



Delivering Systemic Software Quality in the Enterprise

Contents

Beyond the Test Team - Systemic Software Quality

Test Early Test Often and Beyond

Understanding Software Architecture and Dependencies

Static-Analysis - Finding crash causing defects in software applications.

Empowering the test team

- Test Management
- Test Data Management
- Functional Testing
- Load Testing

Application Performance Management

Conclusion

Software Quality is often associated with the testing processes and practices employed by organisations developing and delivery software. The responsibility can often fall entirely on the test team to find defects in the applications.

This whitepaper explores the technology available to organisations to enable them to deploy a more systemic and holistic approach to delivering high quality software and business applications to customers inside or outside the enterprise.

This approach leads to a greater visibility of potential risk, fewer defects, faster time to market and improved customer satisfaction.

Beyond the Test Team - Systemic Software Quality

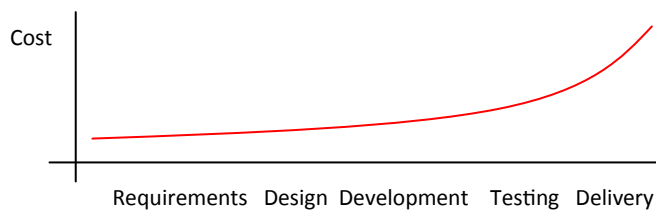
Ask yourself a simple question. Who has the responsibility for Software Quality in your organisation?

- ◆ The Test Team
- ◆ Business users
- ◆ IT Operations
- ◆ Software Architects
- ◆ Developers
- ◆ Human Resources
- ◆ Marketing



The truth is they all, and probably others, have a role to play in the ultimate quality of the applications you deliver. Gaining visibility of defects in the application sooner rather than later and communicating that throughout the application is vital. Costs of finding and fixing software defects are well publicised, and the perceived wisdom is that the later in the process the defect is uncovered the more costly it becomes. So the classic cost curve always approximates to that shown below for traditional development processes. Proponents of modern software development practices, such as Agile, point to the fact that such development practices will flatten the curve because working software is

delivered quickly and it is easier to respond to changing requirements and functionality.



Often though this flexibility can come at the expense of difficulties in automating the testing process. In addition, once software is deployed, the cost of a defect is likely to increase substantially due to the cost of extra support and upgrades let alone the cost of brand reputation and lost opportunity.

Test Early, Test Often

Testing early and often is the best way to maximise the impact of improved software quality in terms of reduced cost, timely delivery and robust applications. This includes static analysis, unit and regression testing, as well as ad-hoc performance and stress testing and application performance monitoring. Building in quality can be argued to start with the architecture, which can be considered as the first point in the software creation process where quality requirements can be addressed. Business considerations should have a major impact in an application or systems architecture. The functionality of the system is often the major driving force in the development process but other factors can have significant impact. For example applications with identical functionality may be re-designed because they are difficult to maintain, port, scale or are too slow.

" Over 80 percent of errors are introduced in the coding/unit testing stage, but well over half of these errors are not found until downstream in the development process."

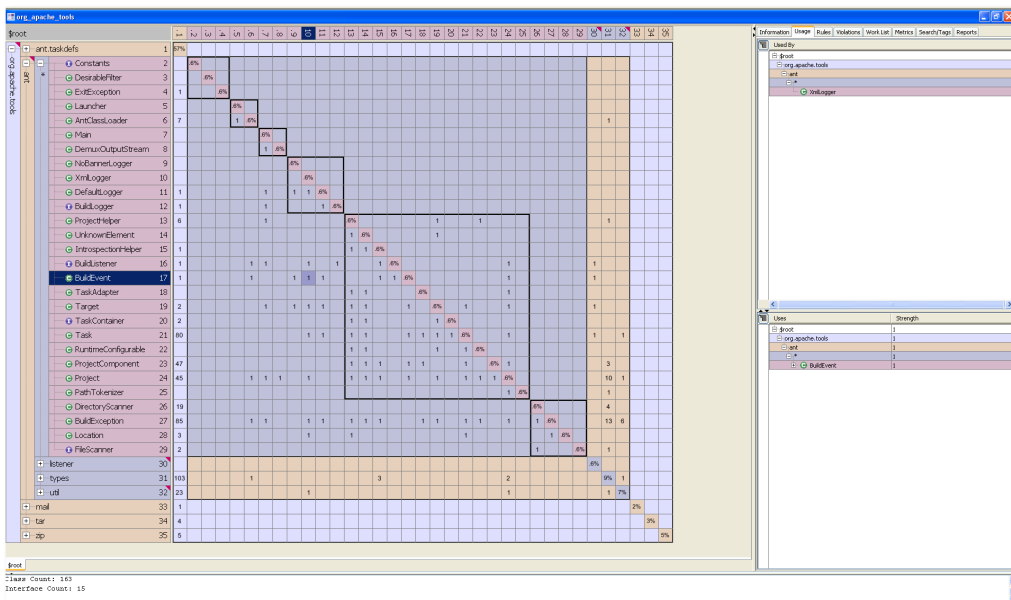
--National Institute of Standards and Technology Study (2002)

Understanding Software Architecture and Dependencies—the first step in building quality software

In any software, there is code that calls other code as a part of its execution process. Code is often organised into modules to help map to a chosen architecture and to make the development process easier. However, cyclical dependencies and dependencies that change frequently can introduce difficulties in finding and fixing defects in the code and will make the challenge of understanding the overall structure of the application more challenging for the developer. With code often amounting to millions of lines, written by many developers, which changes over time, it can be difficult for an individual developer or team manager to maintain a clear view of the relationship of the code to the original design intent. It is always a challenge to have developers perform this sort of analysis because unless there is an immediate bug or functional issue with the code, architectural issues that might affect the maintainability of the code and application over its entire lifecycle are often ignored. This will ultimately result in code that is difficult to maintain and refactor.

Lattix is an application that allows developers to analyse the dependencies in their applications from the initial UML design through the development of the application across multiple application tiers such as a Database, Application Frameworks and Code. Lattix's Dependency Structure Matrix (DSM) based technology allows you to quickly and easily visualise and analyse your architecture in detail, edit the structure to create what-if and should-be architectures, and then create Design Rules to formalise and communicate that architecture to your entire development organisation.

The Lattix System enables the extended development team to see and understand the Dependency Model. This allows all developers to better understand the architecture of the system they are enhancing, and to check their latest changes against Design Rules before they commit those changes into the revision control system. Lattix can also be utilised to provide visibility to managers, software quality assurance staff and product managers who have a vested interest in more than just the software's schedule and planned release dates.



Static Analysis—finding crash-causing defects in software applications

Static analysis techniques infer information about software behaviour based on a static representation of the software. This contrasts with dynamic analysis techniques, which gather information by observing software as it is running. Because code is analysed instead of executed, static analysis does not require test cases, which means it doesn't need to be performed by testers and can be completed by developers earlier in the development cycle.

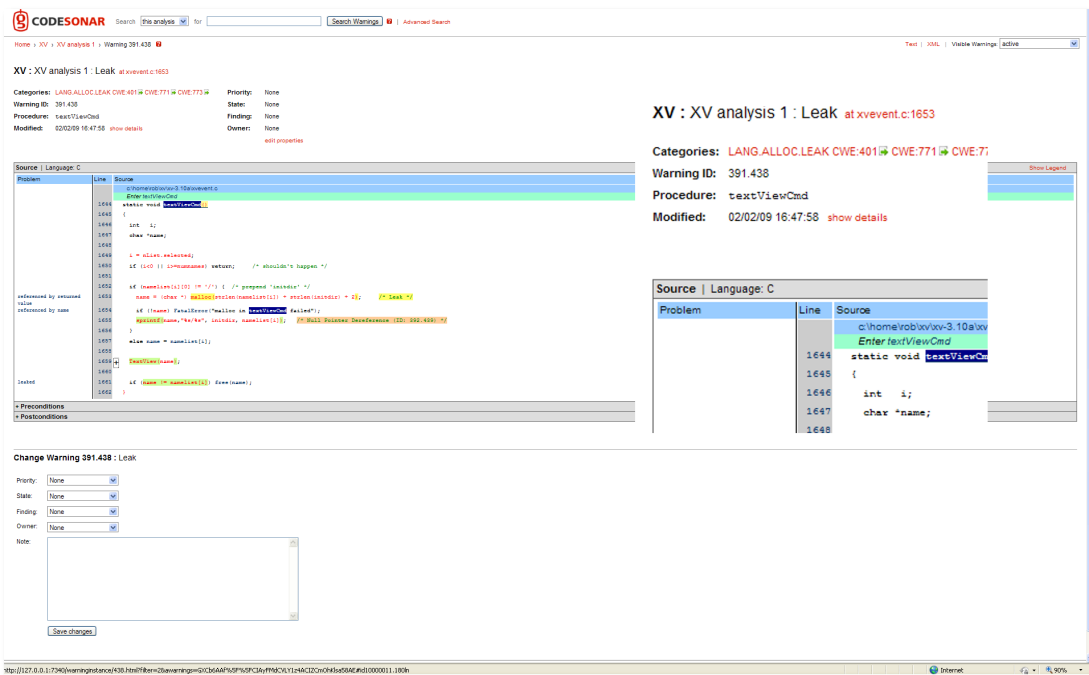
Static analysis can be seen as a two-phase process. The first phase involves extracting semantic information from the software's source code. The second phase involves using this information to discover defects or other properties of interest.

Until recently, static analysis tools were impractical. At one end of the spectrum were tools that could only check the surface structure of the code (e.g. Lint). These are useful for enforcing coding standards, and for some superficial checks, but they are not powerful enough to detect many classes of serious problems. Lint-like tools also generate a large number of false positive warnings making it difficult to review their output. At the other end of the tools spectrum were very sophisticated model checkers. These tools can verify sequencing properties of a system, but may be difficult to apply directly to source code, working instead on an abstract model of the system provided in an esoteric language. Model-checking tools have also historically suffered from scalability problems. Recently, advanced static analysis tools have emerged that overcome these obstacles and make it possible to apply sophisticated analyses to large programs.

Code Sonar is a static analysis tool for C and C++ developers that allows them to find significant hard-to-find defects in applications with a low false positive (reported defects that turn out not to be defects) and a low false negative rate (those real defects not found). CodeSonar compiles your source code without modification or instrumentation, usually as part of the build process, stores the results and delivers a report of the defects via a browser interface. The report displays your source code along with detailed information that shows the precise cause and location of the defect.

"CodeSonar does a better job of finding the more serious problems, which are often buried deep in the code and sometimes hidden by unusual programming constructs that are hard for other static-analysis tools to parse. CodeSonar is not easily fooled."

--GE Aviation



Empowering the test team

The holy grail of any test management and automation initiative is to maximise defect discovery rates and lower the incidence of defects discovered in production systems, whilst minimising the manual effort required in the testing process. Sadly with most functional and UAT testing this can be difficult to automate. Changes in the application can cause the effort required to record and maintain test scripts to incur more cost than continuing a manual testing strategy. In many respects the difficulty in automating the testing process has been made more difficult with the adoption of modern development techniques such as Agile and testing-led development where changes to the application can be both rapid and significant.

Test Management

A coherent planning and management platform is a key success factor in the adoption of any project. Unless the current extent and success of testing can be properly managed and measured it will be challenging to deliver full benefit and maximise the return on investment. Lack of coherent test management can lead to:

- ◆ Inefficient testing because of poor knowledge, communication and a lack of clearly defined process.
- ◆ Increased risk of poor quality applications and higher costs of maintenance and support.
- ◆ Low team morale due to moving deadlines, mixed priorities, vague leadership and a lack of communication between the test team, development and business stakeholders.
- ◆ Hours spent compiling reports that are out of date before they are finished.
- ◆ Poor management visibility of project status and progress which could result in missed deadlines, and rushed testing.

Qualify is a comprehensive Application Quality Management solution that comprises fully configurable graphical dashboards, class leading integrated communication channels and multi format reporting.

Qualify has been designed by QA professionals to fit your business and processes, with comprehensive role-based access and privileges. Qualify will allow you to define what information you want to store and see, and have full control over screen layout to easily map to your requirements. Qualify makes it easy to communicate information with clear graphical dashboards that can be configured to suit your needs. Project information can be shared using email and via reports in a variety of formats including HTML and PDF with full access to historical records for every project to provide a comprehensive and complete audit trail.

Test Data Management

Creating and managing test environments can be time consuming, expensive and may not produce the desired result. Being able to take production data, intelligently subset it based on an automatically derived hierarchy model to retain data integrity, and quickly and reliably build ready-to-go test environments can dramatically reduce the costs and time taken for environment preparation during testing. Provisioning a Test Data Management solution can reduce the amount of storage required for data, which allows you to manage, without increased effort, the number of environments that require maintenance, and decrease the amount of preparation time between each test so that the tests can be run more quickly.

Using production data in a test environment can often violate legislation and good practice, as well as increasing data security risks. Scrambling data can avoid these issues by masking or mixing data to make it untraceable so that valuable production like data can still be used.

TestBench provides a complete environment for managing test environments and test data. Checkpoints and Rollback capabilities allow you roll back to a convenient checkpoint in the case of a mistakes or a transaction that invalidates or corrupts the data that you have carefully created

The integrity of the database is key in most significant business applications, and it is also one of the hardest aspects to check. TestBench provides real in-depth and independent verification of the database showing developers exactly what happened; writes, updates, deletes file by file, record by record, field by field and program by program – whether batch or interactive. It shows this wherever it happened, whether it was to be expected or not and provides the user with the ability to create rules to check integrity and validity as the events happen. As these data rules accumulate with knowledge and time, the database aspects are subject to increasing examination, closing the net on illusive errors and ensuring accuracy where it counts – in the database.

“Our business is based around providing accurate services and up-to-date information to millions of people; we need to be able to spot potential problems before they happen.”

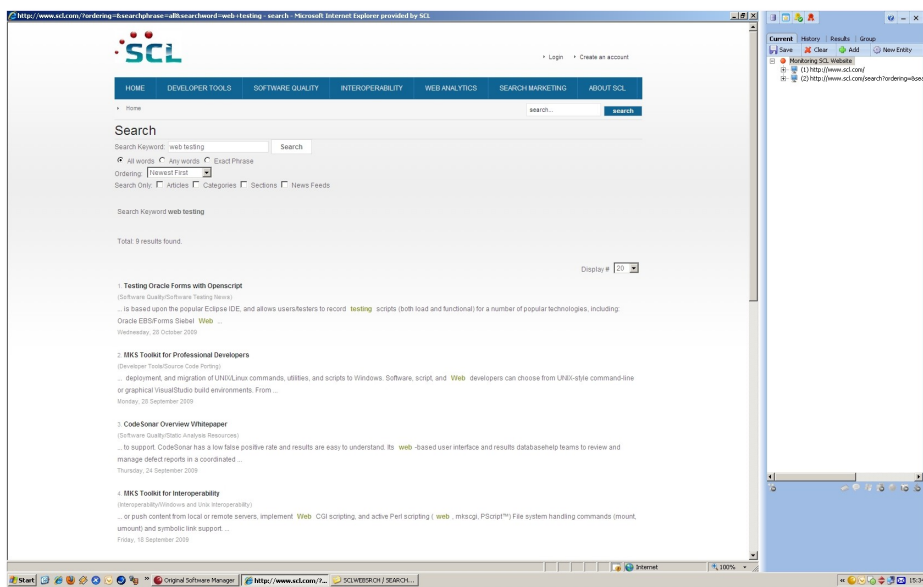
--UCAS

When carrying out regression testing, the key thing is to identify what has changed. TestBench readily supports this requirement with the ability to baseline a set of results that can then be compared to the latest run to find the smallest differences. TestBench will quickly find all the differences so you only the changes you expect are there with no unwanted omissions or additions.

To make regression testing as easy as possible, TestBench enables you to capture a complete set of results (including those with screen and web content when used in conjunction with TestDrive), and to define this data as a ‘Baseline’. Subsequent test runs can then automatically be compared to the baseline set and any differences highlighted, even down to the database. This gives you total peace of mind and in-depth validation of the whole test, you can be assured that any difference has been detected.

Functional Testing and Test Automation

Despite the proliferation of automated solutions, manual testing still accounts for at least 80% of all testing. Automation can only be justified where repeatable consistent tests can be run over a stable environment. When this is not the case, as frequently happens, then testing teams almost always revert back to manual testing. To gain the advantage of automation software needs to allow users to quickly and easily incorporate and maintain the necessary interactions with the application under test.



SCL provides software for functional testing that can be used on GUI, web AJAX, Java and Green Screen/Legacy applications written in a variety of technologies including AJAX, .NET, Oracle, Java, JSP's, Perl, Python, PHP etc., as well as all major development languages.

TestDrive-Assist is a totally new concept in testing that with powerful tracking functions, operates in a non intrusive and natural fashion. Testers can detect and highlight defects fast and efficiently. This means defects can simply be reviewed at a touch of a button, whilst the results can eventually be used to build fully automated tests.

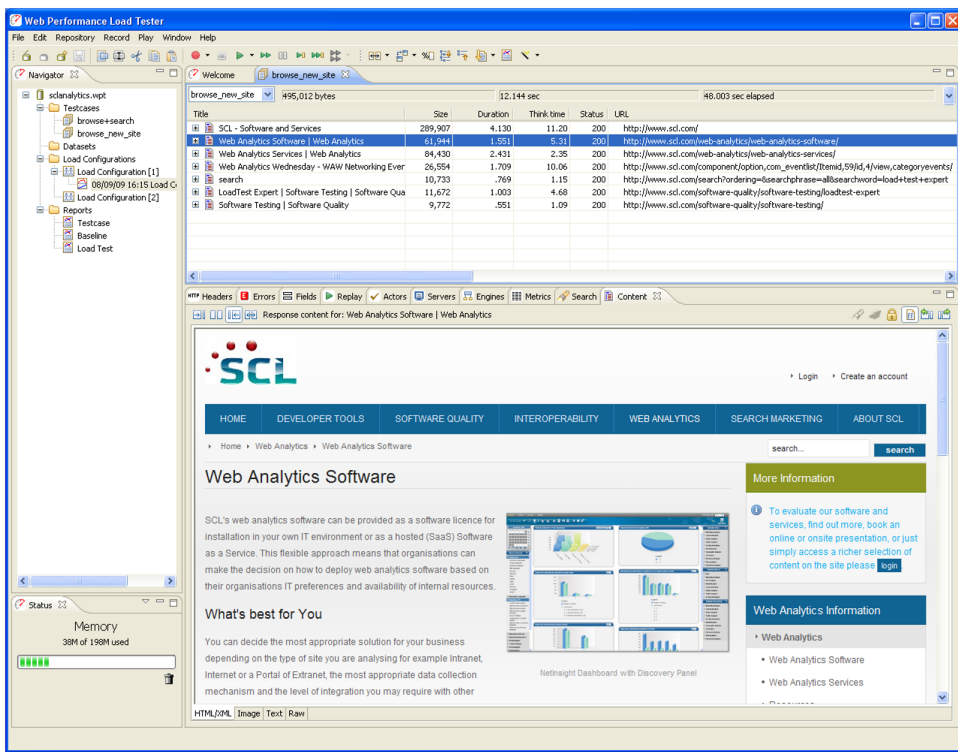
TestDrive-Assist automatically recognises when registered applications are launched. It then softly tracks every user action at both the visual layer and at the database level. For iSeries and Oracle users, all database updates/additions/deletions, can be tracked and recorded. In addition rules can be placed on these updates allowing for greater confidence of data integrity. It is then easy to convert TestDrive-Assist's manual test scripts into fully automated test scripts in its sister product, TestDrive. These scripts then are run by TestDrive unattended, much faster than by manual input, without the user error often introduced during high volumes of repetitive tests, and can be looped hundreds or thousands of times using vast volumes of test input data. The benefits to organisations that have adopted automated functional testing such as provided by TestDrive are clear: much increased test coverage per release, which increases confidence that applications will behave in accordance with the original business requirements and with a much smaller incidence of defects being detected later on in production.

TestDrive is designed to test GUI and browser applications "out-of-the-box". It has in-built technology to deal with a multitude of controls and techniques (such as Lotus Notes, HTML 2.0 etc.), without you having to worry about them. TestDrive's unique technology ensures that users are able to spend less time worrying about technology and test more, to a greater depth than ever before. The result is higher quality applications, delivered to market in a shorter period of time. In essence, the whole application development process becomes more productive.

Load Testing

Making sure an application scales before deployment can be a costly process, but failure to make sure the application performs with an end to end test can be even more expensive in terms of resolutions costs, missed revenue and affected reputation. Load Testing pinpoints bottlenecks in the application infrastructure that could limit performance and cause application slow-downs or failure. Tests are often conducted inside and outside the firewall to ensure that all of the infrastructure is tested from the ISP, network, servers and software. Building load testing into the testing process is vital, but can often be compromised by the time and effort required to configure the tests and the cost of providing a high volume of simulated or 'virtual users'.

Traditional load testing tools have been script-heavy and difficult, time consuming and costly to configure and maintain.



Our load testing solutions provide a fast and cost effective way to validate the performance and scalability of your Web applications and Web Services. Offered as on premise software licences or as a hosted testing service, LoadTEST Expert, our solutions can simulate thousands of users accessing the application or service. The realistic solutions that can be configured to handle the most complex applications and user scenarios. With LoadTEST Expert you also have access to an experienced consultant throughout the test window.

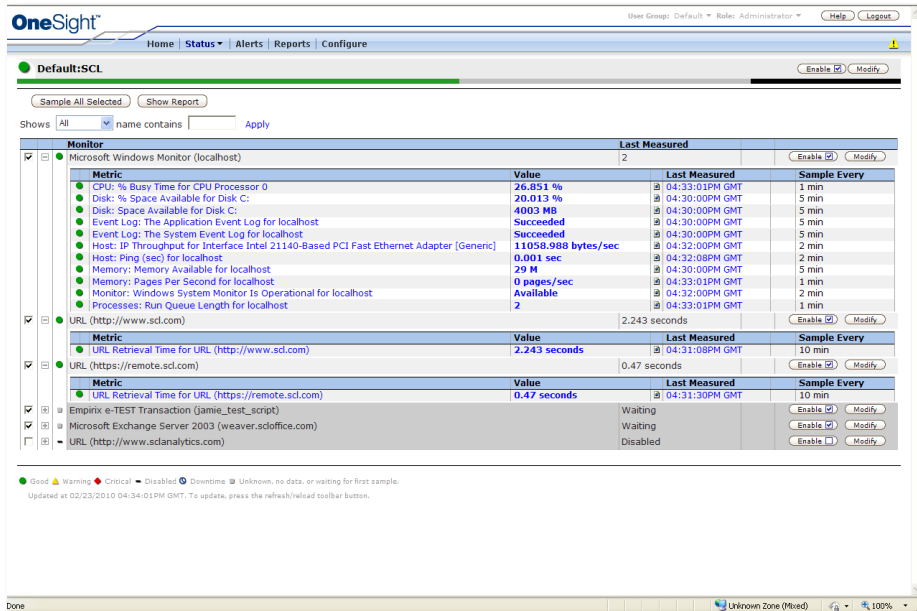
Application Performance Management

So you now have the application, it's in production and everyone is happy. There is an old joke that goes something like 'How many software engineers does it take to fix an application.... None, it's a hardware problem. So how many Hardware Engineers does it take to fix the application..... None, its a network problem.... And so on.....'

The truth is that most users of a web application don't care if the CPU is at 90% capacity or the operations staff are running around to fix issues as long as their use of the application is not compromised. Most if not all providers of web based applications have some form of monitoring in place to ensure the infrastructure supporting the application is functioning correctly. However the application performance for the end user can easily be overlooked. There are two broad classes of application performance management that address this issue by checking the availability of the URL and the compliance of key transactions and business processes on the web site. This can be achieved by using synthetic transactions, which have the advantage of repeatability, to check using a 'virtual' user the compliance of otherwise of key business transactions. In addition to this, real users' transactions and activity on the site can be tracked to identify individual issues and classes of defect not always identified with synthetic transactions. Sadly users don't always do what you expect them to do !

Synthetic Transactions

Onesight is an application performance management solution that allows synthetic transactions to be completed for web applications. These transactions can be correlated with the underlying application hardware and software so that issues can be identified and escalated more quickly. This allows a preventative approach to application management rather than disaster recovery.



Real User Experience Monitoring

Our solutions in this area allow enterprises to gain insight into the real end user experience of their applications. Gaining an understanding of the experience of all users of the web application allows business, operations and IT stakeholders to develop a shared understanding of their applications user experience by integrating performance and usage analysis into a single solution. Using sophisticated Network Protocol Analysis for data collection, requiring zero changes or instrumentation of the application, enterprises can now have a real time view of the end user experience.

Conclusion

Adopting a systemic approach to software quality in the enterprise will allow applications and services to be delivered to end users and internal customers more quickly, with fewer defects and at lower overall cost. Testing early, often and throughout the development lifecycle into operations ensures that all stakeholders will have greater visibility, ownership and interest in the quality of the end result.

For More Information:



SCL
Jubilee House
Jubilee Walk
Crawley
West Sussex
UK
RH10 1LQ

Web: www.scl.com
Email: info@scl.com
Tel: +44 1293 403636

All trademarks are copyright of their respective owners.