.1 Problem background

A database server provides verification of customer information for small retail companies. The current system has one server and N clients. The user works off-line preparing query data. The client machine then calls the server via modem. Once a connection is established, it sends the query, retrieves a report, and hangs up. The user then displays/prints the report off-line.

The system has been in place for a number of years and was designed and built before the widespread availability of the Internet. The users are becoming more sophisticated and they want to replace the telephone connections with an Internet connection.

2.7.2 Computing obstacle and resolution

The best of all possible worlds would be to find an existing Internet interface package that provides the same procedure calls as the modem interface package. At present we have not found such a package.

One alternative is to re-write the modem interface, to establish the connection using sockets. The calling sequence and actions are roughly the same whether you are using a modem or a socket. The sub tasks are to link up with the remote machine, login a session, pass data back and forth, and then disconnect.

Another alternative is to strip out all the modem interface code and replace it with remote procedure calls directly to the various routines in the server module that do the work, then use the Microsoft DCOM concept to establish the connection between the server and the client.

This task is on the low priority list so we are still looking at technologies and evaluating our alternatives.

3. DISCUSSION

As mentioned in the introduction my biggest ongoing problem is the difficulty of adapting existing software to new projects, creating the glue that makes separate systems work together, and scaling/hardening prototypes for production use.

The examples show some of the problems that I deal with on a day to day basis. They illustrate the classes of architectural problem that make a large fraction of my work:

- Interactive programs and procedures are sometimes difficult to convert over into a batch environment.
- Different components sometimes can not be joined together even though the data that one component is outputting is of the type expected by other. The problems are both simple, like format and sequence of information and more complicated, like incompatible demands and assumptions.
- Reasonable existing solutions may be precluded by economic restrictions. Either projects just do not get done, or in-house solutions are developed which have a limited but adequate set of capabilities.
- Vendor upgrades will generally cause problems for existing systems that are built on top of the product. Vendors tend not to have backward compatibility.
- Software may not scale in size or performance.
- Not all default assumptions are documented, or their documentation is scattered in none obvious places.

I am trapped in a Turing tarpit: Everything is possible, but nothing is easy. The tarpit is not one of creating individual programs to use as components: that's pretty straightforward now. The tarpit is filled with the glue that we create ad hoc to stick together components. The glue is necessary either because the parts were not initially designed to fit together (different vendors), or the connection tools are primitive (command scripts), or the style of use does not meet production needs (interactive vs batch).

ACKNOWLEDGEMENTS

The work reported here was, for the most part, in-house software development and configuration for Michael Baker Corporation and was supported by the company. I wish to thank Joel Platt for one of the examples. I wish to thank Mary Shaw for her encouragement in getting me to write this paper and for making red marks all over the pages on many drafts.