Accountworks:
Users Create Accounts on SQL, Notes, NT, and UNIX

Bob Arnold
Sybase, Inc.

Accountworks: Users Create Accounts on SQL, Notes, NT, and UNIX

Bob Arnold – Sybase, Inc.

ABSTRACT

Accountworks is a system which allows any employee at Sybase, Inc. to use a web form to create accounts for new employees. Every new hire gets a personal account in SQL, Notes, NT, and UNIX administrative domains. Accountworks also creates initial stub entries in our SQL personnel database. It allows the user to make a number of initial choices for their new employee, including access to popular applications and whether to use Notes or UNIX email. Typically all new accounts are available within four hours after the web form is submitted. The system operates 24 by 365 to support our worldwide infrastructure. When the accounts are created, it guarantees a consistent, unique login, UID (for UNIX), Firstname.Lastname record, and password across all domains. It went into full production in July 1997, and has been used to create 1900 new accounts since then. Because this paper is intended to help anyone tackling cross-domain account management problems, it describes the architecture of Accountworks, the process of building it, numerous design decisions, and future directions of the project.

An Apology, By Way Of Introduction

There are a number of itemized lists in this paper, which will, probably, make for dry reading. However, it is hoped that they will also provide a valuable reference. If, at the beginning of the Accountworks project, we had started with a comprehensive set of issues, it would have helped us enormously. As it was, we had to muddle through as we discovered more and more questions that demanded answers. Given the complexity of the problem we tackled, and the limited space to discuss its solution here, the decisions are at least as important as the technical methods of implementing them.

Hopefully, this paper will be helpful to anyone who tackles similar problems. Certainly other sites will have other needs, and would make other choices. However, it seems likely that many organizations could use similar techniques to solve cross-domain account management problems. The descriptions of the Accountworks feature set, and the reasoning behind all these decisions, should at least serve to illuminate the many questions involved.

In The Beginning, There Was Mud

By early 1997, the process of bringing a new person into the company and putting all their necessary working environment in place was widely seen as a major problem. The infrastructure to support this process had not kept pace with the rapid growth of the company. Although some parts of the process worked well, they didn't always work together. In addition to regular employees, the company brings in student interns, contractors and temps; employees of our distributors and other business partners need accounts too. Everything from getting a phone to setting up super-user privileges for a system administrator was taking far too long, sometimes as long as a month. Sometimes the hiring manager didn't begin the process until after their new person was already at work, which caused the predictable frustrations, phone calls, interrupts, emergencies, and escalations.

The Information Technology (IT) department is responsible for supporting most of this process. We have 7000 accounts in each domain, 15000 hosts, 100 locations around the world, and a WAN with links ranging from 28.8 modems to fibre to VPN. Our call-track system receives 10000 calls per month, many of which are linked to account administration.

In January 1997, a meeting with 40 interested people was held to fix the problems with the new hire system. These stakeholders helped define the overall project goals, and the group rapidly dropped to fifteen participants and a core of ten people.

Project Charter

Our primary project goal was to improve the process of enabling a new employee to become productive as quickly as possible. We took a broad view of this. We knew we would eventually manage the entire account-related life cycle of an employee at the company – we had to look ahead to termination and re-hiring issues. The account creation process had to work for contractors, temps, student interns, distributors and other business partners, as well as full-fledged employees. The charter included looking at, and sometimes re-engineering, other business processes related to hiring.

For example, early on in the project, we briefly considered building a semi-manual account creation process. Hiring one or two entry level staff to do nothing but create accounts would definitely have been cheaper in the short run. Such a solution had obvious disadvantages though, in accuracy, speed, consistency, and data integrity. Furthermore it would still leave the IT organization as a potential bottleneck in the hiring process.

For a number of issues, we simply put documentation on the Accountworks web site. While short of a true one stop shopping solution, at least anybody could go to our web site to begin the hiring process. There, they would find all the necessary instructions, web links, and the Accountworks application itself.

One major change was the role of the Human Resources department in the new hire process. Our HR procedures vary from country to country, and sometimes among business units in the same country. Many of our European and a few North American business units relied on their HR staff to handle or coordinate many aspects of the new hire process, including the initial data entry. Our European IT operations depended on a fully enabled HR record to begin the account creation process. Accountworks required a fundamental business process shift, to make the hiring manager responsible for beginning the new hire process.

The other major process change affected some of the various help desks and systems administration groups around the world. Before Accountworks, half of these organizations were involved in the account creation process, occasionally in some cases and routinely in others. These processes were sometimes clearly defined, and sometimes not. Now it is crystal clear – none of these organizations have to do new hire account creation any more. The burden of the work is squarely placed in the ideal location – the person who cares about it most. And the person who cares, typically the hiring manager, has every opportunity to see to it that the job is done right – all they have to do is enter the correct data on the Accountworks web form.

Political Hurdles

In many respects the project was fortunate. We started with a number of advantages:

- The project was initiated and backed by new top management.
- With very few exceptions, the entire project team reported into the same IT organization.
- Everyone in the company could see the importance of the project.
- We had plenty of motivation – the project was an opportunity to fix our own long-festering problems.
- The core team had the necessary planning, architecture, programming, documentation, and user interface design skills.
- Some related systems, like our HR personnel database and calltrack systems, were SQL-based.
- We didn't have to actually manage these domains, or even coordinate them, we just had to create consistent, unique, new accounts in them.
- A number of other simultaneous IT projects simplified our work:
  - Consolidation of roughly 90 NT security domains worldwide into three NT security domains.
  - Conversion of several MS Exchange email systems to Notes.
  - A UNIX home server consolidation project in Emeryville.
  - The only separately administered NIS subdomain was moved under the central NIS management system.
  - Consolidation of all desktop and laptop purchases into the IT department budget, instead of separate hardware budgets for each department.

We had a few disadvantages too.

The above consolidation projects, and other unrelated work, competed for staffing resources with the Accountworks project. Most of the core members were stretched thin, some of them chronically.

Years of neglect of each administrative domain had left them in a predictable mess. Clean up efforts simplified the project's work, but competed for the attention of project members. (Some clean up efforts were deliberately put off because they weren't required for the success of the project.)

Scope creep was a constant danger. We kept surfacing related issues which also needed to be solved. For each of these issues, we had to decide whether to ignore it, provide instructions and/or links to relevant web sites, or tackle it. Here are a few examples; many more came up along the way:

- How much of the entire new hire process should we really address? What about setting up the new employee's phone? ID badge? Building access? Company credit card? Network drop? Computer? What HW/OS should their computer be? Do they need more than one? (Various strategies were used.)
- Does the locations table in our personnel database reflect reality? (No, but we have to make sure it does.)
- Do we have to guarantee that logins are never re-used? (No.) What about guaranteeing unique UIDs? (Oh no, we didn't think of that, but we definitely have to do it.) Can UIDs be re-used after someone leaves the company? (Yes.)
- How do we handle accounts when we acquire another company? (These have to be handled on a case by case basis, so document a general strategy and put it on our web site.)

Top management originally thought the project would be quick and easy. Significant effort was required to establish a more realistic timeline and staff allocation.

One of our core members was in Europe, a nine-hour time difference from the rest of the team in Emeryville, California. Coordinating our efforts with him was difficult. Two others, including our most important technical person, were in Ottawa; the three hour difference was more manageable. Two of these key participants had to travel to Emeryville for the roll out.

For some of our business units, HR had been responsible for the initial data entry for a new employee. HR staff naturally had concerns about making managers responsible, due to the potential for unclear process ownership and poor data entry. Management was not keen to assume new data entry duties at these locations either. With a lot of work and the backing of top management, we were able to work through these issues. Also, one of our core members from HR traveled to a number of offices around the world to address local concerns before the project rolled out.

Design Overview

What Accountworks does do:

- Allows any employee in the company to begin the process of hiring a new person, including automatic account creation, through an easy web form.
- Creates initial stub entries in our SQL personnel database for our Human Resources department to review and finish processing when all the approvals are received.
- Creates accounts in SQL, Notes, NT, UNIX, and upon request, a number of popular applications. These accounts are typically available within four hours.
- Guarantees unique, consistent login, UID, Firstname.Lastname records and passwords at the time when accounts are created.
- Creates calltrack requests for phone and equipment installation.

What Accountworks does not do (yet):

- Provide a multi-domain password changing tool.
- Provide an authoritative, automatically enforced, guaranteed-to-be-correct-across-all-domains database of what a person's login, UID, and Firstname.Lastname record is supposed to be.
- Manage accounts after they are created.
- Handle account terminations.
- Handle re-hires (because each person is supposed to have only one set of records in our personnel system no matter how many times they have left and returned to the company).
- Handle generic accounts.
- Handle large batch jobs of new hires (this happens when Sybase acquires a company, brings in student interns, or a group of temps).
- Handle account changes, such as moving from one home server to another or moving between Notes and UNIX email.
- Handle login groups (NT Global/Local Groups, UNIX netgroup entries).
- Handle permissions groups (Notes groups, NT Global/Local Groups, UNIX group entries).
- Handle mailing lists (Notes mail groups, UNIX group aliases; luckily, auto-generated location-based .UNIX email lists were already being handled by another application).

We lumped the last three items into the "general group problem," and decided that managing groups was too hard to do within the project deadline.

We built an application with six major components:

1. A web server front end, accessible to any employee, with instructions and links to related sites, and the web form which allows them to create accounts for new hires and to check the status of their requests.
2. The Accountworks SQL database, with the necessary knowledge of our environment to make intelligent decisions based on user input.
3. A large set of client tools to create accounts in SQL, Notes, NT, and UNIX; create calltrack requests for phone and computer setup; automatically grant access to some applications; send email or open calltracks to request access to other applications; report status back into the Accountworks database; and open service calltracks if any of these clients fails.
4. The Extraction SQL database which receives login, UID, Firstname.Lastname and other data from 34 data sources, and merges all that data into one table in the Accountworks database.
5. A set of programs to extract and parse the data from the 34 sources for the Extraction database.
6. Three client applications to administer Accountworks tables and help debug problems. These are accessible only by certain support staff.

Account Creation: A 12-Step Program

When someone wants to bring a new employee into the company, they go to the Accountworks web site. The first screen they see contains information, instructions, and a link to the Accountworks application itself. When they click on the link, the Accountworks data form comes up. Figures 1a and 1b detail the subsequent actions.

The **login** is used when creating accounts in SQL, NT, and UNIX. It is also stored as a "shortname" field in Notes.

The **Firstname.Lastname** record is used to create the Notes access key, which consists of Firstname,

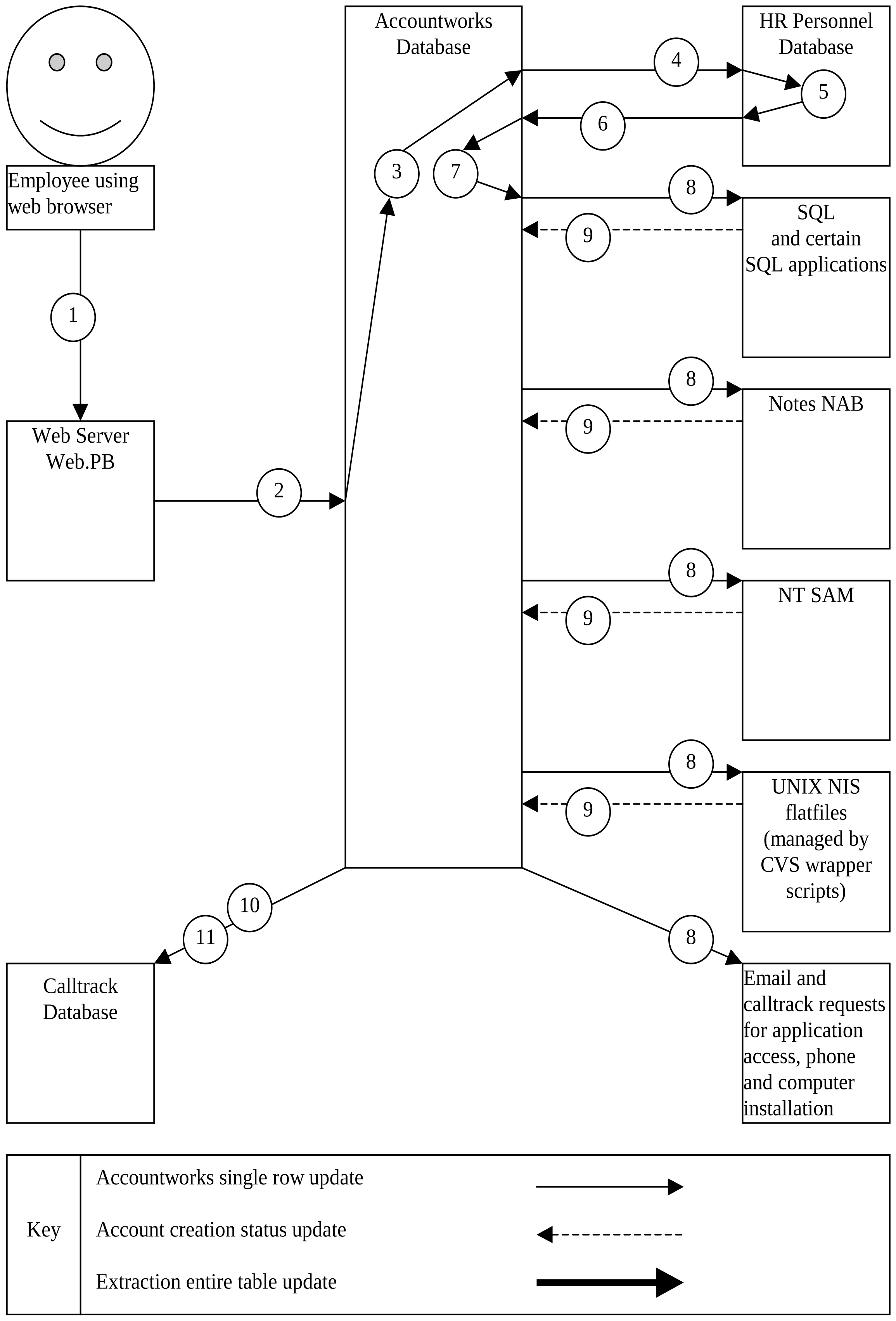
Employee using web browser
1
Web Server Web.PB
2
Accountworks Database
3
4
HR Personnel Database
5
6
7
8
9
SQL and certain SQL applications
Notes NAB
NT SAM
UNIX NIS flatfiles (managed by CVS wrapper scripts)
10
11
Calltrack Database
Email and calltrack requests for application access, phone and computer installation
Key
Accountworks single row update
Account creation status update
Extraction entire table update

Figure 1a: Account Creation Process – Web interface, Accountworks database, and account creation tools.

1. The user enters the data for their new hire into the web form, and presses a Submit button. The data is written back to the web server.
2. The web server feeds that data to a stored procedure in the Accountworks database to begin the process of creating accounts.
3. The Accountworks database automatically generates a unique **login**, based on the new hire's name.
4. That **login**, together with other relevant information, is fed into a stored procedure in our personnel database.
5. The personnel database creates a new **emplid** (employee ID number) and a stub entry for the new hire.
6. The **emplid** is returned to the Accountworks database.
7. The Accountworks database tries to generate a unique **Firstname.Lastname** record. If it fails, it returns a web form to the user telling them what happened, and asking them for a different Preferredname (nickame) and a Middlename. When that is supplied, the process starts over, repeating until it succeeds.
8. Now the Accountworks database has all the information it needs to create an account in each domain. It uses the appropriate backend tools to do that for SQL, Notes, NT, and UNIX. It also opens requests in our SQL calltrack database for application access, phone setup and computer installation.
9. The account creation tools for each domain report back to the Accountworks database, describing their progress. Normally this all goes well, and each domain sets a "Complete_Success" status for itself.
10. If any of the backend tools report a "Complete_Fail" status, Accountworks opens a trouble ticket in our calltrack database for human intervention.
11. If 24 hours pass, and the Accountworks database sees that one of the domains has not reported "Complete_Success," it opens a trouble ticket for that domain.
12. The requestor, hiring manager and an optional third contact may check the status of the requested accounts at any time.

Figure 1b: Twelve steps to creating a signon.

Lastname, and an optional Middle_initial. The same data is stored in comment fields in the NT SAM and the UNIX passwd map. It is also used to create "login: Firstname.Lastname@notes-gateway" records in the UNIX aliases map for new hires who will be using Notes as their primary email system.

Troubles Come In Threes

There were three major problems that required solutions. Guaranteeing unique names for use by all systems was one. To solve this, we created the concept of an 'access key', which is an abstraction of the name which must be unique within a given system, and further must also be unique across all systems. Examples of 'access keys' are the UNIX logins from the NIS passwd map, email aliases from the aliases map, mailing list names in both SMTP and Notes, Notes ACL groups, NT username, and Notes login. We ended up with 34 different systems that needed to be synchronized by this concept of an 'access key.' A significant, beneficial side effect of this process was the identification of the systems and the ability to simply track (but not control) them from a single table.

Every evening, a set of scripts and stored procedures gathers access key data from the various sources, parses it into fields, and loads it into the appropriate tables in the Extraction database. Each data format, such as passwd and aliases file formats, has its own Extraction table. An hour later, we merge all the Extraction records into the "access_key" table in the Accountworks database. Each record in the "access_key" table knows its original data source and when it was first inserted.

When generating login and Firstname.Lastname guesses, Accountworks checks the "access_key" table to see if its guess is available. If so, that ends the guessing game. Otherwise, it moves on to the next guess. This is how we guarantee that any access key we generate is unique.

For example, our Extraction "passwd" table has four data sources. We gather /etc/passwd files from three important and representative UNIX hosts. These files only contain the typical system accounts like "root," "bin," etc. The fourth source is the flat file for our NIS passwd map, which contains 7000 records. It includes personal accounts for most of our employees, some generic accounts, but no "root" account.

Thus, the Extraction "passwd" table has three "root" records. All three "root" records are merged into the Accountworks "access_key" table. The "access_key" table also has a "root" record from the NIS aliases map (to forward mail from "root" to "postmaster"). A simple query against the "access_key" table will show us four data sources which use the "root" access key. If we ever hire a "Jennifer Root" or "Robert Oot," any one of these records is sufficient to keep us from creating a "root" login for them.

The second problem was collecting and modeling the data required to correctly map people to the correct login domains and home servers. It turned out that much of the information required to do this mapping, such as home server names/domain, office locations, and city to country mappings, existed in various databases, spreadsheets, and in many cases just a person’s head. Often the information was incomplete or inconsistent, and there was not a known master copy of the data. At one extreme, some offices have no home servers of any sort. At the other extreme, our Emeryville headquarters has perhaps 50 UNIX home servers, and numerous NT and Notes home servers too. Also, our personnel database has records for inactive locations as well as active ones, and we discovered that the locations data had not been well maintained. Once again, a significant side effect of automating the account creation process was the consolidation and cleanup of this required mapping data.

For each active location, we mapped three home servers: Notes, NT, and UNIX. A small office might have only a few PCs, or a few Suns. If a real local home server could not be identified, we picked a home server in a more central office. The WAN topology dictated this choice, so we had to get accurate maps and information about this too. For example, our Dallas office has no UNIX boxes, so UNIX accounts for Dallas new hires were mapped to a Sun server in Chicago, the nearest WAN hub. Ditto for Notes. But the Dallas office does have an NT home server, which we used for the Dallas NT mapping.

Somewhat larger offices might have a few home servers, owned by various organizations. This forced us to create an organization pick list. For example, the technical support staff might be on one server, and everyone else might be on another. We created a “Tech Support” organization, and mapped new hires

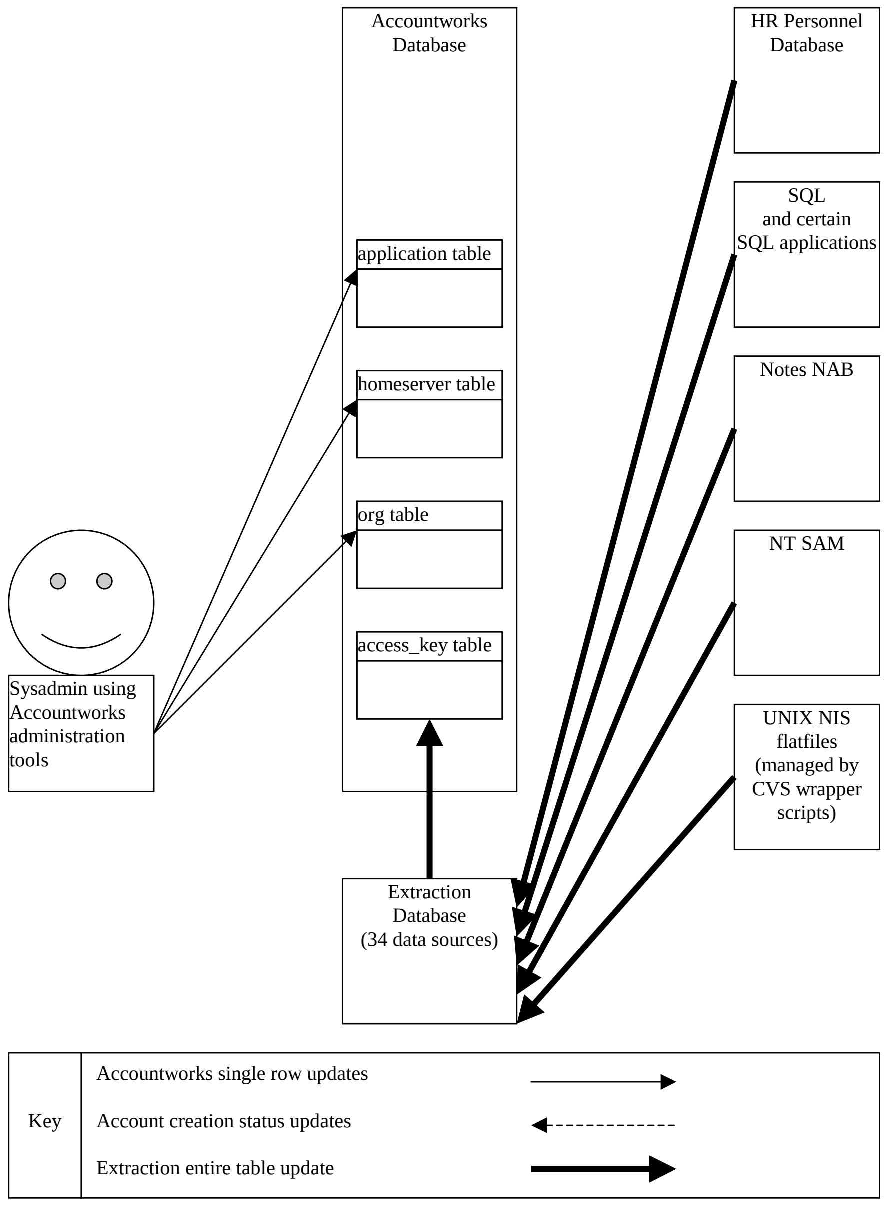
Accountworks Database
application table
homeserver table
org table
access_key table
HR Personnel Database
SQL and certain SQL applications
Notes NAB
NT SAM
UNIX NIS flatfiles (managed by CVS wrapper scripts)
Sysadmin using Accountworks administration tools
Extraction Database (34 data sources)
Key
Accountworks single row updates
Account creation status updates
Extraction entire table update

Figure 2: Extraction Database, Administrative tools.

for that location to their server; all other new hires would go on the other server.

Large offices might have many home servers, and even some departments are split between various servers. "IT Systems Administration" and "IT DBA" are on different UNIX home servers in our Emeryville headquarters. So we had to add a second level to the department tables. Appropriate rollups are done for sub-departments if someone chooses a department which doesn't have a specific home server at that location.

Although locations are hardwired to our personnel database, Accountworks "organizations," surprisingly, are not. Trying to track all the re-organizations and changes in department names and numbers had already doomed an earlier project to failure. Two of our core members had worked on that project, and kept us from making the same mistake. We decided that no matter what the official name of the department was, people could always identify with departments like "Sales" and "Engineering." This has proved a successful strategy.

It was a big help to know that the general direction had recently changed from splitting off new UNIX home servers to consolidating them. Even so, it took a surprising amount of time to come up with a mapping that would work. There were a number of reasons for this. One major factor was all the required research on the WAN topology and which locations were active or coming on line. Another was that it was hard to explain, or even remember, that we only needed to know where new hires would go now, not where everybody had been put in the past, and the new application would not move anybody's old home directory. In a few cases, we had non-UNIX machines providing NFS home services. But mostly, we had a delicate balancing act between adequately modeling the real world, and keeping the organization picklist small. We discovered the problem was complex enough that it was easier to interview key local sysadmins than request data via email. Our development centers, most of which have multiple buildings and a long history of creating a UNIX home server for every new department, were the hardest to model.

The third problem was the design of the request web form. Because someone might use it only once, we tried to make it as easy as possible to fill out. We minimized the amount of required information, and provided defaults, auto-populated fields, radio buttons, and pick lists wherever we could. We have only 16 input fields:

- First name*
- Preferred name (nickname)*
- Middle initial/name*
- Last name*
- Organization*+
- Location*+
- Notes/UNIX email*
- Start date
- Department number
- Job code (corresponds to a job title, not a specific opening)
- Company code+ (corresponds to country or business unit)
- Cube/Office
- Manager's login
- Alternate contact login (optional)
- Contact phone number
- Application requests+ (optional)

To make the web form easier to maintain, we drive pick list and checkbox creation with tables; these are tagged with plus signs (+) above. The fields marked with asterisks (*) are used for account creation; the others are necessary for personnel, contact, and equipment installation purposes.

What's In A Name?

A tremendous amount of time was spent on design issues surrounding names. Some of these decisions were easy, but others were not. Here are our choices, as they stand today:

- What names do we ask for, and how do we ask for them? We ask for four separate fields: Firstname, Preferredname (usually a nickname), Middlename, and Lastname. Our personnel system already required these fields, and it was impossible to reliably parse them out of a single "Name" field because so many firstnames and lastnames have embedded spaces in them (like "Mary Jo" or "van Beethoven").
- Should we ask for a middle name, not just a middle initial? Yes. When we initially put Accountworks into production, we only asked for a middle initial. Our Notes domain was the gating factor – it only allowed middle initials when creating Firstname.M.Lastname accounts, and no other system really needed a middle name. Since then, we have found it useful to accept a middle name for human eyes and paycheck records, so the middle initial field became a middle name field.
- Should the login and Firstname.Lastname record be the same across all systems? Yes, because that is simpler for everyone. (The answer could have been "No." Powersoft, before being acquired by Sybase, made sure that a person's PC login, UNIX login, and dialup login were all different. This was done for security reasons.)
- Should we use the SQL/NT/UNIX login as the access key for Notes, or the Firstname.Lastname record as the access key for SQL/NT/UNIX, so we could have a consistent access key across all domains? No, our Notes installation was too hard to change to use logins, and it was technically impossible to use

Firstname.Lastname records as SQL/NT/UNIX logins.

- Can we enforce correct capitalization of names? No. There are simply too many possibilities. For one person, "de Silva" might be correct, and for another it might be "De Silva." However, we do assume that two or more capital letters in a row are a typo, and convert them to an initial cap followed by lowercase. This mostly works, but not for Firstnames like "PT (Barnum)."
- How should we handle European or Asian character sets? Because the 7bit ASCII character set guaranteed portability across all four domains, we decided to convert 8bit alphabetic characters to 7bit, and not support double-byte characters.
- What special characters did we have to allow in the name fields? Hyphens, single quotes, periods, and spaces, for names like "Smith-Jones" and "O'Malley," "Joanie Caucus Jr." and "Peggy Sue."
- How do we make sure that people don't make typos, or use all capital letters? We can't. But we did add a confirmation screen to encourage users to make sure their data was correct, which helped a lot.
- How long could a login be? 8 characters. The UNIX login domain drove this decision.
- Could the login include the hyphens, periods, spaces, and singlequotes we accepted in the name fields? No. Hyphens had historically been generally discouraged in logins, and the others would lead to all sorts of technical trouble in many of the domains.
- Could the Firstname.Lastname record include the hyphens, periods, spaces, and singlequotes we accepted in the name fields? Periods, no; the others, yes. Firstnames like "E.T." would turn into "E.T..Phone-Home" which would run into trouble because of the double period.
- What are the valid characters for a login? Lower case letters and digits.
- How should we generate a Firstname.Lastname record? We actually try Preferredname.Lastname, then Preferredname.M.Lastname (if we got a Middlename). (We don't try the actual Firstname field, because many people use that only for legal and paycheck purposes.) If both of those are already taken by another employee or generic account, we ask the user to choose a different Preferredname or provide a middle initial. This is not a pretty solution, but it's as friendly as we can be, since the Firstname.Lastname record has to be unique. In such cases, we had to rely on the user consulting with their new employee.
- Should we let the user choose or request the login for their new hire? No. It was more work, time was very tight, and we knew of cases where inappropriate login names had been chosen. We decided to automatically generate a unique login instead. This was a controversial decision, which continues to raise occasional questions.
- How should we generate the login? By using lowercase combinations of Preferredname, Firstname, Lastname, all the initials, and an appended digit if necessary.
- When we are generating the login and Firstname.Lastname record, should our checks for access key uniqueness be case-sensitive? No. If we somehow had used "jim.smith" already, a new "Jim.Smith" would have to choose another Preferredname or provide a middle initial.
- Do we have to guarantee that logins, UIDs, and Firstname.Lastname records will never be re-used after a person leaves the company? No, it's too late, it has already happened a lot. This was easier to implement, and is friendlier to the new hire because their preferred name is more likely to be available. The downside is that many in-house applications use login as a key (under the assumption that a login name would not be re-used), so occasional tweaks to these applications are sometimes required to keep the new hire from acquiring the attributes and history of an ex-employee.
- Could a new login name be the same as:
  - An existing login? No. Login names must be currently unique. (They do not have to be historically unique – see previous bullet.)
  - A mailing list name? No. Logins and mailing lists overlap in UNIX. If "all" is a mailing list, an "Allison L. Lucky" should never get "all" as a login.
  - A mailing list member? Yes. A member is either an external address (which is not a problem), or internal login name or mailing list name (which we already guarantee against conflicts.)
  - An NIS group? Yes. These namespaces do not overlap.
  - An NIS netgroup? Yes. These namespaces do not overlap.
  - An NIS hostname? Yes. A potential danger was that some local sysadmin groups used "login" as a hostname for DHCP clients. This procedure was changed to "login-pc" to avoid namespace overlap.
  - A Notes group? No. A Notes group can be used for permissions purposes and/or mailing lists, so the Notes group namespace can overlap with "Firstname Lastname" Notes ID's.

Security

We had to address a number of security issues, of course. Other security choices could have been made – there are tradeoffs for all of them. All of these security design decisions were implemented in the initial roll out; only two of them were changed based on our real world experience.

- Any employee can use Accountworks. More precisely, any person in our HR database can use it. We initially planned to restrict access to managers, on the theory that only managers would hire people. However, it turns out that in various parts of the company, technical/team/project leaders, supervisors, administrative assistants, and even contractors and outsource vendors bring in new employees. For some of our business units, HR had been responsible for beginning the hiring process for other departments. It soon became clear that the headaches of managing the authorization process would outweigh any potential security benefits. Besides, we were trying to enable the hiring process, not create another bottleneck.
- To use the Accountworks system, employees need a web browser that supports HTTPS, and they must enter their UNIX login and password.
- The requestor is allowed to request access to nearly twenty widely used applications for the new hire, via checkboxes on the web page. In some cases, these are granted automatically, and some are granted for new employees of the right department. Others require review, so calltracks or email messages are generated to initiate these requests. The checklist does not include super-user privileges of any kind.
- A new employee's initial password is created by a random password generator. These random passwords set a good example for the new employee. However, they are also ugly, which encourages the new employee to change it (good), or write it down (bad).
- The same password is used when creating accounts on all systems. (After the new employee actually starts work, they can change their password on each system, of course.)
- The password, access key, and other account data are transported over the WAN in cleartext on well known ports, from the central Accountworks database to the machines controlling each administrative domain. This is in keeping with our general security model. However, the Accountworks machines are especially attractive targets. For that reason, login access to them is limited, file sharing access via SMB or NFS is limited or turned off, and they were among the first boxes to be attached by our Datacom group to switched ethernet hubs to make packet sniffing harder.
- Naturally, the new hire person needs to know their initial password. That means the password has to be stored on-line, so it can be retrieved. Very few support staff have direct access to the machine and database which stores the new hire passwords.
- Who should be responsible for retrieving the password and passing it to the new hire? Ideally, the person who cares the most about putting the new hire to work. So, the account requestor, hiring manager, and an optional third contact can log into the Accountworks status web page, check the status of the accounts, note their new employee's password, and pass it on to them.
- Every status page access is logged.
- If a prospective new employee eventually decides to turn down an offer from the company, our HR department initiates an employee "decline" procedure which removes all their Accountworks records and system accounts.

One major security dilemma centers around Notes. Access to a Notes database requires a Notes ID. This consists of a Firstname record, Lastname record, an optional Middle_initial record, a password, and a Notes ID file which contains the name records. Unfortunately, in the real world, people do forget their passwords. For many security systems, the standard fix is to have support staff reset the password, give the forgetful party their new password, and then tell them (or force them) to change it. Unfortunately, this has an ugly side effect in Notes. If the user has used Notes-encryption on any of their files, they can't decrypt those files any more, because resetting the password makes the Notes ID file out of sync with the database. Thus, Notes forces organizations to either a) abandon all encrypted files with forgetful owners, or b) store all Notes passwords so sysadmins can help forgetful owners retrieve their files. Long before Accountworks came along, our Notes administrators were storing the original Notes ID file, but not in the Notes default location. The project chose to continue that practice.

We are aware that complex systems are very hard to secure, and that a system's security is only as strong as the security of its weakest subsystem. Clearly, Accountworks is complex, and has many subsystems. The security implications are obvious.

Trouble In Paradise

There were, of course, a number of problems with the system when it was first launched, in spite of a lot of testing prior to going into production.

Testing is a tricky business. Using an isolated test environment is great for protecting the production systems, but sometimes it's hard or even impossible to recreate a realistic copy of a production system in a test domain. We used various hybrids of test and production environments, which caused various problems.

For the UNIX domain, much of the testing was done against production systems. We got complaints from the user community about "Micky Mouse" and other silly passwd map entries created by the test data.

This wasn't a problem for the Notes domain, because we were unable to totally automate the creation of accounts by rollout, partly because of difficulties with the C language API for Notes. One person still had to press a few buttons to get the accounts created, and they exercised good judgement, so Notes users never saw the "Micky Mouse" accounts. On the other hand, the Notes process was slower than the others because human intervention was required.

For NT, most testing was done against an isolated test domain. But our production system has three NT security domains, and we realized shortly after we went live that we were only able to create accounts successfully in one of them. It took some time to get this fixed, so a number of NT sysadmins found themselves creating these accounts by hand. This didn't win the project team any brownie points.

Shortly after the rollout, it was realized that someone could create an account, get its password, use the first new account to create a second new account, and so on. Even worse, this method would allow someone who was leaving the company to create permanent dial-in access for themselves. So, we restricted Accountworks to accounts with fully activated HR records, and implemented a time hold before the password is released.

We initially designed the system to remove the password as soon as it had been viewed by the requestor, hiring manager, or the optional third contact. This caused more headaches than it was worth after rollout – too many users didn't actually remember the password they had seen, which meant phone calls to get the password reset, by hand, for each domain. We now have an automated routine which deletes the password after a period of time.

We had decided that the table of UNIX home servers would be validated against a comment field in our NIS hosts map, which had historically been maintained by our sysadmin staff. This turned out to be a bad idea, because responsibility for maintaining the validation data was too diffused. Each UNIX sysadmin is responsible for certain home servers, but because they didn't set one up very often, they sometimes didn't put the correct home server information in the NIS database. In such cases, the UNIX account creation script would refuse to create the account. We decided to turn off this validation, and focus the responsibility for maintaining the table of UNIX home servers on the Accountworks administrators.

We could have saved ourselves a lot of trouble if we had rolled out the initial version with a confirmation screen. Our internal marketing efforts focused on the automated account creation benefits, not on the need for accurate data. Under the circumstances, some people entered test data just to see how well it worked. Other people entered real new hires, but they weren't particularly careful since it was just for system accounts, and they didn't know how much work it would be to fix the problems by hand. We added a confirmation screen to remind the user that they were creating real HR records, and to check their work before submitting the request. This helped a lot, but typos and incorrect data are still an occasional problem.

The various problems we had with the system at rollout had a domino effect. Some requestors would check the status, see that there were problems, and enter their new hire again. Even in the best case, we had to decide which records to delete from all systems. Other times, a sysadmin would fix a problem in one domain, without coordinating with other sysadmins or the Accountworks team. Also, the second request would often fail because the application could not create a unique Firstname.Lastname record. It doesn't matter how unusual or uncommon someone's name is – once their name is entered into the system, it's taken.

Support Complexities

Since Accountworks creates initial stub records in our HR database, this has relieved HR from some data entry work. But it has also created problems. HR staff has to delete records for prospective employees who never actually end up working at the company; this happens more often than it used to. HR staff has to correct bad data, such as typos in names, and delete records entered by people "just trying the system." This problem was particularly bad before we added the confirmation screen. Finally, HR staff have to delete test records entered by Accountworks application development and maintenance staff. Through all of this, our HR staff have been unusually patient and understanding.

Although the core technologies are SQL and web-based, many tools were used, particularly in the account creation and extraction scripts. Some of these are publicly available, including Perl [9], Sybperl [10], CVS [11], and the Systems Administration Environment [12]. Others are commercial: PowerBuilder, Web.PB, Transact-SQL, Adaptive Server Enterprise, Replication Server, and Open Server (Sybase, Inc.), FINAL (FastLane Technologies Inc.), Notes and NotesPump (Lotus Development Corp.), Netscape Enterprise Server (Netscape Communications Corp.). A third group comes with other products: Bourne shell and friends (with UNIX), and isql (with Adaptive Server Enterprise). The diversity of the domains required a very diverse toolset.

The staff required to support Accountworks is small. Occasional operational problems can often be solved by junior support staff. Maintenance of organization, home server, and application tables requires minimal effort by trained staff. However, improvements and occasional problem debugging still require a diverse set of high skill levels. As of this writing, we have half a dozen more or less irritating bugs. None of them are critical, but most of them require a high skill level to fix.

Lessons Learned

When architecting the Accountworks application, our primary concern was data integrity. We knew all too well how messy our account domains were. If there was a way to foul up our namespaces, we had done it. We had been through numerous "final cleanups" before, but these heroic efforts were largely wasted without an automated system to keep the account domains in sync.

Therefore, we actively resisted statements like "We'll never hire a Robert Oot" or "That problem will never happen." Murphy's Law had struck far too often. The Accountworks database is highly normalized, with many integrity constraints. Wherever possible, we have tightly coupled our personnel database with Accountworks, using direct replication of tables. Entity relationships were rigorously defined with a conceptual modeling tool, which was then used to autogenerate the physical database structure. The web form is designed to minimize the possibility of bad data entry. Although we initially had a number of troubles around the edges of the application, the core database structure is clean and rock solid.

The SQL strategy has been a major win, because it enabled us to do this. In combination with several other projects, SQL is becoming the glue that ties our various management systems together.

Although Accountworks does not provide an automated system to keep accounts in sync, it is still a major step forward. New hires had been the major source of inconsistent account data. (The other three sources are rehire accounts, generic/system/test accounts, and human error.)

One concept which has been difficult to communicate to our user community, and even to our immediate coworkers, is that we still have no authoritative place to go to find out what someone's login or Firstname.Lastname record *should* be across all domains. Even the project architects didn't realize this problem until shortly before rollout, and we are a long way from having it completely fixed.

New account data is guaranteed to be consistent and unique only at the time it is created. The primary domains are still separately administered. Accountworks does not manage any of them. Thus, Accountworks is merely a multi-domain account creation tool, a glorious "adduser," if you will. Nothing prevents authorized personnel from changing someone's SQL login, or the Firstname.Lastname record we keep in the NT SAM, or giving them a second personal UNIX account, or several entries in the aliases map. When someone changes their name, when they marry/divorce for example, every system has to be changed accordingly, by hand. One consideration is that access to old encrypted Notes documents is impossible for someone who gets a different Notes ID cut for them with their new Firstname.Lastname record.

The Extraction database is downstream from the personnel, SQL, Notes, NT, and UNIX account management systems (see Figure 2). Because of this, it can determine which domains are using which access keys, but it can't manage the account domains in any way. Except for the personnel data, it can't even tell, programmatically, which human beings (if any) are attached to which records. It can't tell if someone has an account or what its access key is. The lack of an automatically enforced authoritative account data system has proven to be a major headache.

Our ultimate goal is what we are now calling "Datamart." This project will define a set of authoritative data sources. To enforce that authority, we will automatically copy data from the authoritative sources to all downstream systems, including SQL, Notes, NT, and UNIX. When the Datamart project is complete, Accountworks will still be a front end to the various authoritative data sources.

Oh, Happy Day!

Everyone is quite happy with the progress to date, in spite of the initial rollout problems and remaining work. Our user community seems to have forgotten how far we have come – Accountworks is just part of the common toolset now. Sysadmins and help desk staff still have rehire, termination, and generic account issues to deal with, but these are much less disruptive and time consuming than our old new hire crises used to be. Naturally, many people can see ways to improve the system, but overall it functions smoothly and in many cases problems are fixed before the user even notices.

Finally, we have learned a lot. We have surfaced hidden problems, identified poorly designed systems, and examined dirty data sources. We are tackling them with various strategies. Although we still have a long way to go, we know where we are going and have a pretty good idea of how to get there.

Other Account Management Systems

Because of the need to integrate the administration of the four primary administrative domains (SQL, Notes, NT, and UNIX) with our personnel system, on a global basis, we were sure that no commercial product or public domain tools would meet our needs. An in-depth examination of one commercial product, and technical meetings with other a few other vendors did not turn up anything we could use.

Account management solutions have been frequently published in the Large Installation and Systems Administration (LISA) conference proceedings. Eighteen papers were published on this topic in the first four years, and twenty-three total so far. Their requirements and methods, not surprisingly, were mirrored in many ways by our later work. A few quotes will illustrate what we have in common. The very first of these papers says:

"The solution at Athena was to create a central database of user information. The database is implemented in RTI Ingres and contains data on our users, courses and projects, clusters, the local systems, such as password files and mail aliases, are propagated from the master system several times a day. [. . .] For security reasons, the database resides on a restricted machine and can only be accessed directly by privileged users. Users and administrators access and modify the data through various utility programs." [1]

A centralized, secure, master SQL database, modelling our user community's needs, and accessible via external utilities – this summarizes some of our basic ideas nicely.

From the second LISA conference:

"We have (1) established a centralized Network Information Registry, (2) established . . . policies . . . and (3) designed a relational database to integrate the various administrative databases (including several Yellow Pages maps) and to reduce duplication of information. . . . [W]hen a new account is created, the loginname and uid are checked for uniqueness in the NIR as well as in the YP passwd map and /etc/passwd file entries." [2]

The requirement for unique logins and uids, compared with multiple sources of this data, was critical to our own success. Again, we are following in other footsteps.

Two years later, the LISA proceedings contained this quote, which we could have taken almost word for word:

"The system selected had to meet several criteria, including:
• Centralized data storage
• Machine and vendor independence
• Flexibility in data to be stored
• Minimal changes to existing software
• Automated account installation
• Easy recovery from crashes
• Automated account deletion
• Simultaneous access for multiple users" [3]

Finally, the AGUS system [4], had we been aware of it, might have formed a foundation for some of our work. Here is the key quote:

"We wanted to use the same system to create accounts on UNIX, VMS, and Novell based networks. The system should also be designed in such a way that it is simple to add additional system types to the configuration. For example, if the University decides to support user accounts on HP MPE systems, it should be relatively easy to extend AGUS to handle account creation under MPE." [4]

Here we have an extensible architecture which supports multiple non-UNIX operating systems. AGUS also embodies many of the design elements of earlier systems. For better or worse, it simply never crossed our minds that anything might already exist which came close to meeting our requirements, or which could be tailored to meet our needs with less work than building something from the ground up.

And, in the end, that is still true. The major differences between AGUS and Accountworks are:

- Trained support staff define account data prior to activation for AGUS; Accountworks builds account data on the fly.
- Both AGUS and our old system required a prior personnel record to create an account – this was one of the major bottlenecks that the Accountworks project had to fix.
- AGUS users request that pre-defined accounts in selected domains be built and activated; Accountworks users request that brand new accounts be created in all domains. Accountworks users also have to give enough information to make this possible.
- AGUS supports UNIX, VMS, and Netware on a few networks. Accountworks supports SQL, Notes, NT and UNIX; two email systems; and over 100 locations worldwide.
- AGUS is mostly written in C with a bit of Perl; the core of Accountworks is SQL based although many other tools were also employed.

For Accountworks, AGUS might have been able to help with the tools to build the UNIX accounts, although that was one of the easiest parts of the project. However, we still would have had to build the user interface; the database of logins, UNIX UIDs, and Firstname.Lastname records to guarantee uniqueness; and the intelligence necessary to configure accounts properly for each location and department.

Availability

Accountworks is not freely available. The company is interested in deriving value from this project. Please feel free to contact the author at rca@sybase.com for the current status of this effort or any related questions.

Roll Those Credits

Thanks to Paul Riddle, Paul Danckaert, Jack Seuss and Rob Banz for their email and conversations about AGUS. They provided useful information on the current status of the AGUS system.

Because of the complexity of the business processes and computer systems we were changing, many skill sets were required. Sixty or more people were involved in the implementation of Accountworks in one way or another. This core group was deeply involved with the design decisions and implementation:

- Jim Leask, Sybase Professional Services: Accountworks and SQL database architect, PowerBuilder Accountworks maintenance GUI tools, NT account creation and access key extraction scripts.
- Bob Arnold, Tools and Architecture Group: Accountworks architect, UNIX account creation and access key extraction scripts, NIS domain cleanup.
- Celeste Barker, IT Customer and Quality Services: Project Management, customer requirements.
- Jill Furman, Human Resources: Human Resources requirements and business processes.
- Bruce MacDonald, Tools and Architecture Group: NT requirements and planning.
- Eric Mittler, Team Notes: Notes account creation tools and access key extraction scripts.
- Chris Osterdock, Application Technical Services: DBA, SQL account creation tools, SQL and application access key extraction scripts.
- Geurt Schimmel, European Information Systems: Web-based Accountworks support tools, UNIX and NT account creation tools.
- Marcy Shaffer, Human Resources Operations: Web interface and calltrack programming.
- Sue Tran, Human Resources: Personnel operations and problem tracking.
- Shel Waggener, Response Center: Project sponsor, customer requirements.

References

The first four references have been cited in the paper. A few others of interest are also listed; references [7] and [8] are interesting because they see account management as part of a larger problem set.

[1] Janet Abate. "User Account Administration at Project Athena." *Proceedings of the Large Installation System Administration Workshop*, 1987.

[2] Deb Lilly. "Administration of network passwd fies and NFS file access," *Proceedings, Workshop on Large Installation System Administration*, 1988.

[3] David Curry, Samuel D. Kimery, Kent C. De La Croix, and Jeffrey R. Schwab. "ACMAINT: An Account Creation and Maintenance System for Distributed UNIX Systems." *Conference Proceedings, Workshop on Large Installation System Administration*, 1990.

[4] Paul Riddle, Paul Danckaert, Matt Metaferia. "AGUS: An Automatic Multi-platform Account Generation System." *Proceedings of the Ninth System Administration Conference (LISA IX)*, 1995.

[5] Henry Spencer. "Shuse: Multi-Host Account Administration." *Proceedings of the Tenth System Administration Conference (LISA X)*, 1996.

[6] Henry Spencer. "Shuse At Two: Multi-Host Account Administration." *Proceedings of the Eleventh System Administration Conference (LISA XI)*, 1997.

[7] Dr. Magnus Harlander. "Central System Administration in a Heterogeneous Unix Environment: GeNUAdmin." *Proceedings of the Eighth System Administration Conference (LISA VIII)*, 1994.

[8] M. A. Rosenstein, D. E. Geer, Jr., and P. J. Levine. "The Athena Service Management System." *USENIX Conference Proceedings*, Winter 1988.

[9] Larry Wall, Tom Christiansen, Randall L. Schwartz. *Programming Perl, Second Edition*. O'Reilly & Associates, Inc., 1996.

[10] Michael Peppler. "Michael Peppler's Home Page." http://www.mbay.net/˜mpeppler. This web page contains links to Sybperl documentation, source, and related information.

[11] Pascal Molli. "CVS BUBBLES." http://www.loria.fr/˜molli/cvs-index.html. This web page contains links to CVS documentation, source, and related information.

[12] Bob Arnold. "If You've Seen One UNIX, You've Seen Them All." *Conference Proceedings, Workshop on Large Installation System Administration*, 1991. See also ftp://ftp.uu.net/usenet/comp.sources.unix/volume28/saenv-5.01.