From

Automated Software Quality:

Taking Client Server Computing to the Next Level

by
Ben Z. Rose
Ben_Rose@ahh.com

Adams, Harkness & Hill, Inc.
(617) 371-3714

Industry Analysis

Client Server Computing: An Inexorable Trend

Global competition has forced companies to improve their responsiveness to customers or perish in the process. Increasingly, traditional, hierarchical modes of management are no longer appropriate, as decisions must be made quickly and decisively by workers further down the chain of command, many of whom are in favor of more contemporary ways to combat their competition. Information technology has played a key role. Within the corporate enterprise, centralized, mainframe computers have been superseded by lower-cost personal computers, indicating a shift in power away from the control room onto the desktop. Often referred to as client server computing, this paradigm has spawned a new generation of software applications that exploit low cost, distributed hardware, and newer, more visual and intuitive methods of computing. No longer the exclusive purview of the corporate MIS director, critical data regarding sales, finance, manufacturing planning and customer service can now be shared across an entire company from the front line to top management.

The Forrester Group, an industry market research consultant, claims that 60% of the Fortune 1000 have shifted toward client server development, up from 30% in 1990. Several key trends, in our judgment, should continue accelerating the adoption of client server computing over the next several years:

1. **Rising acceptance of Microsoft Windows**. According to Forrester, Windows will have expanded from 64% of all corporate desktops in 1994 to 95% of all corporate desktops by 1999.

2. **Improved price/performance of client server hardware.** Microprocessor price/performance doubles every 12 months. Today, a multiprocessor Compaq Pentium Processor offers twice the computer horsepower of a comparably priced mainframe four years ago.

3. **Standardization on RDBMS engines.** SQL-based relational database management engines from Oracle, Informix, Sybase and Microsoft have become the standard electronic "file cabinets" in which data are stored.

4. **Trend toward team-oriented development.** Distance between and among software programmers has introduced another obstacle to timely release. Whether for advantages of cost, or as a result of consolidation, geographically dispersed teams of software developers are more often than not the norm.

5. **Adoption of the data warehouse.** Corporations with critical legacy data on customer orders, sales trends, financial and manufacturing plans stored in a variety of relational databases wish to exploit this data for competitive advantage by performing queries and providing access to a variety of functional groups within the enterprise.

6. **Rise of corporate intranets.** The increasing adoption of company-specific, intracorporate communication networks will give rise to a new breed of client server applications that will require new tools to develop and test their performance.

Obstacles To Further Adoption

Despite the promise to deliver greater productivity to the enterprise, first generation implementations of client server computing have often failed to meet expectations. According to an industry study by the Standish Group, only 16% of software development projects are completed on time and on budget, and only 42% of projects completed by companies with over $500 million in revenue have the desired features and functionality. About 30% of the development projects are never completed. Moreover, while client server offers the user community the benefit of working independently of the IS organization, a general feeling persists that levels of security, reliability and performance commensurate with traditional mainframe computing do not yet exist.

Development cost overruns and reality's falling short of expectations have become the dark side of First Generation Client Server. While various tools to construct client server applications abound, several obstacles hinder the further adoption and acceptance of client server into the mainstream of corporate computing. In order to move from the trial phase to general deployment, a number of tools to test, simulate, track and fix software defects as part of a total quality solution need to be adopted and applied to the mission of building and deploying applications.

The chief challenges involved in moving to the next generation include:

1. **Client-server complexity.** Rationalizing, or, more frequently cobbling together the various hardware, networking, middleware, database, application and tool components has presented a key obstacle to achieving solid application performance. Like snow flakes, no two client server implementations are exactly alike. Resolving defects and performance issues between and among the various components creates a critical challenge.

2. **Proliferation of application development tools.** Ever larger amounts of software code need to be properly tested and tracked. While Visual Basic and PowerBuilder are often cited as corporate standards, the reality is that thousands of applications are still developed by RAD tools provided by Informix, Oracle, as well as numerous independent, smaller vendors. In addition, various languages and object-oriented frameworks are emerging that promise to increase software complexity.

3. **New Internet tools promise to increase complexity.** Companies pioneering Internet strategies are finding that new languages like Java, despite the promise of alleviating complexity, are actually lengthening the deployment process, due to incompatibilities among hardware and software components.

In summary, the complexity of components involved in the design, build and test process, coupled with the communication obstacles presented by process/workflow and geographical dispersion increase the probability of both lengthening the process and introducing errors and defects.

The Case For Quality

The rising number of requests for new applications in the face of substantial unfulfilled backlogs has placed pressure on IT departments and software developers to bring products to market at an increasingly rapid pace. The increased complexity and time-to-market pressures have placed a strain on the software development process that greatly increases the probability of introducing software bugs or defects into the process, which can cause product delays, lost revenue/lost market opportunities and additional problems.

Faulty, defective software can have various implications. A poorly implemented manufacturing planning system can result in lost market share by leading to inefficient use of manufacturing capacity. The problems associated with the baggage handling system of the Denver airport, and the famous "Year 2000" dilemma, in which faulty database design will cost the U.S. government alone an estimated $1 billion, are best known to the public. However, poor project implementation and software product delays are common in client server implementations. Too often Quality Assurance has meant post-hoc implementation, where quality, to paraphrase the Ford Motor Company, has become Job 1.01.

Moving to the next generation of client server requires a series of software tools to design, build, test, and manage software development. We have defined a universe of second generation client server tools that together will improve the software development process. These tools include products to manage and track application development, test for and detect and report software bugs, and simulate performance prior to product release.

Key objectives include improving software reliability, reducing development costs, and reducing the time to deployment, by eliminating the need for manual testing. Taken together, these products help reduce the time to market for new applications and predict and manage the length of the software development cycle.

The Software Development Process

An overview of the client server application development cycle is useful in understanding the case for software quality. The cycle can be broken down into four discrete tasks: Design, Develop, Integrate and Test. While the tasks are discrete, the process is iterative, as the task of managing, identifying and fixing problems along the way occurs at each of the aforementioned steps.

We offer below an example of how a new sales tracking systems might be developed by tracking the various stages of development:

1. **Design.** In order to determine the feasibility of the application, a "blue-print" is created, with the sequence tied to the company's business so that the application and/or database does not exist in a vacuum relative to how the company performs its operations. During this phase, the basic functions of both the application and database structure are defined and modeled, so that a master specification is broadly understood. In the case of our sales tracking application, a programmer analyst would sit down with a salesperson and learn how sales leads are generated, disseminated internally and externally, how and by whom the prospect is tracked, etc. Through an iterative process, the programmer would map out the "blue-print" of both the application and the database structure, which leads to the next phase.

2. **Develop.** During this phase, professional programmers write the code that will become the fundamental guts of the application and database structure. Programmers could use any one of a number of tools such as Microsoft Visual Basic, Sybase, Powersoft PowerBuilder, Gupta SQLWindows, as well as various languages, such as C, and its variants, to write the syntax and commands that result in the listing and placement of the various fields on the screen, sequences of menus, etc. It is during this phase that the fundamental look and feel of our sales tracking application materializes.

3. **Integrate.** The various pieces of the application-the user interface, programming logic and database are brought to together to create a product prototype. At this time programmers see for the first time how the application works, which parts are defective and require adjustments or fine tuning. While we list this process here arbitrarily, as with the construction of a group research paper, various drafts of the final version can be constructed at any time, with individual members always working on the component parts.

4. **Test.** During this phase, various tests are run to ensure the integrity and performance of the application. It is also referred to as the QA (Quality Assurance) process. Programmers identify defects and assign programmers to fix them. While the tests are tedious and time consuming, this phase is extremely important, because such tests ensure the integrity, quality and ultimate usefulness of the application. In addition, simulation tests are run as well. Whether the application works as planned, slows down unacceptably when too many people access it, or crashes on peak loads, can all be simulated prior to rolling out the application to the targeted user base. Automated software quality products are targeted most frequently to this stage, as the benefits of automating the process reduce time to deployment considerably.

5. **Beta site /remote testing.** Just as there are no formal boundaries between conceptual design, establishing feasibility and testing, what constitutes a "final" product in the software business tends to range from vendor to vendor. How precisely and meaningfully the product has been tested, by how many users and under what circumstances determine the ultimate success of a product. However, time to market pressures and limited budgets often place strain on projects, and a fairly arbitrary product release schedule, often negotiated under duress, can impact a product's performance and usefulness. Ideally, a formal trial or beta test period

involving actual users of the sales tracking system will identify as yet concealed flaws or defects, presumably to be incorporated into the final product.

6. **Product release/deployment.** During this phase, the application moves from prototype to roll-out as a production ready release. This "completed" version of the product is now ready to be used by the entire sales department to increase productivity and generate business.

Automated Software Quality: Defining the Category

Historically, software has been developed using a variety of languages and programming techniques. Code changes in development teams were tracked through a series of ad-hoc procedures, failing to capture the intent or methods used by key programmers in building or enhancing products. Various companies have developed home grown systems, or process control tools, that have tracked software bugs and the need to test, although these have proven to be inadequate.

Just as the software programming task is evolving from black art to profession, so is testing transitioning from manual to automated methods. The need for a new class of products to ensure the integrity of the products under development has therefore become compelling. While the need for quality assurance has always been apparent in software development, increasing complexity has heightened awareness of the problems involved in the development and tracking of client server applications. While one could argue that the need for quality assurance is confined to the testing phase, we would argue that quality assurance is required throughout the development process. Because of the need for quality at every phase-design, build, deploy and manage-we believe that testing tools address a broader audience than application tools and that these products will be used in the up front development process, rather than in isolated testing stages.

Another factor supporting our view that the ASQ market provides strong growth potential is the broad set of capabilities that are evolving into a group of discrete products. We list below the principal products comprising the category:

Configuration management/version control. Configuration management tools and version control programs enable organizations to track all aspects of the application development process, including what changes, how it changes, and whom will be affected by the change. Configuration management tools address a broad spectrum of team based developers, and are increasingly important, given the large volume of code generated on various projects as well as the difficulty of keeping solid programmers in house, due to the rising demand for good software talent.

Error detection. These products detect and monitor applications for execution errors and memory leaks, critical to diagnosing applications that crash repeatedly.

GUI testing. Given the wholesale move from cryptic character based applications to those based on the Microsoft Windows GUI, GUI testing is generally recognized as the largest category within ASQ. A number of firms including Mercury, SQA and Segue have developed a series of proprietary testing scripts to automate the testing process by simulating whether a particular menu sequence acts as planned. These products dramatically reduce the amount of manual regression testing in the development process by simulating test cases by end users that identify flaws or bottlenecks.

Performance testing. By deploying a prototype on a series of desktop computers, or a single server, this class of products simulates how a product will perform under various conditions of load and stress, whether the fifty-first user will fatally "hang" the server or wreak havoc on the productivity of the first 50 users.

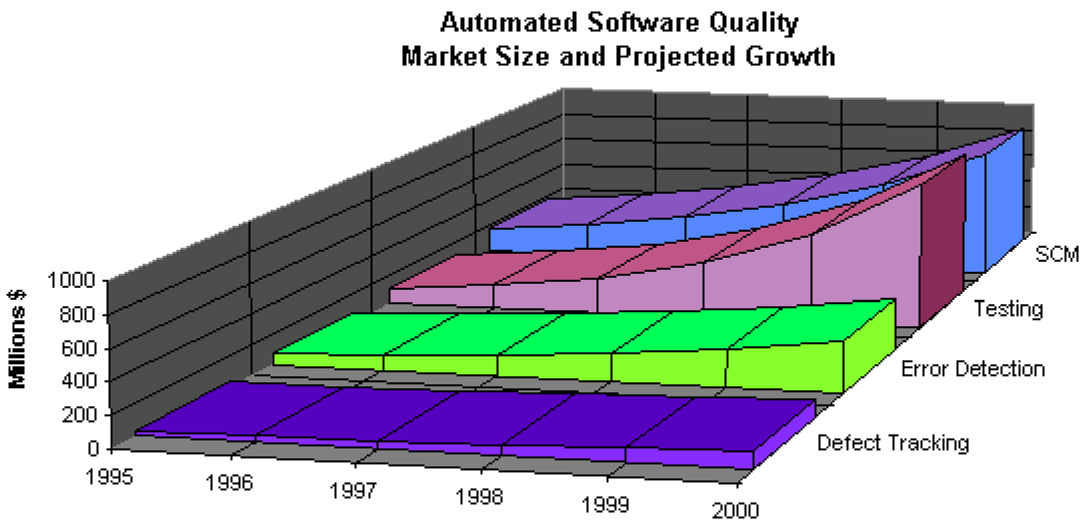
Defect tracking. This class of products identifies software defects, or bugs, and assigns a priority and person to resolve the issue.

Remote beta testing. This new class of products remotely monitors a trial group of users. The information collected on how users access the parts of an application, in terms of frequency of use, etc., can be reported back to development teams to incorporate into the next release.

Test management. Just as workflow products track the business process flow within organizations, so do a new class of process management products, that track the development process from start to finish. We believe such products may enable organizations to identify bottlenecks to productivity, as well as assign responsibility to improve efficiency in the development process.

Market Size

We segment the Automated Software Quality (ASQ) market into four categories: software configuration management (SCM), software testing, error detection and test management. We believe the ASQ market was a $380 million revenue opportunity in 1995 (license and services), with SCM the largest segment ($170 million) and testing the second largest ($105 million). Our estimates include tools for the workgroup and enterprise client server markets (UNIX, NT, Windows 95 and Windows 3.x-based tools), and do not include mainframe-based debugging, CASE and testing tools. We believe the ASQ market will grow at a rate exceeding 40% compounded annually over the next five years, and will become a $2.3 billion market by the year 2000. Within the ASQ market, we believe the two fastest growing segments are testing and SCM, with each segment slated to grow at a rate exceeding 50% per year.



The quality space has been viewed primarily as discrete sub-segments, with leaders in each group. For example, Atria Software is the leader in high end configuration management software, through its ClearCase product offering. Mercury Interactive, the pioneer in software testing, is the leader in its category. In the emerging segment of error detection, Pure is the market leader. We believe that the market has passed from the introductory phase to the growth phase within the life-cycle of the industry.

At this stage of market development, the ASQ business remains highly fragmented without a single vendor holding more than 12% of the market.

As the market matures and the need for a total quality solution to software development becomes apparent, we believe that a unified, agreed upon definition of the category will emerge. We believe that the category will continue to evolve, and would not be surprised to see more tools introduced to alter the definition of the category.

Critical Issues

As the ASQ Category evolves, we believe the following issues must be constantly monitored and revisited in order to draw conclusions about the growth of the industry.

- **Moving up the food chain.** While ASQ tools have been largely isolated within software quality assurance groups, we believe the opportunity exists for a class of diagnostic and bug prediction tools that isolates software defects before they are identified downstream. Several tools like Pure Software's Purify, a product to detect memory errors, are already being used in such a fashion. We believe that more tools will be developed to assist the professional programmer.

- **Rate of customer acceptance.** While the industry is still in its infancy, we believe that several issues will determine the rate of customer adoption, and by consequence the industry growth rate. These include product ease of use, which determines how rapidly customers will move from prototype to deployment, as well as the process change required to implement the tools, including training of programmers and testers to execute their jobs differently. Product pricing will determine the number of users within an organization. The financial health of vendors supplying the industry as well as the desperation tactics applied by one or more vendors to gain share in the "early innings" of market development will all impact growth.

- **OS Wars: UNIX vs. NT/Win 3.1 vs. Win 95.** While Microsoft Windows has become ubiquitous as the desktop operating system and user interface within the commercial and consumer environments, UNIX remains pervasive at the server level inside commercial firms prototyping and deploying client server applications. Moreover, UNIX remains entrenched on the desktop within scientific and technical computing environments. We believe that the desktop will eventually migrate completely to Windows 95, and that NT will become the server operating system of choice. This impacts the ASQ vendors in several ways: ASQ tools primarily oriented to the client/desktop will need to become Win95 enabled very rapidly in order to succeed, while server based tools will need to be NT enabled.

- **Suites vs. discrete tools.** While it took desktop productivity applications several years to evolve from discrete to a bundled status, it is remarkable how quickly the ASQ space has evolved from discrete applications, i.e. GUI testing, to a more complete series of capabilities comprising a suite. Just as several last generation CASE vendors became "screen scrapers", so do ASQ vendors risk becoming pigeon-holed into a single category without the ability to compete on a broad range of capabilities.

- **Market segmentation.** Already vendors are targeting either end-user/MIS, ISVs or embedded systems developers with a range of products. Understanding a vendor's key strengths and weaknesses within a product category as well as a customer set will be critical in determining long range success.

- **Impact of Microsoft.** As the dominant global vendor of corporate and consumer software, Microsoft already develops Visual Basic, the most widely used application development program on a unit basis within the corporate environment, as well as products for version control and testing. While Microsoft's products are not fully functional vis-à-vis the competition, the firm's price points are lower, and its growing base of popularity within MIS is growing, given its arsenal of additional products, including SQL Server, its relational database management system.

- **Whither the database vendors?** While the three horsemen of the RDBMS client server marketOracle, Informix and Sybaseare currently uninvolved in the tools market, we believe that integration between client server apps and database technology will impact the tools market. The database vendors are active users of ASQ tools, and we would not be surprised to see these firms begin to offer such tools to their customers.

For more information about **Visual Intercept**, the fastest-growing name in automated software quality, call:

Raleigh Group International
1-800-364-5467

phone 919.878.3731 tollfree: 800.364.5467 email: info@