Integrated Intelligent Modeling, Simulation and Design of Experiments for Software Testing Process

LJUBOMIR LAZIĆ
Department for Mathematics and Informatics
State University of Novi Pazar
SEROBIA
llazic@np.ac.rs, http://www.np.ac.rs

NIKOS E MASTORAKIS
Technical University of Sofia,
English Language Faculty of Engineering
Industrial Engineering, Sofia, BULGARIA
http://www.wseas.org/mastorakis

Abstract: This paper presents some original solutions with regard to deployment of the US Department of Defense Simulation, Test and Evaluation Process (DoD STEP), using an automated target tracking radar system as a case study of Integrated and Optimized Software Testing Process (IOSTP) framework deployment. This paper is a composite of what is in hand and within reasonable reach in the application of many science and engineering areas that can be successfully applied to software projects management to assure stable (observable and controllable) development process. Besides the integration of modeling and simulation, to form a model-based approach to the software testing process, the number of experiments, i.e. test cases, have been dramatically reduced by applying an optimized design-of-experiment plan and an orthogonal array-based robust testing methodology. Intelligence is gathered throughout the project lifecycle, from early reviews to final acceptance testing. The intelligence gathered, enables project and stakeholder management to judge how and whether progress is being made. Computer-based simulation at various abstraction levels of the system/software under test can serve as a test oracle too. Simulation-based (stochastic) experiments, combined with optimized design-of-experiment plans, in the case study, have shown a minimum productivity increase of 100 times in comparison to current practice without IOSTP deployment.

Key-words: project management, software testing, optimization, modeling, simulation, test evaluation, measurement.

1 Introduction

Applications now routinely consist of hundreds of thousands or even millions of lines of source code. In software development organizations, increased complexity of product, shortened development cycles, and higher customer expectations of quality mean that software testing has become an extremely important software engineering activity. Our research [7] concluded that developing software is for most organizations no longer an independent software project, but is part of a business case which includes all disciplines involved.

Software development activities, in every phase, are error prone so defects play a crucial role in software development. The software testing problem is complex because of the astronomical number of scenarios and states of use. The typical testing process is a human-intensive activity and, as such, it is usually unproductive, error prone and often inadequately done. To address the complex requirements of today’s embedded systems, software developers need an embedded software testing process that can be realized and managed in a systematic, planned, disciplined, controlled and incremental way according to many established software standards [3-5] that can address all aspects of a problem – whether it be multitasking, multiprocessing, real-time, or object-oriented [1,2]. In order to establish a controlled and stable (predictive) testing process with regard to time, budget and software quality, the software testing process must be modeled, measured and analyzed [8]. An integrated system of project management and control that enables a contractor and their customer to monitor the progress of a project in terms of integrated cost, schedule, and technical performance measures must be employed, as explained in [8,9]. Managers should be focused at all times on the goal of their project and the risks that threaten them. Projects need to manage goals achievement, not activities to ensure work is done according to bad plan, deliverables are delivered and they are acceptable. Early goal and risk analysis enables management to have influence over the way projects are tested and how the gathering of intelligence will be performed. Intelligence is gathered throughout the project lifecycle, from early reviews to final acceptance testing. The intelligence gathered, enables project and

---

1 This work was supported in part by the Ministry of Science and Technological Development of the Republic of Serbia under Grant No. TR-13018.
stakeholder management to judge how and whether progress is being made. It dramatically enhances the value of testing to their projects and enables them to manage their projects with more confidence. The purpose of embedded Software Quality Assurance (SQA) processes is product quality assurance. The activities are part of the development life cycle which will “build-in” the desired product quality. This focus allows identification and elimination of defects as early in the life cycle as possible, thus reducing maintenance and test costs. Embedded SQA methods include formal inspection, reviews, and testing activities. Testing is one of the most important parts of QA and the most commonly performed QA activity. Testing involves the execution of software and the observation of the program behavior or outcome [4, 8]. Testing is the key instrument for making this process happens. However, the fundamental approach as presented here focuses on testing as a fully integrated but independent activity with development that has a lifecycle all its own, and that the people, the process and the appropriate automated technology are crucial for the successful delivery of the software based system. Planning, managing, executing, and documenting testing as a key process activity during all stages of development is an incredibly difficult process. Software engineers generally agree that the cost to correct a defect increase, as the time elapsed between error injection and detection increases several times depending on defect severity and software testing process maturity level [8]. Until coding phase of software development, testing activities are mainly test planning and test case design. Computer based Modeling and Simulation (M&S) is valuable technique in Test Process planning in testing complex Software/System under test (SUT) to evaluate the interactions of large, complex systems with much hardware, user, and other interfacing software components such are Spacecraft Software, Air Traffic Control Systems. The office of the US Secretary at the Department of Defense has developed a framework [6], called the Simulation, Test and Evaluation Process (DoD STEP) to integrate M&S into the test and evaluation (T&E) process of the system/software under test (SUT). Deficient requirements from system level down to the lowest configuration component of the system are the single biggest cause of software project failure. From studying several hundred organizations, Capers Jones discovered that requirements engineering (RE) is deficient in more than 75 percent of all enterprises [17]. The application of computer-based M&S in RE activities appears to be a promising technique from the case study presented in this paper and others [16].

There is strong demand for software testing effectiveness and efficiency increases. Software/System testing effectiveness is mainly measured by percentage of defect detection and defect leakage (containment), i.e. late defect discovery. To reach ever more demanding goals for effectiveness and efficiency, software developers and testers should apply new techniques such as computer-based modeling and simulation M&S [6-12] as well as Design experiments (DOE) [8, 18-21]. The results of computer-based simulation experiments with a particular embedded software system, an automated target tracking radar system (ATTRS), are presented in our paper [8]. The aim is to raise awareness about the usefulness and importance of computer-based simulation in support of software testing.

This paper is contribution to an integrated system of project management and control that enables a contractor and their customer to monitor the progress of a project in terms of integrated cost, schedule, and technical performance measure by intelligence that Integrated and Optimized Software Testing Process framework (IOSTP) provide. In order to significantly improve software testing efficiency and effectiveness for the detection and removal of requirements and design defects in our framework of IOSTP, during 3 years of our IOSTP framework deployment to STP of embedded-software critical system such as Automated Target Tracking Radar System (ATTRS) [8, 11], we calculated overall value returned on each dollar invested i.e. ROI of 100:1.

The paper begins with an outline of Integrated and Optimized Software Testing Process framework, in section 2, which provides the “Intelligence” to Support Decision Making and state-of-the-art software testing methods implementation. The main contribution of M&S, Design of Experiments with illustrative details and experience of methods implemented in IOSTP are presented in section 3. Finally in section 4, some concluding remarks are given.

2 IOSTP framework state-of-the-art methods implementation

Unlike conventional approaches to software testing (e.g. structural and functional testing) which are applied to the software under test without an explicit optimization goal, the IOSTP with embedded Risk Based Optimized STP (RBOSTP) approach designs an optimal testing strategy to achieve an explicit optimization goal, given a priori [6, 8]. This leads to an adaptive software testing strategy. A non-adaptive software testing strategy specifies what test suite or
what next test case should be generated, e.g. random testing methods, whereas an adaptive software testing strategy specifies what testing policy should be employed next and thus, in turn, what test suite or test case should be generated next in accordance with the new testing policy to maximize test activity efficacy and efficiency subject to time-schedule and budget constraints. The use of state-of-the-art methods and tools for planning, information, management, design, cost trade-off analysis, and modeling and simulation, Six Sigma strategy significantly improves STP effectiveness. Figure 1 graphically illustrates a generic IOSTP framework [8, 9].

The focus in this paper is description of IOSTP with embedded RBOSTP Approach to Testing Services that:

- Integrate testing into the entire development process
- Implement test planning early in the life cycle via Simulation based assessment of test scenarios
- Automate testing, where practical to increase testing efficiency
- Measure and manage testing process to maximize risk reduction
- Exploit Design of Experiments techniques (optimized design plans, Orthogonal Arrays etc.)
- Apply Modeling and Simulation combined with Prototyping
- Continually improve testing process by proactive, preventive (failure mode analysis) Six Sigma model
- Continually monitor Cost-Performance Trade-Offs (Risk-based Optimization model, Economic Value and ROI driven STP)

Framework models are similar to the structural view, but their primary emphasis is on the (usually singular) coherent structure of the whole system, as opposed to concentrating on its composition. IOSTP framework model targeted specific software testing domains or problem classes described above. IOSTP is a systematic approach to product development (acquisition) which increases customer satisfaction through a timely collaboration of necessary disciplines throughout the life cycle. Successful definition and implementation of IOSTP can result in:

- Reduced Cycle Time to Deliver a Product
- Reduced System and Product Costs
- Reduced Risk
- Improved Quality

3 Modeling, Simulation, and Design of Experiments for Software Testing

As far as this paper is concerned, computer-based simulation is “the process of designing a computerized model of a system (or process) and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies for the operation of this system”[18] as depicted in figure 2. Simulation can provide insights into the designs of, for example, processes, architectures, or product lines before significant time and cost have been invested, and can be of great benefit in support of the testing process and training therein. There are several distinct purposes for computer-based simulation. One is to allow the creation of a physical object or system such as the ATTRs, as a logical entity in code. It is practical (and faster) to develop a simulator for testing physical system design changes. Changes to the physical system can then be implemented, tested, and evaluated in the simulation. This is easier, cheaper, and faster than creating many different physical engines, each with only slightly different attributes. Because of these features, network-based simulation tools allow one to develop large detailed models quite rapidly. The focus thus becomes less on the construction of a syntactically correct model and more on the model’s semantic validity and the accuracy of its numerical driver. The simulation tools in today’s market place, such as SLAM II, SIMSCRIPT, SIMAN, GPSS, PowerSim, MATLAB, etc, are robust and reasonably inexpensive. M&S is being used worldwide by the military, industry, and academia as a technological enabler to enhance training, analysis, and acquisition activities.

![Fig. 1 Integrated and optimized software testing process (IOSTP) framework [8]](image-url)
Military forces have determined that M&S can provide a realistic, and sometimes cheaper, way to train. The military acquisition community uses M&S: (1) to evaluate requirements for new systems and equipment; (2) to conduct research, development and analysis activities; (3) to develop digitized prototypes and avoid the building of costly full scale mockups; and (4) to plan for efficient production and sustainment of the new systems and equipment when employed in the field.

Industry uses M&S in much the same way. The quality of training and analysis activities can be dramatically enhanced through the effective and efficient integration of M&S capabilities. Commercial firms incorporate M&S into all phases of the development of new products, covering the entire life cycle from concept development to sustainment.

Advanced M&S may integrate a mix of computer simulations, actual war fighting systems, and weapon system simulators. The entities may be distributed geographically and connected through a high-speed network. Warriors at all levels will use M&S to challenge their military skills at tactical, operational, or strategic levels of war through the use of synthetic environments representing every potential opponent in any region of the world, with realistic interactions. Acquisition personnel may use the same synthetic environments for research, development, and test and evaluation activities [8-17].

In summary, M&S is an enabler that provides a bridge to the technology of the future. It will help you and your organization get to where you want to be in an effective and cost-efficient manner, as deployed in our IOSTP [8-15].

3.1 Model-Based Testing through Simulation
The IOSTP framework is a multi disciplinary integrated engineering solution to the testing process incorporating modeling and simulation, design of experiments, software measurement, and the Six Sigma approach to software test process quality assurance and control, as depicted in Figure 1 [8]. Its many features can be utilized within the DoD STEP approach as well. Unlike conventional approaches to software testing (e.g. structural and functional testing), which are applied to the software under test without an explicit optimization goal, the DoD STEP approach designs an optimal testing strategy to achieve an explicit optimization goal, given a priori. This leads to an adaptive software testing strategy as shown in figure 3 [6]. A non-adaptive software testing strategy specifies what test suite or what next test case should be generated, e.g. random testing methods, whereas an adaptive software testing strategy specifies what testing policy should be employed next and thus, in turn, what test suite or test case should be generated next in accordance with the new testing policy to maximize test activity efficacy and efficiency subject to time-schedule and budget constraints. The process is based on a foundation of operations research, experimental design, mathematical optimization, statistical analyses, as well as validation, verification, and accreditation techniques.
equally necessary to create a reliable model that correctly reflects the real world, and also contains all attributes necessary to make the model a useful tool for prediction. The steps of abstraction and validation are in themselves, however, not totally sufficient to create a valid and usable model. Other steps are necessary to create a model that is of sufficient detail to be useful. These steps that describe the process of producing and using a dynamic simulation are described by Pritsker [22].

One of the most important problems facing the developer of a real-world simulation is that of trying to determine whether the simulation model is an accurate representation of the actual system being studied. In M&S based systems acquisition, computer simulation is used throughout the development and deployment process, not just as an analysis tool, but also as a development, integration, test, verification and sustainment resource. In IOSTP we deployed Model-Based software testing strategy as shown in figure 4, because of well known fact: when we model SUT we test the design at the same time and vice versa, when test we get data to improve model. Because of this, verification and validation (V&V) in the simulation development process are most important tasks. If the model is to be credible, and a predictor of future behavior, it is critical that it is validated as shown in Fig. 5. [23].

The focus in this paper is on the application of M&S and DOE to minimize test suite size dramatically through black box scenario testing for the ATTRS real-time embedded software application, and also on using M&S as a test oracle in this same case study.

3.2 Design of Experiments for Software Testing

The process of software testing is typically divided into various phases: Unit testing (testing of small pieces of code written typically by one programmer), Integration testing (testing of several subsystems, each of which is comprised of many units) and System testing (testing of combination of subsystems). Besides these stages of testing, there are many different methods of testing. Structural testing or, White box testing, refers to the type of testing in which tests are designed on the basis of detailed architectural knowledge of the software under test. Functional testing, or Black Box testing, refers to the type of testing where only the knowledge of the functionality of the software is used for testing; knowledge of the detailed architectural structure, or of the procedures used in coding, is not utilized. Structural testing is typically used during unit testing, where the tester (usually the developer who created the code) knows the internal structure and tries to exercise it based on detailed knowledge of the code. Functional testing is used during integration and system test, where the emphasis is on the user’s perspective and not on the internal workings of the software. Thus, functional testing consists of subjecting the system under test to various user controlled inputs, and watching its performance and behavior. The primary focus of this paper is on functional testing. Since the number of possible inputs is typically very large, testers need to select a sample, commonly called a suite, of test cases, based on
effectiveness and adequacy. Much functional testing is done in an intuitive and less formal manner. Typically, testers, working from their knowledge of the system under test and of the prospective users, decide on a set of specific inputs. Clearly there is the possibility that important interactions among the inputs will be missed. Herein lay significant opportunities for a systematic approach, based on ideas from sampling and experimental design theory.

Consider the following example. In a stochastic simulation like in our case study of Automated Target Tracking Radar System [8] you’d really like to know all about the output variables distributions. So you usually have to settle for various summary measures of the output distributions. Traditionally, people have focused on estimating the expected value (or mean) of the output variable distribution and this can be of great interest.

For people without statistical training, it can be difficult to organize information about the system under study in a way that aids the design of the experiment. To help clarify this process, we break the design task itself into five separate steps.

1. Define the goals of the experiment.
2. Identify and classify independent and dependent variables.
3. Choose a probability model for the behavior of the simulation model.
4. Choose an experiment design plan.
5. Validate the properties of the chosen design.

After the goals definition, appropriate DOE plan should be chosen. One determines the number of distinct model settings to be run and the specific values of the factors for each of these runs. There are many strategies for selecting the number of runs and the factor settings for each run. These include random designs, optimal designs (one of them is chosen in ATTRS case study in [8]), combinatorial designs, mixture designs, sequential designs, and factorial designs. Here we want to emphasize one method of experimental design applied to software testing i.e. the Orthogonal Array-based Robust Testing [18], based on Taguchi Robust Design which has a mathematical foundation in linear algebra—specifically, the Galois field theory—began with Euler as Latin squares, which is exploited in ATTRS field testing (see [8] section 5). Black-box testing of software components and system is indispensable and requires test input data for all input parameters. The number of test cases needed for exhaustive testing i.e. for all possible combinations of input data is usually extremely large – almost always too large for allocated testing resources that are always limited. The only solution is intelligent test case generation to cut down costs and improve the quality of testing. The Orthogonal Array-Based Robust Testing method has been used to test software from many diverse industries, e.g., telecommunications, automotive, and electromechanical systems. The users have typically reported a factor of 2-to-1 or better productivity improvement compared to their traditional testing methods [18]. The number of tests needed for this method is similar to the number of tests needed for the
one-factor-at-a-time method, and with a proper software tool, the effort to generate the test plan can be small. Its ability to find faults is much better than one-factor-at-a-time method and approaches 100 percent, especially when used in conjunction with code coverage analysis tools. The test cases generated by this method have the highest effectiveness, measured in terms of the number of faults detected per test. Let us set forth a simple scenario for software testing. Consider a system under test with 4 components (factors designated as A, B, C, and D respectively) which has 3 possible elements (levels designated as 1, 2, 3):

<table>
<thead>
<tr>
<th>Test Case No.</th>
<th>Test Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

The rows correspond to test cases; the columns correspond to the test parameters. Thus, the first test case comprises Level 1 for each parameter, i.e., it represents the combination A1, B1, C1, D1. The second test case comprises combination A1, B2, C2, D2, etc. An orthogonal array (OA) has the balancing property that, for each pair of columns, all parameter-level combinations occur an equal number of times. In OA L9, there are nine parameter-level combinations for each pair of columns, and each combination occurs once.

By applying the algorithm based on Orthogonal Arrays and Covering Arrays, the overall number of test cases can be dramatically reduced compared to exhaustive testing with certainty that test coverage of all pair-wise input parameter or software components (i.e. configurations) combinations is achieved [8] such as in next example. Suppose that another SUT has 13 input parameters each with 3 input parameters domains. Then exhaustive testing would require \(3^{13}=1,594,323\) possible input parameter combinations. The OART algorithm calculates only \(N=15\) set of all pair-wise configurations that are already sufficient to cover all pair-wise input parameter interactions. The OART is applicable to unit test for test input data derivation for black-box testing methods, for integration and system scenario black-box test method, for configuration testing, for interoperability testing as well as for web testing.

The main outstanding technical problem is that of constructing efficient covering designs in a unified manner. It is not clear that combinatorial complexity theory has anything to say about how difficult this problem is. The approach we chose for this paper uses formally designed experiments, called screening designs, that are highly economical and whose primary purpose is to identify important low-order effects, i.e., first-, second-, or third order effects, where an \(n^{th}\)-order effect is an effect caused by the simultaneous interaction of \(n\) factors. For instance, for certain web server applications, a 1\(^{st}\)-order effect might be that performance slows considerably when logging is turned on and another might be that it also slows when few server threads are used. A 2\(^{nd}\) order effect involves the interaction of two options, e.g., web server performance may slow when caching is turned off and the server performs blocking reads.

Why use this technique?
Test case selection poses an interesting dilemma for the software professional. Almost everyone has heard that you can't test quality into a product that testing can only show the existence of defects and never their absence, and that exhaustive testing quickly becomes impossible -- even in small systems. However, testing is necessary. Being intelligent about which test cases you choose can make all the difference between (a) endlessly executing tests that just aren't likely to find...
bugs and don't increase your confidence in the system and (b) executing a concise, well defined set of tests that are likely to uncover most (not all) of the bugs and that give you a great deal more comfort in the quality of your software. The basic fault model that lies beneath this technique is:

- Interactions and integrations are a major source of defects.
- Most of these defects are not a result of complex interactions. Most of these defects arise from simple pair-wise interactions such as "When the font is Arial and the menus are on the right the tables don't line up properly."
- With so many possible combinations of components or settings, it is easy to miss one.
- Randomly selecting values to create all of the pair-wise combinations is bound to create inefficient test sets and test sets with random, senseless distribution of values.

OART provides a means to select a test set that:

- Guarantees testing the pair-wise combinations of all the selected variables.
- Creates an efficient and concise test set with many fewer test cases than testing all combinations of all variables.
- Creates a test set that has an even distribution of all pair-wise combinations.
- Exercises some of the complex combinations of all the variables.
- Is simpler to generate and less prone to test sets created by hand.

As an example of the benefit of using the OART technique over a test set that exhaustively tests every combination of all variables, consider a system that has four options, each of which can have three parameter combinations will find most multi-mode faults. Often, all 2-way (pairs) and 3-way parameter combinations will find most multi-mode software faults. Identifying a minimum set of tests that check each parameter interacting with every other parameter (i.e., all pairs of parameters) is often a very difficult venture if pursued in an ad hoc, non-systematic fashion. Orthogonal arrays (OAs) provide a systematic means for identifying a minimal set of highly effective tests. Unfortunately, some training in combinatorial software testing techniques available in the industry today is not very helpful in teaching this.

In our case study [8], deployed OART method testing of an Automated Target Tracking Radar System (ATTRS) is described. At the beginning of ATTRS development it was known only requirement specification for automated target tracking quality described in natural language as:

**Specification:** Maximum mean and standard deviation error \((\mu, \sigma)\) of estimated target range from radar position should be less then 100 m and for azimuth of target position should be less then 1° with confidence level \(\beta=0.95\) under conditions: target speed in range 0-300 \(\frac{m}{s}\), radial acceleration of target maneuver in range 0-6g, radar target detection probability \(P_d\geq0.8\), relative error in \((\mu, \sigma)\) estimation should be \(r\leq10\%\).

Design and test engineers worked together in order to exchange application domain knowledge. They identified that target trajectory can be modeled by combination of straight line, circle and parabolic curve. During planning phase we identified that accuracy of extractor in estimating target echo center position \((\mu, \sigma)\) high importance because extractor’s output is input to Kalman tracking filter. High importance is maneuver detector \(K\)-constant in tracking algorithm, tracking window dimensions and radar target detection probability \(P_d\). We needed information how this parameters impact to automatic target tracking quality. To test this software system, combinations of all these inputs must be provided and the output from the software system checked against the corresponding physics. Each combination tested is called a test case. One would like to generate test cases that include inputs over a broad range of permissible values. Since in this example, we have continuous variables as inputs, the total number of possible test cases is unlimited. To reduce the number
of test cases, testers have developed a number of heuristic strategies. Two guiding principles are: 1) non-redundancy (test cases are chosen so that for each test case that is included, the number of test cases which remain to be tried is reduced by at least one), and 2) generality (the outcome of the test case is generalizable beyond the specific inputs used). To implement these principles a number of concepts have been developed which can be applied in conjunction with each other. One of these relates to the notion of Equivalence Partitioning. It is assumed that the range of each of the input variables can be divided into a number of mutually exclusive classes, called equivalence classes, with the property, that the outcome of a test is generalizable to the entire equivalence class. That is, the same outcome would be expected regardless of a specific input value from that class. One can reasonably assume that a test of a representative value of each class is equivalent to a test of any other value. Since one cannot “reasonably assume” unless there is only one member in an equivalence class, in practice testers divide the input domain into a number of possibly overlapping classes (but usually with very little overlap) and select from 1 to 3 distinct inputs as representatives from each class. Typically there is much freedom in choosing the partitioning. Having formed the equivalence partitioning, one still needs to decide which members be considered as representative members. This is where another notion, that of Boundary Value Analysis is applied. This is based on the experience that test cases that explore boundary conditions have a higher payoff than test cases that do not. Boundary conditions are described as those situations directly on, or above, and beneath the edges of input equivalence classes. Thus, this concept is similar to that of a minimax strategy. Let us illustrate these concepts in context of the example we discussed above.

The system test team took the approach of developing a family of simulations, physical experiments (in laboratory and field facilities) to predict and confirm system performance using optimized test scenario based on reduction hypotheses, SRBTAC approach and design of experiment to minimize number of tests described in [15]. Because precision of automatic target tracking i.e. \( \mu, \sigma \) error of estimated target range and azimuth from radar position with confidence level \( \beta=0.95 \), is statistical in nature we estimated number of experiment replication in order to assess \( (\mu, \sigma) \) with required precision \( r \leq 10\% \). As described in [8] using well known equations from statistics we can assess lower limit \( \sigma_1 \) for \( \sigma \) with confidence level \( 1-\alpha_2 \), and higher limit \( \sigma_2 \) for \( \sigma \) with confidence level \( 1-\alpha_1 \). In our case we have to test hypothesis:

\[ H_1: \text{that } \sigma_2 \leq \sigma_{\text{max}} = \sigma_0 + r \cdot \sigma_0 \text{ where } \sigma_0 \text{ is 100m for } R \text{ and } 1^\circ \text{ for } \theta \text{ with confidence level } \beta=1-\alpha_1=0.95. \]

After calculation we plotted curve of needed number of measurement, i.e. experiment replications to test our hypothesis in figure 6. Also we assess mean error interval where \( \mu_1, \mu_2 \) are lower and higher value of assessment interval with confidence level \( \beta=1-\alpha =0.95 \). Also, in our case we have to test hypothesis:

\[ H_2: \text{that } \mu_2 \leq \mu_{\text{max}} = \mu_0 + r \cdot \mu_0, \text{ where } \mu_0 \text{ is 100m for } R \text{ and } 1^\circ \text{ for } \theta \text{ with confidence level } \beta=1-\alpha=0.95. \]

Let \( n_1 \) is required sample size for hypothesis \( H_1 \) and \( n_2 \) is required sample size for hypothesis \( H_2 \) then to estimate required sample size to test both hypothesis is \( N=\max(n_1, n_2) \). It is obvious that \( N \) depends on \( m, s \) and \( N \leq 110 \).

Result of this analysis told us that we could expect high number of experiment replication regardless we apply simulation or physical experiment. In order to minimize number of experiments we applied experimental design theory. We identified five input variables (factors) i.e. \( x_1 \) – target speed [m/s], \( x_2 \) – target detection probability [0,1], \( x_3 \) – target radial acceleration [g], \( x_4 \) – extractor error of r i.e. \( \mu_r, \sigma_r \) [m] , \( x_5 \) – extractor error of \( \theta \) i.e. \( \mu_\theta, \sigma_\theta \) [mrad], their ranges such is \( x_1 \in [0.300] \), \( x_2 \in [0.7,0.9] \), \( x_3 \in [0.6], \).
\(x_{i} \in [60,100], \ x_{i} \in [9,17] \) and variation intervals [75,0.05,1.5,10,2] respectively, five levels of variation (-2,-1,0,1,+2) and output variables. Our goal was to find out a mathematical model in the form of a second order polynomial as mathematical description of a research subject (factor influence to quality of automatic target tracking) with an adequate precision as next equation:

\[
\hat{y} = b_{0} + \sum_{i=1}^{k} b_{i} X_{i} + \sum_{i=1}^{k} b_{ij} X_{i} X_{j} + \sum_{i=1}^{k} b_{i} X_{i}^2
\]

where \(\hat{y}\) - approximation of output variable i.e. trial response, \(k\)- number of factors, \(b_{0},b_{ij}\) - regression coefficients, \(X_{i} = \frac{x_{i} - x_{i0}}{\Delta v}\) - coded \(i^{th}\) factor values, \(x_{i0}\) - real \(i^{th}\) factor values, \(x_{i0}\) - real factor value in “NULL” point (point in experimental center) and \(\Delta v\)- variation interval.

In this case we will find out requirement specification for extractor algorithm precision, \(K\)- design parameter of maneuver detector in order to minimize target tracking error with minimal number of experiments whether we apply physical or simulation trial. In order to get a mathematical model in the form of a second order polynomial with an adequate precision we applied **SECOND ORDER CENTRAL COMPOSITE ROTATABLE DESIGNS (CCRD - Box’s design)** which require \(N=32\) trials, with \(n_{0}=6\) half replica in “NULL” point, instead of \(5^{5}=3125\) trials that full factorial design plan requires. Regression coefficients in case of CCRD are calculated according to next formulas, where, \(k=5\) number of factors, \(N=32\) total number of trials, \(y_{u} - u^{th}\) trial response, \(n_{j} = 2^{k+1} = 16\), \(X_{u}\) - coded \(k^{th}\) factor’s value in \(u^{th}\) trial:

\[
b_{0} = a_{1} \sum_{u=1}^{N} y_{u} - a_{2} \sum_{i=1}^{k} \sum_{u=1}^{N} X_{i}^{2} \cdot y_{u}
\]

\[
b_{i} = a_{3} \sum_{u=1}^{N} X_{i} \cdot y_{u}
\]

\[
b_{ij} = a_{4} \sum_{i=1}^{k} \sum_{u=1}^{N} X_{i} \cdot X_{j} \cdot y_{u}
\]

\[
b_{i} = a_{5} \sum_{i=1}^{k} \sum_{u=1}^{N} X_{i}^{2} \cdot y_{u} + a_{6} \sum_{i=1}^{k} \sum_{u=1}^{N} X_{i}^{2} \cdot y_{u} - a_{7} \sum_{u=1}^{N} y_{u}
\]

and \([a_{1};a_{2};a_{3};a_{4};a_{5};a_{6};a_{7}] = [0.1591;0.0341;0.0417;0.0625;0.0312;0.0028;0.0341]\).

Lack of fit of the obtained regression model, for the case of rotatable designing with trials replicated only in design center, is checked by relations applying Fisher’s criterion i.e. calculate \(F_{R} = \frac{S_{AD}^{2}}{S_{E}^{2}}\) and find \(F_{T}\) -tabular value of Fisher’s criterion if degrees of freedom are \(f_{AD}=32:21:5=6; f_{E}=n_{0}-1=6-1=5\) and \(1-\alpha=95\%\), where:

\[
S_{AD}^{2} = \frac{\sum_{u=1}^{N} (y_{u} - \hat{y}_{u})^{2} - \sum_{i=1}^{n_{0}-6} (y_{0j} - \bar{y}_{0})^{2}}{N - (k+2)(k+1) - (n_{0}-1)}
\]

\[
S_{E}^{2} = \frac{\sum_{j=1}^{n_{0}-6} (y_{0j} - \bar{y}_{0})^{2}}{n_{0}-1}
\]

If \(F_{T}>F_{R}\), we may consider the regression equation (1) adequate. Significance of regression coefficients is checked by expressions (2).

\[
\Delta b_{0} = \pm 0.798 \cdot S_{\gamma} ; \Delta b_{i} = \pm 0.408 \cdot S_{\gamma} ; \Delta b_{ij} = \pm 0.369 \cdot S_{\gamma} ; \Delta b_{ij} = \pm 0.5 \cdot S_{\gamma} \]

The study has identified a hierarchical approach to apply these capabilities in early decision situations, M&S and measurement driven approach to continue to apply the capabilities across the software and system lifecycle. The system test team took the approach of developing a family of simulations and physical experiments to predict and confirm system performance. For every feasible experiment (live or simulation) to get above mentioned information we calculated or estimated required time, cost and importance for various test scenarios to find optimized resources allocation T&E model. We did what-if analysis and find out that optimized STP scenario, in short, was:

1. Model extractor algorithm trough simulation, based on assumption that number of reflected radar pulses fitted in \(N^{th}\) range gate has an optional distribution (the first credible option is Normal distribution, then Weibull, Uniform etc.) and satisfy target “echo end” criteria described in [8]. Same assumption is that intensity of reflected radar pulses for azimuth estimation center of target has Normal distribution because of antenna beam shape. Error of range and azimuth estimation should have Normal distribution.

2. Design simulation based experiments of target tracking algorithm for full-scale attack according to various target trajectories, speed and environment characteristics following CCRD - Box’s experimental design as planned above in this section.

3. Calibrate extractor algorithm during ATTRS integration test phase using hardware target simulator.
(Doppler transponder). Iterate M&S based on experiment results.

4. Validate and calibrate ATTRS in field test for full-scale real target attack according to various target trajectories, speed and environment characteristics applying OART to select minimal test cases. Iterate M&S based on experiment results.

The highest level simulation predicted the performance of the entire system to a full-scale attack. Some M&S and field-test results with real flights are presented in next figures.

Fig. 7 Simulation result of straight-line target attack: target speed \( v = 150 \, [\text{m/s}] \), \( \text{Pd} = 0.94 \) (Fig. a), and parabolic-curved trajectory (Fig. b): \( v = 300 \, [\text{m/s}] \), \( \text{Pd} = 1.0 \), maneuver 6g and corresponding 50 runs of Monte Carlo simulation in Fig. c) and d) with same parameters.

To facilitate the design of the simulation, the models of subsystems (extractor and Kalman tracking filter) were only as detailed as required to enable the system simulation to model overall system performance. Detailed simulations of all major subsystems validated the high-level models. In some cases, the phenomena modeled in those subsystem simulations were based on even more detailed simulations accounting for the fundamental physics involved. We determined regression coefficients of equation (1) with adequate precision i.e. Fisher’s criterion \( F_T > F_R \) was satisfied. Also, we determined that all included factors and their 2-factor interactions have statistically significant influence on tracking precision and higher number (3 to 5) of factor interactions are not statistically significant. Applying computer-based simulation experiments, i.e. the DoD STEP
methodology, aircraft-flying time in open air was maximally reduced, on average 20 times (recall the required sample size from Figure 6 and real flight results presented in [8] (Table 2 from ATTRS field-testing, i.e. experiment replications for confidence level $\beta = 0.95$). Applied M&S avoided some difficulties in field tests such as controlling the physical constraints involved, i.e. target detection probability etc. In addition, efficient plans for field-testing of the system were made. Extensive simulations were run prior to field tests. This approach predicted system performance, found software errors, and eliminated surprises and error leakage to the field-test reducing bug-fixing expenses. Before any test, multiple simulations were run (Monte-Carlo simulations) to obtain a statistical distribution of predicted performance. Since there were many variables in each test, the precise results could only be predicted statistically. Once the test was conducted and the real data were obtained, the simulation was rerun using measured data. If differences were noted then the models were recalibrated.

![Image](image1)

**Fig. 8** Some field trial results with same conditions as in simulation cases shown on Fig. 7

### 4 Conclusions
In order to significantly improve software testing efficiency and effectiveness for the detection and removal of requirements and design defects, model-based testing activities using simulation have been advocated. In particular, using an automated target tracking radar system as a case study, some original solutions have been presented as to how modeling and simulation can be efficiently and effectively implemented in the US Department of Defense software testing framework known as the Simulation, Test and Evaluation Process (DoD STEP). By combining the process with optimized design of experiments and the orthogonal array-based robust testing methodology, the number of experiments, i.e. test cases, was dramatically reduced. The number of test cases needed for exhaustive testing, i.e. for all possible combinations of input data are usually extremely large – almost always too large for the allocated testing resources that are always limited. The only solution is intelligent test case generation to cut down costs and improve the quality of testing. To minimize the number of test cases an optimized DOE plan (Box’s CCRD design) was applied which required $N = 32$ trials, with $n_0 = 6$ half replicas at the “NULL” point, instead of the $5^5 = 3125$ trials that a full factorial design plan would have required, to determine a model in the form of a second order polynomial as a mathematical description of the research subject (quality of automatic target tracking) with adequate precision. Also, by applying the algorithm based on orthogonal arrays and covering arrays, i.e. the OART method, the overall number of test cases can be dramatically reduced compared with exhaustive testing, yet with certainty that test coverage of all pair-wise input parameter or software component combinations is achieved, such as in the example of the field trial results of ATTRS.

### References:


