Ills Cured With a Dose of Remedy

Michael Critchfield
California State University, Chico
Chico, CA, 95929
(530) 898-6000
mcritchfield@csuchico.edu

Michael Murray
California State University, Chico
Chico, CA, 95929
(530) 898-6000
mmurray@csuchico.edu

ABSTRACT

We presented a paper in Denver that addressed our Remedy design process. At that time we talked about our current development process, but we hadn't implemented the product at that point. We received a lot of wonderful feedback from the session, and we found that many people wanted to know how things turned out.

We were intrigued by the amount of people that were considering Remedy as their helpdesk solution. Many of these people were hesitant to go with Remedy because of the cost issues, or were concerned about the impact it might have on their own help desks.

Because we hadn't implemented Remedy at the time of our presentation, we couldn't answer many of their questions. Since the last SIGUCCS conference, we have not only fully developed the product, we have also had our project leaders attend multiple Remedy training sessions, and have had these staff members further develop the product that we currently use as our User Services workflow system.

We will discuss every aspect of our development, to include:

- Project management. A Remedy implementation is a major undertaking. Every college has different business rules, and how you handle this area is crucial.
- Development. The development stage should involve anyone that will touch the application. Involving your employees benefits everyone.
- Testing. Every individual should test the product. This will minimize the number of issues after implementation
- Cross-platform issues. Remedy is now issuing version 4.5. The last version of Remedy that supported Macintosh clients was 3.2. Organizations need to think about this before jumping onboard.
- Licensing issues. The most expensive part of Remedy is the licenses. What is more important is understanding how much you will need to spend on licenses before you buy.
- Other topics
  - Development server. Is it worth it?
  - Crystal Reports. Compatibility issues and importance of reports.

This presentation will be of great value, not only to those campuses that are considering Remedy, but also to those that have been using Remedy, but haven't been satisfied with their current implementation.

LEAVE THIS TEXT BOX IN PLACE

AND BLANK

1. PROJECT MANAGEMENT

All projects need a good manager and Remedy is no different. When we started our project, we opted to bring in a project manager from outside our organization. We felt that a project manager with no prior knowledge of our operations would have an unbiased view and would help us to better define our business rules.

It was the project manager's responsibility to ensure that all of the groups involved in the use of the final product got a chance to provide input into the development. It was also the responsibility of this project manager to create a list of needed resources and to make sure they were purchased and available when needed.

The project manager was also responsible for overseeing our outside developer. He was to ensure that our application was being developed to the specifications agreed upon in our Requirements Analysis.

Unfortunately for us, we discovered four months before our rollout that our chosen project manager was doing none of the above.

To make matters worse, our developer was not even close to making his deliverables. We found out later on that he had not even been through training on Remedy's most current product, which is what he was developing for us. This meant that many of the features that were available to us were not being used.

In April of this year, we felt it was time to shift gears. We decided to take two of our professional helpdesk staff members and make them responsible for the Remedy implementation.

2. A NEW BEGINNING

First, we created a list of all necessary purchases and completed a project cost analysis. Realizing that purchasing equipment and software would take months and that this project had a drop-dead date of June 30th, it became evident that this needed to be done quickly.

We spent a few days determining exactly what we would need to make this project a success, and then presented our list of needs to our director. While he wasn't pleased with the amount of money he would need to spend, he knew that there was no choice.

A week after taking on this project, our developer came to campus to install his "completed" work on our Remedy server. After reviewing the product we discovered that several parts of the program simply did not work and several areas were not developed per our Requirements Analysis.

We were becoming concerned that we were getting so close to our deadline, yet had very little to show. The development was going much slower than we would have hoped, and the product simply was not acceptable.

We were then notified that Peregrine, a Remedy competitor, acquired the company who we contracted with for development and that our developer had left said company to take a position with another company.

It was now two and a half months to our target rollout and we had a non-functional product, and no developer.

We decided that with the short time we had left it would make the most sense for us to finish the application. While this would be difficult to do in our limited timeline, we were unhappy with the current progress, and basically reached our breaking point. It was time to move on.

3. REMEDY TRAINING

One of our problems was that we had no support for our customized application, only Remedy support. We needed to be familiar enough with the Remedy product to ensure that any bugs discovered during rollout could be quickly solved.

Therefore, we scheduled three Remedy training sessions, which we felt would provide us the tools necessary to handle our responsibilities.

We are breaking out "Remedy Training" in this paper for one reason and that is to emphasize its importance. If you are planning a Remedy development whether developed in house or out, learn from our experience and send two or more of your staff to Remedy training. It will help you to know what to expect and it will give you the understanding of how Remedy works so that you can better oversee your developer.

Both of us attended the following training sessions:

- **Administering the Action Request System**. This class is an introduction to the ins and outs of Remedy administration. They cover all of the basics and even touch on some advanced features.
- **Advanced Topics**. This class was one of the most valuable training classes we attended. This class would probably be better titled. "Cool Stuff You Can Do With Remedy". We took a lot of great information back to Chico after this class
- **Performance Tuning & Troubleshooting**. This class teaches Remedy administrators how to keep their applications running smoothly and quickly. They touch on server tips, design tips, and much more.

These classes really launched us into our roles. We came back with a much better understanding of the system and were able to design it the way CSU, Chico had intended.

However, we only had 6 weeks left to produce our application. Our first task was to determine what currently worked and what we needed to fix.

After reviewing the workflow that was currently in place, we realized that we would have to almost completely revamp the system.

4. DEVELOPMENT

After reviewing the results from our testing we created a list of modifications that needed to be made before rollout and began hammering them out one by one.

Since we purchased a development server as well as a production server we could have testing occurring at the same time as development. We cannot stress enough how valuable this development server was.

To our benefit, we had a well thought-out Requirements Analysis and a good idea of how the end product was to function. This made our redesign process much quicker.

We started with the application's Control Panel. In Remedy, the database is accessed through "forms". During the development process, many forms are created. In our application, we have 41 different forms, which are used for a variety of different functions, such as:

- **Service Request Form**. This is the main form in our system, as it contains all of the key information for work order tracking..
- **Assets**. Every computer, monitor, printer, etc. on our campus is stored in this form, along with the state decal number.
- **Locations**. Every building and their associated room numbers are stored in this form.
- **Customers**. All campus employees and their related information are stored in this form.

There are many other forms that contain crucial data, and a number of forms which are strictly used for our workflow.

The problem with this is that the amount of forms can be confusing for the end user. Also, we wanted to limit the number of forms the end user had to see.

The easiest way to interface these forms is through a "control panel". In Remedy, the Control Panel is the starting point for all users. You could think of it as your "one stop shop" for the Remedy system.

Our Control Panel was nothing but a few search buttons and a button to create a new request. We felt that the control panel should be used for much more.

We added such features as:

- Tables, which display the technician's open requests, their groups open requests, and associated groups open requests.
- A documentation area. All of our commonly used documents, such as remote access dialup instructions, are stored in this area so our Help Desk staff can quickly access them for our customers
- A reporting area, which allows management to view various reports.
- Special Buttons:
    - Quick Closes. This button allows a technician to select a pre-defined service request type, such as a password reset, hardware consultation, or training request, then click a button and have a request created and automatically closed.
    - Major events. In the event of a network outage, for example, a request can be created that acts as the master event, then requests can be quickly added to it with the click of a button. Once the event has been repaired, a technician can close all of the request with a click of a button. A description of the work done is automatically added to each request, then they are all closed.

There are many other features we added to the control panel as well. It is definitely the way to go with Remedy.

In addition to the control panel, we basically reworked all of the forms and associated workflow. Major areas, such as billing, took an incredible amount of time and effort, but we were able to develop them as we had intended.

In the end, our product was barely recognizable compared to the product we were originally given.

5. TESTING

Before we took over development, early versions of our application were e-mailed to us in the form of "definition files", which we would load onto our server. It was our job to check every aspect of the application for functionality, and to compare its features to our Requirements Analysis.

As we tested, we noted bugs, or the parts of the application, that either did not work, or did not match our requirements. These bugs were tracked in an Excel spreadsheet that was sent back and forth between the developer, the project manager, and our testing team.

This proved to be a mess after a while, as we would often get new versions of the bug report that did not contain all of the proper information. We were losing data between all of the people involved since we were not using one common copy of the report.

Our solution was to set up an online database that contained the bugs. Only one person would be allowed to maintain the database, therefore eliminating loss of data. A bug submission form was also created that was to be used by the people involved in the Remedy project. This form did not write to the database directly, as we wanted to verify each bug report and also make sure that the bug report was not a duplicate entry.

Bugs were assigned identification numbers (ID's) that could be referred to any time updates were made. If a bug was fixed, the developer would e-mail the database administrator with the bug ID and solution. The fix would then be verified by the testing team. If the bug were indeed repaired, then the bug would be marked as fixed, and would no longer appear in the bug report.

6. IN-HOUSE TRAINING

We were now at the point where we were ready to roll out our production system. However, we had developed a system completely different than our existing work order database.

We quickly developed a training program for our staff. Since each area handles their workflow differently, we felt the best approach was to split the classes up, addressing each of our key areas separately:

- Help Desk
- Field Services / Tech Shop
- Network Management & Design
- Administration

We had one of our students create a Remedy user guide that people could refer to if they needed help, and we also followed up with each area after the product was rolled out.

After the initial training sessions, we felt that it might help to have some specialized training. Since our Help Desk staff had helped us test our application throughout development, they were far more familiar with the system than the other areas. We would send two or three of our Help Desk staffers to the other areas for an hour or two each day. During this time, the new users could get answers to any questions they had.

This also helped us get over the "old school" attitude that some people tend to have when they have to move to a new system. These folks are comfortable with the status quo and often can make a transition such as this difficult. We found, with the extra attention we gave them, that these attitudes were not prevalent at all.

7. CROSS-PLATFORM ISSUES

We were concerned when we found out that Remedy no longer offered a Macintosh client. Our campus user population consists of 40% Macintosh platforms. This makes for a large portion of our call volume. Because of this, it is imperative that our Mac support technicians use Macs in their daily routine.

If you've ever tried to troubleshoot an operating system problem without actually sitting at a computer with a similar setup, you can relate. For our student technicians, who typically do not have the support experience that our professional staff has, this becomes even more of an issue.

When we first started the initial Remedy project, we were informed that both the PC and Mac platforms were supported. Early versions of Remedy did have a Mac client, but they decided that Macs were such a small percentage of their

customer base that it no longer made financial sense to develop one.

Macintosh users can still get the last Remedy client, but the features that have been added since that version (v 3.2) are not supported. We developed our application in version 4.03, and many of the features we have will simply not work in the older Mac client.

We decided that we had three options:

1. Put PC's on the Mac technician's desks, in addition to their Macs.
2. Put PC Cards in each of the Macs.
3. Use Virtual PC.

Option one simply would not work. Our workspace is very limited, and adding an additional CPU, monitor, etc. on every Mac technician's desktop was simply not an option at all.

Option two appeared to be a good one at first. We had heard good things about these cards, and the fact that they had their own processor and memory on board made us believe that they would be a great fit.

We called Orange Micro and arranged to have a demo PC Card sent to us for testing. The PC Card was tested on a variety of Mac platforms, including a G4. We had a lot of problems with the card from the start, and never were satisfied with the product.

We were left with our final option. We loaded Virtual PC on a G3 and proceeded testing. Unfortunately, the performance of this product left a lot to be desired.

The act of loading the Windows environment alone took way too long. Once Windows loaded, things did not get any better. Accessing Remedy took too long, and creating service requests took even longer. At this point, we were getting concerned.

Then we decided to give Virtual PC a shot on the new G4s that were floating through our office. The performance increase was astounding. Windows loaded quickly, and Remedy responded much better. To this day, Remedy is not as fast on the Mac side as it is on the PC side, but it isn't bad.

To this point we had already spent a large chunk of money, and now we had the pleasure of informing our bosses that we were going to need to purchase Mac G4s for every Mac technician that needed to access Remedy. While the Mac techs rejoiced, the director was not pleased. Again, there was really no choice, so the G4s were purchased.

8. LICENSING ISSUES

Remedy makes the majority of their money through its product licensing, and to be honest, they are experts at getting the most money they can. Remedy licenses do not come cheap, but there is really no choice in the matter.

Their main product, the Action Request System (ARS), is licensed in two different ways:

- **Fixed licenses**. These are tied to an individual user. They are not available to any other user.
- **Floating licenses**. These are assigned to a general pool and "checked out" by a user. When a user logs in, a floating license is granted to the user and becomes unavailable for any further use until the user logs out.

While Floating licenses sound like the best way to go, it should be noted that they cost 2.5 times as much as a Fixed license. Also, once all of the Floating licenses are in use, subsequent user logins can only view Service Requests, without editing.

You will have to decide how many of each type to purchase for your organization. At CSUC, we purchased 45 Fixed licenses and 5 Floating licenses. Since most of our users are on the system throughout the day, we did not have the need for many Floating licenses.

There is one other licensing issue to be aware of. If you decide to use a development server, you will need to have the same licensing configuration that your production server has. However, the development licenses only cost half the price of production licenses. Just be sure to tell Remedy that they are for your development server, because they will likely try to charge you full price.

9. OTHER TOPICS

There are a couple of other things any organization should consider when moving to the Remedy system.

9.1 Development Server – Is It Worth It?

We were surprised to find out that the majority of companies running Remedy do not use a development server. The purpose of this server is to allow the developers the chance to add new features to the application without risking the data and the health of the live server.

If a mistake is made on the live server during production hours, the effects can be extremely detrimental to the organization. Support organizations cannot afford downtime or lost data.

For this reason, we decided it would be worth the cost of the extra Remedy server license to purchase a development server.

Our production server is pretty robust:

- Dual P450 processors
- 256 MB ECC RAM
- Four 9 GB Fast/Wide SCSI hard drives in a RAID 5 configuration

The good news is that the development server does not support users, so the hardware can be far less than the production server. We settled on a P450 desktop computer with 128 MB RAM and a 13 GB hard drive. This would be more than enough to develop our application.

Since we implemented our development server, we have done the majority of our enhancements on it. We have had a few occasions where we have had to make modifications on the production server, but only in cases where something was not functioning properly.

The use of a development server is highly recommended. The headaches you can save your organization are worth the extra cost.

9.2 Crystal Reports

One of Remedy's real weaknesses is the reporting feature. Reports can be run on any form, but the output is as bland as it could possibly be. Reports appear as plain text in rows, and there are no real formatting options.

Seagate offers a product called Crystal Reports that works extremely well with Remedy. The reports are designed from scratch, and are made with relative ease.

There are some limitations, however. For example, there is one field type (tables) in Remedy that will not display in a Crystal report. These table fields usually display a collection of information from a different form, and because of this, they cannot be displayed in Crystal Reports. Of course, there are workarounds.

The limitations of Crystal Reports are minor, while the benefits are major. Before you purchase Crystal Reports, try using the Remedy reports and see if they are enough for your organization. If they're not, go with Crystal Reports.

10. IN SUMMARY

When we presented our paper on Remedy last year, we still had not implemented our application. We were asked how much money we had spent at that time on Remedy, and the crowd let out a gasp. Yes, we have spent a considerable amount of money on this product, but we wanted to do it right.

We had heard enough tales of horror from other organizations about their implementation that we decided we would do whatever it would take to make it work the way we wanted it to.

In hindsight, we would have probably forgone the outside developer and project manager, and done it all ourselves. We could have easily sent our own people to Remedy training in the first place, and had them develop the application from scratch, rather than have them have to go through and rebuild a bad application.

11. ACKNOWLEDGMENTS

We would like to personally thank the people from the campuses of CSU, San Marcos, CSU, Sonoma, and Stanford University. These people answered every question we had about Remedy and made our jobs much easier down the line.