Arizona State University is located in Tempe, Arizona. It has 19 colleges and schools on three campuses with an enrollment of over 47,000 students.

Since most date and semester fields were stored on ASU’s Student Information Systems database without the century, the university’s heavily date driven system will not function correctly as the new century approaches. Basic functions such as Admission, Registration, and Financial Aid will be unusable as 2000 approaches. Solving this problem has snowballed into one of the largest application projects Applications and Consulting Information Technology has ever undertaken.

ASU has been working on the project since 1994 and ten programmer/analysts are currently working to complete it by the end of 1998. Project coordination, consistent and efficient coding, and elimination of duplicate work has been a challenge.

This presentation will summarize the project goals and actual progress, concentrating on encountered problems and their workable solutions.
2000 and Beyond: Getting Ready for the Century Change

The Problem
As the century change nears, many computerized systems will fail because the century is not stored on date fields. Since space was a premium and software was thought to be short-lived, the century was not a consideration. However, that short term software HAS chugged along quite efficiently and correctly for more years than expected and as the millennium approaches, calculations that have worked for 20+ years will be incorrect. For instance, current age calculation is: subtract the birthday year from today’s year. Someone born in 1977 would be 97 minus 77 or 20 years old. This same calculation in 2000 is: 00 minus 77 or -77 years old. That is a simple date calculation that goes on every day in many systems; other date calculations are more complex and the ramifications far reaching. Because many of those systems, which were expected to be obsolete by this time, will still be in use for four or more years, the century must be calculated or stored with each year to ensure correct calculations.

At Arizona State University, the Student Information System (SIS) was created in the late 1970’s. The SIS resides on an IBM/MVS mainframe utilizing an IDMS database which stores data needed for Admission, Registration, Financial Aid, Residence Life, Alumni, etc. Virtually every facet of student life has information stored on the SIS. The information is retrieved using COBOL, ADSO, and EZ+.

When the century problem became evident, options were explored, such as reengineering the current system and putting it on a client-server environment. This new technology was the most coveted, but the SIS is very complicated. There was no guarantee the IDMS database would be gone, in fact, it was quite likely some of the information would always reside on the mainframe and possibly in IDMS. Another roadblock was timing. What would happen if the reengineering was not completed in time? Most applications would start to have problems by 1998 and by 2000 would be nearly useless. The risk was too great and the decision was made to make the current systems year 2000 compliant while the reengineering project progressed.

Research
The Information Technology management team backed the project early in the process. They were knowledgeable about the problem and did not need to be convinced that it was important; in fact they pushed to get the project going. So, in April of 1994, a team of programmer/analysts and DBAs was formed to study and define the problem, recommend approaches, and set a timeline.

The team found a number of data types containing dates. There were dates directly on the database that were stored as six digit year-month-day and there were many dates consisting of year and term. These dates not only had different formats, but were also stored in three different ways: numeric, display, or packed.

After determining the types of dates, the question became how many programs and IDMS records are affected. The “manual” method was used to inventory each IDMS record date and each program. The final tally was approximate: 200 IDMS records with dates, affecting at least 3000 IDMS/COBOL programs, 600 EZ+ programs, and 2000 ADSO modules. The next question was: how soon would an application fail.
As the following table shows, the anticipated graduation date started failing as early as 1994 when many part-time students filed an anticipated graduation date of 2000 or beyond. In 1998, Admissions would begin processing students for spring semester of 2000. In early 1999 Financial Aid would have problems, and by 2000 every system would have major problems.

As ASU Computer Systems Close in on 2000

- Student transcripts cannot reflect information
- Students cannot graduate
- Grades cannot be posted
- Students cannot register
- Students cannot pre-register
- Parking cannot be assigned
- Financial aid cannot be awarded
- Students cannot apply for dorm assignments
- Students cannot be admitted
- Anticipated graduation date beyond Fall 99 cannot be posted

Estimating the length of time to complete the project was no easy task. Using the previously completed inventory, a general rating (easy, moderate, difficult, impossible) was given each program or module. This rating was based on program type, (IDMS-DC, batch with flat files, ADSO, etc.) and also general knowledge such as the fact that the on-line registration program is complicated and very difficult. Then each degree of difficulty was given an average completion time per program: 4 hours to do an easy program, 40 hours to do a difficult one, etc. After multiplying and totaling the hours, our manual estimate was staggering. We estimated the project would take 58,000 hours (9,000 days) at a cost of $5.6 million. It was even more frightening when a programmer/analyst tried converting an easy program and found that it actually took four times longer than the first estimate.

Committee Recommendations and Actions
Approaches to solving the problem were researched and the following recommendations were made:

- Make minimum changes. In other words, do not put the century on screens or reports. It will become commonplace to see 00 and although it looks strange now, it will not for long. Also, if a screen has to display the century, somewhere that century would have to be input. It would be far easier to compute the century behind the scenes.

- Have minimum down time. When the cutover is made, the system must be down. In general, at ASU, a cutover is done in one weekend. It is impossible to change all records or all programs in one weekend. So a phased approach must be used.
• Be consistent with coding across the system; have standard century calculations.

• Make the least impact possible and the least number of changes. When a date is compared to
another date for equality, the century is not needed. It is only necessary to include the century for
unequal comparisons (greater than or less than).

• Add the century to all date and semester fields residing on the database. Analysis of the 200 date
elements would take more time than making the changes. This would also ensure that, if needed in
the future, the century would be available.

• Group phases by related database records. Since the systems were so integrated, changing one data
element would necessitate changing many programs across many systems. Unfortunately, this means
the programs need to be changed multiple times.

• Do not change field names but add century fields and change level numbers. The additional fields
would be the element name with the suffix ‘-cc’. The level numbers would change, but not the
current field name, causing fewer program changes. For instance:

Example:

BEFORE
01 EXAMPLE-SEM-FIELD
  05 EX-SEM-YY pic xx.
  05 EX-SEM-TERM pic x.

AFTER
01 EXAMPLE-SEM-FIELD-WITH-CC.
  05 EXAMPLE-SEM-FIELD-CC pic xx.
  05 EXAMPLE-SEM-FIELD
    10 EX-SEM-YY pic xx.
    10 EX-SEM-TERM pic x.

• Expand packed fields to include the century. As long as the name was not changed, and the century
was populated, there would be few, if any, logic changes needed for these fields. See the following
example.
Other more general recommendations included:

- Complete a pilot project.

- Retrieve the system date with the century. The COBOL version at ASU does not return the century with the system date.

- Inform the ASU community of the severity of the problem.

- Purchase analysis tools.

- Use in-house staffing. Since the database is so integrated and the educational system so specialized, it was felt that in-house staffing would complete the project more quickly and efficiently than using consultants. After the project is done, the new staff will have plenty to do with the reengineering project.

- Set up a separate area for testing the year 2000 changes. This will alleviate problems between regular maintenance and the year 2000 project.

- Declare a moratorium on changes to the current system.

- Files from outside agencies must include the century.

- Any newly purchased software products must be century compliant.

Team actions:
• An assembler program was written to return the system date with the century. A call to this program is used in place of the ‘accept’ verb within all programs. This program not only returns the century, it also delivers the data in various formats: ccymmddd, mmdccyy, yymmdd, etc.

• Posters were exhibited to bring attention to the project.

• A separate test area was set up.

Management actions:
• Committee recommendations and research statistics were distributed to various areas of the university.

• Supported the recommended pilot study with the following objectives:
  1. Test the committee’s recommended approaches for viability.
  2. Test an analysis tool for use functionality for the entire project.
  3. Verify the committee’s findings/estimates using the analysis tool’s estimating capabilities.
  4. Determine cost/benefit of using consultants versus in-house staff.
  5. Complete the project in 90 days or less.

Pilot 1
The first pilot, begun in February of 1995, consisted of changing four dates on two key IDMS records. This affected 200 COBOL, EZ+ and ADSO programs. The team to complete the pilot consisted of three ASU analysts and two consultants from the analysis tool company.

The pilot did not go as smoothly as hoped. Three months were spent trying to use the analysis tool which was really not ready for IDMS or for ASU in general. Much time was spent documenting the project and defining procedures for the consultants. The consultants made suggestions; one based on saving space. They suggested using a code of ‘0’ for 19 and ‘1’ for 20 in the packed fields. Numerically, any computation would still work, and no additional space would be used.

Because the consultants did not understand how integrated the IDMS database was, they insisted that bridges be built and entire programs be made century ready so a program would not be entered multiple times. The consultants insisted the system could be molded into their methodology, but finding a way to accomplish this was impossible.

Some documentation completed during this time has been very beneficial. One helpful set is called simply ‘Programmer Documentation’. This is in a manual format and gives a description of the current phase, a listing of the records and fields that are changing, a description of the field function, and a Bachman diagram of the area. The documentation also lists the ways to make the changes, describes how to use the change management procedures, and lists the libraries the programmer/analyst will use.

The stumbling analysis tool reported the actual lines of code that would possibly need changing, estimates on the time and cost to make the changes, and various other reports. The output was used to determine the program groupings, the degree of difficulty of changes, and as a starting point when making the actual changes.

After 90 days, the pilot was not even close to completion. In fact, no actual coding had been started, the set-up was not quite complete, and results from the tool were still incomplete and inaccurate.

The pilot experience was disenchanting. The consultants were not knowledgeable in our environment, they did not understand a network database, and they did not make a great effort to learn. They insisted our approach to changing records instead of systems would take too much time and their approach was
much more efficient. While in theory it would be more efficient, in actuality it was impossible to implement with our system structure.

Lessons from the Pilot
However, valuable information and experience was gained. To be most efficient, the consultants need to be somewhat familiar with the environment, familiar with year 2000 project concepts, and flexible enough to listen to the system experts. The analysis product should have a proven record in the environment type, and have the capability of being used by in-house programmer/analyst’s -- not only company consultants. The tool should be flexible to allow reports on chosen data and the reports should be easily understood.

Another lesson was using the codes for comp fields. In theory, this would save space, but in practice it actually cost time. Since the decision was made to add the century to all fields, all display fields had the real century, while the comp fields had the code. Compared fields were not necessarily in the same format, so now each field had to be examined to determine if it had the century or a code, and one or the other had to be converted if they were not the same. This was time consuming and error prone so the decision was made to include the century on ALL dates, whether stored as packed or display fields.

Pilot 2
Since the first analysis tool did not work as anticipated, the first pilot was put on hold while another tool was tested. The second pilot started in May 1995, and consisted of changing six dates on three IDMS records. This affected 200 COBOL, EZ+ and ADSO modules. This time, the team would consist of only the three ASU analysts. The tool tested was ADPAC’s SystemVision (now marketed by Platinum Technology). This tool proved to be user friendly, it was menu driven, and the process was easy to step through. After a two-day training session from ADPAC, the team ran the analysis. Analysis for the pilot was done in approximately one month.

The output from the software included the lines of code that may need to change, various reports and estimates on how long it would take to complete, and the cost, with cost factors input by ASU. This tool was subsequently purchased.

Staffing, Training, and Procedures
The next challenge was staffing the project. The project itself, though not glamorous, is very important. Since many of the current staff had started working on the reengineering project, personnel had to be added to do the year 2000 conversions. Two analysts from the initial pilots became coordinators/mentors for the project, two other in-house programmer/analysts were added to the team and six new programmer/analysts were hired.

Hiring six new people when the competition for COBOL programmers is so acute was not an easy task. The personnel needed to be knowledgeable and experienced COBOL programmers who were quick studies, who could pick up IDMS, EZ+ and ADSO in a short amount of time, and learn the SIS quickly. The new staff definitely needed to be team players. This project also allowed new analysts to learn the general business practices of ASU giving them a broad knowledge base for later work on the reengineering project.

In order to evenly distribute the workloads, packets were made of related programs with varying modification difficulty and a mixture of batch and online. Each programmer/analyst was given documentation for tools and procedures. This consisted of the programmer documentation previously mentioned. Formal and informal training was provided. A mentor was assigned to each person and group classes were scheduled whenever possible. Training was very important. If the training was
thorough, everyone would be able to contribute faster and efficiently. Regular team meetings focused on the project and team building. Difficulties and solutions to those difficulties were also discussed. Walk-thrus were completed on each program to ensure correctness, not only for the newer people, but for everyone on the project. Walk-thrus also served to introduce new ideas and improve procedures.

Copy modules were developed to aid in coding consistency. These modules consist of various ways to compute the century by passing and returning generic fields and standard documentation.

Tracking the Project
Microsoft Access is used to track the project progress and reports are easily generated from the tables. The tables and fields for each phase include: program names, program type, analyst assigned, move request number, status, etc. Each programmer/analyst may update the database with pertinent data as necessary keeping the database current. Some of the queries include which programs are complete, how many programs are still in progress, how many batch or online programs are there, etc. Current reports can be generated at any time.

Testing Importance
Thorough testing is the most time consuming activity of each phase. The initial testing is completed by the analyst who made the changes. Next the changes are moved to the quality assurance area and customers are asked to test. They are given one month to complete their testing. Often, the programmer sits with the customer who is testing, allowing the analyst to see how the application is used in the real business world while monitoring the customer’s testing. The programmer may make suggestions about what to test, but the customer tries to test as many functions as possible.

Initially, customers did not fully understand how important testing was. Unlike most projects, one cannot see the changes, no screens have changed, no glaring changes are on the reports. The only indication that something is wrong is when something is different, and the only way to find that is to test for everything from the mundane to the bizarre. Unfortunately, the importance of testing became apparent after one production move. After the phase was in production, the customer found many problems with a particular application which affected students who were asking for verification letters. It took more than a week to fix the problems. After the inconvenience of waiting for the production fixes, the value of customer testing increased proportionately.

Migration Processes
Moving from the test environment through Quality Assurance and then to production is a multi-step process. The programs are migrated manually from the separate test environment into the regular test environment where they reside for approximately one week. From there the migration is automated using PANAPT. Move requests are created for each group of programs, records and copy modules by the programmer/analyst who made the changes. Then using approvals, dates and move types, the programs are migrated to the quality assurance environment. If anything goes wrong in the migration process, this is where it should happen -- during the move to Quality Assurance, where it can be corrected before any harm is done. After one month of testing, the corrected automated migration process is used to move the programs and modules to production.

Visibility
Keeping the project visible to the ASU community has been important. One of the fun team activities was to create a measurement tool or storyboard which visually displays the project’s progress. The project motto is ‘Rising to the Year 2000’ and hot air balloons are used as symbols. The measurement tool is a large foam colorful hot air balloon. A graph is attached to the balloon showing the separate phases on the bottom and the percentage completed on the side. As each phase is started and progresses to completion, a miniature balloon rises up its column marking the percentage completed. There is also an
area which can pictorially show the phase we are working on -- such as Phase 4 -- which was “Previous Institution Information”. The picture shows a number of institutions with a wavy line going into ASU. This measurement graphic changes with each phase. It is placed in a prominent hallway to meeting rooms, which allows other analysts and customers to see it enroute to a meeting. See the graphic below.
A moratorium has been implemented on changes to the current system. This of course cannot be a complete moratorium since changes are mandated by various agencies. But, as much as possible, the ASU community is asked to wait for any enhancements until the century changes are complete. To heighten awareness, management has taken the opportunity to present the problems and solutions at meetings, etc. The team members also mention the project when talking to customers. Articles about the problem are posted outside of cubicles, and “shock treatments” are used at various opportunities. For instance, at one meeting with customers, a skit was presented which essentially stated that Admissions could no longer admit students. Meeting participants were asked how it would affect their area. Another action being used is to close enhancement estimates with a reminder such as: “When prioritizing this project, please remember that the year 2000 project must be completed for the university to continue business as usual.” These efforts have heightened awareness.

**Total Project Planning**

In order to break the project into phases, records were grouped into ‘processes’ such as records created by graduation, records created by admissions, etc. Then these processes were inventoried according to the number of programs and modules per process. Since a cutover must be completed in one weekend, the record types had to be considered when determining how long it could take to do the cutover and compile the programs. (It has been found that approximately 300 programs can be recompiled in the allotted time period.)

Each process was given a ‘fail’ date, i.e., when it would quit working correctly. (As already stated, the anticipated graduation date has been a problem for a number of years. Admissions will start admitting students for spring 2000 in 1998; but Parking does very little future dating, so it will not fail until the
1999-2000 school year.) Breaking each process down into timing fail areas allows prioritization of phase completion.

The ASU Year 2000 project was grouped into 18 phases. During the project, over 200 IDMS records will be changed, and 3000 programs and modules will be changed multiple times. In general, each phase takes 4-6 months to complete from start to production, with some overlap as the customer tests in quality assurance. As a program is changed, the objective is to get it as century-ready as possible. In other words, the first time into a program there will be many changes since all working storage will be made as century-ready as possible. But the next time into the program, only minor changes will be needed since working storage is already changed. So, as the project progresses, less time is needed to complete the changes. Also, as experience is gained, the changes are made in a more timely and efficient manner.

The project completion date is early 1998. The rest of the year will be used for ‘clean-up’. This does not allow for much slippage; it’s a very tight schedule, and January 1, 2000 will not wait for the project to be completed.

Summary
ASU’s project started in 1994, and coding started in 1995. Currently five phases have been completed and Phase 6 will be completed in July. There are two other phases running concurrently: Financial Aid and Residence Life. Residence Life will also be completed in July; Financial Aid by the end of the year. The turn of the century is two-and-one-half short years away. Many computer systems will not be ready in time. At ASU we are confident that business will carry on as usual in 1998, 1999, 2000, 2001 and beyond.