Understanding Y2K: A Programmer's Assessment

As a computer programmer, I'm frustrated by all the news stories going around about Y2K. I'll admit that it's a bit too late to contribute to technical solutions to the problem, but I can at least offer some more detailed background information to people wanting to understand the problem. I can also give you a basis for evaluating the various predictions about the consequences of the problem.

by Michael Goodfellow

I have been reading a lot of news stories about the Year 2000 computer problem (also known as Y2K). As a computer programmer, I'm frustrated by these stories since at best they lack detail, and at worst they are downright misleading. You may find this information useful for evaluating a Y2K project at your company, institution or organization. If you are a business owner or leader in your organization who has not started a Y2K project, this article may encourage you to start one as soon as possible.

The Y2K Problem

In the beginning computers were small. Instead of the megabytes that a modern desktop computer has, computers that took up a room had memories of a few kilobytes. So when financial programs were first written, programmers kept the data as small as possible. And if you could represent 1970, which takes 4 digits, as "70," which takes two, that was a major savings; especially when multiplied by thousands of records in a banking system.

A few years ago, when people first started talking seriously about this (I remember kidding with my father about it back in the 70's), most programmers thought the problem was a minor nuisance. From a technical point of view the problem was completely understood decades ago, but until recently, no one really appreciated the number of systems it would affect (even 10 years ago, there were a lot fewer computers in the world). In particular, the problem of embedded systems was not discussed.

Embedded systems include the chips buried inside your appliances as well as inside things like power plants, refineries, the phone system, etc. In the last couple of decades, embedded systems have become very cheap, and this has made them pervasive. Just looking around me now, I know there are computers in my phone, fax machine, VCR, TV, CD player, camcorder, microwave oven, thermostat, and car. It has been years since I shopped anywhere that did not use a computerized cash register or inventory system.

Since the Y2K problem was never really addressed by the software industry, it has spread widely, along with the use of computers generally. As companies have embarked on projects to fix Y2K, estimates of costs have gone up repeatedly. The consequences of failure to fix the bug (including massive legal bills) are now very apparent. So much so that serious people (including programmers) are worried about a complete collapse of the world economy. Some people are predicting mass starvation in the United States due to failures in the utilities. On the other hand, many experts appear on TV declaring that the problem is minor, that fixes are well under way, etc. The US government gives its major departments grades, ranging from A's to F's, but then reassures people that everything will be OK. Faced with such a wide range of predictions, from "you'll never notice it" to "head for the hills," what should you believe?

You Will Definitely Notice Y2K

Some people will tell you that Y2K is a silly little problem hyped to the max by consultants who stand to make money off of it, religious doomsayers looking for a secular reason to predict the end of the world in 2000, and sensationalistic TV reporting.

But as they say, "just because you're paranoid, doesn't mean someone isn't out to get you." People with no money to make and no axe to grind have agreed that this is a serious problem. Some reality checks for the optimists:

* Big companies, including big banks, are spending millions of dollars to fix this problem. The optimists must assume that all of these companies are run by idiots who can be talked into anything by their computer staffs. They must also assume

that the hundreds of programmers working on these projects are all either oblivious as well, or are in on the biggest practical joke in history (and can keep a secret). This is not reasonable.

* Some will say that nothing will happen because everything will be fixed. However, even if big companies are solving a real problem, where does that leave smaller companies that don't have the budget to solve it? Despite the attention given to the Fortune 500, the economy is mostly made up of small and medium sized companies. Not only do they employ most of the workforce, but they are also critical suppliers to the big guys.

* Year 2000 problems have already been detected in many businesses that deal with long-range dates (life insurance, mortgages, etc.) as well as shorter-range items like credit cards. Until just recently, most credit card companies were not issuing cards that expired in 2000, because point-of-sale machines would not take them. One retail store sued the manufacturer of its cash registers over this problem. As we approach the end of 1999, more and more companies will be dealing with things like delivery dates that extend into 2000, and will have problems.

* The web sites of many software companies (including Microsoft) explicitly list versions of their software that will not handle dates in 2000. Users of this software must at least upgrade to newer versions. Since most people simply haven't done this, businesses everywhere are going to be affected. In many cases, new versions aren't available yet. Non-US companies must wait for foreign-language versions of US software.

* Many embedded systems simply cannot be fixed. This may sound incredible, but consider your VCR. The 7-day programming feature relies on the date (and day of the week) being correct. On an older VCR, it will not be right on January 1, 2000. For your VCR, this is a minor problem.

But say you wanted to fix it. You would have to: (1) take apart the VCR to find the chip that controls the 7-day programming; (2) use a soldering iron to remove the chip from the circuit board (some are in sockets), (3) find a replacement chip. If the technology is too old, they aren't making chips in that format anymore, so a new one would have to be designed, (4) solder the new chip back onto the board, making sure you are neat enough not to break anything around it and (5) put the VCR back together and test it. This is the kind of thing that would have to be done to fix each of the millions of embedded systems estimated to have some kind of problem. Even if chips were readily available, this could cost more in labor than the device is worth. If the chips were not available, it would take months at minimum (more like a year) to get new ones.

But there's no time to do that now even for vitally important machinery. If there's no replacement for the unit (because the manufacturer is out of business, or the device is some quirky military hardware that no one else uses), you are stuck with a bad unit. As we'll see below, that's not the end of the story, but you certainly have a problem. If this causes power or phones to go out, it's everyone's problem.

* Globalization of trade and conversion to just-in-time manufacturing have made big companies more vulnerable to disruptions. A multinational like GM has literally thousands of suppliers scattered all over the world. To build cars, they need their US and foreign operations working. Needless to say, they need power, phones, banks, and shipping in all these countries to work. They also need their customers employed and optimistic enough to buy cars. If GM can't make cars for awhile, it's not the end of the world for people generally (although GM employees are hard hit), but it will be noticed. Especially if many major companies all over the world are in the same situation, all at the same time.

* Panic before 2000, and legal action afterwards, can turn even a minor problem into a major one. Many people on the net believe this is why governments generally are making reassuring statements.

Not the End of the World

On the other hand, the doomsday predictions (planes falling out of the sky, reactors blowing up, permanent blackouts followed by starvation) are also not reasonable. Some points to consider:

* Power could very well go out, even all over the US. This does not mean it can't be restarted. Many plants have cold-start capability, and can bring the grid back up. If computer problems prevent load sharing on the grid, you might see plants come up and serve only their local customers. Restoring at least some power could take days, but not weeks. Some areas might be blacked out longer, but no power company is going to let a major city sit in the dark any longer than it has to. The people who work at the power plant are not going to run around and panic when the lights go off. They are going to do their best to fix it.

* Some have said that even if power is back up for awhile, the plants will rapidly run out of fuel, since railroads will be out due to lack of computerized controls. I would point out that a lot of power in the US is hydroelectric and nuclear. These don't need steady deliveries of fuel. In any case, if the power is up, I'm sure the railroads can get necessities deliv-

ered given a few weeks to work on the problem. Again, I expect the people involved to burn the midnight oil fixing things, including putting together whole new systems, if necessary.

* The same is true of phones. International service could be affected severely, but it is just not reasonable to assume that given power and some time to work, the phone companies cannot get some kind of local service running.

* Boeing has checked its planes and found only a couple of problems, on older aircraft. None of the problems would cause a plane to fall out of the sky. If things go dark on New Years, you may have problems finding an airport that isn't stacked up with planes waiting to land; or you might not be allowed to take off, but the plane will not fall out of the sky.

We're still talking about a major worldwide depression here, but that's not the same as starving to death in the dark.

What Will Happen?

Between these two extremes, there is obviously a lot of territory. But to evaluate the news stories and official pronouncements, you really need to know about the Y2K problem in more detail. It's just not enough to say things like, "the Y2K problem affects computers, my car has a computer chip, so then my car has a problem." So before you get all worked up over Y2K, read the rest of this article and try to learn some details. If you can master the material I'm presenting here, I guarantee you that you will be as well informed as some people you see posing as Y2K "experts."

One thing you may be looking for is a list of which industries, countries, etc., will be hardest hit. There's no way I, or anyone else, can give you that list. For one thing, companies are not releasing even the little information they have. They are afraid that if they admit a big problem, their stock will crash. If they say the problem is small, and later have a bigger problem than they said, they will be sued for misleading investors. The simplest solution is to put out a generally reassuring statement which has a paragraph near the end saying that there's no way of knowing what shape their suppliers are in, and so no way to really predict future company performance. Governments have some information on their own internal readiness, but have no incentive to release it. The news is generally bad, and releasing it would just cause panic. Since they don't have to worry about shareholder lawsuits, they can just say nothing.

If you can master the material I'm presenting here, I guarantee you that you will be as well informed as some people you see posing as Y2K "experts."

In any case, if I did put in a list of which industries were in trouble, it would be out of date by the time you read this. Instead, I'll describe the kind of factors you should take into account when making an evaluation on your own. You can read news reports, and look around at your own company, community, etc., and make up your own mind.

To begin with, I'll write a short section on "What will fail?" This is meant to be reassuring, but is necessarily vague. I can tell you what kinds of programs are not affected by Y2K, but other than some obvious things, there's no way for you to tell if a program has a problem or not. Still, you should be aware that the problem is not universal.

Then, I'll write about the systems that will fail. There are lots of ways to categorize the world of computers, based on what your role is in all this, and what you want to know. I'll break things down three different ways, based on attempts to answer the question, "Who is going to fix this?"; "What can be fixed?" and "How will this affect me when it fails?" Finally, I'll give you my best case and worst case scenarios and talk about events that would determine which it will be.

Before I get started on details, one general reminder: Although lots of systems will fail at midnight on 12/31/99, many other computer systems use Universal Time (Greenwich Mean Time) internally. This means they may fail at 7:00 PM Eastern Time, not midnight.

What Will Fail?

It's not like every computer in the world is broken. Lots of computers don't use dates at all. They are not affected. This includes lots of embedded systems. For example, take a CD player. There's no way to set the date on one, and it will be unaffected by Y2K.

This is a rule you can use. Especially with an appliance, if you can't set the date, then it's not going to crash. If you've never set the date, and it works fine anyway (like a VCR that you don't use the timer feature on), then you also have no problem. It's estimated that over 90% of embedded systems are the same way. That still leaves a lot of things to go wrong, but it's also good news. You can stop worrying about the microwave oven, the car, the phone, etc.

Even when you can set the date on a device, it doesn't mean it is used

for anything important. It might just be for writing the date on a log (think of a FAX machine that stamps each page with the date.) In one audit of power plant equipment, many devices were found with random dates. The date had never been set when they were first installed, and they had worked normally without anyone noticing.

Other devices will fail, but in fairly trivial ways. An old VCR will get the day of the week wrong. Even fancier systems will make the same kind of mistake. A building heating system might run when it's not supposed to, because the computer is indicating a weekday when it's actually a weekend. There was a news report about a city that found that its traffic lights would use the weekday pattern instead of the weekend pattern. All of these systems will be described as not ready for 2000, but it's hardly a major problem.

If you really want your VCR or other device to get the day of the week right, there's a simple fix. Just set the year to 1972. That year started on the same day of the week as 2000, and is also a leap year. So the year will display incorrectly, but everything else will work fine. The city with the stoplight problem is doing exactly this. Some systems will not handle 1972 though, because it is too early (more discussion later.) For these, you might have to set the date to 1994, which at least starts on the right day, and then change it on March 1, to 1995 after the leap year.

Some programs use dates extensively, like financial packages, and are almost certain to have problems unless they have been fixed. If nothing else, I expect 2000 to be the year of the bad bill. People will get billed ridiculous amounts, not billed at all, refunded ridiculous amounts. Payments will not be credited to accounts, etc. There are just too many small businesses that have not even started to upgrade their systems.

Who is Going to Fix the Problem?

To answer this question, consider who owns the computer and what amount of support it has. That produces the following categories:

1. *Systems with full-time support people.* This includes the big mainframe computers, which live in their own air-conditioned rooms and have staffs of programmers and operations people in constant attendance. Some of the software on that computer was written by the company that owns it. That support staff will have to fix the computer, either by rewriting parts of the software, or installing new versions. This category also includes networks of PC's, as long as the company has a support staff. They will be responsible for upgrading all the PC's owned by the company.

Other than these obvious categories (things without dates, things with lots of dates), there's no way to tell what will fail. That's the most frustrating aspect of this whole mess.

2. *Desktop systems, at work or at home.* If you bought it, or installed software on it, it's going to be your job to fix it. This is true even if the computer is owned by your employer. Look around—is there a support staff? Is there money to hire consultants? Are there any consultants worth hiring left by the end of the year? Probably not.

3. *Embedded systems from major manufacturers.* These might be upgraded by replacing a whole circuit board. There might be a new version from the manufacturer. It might even be in stock. It's probably not free. The company that owns the system still has to find it, order the replacement, test the new one when it arrives and of course pay for it. This all sounds simple, but it is amazing what a company can get away with. Years ago, an IBM Customer Engineer (repairman), told me he went to a refinery to do a scheduled upgrade of an embedded system. The company insisted they had no such system. From the paperwork, they found a system number, and from plans, a column number in the plant where the machine was supposedly installed. They went there, and sure enough, a computer was sitting there—covered completely with soot and tar, invisible, and clicking away. The first problem most companies have had with Y2K fixes is figuring out what computers and software they own in the first place, let alone what has to be fixed.

4. *Embedded systems without support.* If the manufacturer is overwhelmed with orders, or never made a replacement model for the unit, or has gone out of business, the system is unfixable. Start looking at workarounds or contingency plans.

People say Y2K is a "computer" problem, but let's be more specific—it's really a *software* problem. Most people (other than TV reporters) know the difference now. Software is the game you install on the computer (hardware) that you bought. If you've never touched a computer, consider the software the CD (or record) and the hardware the record player. With embedded systems, this distinction is a bit thin, since they have all their

software burned onto a special kind of chip, called a ROM (read-only memory), and mounted onto the circuit board along with the computer. Sometimes, the ROM is inside the computer chip, to make things even cheaper. But it's still the software that has to be fixed.

A Software Problem

So when we ask what can be done, we categorize computers by the software they run.

1. Big companies typically run a mix of software written by the company and software that they've purchased. The company can fix it's own software, although sometimes at huge expense. More on this later. As a user of desktop PC's, you may also have written software, in the form of a spreadsheet or word processor macro. If so, you can fix it.

2. Most software is purchased off the shelf. This software may not work, but the owner cannot fix it. For one, under many customer agreements, it's not legal for you to modify it. For another, to change a big piece of software, you need what's called the *source code*. This is the human-readable version of the software. A special computer program called a *compiler* converts this into the *object code* or *executable* version, which is what is sold. Without the source code, you can't change (or fix) the program. There are *decompilers*, (which your customer agreement specifically prohibits) which can convert object code back into source code, but they are nearly worthless for big projects.

The problem is that source code contains (semi-)readable names, and comments on what the program is supposed to do. For example, "balance = balance + deposit" might be part of a program to update your checking account. When this is compiled, the names are thrown away—the computer cannot read and understand them, after all. So when the decompiler converts the object code back to the source, it has to make up names. Being just a program, it has no idea that the code is working with checking accounts. So it just writes something like "a = a + b". All the comments are gone, since they never made it to the object code. The output of a decompiler is many times harder to work with than the original source. The bottom line is that for packaged software, you have to contact the manufacturer for a new version.

3. Finally, there is software in ROM (read-only memory). Think of this as a floppy disk that has been turned into a chip and placed permanently inside the machine. To fix this, you have the same problem as with purchased software—you don't have the rights, and don't have the source code. You also have a new problem, which is that you can't update the code even after you fix it without replacing the ROM chip. If you are lucky, the ROM has a socket, so it's easily removable, and you can make a new one. If you are unlucky, the chip is soldered into the board, and can't be replaced. This is the worst case, but there's a lot of it around. Of course, when I say "you" here, I mean the manufacturer of the system, but "you" could also be a customer with one of these things, after the manufacturer has told you it's too late, or has gone out of business.

The pessimists are perfectly correct to say that two months of complete blackout in the US would cause starvation. I just don't buy the assumption that power will be out that long.

How Will This Affect Us?

There's a lot of software to fix, and it will certainly not all be fixed in time. So we have to do triage—fix the important systems first, and let the other ones go. Which are the important computers?

1. Utilities, Phones and Transportation. The most important utility of all is electrical power. The pessimists are perfectly correct to say that two months of complete blackout in the US would cause starvation—I just don't buy the assumption that power will be out that long. Not far behind is the phone system. All of modern commerce, as well as things like the supply chain for your local supermarket, depend on communications. Banks cannot function without the phone system. Neither can the military. After that, some people place water and sewerage systems—glitches in computers have already caused release of sewage into waterways. On the other hand, some people put transportation next. The railroads are particularly important. Things like coal for power plants, shipments of grain and other produce, etc., are moved by train. I think we could do without passenger planes for a while (I hate to fly), but lots of goods (and mail) are moved around by air. Internationally, the movement of aircraft and containerized cargo ships is vital. There are probably places in the world that will starve without grain imports from the US and elsewhere.

2. The Military. Although I think the chances of the US being attacked while vulnerable from Y2K shutdowns are minimal, the military probably considers itself very impor-

tant. If we get to the situation where troops are needed to police cities, they certainly would be nice to have. Coordinating them and getting their supplies to the right place at the right time is certainly easier with software. For example, if a big warehouse were to lose track of its inventory due to computer failures, it might as well be empty. An item would be as hard to find as a single book in a pile of thousands.

3. Banks depend on so many other systems working that I have almost no hope that they will all come through this OK. Banks are highly leveraged now, keeping only tiny reserves of cash. A bank needs to fix its own systems, but also depends on the utilities here and internationally, other banks, domestic and international, the industries who are its borrowers, the individuals who are its creditors, and the stock market where it may have placed a lot of its money. Banks are at the intersections of the financial network, and I can't see any scenario that does not hit them hard.

4. Government Help. Almost half the population of the US receives a government cheque each month, either from Social Security, welfare programs, farm income supports, etc., or as an employee of some level of government. Governments are big organizations and tend to have just the sort of older mainframe-based systems that are most at risk. Failure to write checks would severely affect ordinary citizens, hospitals and government contractors. By its own rating, the US federal government will not finish Y2K conversion in time. Some states are farther behind. Internationally, Europe is running a little behind the US, and Asia far behind.

5. Tax Collection. You may consider inability to collect taxes a blessing, but you'll miss the government services. By most accounts, the IRS computer systems are a mess, and Y2K work is not helping. The 2000 tax season will be bumpy. There's some worry that cheating the system will run wild. I surely hope not.

Get all of the above running, or at least limping along, and you are down to the level of individual companies/organizations. Some are obviously more important than others, but all are at some amount of risk due to their dependency on suppliers and customers, both domestic and international.

Foreign Countries

Countries that are more computerized are more at risk. The US is probably worse off, but Europe and Asia are almost as computerized. Some poorer countries will probably never know Y2K has happened.

Countries that have waited the longest to begin Y2K work are more at risk. Again, the ranking is US, Europe, Asia. It's hard to tell if Asia's late start will be balanced by its relatively less automated economy. (Japan though is both very automated and very late.)

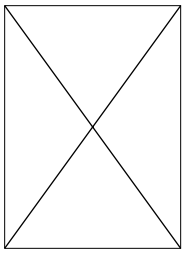
Countries that import basic commodities are at risk, especially if the international economy is severely disrupted, and currency values are changing from day to day. The US is very dependent on oil. Japan is dependent on oil and food imports. Russia is importing lots of food, even to the point of needing food aid during the latest financial crisis.

Countries that depend on export revenue will be hurt more by a disruption of trade. Asia has already been hurt badly by the economic turmoil. Russia was hurt by falling prices for oil. Trade disruptions will probably also hurt Latin America in 2000, if financial problems haven't already hit by then.

On the supply side, if the countries that export oil were to have major problems, the whole world would be affected. The US is both a major customer and a major food exporter to the rest of the world. Japan is a major exporter of electronics and car parts. Failures in these countries will affect industries that rely on their parts.

Within an economy, I've listed the kinds of things that everyone is dependent on—power, phones, water, bulk transport, government. We really don't know how far these networks reach. If a European government system is not running, will that mess up their exports, and then US imports, companies? Will the shock reverberate back and forth, or die down?

[Editor's note: Goodfellow in his original article explains in greater detail the Y2K problem to show readers various pieces of computer software, interdependencies and contingency factors. The author believes that if we are to get a feel for the Y2K problem we need to understand some of these details. For lack of space we were not able to print them here. However, for those who want to access and receive the details contact the IJFM editor, or Goodfellow's home page at http://www.best.com/~mgoodfel or at his email mgoodfel@best.com



Michael Goodfellow works as a computer programmer in Campbell, CA. He was educated in upstate New York and Houston, TX, Michael has been programming computers since age 13. His interests are computer graphics, user interfaces, programming languages and the Internet. Michael's work has included several years as a Research Staff Member at IBM Almaden Research Center as well as Senior Software Engineer at Oracle Systems, Momenta Computer, Strategic Mapping, and Symbol Technologies.