Applying STAT to Defense Business Systems

Authored by: Bill Rowell, PhD

09 May 2019

The goal of the STAT COE is to assist in developing rigorous, defensible test strategies to more effectively quantify and characterize system performance and provide information that reduces risk. This and other COE products are available at www.afit.edu/STAT.

STAT Center of Excellence
2950 Hobson Way – Wright-Patterson AFB, OH 45433

Table of Contents

Executive Summary

The intelligent application of Scientific Test and Analysis Techniques (STAT) to Department of Defense (DoD) weapon systems has proven to be a key component in enabling the efficient, effective, and rigorous testing of weapon systems to support decision making at all levels. With the recent emergence of Defense Business Systems (DBSs) as defined by DoDI 5000.75 Business Systems Requirements and Acquisition, there has also been a similar interest throughout the DoD testing community in leveraging STAT for DBSs. This paper has two goals:

- Provide insights into the appropriate STAT tools to apply to the test and evaluation of DBSs
- Present lessons learned from applying the STAT process to the test and evaluation of DBSs

Keywords: defense business system, combinatorial optimization, information technology, design of experiments, scientific test and analysis techniques, test and evaluation

Introduction

Scientific Test and Analysis Techniques (STAT) are deliberate, methodical processes and techniques that create traceability from requirements to analysis. All phases and types of testing (developmental, operational, integrated, and live fire testing) strive to deliver defensible and decision-enabling results in an efficient manner. The incorporation of STAT provides a rigorous methodology to accomplish this goal. The February 2017 publication of DoDI 5000.75 Business Systems Requirements and Acquisition and the subsequent establishment of a web-based Business Community of Practice testify to the distinctive nature of the acquisition of Defense Business Systems (DBSs). These developments also highlight the need to tailor the acquisition process, including Test and Evaluation (T&E) and the application of STAT, to the nature of DBSs.

The STAT processes and techniques appropriate to a specific requirement can be identified by applying the STAT Center of Excellence (COE)-defined STAT process to the requirement. The Tools section provides insights that the STAT COE has gained into STAT tools frequently applied to the T&E of a DBS's stochastic and deterministic processes. The subsequent Lessons Learned section contains insights and recommendations gained from applying STAT to DBSs.

Tools

Stochastic Processes

A stochastic process is one with an output (response) that is a random variable, that is, the possible values are the outcomes of a random phenomenon. The response time to retrieve and render a report

from a database is an example of a stochastic process because even when identical reports are run at different times the response-time values vary randomly based on the inherent variability (noise) associated with the time to transmit the query to the server, perform the query, transmit the results to the user, and display the report to the user.

System performance

DBS stochastic processes frequently have system performance measures, that is, system attributes that can be expressed typically in units of time or capacity. Typically, system performance measures include measures such as transaction execution time, query response time, document generation time, document rendering time, document download time, workflow process time, and maximum active users where the mix of objects involved in the process (transactions, queries, documents generated, users, etc.) is representative of the expected mix in the production environment. This type of requirement may be expressed in absolute terms for a long running process, for example, a requirement could stipulate that a work flow process cannot exceed 24 hours. For processes that typically take seconds, the nature of the requirement is usually to ensure that the experience of the user of the system is acceptable for a high proportion of the time. For example, a requirement may be stated that the user must be able to view the results of 95% of all queries within 5 seconds. The key STAT tools for addressing system performance requirements are statistical methods (descriptive and inferential statistics) and Design of Experiments (DOE).

Statistical Methods

Descriptive statistics summarize data using graphical approaches and numerical summaries. As their name implies, they are helpful in understanding test data outputs but by themselves do not provide sufficient evidence to make the kind of rigorous statement necessary for requirement verification. Figure 1 displays a sample list of descriptive statistical methods.

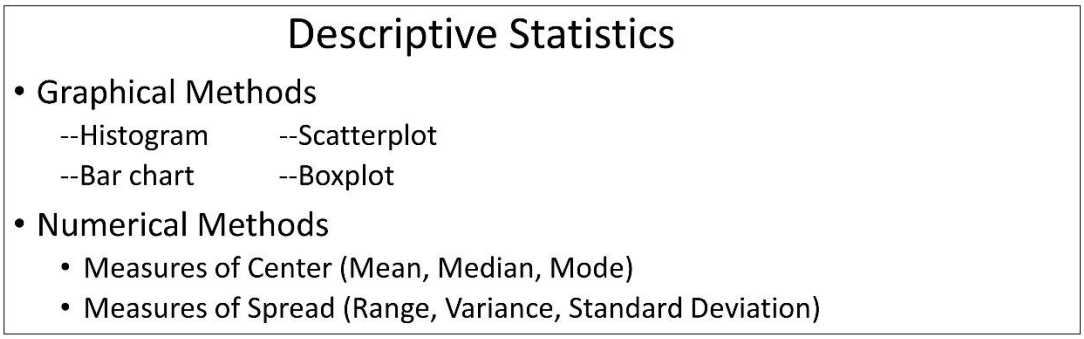


Figure 1: Summary of descriptive statistics methods

On the other hand, correct use of inferential statistics can provide evidence for rigorous verification of system performance requirements. Inferential statistics are methods for making decisions or predictions about a population based on data obtained from a sample. Figure 2 displays definitions for a population and a sample. In the case of a system performance requirement, the population may be the set of report retrieval times for all reports in the application and the sample may be a random subset of all the reports typically requested by users in the operational environment. By taking a random subset, we help ensure that the sample will be representative of the desired population.

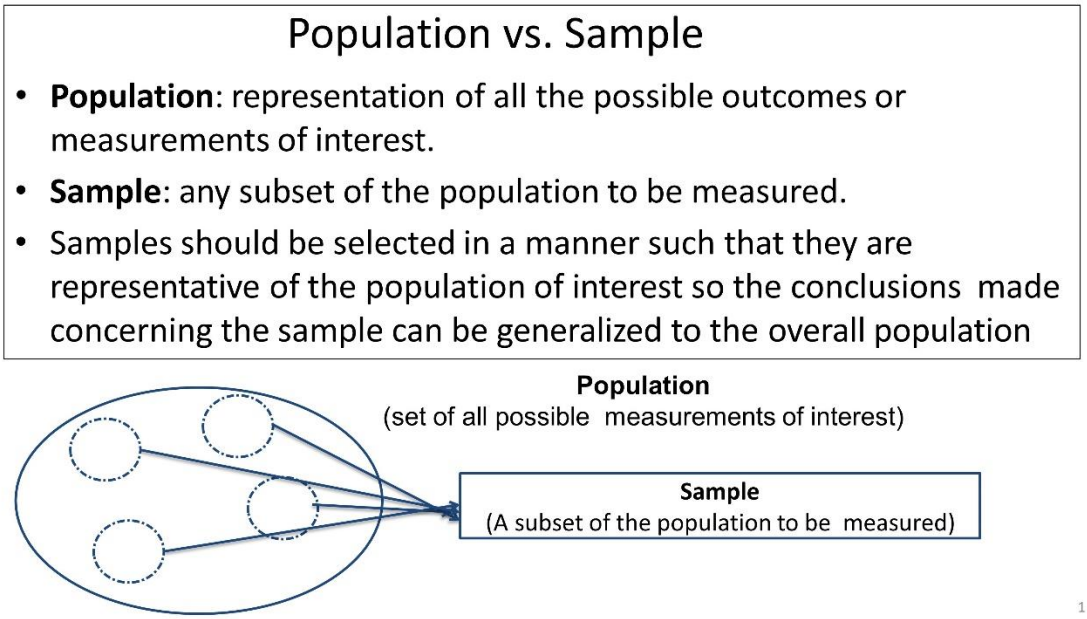


Figure 2: Definitions of Population and Sample

Figure 3 illustrates a high-level example of how we use inferential statistics to verify a requirement. Let's say for example we want to make a rigorous statement about the time it takes a DBS to retrieve a report. We have formulated a null hypothesis that the population mean (μ) of the retrieval response time is less than or equal to 7 seconds 95% of the time and selected a sample size (12). Next we identify the appropriate statistic to compute (the 95% one-sided upper confidence interval). Next we collect the representative sample and compute the statistic. If the computed statistic (one-sided confidence interval [t, ∞]) does not contain the hypothesized mean (7 seconds), that is 7 seconds is less than the lower bound on the one-sided confidence interval, we can conclude that we fail to reject the null hypothesis (requirement has been verified); otherwise, we can conclude that we can reject the null hypothesis (requirement has not been verified).

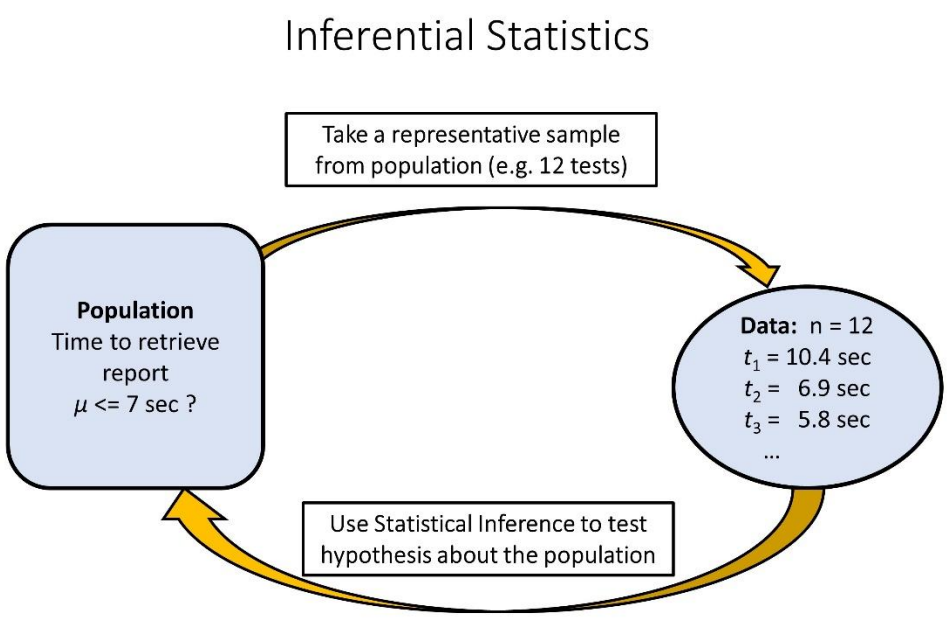


Figure 3: Inferential Statistics Process Example

Design of Experiments (DOE)

The purpose of this section is to discuss how DOE is applied to a DBS stochastic process and illustrate this application with a realistic example.

STAT tools are frequently used to determine how the value of the response is influenced by the values of factors or inputs associated with both stochastic and deterministic processes. In our report retrieval time example some of these potential factors include the number of queries that must be performed to generate the report, the number of tables in each query, the current workload on the database server, the size of the report, the current traffic on the network, and the download speed to the machine submitting the report request.

Figure 4 summarizes the key concepts behind statistical DOE as applied to stochastic processes

Design of Experiments (DOE)

- Focused on metrics that are outcomes of a stochastic process
- Not simply observing factors/responses but systematically manipulating factors
- Trying to characterize the effects of factors (causality) on responses over the test/factor space not the less rigorous correlation of factors and responses
- Assumes existence of an underlying model connecting factors to responses enabling predictability over the test/factor space

Figure: 4 Key Aspects of Design of Experiments

Figure 5 shows a high-level view of the various phases of the testing process and how it supports the program's decision making. The figure explicitly breaks out the elements that are performed during the planning phase as part of applying STAT, specifically DOE in this case.

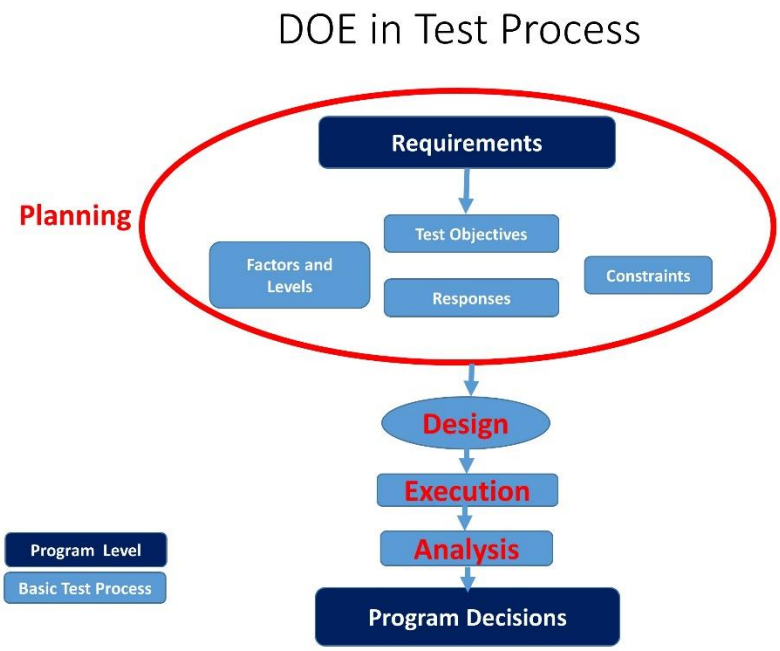


Figure 5: How DOE Fits into Test Process

Figure 6 presents the information for each of the five elements of the planning phase for our example DBS stochastic process—the report retrieval response time. In our scenario Subject Matter Experts (SMEs) have identified four factors (# of tables in query, report size, system load, and download speed) that may affect the report retrieval response time and two levels for each factor representing the range of possible factor values. To avoid unnecessary complications, we have assumed that the number of tables in the query generating the report can take on any value between two and five. During actual execution of this test, only discrete numbers would be considered for the number of tables.

DOE Planning Phase

- Report Retrieval Response Time Requirement
  - The report from 80% of all queries will be displayed to the user within 15 seconds or less of the time the query was submitted
- Test Objective
  - Characterize report retrieval response time performance over the range of values of significant test factors
  - Rigorously verify that the requirement has been met
- Response
  - The time in seconds from when the user submitted the request until the report is displayed
  - Does not include network propagation delays/Includes all types of queries and report download
- Factors and levels

Number of Tables in Query	Report Size (MB)	System Load (Txns Per Sec)	Download Speed (Mbps)
2	5	100	10
5	20	500	40

- Constraint(s): Maximum number of test cases

1

Figure 6: DOE Planning Phase Info

Figure 7 depicts both the design (8 test cases in the first four columns) and response times (results) of running all 8 test cases (last column). JMP (a statistical too)l was used to generate the 2-way interaction test design and simulate the response times based on the fitted, normalized linear regression model displayed in Figure 8. The model's equation can be used to estimate report retrieval response times for the range of factor values (the first test objective in Figure 6). Specifically, the chart shows that the range of response times without taking into account the normal error term is between 2.7525 and 10.5275 seconds. The non-zero coefficients in the equation of the underlying model, which has been standardized, indicate that all of the main factors and the interaction between the Number of Tables in Query and Report Size are statistically significant, that is, significantly influence report retrieval response time; and the other 2-way interactions have been dropped as not being sufficiently significant. The magnitude and direction of the coefficients of the main factors and single interaction term shows the

strength and direction of the change in response time with changes in the factor value. For example, not unexpectedly as Download Speed increases Report Retrieval Response Time decreases. Given the absolute value of the Download Speed coefficient is 0.9 and the difference in the upper and lower bounds is 30 Mbps, for each 1 Mbps increase in Download Speed retrieval response time will decrease by 0.03 seconds.

DOE Design/Execution Phase

- JMP randomized screening design with all 2-way interaction terms in model (8 test cases generated)

Number of Tables In Query x_1	Report Size (MB) x_2	System Load (Txns Per Sec) x_3	Download Speed (Mbps) x_4	Report Retrieval Response Time (sec) Y
2	20	100	40	3.81
5	5	500	10	9.84
5	20	500	40	9.00
2	5	100	10	4.18
2	20	500	10	12.31
5	5	100	40	1.39
5	20	100	10	7.66
2	5	500	40	5.69
				6.74 (Mean)

Figure 7: DOE Design/Execution Info

DOE Analysis Phase (1)

- Linear regression model with 2-way interaction

$$y = \beta_0 + \sum \beta_j x_j + \sum\sum \beta_{ij} x_i x_j + \varepsilon$$

  - Dependent/response variable: y
  - Y-axis intercept: β_0
  - Independent variables/factors: x_i, x_j where i, j = 1, 2, 3, 4
  - First-order terms coefficients: β_j where j = 1, 2, 3, 4
  - Interaction terms coefficients: β_{ij} where i is not equal to j
  - ε: normally distributed error term
- Fitted, normalized model

y = 6.5575 - .0825*$\mathbf{x_1}$ +.875*$\mathbf{x_2}$ + 2*$\mathbf{x_3}$ -.9*$\mathbf{x_4}$ - .1125*$\mathbf{x_1}$*$\mathbf{x_2}$ + Error(Normal (0,1))

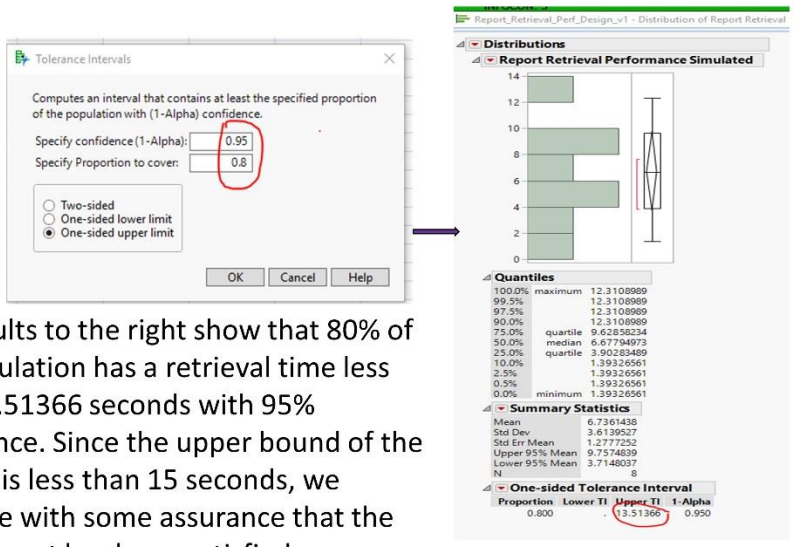
where $\mathbf{x_i}$ = +1 (upper) or -1 (lower). Low: 2.7525 High: 10.5275 without error factor

All main effects plus the interaction of Number of Tables in Query ($\mathbf{x_1}$) and Report Size ($\mathbf{x_2}$) are significant at 95% confidence level.

Figure 8: Fitting Model with Test Results Data

Figure 9 depicts the analysis used to rigorously verify the hypothesis that 80% of all queries will be displayed to the user within 15 seconds or less of the time that the query was submitted (the second test objective in Figure 6). This type of requirement was verified using the tolerance interval calculator found in JMP.

DOE Analysis Phase (2)



The results to the right show that 80% of the population has a retrieval time less than 13.51366 seconds with 95% confidence. Since the upper bound of the interval is less than 15 seconds, we conclude with some assurance that the requirement has been satisfied.

Figure 9: Verifying Satisfaction of Performance Requirements

Note that load testing and instrumentation are two critical areas for correctly performing DBS system performance tests. Employment of load testing is done to create the expected operational profile of transactions in the testing environment, that is, a specified number of transactions per second of each type of transaction that will be running when the actual performance testing is occurring. At a minimum, the end-to-end segments that contribute to the system's performance measure must be instrumented. Although the formal Threshold/Objective value itself may not include all segments of the end-to-end system performance time, it is important that the instrumentation captures the time associated with all segments of the end-to-end performance time. The end-to-end data values will be helpful in fully assessing the contribution to the users' experience of all segments and assisting with taking the actions needed to adjust the users' experience.

System scalability

A typical requirement for a DBS is that its performance does not degrade or degrades gracefully as workload beyond the expected daily level, which is typically the maximum daily load, is added in the form of concurrent users, batch jobs, increased mission tempo (transactions per second), etc. Meeting this type of requirements ensures that the system is designed so that additional capacity is available when needed.

In addition to statistical methods and DOE STAT techniques load testing and instrumentation are frequently essential to achieving test objectives. Note that under some circumstances it may not be feasible to scale the size of the system (per requirements) to empirically test whether the system meets its scalability requirements. For example, substantially increasing the number of service members in a service's personnel system to conduct a system scalability test in anticipation of a national emergency, requires generating an extremely large amount of custom personnel data. In these cases, in lieu of full-scale testing, a program may accept verifying the scalability requirements via an analysis of an architectural performance model of the system.

Mobile performance

Increasingly, DBSs need to service mobile devices which deploy in no- and low-communication environments. The Joint Operational Medicine Information Systems (JOMIS) Mobile Computing Capability (MCC) and the Navy Maritime Maintenance Enterprise Solution – Technical Refresh (NMMES-TR) are good examples of DBSs that face these types of communication challenges. Compared to performance testing of always-connected devices, mobile performance testing involves a much greater number of variable factors, such as wireless network conditions, the types of devices, device performance, packet loss, latency and bandwidth, which may have significant effects on system performance. Although many of these factors may not be under direct control of the system's Program Management Office, it is still important that the mobile system testing design take them into account in order to characterize the performance of the system throughout the range of operating conditions.

Deterministic Processes

A deterministic process is one with an output (response) that is fully determined by the process's initial conditions and input values. For example, assuming there are no stochastic processes within a software program, such as ones that generate random numbers, the output of running a software program for the same inputs and the same initial conditions remains the same every time the program is executed. On the other hand, a stochastic process can generate a different output for the same initial conditions and same input values, thereby creating a distribution of output values instead of a single output value for multiple executions of the process.

Software testing

The purpose of an individual software test case is to verify that valid inputs and behavior produce the expected outputs (positive testing) and that invalid inputs and behavior are detected and handled appropriately (negative testing). Positive testing ensures that the system can accomplish the intended functionality whereas negative testing ensures that the system behaves gracefully, for example, by catching invalid inputs and helping users to correct their inputs and behavior so the desired functionality can be exercised. Faults are detected whenever the given inputs to a test case do not produce the expected outputs. Fault detection applies to both positive and negative test cases. Ideally a set of test cases could be generated to identify all possible faults—an objective that is impractical except in trivial

software programs. The three sections below describe how STAT tools can help design and execute a set of software test cases that will effectively and efficiently identify faults.

Combinatorial Optimization

STAT can be especially helpful when there is a need to test a deterministic process with a binary (no fault/fault) output (response) that is dependent on a large number of discrete inputs/factors, each with many potentially valid settings (levels). For example, consider the parts ordering form with 5 input fields: Availability, Delivery Mode, Urgency, Delivery Location, and Funding Source depicted in Figure 10. With each of the first 3 fields having 4 different choices in the dropdown and the remaining 2 each having 3 different choices, a total of 576 (4x4x4x3x3) possible test cases must be executed to test every factor combination unless some of the combinations are not feasible, such as ordering a part to be delivered overnight by a re-supply ship.

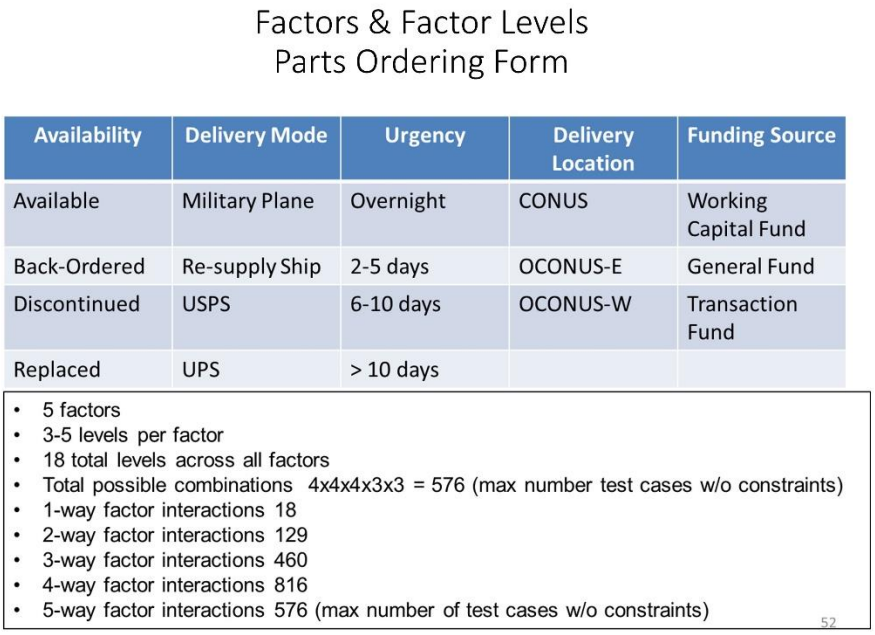


Figure 10: Analysis of factors in software application form

Because of resource and other limitations, it is frequently impossible to execute all of the possible test cases (exhaustive testing) in this kind of situation. The questions to be answered are how many test cases to execute and which test cases to execute. Fortunately, empirical studies have shown that regardless of the actual number of factors in a software application form, a very high percentage of software faults arise from the interaction of a small number of factors (6 or less) (Kuhn, 2004). Combinatorial Optimization (CO), an advanced mathematical technique, can be used to identify a

significantly smaller (less than exhaustive) number of test cases based on identifying 6-way or less factor interactions that are highly likely to cover a high percentage of software faults. Figure 10 displays the factors and levels in the example parts ordering form and the number of possible t-way factor interactions for t = 1, 2, 3, 4, 5. Applying the National Institute of Science and Technology (NIST)-developed CO tool, Automated Combinatorial Testing for Software (ACTS), to the parts ordering form described above, we can identify effective and efficient sets of test cases over a range of different t-way factor interactions (Automated Combinatorial Testing for Software). Figure 11 shows the test case set and coverage metrics for application of the ACTS 2-way factor solution while Figure 12 shows the same data for the ACTS 3-way factor interaction. Thus, ACTS provides the decision maker with valuable tradeoff information (number of test cases vs interaction coverage) to help allocate testing resources.

ACTS 2-Way Solution

16 Test Cases

Test Case #	Availability	Delivery Mode	Urgency	Delivery Location	Funding Source
1	Available	Military Plane	2-5 days	OCONUS-East	General Fund
2	Available	Re-supply ship	6-10 days	OCONUS-West	Transaction Fund
3	Available	USPS	>10 days	CONUS	Working Capital Fund
4	Available	UPS	Overnight	OCONUS-East	Transaction Fund
5	Back ordered	Military Plane	2-5 days	CONUS	Working Capital Fund
6	Back ordered	Re-supply ship	6-10 days	OCONUS-East	General Fund
7	Back ordered	USPS	>10 days	OCONUS-West	General Fund
8	Back ordered	UPS	Overnight	CONUS	Transaction Fund
9	Discontinued	Military Plane	2-5 days	OCONUS-West	Transaction Fund
10	Discontinued	Re-supply ship	6-10 days	CONUS	Working Capital Fund
11	Discontinued	USPS	>10 days	OCONUS-East	Working Capital Fund
12	Discontinued	UPS	Overnight	OCONUS-West	General Fund
13	Replaced	Military Plane	2-5 days	CONUS	General Fund
14	Replaced	Re-supply ship	6-10 days	OCONUS-West	Working Capital Fund
15	Replaced	USPS	>10 days	OCONUS-East	Transaction Fund
16	Replaced	UPS	Overnight	CONUS	Working Capital Fund

Factor Interaction Coverage

1-way factor interactions	18/18 100%	4-way interaction	80/816 10%
2-way factor interactions	129/129 100%	5-way interaction	16/576 3%
3-way factor interactions	159/460 35%		

Figure 11: Coverage Analysis of 2-way Factor Interaction Solution

ACTS 3-Way Solution
64 Test Cases (first 16 shown)

Test Case #	Availability	Delivery Mode	Urgency	Delivery Location	Funding Source
1	Available	Military Plane	Overnight	CONUS	Working Capital Fund
2	Available	Military Plane	2-5 days	OCONUS-East	General Fund
3	Available	Military Plane	6-10 days	OCONUS-West	Transaction Fund
4	Available	Military Plane	>10 days	CONUS	General Fund
5	Available	Re-supply ship	Overnight	OCONUS-East	Transaction Fund
6	Available	Re-supply ship	2-5 days	OCONUS-West	Working Capital Fund
7	Available	Re-supply ship	6-10 days	CONUS	General Fund
8	Available	Re-supply ship	>10 days	OCONUS-East	Working Capital Fund
9	Available	USPS	Overnight	OCONUS-West	General Fund
10	Available	USPS	2-5 days	CONUS	Transaction Fund
11	Available	USPS	6-10 days	OCONUS-East	Working Capital Fund
12	Available	USPS	>10 days	OCONUS-West	Transaction Fund
13	Available	UPS	Overnight	CONUS	General Fund
14	Available	UPS	2-5 days	OCONUS-East	Working Capital Fund
15	Available	UPS	6-10 days	OCONUS-West	Transaction Fund
16	Available	UPS	>10 days	CONUS	Working Capital Fund

Factor Interaction Coverage

1-way factor interactions 18/18 100%
2-way factor interactions 129/129 100%
3-way factor interactions 460/460 100%
4-way interaction 316/816 39%
5-way interaction 65/576 11%

Figure 12: Coverage Analysis of 3-way Factor Interaction Solution

Automated Software Testing (AST)

AST involves the use of software tools to execute pre-scripted tests that would normally be executed manually. Automated testing tools are capable of executing tests, reporting outcomes, and comparing results with earlier test runs. Tests carried out with these tools can be run repeatedly throughout a DBS's software development life cycle.

AST can offer significant improvements in test efficiency and effectiveness for DBSs, but the benefits of implementing AST in any particular situation must be weighed against the required investment in personnel, process, and technology. Most successful AST implementations follow a deliberate six-phase process (Pre-plan, Plan, Design, Execute, Analyze, and Maintain). In the planning phases of an AST

program, a return on investment (ROI) or business case analysis must be performed that, among other considerations, takes into account the relevant characteristics of the expected software development environment (Agile, DevOps, etc.) . An AST Best Practice (Pestak, 2017) and an AST Implementation Guide for Managers and Practitioners (Simpson, 2018) are available through the STAT COE.

Pseudo-Exhaustive Verification (PEV)

A rule-based DBS, such as an enterprise system that determines the eligibility of military service members for various financial entitlements, may contain a large number of complex business rules that need to be coded correctly. An example rule is depicted in Figure 13.

Example Business Rule

A member is eligible for Continental United States (CONUS) Cost of Living Allowance (COLA) with dependents for a period not to exceed two months from the first day of absence if each of the following is true:

1. The member is one of the following:
 a. in a Regular Component **(rc)** **or**
 b. in a Reserve **(r)** or Guard Component **(gc)** and is called (or ordered) to Active Duty **(ad)** for a period of not less than 140 days **or**
 c. in a Reserve **(r)** or Guard Component **(gc)** and is called (or ordered) to Active Duty **(ad)** in support of a contingency operation **(co)** **and**
2. The member has dependents. **(d)** **and**
3. The member is one of the following:
 a. E-1 **(e1)** **or**
 b. E-2 **(e2)** **or**
 c. E-3 **(e3)** **or**
 d. E-4 with four or less years of creditable service for Basic Pay **(e4b)** **and**
4. The member is in a status of Excess Leave **(x)** **and**
5. The member is anticipated to return to duty. **(rd)**

Figure 13: Complex Compensation Eligibility Rule

Fully verifying that the software correctly captures the logic of only a single complex rule may require a few hundred test cases. NIST has developed a STAT tool, Psuedo-Exhaustive Verification (PEV), to rapidly generate the complete set of test cases required to verify each complex business rule (Kuhn, 2016). Figure 14 shows all 20 test cases generated by PEV to verify rules that result in the member being eligible whereas Figure 15 depicts 36 of the 250 test cases required to verify rules that result in the member being ineligible. To fully realize the value of the PEV capability for systems that have a large number of business rules, this tool must be integrated into an end-to-end process. This process starts with translating a business rule into logic expressions that the tool can then use as input and ends with the System Under Test executing each of the test cases generated by the tool.

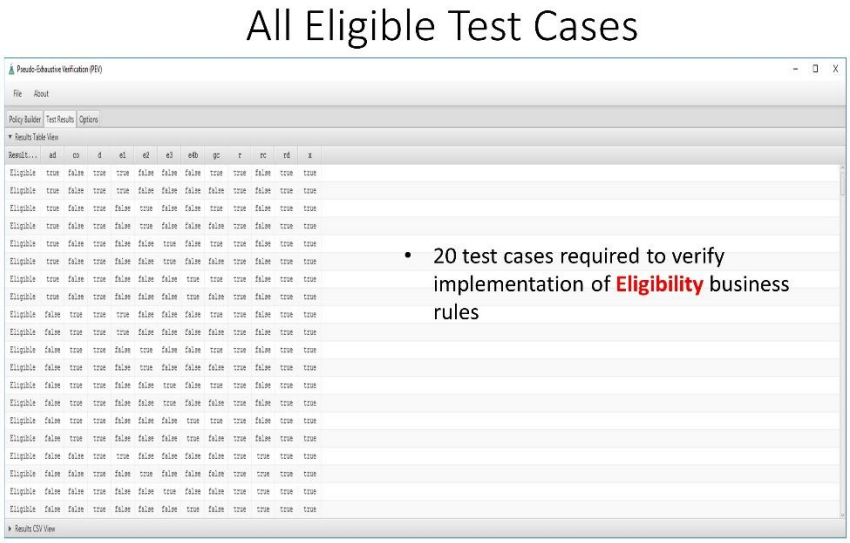


Figure 14: PEV Generation of All Eligible Test Cases

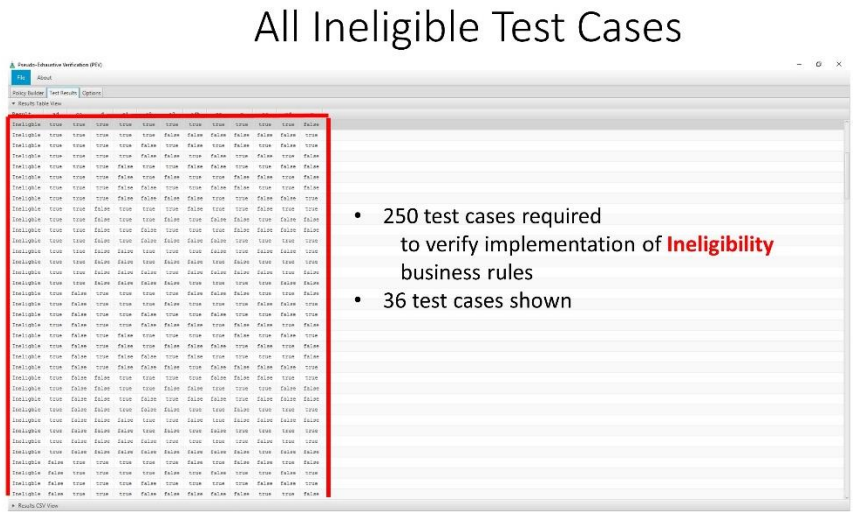


Figure 15: PEV Generation of All Ineligible Test Cases

Interface/Business Process Testing Prioritization

A DBS, especially an enterprise system, frequently has multiple interfaces and business processes. Associated with each interface and business process are the set of detailed data exchanges that define the total functionality provided by the interface or business process. Consequently, testing these interfaces and business processes is usually lengthy and complex requiring substantial resources. A key

issue is how to sequence the testing of the interfaces and business processes to increase the likelihood of finding critical interface and business process problems early in the testing process.

Multiple Attribute Decision Analysis (MADA) has been applied to sequence the testing of interfaces and business processes to minimize the risk associated with the overall plan for interface and business process testing (Stimpson, 1981). Interface/business process risk is first broken down into various risk factors associated with testing a set of interfaces/business processes, such as criticality, complexity, and maturity. Each of the interfaces/business processes will be assigned a value for each factor indicating the relative risk associated with finding a problem with that interface/business process--the higher the value assigned, the greater the likelihood that the interface/business process will experience problems during interface testing attributable to that factor. Factors can be weighted based on their relative importance, and testing-order dependencies among interfaces/business processes can be captured. The MADA-generated interface/business process test sequence will insure high-risk interfaces/business processes are addressed early in the testing process. Figure 16 depicts a simple notional example of how to use MADA to sequence the testing of DBS interfaces. Note that the general approach of executing test cases by frontloading the testing with the high risk test cases can also be applied to other areas of testing.

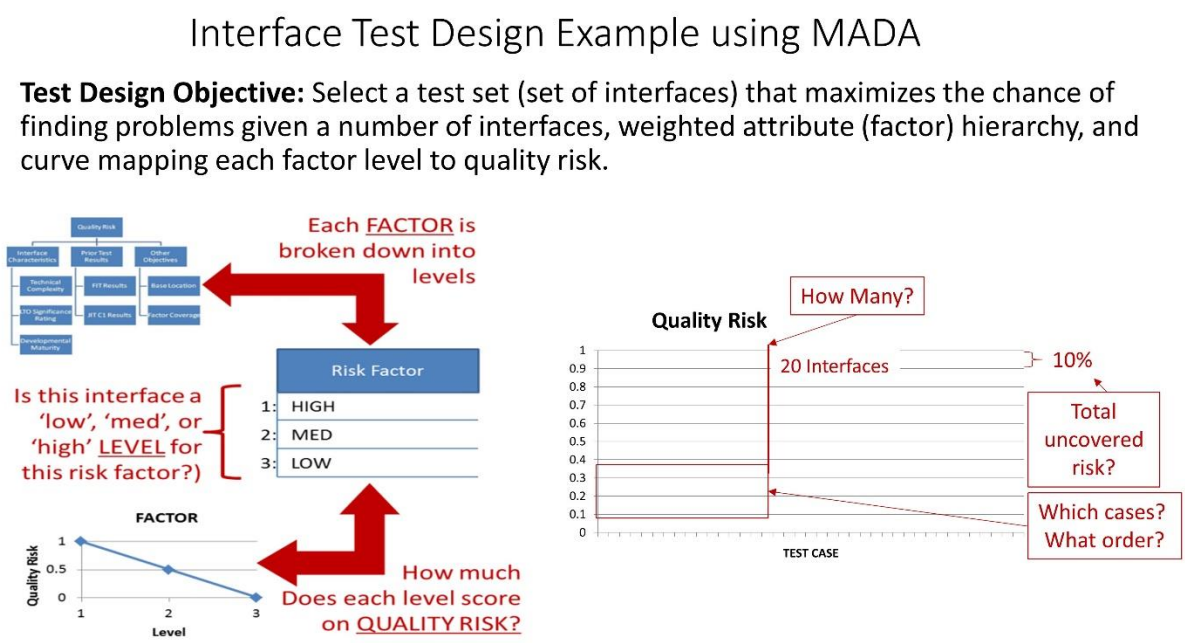


Figure 16: Applying MADA to reduce risk of interface testing

Lessons Learned

User Response Time Requirements

A key aspect of T&E of DBSs is formulating user response time requirements that not only take into account user expectations and experience but are also enable use of rigorous methods to verify.

User Expectations

Sometimes user response time requirements appear to be set based more on what can be easily achieved rather than on user expectations. User response time requirements should take into account user expectations. A good approach is to formally survey or interview users to identify an acceptable range of response times for each type of transaction.

User Experience

Additionally, transaction response time requirements can be set up to measure only the processing time done by the system under test (SUT) based on the argument that the program is only responsible for that part of the end-to-end response time that it controls. While technically correct, such an approach does not acknowledge the unavoidable responsibility of the program to work with all segments on the end-to-end request-response time path to insure the total user experience is acceptable. A better approach is to specify an end-to-end user response time that reflects the desired user experience and include explicit assumptions about non-SUT components of the end-to-end path, such as continuous network connectivity and continuous information exchange partner availability.

Rigorously verifiable

Finally, sometimes user response time requirements are specified in pass/fail terms,; for example, the transaction response time must be less than or equal 10 seconds. Such brittle requirements don't accurately reflect the stochastic nature of user response times. There is no rigorous way to determine a sample size that would quantify the risks of not meeting this type of requirement. Furthermore, it is not clear how to proceed with additional testing if response times exceed the maximum.

A more rigorous way is to formulate user response time requirements as statements that can be straightforwardly transformed into statistical hypotheses such as the following:

- The % of the population (proportion) have a response time less than or equal to x seconds
- The mean response time of the population is less than or equal to x seconds

With addition of suitable confidence levels the first requirement above can be translated into one-sided tolerance interval and the second into a one-sided confidence interval.

Contractual Considerations

In addition to the STAT support provided by the STAT COE DBS programs need to ensure that the system implementer is also on contract to employ STAT in their testing activities. Below is a list of STAT-related tasks that can be incorporated into the Performance Work Statement (Harman, 2018):

- Use STAT during the process of planning, designing, executing and analyzing the testing of requirements with responses/outputs that are random variables such as user response times and download times.
- Use optimization techniques, such as Combinatorial Optimization, to generate a covering set of efficient test cases for complex user screens, that is, ones with a large number of input fields (factors) that each have numerous different discrete values (levels) from which to choose.
- Demonstrate that test sets exhaustively cover complex, critical business rules by using validated tools such as the NIST Psuedo-Exhaustive Verification tool.
- Incorporate STAT-related information in the T&E Contract Data Requirements Lists (CDRLs), such as Software Test Plan, Software Test Description, and Software Test Report.
- Prioritize and scope test activities involving business processes and system interfaces using a capabilities risk-based approach such as Multiple Attribute Decision Analysis.

System Reliability Availability Maintainability (RAM)

It's critical that DBS programs develop a simple, comprehensive but effective approach to handling system RAM. Otherwise, programs can easily get involved in a quagmire debating the value of meaningless metrics, often based on hardware systems rather than software systems, that fail to drive behavior in the desired direction. Here are three general guidelines for an effective approach to RAM that avoids these pitfalls:

- The overall focus of RAM should be on achieving the threshold/objective operational availability of the system to perform its mission.
- The reliability component should emphasize a process and a set of metrics that drive rapid identification and removal of software defects that most affect mission accomplishment.
- The maintainability component should likewise be process-oriented with metrics that drive behavior that supports operational availability.

Conclusion

Table 1 summarizes the common applications of STAT tools to the T&E of DBSs. The effective use of STAT results in an iterative process that begins with the requirement and proceeds through the generation of test objectives, designs, and analysis plans all focused on definitively addressing the requirement. Moreover, the effective use of STAT in T&E of a DBS ensures adequate coverage of input ranges and use cases as well as enables optimal employment of test resources.

Table 1: DBS Application Area-STAT Tool Mapping

Defense Business System
Application Area-STAT Mapping

Process Type	Application Area	Example Metric(s)	STAT Tool(s)
Stochastic	System Performance	Transaction Execution Time, Query Response Time, Document Generation Time, Workflow Process Time	Statistical Methods, Design Of Experiments
	System Scalability	Degradation in Document Download Time At Maximum Concurrent Users	Statistical Methods, Design Of Experiments
	Mobile Performance	Packet Loss, Bandwidth, Wireless Network Latency, Document Download Time	Statistical Methods, Design Of Experiments
Deterministic	Software Testing	% Of Faults Covered	Combinatorial Optimization
		Cost Per Fault Covered	Automated Software Testing
		% Business Rules Fully Verified	Psuedo-Exhaustive Verification
	Interface/Business Process Testing Activities	% Reduction In Risk	Multiple Attribute Decision Analysis

Key lessons learned on applying STAT to DBSs include:

- User response time requirements need to be carefully scrutinized to ensure they are rigorous and take into account user expectations and experience.
- DBS Performance Work Statements need to include specific tasks to ensure that the contractor provides adequate STAT support in their testing activities.
- DBS programs need to develop a simple, comprehensive but effective approach to handling system RAM that focuses on meeting operational availability requirements supported by process-oriented reliability and maintainability components.

References

"Automated Combinatorial Testing for Software." National Institute of Standards and Technology, csrc.nist.gov/projects/automated-combinatorial-testing-for-software. Accessed 13 February 2019.

Harman, Michael and Rowell, William. "Specifying STAT Requirements In Defense Contracts Revision 2," STAT COE-Report-36-2018, November 2018. www.afit.edu/stat/statcoe_files/Specifying STAT Requirements in Defense Contracts R2.pdf. Accessed 2 May 2019.

Kuhn, R. et al. "Pseudo--exhaustive Testing of Attribute Based Access Control Rules", Ninth IEEE International Conference on Software Testing, Verification and Validation Workshops, ICST Workshops 2016, pp. 50-58.

Kuhn, R. et al. "Software Fault Interactions and Implications for Software Testing," *IEEE Transactions on Software Engineering*, vol. 30, no. 6, June 2004, pp. 418-421.

Pestak, Thomas and Rowell, William. "Automated Software Testing Practices and Pitfalls," STAT COE-Report-02-2017, April 2017. www.afit.edu/stat/statcoe_files/AST_Practices_and_Pitfalls.pdf. Accessed 13 February 2019.

Simpson, Jim et al. "Automated Software Test Implementation Guide for Managers and Practitioners," STAT COE-Report-05-2018, October 2018. www.afit.edu/stat/statcoe_files/0214simp 2 AST IG for Managers and Practitioners.pdf. Accessed 23 May 2019.

Stimpson, Wayne A. "MADAM: Multiple-Attribute Decision Analysis Model Volume 1", Thesis, Air Force Institute of Technology, December 1981. www.dtic.mil/dtic/tr/fulltext/u2/a111104.pdf. Accessed 13 February 2019.