CUTTER IT JOURNAL

Millennium Rules!

by Richard T. Dué

Most Year 2000 system failures will be hidden from view. No one will want to air their dirty laundry. Corrupt databases, failed production runs, interface and network problems, and embedded systems malfunctions will all be handled in the same way errors have been handled for the past 50 years — dig into the code, come up with some sort of solution that appears to work, and then go on to the next problem. However, because of the unprecedented scope and systemic nature of the Year 2000 problem, some failures will be very public. Failures in the so-called "iron triangle" infrastructure of telecommunications, utilities, and transportation will most likely occur in Europe, South America, and Asia. The cascading effects of these failures will spread throughout the entire world. Application system failures will most likely occur in the health care field and in smaller business and regional and local government organizations worldwide.

Some observers suggest that the social, legal, and economic effects of these serious, public Year 2000 failures will result in a reappraisal of current software development practices. The result of this reappraisal may be a new emphasis on the use of formal development methods and on the professional certification of software developers. Other observers suggest that Year 2000 will be no more than a "bump in the road" before an unprecedented explosion in the growth of new information technology investment, especially in e-commerce and other Internet-based systems development. In this scenario, there will be neither need nor time to reevaluate current practices.

It is my belief that it does not matter which Year 2000 scenario actually occurs. Regardless of whether there will be a complete collapse or merely a bump in the road, my experience as a consultant and educator indicates that information systems development, maintenance, and enhancement in the new millennium will have to recognize and follow certain principles. I call these principles the "Millennium Rules."

MILLENNIUM RULES

1. All information technology projects, including applications and hardware, software, and infrastructure, require effective project management.

Chief programmers, team leaders, and crisis managers are not substitutes for project managers who are able to estimate, schedule, monitor, analyze and minimize risks, and carry out contingency planning, change management, and team building. Project managers must be able to plan, control, administer, staff, and organize projects. Project managers must be recruited from the user side of the organization and be certified in the skills of project management. Professional project managers must give their ultimate loyalty to their profession, not to the organization that employs them.

Information technology projects require a project management methodology and a process management methodology. The project manager does not need to have a technical background to manage both of these methodologies, nor to recognize the required outputs of each stage of the process, nor to recognize that the outputs correspond to the information needs of the organization. The project manager does, however, need to be able to ensure that the requirements of the people who are actually using and paying for information systems are satisfied. One important technique that

these professional project managers must use to do this is discussed in Millennium Rule #2.

2. All information technology projects, including applications and hardware, software, and infrastructure, require user-centered documentation techniques.

User-centered documentation techniques such as Ivar Jacobson's use case or Ian Graham's task case describe the services, responsibilities, and behaviors of a system from the point of view of a typical user of the system. Traditionally, system requirements have been described in terms of features and services from the point of view of the system. In practice, this traditional approach requires such redundant and time-consuming activities as documenting the requirements, writing user training manuals, developing testing scenarios, and instituting change management and project management reporting procedures. Systems personnel have needlessly spent significant amounts of time trying to analyze and understand the user's current operations in the hope that this will somehow lead to the design of a new system. Users already understand their current operations. What they need is a way to describe their new information system requirements to systems personnel who are experts in implementation.

User-centered techniques can be used to produce a single document in the first few hours or days of a project that records the user's system requirements, the user's training manual, the user's test acceptance criteria, the user's change request form, and the user's key vehicle for project management planning and progress reporting. Proper application of user-centered techniques can speed up system development and allow the users of the information system to be 100 percent involved in the specification, development, testing, and maintenance of their own systems. Errors in specification can be caught early in the systems development process, when they are easy and relatively inexpensive to correct. Implementation begins immediately, starting with the most significant and most technically challenging of the user-centered requirements. If, in trying to implement these first critical parts of the system, it is determined that the rest of the system cannot be developed in an economical, efficient, and effective manner, the project can be modified or abandoned before additional resources are wasted. The real power of the user-centered techniques, however, requires the application of Millennium Rule #3.

3. All information technology projects, including applications and hardware, software, and infrastructure require reuse "in the very large."

Twenty-first century reuse means the reuse of documentation, design, testing, and system interfaces. Reuse of code is not necessarily an appropriate objective. Code will still need to be continually updated and enhanced, even while systems are in operation, as more efficient algorithms and programming languages become available. Instead, effective reuse requires identifying the information services required by the organization's information systems users in terms of previously proven implementations.

Traditional approaches to systems development have resulted in the redundant production of billions of lines of code, each line of which has its own unique opportunities for errors. At a higher level of abstraction, however, these billions of lines of application code apparently perform as little as 10 or 20 different basic information processing functions. Instead of continually embarking

Systems personnel have needlessly spent significant amounts of time trying to analyze and understand the user's current operations in the hope that this will somehow lead to the design of a new system.

on "green field" projects, we need to start the development of new systems with an essential core library of prewritten, tested, proven, generic user-centered requirements documents. Systems specification in this case becomes a process of making changes or adding enhancements to these generic user-centered requirements documents. Ideally, these documents will already have been implemented with generic designs and code, which in turn will just need to be modified and enhanced instead of developed to produce the new system. If we can identify and reuse the essential requirements of a system, we can reuse the previous implementation of these requirements to speed up the development process, cut costs, and improve quality. The real power of reuse in the very large, however, requires the application of Millennium Rule #4.

Patterns are a very powerful way to transfer knowledge.

4. All information technology projects, including applications and hardware, software, and infrastructure, require the use of components.

Components are encapsulated sets of standardized data and standardized methods of processing data that together provide standardized information services. Components fall into three categories: presentation, application, and design. *Presentation* components are used by presentation and interface specialists to assemble user or system interfaces for information systems. *Application* components can be thought of as information models of the real world objects in the information system. These objects include the persons, documents, products, resources, transactions, etc. that are found in the domain of the information system. These application components are responsible for providing the information processing requirements of the application as specified in the user-centered requirements documentation. *Design* components are used by systems designers to provide resources to meet all of the design constraints of an information system. These design constraints include security, response time, availability, legacy system salvage and reuse, and the persistent storage or database requirements of the system. The real power of assembling components into presentation, application, and design constraint systems, however, requires the application of Millennium Rule #5.

5. All information technology projects, including applications and hardware, software, and infrastructure, require the use of patterns.

Patterns are descriptions and examples of proven best practices in the development of flexible, robust, and effective information systems. Patterns offer us alternative ways of assembling previously developed and proven components into new information systems. Patterns are a very powerful way to transfer knowledge. The participants in the system development process do not have to be trained in or even be aware of the underlying theory that is the basis for a particular pattern's success. Instead, the participants, no matter what their level of experience, training, or background, only have to be able to recognize the pattern and understand the tradeoffs involved in employing the pattern. The quality of a system is audited by judging its design with respect to proven design patterns. The real power of patterns, however, requires the application of Millennium Rule #6.

6. All information technology projects, including applications and hardware, software, and infrastructure, require the use of contracted interfaces.

An interface is a description of the information services specified in each step of the user-centered requirements documentation. Instead of trying to analyze all of the complex relationships among all of the possible components of an information system, as is typically done in constructing a data-centric entity-relationship model or an object-oriented class diagram, developers using interfaces only specify the information that is required by the current step of the user-centered requirements. It is possible that one existing component or a collection of existing components will actually provide the services required by this step in the requirements. The allocation of required services to components is made by examining the contract of the required interface with the contracts of the existing components.

Contracts are formal descriptions of the preconditions for using the interface or component, the postconditions, or results, of using the component, the business rules or constraints that will be in effect during the actual processing of information by the component, and the responses of the component to error conditions. Systems are then assembled by matching user-centered requirements contracts with component information services contracts. The results of designing to an interface instead of a particular component implementation are:

- The various components that actually provide the services to implement a particular interface may be written in different programming languages, may be running on different operating systems and hardware platforms, and may even be distributed across the Internet.
- The actual components that provide the services to implement a particular interface may be modified, enhanced, or replaced while the system is in operation. As long as the new component has the same contracted interface as the one that it replaces, the substitution will be transparent.
- Generic systems can be designed to transform and substitute components to accommodate specialized runtime requirements. For example, systems can be designed to be independent of any particular hardware or software environment. At runtime, when the system is actually being used, the collection of components that provide information services could be dynamically substituted to produce a system that runs on a specific hardware or software platform. In effect, a new system can be assembled during execution of the system to accommodate the user's current information system requirements.

YEAR 2000 RECOVERY AND BEYOND

I believe that generic user-centered requirements documentation, components, patterns, and contracted interfaces must become a part of the information technology infrastructure of our organizations and even our global economy in the new millennium. These approaches must be developed and taught in our schools, be taken out and proven in the real world, and then returned to our schools as the teaching medium for the next generation of systems users and developers. Today's redundant training and systems development practices must be abandoned. Instead, training and development must be based upon the reuse of proven practices and proven components assembled into proven patterns.

Today's redundant training and systems development practices must be abandoned.

Richard T. Dué is the president of Thomsen Dué and Associates Limited. He specializes in developing and presenting object technology training courses and in managing and mentoring object technology projects. He is a member of the OPEN methodology consortium.

Mr. Dué can be reached at Thomsen Dué and Associates Limited, 9712 – 83rd Avenue, Edmonton, AB T6E 2B5, Canada. Tel: +1 780 439 4627; Fax: +1 780 431 0332; E-mail: rtdue@ccinet.ab.ca.

These millennium rules are essential for all new systems development. However, they will be vital in the case of the Year 2000 "worst-case" disaster scenario. In the worst case, where critical systems failures have disabled organizations, localities, regions, or even countries, "tiger teams" are expected to be dispatched by organizations, vendors, or government agencies, along with disaster recovery and security personnel, to restore essential systems. Presumably these "tigers" will be the very best information systems trouble-shooters around. But merely having teams of good people will not be enough. It is my belief that these teams must be equipped with all of the concepts and resources of these six Millennium Rules. Certainly, Year 2000 mission-critical contingency planning should include the provisioning of generic user-centered requirements documentation and previously proven components that can be assembled quickly into generic systems within a framework of proven project management processes. Organizations that may be facing Year 2000–related mission-critical system failures should at least begin immediately to develop user-centered documentation of the requirements of their critical systems to facilitate the Millennium Rules recovery process.

Short of an actual Year 2000 "worst case" disaster, how can you evaluate the efficacy of these six Millennium Rules in your own organization? One way would be to construct an all-too-likely scenario in which your organization has developed a system, or has trained people who have developed a system, that is now the subject of litigation. Assume that an expert witness for the plaintiff begins his or her evaluation of your system with an explanation of the six Millennium Rules. Nothing too technical here for a judge or even a jury, just some logical principles for the development of information systems. How will your current systems development process or educational curriculum compare to the Millennium Rules? Did you train your students or employees in professional project management? Did you apply and document the use of this professional project management process? Did you teach or use a systems development and a project management methodology? Did you teach or use a user-centered documentation technique? Did you teach or reuse proven generic templates for requirements, design, testing, change management, training, and testing? Did you use proven components? Were your components assembled according to proven patterns? Were your components connected by contracted interfaces to the user-centered requirements specifications? And, if you did not do all or any of these things, what process you used that will cause the judge or jury or mediator to decide the case in your favor? I certainly hope the process you did use was not the one that has resulted in the industry's record of cost overruns, missed deadlines, and complete project failure rates of 50 percent or more.

Unfortunately for most potential defendants, no university or technical school, no systems development organization, no "tiger team," and no systems development methodology currently incorporates all of the six Millennium Rules into its process. Instead, there appears to be the same continuing fascination with technology, with tools, and with notations, and a continuing reinvention of not very good wheels. Whether we experience a Year 2000 disaster or unprecedented growth in new systems developments, the millennium will rule. It's time to stand aside or join the parade.

