Y2K Defect Propagation Risk Assessment using Achilles

Donald J. Welch
Department of Electrical Engineering and Computer Science
U.S. Military Academy

Kevin Greaney
U.S. Army Information Systems Software Development Center – Washington

Solving date-related problems associated with the Year 2000 (Y2K) in a single computer system is not technically challenging, but simultaneously repairing a large number of cooperating computer systems is a daunting task. Assessing the risk of failure posed by the passing of defective data through system interfaces is a difficult task. We developed a process and supporting tools to assess the defect propagation risk in interconnected systems. We use a standard methodology to analyze the systems and interfaces individually and a discrete event simulator to recompose the model.

In all likelihood the Army and the rest of the Department of Defense (DoD) will not be totally ready for the date-related problems associated with Y2K. Senior decision makers are using software triage to determine where to apply scarce resources. In addition, contingency and recovery planning are also under way. An understanding of the risks posed by the interconnected nature of these computer systems is critical to successfully undertake these tasks.

Part of this process is the identification of *Mission Critical* computer systems. These are computer systems that the Army must have to perform its missions. Mission-critical systems make up around 10 percent of DoD systems (2800 of 25,000). DoD's leaders predict that all mission-critical systems will be prepared for the millennium change. However, Y2K vulnerability cannot be determined by examining just the computer system in question. Each mission-critical computer system relies on some number of computer systems to supply it with data that it needs to operate. In turn, each of those

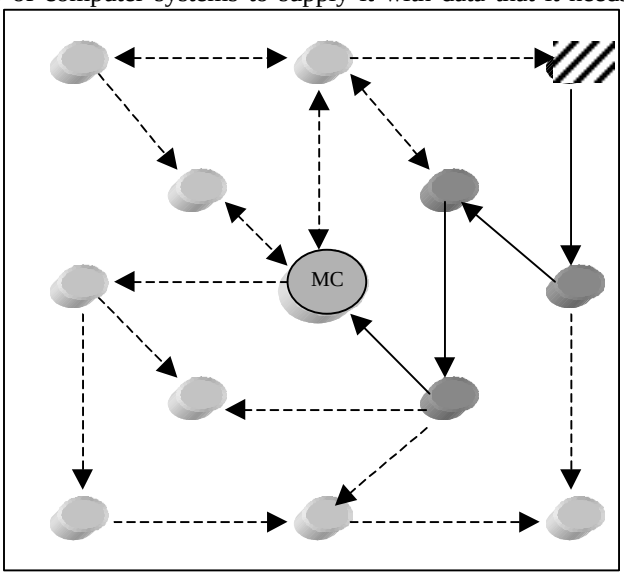

Figure 1:Defect Propagation Example in an Interconnected system of Computer Systems

computer systems has its own set of computers that provide it critical data.

The faults that propagate may not be easy to identify or overcome. Should a program fail by crashing or producing obviously incorrect data, most computer systems will survive — albeit in a degraded state. With this type of failure it will be obvious that there is a problem and contingency plans can be executed. The program may eventually become unusable because its data has aged, but this can be overcome during recovery operations. However, should the failure be subtler, recovery may be impossible. Inaccurate data can get into the database of an information system and spread to other computer systems. Recovery by backing out the incorrect information will not be possible after a relatively short time.

Figure 1 illustrates a simple example by representing a system of computer systems as a graph. A mission-critical computer system (MC) is in the center of the diagram. The interfaces are indicated by the edges. The original fault occurs in the node with the diagonal lines. The defect propagates through the interfaces represented by solid lines until it reaches the mission-critical computer system causing a failure. The original failure occurs so far away from the mission critical system that it is not easy to identify the risk this posed to the mission-critical system.

The U.S. Army Software Development Center-Washington (SDC-W) is responsible for 29 Army computer systems. We have developed a two-step process to help assess the risk of propagating Y2K defects to support SDC-W's Y2K. The first step is to model the system by following a definable, repeatable process. The second step is to feed the model into a discrete event simulator. This results in a better understanding of the risk posed by the interconnected nature of computer systems.

The process is iterative. As we analyze the results, we identify computer systems and interfaces that present the most risk. This risk is usually associated with a lack of knowledge. By putting more effort into understanding the Y2K behavior of those systems we can sometimes reduce the risk associated with those computer systems. In other cases we can increase the repair effort, or as a last resort we can begin contingency planning. We continually update our model as we learn more about the computer systems involved.

MODELING THE SYSTEM

The initial step of the risk assessment methodology is to model the system. The main concept is to break down the system into smaller parts. We use a definable, repeatable process to assess the relative risk associated with those parts. Because we have some consistency in the risk assessment of the parts, we can recompose the risk and have a meaningful understanding of the relative defect propagation risk.

Each computer system is considered a *black box*. We are interested in whether or not it will fail.We are not concerned with failure modes, or failure times. In our abstraction, a computer system can fail in two ways: on its own, or due to a defect passed to it through an interface.

A program is modeled by three primary factors: an overall probability of failure, the impact of a failure, and the criticality of the system to the Army. The probability of failure is not an absolute assessment, but an assessment relative to the other computer systems. An initial assessment of the probability for the computer system is adjusted by factors derived from an assessment of risk factors common to other systems. The methodology explains the process through which we derive these numbers. The impact concerns the impact on that specific system. Given that there is a failure, this indicates the expected severity of the fault. Conversely, the criticality factor concerns the impact on the system. This factor models the importance of this program to the correct functioning of the system of computer systems being assessed. We use the methodology described in Risk Assessment Methodology described below to determine the probability of failure, impact of failure on the computer system, and impact on the Army's mission.

Each individual computer system has interfaces with other computer systems. We model these interfaces simplistically. Each interface is unidirectional and includes all information and defects passed from the source to the sink computer system. Again we follow the methodology, described below, to determine the probability that given the source computer system fails, it also will cause the sink to fail. This probability is relative as it is for the computer system assessment.

The output of this activity is a directed graph. The nodes of the graph represent computer systems while the edges represent interfaces. For simplicity edges are unidirectional. Two edges are used to model the interface for cases where two computer systems pass information to each other.

Risk Assessment Methodology

Our risk assessment methodology consists of three steps which are performed concurrently. While working on one computer system we discover information about another and include it in all appropriate assessments.

1.1 Identifying Computer Systems

The first step is to identify the computer systems to model. As a minimum, all the computer systems that share direct interfaces with mission-critical systems must be modeled. Determining how many links away from the mission critical system is a trade-off between cost and knowledge. It can take days to collect and assess all the information concerning a computer system that is outside the control of the SDC-W, so weighing the effort against the benefit is necessary.

A necessary part of identifying the computer systems is identifying the interfaces to model for each computer system. Some computer systems have hundreds of interfaces and once again decisions have to be made concerning cost vs. the benefit of analyzing an interface.

1.2 Analyzing Computer Systems

The risk assessment of computer systems includes a base determination of the relative failure probability and four adjustment factors. After the base is adjusted, the result is a final relative probability of failure. The amount of information that we have on the computer systems we model varies from very detailed (for the computer systems the SDC-W manages) to very general. Our use of a definable and repeatable process gives some consistency to the results. Our model allows us to incorporate this wide variance in detail into single abstraction.

1.2.1 Base Probability of Failure

The analyst looks at many factors to generate an assessment of the system's base probability of failure. These factors include organizational, process and product factors.

The support of the functional proponent for Y2K work is critical. Functional proponents who prioritize new work over Y2K repairs add risk to a project. The reputation of the developer also plays a factor. An organization that has trouble producing quality code will also probably have trouble making Y2K repairs.

A key process factor is the Y2K project progress, with respect to where it should be at any given time. In addition, the types of tests run give a very good indication of the risk. Standalone tests with synthetic data are useful, but live end-to-end tests provide a high degree of assurance that computer systems are Y2K compliant. Since there is a great deal of product uncertainty until January 1, 2000, process plays a major part of the risk assessment.

Looking at the product itself is difficult. In most cases we do not own the product or have access to it. One indirect measure that we use is whether or not the developer has access to the complete source code and documentation. Proper assessment, testing, and repair requires an understanding of the software that is difficult without source code and documentation. Engineers must work much less precisely in a black-box environment. We also look at the repair, both the technique used and the extent of the repairs. A system written from the start to be Y2K compliant carries far less risk than one which undergoes extensive repairs. We have also determined that there is a different risk associated with computer systems that are repaired by altering their internal logic when compared with computer systems that employ wrappers or similar techniques.

1.2.2 Adjustment Factors

These factors are used to adjust the base probability of failure if there is a risk to the system identified in this category. Items that fall into these categories play a part in the assessment of numerous systems and are kept in a central database.

Commercial-Off-the-Shelf (COTS) Software

This includes all the software components that the computer system relies on in its run-time environment. The most obvious and universal is the operating system. Different versions and patches are tracked in terms of their relative Y2K risk. Database management systems and other system software that comprise the computer system are included in this assessment.

COTS Hardware

This category is similar to the COTS Software category, except it includes the hardware components of a computer system. The actual box or boxes on which the software executes is the primary example. However, we also include the peripherals that the computer system requires to perform its function. Many computer systems rely on a large set of devices, such as uninterruptable power supplies (UPS), and automated tape back-up systems. The risk associated with these devices is

incorporated here.

Infrastructure

This includes the critical parts of the infrastructure that must be available for the computer system to function. The most obvious examples are the networks the systems rely on to transfer data. This category can also be used to delve into great detail about the infrastructure. Items such as the power grid and building security access systems also pose a risk to a program's function. We have considered this level of detail, but did not use it in our assessments because the increased fidelity does not increase the accuracy.

Application Portfolio

This category includes components of the application software that are common to more than one computer system. Programming languages and libraries are part of this category.

1.2.3 Impact

To determine impact we look at the importance of date-related calculations to the functions of the computer system. Our definition of impact is confined to the computer system we are analyzing, and not its interfaces to other systems. We take a v simple view of the impact of a failure on the system. We describe the impact as low, medium, or high. We arrived at this abstraction after many experiments with more detailed descriptions of the impact. We found that higher fidelity descriptions of impact did not provide us better quality risk assessments.

1.2.4 Criticality Factor

In many risk assessment models the criticality factor would be rolled into the impact. However, since we are specifically looking at the risk associated with the interconnected nature of the computer systems we broke the category out of impact. We determine criticality factor by comparing the Army's functional area core business processes with the business processes performed by the computer system. The criticality factor considers the importance of the computer system to the Army's mission.

1.3 Assessing Interfaces

The third part of our assessment is examining the interfaces between computer systems. After experimenting with much more detailed models we found that we could maintain an acceptable level of accuracy by describing an interface with one number. That number is a relative probability that given a failure of the source computer system the sink computer system will also fail. We base this assessment on the existence and quality of the System Interface Agreements(SIA)/Memorandums of Agreement(MOA) between the two responsible parties. The SIA/MOAs also tell us the type of information passed across the interface. We use this to assess the sensitivity of the interface to date-related errors. We also include the level of testing of the interface. A tested interface is better than one that has not been exercised. Actual data is better than synthetic data, but nothing tells us more than a live test.

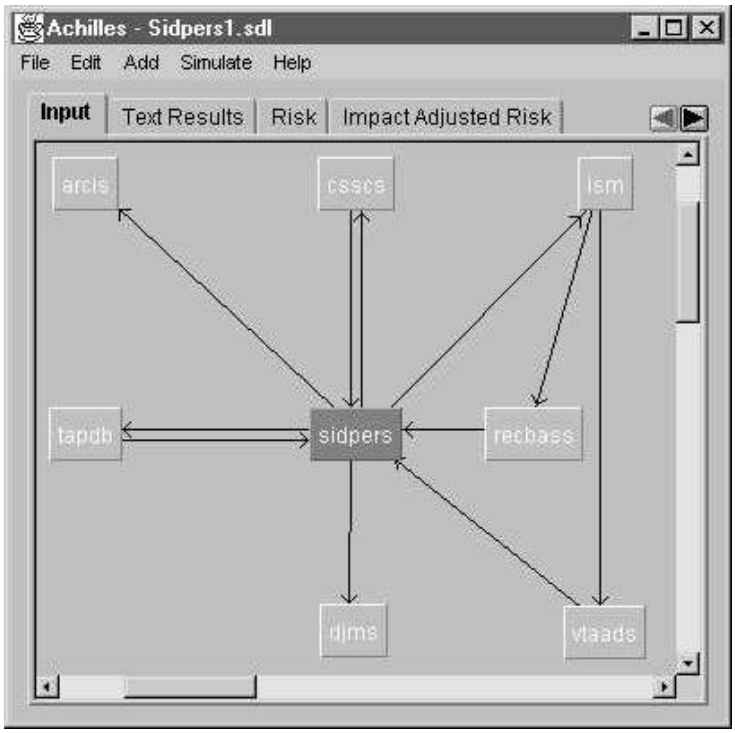

Figure 2: Achilles Input Screen for a simple example.

ACHILLES

Achilles is the discrete event simulator used to combine the behaviors of the system components. It uses a Graphical User Interface (GUI) to collect the assessment results and create a model. We then use Achilles to simulate the system interactions and analyze the results. We can look at the results both graphically and textually.

2.1 Creating the Model

Computer systems are added to the model by "pointing and clicking." The user chooses the location to minimize crossing interfaces and maximize clarity in the final model. Figure 2 shows the input GUI with a simple notional system model.

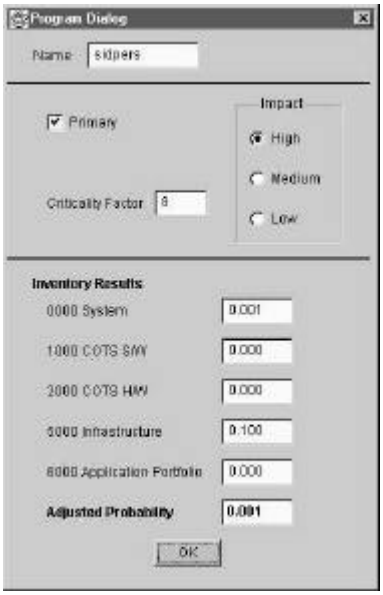

Figure 3: Computer System Model Input Dialog Notional Example

When a user identifies the computer system location, he enters or modifies the modeling data through a dialog box (Figure 3). The dialog box allows the designation of the computer system name. This is where the user designates the system as mission critical (primary), enters the criticality factor and the impact. The base failure probability and adjustment factors also go here.

Adding interfaces to analyze is done in a similar way. The user indicates the source and sink computer systems using the mouse. A dialog box for entry of failure probability is presented in Figure 4. Once the model is complete the simulation is executed and the results can be analyzed.

2.2 Results

Achilles provides both textual and graphical output. The textual output offers detailed insight into the behavior of the system, while the graphical output gives a high-level portrait of the defect propagation risk.

2.2.1 Graphical Results

The graphical results take the same form as the input, the difference being that they are color coded to better illustrate the risk (Figure 4). The SDC-Washington codes Y2K computer system risk using a simple "red," "yellow," and "green" status which we use for Achilles output. Achilles displays the computer systems and interfaces using this same scheme.

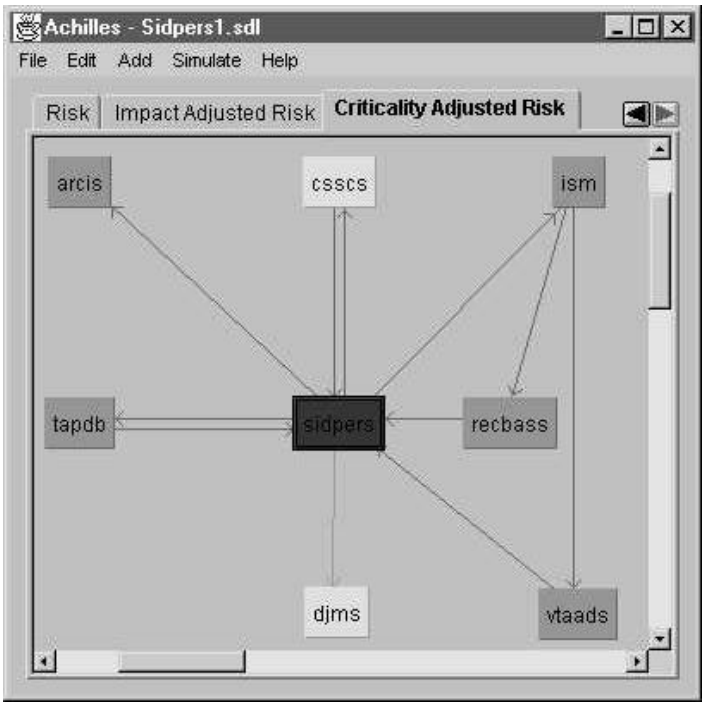

Figure 4: Example Notional Graphical Risk Results

2.2.2 Text Results

Figure 5 shows an example of the output provided for each computer system in the model. After the computer system identification the relative, impact-adjusted and criticality adjusted risk of a Y2K failure is shown. The rest of the output shows a breakdown of the relative risk from each adjustment factor as well as the probability that a failure will be caused by a defect passed through an interface.

```
Program ARCIS
         0.691 risk
         0.691 impact adjusted risk
         0.469 critical adjusted risk
   5000 Interfaced System failure rate:         0.423
   1000 COTS H/W failure rate:                  0.000
   2000 COTS S/W failure rate:                  0.000
   5000 Infrastructure failure rate:            0.041
   6000 Application Portfolio failure rate:     0.000
```

```
   Propagated Defect failure rate:              0.227

Interface: SIDPERS->ARCIS 0.227 risk
Interface: SIDPERS ->CSSCS 0.275 risk
```

Figure 5: Example Notional Detailed Risk Results

For each interface in the model, output similar to that shown in Figure 6 is presented. This indicates which interfaces have the most potential to pass defects that cause failures.

CONCLUSIONS

The Y2K problem for an individual computer system is not a technically difficult problem. The tough issues in implementing single system solutions are mostly project management dilemmas. As difficult as those problems may be for computer systems in isolation, the difficulty increases dramatically as the number of interfaces increases. DoD systems typically have interfaces that number in the hundreds and estimating the risk posed by those interfaces is a daunting task.

Knowing the risk to mission-critical computer systems is important to three critical areas of work now on-going: assessment, contingency planning and recovery. DoD computer systems cannot be assessed in isolation. Due to the uncertainty involving other systems, risk management techniques that take into account the interconnected nature of the computer systems must be used. Simulation is a powerful tool in this respect.

To deal with the Y2K problem we needed a definable, repeatable process to assess Y2K risk and use risk in Y2K project planning. The driving consideration to using simulation as opposed to numerical analysis was the effort required to assess the risk. Because of the dynamic nature of theY2K risk understanding, we also needed a model that was adaptable. We found our first attempts to create a model too detailed. Our higher fidelity model was not showing a corresponding higher accuracy result and the cost increase was large. We found that the simplified model presented in this paper to be much more practical.

ACKNOWLEDGEMENTS

We would like to thank the Y2K Project Team at USA CECOM Software Development Center-Washington for assistance and funding.

ACHILLES TOOL AVAILABILITY

The tool is available to anyone by contacting LTC Welch at dd2354@usma.edu

ABOUT THE AUTHORS

Lt. Col. Welch is an associate professor of computer science at the U.S. Military Academy. He teaches software engineering, and has been an Army software engineer. His military assignments include Infantry and Special Missions units. His current research interests include dynamic reconfiguration, software engineering, managing the risk from Y2K failures, and distributed simulation. He received his bachelor of science degre from USMA, a master's degree in computer science from California Polytechnic State University, and his doctorate from the University of Maryland.

Col. Kevin J. Greaney is the Commander of the U.S. Army Software Development Center - Washington. He has extensive experience in information technology and systems acquisition from the tactical to the strategic level. He has served in information technology positions with Special Operations Command, 1st Special Forces Operation Detachment - Delta, the U. S. Army War College and the 82nd Airborne Division. He is Acquisition Corps Level III certified in Program Management and Computers-Communications.

REFERENCES

Directorate of Information Systems Command Control Communications and Computing. "U.S. Army Year 2000 (Y2K) Action Plan, Revision II, Proactive Actions Supporting Force XXI and the Department of the Army C4I Technical Architecture." 1998.

Directorate of Information Systems Command Control Communications and Computing. "Army Year 2000 (Y2K) Action Compliance Certification Checklist, Revision II." January 1998.

General Accounting Office, "Year 2000 Computing Crisis: Business Continuity and Contingency Planning." (August). Available at http://www.gao.gov/y2kr/GAO/AIMD-10.1.19.htm 1998.

General Accounting Office," Year 2000 Computing Crisis: An Assessment Guide." (September). Available at http://www.gao.gov/y2kr/GAO/AIMD-10.1.14, 1998.

General Accounting Office, "Report to Congressional Requesters: Defense Computers Year 2000 Problems Threaten DoD Operations." (April). Available at http://www.gao.gov/y2kr/AIMD-98-72.htm, 1998.

Evan, James R. and David L. Olson. *Introduction to Simulation and Risk Analysis*. Prentice-Hall, New York. 1998.

Kappelman, Leon, Darlu Fent, Kellie Keeling and Victor Prybutok. "Calculating the Cost of Year 2000 – Compliance. " *Communications of the ACM*, Vol 41, No. 2, February 1998.

Khoshgoftaar, Taghi, Edward Allen, Robert Haltreed, Gary Trio, and Ronald Flass. "Using Project History to Predict Software Quality." *IEEE Computer*, Vol. 31. No. 4, April 1998.

Perry, William E. "Effective Methods for Testing Year 2000 Compliance." *Crosstalk*, Vol 12, No. 1 January 1999: 16-18.

Pressman, Roger, Software Engineering A Practitioner's Approach. McGraw-Hill. New York. 1995.

Rubin, Howard. "Bracing for the Millennium." *IEEE Computer*, Vol. 32, no. 1 January 1999:51-56.

About the Authors

Donald J. Welch
Department of Electrical Engineering and Computer Science
United States Military Academy
West Point, N.Y.10996
Voice: +1 914 938-2858
E-mail: Donald-Welch@usma.edu

Kevin Greaney
U.S. Army Information Systems Software Development Center – Washington
4035 Ridgetop Road
Fairfax, Va.
Voice: +1 703 275-9552
E-mail: greaneyk@fairfax-emh1.army.mil