“Enosis Solutions’ Testing and Quality Assurance team ensures in time knowledge acquisition, cross-platform validation, adequate test coverage, and early defect detection”

SUCCESS STORY FOR QA FOR ENGINEERING SIMULATION

BACKGROUND

The client develops software to solve the most challenging engineering problems allowing engineers to refine and validate designs at a stage where the cost of making changes is minimal. The software suite adheres to plug-in architecture where the applications are developed and deployed as plug-ins to a software framework internally referred as ‘HOST’. The framework follows Document-View architecture and provides highly performing development tools for the application developers facilitating modeling, interoperability and visualization in seven different platforms namely, Windows (both 32 and 64 bit), Linux (both 32 and 64 bit), IBM, Sun and HP. Client was faced with the challenge to continuously test the HOST framework on various platforms. This created problems with time consumption, costs and reliability. Due to its proven track record of ensuring product quality by providing end-to-end QA and testing solution the client chose Enosis solutions, which came out with a plan that covered each stage of the product lifecycle and satisfied the client’s requirements.

THE CLIENT

A world leader and pioneer of developing engineering simulation software used to predict how product designs will operate and how manufacturing processes will behave in real-world environments. They are developing software to solve the most challenging engineering problems allowing engineers to refine and validate designs at a stage where the cost of making changes is minimal. Their simulation solutions are deployed across automotive, aerospace, defense, electronics, marine and shipbuilding industries serving engineers and researchers in corporations that include Airbus Industries, Air Force Research Lab, Bell Helicopter, Boeing, Rolls-Royce, John Deere, LG Electronics, Lockheed Martin, NASA, Toshiba Corporation, US Navy, GE, Hitachi, Toyota, Honda, BMW and Ford.

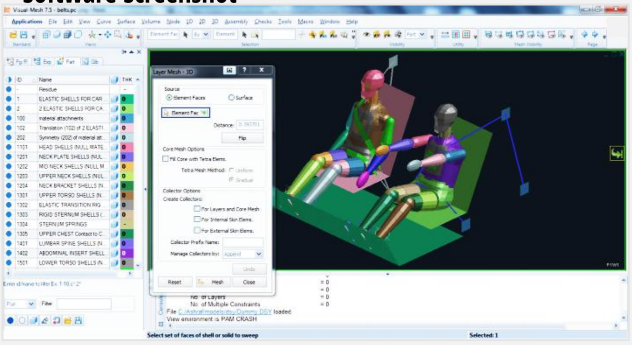
BUSINESS SCENARIO

HOST enables the application developers to define their own application from a very large set of packaged tools characterized by their friendly usability and interoperability. Client demand was to continuously test the application on various platforms. This required significant investment of time on the part of the Quality Assurance (QA) and testing team. Such frequent testing also had implications on the cost and quality of testing.

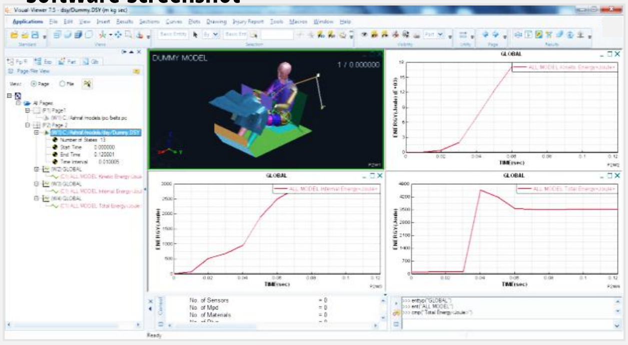
The business needs, therefore, were:

- To develop a reusable and user-friendly test framework
- To integrate the testing activities and improve the scheduling capability in various environments

Software Screenshot



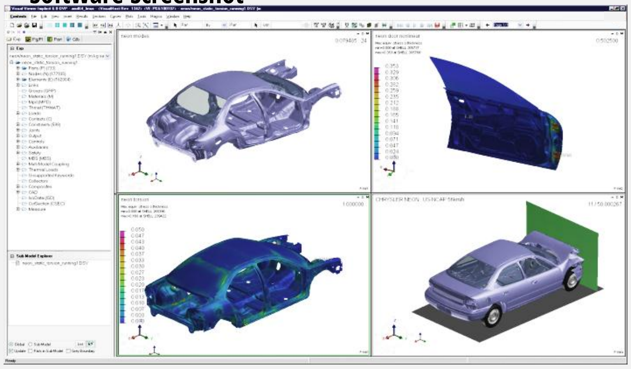
Software Screenshot



CHALLANGES

- Understanding the software requirements, analyzing the UCD (Use Case Documents) thoroughly, extracting technical implementation and business logic details by conversing with the Project Manager and client's project team.
- Understanding HOST requires domain knowledge and process workflow understanding since it is the framework for an engineering simulation application suite that includes crashworthiness testing, stamping, casting, electromagnetism, comfort, aerodynamics and vibration simulation software.
- The application runs on different platforms e.g. Windows 32 & 64 bit, Linux 32 & 64 bit and HP OS. Therefore for every release we have to set up different test environments for the several operating systems where the regression suite and the available test cases are executed. Before every release, we have to make sure that the developed solution performs seamlessly across various platforms.
- The heterogeneous system necessitated multiple checkpoints, and it was not feasible to simultaneously make available all systems for integration testing.
- Conventional testing approaches were not a good fit as there was limited user interface exposure to the software framework that needed to be tested. For instance, middleware / Application Programming Interfaces (API) were needed to be tested by developing plug-in applications.
- By the time we started our quality assurance endeavors the HOST application was already under development for more than twelve years. The current application is extremely complex in terms of features and functionalities. A centralized repository of test cases was not available. A large number of quality assurance and testing tasks have to be executed within a limited timeframe while consistently maintaining application quality over the bi-quarterly releases.
- The HOST application is frequently updated with new and enhanced features and functionalities. For the added and modified features, the application not only has to be tested across different platforms e.g. Windows 32 & 64 bit and Linux; but also for different versions of the application for compatibility and languages that are supported. We had to ensure that the application modifications did not affect the application's stability and the fixes incorporated are compatible across different platforms and with the supported languages.
- The application achieved and exceeded every performance target specified in the requirements.
- It was user-friendly and accurately determined user preferences based on location.

Software Screenshot



- This application allowed consumers to receive the core features of the Web Application while being mobile.
- Since information from the application mostly uses the storage available in the main server it does not take up space in the device.
- The HOST framework follows plug-in architecture and consists of extensive features. In order to test these features, we had to develop plug-in applications that communicate with HOST. The framework APIs (Application Programming Interface) are invoked by writing test codes according to the context.
- Developing the plug-in applications and performing tests afterwards required significant effort from the QA team.
- Analyzing and identifying the necessary steps that are to be executed to replicate the problems reported by the users from client's side while understanding the cause of such problems and recommending solutions.
- Verifications of memory usage by the subsequent versions of the application to ensure that it does not take up memory at runtime without resource de-allocation afterwards and to avoid memory leakage, which makes the system lethargic.
- The necessity of running HOST on multiple platforms limits the opportunities of using standard platform specific controls since they won't allow modification of the control source code. Therefore the HOST development team, have to develop custom UI controls which need to be tested and compared with the dynamics of the standard controls behaviors.
- Modification of a single function in the software may affect the behavior of other features. After major enhancements or modifications, a number of test cases have to be executed across the pertinent functions to detect anomalies. Such frequent changes demanded a centralized repository of test cases suite to capture the existing behavior and the behavior after incorporating the modifications.
- No uniform methodology and standards were practiced for development and quality assurance due to multi-vendor, multi-package scenario.

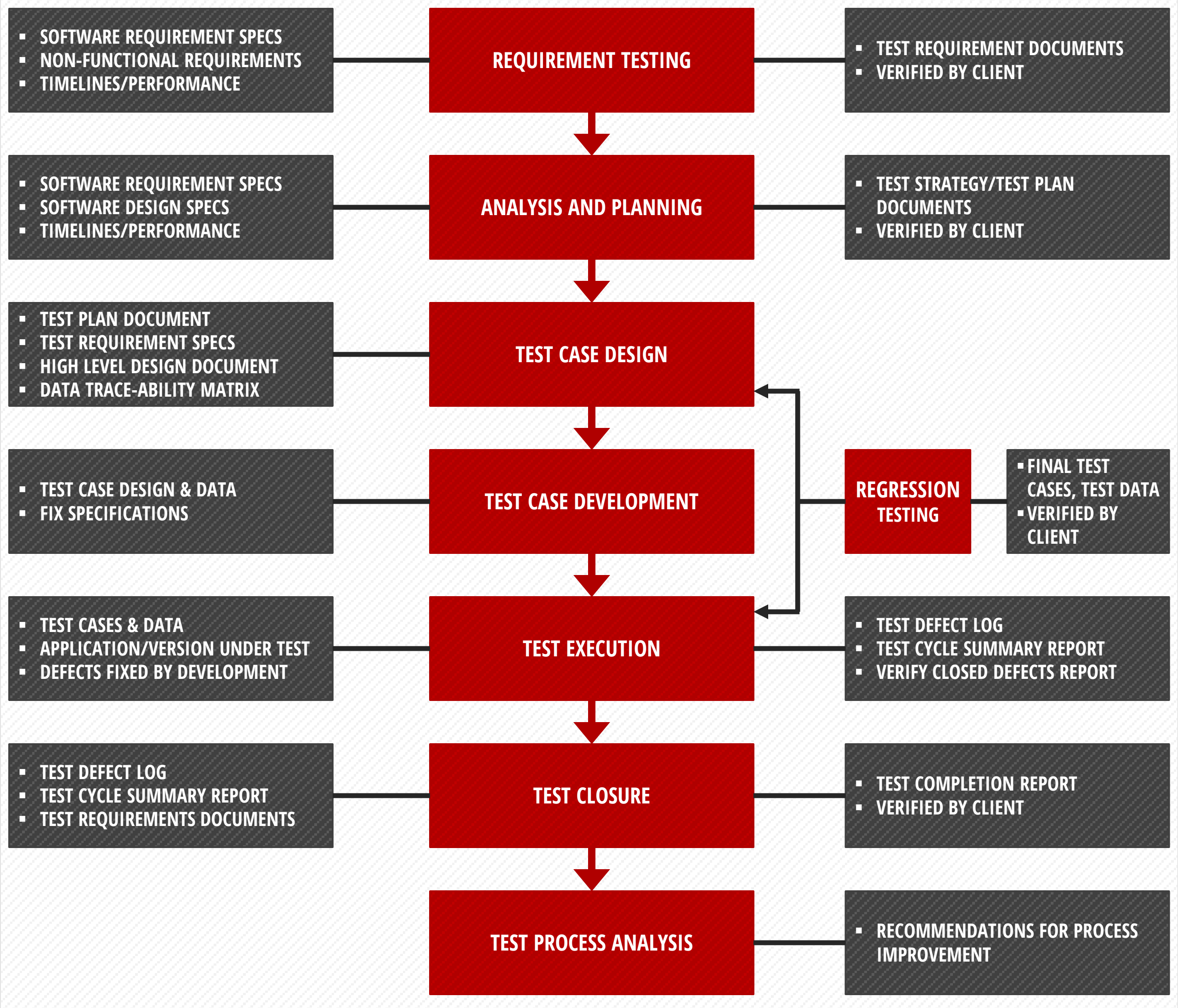
ENOSIS SOLUTIONS' APPROACH

Lack of in house verification and validation expertise, forced the client to look for a strategic partner who could plan and test the full functionality of the product across various platforms and conduct performance evaluation of the application as well.
Enosis came out with a plan that covered each stage of the product lifecycle and satisfied the client's requirements. The client selected Enosis, for its proven track record of ensuring product quality by providing end-to-end QA and testing solution.

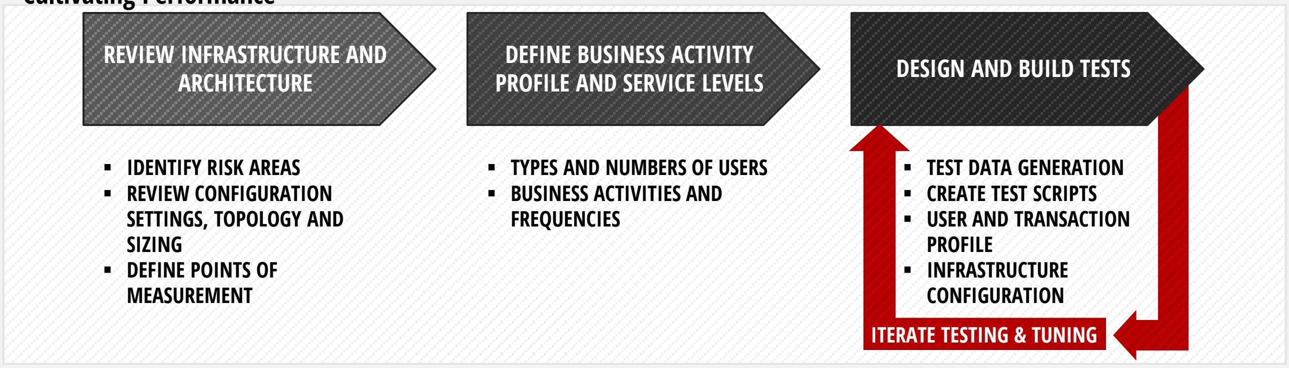
Adopting best practices led to better-managed and well-executed delivery:

- Enosis provided complete pre-release testing of the HOST application including functional testing and performance engineering
- Deployed a scalable and robust testing mechanism within short period
- Configured a critical mass of test cases for around 100 workflows and executed them in time for the roll out
- Took an 'application level' testing approach that ensured that system components such as middleware were tested and interfacing systems were simulated to provide adequate coverage
- Configured a critical mass of test cases for around 100 workflows and executed them in time for the roll out
- Took an 'application level' testing approach that ensured that system components such as middleware were tested and interfacing systems were simulated to provide adequate coverage
- Around 1000 bugs, out of which 300 were major bugs, across 3 major releases, were posted. The product was tested on all OS Versions/flavors. Functional, regression, compatibility, and performance testing were done to discover the bugs

Product Testing Lifecycle



Cultivating Performance


REVIEW INFRASTRUCTURE AND ARCHITECTURE
DEFINE BUSINESS ACTIVITY PROFILE AND SERVICE LEVELS
DESIGN AND BUILD TESTS
IDENTIFY RISK AREAS
REVIEW CONFIGURATION SETTINGS, TOPOLOGY AND SIZING
DEFINE POINTS OF MEASUREMENT
TYPES AND NUMBERS OF USERS
BUSINESS ACTIVITIES AND FREQUENCIES
TEST DATA GENERATION
CREATE TEST SCRIPTS
USER AND TRANSACTION PROFILE
INFRASTRUCTURE CONFIGURATION
ITERATE TESTING & TUNING

ENOSIS' APPROACH (CONT.)

- A series of Baseline and Performance runs were conducted to find and remove the root cause of performance degradation. Recommendations for improving the scalability and stability of the application were provided by Enosis' QA team
- Hardware benchmarking tests were performed to find the scalability of the application with respect to availability of CPU and Memory
- Configured and executed complex scenarios involving multiple steps and interventions with ease
- Ensured intensive communication between the client and testing team for better understanding of the product. Various sessions to outline strategies, analyze and finalize the testing tools also took place
- Regression tests were performed to validate that the existing business processes would not be adversely impacted. The focus of this validation program was on:

  - **Flexible and dynamic planning:** to minimize the cascading effect of upstream timeline delays
  - **Business scenario driven test scripts:** to replicate the end users' usage in production
  - **Scenario review workshops, smoke test and script pass cycles:** for early detection of defects
  - **Optimal sequencing and scheduling:** to minimize test data setup needs and overheads
  - **Reverse engineering of legacy applications:** to derive the requirements for regression testing
  - **High reusability of artifacts:** for future releases and implementations

- By utilization of in-house tools, the QA team monitored memory status for running the HOST application after substantial enhancements and modifications have been incorporated
- Validation of all the messages, labels and customizations were performed to ensure adherence and compatibility with the client's region, language and culture

The Enosis Verification and Validation team helped the client to achieve several business benefits through this engagement.

- **Increased productivity of development team:** by engaging the QA team to start testing early in the development phase and consequently identify bugs earlier in the product development lifecycle
- **Shrinking future test cycles:** by providing a repeatable test strategy and several reusable artifacts
- **Pre-empting discovery of several business design gaps:** through scenario review workshops during the test-planning phase
- **Providing verification checklists:** to the infrastructure for verification of test environment and initial data setup
- **Increased confidence in the quality of production releases:** by the ability to run tests more often and in short time frame. Increased quality by shrinking regression test cycles of modules by discarding tests that were not directly affected by enhancements.
- **Developing an integrated code** migration methodology: which could be used across all the validation environments and production
- **Optimized resource utilization:** Users were able to spend more time on testing newly implemented modules due to the sustained stability of older modules

TOOLS AND TECHNOLOGIES

Programming Language: C ++, MFC, Python, Perl
Shell Platform: Win XP 32 & 64, Vista, Linux 32 & 64
Tools: Valgrind, In house tools for memory, performance, UI & custom controls testing