Paper 192

Web-based Reporting and Tools used in the QA process for the SAS System® Software

Jim McNealy, SAS Institute Inc., Cary, NC
Dawn Amos, SAS Institute Inc., Cary, NC

Abstract

SAS Institute Quality Assurance faced many challenges with the software validation cycle for Version 7 of the SAS System® Software. Just a few of those challenges were metric collection, reporting, dissemination, and thin client technology for lightweight tool development. To meet these challenges QA turned to the emerging Web enablement in the SAS Software. The QA website evolved into a rich set of interactive reports and tools that increased the quantity and quality of information available to the Quality Analyst, Management, and R&D interested parties.

Introduction

The Intranet has become an integral part of the Quality Assurance process for testing SAS Software. Using SAS/IntrNet® software, QA began creating HTML-formatted reports, then moved to dynamic, interactive reports with live data using the greater integration of the emerging Web enablement in the SAS Software. In turn, the reports have now become lightweight applications that provide drill-down and update abilities that in the past would require multiple applications. These lightweight applications provide complete use of the data. The user can now drill down for more detail in specific areas, and the data can be updated through the report. In the instance of problem tracking this concept is especially significant. Now the user can research a problem with more detail from the problem tracking system. When appropriate, the quality analyst can confirm fixes by adding verification records to the data, completing a common and time-consuming activity.

The move to Web-based architecture frees the user from specific platform requirements. The lightweight application is now available on most of the same platforms as the systems under test. This positions the quality analyst, the target test system, and the lightweight application in an environment to increase productivity.

Adding value and usability to the content of our Intranet has increased the quality and quantity of the information. More divisions used the information for tracking the Version 7 software validation cycle. Research & Development interested parties were asked to critique the site for comment on the information regarding their respective areas. What resulted was content directed at the divisions instead of at quality analysts and management.

Where We Started

Quality Assurance started with the flat file reports common in all businesses in the early 1990's, with a different physical file for each query at greater levels of detail. As the access to a web server became more common, the interest in providing and consolidating more information arose as well. The simplest answer was to provide the graphics found in the QAInfo application on a Web page. These graphics were produced in nightly or weekly batch processes and made easily accessible to the Web page. This stage of Web use was where we stayed for some time. We explored the ability to link graphs in sequence to produce the effect of drill down and data exploration until Web enablement became an emerging feature of the SAS Software.

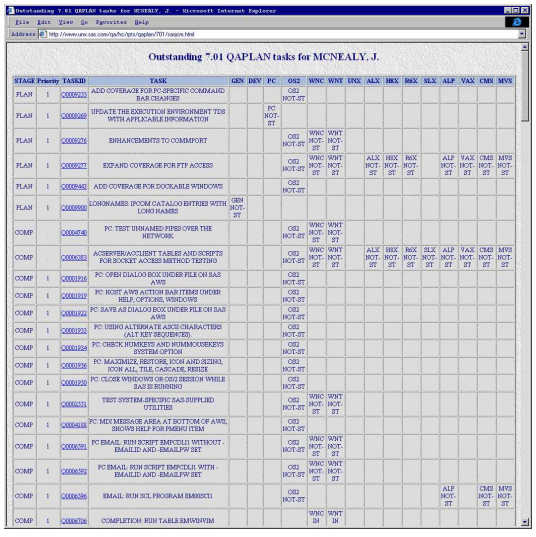
Taking Small Steps

With the release of SAS/IntrNet® and the Web Formatting Tools® in Release 6.12, the ability to easily change the flat file report into a Web page became a reality. Literally over night QA was able to produce HTML-formatted reports to the web server. Initially the static text listing reports were generated as pre-formatted text in HTML using the Output Formatter. Soon after, we began using the Data Set Formatter to generate Web pages with sophisticated tables. These pages were still static, being generated typically overnight and aging during the day.

To provide more real-time data, we began using the Application Dispatcher to query Data Sets

and stream the output directly to the browser. This provided up-to-date information, but still restricted the user to queries built for them by the programmer. As the technology built momentum, data source definitions for common data sets were provided for SAS/IntrNet htmSQL based query forms. This allowed fast and easy dynamic reporting which allowed the user to create the query.

To be effective, the reports need to provide the data to the user to answer a question. In the daily life of the quality analyst the question is often 'what do I need to focus on today?' QA produced HTML-formatted reports from QAPlan, our project tracking application using SAS full-screen products on the UNIX network, to answer this question. The next step was prompted by many of the analysts, 'Can I have a Web-based report that allows me to update the items I've finished?' To help provide this feature the HTML-formatted reports were combined with links via SAS/IntrNet htmSQL software to update the data set as items were finished. Then the reporting code was added to the growing library of Application Dispatcher software to allow the dynamic regeneration of the report. This resulted in the QAPlan report in Figure 1.



Outstanding 7.01 QAPLAN tasks for MCNEALY, J.

STAGE	Priority	TASKID	TASK	GEN	DEV	PC	OS2	WNC	WNT	UNX	ALX	HIX	IbX	SLX	ALP	VAX	CMS	MVS
PLAN	1	Q0009233	ADD COVERAGE FOR PC-SPECIFIC COMMAND BAR CHANGES				OS2 NOT-ST											
PLAN	1	Q0009248	UPDATE THE EXECUTION ENVIRONMENT TDS WITH APPLICABLE INFORMATION			PC NOT-ST												
PLAN	1	Q0009276	ENHANCEMENTS TO COMMPORT				OS2 NOT-ST	WNC NOT-ST	WNT NOT-ST									
PLAN	1	Q0009277	EXPAND COVERAGE FOR FTP ACCESS				OS2 NOT-ST	WNC NOT-ST	WNT NOT-ST		ALX NOT-ST	HIX NOT-ST	IbX NOT-ST		ALP NOT-ST	VAX NOT-ST	CMS NOT-ST	MVS NOT-ST
PLAN	1	Q0009440	ADD COVERAGE FOR DOCKABLE WINDOWS				OS2 NOT-ST											
PLAN	1	Q0009900	LONGNAMES IPCOM CATALOG ENTRIES WITH LONG NAMES	GEN NOT-ST														
COMP		Q0004748	PC: TEST UNNAMED PIPES OVER THE NETWORK				OS2 NOT-ST	WNC NOT-ST	WNT NOT-ST									
COMP		Q0006383	ACSERVER/ACCLIENT TABLES AND SCRIPTS FOR SOCKET ACCESS METHOD TESTING				OS2 NOT-ST	WNC NOT-ST	WNT NOT-ST		ALX NOT-ST	HIX NOT-ST	IbX NOT-ST	SLX NOT-ST	ALP NOT-ST	VAX NOT-ST	CMS NOT-ST	MVS NOT-ST
COMP	1	Q0001914	PC: OPEN DIALOG BOX UNDER FILE ON SAS AWS				OS2 NOT-ST											
COMP	1	Q0001919	PC: HOST AWS ACTION BAR ITEMS UNDER HELP, OPTIONS, WINDOWS				OS2 NOT-ST											
COMP	1	Q0001922	PC: SAVE AS DIALOG BOX UNDER FILE ON SAS AWS				OS2 NOT-ST											
COMP	1	Q0001932	PC: USING ALTERNATE ASCII CHARACTERS (ALT KEY SEQUENCES)				OS2 NOT-ST											
COMP	1	Q0001934	PC: CHECK NUMKEYS AND NUMMOUSEKEYS SYSTEM OPTION				OS2 NOT-ST											
COMP	1	Q0001936	PC: MAXIMIZE, RESTORE, ICON AND SIZING, ICON ALL, TILE, CASCADE, RESIZE				OS2 NOT-ST											
COMP	1	Q0001938	PC: CLOSE WINDOWS OR OS/2 SESSION WHILE SAS IS RUNNING				OS2 NOT-ST											
COMP	1	Q0002331	TEST SYSTEM-SPECIFIC SAS-SUPPLIED UTILITIES				OS2 NOT-ST	WNC NOT-ST	WNT NOT-ST									
COMP	1	Q0004818	PC: MDI MESSAGE AREA AT BOTTOM OF AWS, SHOWS HELP FOR PMENU ITEM				OS2 NOT-ST											
COMP	1	Q0006391	PC EMAIL: RUN SCRIPT EMPCDLLI WITHOUT - EMAILID AND -EMAILPW SET				OS2 NOT-ST	WNC NOT-ST	WNT NOT-ST									
COMP	1	Q0006392	PC EMAIL: RUN SCRIPT EMPCDLLI WITH - EMAILID AND -EMAILPW SET				OS2 NOT-ST	WNC NOT-ST	WNT NOT-ST									
COMP	1	Q0006396	EMAIL: RUN SCL PROGRAM EMMSG1				OS2 NOT-ST								ALP NOT-ST		CMS NOT-ST	MVS NOT-ST
COMP	1	Q0006706	COMPLETION: RUN TABLE EMWNWIM					WNC IN	WNT IN									

Figure 1

The analyst would click on the TASKID of the item to get more information. When the item was complete the analyst would drill to the update form to enter changes in status, shown in Figure 2.

htmSQL-generated QAPLAN Taskid Report

Figure 2

This model worked well for a number of reasons. The users were familiar with the report from the text-based format. The interface traveled to most platforms where the tasks were needed to be performed. The activity was consolidated into a single interface, the browser. It was common to have either a hardcopy or a browser version of the report at hand. It no longer required a separate UNIX application to update the items.

Replicating Our Success

This simple model of consolidating multiple applications into a single interface was successful with the query and update of QAPlan items. Using this success we attempted the same type of conversion with the activity of reporting the verification of fixes in target systems. This was another very common task that often required a report on hand, a SAS application on UNIX to provide detail on the problem, and another such application to update the data. Unlike the QAPlan interface, which required a number of data sets to be updated, the verification required a single data set, making it a good choice for a SAS/IntrNet Application Dispatcher software program.

Alert table | High table | Unprioritized table

Alert Unverified Defects tracked by MCNEALY, J. in 7.01

Obs.	Defect#	Product	Component	History Date	Wanted Platform in Bugver	Wanted State in Bugver	Verify It	Defect
1	S0028718	BASE	PC PRINTING	26AUG98	WNT	DEFER	Verify	TOP MARGIN SOME IF USE BOTTOMMARGIN WITH GRAPHICS
2	S0033246	INTRNET	WEB SASCGI	12OCT98	MVS	DUP	Verify	INSTGN CREATES UNUSABLE OUTPUT ON MVS
3	S0032233	BASE	FILE I/O	29SEP98	DEV	FIXED	Verify	BUILDING THE INDEX FOR THE HELP FILES IS FAILING ON MVS
4	S0029722	INTRNET	GRAPH	16SEP98	DEV	FIXED	Verify	APP SERVER GETS INTEGER DIVID BY ZERO WITH PROC GPLOT
5	S0028136	INTRNET	WEB LOADMOD	24AUG98	DEV	FIXED	Verify	LOADMOD DOES NOT WORK IF ONLY ONE APPSERVER IS UP
6	S0028501	INTRNET	WEB LOADMOD	27AUG98	DEV	FIXED	Verify	EMDSAS DOES NOT WORK WHEN USING LOADMOD FOR LOAD BALANCING
7	S0026909	INTRNET	WEB MISC	19AUG98	DEV	FIXED	Verify	URL.HTML ENCODE/DECODE FUNCTIONS DO NOT WORK ON EBCDIC HOSTS
8	S0026106	INTRNET	WEB MISC	23AUG98	DEV	FIXED	Verify	MEMORY LEAK IN APP SERVER RUNNING UNDER DIGITAL UNIX
9	S0031355	INTRNET	WEB MISC	22SEP98	DEV	FIXED	Verify	WHEREIS_BLD HAS NOT BEEN GENERATED
10	S0022197	INTRNET	WEB SASCGI	17JUL98	DEV	FIXED	Verify	LONG VARIABLE NAMES GET TRUNCATED IN APPSERVER SYMBOLS
11	S0027067	INTRNET	WEB SASCGI	12AUG98	DEV	FIXED	Verify	BROKER HANGS WHEN EMDSAS PROGRAM SENT TO SERVICE WITH TWO SERVERS
12	S0028172	INTRNET	WEB SASCGI	24AUG98	DEV	FIXED	Verify	SRVAUTO DOES IS NOT INCLUDED ON VAX
13	S0026382	INTRNET	WEB SASCGI	26AUG98	DEV	FIXED	Verify	REMOTE OBJECT TEST HANGS THE APPSERVER
14	S0029719	INTRNET	WEB SASCGI	08SEP98	DEV	FIXED	Verify	BAD LIBNAME REFERENCE IN SRVAUTO CAUSES WEBOUT NOT TO BE DEFINED
15	S0030686	INTRNET	WEB SASCGI	10SEP98	DEV	FIXED	Verify	APPLICATION SERVER NEEDS TO PRINT FOUR DIGIT YEARS
16	S0030370	INTRNET	WEB SASCGI	14SEP98	DEV	FIXED	Verify	METHOD DEFINITION IS INCORRECT FOR _PROGRAM=EMDLOADMOR
17	S0030126	INTRNET	WEB SASCGI	10SEP98	DEV	FIXED	Verify	APPSERVER IS BROKEN ON MVS
18	S0031489	INTRNET	WEB SASCGI	21SEP98	DEV	FIXED	Verify	APP DISPATCHER SAMPLE HTML FILES ARE INCORRECT
19	S0032552	INTRNET	WEB SASCGI	30SEP98	DEV	FIXED	Verify	FILESPEC BEING CREATED WITH DEFAULT PORT AS NUMBER INSTEAD OF STRING
20	S0027070	INTRNET	WEB SASCGI	19AUG98	MVS	FIXED	Verify	GRAPH TEST DOES NOT RUN ON MVS
21	S0027660	BASE	PC PRINTING	19AUG98	OS2	FIXED	Verify	CRASHING IN PRINT SETUP FROM EIS WHEN NO DEFAULT PRINTER IS SPECIFIED
22	S0026054	BASE	PC EXT I/O	20AUG98	PC	FIXED	Verify	SAS HANGS WHILE COMMPORT WAITS FOR READ
23	S0029680	BASE	PC NAMEPIPE	04SEP98	PC	FIXED	Verify	NAMED PIPES FAILING SHIPPED SAMPLE 3
24	S0029513	BASE	PC NAMEPIPE	14SEP98	PC	FIXED	Verify	RECONNECT SUBROUTINE NOT FOUND IN V 7
25	S0031627	BASE	PC NOTES	28SEP98	PC	FIXED	Verify	CRASH IN SASNOTE (PCDYNOTE.OYDNHPO)
26	S0024644	BASE	WNT EXT I/O	21AUG98	PC	FIXED	Verify	READ ACCESS VIOLATION IN TASK [DATASTEP] WITH NAMED PIPE SERVER
27	S0029159	BASE	VMS EXT I/O	13SEP98	VMS	FIXED	Verify	APPSERVER HANGS ON VMS
28	S0027398	BASE	WNC PRINTING	20AUG98	WNC	FIXED	Verify	FLOATING POINT DIV BY ZERO ERROR WHEN TRYING TO PRINT PREVIEW WITH A PARTICULAR PRINTER DRIVER

Return to Top

Alert table | High table | Unprioritized table

Figure 3

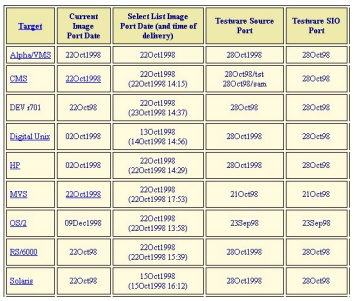
This report in Figure 3 allows the analyst to see the detailed information from the problem tracking system and to use the verification form to add history records to the data set. A column was added to the report for each item. The column is an HTML link to the verification form. The form provides some customization by the user through list boxes for verification status, comments, and platform. The resulting URL includes the pertinent details for the verification program to produce a proper verification record on the problem, shown in Figure 4.

Figure 4

This attempt was also a success, eliminating the need for up to three applications to be navigated by the analyst. The application provided an interface available to any platform with a browser. Because a growing minority in QA used a PC as their primary workstation, this interface migrated with the user. Not to mention the ability to use the application on the same platform where the analyst is verifying code changes.

Broadening the Approach

Take the lessons and apply them to other reports and data sources.



Target	Current Image Port Date	Select List Image Port Date (and time of delivery)	Testware Source Port	Testware SIO Port
Alpha/VMS	22Oct1998	22Oct1998	28Oct1998	28Oct98
CMS	22Oct1998	22Oct1998 (22Oct1998 14:15)	28Oct98/tst 28Oct98/sum	28Oct98
DEV v701	22Oct98	22Oct1998 (23Oct1998 14:37)	28Oct98	28Oct98
Digital Unix	02Oct1998	13Oct1998 (14Oct1998 14:56)	28Oct1998	28Oct98
HP	02Oct1998	22Oct1998 (22Oct1998 14:29)	28Oct1998	28Oct98
MVS	22Oct1998	22Oct1998 (22Oct1998 17:53)	21Oct98	21Oct98
OS/2	09Dec1998	22Oct1998 (22Oct1998 13:38)	23Sep98	23Sep98
RS/6000	22Oct98	22Oct1998 (22Oct1998 15:39)	28Oct1998	28Oct98
Solaris	22Oct98	15Oct1998 (15Oct1998 16:12)	28Oct1998	28Oct98

Figure 5

The Build and Test Status page

The build and test status page, Figure 5, is used by developers and testers to check the current image and testware status of any host. The page provides this necessary information in a clear and concise format. The folks who handle the ports and builds are responsible for updating their information on the page. Knowing that there is a main repository for this information, we have created a tool (image) that allows a user to query the data outside of a Web browser, and a tool (image_updater) that simplifies page updates. Most host groups have host specific build and test status pages, which are linked off the main build and test status page. Without a single source of information regarding builds and tests status, developers and testers would have to hunt in various locations to determine what image and what version of the testware are available on each host.

Defects Web Client

The main problem tracking system used by Research & Development received another integration milestone during the Version 7 development cycle. The Defects Web client was announced. This Web-based application uses SAS/IntrNet software. The application is designed to provide the ability to add and update problem reports through the Web browser.

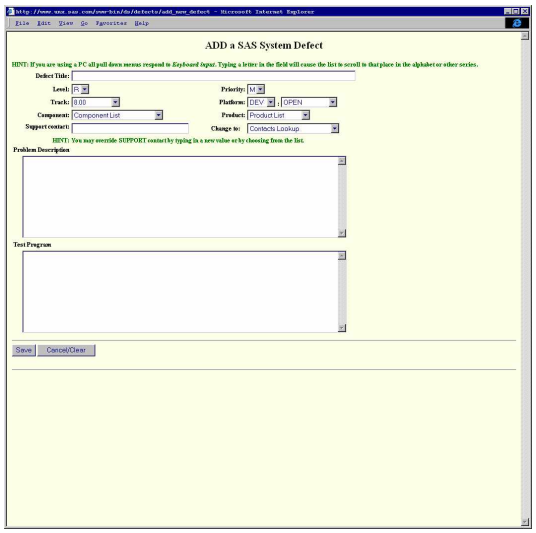


Figure 6

The Defects Web client, in Figure 6, is successful for the same reasons that the previous Web-based applications succeeded. Using a familiar User Interface significantly reduced the learning curve. The analysts could have the Web-based application available on any platform with a browser.

Using Defects Web Client with other reports

Including links to the Defects Web client in existing defect reports helped expedite the review and approval process. During the end of the Version 7 Validation cycle, the Defects Web client proved an expedient method of dealing with the class of problem still to be reviewed. The link in the DEFECTID field of defect reports was changed to bring up the defects Web client. This produced a lightweight application for assessing and approving those changes in the SAS Software that were truly necessary.

Managing software coding projects using the problem tracking system

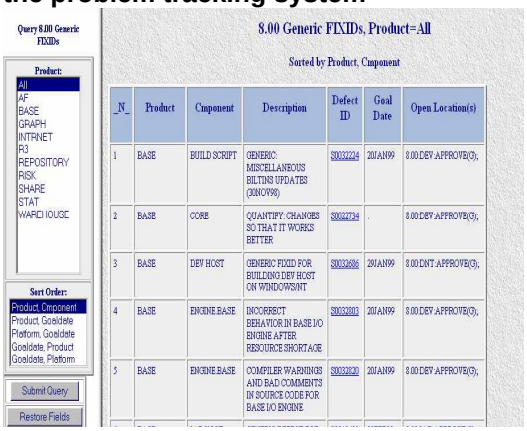


Figure 7

In the past, new features in the SAS System were typically documented by each group writing the feature, scattering the information across multiple locations and in inconsistent formats. Using the new Defects Web Client, we began tracking these consistently at the highest level in the problem-tracking system, one DEFECTID per project; the problem description would supply goal date information and a pointer to the location of additional details. Seen in Figure 7 above.

Our methods for producing reports from the problem tracking system then showed the way to present the software projects to management in a versatile fashion. Using SAS/IntrNet htmSQL and the Data Set Formatter, we provide an initial screen that sorts all software projects by product. Additionally, using frames we also offer list boxes allowing the user to limit the display by product, as well as specify alternate sort orders, such as by goal date or platform. The contents of the list boxes are created dynamically using a separate SQL query, so that the list of products offered is always limited to those that actually have software projects in the works. Finally, we provide the drill-down effect by turning the DEFECTID field into a link to the Defects Web Client page that offers the details of the project. In turn, any URL specified in the detail section of the Defects Web Client page would be drillable by the user, allowing navigation to the complete project documentation. Since this front-end could be linked into any page on the Intranet, for the first time we had a central point for managing the content of the

next version of the software, including evaluating which projects were feasible to include in the release given the development and testing schedule. The major advantage over the past method of managing this information is that the pointers to each level of detail tend to remain live and updated, rather than aging and becoming outdated or misplaced altogether.

Tracking daily code changes

The problem tracking system obviously is also used for identifying defects and specifying their fixes. Developers use the application to send an email signal to quality analysts indicating that a fix is available; the quality analysts in turn determine the required retesting effort, and via the application send an email signal indicating they are ready to accept and test the fix. The process can be susceptible to network or email delivery glitches, or is disrupted if one of the signaled personnel happens to be out of the office for an extended period.

To give management a way to remain current with requests for FIXIDS, we developed a live query, again using the Defects Web Client along with the SAS/IntrNet Application Dispatcher and SAS/IntrNet htmSQL software.

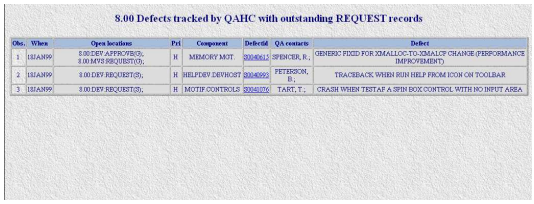


Figure 8

Figure 8 shows the outstanding requests in real time. Again the link to the Defects Web Client is built into the page, allowing drill down to the details of the code fix and additionally to the link allowing the manager or analyst to add the approval record over the Web. These interfaces proved to be quite a bit faster and easier to use when the user's browser was already running but the actual problem-tracking system was not.

QATrack Web Reports

The QATrack tool is used to analyze and resolve batch testing performed on all releases, products, and systems in the QA cycle. The underlying data are stored in SAS Data Sets that are ordinarily accessed via a SAS/AF® FRAME-based application. During the testing cycle associated with the Version 7 release, QA decided to leverage the new ODS and SAS/IntrNet functionality to produce a Web-based system that accessed and updated the same data.

Users of the new system must first define subsetting parameters to limit the batch testing data to the products of interest. For example, a QA analyst might only care to see the results for the SAS BASE® software across all systems in Version 7. These parameters are entered via a SAS/AF® menu and the SAS ODS-based Web report is generated. Once the parameters have been established, this report will be refreshed automatically on a nightly basis via a background job.

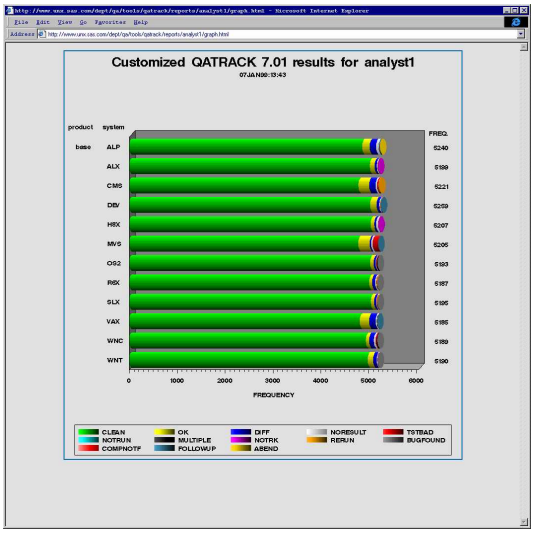


Figure 9

The top-level report produced is a vertical bar chart showing frequency counts for the various test result statuses (clean runs, abends, bad return codes, differences from known benchmarks, etc.). An example of such a chart is in Figure 9. The user might also want to view the same results, but would ignore those tests that ran without problems. Figure 10 shows such a chart. Multiple products could also be displayed on the same report, thus supplying a snapshot of the testing status for large portions of the SAS Software.

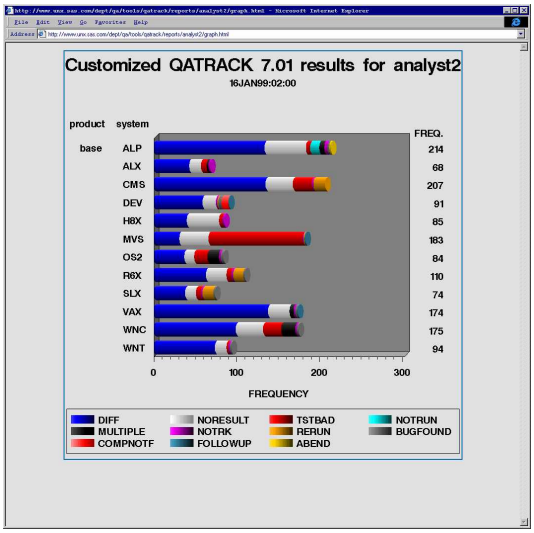


Figure 10

All vertical bar charts are interactive, allowing the user to drill down to a greater level of detail. Clicking on the bars will take the user to a SAS ODS table showing the specific tests run, along with other pertinent information. Figure 11 shows a portion of such a table.

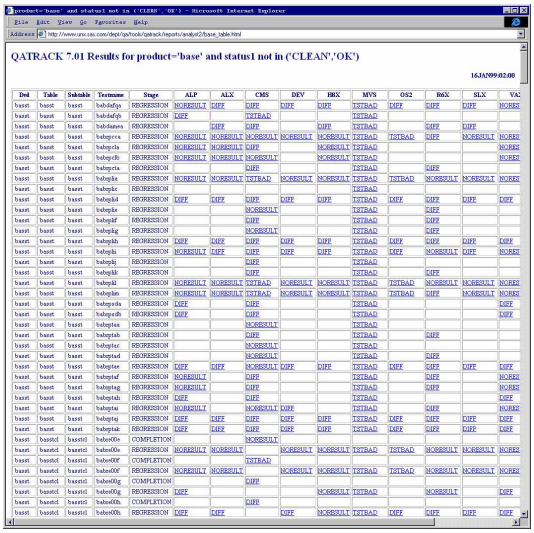


Figure 11

The user can drill one level further by clicking on any of the cells showing test status. This brings up the SAS/IntrNet generated form in figure 12 showing details like when/where the test was run, what output was produced, and which SAS System image was used. Through SAS/IntrNet htmSQL software queries, the form lets the user update key values in the data. Previously, such functionality was restricted to the qatrack SAS/AF software based interface.

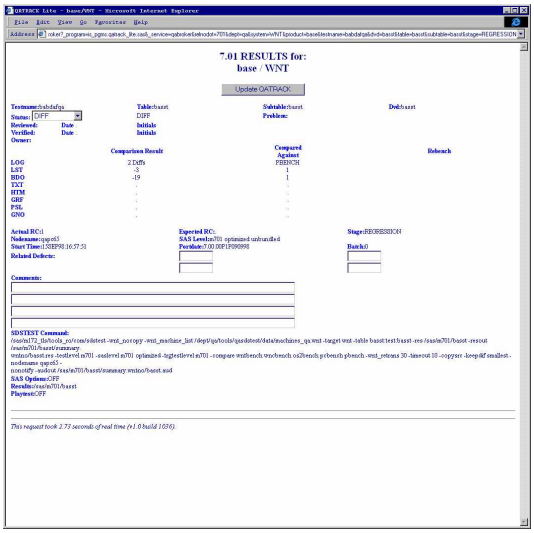


Figure 12

The Web-based approach to the batch testing data allows users to summarize the latest test runs from a variety of hosts on the network, not just the one on which the SAS/AF based application is hosted. Furthermore, for quick editing tasks, the Web-based approach has frequently proven more convenient since users typically have Web browsers running constantly, eliminating the need to launch a separate SAS/AF software based application.

Conclusions

Selective application of Web-based technology has allowed analyst productivity and information access gains for Quality Assurance. Beginning by converting static reports to Web documents, we first improved the visual presentation of the information, then provided access to increasingly live up-to-the-minute data, finally allowing users to create their own queries of the live data as well as offering point-and-click tools for tailoring the presentation for their own purposes. We

learned that managing information over the Intranet in this way short-circuited the tendency of static documents to become obsolete or misplaced. We also obtained unprecedented productivity gains by using the browser as the swift focal point of information delivery, using the power of SAS Software to manage the information, especially since the browser is so universally available and typically more streamlined and responsive than having to use multiple applications. Finally, we discovered that these techniques unified the organization by providing many separate departments a centralized view of the development process, which in turn encouraged enterprise-wide teamwork in shipping quality software.

Acknowledgements

The authors would like to express their appreciation for the details and application images provided by Meg Arnette, Ed Overton, Mark Schneider, and Jeff Simpson all of the Quality Assurance Information Systems group.