

**AUTODRIVE TECHNOLOGY**  
**WHITE PAPER**

**By Anthony A. Lombardo**

## **INTRODUCTION:**

Today's web-based software solutions have development lifecycles that are becoming shorter daily. One of the many challenges facing software companies is to ensure only the most reliable, efficient and user savvy products reach today's cutthroat marketplace. AutoDrive Technology is focused on easing the burden involved with demanding software quality assurance standards, while keeping you on schedule to meet critical deadlines. AutoDrive technology has accepted and capitalized upon this exponentially growing challenge. The solution is developing robust, flexible, integrated automated testing processes that will endure the entire software development lifecycle. Automated testing experts implement this solution by unifying software developers and test engineers, via state of the art technology and an objective viewpoint. Utilizing creative expertise, AutoDrive can provide the time and skills necessary to test all your client/server or web applications. AutoDrive's methodology, testing process, template model, development tools and script design are setting the global standard for automated testing technology.

## **AUTOMATED TESTING METHODOLOGY:**

AutoDrive's automation testing methodology is one that is made to withstand and endure the entire software development lifecycle. This methodology eliminates the possibility of test procedure entropy as applications evolve over time. More often than not, automated test scripts become irrelevant within only an iteration of the software development lifecycle because script maintenance is exhausting and expensive. As a result, the AutoDrive technology is focused on three major ergonomic scripting factors: reusability, maintainability and flexibility. As a result, each one of these factors actually expound upon the next.

The most challenging aspect of automated testing for companies in today's accelerated software industry is test script reusability between consecutive builds of an application. The scripts built today have to be able to pass execution tomorrow with minimal rework involved. Industry wide, companies are attempting to implement an automated testing process only to be left with an unpleasant and expensive experience due to the exhausting man-hours expended rewriting or fixing degenerate automated test scripts. AutoDrive is designed to eliminate this discord between automated testing and shipping a dependable product with important deadlines.

The test scripts produced utilizing AutoDrive's technology have many advantages as compared to the normal "Record and Playback" methodology of automated testing. The scripts uncluttered design utilizes intelligent stored procedures that can handle any unforeseen application change no matter where it has been implemented. They are triggered to capture imperative data when an error or unexpected window is present. These custom stored procedures also compose explicit notations in log summary reports explaining where these changes occurred and if a fatal error has been uncovered. This allows test engineers, software developers and members of management to quickly see where trouble spots are at the completion of an automated regression test.

This leads to the next advantage of AutoDrive's technology, which expounds upon the aspect of script reusability. It is essential that any automated script development process foresee the absolute fact that many test scripts will have to be revisited at some point in time due to application and/or environmental changes. The AutoDrive automation process is built upon this reality and carefully designs each script in such a way that any maintenance efforts are reduced to a bare minimum. The scripts are written by taking advantage of the fact that test engineers can quickly and easily maintain an automated test script that has all the unnecessary and confusing script text eliminated from it. Furthermore, scripts built using AutoDrive's object-oriented scripting syntax enable test engineer's to easily navigate and trouble-shoot scripts. Information such as field types, object names and ambiguous data values are the catalysts for every script. The AutoDrive process only displays the most pertinent information in a formal and structured way that enables any test resource to pinpoint the exact actions of these revolutionary designed scripts. The automated script output, as of a result of this process, can therefore be easily read and understood by any test engineer due to its simple design, readability and unequivocal structure.

Industry wide, companies have automation tools and test scripts gathering dust because of the lack on knowledge and vision on how this type of testing effort can save millions of dollars in time and expenses. Companies such as these are developing automated test script scenarios by simply recording and playing back user actions that are captured by automated testing tools. Unfortunately, with the complexity of today's applications automated testing is not this simple. Automated test scripts must be designed to be just as flexible and dynamic as the applications they are testing. No matter how robust a test script may seem it is worthless if it is reliant upon hard coded values that may or may not be present at run time. Test Scripts must also be modular enough to be run independently of other test scripts, thus eliminating the possibility of script failure. Script modularity is easily achievable due to the utilization the powerful stored sub procedures and functions that perform all the work. The only values that are present in the scripts are those that direct the stored procedures and/or functions.

Test scripts designed with AutoDrive's flexible scripting approach take advantage of the information present when the test script is executed. First, the test scripts draw upon large amounts of ambiguous data from computer generated data pools that are populated with thousands of random data values of every type and size. Next, the test scripts input this random, untainted data into the application to produce true non-skewed regression test scenarios. Then, if there are any calculated or mathematical computations to be done, the scripts are intelligent enough to extract the appropriate data, do arithmetic and output dynamic results during script execution. No prefabricated numbers or expected results are present because everything is being tested during live execution of the scripts with entirely different data each time. This non-partial and dynamic standard script development methodology is embedded deep in the core architecture of the AutoDrive Technology.

## **AUTOMATED TESTING PROCESS:**

AutoDrive's automation process is built on over a decade of proven software testing experience in a diverse application development environment. Every element of this process is essential to designing a reliable and custom software automation process because each aspect cyclically expounds and builds upon another. AutoDrive's testing process is highly unique in that it is focused on minimizing the manual testing effort and maximizing the utilization of automated testing processes. AutoDrive is designed with the belief that 99% of all testing can be automated with the exception of tangible by-products resulting from all tests, such as printed reports. True regression testing is accomplished by eliminating human intervention and error while executing verified test scenarios. That is what makes AutoDrive so imperative in today's software testing arena.

AutoDrive's technological process is focused on true user emulation test standards. AutoDrive automated test scripts are designed to realistically emulate every possible valid or invalid user action associated with a software application. Thus, testing all valid entries and possible error messages. Intricately developed manual test case scenarios are the backbone to true emulation testing because they are developed with every possible iteration through an application.

The first step in manual test case development is a technical specification review. The technical specifications are scanned by test engineers for discrepancies and possible enhancements that are significant to the development of manual test cases. Data that is imperative to thoroughly testing an application is included in the final technical specification. Once the review and enhancement process of the technical specifications has been completed the test engineers begin the development process of front-end, back-end and business logic manual test cases and their respective test scenarios. Front-end manual test cases test the user interface functionality of an application. Back-end manual test cases test the referential integrity and cardinality of data committed to the database. Finally, business logic manual test cases test the applications adherence to all business rules outlined in the technical specification. Manual test cases are composed of multiple test scenarios, which are logical and illogical circumstances and paths that perform an action in application. The logic that AutoDrive technology is built upon not only encapsulates automated functional testing but also the following test scenarios.

- Read User Interface
- Create Record
- Update Record
- Delete Record
- Miscellaneous
  - Mandatory Fields
  - Field Lengths
  - Hot Keys
  - Field Formatting
  - Data Type
  - Screen Navigation
  - Message Box Display
  - Data Type
  - Encryption Data
  - Boundary testing

Test scenarios are a quick synopsis of the functionality under test. By taking into consideration every possible iteration through a transaction of the application, test scenarios can be quickly developed. Rapidly developing manual test cases by only including test scenarios, as opposed to outlining all data values and user actions, is another aspect that sets AutoDrive technology apart from the rest of the industry. Once the test scenarios are in verified the automation process is set to begin.

Test scenarios are then transposed into automated test scripts, while constantly ensuring there is a one-to-one relationship between the two. This relationship is absolutely necessary to guarantee direct trace-ability. This uninterrupted relationship between manual test scenarios and automated test scripts is significant due the frequent modifications characteristic of any software development lifecycle. When application functionality has been altered in any way, the modular manual test cases can quickly be updated, thus triggering minor maintenance to automated test scripts. In such a case, AutoDrive's undemanding readability fosters the ability to rapidly modify scripts and eliminate the arduous practice of script maintenance. AutoDrive enables test engineers to translate manual test cases into automated test scripts by utilizing a few dynamic and potent functions for any user action desired.

AutoDrive's technology has streamlined the automated test development process by carefully designing precise script template architecture. AutoDrive's architecture was meticulously designed to eliminate script reliance upon screen coordinates, images or other characteristics that will vary between each build, platform and environment. This technology specifically relies on object names, which are the most stable part of any application. AutoDrive has taken advantage of the fact that object-naming conventions should be consistent throughout the lifetime of an application. Object names are rarely modified or changed, which make these the perfect variables to utilized in automated test scripts. AutoDrive technology takes advantage of object name driven scripting, thus eliminating long lines of incoherent, unnecessary code present in most automated test scripts. An advantage of the AutoDrive development philosophy is that it eliminates the frequent predicament of script's becoming unusable because a field was moved or a button was added. This script architecture offers a consistent standard and recognizable placeholder that is present in all test scripts so the ability to make any script changes becomes an uncomplicated task. Where as in the past, making script enhancements meant revamping the entire script repository. Script entropy is no longer a hindrance by developing automated test scripts that are completely modular and independent. AutoDrive's simplified design allows a test engineer to pinpoint exactly where a change occurred.

The dynamic retrieval of object names and other data at run-time makes AutoDrive's automated testing process independent of any data captured when the automated script was recorded last. Unfortunately, a vast majority of test engineering groups record and play back user actions using expensive automated testing tools and proclaim an effective automated testing process. This method captures data that is present then and only then, which certainly is a problem due to software evolution. Scripts developed in this manner inevitably fail do to the inherit restrictions automatically captured in user-recorded

automated test scripts. AutoDrive's architecture was carefully designed to take advantage of the inert logic that expensive automated testing tools hold within their scripting language. AutoDrive technology eliminates the dependency of recorded; automatically generated test script data and focuses on capturing real real-time data at the point of playback and automatically changing and executing a test scenario. For example, to populate an entire form, via an automated test script, the only vital information that a test engineer needs is the applications field names. Due to the fact that AutoDrive technology is truly driven by object names and object properties, a test engineer can virtually develop test scripts without seeing a prototype or the finished application. This particular scripting process of building automated scripts along side software developers actually bridges the gap between these groups and streamlines the software development lifecycle. All the essential information needed to automate a test scenario is present in the technical specifications for an application.

Script development using AutoDrive technology introduces a need for ambiguous data entry into many fields. Entering data into a form with the AutoDrive technology, takes advantage of a feature called Datapools, which are randomly generated tables of data.

Address	City	Company Name	Middle Name	First Name
1340 Millersport Hwy	Bala-Cynwyd	California-Nevada M	Karen	Karen
5300 Kings Hwy	Reno	Photocopy Parts Disl	Koganti	Koganti
260 5th St	Wild Rose	Hyatt Regency Los /	Clarence	Clarence
1633 Broadway	Alliance	D'Addario J & Co	Pam	Pam
4600 Air Way	Columbus	Nippon Express USA	Izzy	Izzy
3845 N Blackstone /	Cohoes	McCutchen Doyle Br	Gene	Gene
5757 Wilshire Blvd	Alsip	Queens County	Linsey	Linsey
700 Larkspur Landin	York	Security Pacific Natic	Napoleon	Napoleon
5635 Peck Rd	Santa Rosa	Provident Savings B.	Lester	Lester
73 Crescent Ave	Salt Lake City	Sika Corp	Norman	Norman
650 Davis St	Anaheim	Saint John's Universi	Mike	Mike
3951 Merrick Rd	Cleveland	Melville Knitwear Co	Gerriet	Gerriet
1910 Diamond St	O'Fallon	Empress Hotel	Toni	Toni
847 Sansome St 4th	Venice	Symons Corp	Rex	Rex
141 W Taft Ave	Cleveland	Channel Systems Int	Linda	Linda
331 Fairchild Dr	Hayward	Levi Case Co Inc	Emanuele	Emanuele

A datapool is essentially a test dataset. It supplies data values to the variables in a script during script playback. If datapools are not drawn upon during script playback, the same values (the values that were coded when you created the script) are sent to the server each time the script is executed. For example, suppose you compose a script with a hard-coded value of 12874 to a database server. If you playback this script 1000 times, value 12874 is sent to the server 1000 times, hence the script is not truly emulating a user's actions. By using a datapool, each iteration of the script will send a completely different value to the server. Datapool values can be accessed uniquely to ensure no value gets called twice. They can also be accessed sequentially if a specific order is necessary to the automated test script. Finally, a datapool can be accessed in random order to ensure that a completely arbitrary value is used. Another advantage of datapools is that they are

100% customizable to your data needs. Datapool values have the capability to be auto-generated, user defined or imported from a comma-delimited file. This is just one of the characteristics that make AutoDrive technology a true unbiased and balanced testing methodology.

AutoDrive’s technology is particularly focused on minimizing the duration of automated test script development by easing the conversion process of translating manual test cases to automated test scripts. AutoDrive’s architecture has been structured around this concept and has revolutionized the test script development lifecycle. AutoDrive virtually enables test engineers to design automated test script directly from their associated technical specifications. This innovative development process begins with well-designed technical specifications that foster the few requirements AutoDrive depends upon. It is necessary for the technical specifications to display various screen prototypes, which assist test engineers to envision the final product of the application. This requirement assures that test engineers and software developers work concurrently with the same model in mind. Next, the technical specifications must include the basic framework requirements of proposed field names, object properties and object naming conventions. Due to the fact that AutoDrive relies on the most stable part of any application, which is object names, these entities must be definitively outlined. To ensure that these few requirements are met within the technical specification design, test engineers should be included in every technical documentation review. With these two undemanding requirements met, test engineers can swiftly and effectively design automated test scripts without ever executing the application under test. Thus, enabling automated regression testing to be performed during the software development phase, rather than after development is complete. AutoDrive objective is to reduce risk by testing early in the development cycle and to test iteratively, with every build. This approach enables defects to be removed as the features are being implemented. When automated testing is performed at a late stage in the process testing is rushed and prone to error. AutoDrive’s resourceful new test script development process has been successfully implemented and tested on web, client-server and fat-client software applications.

**AUTOMATED SCRIPT TEMPLATES:**

The result: several state-of-the-art automated script templates that can powerfully emulate any test scenario. To fully appreciate the strength of AutoDrive’s automated test script templates, the manual test case design must be understood. Manual test cases dictate the direction of any front-end, business logic and back-end test. Manual test cases are composed of test scenarios that give detail to exactly what is to be performed during a regression test.

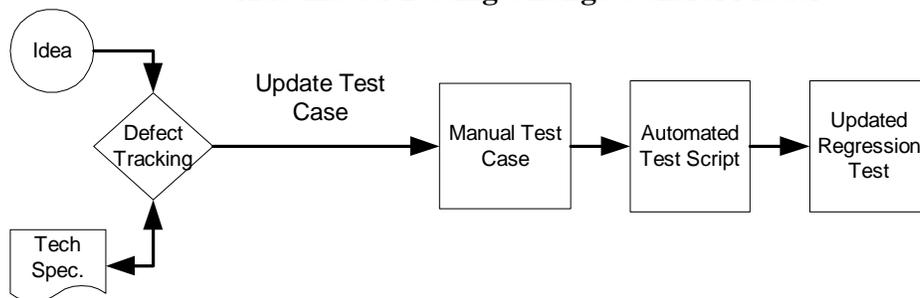
<b>Test Scenarios</b>	<b>Description</b>	<b>Pass/Fail /VI #</b>
FE-TC001-S1	Update of Inventory Item where Item Type = ‘Item’, using the Save button	
FE-TC001-S2	Update of Inventory Item where Item Type <> ‘Item’, using the >> button	

Manual test cases can be further described as a conglomeration of test requirements definitively outlined by the technical specifications and the core functionality of every

feature. The manual test cases ensure that every possible iteration through the application is planned, outlined and executed. For this reason, AutoDrive's automated script templates mirror the testing scenarios summarized in their associated manual test cases. There is a one-to-one relationship between manual test cases and the automated test scripts. Referential integrity is of top importance to all AutoDrive's automated test cases.

The data transition from technical specifications, to manual test cases and then on to the automated test scripts ensure a traceable, conducive testing path. This is especially useful to automated script developers because no longer do the test engineer have to be an expert or receive formal training on the flow of an application. The connectivity of all manual and automated test cases allows for a modular and streamlined maintenance process. For example, when an idea/enhancement is discovered it is documented in the form of an incident. The technical specification is then modified to include the idea/enhancement. Next, the associated manual test case is updated, as per the recent technical specification changes. Finally, the automated scripts are modified and an updated regression test is executed. This entire model relies upon the one-to-one relationship between the manual test cases and automated tests scripts.

### Automated Testing Change Control Model



The core automated script templates test distributed functionality for the following areas:

- Create Record
- Update Record
- Delete Record
- Mandatory Fields
- Field Formatting
- Form Navigation
- General Object Properties
- General Form Properties

There are multiple scenarios encapsulated within each of these templates to ensure comprehensive regression testing is performed. The AutoDrive automation templates have been designed to be universal across the entire application under test. For example, if there are two manual test cases for creating an employee with one test case utilizing the 'Save' button and the other utilizing the 'Next' button; AutoDrive templates are so interchangeable that the only thing that needs to be altered is the script name and the button that is desired to push. Furthermore, AutoDrive's inherent use of datapool technology in all test scripts assures ambiguous data is populating the form and database

when the script is executed. AutoDrive's templates have an adaptable and interchangeable structure due to the creativity and manipulation of known programming standards. The use of nested arrays and modified conditional procedures, such as loops, coupled with database connectivity capabilities make AutoDrive technology distinctive. AutoDrive technology has capitalized on the programmatic foundation in which automated testing tools depend on to auto generate script output by simply recording user actions.

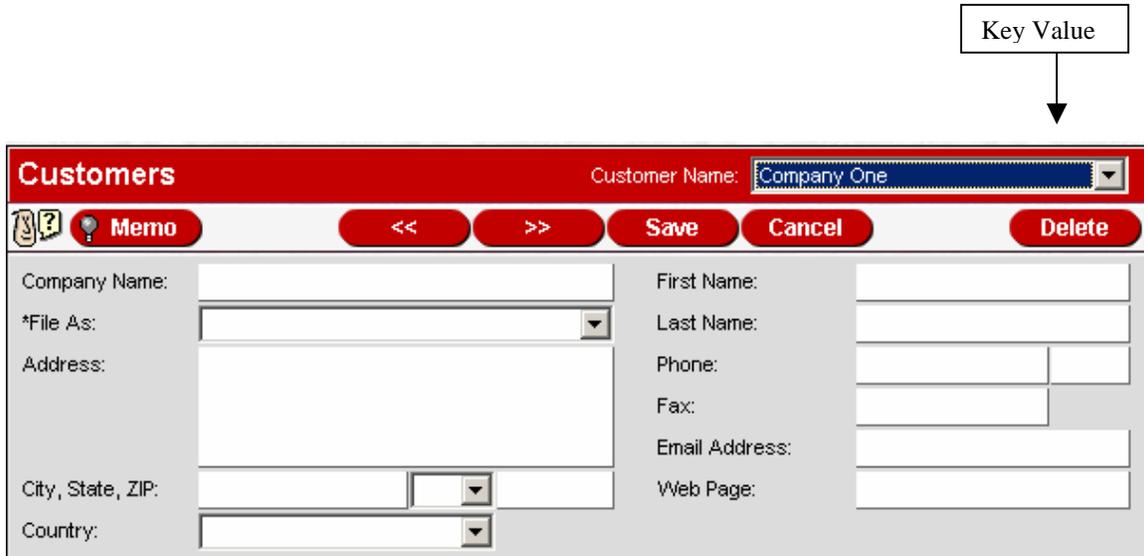
Each of AutoDrive's automated script templates is designed to execute optimally, similar to the applications being tested. The architecture of the script templates has been performance tested and honed to execute faster and more reliable than traditional automated test scripts. The AutoDrive templates have achieved exponential performance gains because they are constructed with high-speed, proven algorithms that can manage any test scenario. The test engineer simply selects with AutoDrive template that matches the manual test case awaiting automation. Then, by simply adding the object names and navigation commands outlined in the manual test case, an automated script is developed. The AutoDrive templates are so dynamic they have even been successfully deployed across multiple applications with no enhancement efforts ever involved. Ingenuity and clever script manipulation are what have made AutoDrive technology focused on not only script functionality but also script performance.

The first automated script template simulates the creation of a record through the user interface of an application. This template was designed to imitate the precise actions and capabilities of a standard user. Navigation from each module and between each field is handled by several customized commands, as opposed to recorded user actions prevalent in automated test scripts throughout the industry. The structure of the create template is only as complex as the associated manual test case and test scenario.

First, the test engineer identifies the key value of the test scenario being automated. This value will be used to retrieve the record and its data after the initial insert has been performed. This particular value is declared as a constant variable in the header of the automated test script. The declaration of a constant variable enables future test engineers to quickly change the name of the record being created very easily, thus making the script highly modular. Then, the appropriate screen navigation and data entry syntax is input into the automated test script. The input of data depends upon the use of ambiguous datapool values that are exercised in every AutoDrive test procedure. Next, the record is committed to the database via the save functionality of the application. At this point the script navigates completely out of the module under test to minimize the possibility of data being stored in cache and never truly being stored in the database. Finally, the module is accessed again and the constant variable is utilized to display the record just entered and perform a quick validation to establish if the record was entered correctly.

For example, the key value of a scenario might be the "Customer Name" of the Customer form. The test engineer declares the key value as a constant variable. Then, after the customer data has been entered into the user interface all the data related to that record can be retrieve by selecting the key value, which is "Customer Name" possibly residing

in a drop-down combo box. When the constant value is located and selected, AutoDrive verifies that the constant variable displayed in the form matches a specified string. If the constant variable does not match the string then a custom error handling routine is enacted, which writes the position and cause of the error to a log file.



AutoDrive's "Update" automated script template is more intricate in nature. The update template design stems from strict requirements necessary to truly verify that a record has been updated correctly. This template begins by either retrieving the record created as a result of the create template or creating a unique record of its own. The record is located by referencing its key value in the module under test. Then, the data displayed in the form is captured and stored in a multi-dimensional array. Next, datapool allocated data updates the forms current data and is committed to the database. The updated data displayed in this form is stored in another multi-dimensional array. Finally, each value in the arrays is compared on a 1:1 basis to ensure that the values stored in the initial array are not equal to the values stored in the updated array. AutoDrive's front-end update template only verifies the data presented on the user interface. All other data verification takes place in the back-end test script templates.

AutoDrive's "Delete" automated script template tests the deletion of a record from an application. This script template begins by simply retrieving a record that will be marked for deletion. Once the record is located and the appropriate 'Delete' buttons have been pressed, the module is completely exited to ensure the validity of the test. Upon re-entering the module where the record has been removed, AutoDrive technology carefully scans all key values for this module for the deleted record. This verification method is error-proof due to the highly specialized AutoDrive algorithm utilized to ensure deletion.

AutoDrive's "Mandatory Field" automated script template and all remaining templates are part of the miscellaneous functionality test-suite. The objective of this template is to test the mandatory field requirements of an application and the warning messages displayed as a result of these particular fields not being populated. Once the mandatory

fields are identified by the technical specification for a module. AutoDrive's use of nested functions quickly populates and tests each mandatory field one at a time, then attempts to save a form with known missing data. Each mandatory field and its associated message box are thoroughly tested based on the response of this attempted save. AutoDrive's custom error handling routine is initiated in the case of any incorrect functionality.

AutoDrive's "Field Formatting" automated script template is focused on testing the validity of data that certain fields will or will not accept. Field masking, boundary testing, incorrect data types and data values are the fundamental testing objectives for this template. This particular AutoDrive template was designed to be completely flexible in order to appropriately test such a diverse assortment of test requirements. A wide variety of encrypted and datapool data are strategically entered into certain fields, which elicit certain field-level validation handling capabilities. This type of data entry is tested by ensuring the correct error messages appear when a record is committed to the database. AutoDrive's custom error handling routine is also initiated with this template in the case of incorrect functionality.

AutoDrive's "Navigation" automated script template may take a minimal amount customization across different applications, due to its window specific verification points. This template verifies the display of certain screens as a result of user navigation through an application. This AutoDrive template relies on the object property, "Title", for all forms. Therefore, any customization to an automated script involves simply redefining the title of the screen resulting from a specified navigation command. This template is functionally reliable to test any variety of navigation through an application because it has the ability to reference any object property of a form. This AutoDrive template is constructed to make all verification points independent of any hard coded values. For example, if a navigation test exits the application and is verifying that Internet Explorer is displayed, AutoDrive can scan any of the inherent object properties of a window and verify if the correct navigation procedure was performed.

The final AutoDrive template, general object and form properties, tests all other visual and non-visual test requirements of an application. Object specific properties that are tested with this template include:

- Tab Order
- Field length
- Data type
- Default Values

AutoDrive simply captures all of the object properties of a form in its default state and stores this data as a baseline. During successive execution's of this script the baseline data values are compared and contrasted to the actual (new) data values. If any discrepancies are identified the run-time log file extrapolates the errors found. Then, it is the test engineer's responsibility to decipher if the reported errors are reportable defects.

### **AutoDrive Script Design:**

AutoDrive combines the principles of quality assurance and the logic of programming. This combination is what the future of software testing is based upon. Test engineering is no longer adhoc testing then assuming an application is ready for release. The future of test engineering is a 100% automated process with definable, repeatable results.

AutoDrive's script design not only effectively performs functional test scenarios but also executes them more efficiently than normal customized scripting processes. AutoDrive achieves functional and performance gains by utilizing tightly coded nested functions and stored procedures that outperform normal recorded scripts. This is made possible by the elimination of countless lines of code that are automatically generated by recorded scripts and replacing them with customized scripting techniques. AutoDrive rarely exercises the record functionality present in automated testing tools due to the unstable output it produces. The auto-generated script writing functionality that testing tools provide is not necessary if you understand the script produced by this generator. AutoDrive not only comprehends this technology, but also understands how to dynamically manipulate the scripting language into output that can be easily understood and modified by any laymen. AutoDrive's readability of automated script code makes this technology a viable solution for companies that desire a repository of scripts that can be easily maintained throughout the life of the application. Thus, eliminating an entire staff of test engineers re-record and re-develop an automated script repository for the latest build of your application.

AutoDrive's utilization of advanced scripting techniques act as an engine for these uniquely designed automated test scripts. The visually presented field and icon naming conventions are what drive AutoDrive's commands. Thus, making this script structure ergonomic because of its reliance on the information directly displayed to the test engineer. Unlike auto generated scripting tools used throughout the industry, AutoDrive has no encrypted syntax involved. It uses language rules that are comprehensible to even the most novice test engineer. AutoDrive design and development process actually makes the task of automated script development independent from a finished or partially developed executable application. This technology allows for script development to begin with a simple prototyped model rather than the finished product because too often companies allocate a small amount of time and resources to sufficiently regression test an application. AutoDrive technology removes this scenario by concurrently developing and testing in conjunction with software developers. By joining the software development cycle early testing efforts are more susceptible to finding defects early before they go unseen due to hectic deliverable deadlines.

Companies rarely invest time and resources into researching the proper ways to automated their testing process. AutoDrive technology is the result of a successful software testing research and development effort. Experienced test and software architects effectively collaborated to design the basis for AutoDrive technology. AutoDrive has gained an advantage over traditional scripting techniques by approaching automated scripting from both a testing and software development standpoint. AutoDrive's script design has also undergone exhausting code reviews to ensure its reliability. It is not difficult to distinguish the vast difference between AutoDrive driven test scripts as apposed to industry standard automated test scripts. The auto-generated

code output produced by most automated testing tools for selecting a simple application icon is as complex as the following:

```
Sub Main
    Window SetContext, "Caption=Microsoft Application X - Microsoft Internet
    Explorer", ""Browser
    SetFrame, "Type=HTMLFrame;HTMLId=frmTopNavigationBar", ""
    Browser NewPage, "HTMLTitle=Top Navigation Bar", ""
    HTMLImage Click, "Type=HTMLImage;HTMLId=imgSales",
    "Coords=26,23"
    Browser SetFrame, "Type=HTMLFrame;HTMLId=frmContent", ""
    Browser NewPage, "HTMLTitle=Sales", ""
    HTMLImage Click, "Type=HTMLImage;HTMLId=imgCustomers",
    "Coords=24,31"
    Browser NewPage, "", ""
    HTML Click, "Type=HTML;HTMLId=divDialogTitle", "Coords=231,14"
End Sub
```

However, the code involved to click a simple icon using AutoDrive technology is as trivial as:

```
Sub Main
    custModule "Sales", "Customers"
End Sub
```

There is an obvious difference in complexity between the two aforementioned script examples. The first example is a feasible approach to automate a test scenario for an application today on the computer it was recorded on. The moment that very same script is executed on a different machine it will fail miserably due to its reliance on screen coordinates and multiple lines of obtrusive auto generated code. AutoDrive has designed its internal stored procedures with intelligent and conditional logic. These stored procedures and functions are what drive AutoDrive Technology to the forefront of all other automated testing techniques. This uniqueness virtually eliminates automated test script entropy between consecutive builds. AutoDrive also makes the test scripts autonomous of the machines they were developed on.

Another feature that sets AutoDrive technology apart from traditionally recorded test scripts are its error handling capabilities. AutoDrive's error handling routine is designed to handle a wide array of defect permutations by allowing the test engineer to mold each error trap in a customized way. By passing certain parameters the error handler can write script specific information to an accompanied log file. This feature enables unattended regression tests to be completely executed even in the case of errors. Upon completion of a regression test the test engineer simply reviews the log file by scanning the customized

error definitions that were output to identify possible defects. Inevitably, conventional automated testing will seize when an error is encountered during automated regression testing. This possibility is removed by AutoDrive's modular test design. AutoDrive technology has expounded upon the concept of a 100% data driven approach to automated script development in a multitude of ways. Scripts are designed to be independent of preexisting, hard-coded data because the data present at design time is rarely identical to that at run time. AutoDrive has the capability of extracting and utilizing data present during playback on any build and platform. Data is read from the application or database at run-time then conditionally manipulated to satisfy a particular predefined test scenario. Every line of code built upon the AutoDrive technology actually has the ability to infer the state of the test in progress and manipulate itself based upon those inferences. This unique quality is due to AutoDrive's engineering from a software development perspective. AutoDrive is built upon the belief that automated test scripts should be functional and adaptable to change as the applications they are testing. By taking this approach AutoDrive test script code is readable, reusable and maintainable.

AutoDrive technology has also taken the ability to display a custom user interface to test engineers to an advanced level. This is highly useful in the circumstance of performing selective regression tests. For instance, if a regression test is only targeted at certain modules, the test engineer can identify which modules they would like to test without having to comment out hundreds of lines of calls to scripts. AutoDrive's customized user interface also allows test engineers to define certain generic parameters if necessary, such as the login information consisting of the user's name or database.

AutoDrive technology has compiled a full library of fully detailed reports, which can be tailored to meet any reporting requirement. AutoDrive Reports provide instant feedback and guidance to assist you in developing quality code standards and deploying a dependable product. These reports are the key to any AutoDrive driven automated regression, functional or performance test. A user simply selects one of the customized reports at the completion of an automated test and all the data is transposed into a completely readable format. Due to AutoDrive's effective report design the reports are automatically formatted and ready to be logged and reported to project managers awaiting results.

AutoDrive is the technological advantage that is mandatory in today's rapid application development environment. The applications of tomorrow change as radically as the requirements presented by the users. AutoDrive technology is the solution to tomorrow's automated testing demands.

### **AutoDrive Generator**

The AutoDrive Generator is a customized script development tool used to virtually eliminate extended automated script development periods. This accelerated script development tool utilizes a custom interface that is specific to your application. The AutoDrive Generator is built upon complex algorithms that take users input, manipulate it and then output AutoDrive designed script code based upon the test engineers

information. The AutoDrive Generator displays a design window enabling the test engineer to easily make alterations to the output if necessary. With a few simple mouse clicks a dynamic, readable, and efficient automated test script can be developed. The AutoDrive Generator is customized according to your applications navigation capabilities, field level design and overall user interface structure.

Easily customizing the AutoDrive Generator begins by analyzing a prototype of the application under test and its navigation process, field level design and overall user interface structure. The capabilities to enter data then navigate between every screen are design characteristics that frequently become test requirements. Therefore, all means of data entry and form navigation are considered a frequently encountered part of any automated script. As a result, the AutoDrive Generator is quickly customized to build and write powerful and dynamic script functions to emulate any test requirement for your application. These functions then interact with customized stored procedures stored in the script repository to perform specific actions, like clicking the ‘Save’ button. Within a matter of minutes a script can be completely developed, then transferred into to the automated testing tool to be compiled and executed.

The main advantage of the AutoDrive Generator is that it eradicates the necessity for expensive automated testing software licenses. This cost is extinguished due to the AutoDrive Generator’s proven ability to produce error free, advanced script output. Companies no longer have to purchase multiple copies of over priced software licenses and technical support for tools that build automated test scripts. AutoDrive technology has taken advantage of the fact that the development, customization and maintenance of test scripts can be done with in the design window of the AutoDrive Generator. The only cost a company will incur is that of a single floating license for an automated testing tool that actually executes the developed test scripts built using AutoDrive technology.

Based on conservative figures, the average Test Engineering group consists of 5 members, with each member having an individual software license costing approximately \$5000 for their automated testing software. As a result, companies are spending close to \$25K per year just in automated testing software licenses. This licensing expense will only increase for organizations that are in a growth mode. The grid below outlines the reasonable escalation in price for automated testing tools based on their increasing demand.

### **Conventional Software Expense Model**

	Year 1 (\$5000 Per License)	Year 2 (\$5500 Per License)	Year 3 (\$5800 Per License)	Total Expense
5 Employees	<b>\$25,000</b>	<b>\$27,500</b>	<b>\$29,000</b>	<b>\$81,500</b>
6 Employees	<b>\$30,000</b>	<b>\$33,000</b>	<b>\$34,800</b>	<b>\$97,800</b>
8 Employees	<b>\$40,000</b>	<b>\$44,000</b>	<b>\$46,400</b>	<b>\$130,400</b>

The combination of both AutoDrive technology and the AutoDrive Generator can drastically reduce the expensive script development and licensing costs outlined above. The ability to effectively develop test scripts with the AutoDrive Generator then execute

the scripts with a single licensed automated testing tool is a proven method for saving time and money.

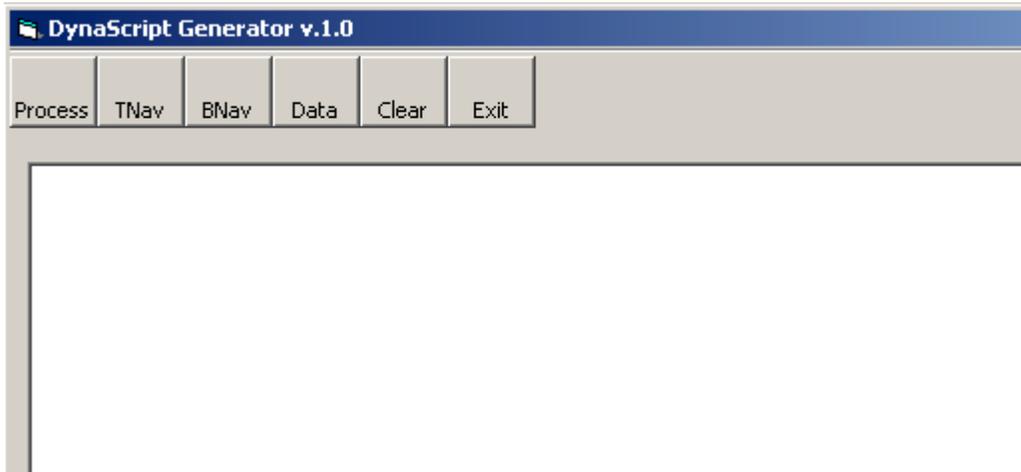
As you can see there is a noticeable difference in savings between the AutoDrive software expense model and the conventional testing software expense model.

### AutoDrive Testing Software Expense Model

	Year 1 (\$5000 Per License)	Year 2 (\$5500 Per License)	Year 3 (\$5800 Per License)	Total Expense
5 Employees	\$5,000	\$5,500	\$5,800	\$16,300
6 Employees	\$5,000	\$5,500	\$5,800	\$16,300
8 Employees	\$5,000	\$5,500	\$5,800	\$16,300

The savings derived from implementing a combination of the AutoDrive Generator and AutoDrive Technology are significant. This is due to the fact that the AutoDrive Generator completely replaces the demand for expensive automated testing tools. By quickly modifying and installing the AutoDrive Generator, a company's automated testing process can begin on the desktop of any test engineer. The key to the AutoDrive expense model is the simplicity and reliability of the output produced by the AutoDrive Generator. This revolutionary tool ends the arduous task of debugging scripts and wasting valuable time that could be spent actually testing software. The AutoDrive Generator allows test engineers to focus on developing robust automated test scripts that can handle a wide array of test scenarios instead of constantly pinpointing script problem areas.

### AutoDrive Generator Interface



Test engineers no longer have to be experts in using complicated testing tools. With a combination of basic computer skills and AutoDrive technology anybody can develop advanced automated test scripts. The AutoDrive Generator displays four basic menus to a user. A user simply selects the appropriate choice from the menu and enters in any additional data they feel necessary. The AutoDrive Generator interprets the users

selections and generates flawless automated test script code. This output then interacts with complex stored procedures and functions to perform designated actions. The AutoDrive Generator was also designed with an editable design window to offer test engineers the ability to further enhance the automated test script output. When the automated test script is deemed complete using the AutoDrive Generator, the output is simply copied and pasted into a script in the single licensed automated testing tool. With the creation of the AutoDrive Generator expensive automated testing tools are only used to compile, execute and library the test scripts. Thus making it necessary to only have a single license automated testing tool. The AutoDrive Generator and AutoDrive technology allows test engineers to focus on testing software thoroughly, while at the same time developing state-of-the-art automated test scripts.

**Advantages:**

The AutoDrive Generator is defining the future of automated testing for companies seeking to save money, increase productivity and deliver dependable software on schedule. These are just a few advantages of the AutoDrive Generator compared to conventional automated testing processes:

- Fosters easy translation and test tractability between manual to automated test scenarios
- Dramatically shortens the script development period by virtually eliminating script troubleshooting instances
- Quickly enables test engineers to develop readable, reusable, dynamic test scripts
- Ensures the uniformity of automated test scripts between different test engineers
- Eliminates the demand for highly paid automated test engineers
- Allows test engineers to focus on “Testing” instead of new/unfamiliar testing tools
- Built for a rapid application development environment where shipping thoroughly tested software in a minimal amount of time is mandatory

The core advantage of AutoDrive Technology is its minimal reliance upon the expensive automated testing tools used to develop automated test scripts. The AutoDrive Generator has successfully enabled test engineers to build test scripts in a minimal amount of time at a fraction of the cost. AutoDrive Technology can be implemented easily with a quick install of the AutoDrive Generator on each test engineer’s machine. Expensive software licenses are no longer necessary with the invention of this revolutionary tool. With the click of a button your testers can exercise their testing logic and allow the AutoDrive Generator build the script code for them. The following hardware and software requirements are necessary to launch and implement the AutoDrive Generator.

**AutoDrive Hardware/Software Requirements**

• Any Windows platform	• 32 KB RAM
• Any text editor	• 200 KB of free disk space

The entire AutoDrive Technology suite of automated testing tools is defining the future of automated quality assurance. AutoDrive Technologies advanced data-driven testing solutions, coupled with minimal operating system requirements make this testing approach mandatory in any test engineering group. Quality Assurance engineering can fully implement this innovative technology into your quality assurance group by contacting our staff at: <mailto:AnthonyL22@hotmail.com>

**AutoDrive Technology Suite includes:**

- AutoDrive Script Design – A 100% data-driven automated script design that retrieves, manipulates and expedites software testing while a test is in progress. The scripts possess internal logic to make choices based on environmental factors.
- AutoDrive Stored Procedures – There is no longer the need for user-action recording to develop automated test procedures with the use of our dynamic technology for performing actions and/or returning values between test procedures (“on the fly”). This technique fosters the concept of test script readability allowing an efficient change control process for script modifications.
- AutoDrive Generator – AutoDrive’s script development application that assembles automated test scripts rapidly with absolute precision. This enables automated test developers to dynamically create and modify test procedures with the click of a button.
- AutoDrive Error Handling – AutoDrive’s dependable approach of pinpointing application errors by combing an application with strategic and unique verification points. This tactic eliminates user error and uncovers true defects found at any level of the application including: database, user interface, DHTML or XML stream and object characteristics.
- AutoDrive Reporting – Provides customized and detailed reports and graphs that give instant feedback and guidance to assist you in developing quality code standards and deploying a dependable product.