Windows 2000 Dependability

Brendan Murphy
Björn Levidow

4th June 2000

Technical Report
MSR-TR-2000-56

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

To appear in Proc. IEEE International Conference on Dependable Systems and Networks (formerly FTCS), June 2000

Windows 2000 Dependability

Brendan Murphy
Microsoft Research
Cambridge, UK

Björn Levidow
NT Reliability Manager
Microsoft, Redmond, USA

Abstract

Windows 2000 development had two apparently contradictory goals, to add functionality whilst improving dependability to the end customer. This paper describes the difficulty in developing the processes to characterizing the dependability goals for Windows 2000. The paper further describes how Windows 2000 met those goals through addition of dependability features and comprehensive testing in laboratories and production environments. These goals could not be achieved solely through improvements to the operating systems but also required addressing additional issues, such as improving the installation process and quality of driver software. The development of Windows 2000 highlighted a number of research opportunities, discussed later in the paper, to impact the development and release of highly dependable software.

1. Introduction

One of the major focuses of Windows 2000 development was in the area of system dependability. To ensure developers focused on this area the company wanted to set exacting dependability goals. Unfortunately there are no industry standards for characterizing system dependability (an IFIP special interest group, of whom Microsoft is a member, is investigating the development of a dependability benchmark [1]); therefore any goal setting for a particular operating system has to be relative to other versions of operating systems. Microsoft set the Windows 2000 dependability goals to be more dependable at its release to systems running NT 4.0. This added an initial complication as the goal was set relative to a moving target as the dependability of NT 4.0 continued to improve through the release of service packs. A further complication in meeting any goals was the significant amount of functionality added to Windows 2000. Adding functionality, to an operating system, tends to decrease the customer's perception of dependability through newly induced bugs and problems understanding and managing this new functionality.

Setting a user focused dependability goals required Microsoft to radically rethink what was required to achieve these goals. User focused goals require an understanding of how customers perceive system dependability and its drivers rather than solely focusing of technical issues such as bug counts.

Microsoft introduced a program to characterize the current status of Windows NT 4.0 dependability, focusing on the end user perception. A large number of customers and partners were interviewed to capture their perception of the problem areas and provide them an opportunity to request new features. These discussions occurred in parallel to a program that characterized Windows NT 4.0 dependability through monitoring and analysing the system behaviour on the end customer site.

These feedback processes highlighted the need for Microsoft to take a much more holistic view of dependability, tackling it from a system/solution perspective rather than being purely operating system centric.

Improving the dependability of the operating system was achieved through the introduction of new features to address both system failures and to reduce the number of planned system reboots. Where possible these 'features' were reverse engineered into previous versions of NT and windows, improving customer satisfaction but complicating the issue of goal setting. Defect removal through testing was a massive effort (costing $162 million) including stress testing in lab environments, running beta versions on production servers within Microsoft and on customers/ partners systems.

Analysis of Windows NT 4.0 data highlighted the biggest impact on availability is recovery time rather than reliability. New processes were developed to address system availability through tool improvements and providing access to knowledge-based systems.

On-site system management processes also have a big impact on system recovery and these are being addressed through publication of best system management practices. While Microsoft is continually attempting to simplify this system management it is also developing guidelines to assist customers to achieve mission critical production performance through Microsoft Operations Framework (MOF) [2], based upon ITIL [3].

This paper will discuss:

- Benchmarking of Windows NT 4.0 to provide dependability goals.
- Dependability feature added to Windows 2000.
- Testing verification and certification.
- Availability improvements.
- Research opportunities identified during the development and testing of Windows 2000.

2. Benchmarking Dependability

Customer discussions yielded a wish list of features for improving the overall system dependability, not all solely focused on quality improvements in the operating system. Two common trends occurred within these discussions

1. Perception of Windows NT's dependability varied significantly between customers (and for large customers between sites).
2. Lab measurements of Windows NT dependability differed significantly from the customer's perception.

Microsoft set up a monitoring program to capture the dependability of customer systems through analysis of their system logs (event, application and security). Initial analysis proved difficult due to significant deficiencies in NT's system logging process. More importantly these deficiencies complicated on-site problem diagnosis. Greater emphasis was subsequently placed into the fault logging architecture with some additional features being added to NT 4.0 Service Pack 4 specifically:

1. Information on the cause of system outages.
   On a reboot the system logs the reason and time of the system shutdown, identifying if a clean shutdown (event type 6006) or a blue screen occurred (event type 6008). The time the system becomes usable by an application (event type 6005) is also logged.
2. Operating System version and upgrade information.
   Following a system reboot the version of the operating system is recorded (event type 6009) including installations of service packs (event type 4353), if relevant.
3. Cause of system downtime.
   Event type 1001 gives a brief description of the system dump recorded on operating system crash (called blue_screen). Any related Dr Watson diagnosis of the cause of application crash is additionally captured (event type 4097).
4. System Timestamps.
   The system writes a regular timestamp/heartbeat to the system event log indicating its availability. This is required as occasionally the cause and time of the system shutdown is not captured in the error log (in common with other operating systems) making availability calculation error prone.

The monitoring tool Event Log Analyst (ELA) [4] was developed to capture the system error logs from hundreds of servers simultaneously. ELA was deployed extensively throughout the internal systems within Microsoft capturing the current and historic behaviour (the historical time was dependent on the data in the error logs). The tool was further developed and deployed on a small number of customer sites. A number of Microsoft partners had developed similar internal tools to ELA, providing Microsoft with NT reliability data captured using these tools.

The analysis of the behaviour of systems on customer sites highlighted great variability in both measurement and the causal factors.

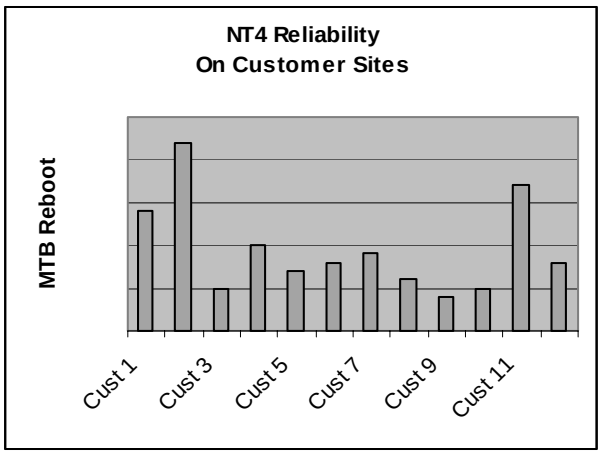

Figure 1. System reliability.

The reboot rate between monitored customer sites varies by as much as a factor of 5 (see **Figure 1. System reliability**). Additionally the reboot rate per system on a single customer site shows a much greater variability. Discussing the results with the customer highlighted two factors

1. The relative reboot rate did not reflect the customer satisfaction or dissatisfaction with the operating system.
2. A large number of site specific and non operating system specific factors affected the monitored reboot rate.

Analysis of the collected data highlighted the difficulty of having a single value for reliability; this is also true for system availability. While the factors affecting system behaviour were similar across the monitored customer sites the impact of the factors on sites and individual systems varied significantly.

A common factor affecting the rate of system reboots across the customer sites was the high percentage of

outages that were planned. A breakdown of the cause of system outages (see **Figure 2. NT 4 cause of system reboots**) on a single customer site running 1,180 servers, highlighting that 65% of all outages are "planned" with only 35% occurring due to system failures.

Whilst planned outages are preferable to unplanned, as the system manager can control their impact on the end users, they still represent a loss of service to the end user and a cost to the customer. Of the remaining "unplanned" outages only 14% are induced system failures. Other reasons for addressing planned outages are that they can often induce additional defects to the system.

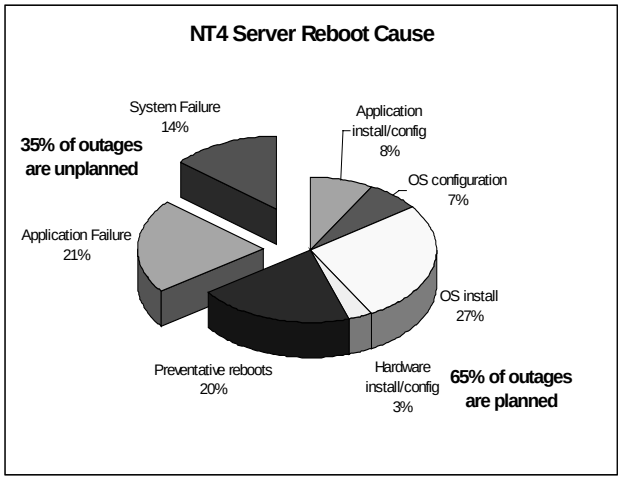

Figure 2. NT 4 cause of system reboots.

Analysis of NT 4 failures reported to Microsoft (see **Figure 3. NT4 cause of system failures**), identifies that 43% of failures were due to core NT defects. This breakdown also highlights the significant impact drivers have on the rate of system failures (a fact well known to most customers and in common with most other operating systems).

A major difference between this breakdown and that observed on OpenVMS systems [5] (and to a lesser extent previously on Tandem systems [6]) is the lack of system management induced failure observed, whereas 50% of OpenVMS failures were due to system management induced defects. This can be explained by the differences in the data collection methods. The analysis of OpenVMS failures was based on detailed analysis of every failure that occurred on the systems. However the NT failure breakdown is based on bug reports. Defects induced by system management activity would be unlikely to raise a bug report to Microsoft as either the system manager would have solved the issue himself or herself or the defect would have been solved by other service providers.

While none of the data collection methods adopted to measure system behaviour highlight the problem of system management induced defects, Microsoft recognizes that NT like all other operating systems suffers a large proportion of these type of failures. A number of collection methods have been attempted to better characterize the cause of failures on the customer site none have been totally successful.

Taking a simplistic approach and combining the two sets of data, the maximum opportunities for improvements in the rate of system shutdowns, through operating system code quality improvements would be 6% of all outages (%System Failure * % Core NT Failures).

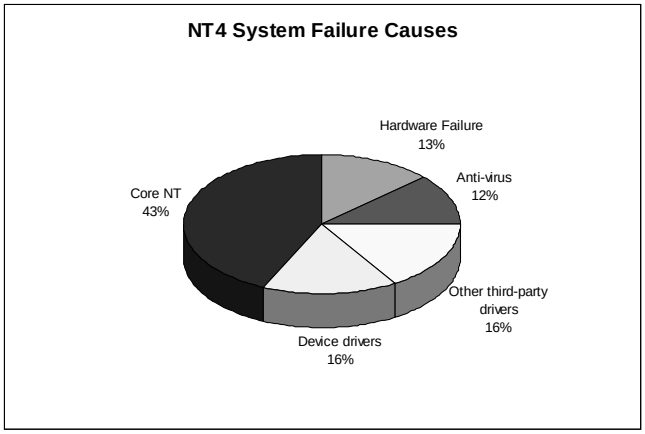

Figure 3. NT4 cause of system failures.

Assuming system events are independent a 6% opportunity for improvement in the reboot rate appears low but is similar to that found analysis of OpenVMS performed 10 years earlier [5]. Whilst the reality is more complex than the calculation suggests as a percentage of the planned shutdowns can be as a result of addressing bugs in NT. Nevertheless solely identifying and correcting system bugs, will not necessarily have a major impact on overall system behaviour. This analysis highlighted.

- Improvements to system dependability required Microsoft to address all systems outages, irrespective of their cause.
- Producing single metrics to characterize the system reliability/ availability/ dependability is very difficult if not impossible.

3. Planned Reboot Reduction

This section describes the features added to Windows 2000 to decrease the number of planned reboots (see **Figure 2. NT 4 cause of system reboots).** Planned reboots are those outages required for system management activities and as such can be scheduled for time periods with minimum impact on the end user. Whilst planned outage allows the system manager control over the timing of the outage they still impact the end user and also contribute to the overall cost of

ownership of the system. More importantly planned outages provide opportunity for errors as any activity is performed outside of the control of the operating system.

The type of planned outage that provides the greatest opportunity for errors (resulting in subsequent system reboots to correct any problems) is the re-configuration of the system through the installation or removal of hardware, operating system or application.

3.1 Software installation and configuration

The installation of software (both operating system and application) presents a number of problems that can significantly affect system dependability, specifically.

- It can be complex and error prone.
  Installation processes often assume the user has in depth knowledge of both the operating system and the application. The installer through answering a number of technical questions can accidentally re-configure the system/application into an error prone state.
- It can corrupt other applications or the operating system itself.
  Early releases of Windows operating system has allowed shared system files to be overwritten by applications, this has been termed DLL hell.

The Windows 2000 development team focused on improving the installation process of both the operating systems and the applications. Previous releases of Windows operating system provided the ability for shared system files to be overwritten by non-OS installation programs. The possibility of a system file being overwritten required the system to reboot, following an application installation, to allow the system to recognize, potentially new, system files. If the system files were overwritten, users would often experience unpredictable system performance, ranging from application errors to operating system hangs or crashes. This problem affects several types of files, most commonly dynamic link libraries (DLLs) and executable files (EXEs).

A significant focus was placed on both simplification and protection of the installation process. Additional effort was placed on working with partners to improve the quality of device drivers and also applications to prevent corruptions occurring (see Section 5. Testing).

The Windows Installer (WI) has been improved through the development of the Windows File Protection (WFP), an independent but related mechanism.

Windows Installer (WI) is a service that manages the installation and removal of applications, providing a basic disaster recovery mechanism through rollbacks. WI keeps a catalogue of all application install files. If, on execution of the application, a file no longer exists WI automatically restores the missing files allowing application execution. This self-healing function reduces the need re-installation of applications if they become corrupt.

The Windows File Protection (WFP) significantly enhances the WI. WFP verifies the source and version of a system file before it is initially installed, preventing the replacement of system files. The WFP additionally runs in the background and protects all files installed by the Windows 2000 set-up program. By default the WFP is always enabled and only allows protected files to be replaced when installing the following

- Windows 20000 Service packs using Update.exe.
- Hotfix distribution using hotfix.exe.
- Windows update.
- Windows 2000 Device Manager/Class installer.

During the installation process WFP verifies the digital signature to validate the new files (see section for details of device signing). If a file is corrupt, WFP retrieves the correct file either from a disk cache of frequently replaced DLLs or from the original installation media.

3.2 Operating system re-configuration

In previous versions of Windows NT many operating system configuration changes required a system reboot. The system would not automatically reboot but request the system manager to perform the task. The system manager may attempt to bundle a number of actions prior to rebooting the system, which may have unforeseen impact on the system dependability. Therefore reducing the numbers of system management activities that require a system reboot is assumed to both reduce the number of planned and unplanned outages.

Windows 2000 redesigned many of the subsystems removing the reboot requirement from 60 configuration change scenarios, some examples are:

- Changing network settings, including:
  - Changing IP settings,
  - Changing IP addresses (if more than one network interface controller)
  - Changing IPX frame type.
- Network addressing, including:
  - Resolving IP address conflict.
  - Switching between static and DHCP IP address selections.
  - Adding or removing network protocols, such as TCP/IP, IPX/SPX, NETBEUI, DLC and AppleTalk.
  - Adding or removing network services, such as SNMP, WINS, DHCP, and RAS
  - Enabling or disabling network adapter.

- Plug and play features for installation or removal of devices e.g.
  - Network interface controllers.
  - Modems.
  - Disk and tape storage.
  - Universal serial bus devices (e.g. mice, joysticks, keyboards, video capture and speakers).
- Installation of applications e.g.
  - Internet Information Server.
  - Microsoft Transaction Services.
  - Microsoft SQL Server 7.0
  - Microsoft Exchange 5.5
  - Device driver kit (DDK).
  - Software developer's kit (SDK).
  - Microsoft Connect Manager.
- System management activities such as
  - Page file management (increasing initial and maximum size).
  - Extending and mirroring NTFS volumes.
  - Loading and using TAPI providers.
  - Docking and undocking laptop computers.
  - Changing performance optimisation between applications and background services.

3.3 Service Pack Slipstreaming

The Windows 2000 installation process allows users to create a single install image, on a network share, containing Windows 2000 and applicable services packs. Users installing from this image do not require a system reboot to install the service pack, as these will already have been "slipstreamed" into the initial install.

3.4 Hardware Install and Configuration

Windows 2000 has a Plug and Play hardware interface allowing users to add hardware, and associated drivers, to a system without requiring a system reboot. When a compatible device is installed on Windows 2000, the Plug and Play manager automatically recognizes the hardware and loads the appropriate devices drivers. While loading, Plug and Play allocates the resources the driver needs to operate correctly.

3.5 Removing Scheduled Reboots

Analysis of NT 4.0 data highlighted numerous customers who were regularly scheduling reboots on their computers (this process is not unique to NT). Frequently, the IT staff cannot articulate why they are performing the reboots other than they 'believe' the machine performs better. For a number of early NT 4.0 releases scheduled reboots did improve behaviour e.g. resource leaks or corruption caused by applications. These are addressed in NT 4.0 through improvements in driver and application quality, Windows 2000 focused on these types of issues through the following features.

Job object API allows application developers to manage a set of processes as a single unit. Developers can place constraints on the unit, such as maximum CPU and memory utilization. These constrains apply to processes explicitly included in the job object and any child processes spawned by the included processes. If an application attempts to perform an operation that would exceed one of the constraints, the call is silently ignored. The job object API also details the resource usage of all of the processes and child processes in a job.

IIS restart allows system administrators to restart the IIS service programmatically without rebooting the operating system. This was previously difficult because of the dependencies of the multiple servers that make up IIS, and the out of process applications that IIS spawns. IIS restart understands these dependencies and handles them transparently to the user.

4. Unplanned Reboot Reduction

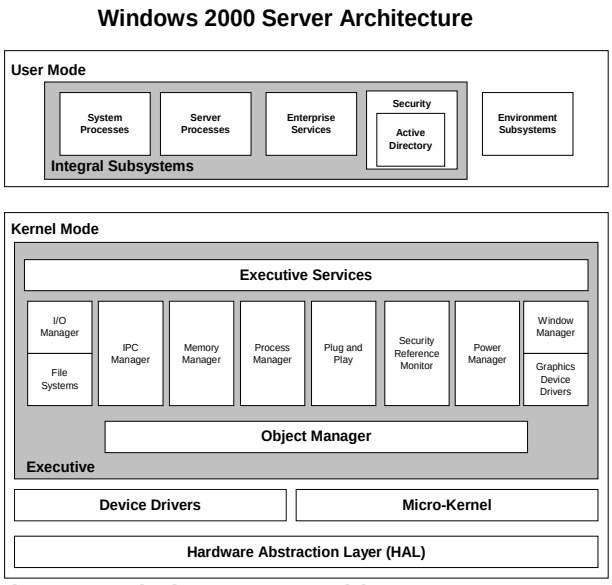

Figure 4. Windows 2000 architecture.

Unplanned reboots (see **Figure 3. NT4 cause of system failures**) are created when the system becomes unstable and in the worst-case scenario this can result in a blue_screen or system hang. Two mayor causes of this is defects in the operating system itself or defects in applications or drivers which result in overwriting operating system code running in the kernel.

The focus during Windows 2000 development is to place as many new features as possible into the user mode and also to improve the verification of software that resides in the kernel mode.

4.1 Application Failures

Unplanned reboots can occur due to applications either corrupting the in-memory operating system or stop responding to requests for service in a timely manner. Windows 2000 incorporates a number of features to address these problems without requiring an operating system reboot.

This is addressed through Kernel memory isolation; the kernel memory space is marked as read-only to user mode applications providing the kernel greater isolation from errant applications.

An addition to the task manager called "end process tree" allows users to kill a process and all of its child processes. In Windows NT 4 if an application stopped responding the system manager could either determine the set of processes associated with the application, killing each manually, or reboot the system. Frequently the simpler option of system reboot was taken, having the side effect of reducing the availability of the system and other applications running on the same machine.

The Internet Information Server (IIS) has become more robust with respect to application failures. By default, all web applications in Windows 2000 run in a separate process from the IIS. Users can elect to run each web application in its own process for further isolation. If any in-process application causes IIS to fail, there is a utility that will cleanly restart all of the IIS services.

4.2 Operating system failures

Microsoft has addressed the issue of operating system failures with a combination of testing, verification and certification. These topics are covered in detail in Section 5.

5. Testing, Verification and Certification

Microsoft invested over $160M in testing Windows 2000, to ensure its dependability, through process and tool development described in this section. The focus has not only been to provide tools and processes for Microsoft developed software but also to provide similar processes to our partners to help improve the quality of their software. A certification process has been put in place to enable customers to identify which applications have been verified using these processes.

5.1 Design Process Improvements

In an effort to improve the design process and code quality prior to testing, Microsoft uses an in-house tool the Prefix source code-scanning tool. Prefix scans source code and identifies potential problems, such as using un-initialised variables or neglecting to free pointers. Prefix identified a large number of issues that were addressed prior to releasing Windows 2000.

A number of tools have been developed for both Microsoft developers and also independent software vendors. These tools focus on software development in kernel and also in user mode.

The following tools and processes have been developed to assist developers to write code in Kernel-Mode.

- Pool Tagging.
  The Windows NT 4.0 kernel contains a fully shared pool of memory that is allocated and released back to the pool. Common errors are for drivers to write to memory outside of their memory allocation resulting in corruption, or to not release the memory on completion of task called memory leak. Both errors are difficult to identify. To assist the developer the pool tagging sets a Guard page at the edge of their allocated memory. If the driver attempts to write into the guard page during testing the developer is alerted.
- Driver Verifier [7].
  The driver verifier is a tool that can monitor one or more kernel-mode drivers to verify that they are not making illegal function calls or causing system corruption. Driver verifier performs extensive tests and checks on the target drivers. Driver Verifier ships as part of the Windows operating system and can be invoked through typing "*Verifier*" at the command prompt. Its settings can be configured using a Graphical User interface (GUI) tool provided as part of the DDK [8].
- Device Path Exerciser.
  The device path exerciser (Devctl) [9] tests how devices handle errors in code that uses the device. The driver can be invoked either synchronously or asynchronously through various I/O interfaces to validate that the driver manages mismatched requests.

The following tools and processes have been developed to assist developers to write code in User Mode.

- Page Heap
  This tool helps developers find memory access errors in non-kernel mode software code. This works in a similar manner to Pool Tagging.

5.2 Testing

Eight million machine hours of stress tests were performed in an effort to eliminate bugs from Windows 2000. These tests can be divided up into two categories: long haul stress and overnight stress.

Fifty machines were dedicated to running long haul stress tests for at least two weeks before upgrading to a newer system build. These tests were designed to find slow memory leaks and pool corruptions. The overnight stress tests had 60 usage scenarios for a variety of component Windows 2000 components. In the weeks before Windows 2000 was released to manufacturing, the Windows team ran stress on over 1500 machines nightly.

Each pre-released version of Windows 2000 was installed onto Microsoft's production data centres, the Microsoft.com website was a very early adopter of Window 2000. Having beta versions of Windows 2000 on internal systems, running in production environments, help in the identification and correction of bugs unique to a real world application with high load.

Along with the automated testing described above, Microsoft employed a full time team to review code in an effort to find potential software problems.

As can be seen in **Figure 3: NT 4 cause of system failures**, hardware drivers, software drivers and anti-virus software drivers cause 44% of NT4 OS crashes. Microsoft has created the following processes to help third parties improve the quality of their drivers.

Driver Development Kit [8] has been improved to include more production quality samples and extra interface documentation on usage.

Vendors of anti-virus and other software that includes file system filter drivers have been participating in special development labs. In these labs, vendors and Microsoft developers use some of the verification tools described below to identify problems and jointly work on solutions. In addition, gathering the vendors together allowed them to discover and address interoperability problems.

5.3 Verification and certification

Microsoft has developed a series of test (Windows Hardware Quality Labs WHQL) to verify that hardware meets the design specifications published in the Microsoft hardware design guidelines. Part of this process includes applying the Driver Verifier tool to the drivers associated with the hardware. Driver Verifier places drivers in a constrained memory space injecting random faults and low memory conditions to stress drivers. Drivers that exhibit problems when run with Driver Verifier are not certified. Drivers that pass the WHQL tests are signed through attaching an encrypted digital signature to a code file. Windows 2000 recognizes this signature.

Windows 2000 can be set to one of three modes to recognize the status of the driver during their installation, these are

- Warn.
  Advises the users that the driver being installed hasn't been signed but allows the user the option to install it anyway.
- Block.
  Prevents all unsigned drivers from being installed.
- Ignore.
  Allows any driver to be installed irrespective of whether it is signed or not.

6. Availability improvements

Improving reliability will not necessarily result in an improvement in availability, as availability is impacted by the system recovery time. The following tools/ processes have been developed to minimize the product down time.

- Recovery console
  The recovery console is a command line console utility available to system administrators. This is useful to allow files to be copied from floppy or CD-ROM to the hard drive or to reconfigure a service that is preventing the system from booting.
- Safe Boot Mode.
  To assist the system administrators diagnose system problems Windows 2000 can be started using safe mode boot. In safe mode only the default hardware settings (mouse, monitor, keyboard, mass storage, base video, default system services, and no network connections) are used. In safe mode the user can request the system to boot under the last known good configuration.
- End Process Tree.
  If an application stops responding the end process tree kills the application and all child processes.
- Automatic System and Service Restart.
  The system can now be set to automatically reboot on system failure. The system can be configured to write the memory contents to a file to assist the system administrator to determine the cause of the failure.
  Administrators can define a specific restart policy for each NT service. The policy allows for the specification of scripts for system recovery, notification of failures via e-mail, etc.
- IIS Reliable Restart
  IIS restart provides a single step to restart the whole of the IIS process.

6.1 System Dumps

Windows 2000 has made system dumping mandatory providing three Systems dump options: full memory dump, kernel dump and mini-dump. The full dump is the same as the one found in Windows NT 4. The kernel dump does not record any of the memory in user space,

significantly reducing the size of the dump on large memory systems. The mini-dump is a 64KB file containing information including the stop code and the loaded modules list.

6.2 Faster Disk Integrity checking

For system problems causing the disk volume to become corrupt, the chkdsk program runs automatically. On large disk volumes, checking the integrity of the volume can greatly increase the downtime of a system. In Windows 2000, the performance of the chkdsk program has increased by between 4x and 8x. For larger volumes, the performance gain is even greater.

7. System Dependability Research

During the course of the development and testing of Windows 2000, the development team spoke to a number of research institutes to investigate the relevance of their methodologies to the development and release of the operating system. The views on the major research areas were

7.1 Software Reliability Engineering (SRE [10])

SRE techniques are used within Microsoft (e.g. within the Microsoft Office development) so it relevance to an operating system was investigated. A number of the SRE processes were viewed as capable of making positive impacts on the software development process, the limitations of the technique from an operating system perspective were

1. Operational Profiles.
   Attempting to develop an operational profile for an operating system is difficult if not impossible. Whilst applications usage is relatively bounded unfortunately operating system are not. Testing the operating system with Microsoft's applications does provide a limited operational profile. But not only are there countless other applications using certain Windows APIs in different ways there are also infinite varieties of hardware and combination of hardware that the operating system and applications are expected to perform.
2. Release process/criteria.
   The system test development of Windows 2000 was based upon regression tests from NT 4, with new tests developed for additional Windows 2000 features. The tests were continually developed up to the release of the product, making the SRE release criteria process impossible to apply. Updating of testing was necessary because
   - Operating systems features are not necessarily used as planned (this is linked to the problems with operational profiles). Therefore the tests had to adjust to reflect the different usage and failure profiles.
   - The testing was focused not only on operating systems failures based on correct usage but also validating stability during incorrect usage. Again the beta testing kept highlighting new ways that customers configured their systems in unforeseen and incorrect ways.
   - New hardware and versions of drivers are continually being sent to Microsoft for testing. These occasionally required changes to the system test process.

7.2 Fault injection

A number of fault injection techniques were considered, none of which were found appropriate to the verification of Windows 2000 (although some are used for some product testing within Microsoft). The issue is the large number of APIs in Windows 2000. Each API has multiple possible fault categories. Even if there existed an infinite time for verification, the rate of change of new drivers and peripheral devices means the items under test are changing daily. Without the existence of an operational profile it is not possible to minimize the number of faults applied.

7.3 Research Opportunities

During the development of Windows 2000 Microsoft looked for information in the following areas and found limited if no information available. Additionally often were information was available it was focused on the fault tolerant space not the high availability area which provides unique challenges. The specific areas, which appear to have limited research focus, are:

- Characterizing System dependability for
  - Individual systems.
  - Multiple systems tightly coupled (e.g. clusters).
  - Multiple systems loosely coupled.
  - Applications (simple and complex).
- Characterizing system (for all system types defined above) dependability drivers especially in the areas of
  - Operating system installations.
  - Application installations.
  - Configuration changes.
- Setting release criteria for complex product development.
- Testing methodologies for high available products.

8. Conclusions

The initial feedback on the reliability of Windows 2000 in the field verifies the success of the testing and release process used for this product. The number of bugs submitted is the lowest proportion for any operating system release from Microsoft. By setting aggressive dependability goals, based on the behaviour on the customer site, required Microsoft to analyse and address total system dependability drivers.

Microsoft will be continuing to develop its development, test and release processes, through analysing the effectiveness of its release process based on the behaviour of the product in the field (i.e. how well was the $162 million spent on testing). A more worrying factor was the limited relevance that system dependability research has in the development and release of this complex high available product.

9. References

[1] IFIP WG 10.4 on Dependable Computing and Fault Tolerance, http://www.dependability.org/

[2] Microsoft's Enterprise Framework, http://www.microsoft.com/msf.

[3] IT Infrastructure Library (ITIL) documented in Central Computer and Telecommunications Agency (CCTA) see http:///www.itil.co.uk/.

[4] Windows 2000 Event Log Analyser, http://win2kready/eventloganalyzer.htm.

[5] B. Murphy, T. Gent, "Measuring system and software reliability using an automated data collection process". Q & R Engineering International Vol 11 341-353 (1995).

[6] Jim Gray. "A Census of Tandem Systems Availability between 1985 and 1990". IEEE Transactions on Reliability, 39(4), October 1990.

[7] Microsoft, Guidelines for using Driver Verifier, http://www.microsoft.com/hwdev/driver/driververify.htm.

[8] Microsoft, Drivers development kits, http://www.microsoft.com/ddk/.

[9] Microsoft, Device Path Exerciser, http://www.microsoft.com/hwtest/testkits/.

[10] Musa J. D.," Software Reliability Engineering: More Reliable Software, Faster Development and Testing", McGraw-Hill, 1998.