Active Server Availability Feedback

James Hamilton

JamesRH@microsoft.com

Microsoft SQL Server

2002.06.12

SQLServer

Agenda

- Availability
  - Software complexity
  - Availability study results
- System Failure Reporting (Watson)
  - Goals
  - System architecture
  - Operation & mechanisms
  - Querying failure data
- Data Collection Agent (DCA)
  - Goals
  - System architecture
  - What is tracked?
  - Progress & results

S/W Complexity

- Even server-side software is BIG:
  - Windows2000: over 50 mloc
  - DB: 1.5+ mloc
  - SAP: 37 mloc (4,200 S/W engineers)
- Tester to Developer ratios often above 1:1
  - Quality per unit line only incrementally improving
  - Current massive testing investment not solving problem
- New approach needed:
  - Assume S/W failure inevitable
  - Redundant, self-healing systems right approach
  - We first need detailed understanding of what is causing both downtime

Availability Study Results

- 1985 Tandem study (Gray):
  - Administration: 42% downtime
  - Software: 25% downtime
  - Hardware 18% downtime
- 1990 Tandem Study (Gray):
  - Administration: 15%
  - Software 62%
  - Most studies have admin contribution much higher
- Observations:
  - H/W downtime contribution trending to zero
  - Software & admin costs dominate & growing
  - We're still looking at 10 to 15 year-old research

Agenda

- Availability
  - Software complexity
  - Availability study results
- System Failure Reporting (Watson)
  - Goals
  - System architecture
  - Operation & mechanisms
  - Querying failure data
- Data Collection Agent (DCA)
  - Goals
  - System architecture
  - What is tracked?
  - Progress & results

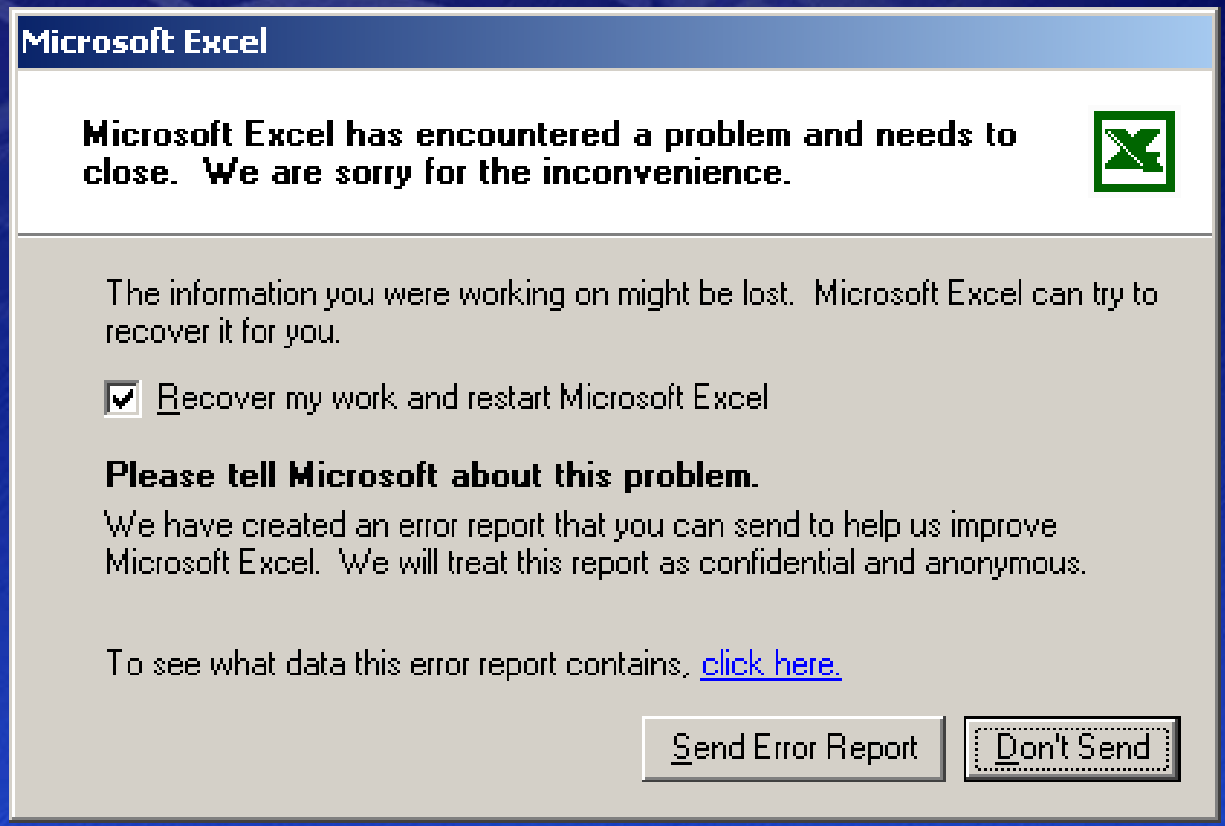
Watson Goals

- Instrument SQL Server:
  - Track failures during customer usage
  - Report failure & debug data to dev team
  - Goal is to fix big ticket issues proactively
- Instrumented components:
  - Setup
  - Core SQL Server engine
  - Replication
  - OLAP Engine
  - Management tools
- Also in use by:
  - Office (Watson technology owner)
  - Windows XP
  - Internet Explorer
  - MSN Explorer
  - Visual Studio 7
  - …

What data do we collect?

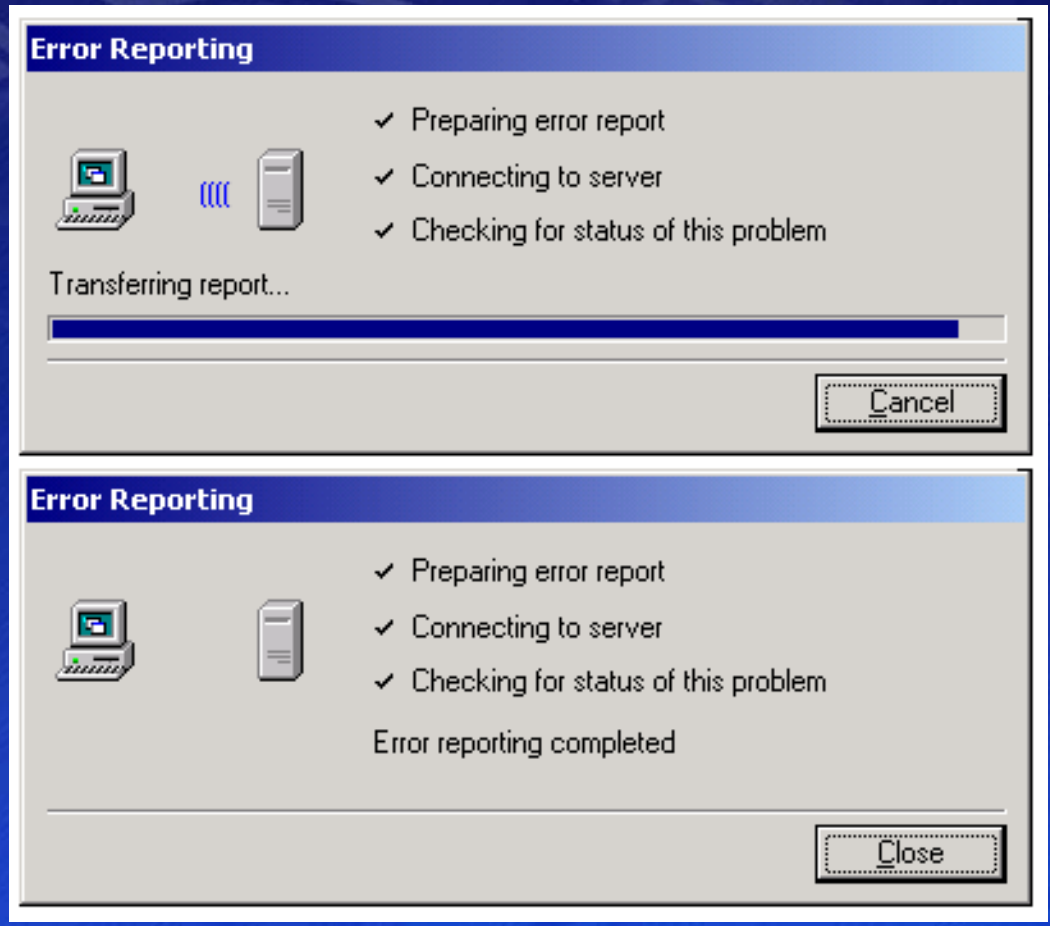
- For crashes: Minidumps
  - Stack, System Info, Modules-loaded, Type of Exception, Global/Local variables
  - 0-150k each
- For setup errors:
  - Darwin Log
  - setup.exe log
- 2nd Level if needed by bug-fixing team:
  - Regkeys, heap, files, file versions, WQL queries

Watson user experience:



- Server side is registry key driven rather than UI
- Default is “don’t send”

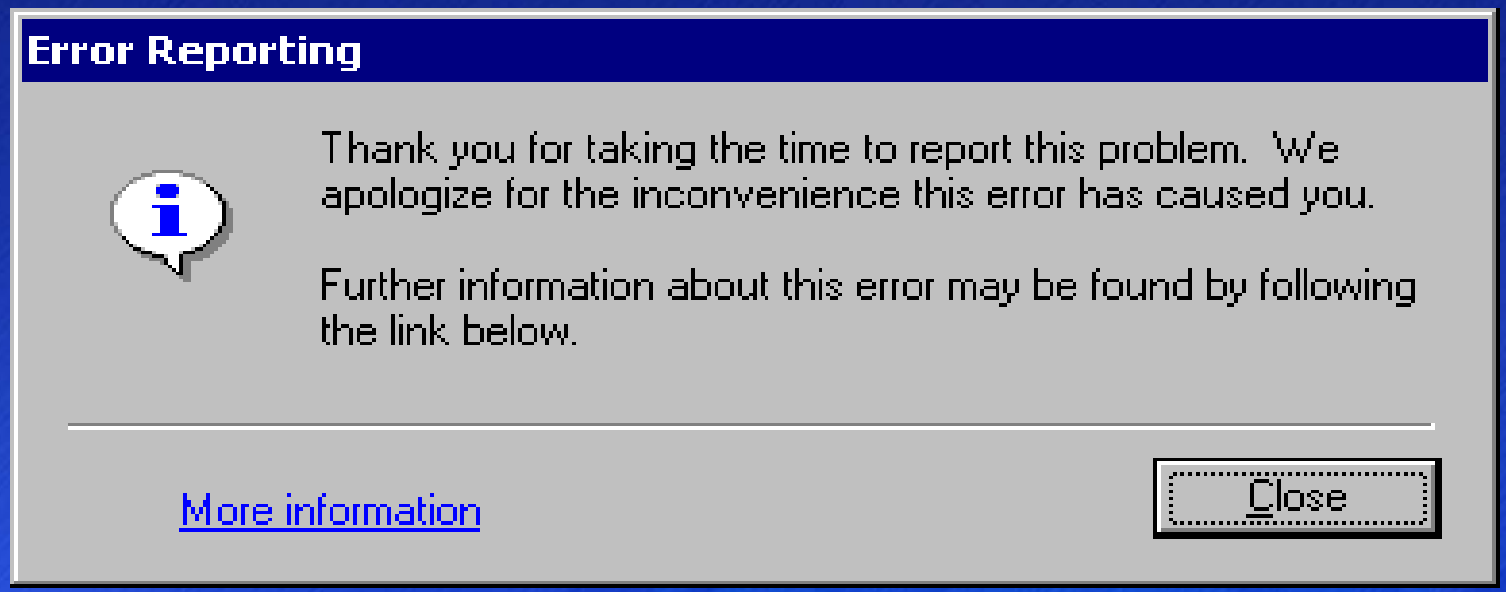
Crash Reporting UI



- Server side upload events written to event log rather than UI

information back to users

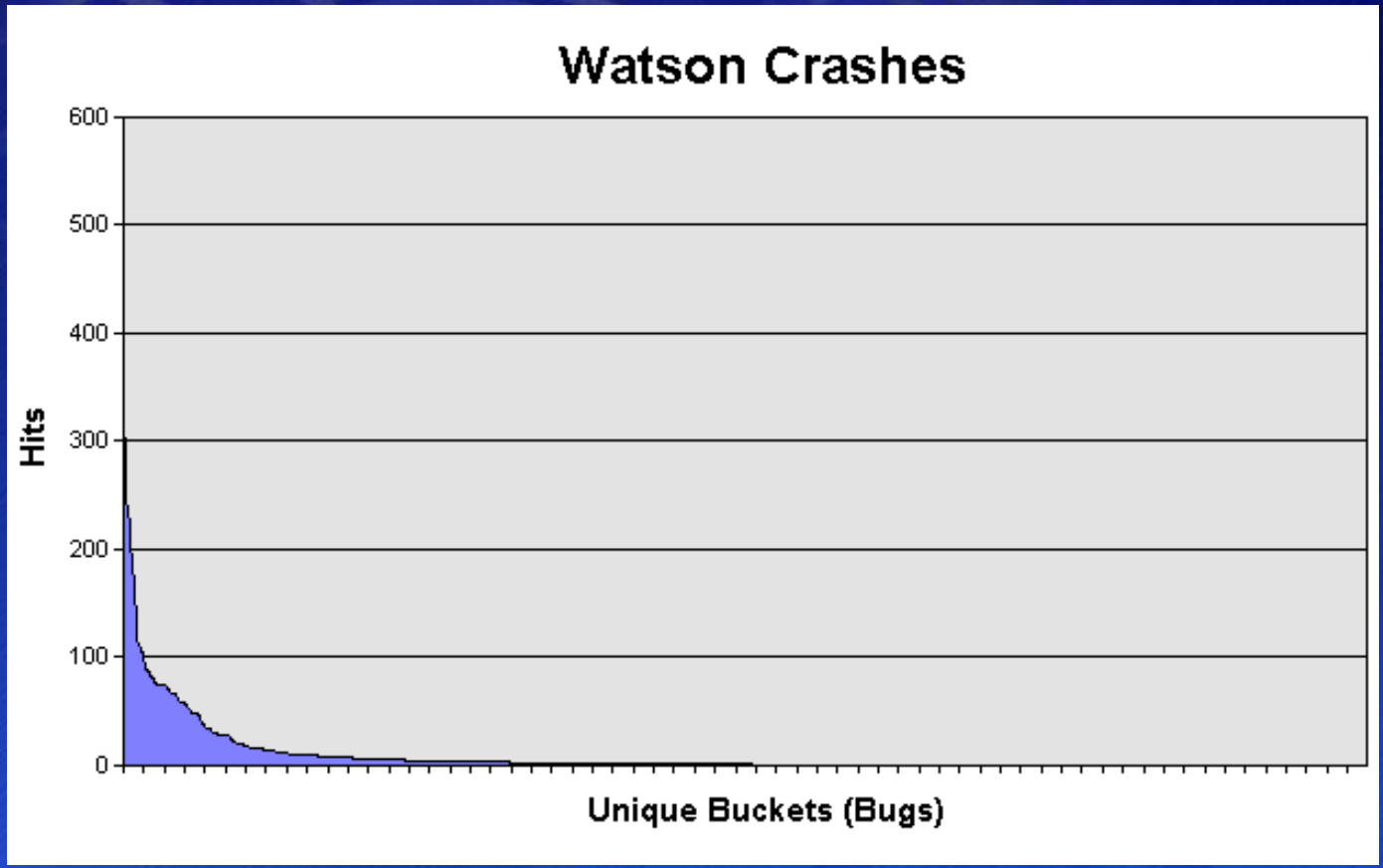
- ‘More information’ hyperlink on Watson’s Thank You dialog can be set to problem-specific URL



Key Concept: Bucketing

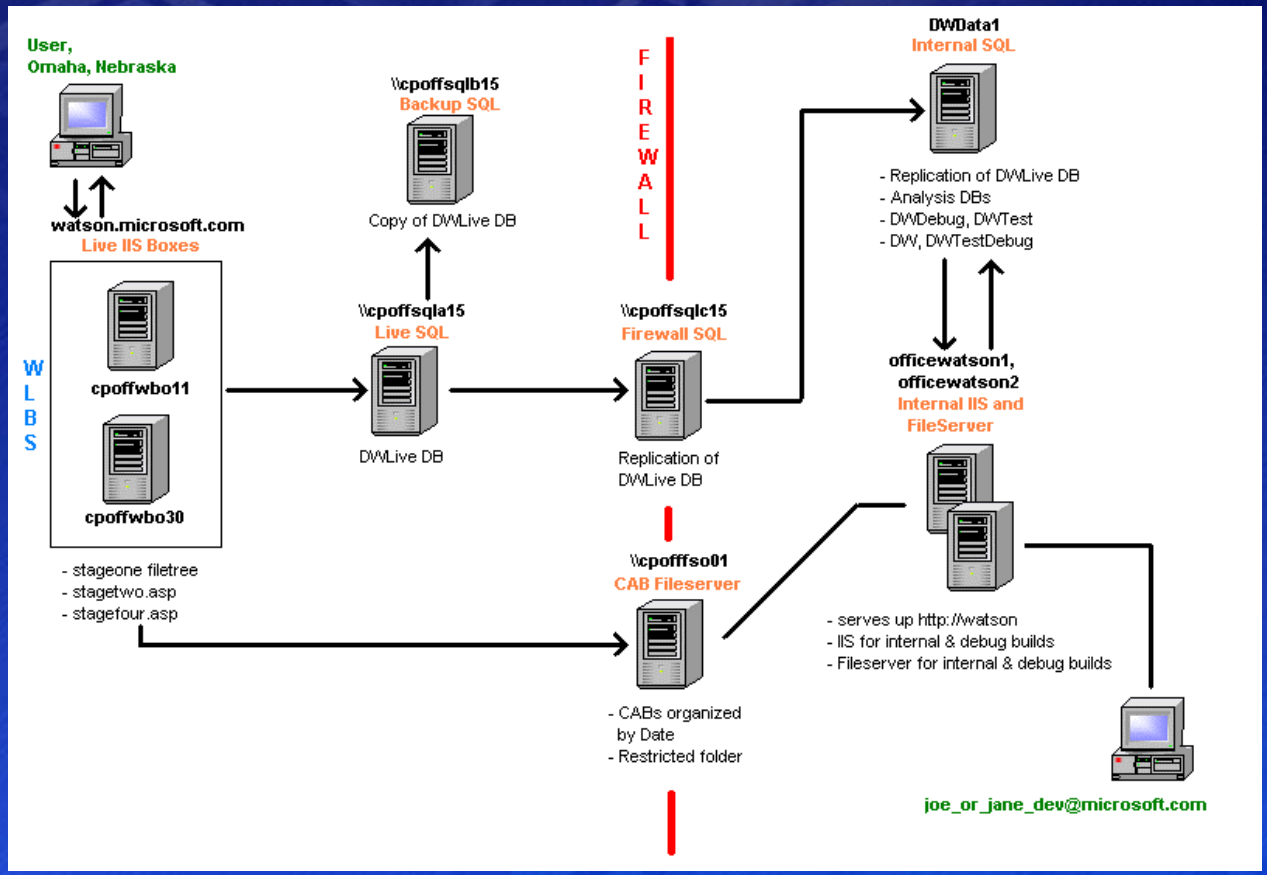
- Categorize & group failures by certain 'bucketing parameters':
  - Crash: AppName, AppVersion, ModuleName, ModuleVersion, Offset into module…
  - SQL uses stack signatures rather than failing address as buckets
  - Setup Failures: ProdCode, ProdVer, Action, ErrNum, Err0, Err1, Err2
- Why bucketize?
  - Ability to limit data gathering
  - Per bucket hit counting
  - Per bucket server response
  - Custom data gathering

The payoff of bucketing

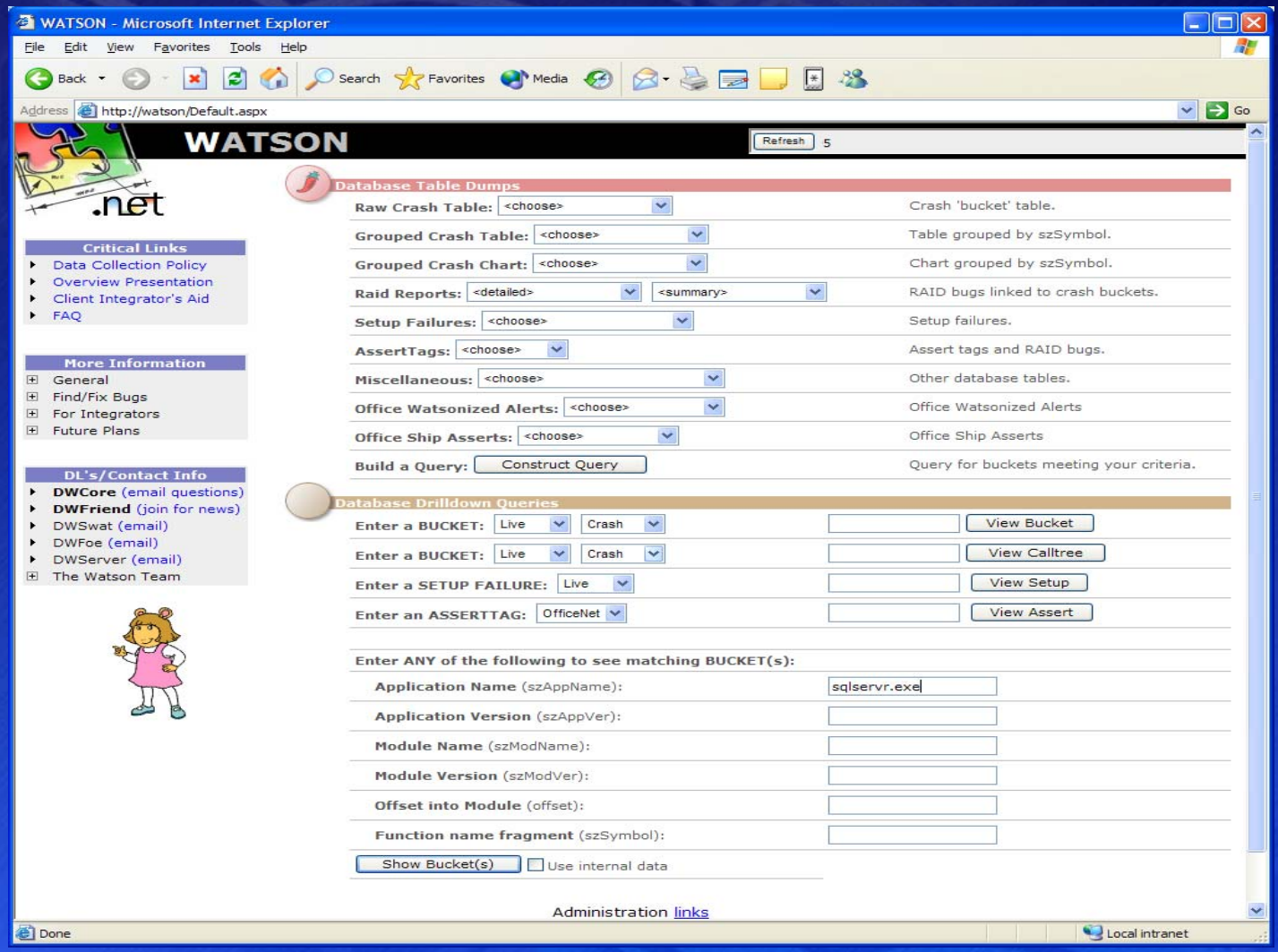


- Small number of S/W failures dominate customer experienced failures

Watson's Server Farm

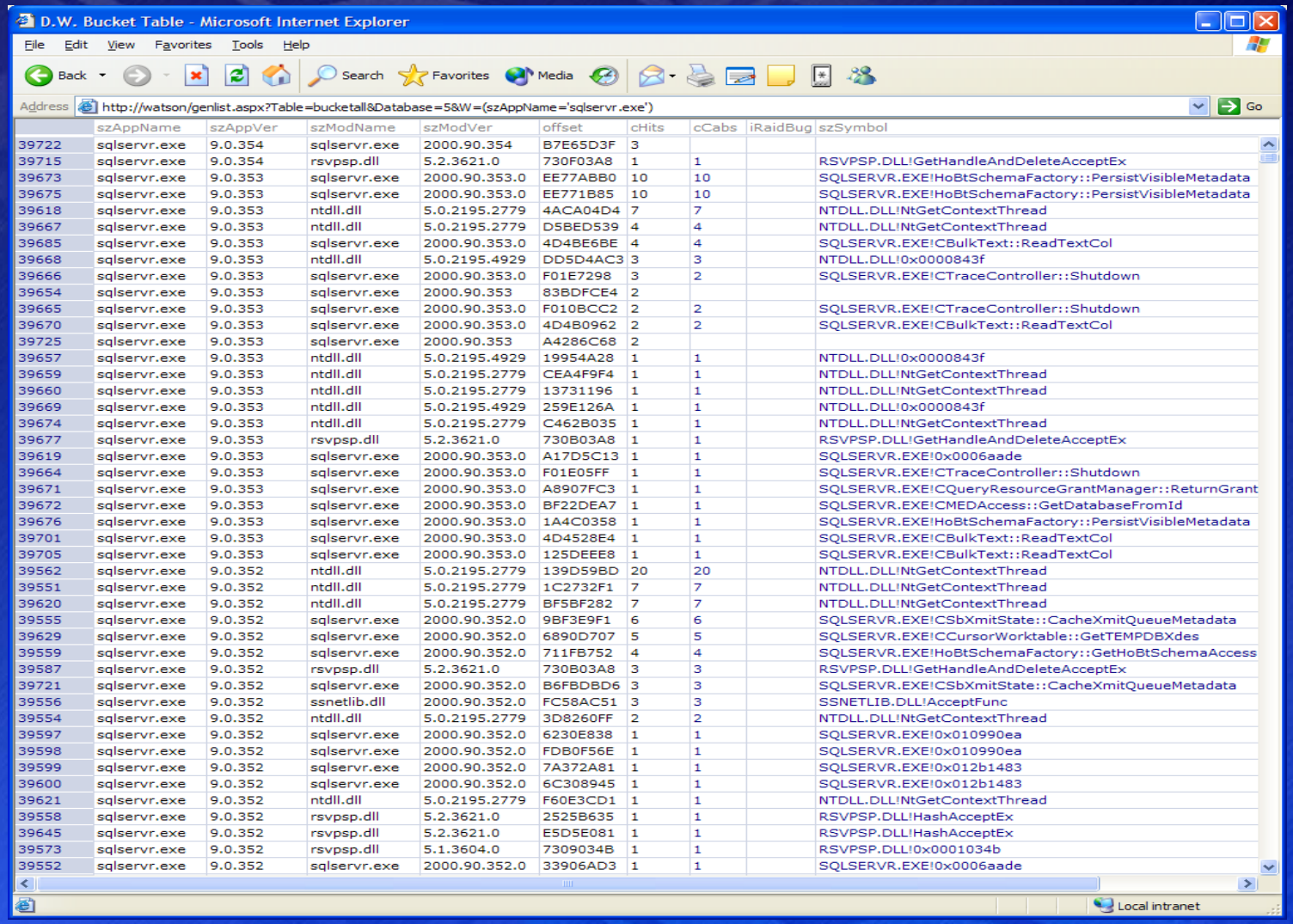


Watson Bug Report Query



SQLServer

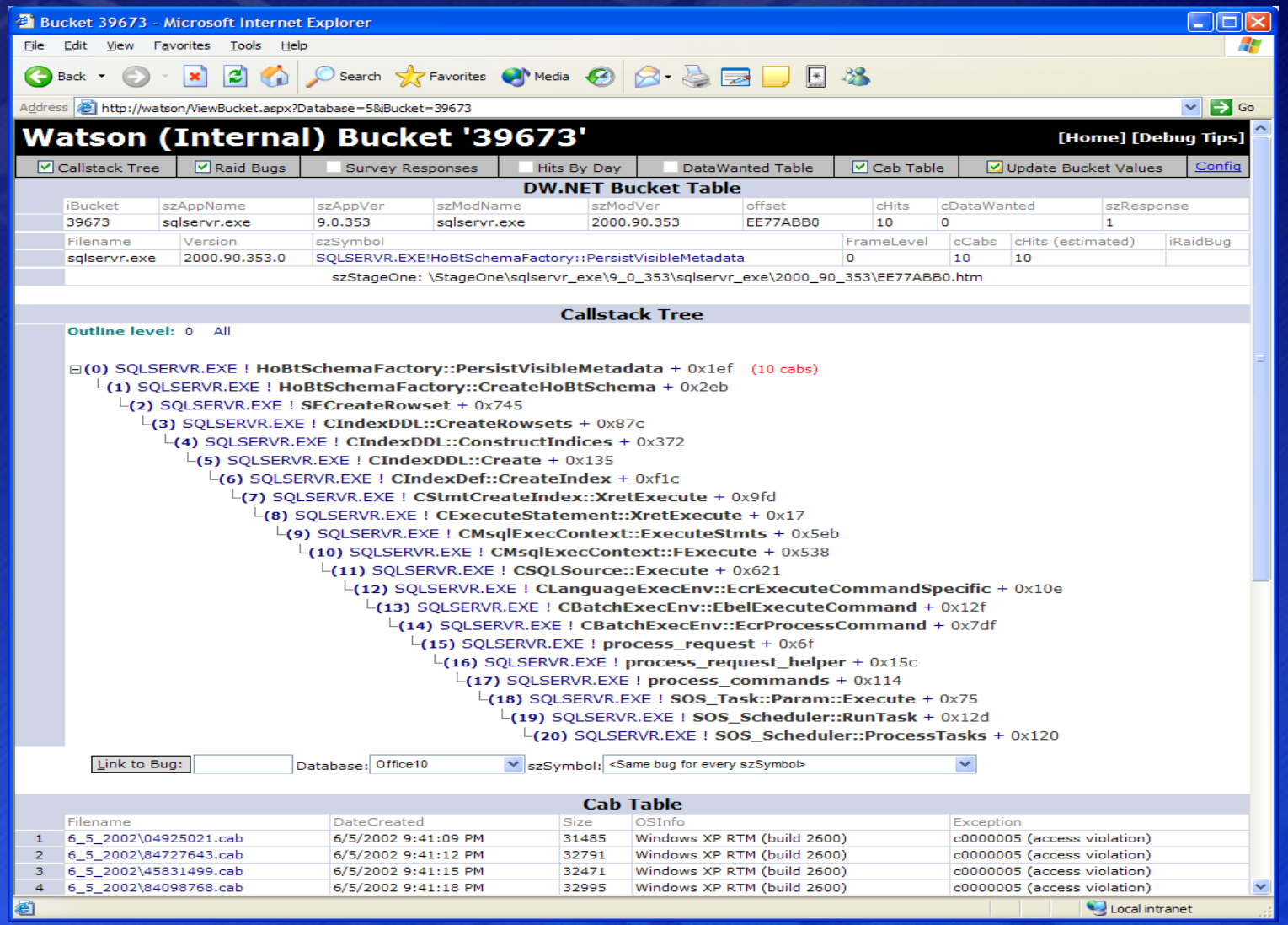
Watson Tracking Data



	szAppName	szAppVer	szModName	szModVer	offset	cHits	cCabs	iRaidBug	szSymbol
39722	sqlservr.exe	9.0.354	sqlservr.exe	2000.90.354	B7E65D3F	3			
39715	sqlservr.exe	9.0.354	rsvpsp.dll	5.2.3621.0	730F03A8	1	1		RSVPSP.DLL!GetHandleAndDeleteAcceptEx
39673	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	EE77ABB0	10	10		SQLSERVR.EXE!HoBtSchemaFactory::PersistVisibleMetadata
39675	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	EE771B85	10	10		SQLSERVR.EXE!HoBtSchemaFactory::PersistVisibleMetadata
39618	sqlservr.exe	9.0.353	ntdll.dll	5.0.2195.2779	4ACA04D4	7	7		NTDLL.DLL!NtGetContextThread
39667	sqlservr.exe	9.0.353	ntdll.dll	5.0.2195.2779	D5BED539	4	4		NTDLL.DLL!NtGetContextThread
39685	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	4D4BE6BE	4	4		SQLSERVR.EXE!CBulkText::ReadTextCol
39668	sqlservr.exe	9.0.353	ntdll.dll	5.0.2195.4929	DD5D4AC3	3	3		NTDLL.DLL!0x0000843f
39666	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	F01E7298	3	2		SQLSERVR.EXE!CTraceController::Shutdown
39654	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353	83BDFCE4	2			
39665	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	F010BCC2	2	2		SQLSERVR.EXE!CTraceController::Shutdown
39670	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	4D4B0962	2	2		SQLSERVR.EXE!CBulkText::ReadTextCol
39725	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353	A4286C68	2			
39657	sqlservr.exe	9.0.353	ntdll.dll	5.0.2195.4929	19954A28	1	1		NTDLL.DLL!0x0000843f
39659	sqlservr.exe	9.0.353	ntdll.dll	5.0.2195.2779	CEA4F9F4	1	1		NTDLL.DLL!NtGetContextThread
39660	sqlservr.exe	9.0.353	ntdll.dll	5.0.2195.2779	13731196	1	1		NTDLL.DLL!NtGetContextThread
39669	sqlservr.exe	9.0.353	ntdll.dll	5.0.2195.4929	259E126A	1	1		NTDLL.DLL!0x0000843f
39674	sqlservr.exe	9.0.353	ntdll.dll	5.0.2195.2779	C462B035	1	1		NTDLL.DLL!NtGetContextThread
39677	sqlservr.exe	9.0.353	rsvpsp.dll	5.2.3621.0	730B03A8	1	1		RSVPSP.DLL!GetHandleAndDeleteAcceptEx
39619	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	A17D5C13	1	1		SQLSERVR.EXE!0x0006aade
39664	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	F01E05FF	1	1		SQLSERVR.EXE!CTraceController::Shutdown
39671	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	A8907FC3	1	1		SQLSERVR.EXE!CQueryResourceGrantManager::ReturnGrant
39672	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	BF22DEA7	1	1		SQLSERVR.EXE!CMEDAccess::GetDatabaseFromId
39676	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	1A4C0358	1	1		SQLSERVR.EXE!HoBtSchemaFactory::PersistVisibleMetadata
39701	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	4D4528E4	1	1		SQLSERVR.EXE!CBulkText::ReadTextCol
39705	sqlservr.exe	9.0.353	sqlservr.exe	2000.90.353.0	125DEEE8	1	1		SQLSERVR.EXE!CBulkText::ReadTextCol
39562	sqlservr.exe	9.0.352	ntdll.dll	5.0.2195.2779	139D59BD	20	20		NTDLL.DLL!NtGetContextThread
39551	sqlservr.exe	9.0.352	ntdll.dll	5.0.2195.2779	1C2732F1	7	7		NTDLL.DLL!NtGetContextThread
39620	sqlservr.exe	9.0.352	ntdll.dll	5.0.2195.2779	BF5BF282	7	7		NTDLL.DLL!NtGetContextThread
39555	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	9BF3E9F1	6	6		SQLSERVR.EXE!CSbXmitState::CacheXmitQueueMetadata
39629	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	6890D707	5	5		SQLSERVR.EXE!CCursorWorktable::GetTEMPDBXdes
39559	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	711FB752	4	4		SQLSERVR.EXE!HoBtSchemaFactory::GetHoBtSchemaAccess
39587	sqlservr.exe	9.0.352	rsvpsp.dll	5.2.3621.0	730B03A8	3	3		RSVPSP.DLL!GetHandleAndDeleteAcceptEx
39721	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	B6FBDBD6	3	3		SQLSERVR.EXE!CSbXmitState::CacheXmitQueueMetadata
39556	sqlservr.exe	9.0.352	ssnetlib.dll	2000.90.352.0	FC58AC51	3	3		SSNETLIB.DLL!AcceptFunc
39554	sqlservr.exe	9.0.352	ntdll.dll	5.0.2195.2779	3D8260FF	2	2		NTDLL.DLL!NtGetContextThread
39597	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	6230E838	1	1		SQLSERVR.EXE!0x010990ea
39598	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	FDB0F56E	1	1		SQLSERVR.EXE!0x010990ea
39599	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	7A372A81	1	1		SQLSERVR.EXE!0x012b1483
39600	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	6C308945	1	1		SQLSERVR.EXE!0x012b1483
39621	sqlservr.exe	9.0.352	ntdll.dll	5.0.2195.2779	F60E3CD1	1	1		NTDLL.DLL!NtGetContextThread
39558	sqlservr.exe	9.0.352	rsvpsp.dll	5.2.3621.0	2525B635	1	1		RSVPSP.DLL!HashAcceptEx
39645	sqlservr.exe	9.0.352	rsvpsp.dll	5.2.3621.0	E5D5E081	1	1		RSVPSP.DLL!HashAcceptEx
39573	sqlservr.exe	9.0.352	rsvpsp.dll	5.1.3604.0	7309034B	1	1		RSVPSP.DLL!0x0001034b
39552	sqlservr.exe	9.0.352	sqlservr.exe	2000.90.352.0	33906AD3	1	1		SQLSERVR.EXE!0x0006aade

SQLServer

Watson Drill Down



SQLServer

Agenda

- Availability
  - Software complexity
  - Availability study results
- System Failure Reporting (Watson)
  - Goals
  - System architecture
  - Operation & mechanisms
  - Querying failure data
- Data Collection Agent (DCA)
  - Goals
  - System architecture
  - What is tracked?
  - Progress & results

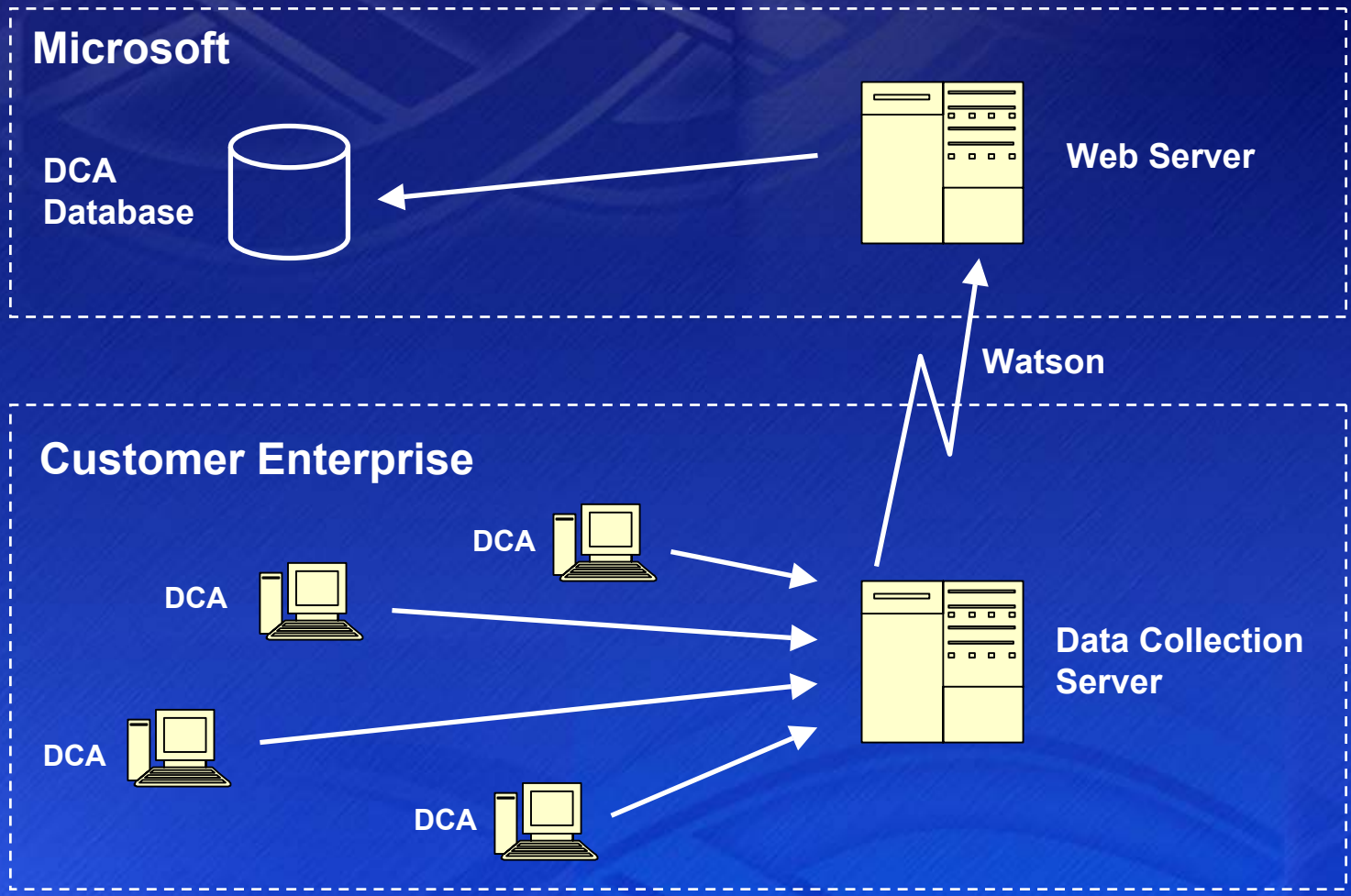
Data Collection Agent

- Premise: can't fix what is not understood
  - Even engineers with significant time with customers typically know less than 10 really well
- Goal: Instrument systems intended to run 24x7
  - Obtain actual customer uptime
  - Learn causes of system downtime – drive product improvement
  - Model after EMC & AS/400 "call home" support
  - Influenced by Brendan Murphy work on VAX availability
  - Track release-to-release improvements
  - Reduce product admin and service costs
    - Improve customer experience with product
    - Debug data available on failed systems for service team
- Longer term Goal:
  - Two way communications
    - Dynamically change metrics being measured
    - Update software
  - Proactively respond to failure with system intervention
  - Services offering with guaranteed uptime

DCA Operation

- Operation:
  - System state at startup
  - Snapshot select metrics each minute
  - Upload last snapshot every 5 min
  - On failure, upload last 10 snapshots & error data
- Over 100 servers currently under management:
  - Msft central IT group (ITG)
  - Goal: to make optional part of next release
- Four tier system:
  - Client: running on each system under measurement
  - Mid-tier Server: One per enterprise
  - Transport: Watson infrastructure back to msft
  - Server: Data stored into SQL Server for analysis

DCA Architecture



Startup: O/S and SQL Configuration

- Operating system version and service level
- Database version and service level
  - Syscurconfigs table
- SQL server log files and error dump files
- SQL Server trace flags
- OEM system ID
- Number of processors
- Processor Type
- Active processor mask
- % memory in use
- Total physical memory
- Free physical memory
- Total page file size
- Free page file size
- Total virtual memory
- Free virtual memory
- Disk info – Total & available space
- WINNT cluster name if shared disk cluster

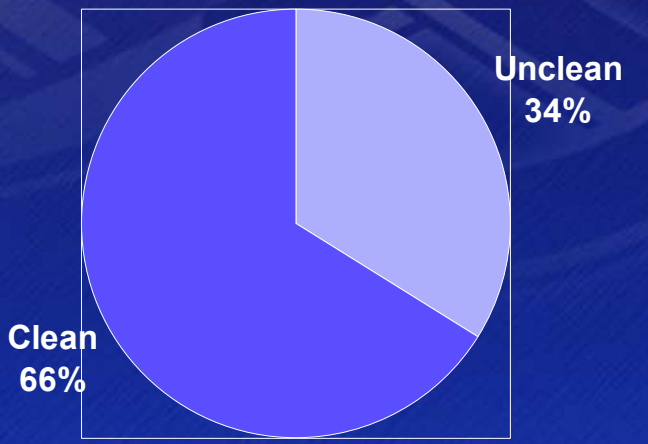
Snapshot: SQL-specific

- SQL Server trace flags
- Sysperfinfo table
- Sysprocesses table
- Syslocks table
- SQL Server response time
- SQL server specific counters
- \\SQLServer:Cache Manager(Adhoc Sql Plans)\\Cache Hit Ratio
- \\SQLServer:Cache Manager(Misc. Normalized Trees)\\Cache Hit Ratio"
- \\SQLServer:Cache Manager(Prepared Sql Plans)\\Cache Hit Ratio
- \\SQLServer:Cache Manager(Procedure Plans)\\Cache Hit Ratio
- \\SQLServer:Cache Manager(Replication Procedure Plans)\\Cache Hit Ratio
- \\SQLServer:Cache Manager(Trigger Plans)\\Cache Hit Ratio
- \\SQLServer:General Statistics\\User Connections

Snapshot: O/S-specific

- Application and system event logs
- Select OS counters
- \\Memory\\Available Bytes
- \\PhysicalDisk(_Total)\\% Disk Time
- \\PhysicalDisk(_Total)\\Avg. Disk sec/Read
- \\PhysicalDisk(_Total)\\Avg. Disk sec/Write
- \\PhysicalDisk(_Total)\\Current Disk Queue length
- \\PhysicalDisk(_Total)\\Disk Reads/sec
- \\PhysicalDisk(_Total)\\Disk Writes/sec
- \\Processor(_Total)\\% Processor Time
- \\Processor(_Total)\\Processor Queue length
- \\Server\\Server Sessions
- \\System\\File Read Operations/sec
- \\System\\File Write Operations/sec
- \\System\\Procesor Queue Length

DCA Results



- 34% Unclean shutdown:
  - 5% windows upgrades
  - 5% SQL stopped unexpectedly (SCM 7031)
  - 1% SQL perf degradation
  - 8% startup problems
- 66% Clean shutdown:
  - 16% SQL Server upgrades
  - 3% Windows upgrades
  - 10% single user (admin operations)
  - 30% reboots during shutdowns

•Events non-additive (some shutdowns accompanied by multiple events)
•Results from beta & non-beta (lower s/w stability but production admin practices)

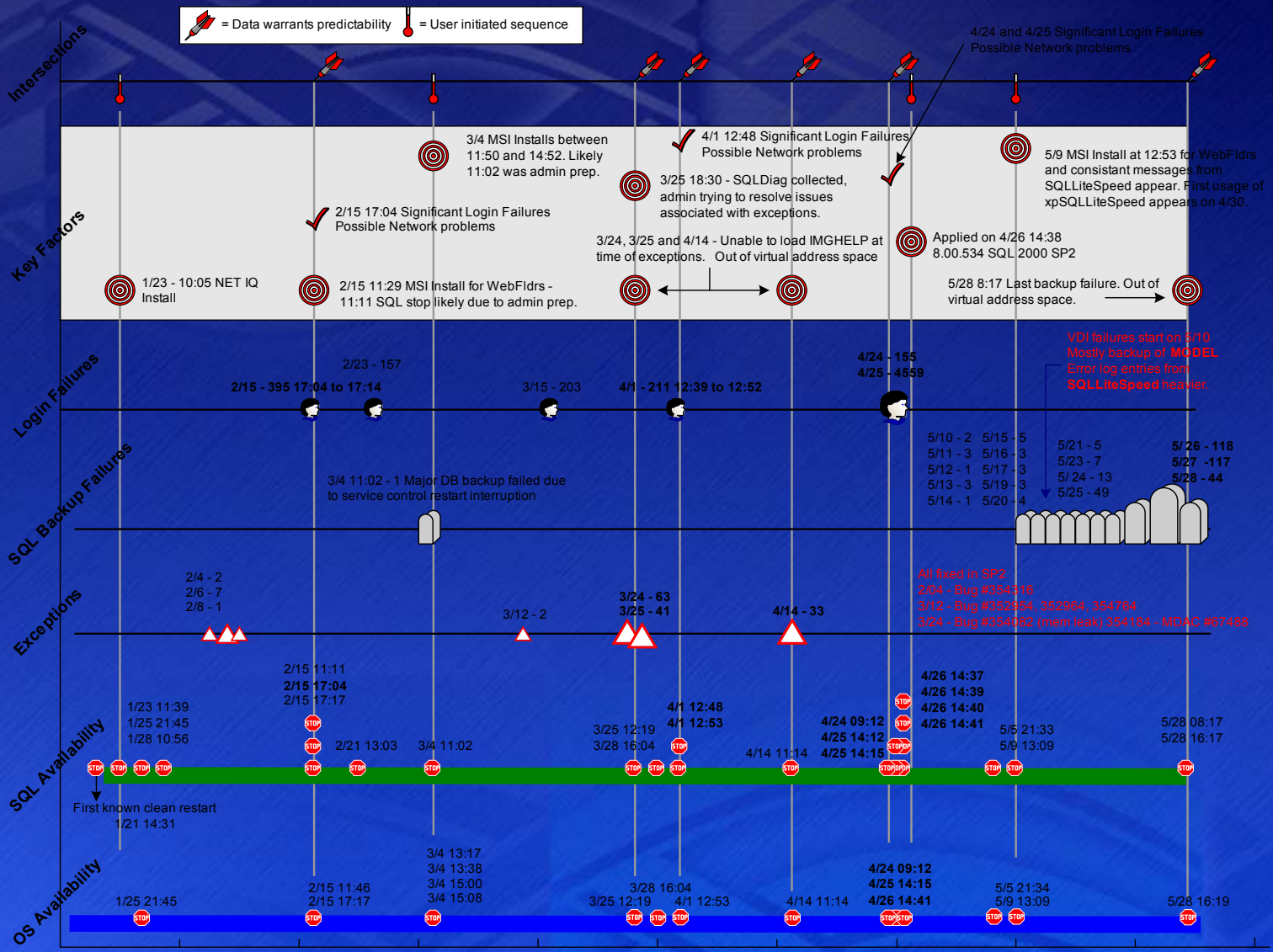
Interpreting the results

- 66% administrative action:
  - Higher than Gray ’85 (42%) or ’90 (15%)
  - Increase expected but these data include beta S/W
- 5% O/S upgrades in unclean shutdown category
  - Note: 5% SQL not stopped properly
  - SCM doesn’t shutdown SQL properly
  - O/S admin doesn’t know to bring SQL Down properly
- Perf degradation & deadlocks often yeild DB restart
- DB S/W failure not substantial cause of downtime in this sample
- S/W upgrades contribute many scheduled outages
- Single user mode contribution significantly
- System reboots a leading cause of outages
  - O/S or DB S/W upgrade
  - Application, database, or system not behaving properly

Drill Down: Data from single Server

- Experiment in how much can be learned from a detailed look
  - Single randomly selected server
- Attempt to understand each O/S and SQL restart
- SQL closes connections on some failures, attempt to understand each of these as well as failures
- Overall findings:
  - All 159 symptom dumps generated by server mapped to known bugs
  - This particular server has a vendor supplied backup program that is not functioning correct and the admin team doesn't appear to know it yet
  - Large numbers of failures often followed by a restart:
    - events per unit time look like good predictor
  - Two way support tailoring data collected would help
  - Adaptive intelligence needed at the data collector

Detailed Drill Down Timeline



SQLServer