The following paper was originally published
in the Proceedings of the
Tenth USENIX System Administration Conference (LISA X)
Chicago, IL, USA, Sept. 29 - Oct. 4, 1996

Institute White Pages as a System Administration Problem

Jon Finke
Rensselaer Polytechnic Institute

Institute White Pages as a System Administration Problem

Jon Finke – Rensselaer Polytechnic Institute

ABSTRACT

With the planned departure of our mainframe, we had to find a new way to maintain and generate our Institute Telephone directory. This gave us the opportunity to examine every aspect of the directory generation process, and make changes to improve the accuracy of the data, reduce the clerical workload in Telecommunications and Human Resources and eliminate some duplication of data and effort. Given that we already had an Oracle based system to automatically create and remove Unix userids for all employees as they are hired and leave, it seemed that with some minor enhancements, this same system could also maintain our directory information.

To this end, we added a directory module to our Simon Account management system that extracts directory information from the Human Resources database, adds additional non employee information and generates LaTeX source for the printed phone book, HTML pages for the web and a raw feed for the **ph** server. In addition, using techniques developed for some of our system administration tools, we gave both individual staff members, and their departmental administrators, the ability to make changes and corrections to their own directory information, which would not only appear in the directories, but also be reflected in the official Human Resource database. This has greatly reduced the delays and paperwork involved in changing this information, and allows us to have accurate and up to date directories. As an added bonus, many of our "traditional" systems administration tools can now directly query the directory information and include contact info in the displays automatically.

Introduction

For the past five years, the Rensselaer Computing System (RCS), a collection of 700 workstations and Unix timesharing machines available to all students, faculty and staff, has had the Unix accounts automatically managed with a locally developed package called Simon [2, 4] which is built on top of an Oracle relational database. Simon connects on a daily basis to the Human Resources[1] database, and the Registrar's database (for student information), and based on changes in these databases, creates or expires RCS (Unix) userids as needed. This is similar to the Moira [7] system developed at MIT as part of Project Athena.

During this time, the Institute phone directory was maintained by staff in Telecommunications (part of the Computer and Information Services Division) using a set of custom, stand alone applications that ran on the, soon to be retired, mainframe. To make matters worse, the programmer who supported these applications had retired, and no one else still employed at RPI had ever even looked at the source code. Despite this, this was our most accurate directory database for staff, as each year Telecommunications sent an "update" form to each department to allow people to check their entries, and return them to Telecom to update the directory. Many people assumed that this was sufficient to keep their personnel records up to date, and never thought to send address changes to HR as well.

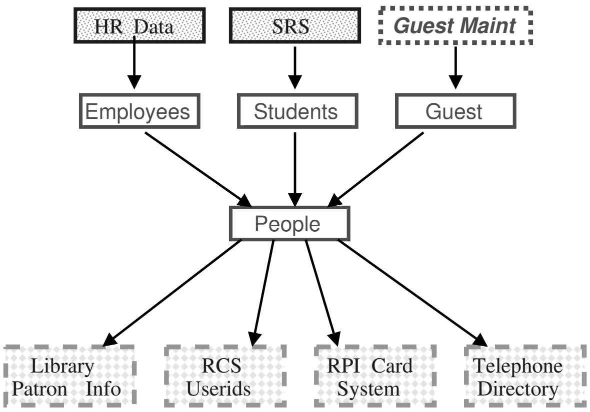

Figure 1: General Information Flow

General Information Flow

For any number of different projects, we have basically the same general data flow required to reach the desired goal. While this particular model applies to the situation at a University, similar models could be applied to Corporate and other organizations.

[1]When Simon was first written, this connection was a payroll tape, once a month. Since then the HR system has been replaced with an Oracle based system called Banner. This is generally know as FAIMS.

We have information from Human Resources, which we load into our `Employees` table[2], information from the Registrar's Student Records System that we load into a `Students` table, and finally, entries into the `Guest` table that are done directly via manual data entry. We actually have a number of types of guests, ranging from Alumni involved in an Alumni computing project, to Adjunct Faculty, contractors, and all sorts of other special cases. Each of these special cases, is maintained by the appropriate staff in other departments as needed. These are cases that do not have a full blown administrative information system to support their operation.

Date from these tables are merged into the `People` table. From this, we can then drive all sorts of other processes, such as the Library automation system patron database, the RCS Userid (Unix Account) management system, the University ID card system (used for identification, food service and access control), as well as the University Telephone directory system. All of these systems need to know when someone has come to Rensselaer, and when they leave, or their status changes in some way.

Directory Objectives

The primary objective of the directory project was to be able to produce an accurate and timely directory, both in hardcopy and online. In addition, we wanted to establish a single, authoritative and accurate source of directory information of employees and make it available to all administrative units. There were simply too many departments, all doing an inadequate job of maintaining overlapping or even duplicate address lists.

A secondary objective was to be able to delegate control and responsibility for maintenance of different directory elements to the appropriate party, in many cases the individual themselves, or their proxy. While a number of employees regularly used the RCS system, many others did not. However, each department had at least one person who used the finance system (FAIMS), so these people were given the proxy for all people within their department. Although the FAIMS system was NOT part of the RCS system, we had coordinated the userid naming so we were able integrate Simon applications into the FAIMS application menus.

Not all data elements were delegated to the individual level. Some things, such as the official name were controlled by Human Resources, and the actual inclusion of a person in the directory continued to be controlled by Telecom. The data elements are described in Table 1. The "Source" of a data element refers to the source of the data for regular employees. Since we include people, such as emeriti and contractors, in the directory who are not in the HR system, elements can also be entered by Telecom staff. In addition, when the individual is allowed to directly update their own information, their department administrators are also able to make those same updates on their behalf. The Telecommunications department can

[2]In a relational database, the data are stored in tables. Each table has a number of fields defined, generally referred to as columns. The each column has a name and a data type, and there may be additional restrictions on the information stored in a particular column. Each entry in the table is usually called a row.

Element	Source	Maintainer	Notes
Include in Directory	Faims	Automatic	Set based on status info.
Name	Faims	HR	**Never** changed manually.
Department	Faims	Automatic	See *Departmental Adjustments* below.
Title	Faims	HR	Also dept admins on a temporary basis.
Preferred First Name	Simon	Person	Copied back to HR.
Campus Address	Faims	Person	
Campus Phone	Faims	Telecom	Recently made available to dept admins.
Home Address	Faims	Person	Can set *Non Published* flag as well.
Home Phone	Faims	Person	Can set *Non Published* flag as well.
Email Address	Simon	Person	Their preferred email address.
Home Page	Simon	Person	
Fax Number	Simon	Person	
Family Info here	Simon	Person	People often list their spouse's name
RPI Class	Simon	Person	RPI Class year for Alumni
RCS Userid	Simon	Computer Center	The *userid* that may get forwarded.
Email Forward	Simon	Person	Intended for dept admins to forward email sent to *userid* `@rpi.edu`.

Table 1: Directory Data Elements

also act as a department administrator for anyone in the directory.

Alternatives Not Taken

Since one of the main objectives was to create one authoritative directory database with distributed updates, a distributed database approach, such as the one at Penn State [1] where the information was distributed among departmental servers, would not suit our needs. There may still be some use for this type of service for some of the lighter weight information such as machine load, but for basic directory information, there would be a single RPI wide source, Simon.

When we established RCS and created a Unix userid for every student and employee, we also established a mail hub with the host name `RPI.EDU`. People who did not read their mail directly on RCS, could access the mail hub via POP, or simply have their mail forwarded to their preferred mail system. With this already in place, an X500 solution, such as the one at University of Michigan [5] did not seem worth the overhead, for essentially just a query server. The updates were already being handled via Simon[3].

Information Flow

One of the early design questions that came up, was if it was possible to generate the directory entirely from information stored in the HR system, or if we had to shadow the data on the Simon system. The eventual decision was to shadow the data to Simon. There were two major reasons for this, the first is that we had to include some non employees in the phone directory, such as ROTC staff, vendors, and others who act as employees but are **not** employed by Rensselaer, and so, not in the HR system. The second reason was that we had to play some games with departmental affiliation (discussion below), and the HR system would not allow for that. Additionally, the HR system is a commercial product, and the folks who maintain that system would prefer to keep changes to a minimum.

Employee Status and Departmental Adjustments

The first problem to solve in producing a directory, is to decide which people are to be included in the directory, and when to take people of the directory again. This is basically the same problem we have solved for RCS userids, ID cards, and the Library system. In Figure 1, we gave a very simplified information flow diagram. In Figure 2, we go into more detail on exactly how we handle the data. Rather than put directory status information into the *People* table, we added a new table, *Dir_Master* to hold the employee status, department, etc.

The *Pebempl* table in Faims contains the employee status and department of each employee. This was originally used by the `employees` program to update the *employees* table, and based on the particular status and department, this would be passed to the *people* table and so on to create or expire RCS userids. The `Dir_Master` program using almost identical logic, maintains the *Dir_Master* table which controls if and where someone is to be included in the directory.

The HR (Payroll) system gives us three keys we use to decide if a person will be included by default, their department, their classification (Faculty, Administrative, Exempt, Non Exempt, Student), and finally, their current status (Active, On Leave, Terminated, etc.) For each new or changing employee (from *Pebempl*, see Figure 2) the `Dir_Master` program gets a default include flag value from the *Dir_Departments*, *Dir_Class_Def* and *Dir_Status_Def* tables for each of the keys named above, and if they are all positive, that person is marked to be included in the phone directory. In this way, people are automatically added and removed from the phone directory based on the ongoing activities of the Human Resources department.

Unfortunately, this is not enough. Despite our best efforts, we have to make exceptions, so for each key in the *Dir_Master* table there is a manual override flag. In most cases, this is left null, but if it has a value ("Y" or "N"), that value is used instead of the default value for that key. In this way, we can manually include someone who normally be excluded, or exclude someone who would have been included. There is also a general manual override flag for the entire record, where we can ignore all keys and include or exclude someone. We also include a *review_date* field, that we can set to remind us when to go back and review the manual settings. The big problem with manual settings, is that they are manual and will not change unless someone goes back in and clears them. We need to incorporate a similar field into our Userid management module, as some manual entries have lasted long past their appropriate lifetime.

Once we determined **who** was to be included in the directory, we then needed to determine **where** to put them. A problem with using payroll data, is that it is based on the financial system. While the financial system uses the institute departmental hierarchy, it is driven by the financial accounting requirements. One of the side effects of this, is that most department heads and senior administrators are not paid out of their own department, instead they get paid out of their supervisor's department. This has the effect of moving management out of their own departments. To correct this, we added a *Dir_Master.Dir_Orgn*[4] field to manually set which department we actually want to

[3] When we first started Simon development, we considered an X500 approach to allowing people to update their account information, but instead followed the Moira model of replacing Unix programs such as **chfn** with programs that updated the database directly.

[4] The financial system refers to departments as organizations, generally abbreviated as "Orgn".

list a person in. With this, we are able to put all the Deans and Directors into the right place.

A second problem we encountered with using raw information from the Finance system, is that the finance people have added "roll up" departments to simplify generating financial reports. These extra departments did not have staff and were not part of the "real" hierarchy, and did not appear anywhere outside of the accounting system. This generally resulted in all the staff of a department, appearing in an auxiliary department one level "below" where they were supposed to appear. To fix this, we added a field to the *Dir_Departments* table with the correct organization code for anyone entering that particular department. In this way, any new entries would automatically have the *Dir_Orgn* field set. This also proved useful when departments re-organized and the actual payroll changes lagged (usually to match up with the fiscal year.)

Propagation and Queue Tables

Once we have determined out who goes into the directory, and what department they go into, we need to get the address information, and provide ways to update it. The data falls into two rough categories, the data that originates (at least for some people) in the HR system, such as phone numbers and addresses (see Table 1), and the data that stays in Simon, such as Email address, Home Page, and things that HR didn't want (or more accurately, did not have a place to store in their system.)

In general, the Simon only information is stored in a table called *User_Dir_Info*. We created a view[5] of the *User_Dir_Info* table called *My_User_Dir_Info* that just has the data for the current user. This is the same technique we have been using to allow users to set their GECOS field (using a modified `chfn` program), and to forward email sent to *userid* `@rpi.edu`, using a program called `forward`. Using the `personal_info` program, people can display and update their own (and just their own) information such as their preferred email address, their home page listing, and similar information via the *My_User_Dir_Info* view. This program does do some very basic validity checking on some of the fields, we even go so far to lookup the hostnames in URLs, but other than that, we don't really "care" what people list here.

The processing for information shared with HR is more complex. These changes are queued for later

[5] A database view, looks like a table, but actually depends on underlying tables for the actual data. This data can be from the combination of two or more tables, or of just some selected columns of the base tables. In addition, it can put additional requirements on which rows of the table can be accessed. Views are used extensively in this project to delegate control and access to data.

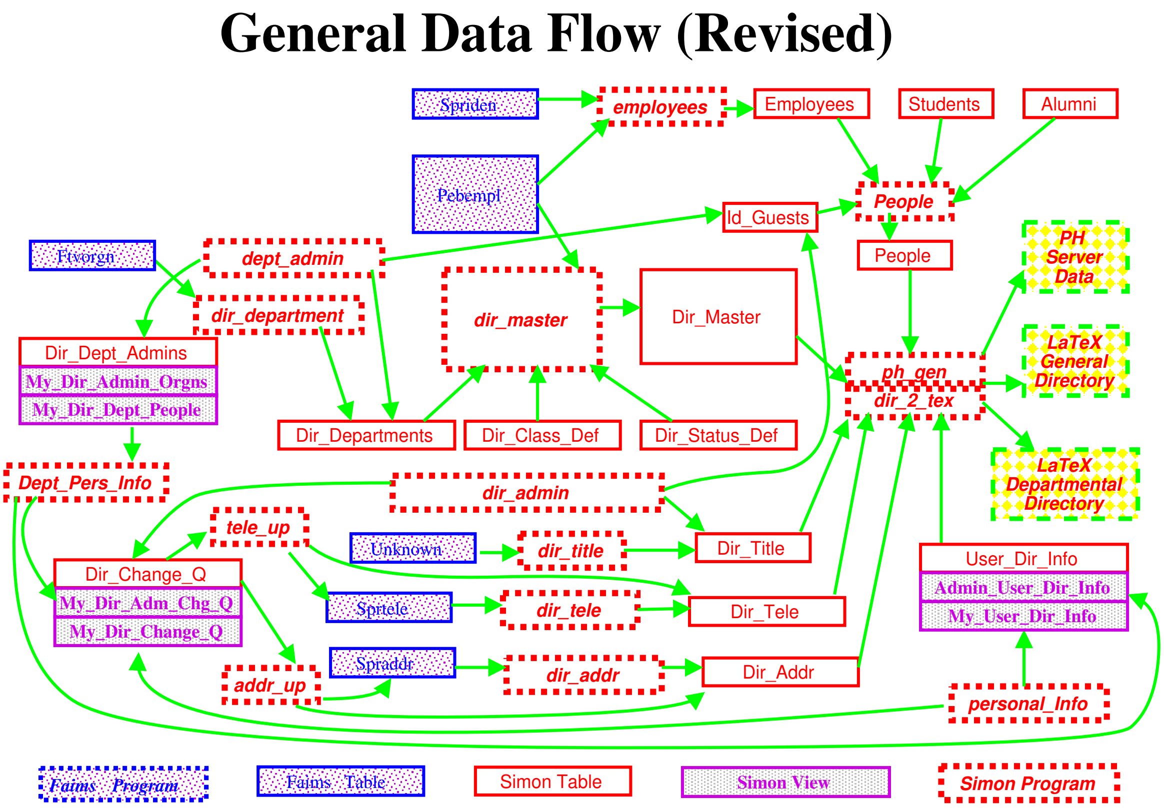

Figure 2: Directory Information Flow

processing in a queue table *Dir_Change_Queue*. The information flow is essentially the same for telephone, address, title and preferred first name information, although we do not actually pass title information back to HR. Both telephone and address information actually break down into either Permanent (Home) and Campus addresses, although this is actually managed by the `personal_info` program and a type field in the records, and the update flow is the same.

Let's look at an address change (but other change types are similar). As you can see in the lower part of Figure 2, if I want to change my address, I use the `personal_info` program, which connects to Simon and puts a change request into the *Dir_Change_Queue* via a view called *My_Dir_Change_Q*. Like the *My_User_Dir_Info*, this only accepts changes for the current user. This change is then applied to the *Spraddr* table in the HR system by the `addr_up` program. Immediately following this, that change in *Spraddr* on Faims, is propagated back to Simon via the `dir_addr` program into the *dir_addr* table.[6] In this way, the change is effected in both databases. If there is some reason why this change can not go to the HR system (the person may have been terminated, or not even listed in the HR system), the `addr_up` program will skip Faims and update the Simon table, *dir_addr*, directly. We currently process queued changes on a daily basis.

Departmental Administrators

The next part of the puzzle, was to give the departmental administrators the ability to make changes on behalf of members of their department. While many staff may not be regular RCS users, or even have ready access to RCS, each department has at least one person who can access the Finance system (FAIMS). For the purposes of connecting to Simon, the FAIMS system is equivalent to RCS. This also removed the need to provide assistance in connecting to these people; they already had that ability due to their use of FAIMS.

The first step was to create a table of these administrators, and the organizations that they were administering (See *Dir_Dept_Admins* in Figure 2). From here, we created the view *My_Dir_Admin_Orgns*, which is the list of organizations that they can administer, and with this and the *Dept_Master* table, a view of all the people they can administer, *My_Dir_Dept_People*. Finally, we used this to create a view of the *Dir_Change_Queue* table called *My_Dir_Admin_Chg_Q*. With a slightly modified version of the `personal_info` program called `Dept_Pers_Info`, departmental administrators are now able to queue address and telephone number changes for people in their departments. In the same way, we created the view *Admin_User_Dir_Info* to enable the departmental administrators to update the *User_Dir_Info* table for their staff.

In the actual implementation, `personal_info` and `Dept_Pers_Info` are the same program that use a run time switch to select which set of views to use. This switch also enabled an additional command to allow the department administrator to select which person (from their list of staff), they want to edit. In the same way, the `dir_admin` program is the same program with a different run time switch, that operates directly on the base tables instead of the views. This lets telecom staff change anyone's directory information.

One handy bit of fallout from this, is we were able to add the `forward` command to this program, so department administrators can now establish and changing forwarding of mail sent to *userid* `@rpi.edu` for their own people. Previously, this would have required the individual to sign on to RCS themselves, or send a request to the PostMaster for this change. While not in the initial plan for the program, it just sort of fell into place for free.

Departmental Information and Virtual People

We had spent a lot of time developing a system that allows us to maintain directory information for people, and to delegate control of that information to other people. We also needed the ability to handle directory information for departments and other organizations. To do this, we simply create a *Virtual Person* for each department. The *person* was inserted into *Dir_Master* with the appropriate Orgn information, so that control of this *person's* information is automatically delegated to the department administrators (just like any other staff member), and they can update their department's address, home page, etc. This required almost no additional development, since routines already existed to edit and extract all of these directory elements. Since these entries are not actual employees, they are not sent back to the HR system, and simply remain in the Simon system.

Along with the departments we obtained from the Finance system, we also manually created departments. In some cases, this was to allow us to insert ''external'' departments into our existing organizational tree. One example of this is our food service contractor, Marriott. There are a number of Marriott staff working on campus, so we simply created a new organization for them, inserted it into our tree, and assigned (manually of course), the Marriott staff to that department. Since the Marriott employees are not RPI employees, they were entered into Simon as ''Guests'', as seen in Figure 1.

Generating LaTeX and HTML

The final part of the information flow, is of course the actual directory. We actually wanted a couple of different breakdowns of the data, such as a general staff directory (all staff, sorted alphabetically), the

[6] A convention followed in much of the Simon project is that an Oracle *table* is maintained by a `program` of the same name.

department hierarchy, and finally a departmental directory with the staff of each department listed with their department. A couple of simple programs were written to extract the information from the database, and generate LaTeX source code for each part. We also found it convenient to generate HTML code at the same time.

Both LaTeX and HTML have a number of special characters that must be avoided or escaped, in order to have the proper things happen when processed. While handling the general boilerplate is easy, some of text that needs to be generated was directly entered by users. Fortunately, we had faced this problem in a previous project that generated HTML from database descriptions [3] and we were able to re-use the **HTMLize_string** and **LaTeXize_string** functions from this program. These automatically handle whatever quoting is needed.

What we wanted vs what we got

Despite months of planning, meetings with Human Resources, and project specifications that were approved by the President's Council, things came up that we did not anticipate, some of which could not be resolved by the publishing deadline.

Conversion Experience

In the past, we had experienced a bit of ''title inflation'' in the directory. People would send us their titles, and they would be included in the directory. While this might not be their official title as listed with HR, there wasn't any systematic cross checking.

In converting to the new system, we had hoped to switch over to using HR titles exclusively, but we quickly discovered that there was a lot of title drift, and given the time constraints involved, it was not reasonable to get everyone's title changes approved and corrected via HR. We now allow for a manually set title (by the department administrator), and if that is not supplied, we go with the HR title. HR now wants to review the situation, and they are now examining all of the differences in titles between what they have, and what is published in the directory.

Another problem we ran into, was with people who had multiple titles and were in multiple departments. In the first release of the directory, we did not allow this, as HR was not supplying that information. In some cases, people had their titles set manually to *title1/title2* within the limits of the size of the title field. We are investigating ways of allowing multiple titles and departments, but since they will be manual entries, we do not plan on encouraging their use.

Data Cleanup

Part of the conversion process involved cleaning up the directory data. We merged some data from the old directory, but frequently found that the old directory info did not match the HR data for that person. Then we had to go back to the individuals and ask them to verify their information. In the past, the Telecom folks had sent out departmental listings to the departmental administrators and ask for corrections. To this end, we modified the departmental directory generation program to generate a departmental entry and all the staff entries for that department, and a source file that would run that through LaTeX, as well as generate a set of mailing labels. This gave us a nice cover letter to the department administrators explaining how THEY could update the department and staff information via the `Dept_Pers_Info` program, as well as a page per staff member listing each directory field with their current information. This format made it very clear to the individuals what fields were available, since some of fields had not been collected in the past. The department administrators had the option of distributing the individual sheets to their staff, or doing the updates themselves. In any event, these updates were done at the individual or department level, and NOT by Telecom staff.

The other part of the data cleanup involved the campus address. Historically, the campus address was generally three lines, with the department name on line 1, the room number and building name on line 2 and the text ''RPI'' on line 3. Unfortunately, there was no consistency in the entries, making it impossible to generate either building or department directories. (There were about 1200 unique department names when we started, we are now down to 177 departments.) To clean this up, when someone enters a campus address change, they have to select a building from a list of buildings, and then provide a room number. The department is automatically obtained from the information in *Dir_Master* and *Dir_Departments* tables. In addition, we also include an internal database building id, that gets propagated to the HR system (they agreed to modified their tables for this one), and back to Simon again. After giving the staff a chance to correct their addresses, we added an option to `Dept_Pers_Info` that displays each ''unclean'' address, and prompts for a new address. This allowed us to clean up the remainder of the addresses in a very short amount of time.

Intended Results

Despite these problems and some last minute changes, we ended up with a 148 page telephone directory with both alphabetical and departmental listings, as well as a solid directory feed to our **ph** server[7]. In addition to the **ph** database update, we also do a nightly refresh of Departmental Directory web pages *http://www.rpi.edu/AutoGen/deptsum.html* and

[7] Our **ph** server is a package developed at the University of Illinois at Urbana, and has not only a Unix server, but clients for many platforms. It was originally developed by Steve Dorner (sdorner@qualcomm.com), Qualcomm, Inc and is now maintained by Paul Pomes (p-pomes@uiuc.edu), University of Illinois Computing and Communications Services Office.

the list of Institute phones (an alphabetical listing of departments and services) *http://www.rpi.edu/Auto-Gen/univtel.html*. Both of these pages include phone numbers and fax numbers (when available), and an link to the department's web page, if it exists. The department directory also includes a link to a staff list for that department.

In doing the Institute Phones directory, we also needed to include services (for example, Computer Repairs). These are added in to another table, and where possible, we indicate which department "owns" that service. In this way, we can include those services at the top of the staff listing for that department, and also as a sub entry in the Institute Phones Directory. Where we encounter an email address for a person, department or service, that email address is automatically inserted into the web page with a `mailto:` URL.

Another nice thing we gained from the project was the ability to collect and publish "less important" data, such as a person's fax number, their home page, etc. Before the `personal_info` program, it was simply too expensive in terms of staff resources to collect and enter this information into the directory. Now that individuals can provide this themselves, we can include more types of information in the directory. In the case of home pages, we don't include them in the printed directory, on the assumption that anyone looking for URLs will be looking online where we do include them.

Unexpected Benefits

This project had a number of side effects that we really hadn't considered in the initial design, or at least were not of major concern at the time.

One of the first departments to benefit from the new directory system was the parking office. Since we now had all employees listed with their campus address in a consistent format, we were able to generate lists of employees, by building. Rensselaer has a number of parking lots located in different areas of the campus, and staff generally prefer to park in the one closest to their workplace. To figure the parking lot assignments, the parking office used to manually sort these lists of employees. With clean address information in the HR database, this three day project turned into a simple database report.

Another area that improved, was our general system administration management tools. We have been managing the host database, software service contracts, and RCS userids with the Simon database. These tools are now able to automatically include the address and phone numbers of the people involved. This has been very well received in our network operations center, where they can now just enter a single command to look up a local host, and get back not only all the host information, and the name of the owner and system administrator, but their address and phone number as well.

The phone directory also includes a building and departmental abbreviations. These abbreviations are stored in the *Buildings* and *Dir_Departments*, and are automatically used when generating directory pages (or hardcopy), and we also extract them as lists to include in the directory itself. This makes it very easy to change the name or abbreviations used for a building or department and ensures that all parts of the directory use a consistent set of abbreviations.

Another application was driven by a new state law that requires RPI to report the names and address of all employees. While this wasn't a problem for the regular staff, Rensselaer hires a large number of students. HR never bothered with addresses for student employees, as all paperwork was handled in person. Given that we had a path in place to update addresses, it was simple to write a program to extract student address information from the Registrar's files (since we already were loading student data into Simon), and send "address changes" into the HR system.

We continue to find administrative applications that can now benefit from accurate and retrievable address data. One recent example of this is the letter distributed to new students telling them who their faculty advisor will be. We can now include their advisor's campus address, phone number and email address. Previously, this was not practical.

Future Work

There are still some parts of the directory that have yet to be fully integrated into this system, such as the Emeriti/Retiree address list, the Student Clubs and Organizations listing, and so on. It turns out though, that this directory system gives us almost everything we need to support these additional listings. In general, we simply create new organizations, which gives us the ability to delegate control over those organizations to others. We are defining one new construct though, an *Aux_Member* table, so we can assign any arbitrary person in the database as an auxiliary member of an organization. We can include these people when generating department listings, and this will likely be used to handle the problem of people who need to be in more than one department.

While we can delegate the ability to add auxiliary members to department administrators, since we are not using the *Dir_Master* table, a department administrator could not add someone, and then update their address information. This allows us to maintain better control over who can make address changes, yet still allow us to delegate auxiliary membership for specific groups. This will be especially useful for the student clubs and organizations, since student address changes are handled via the Registrar's office (using an earlier and different online system). It will be possible for club officers to maintain the membership lists of their clubs, and then we can generate membership lists that include addresses and status information drawn directly from the Registrar's files.

The really big change to the directory system however, will be migrating the `personal_info` to some sort of WWW interface. General computing at Rensselaer is moving away from Unix workstations, and it appears that the Web will be the best common denominator for user interfaces. We are also faced with changes to the Faims system that will require that the `Dept_Pers_Info` program be converted to a real Oracle Form (so it can work in a Client/Server environment), or be removed. We have a number of other personal information services that are available via RCS that we will also want to make available via a web interface, so we will be exploring how to authenticate web sessions connect to the database during the upcoming year. Fortunately, there seems to be a lot of activity in that area, so hopefully we will not have to roll our own solution.

In an effort to make directory information available to more platforms, we are investigating putting up an "X500 Lite" server [6], as described developed by the Center for Information Technology Integration at the University of Michigan.

References and Availability

All source code for the Simon system is available for anonymous FTP. See *ftp://ftp.rpi.edu/pub/its-release/simon/README.simon* for details. In addition, all of the Oracle table definitions are available at *http://www.rpi.edu/campus/rpi/simon/misc/Tables/simon.Index.html* Directory pages can be seen at *http://www.rpi.edu/AutoGen/deptsum.html* and *http://www.rpi.edu/AutoGen/univtel.html*.

Author Information

Jon Finke graduated from Rensselaer in 1983, where he had provided microcomputer support and communications programming, with a BS-ECSE. He continued as a full time staff member in the computer center. From PC communications, he moved into mainframe communications and networking, and then on to Unix support, including a stint in the Nysernet Network Information Center. A charter member of the Workstation Support Group he took over printing development and support and later inherited the Simon project, which has been his primary focus for the past five years. He is currently a Senior Systems Programmer in the Server Support Services department at Rensselaer, where he continues integrating Simon with the rest of the Institute information systems, and also deals with information security concerns. Reach him via USMail at RPI; VCC 319; 110 8th St; Troy, NY 12180-3590. Reach him via electronic mail at finkej@rpi.edu. Find out more via http://www.rpi.edu/~finkej.

References

[1] C. Mic Bowman and Chanda Dharap. The enterprise distributed white-pages service. In *USENIX Technical Conference Proceedings*, pages 349-359. Penn. State University, USENIX, January 1993.

[2] Jon Finke. Automated userid management. In *Proceedings of Community Workshop '92*, Troy, NY, June 1992. Rensselear Polytechnic Institute. Paper 3-5.

[3] Jon Finke. Sql_2_html: Automatic generation of html database schemas. In *Ninth Systems Administration Conference (LISA '95*, pages 133-138. Rensselaer Polytechnic Institute, USENIX, September 1995. Monterey, CA.

[4] Jon Finke. Relational database + automated sysadmin = simon. Boston, MA, July 93. Sun Users Group. Invited Talk for SUG-East 93.

[5] Timothy Howes. Integrating x.500 directory service into a large campus computing environment. In *LISA IV Conference Proceedings*, pages 125-132. University of Michigan, USENIX, October 1990.

[6] Timothy A. Howe. The lightweight directory access protocol: X.500 lite. Technical Report CITI TR 95-8, University of Michigan, July 1995.

[7] Mark A. Rosenstein, Daniel E. Geer, Jr., and Peter J. Levine. The athena service management system. In *USENIX Conference Proceedings*, pages 203-211. MIT Project Athena, USENIX, Winter 1988.