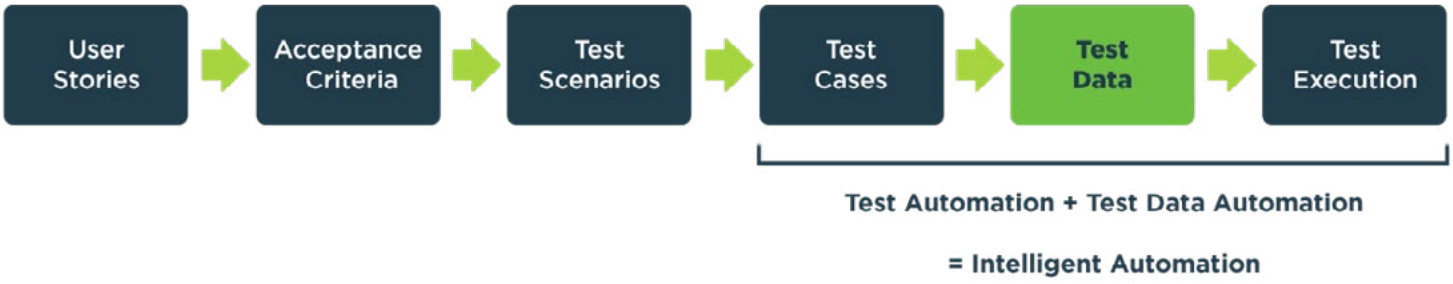
HOW TO USE SYNTHETIC DATA TO MAXIMIZE TEST COVERAGE

There is growing interest among quality assurance professionals in the use of **synthetic data generation** for software testing. Their interest in synthetic data is usually triggered by a requirement for data privacy or the need for accelerated data provisioning for Agile and DevOps. However, the most compelling reason to augment the use of production data, or manually created spreadsheet data, with synthetic data is to have total control over the variety of data needed to maximize test coverage.

Properly designed synthetic test data can increase coverage from under 50% to over 90% when provisioned by an advanced Test Data Automation system.

Using a platform like GenRocket's Test Data Automation (TDA) solution, testers are able to design comprehensive synthetic test data containing all of the possible variations, combinations and permutations needed for positive and negative test cases to fully exercise their code. With TDA, more exhaustive testing can be performed to uncover the defects missed by less sophisticated testing before they escape to a production environment.

And because GenRocket's TDA is an automated self-service solution, it can deliver any volume or variety of test data at the speed of Agile. For more information about using GenRocket for Agile testing, read Test Data Design Should be Integral to Agile and DevOps.



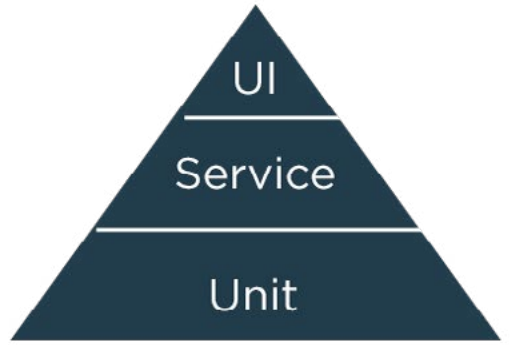
Synthetic data should really be thought of as intelligent test data.

GenRocket's TDA technology can generate synthetic data that follows rules and conforms to business logic. It can generate real-time synthetic data in response to changes in state during complex workflows. It can compute results during program execution and generate test data used to verify the expected outcomes of a functional test.

Using GenRocket's TDA platform, synthetic data can be blended with queried production data to accurately simulate real-world healthcare transactions or replicate financial data flows to assist in bank fraud detection. Synthetic data can be designed to train AI and machine learning systems or scaled to generate billions of rows of structured and unstructured data for testing Big Data applications.

FOLLOWING THE GENROCKET METHODOLOGY

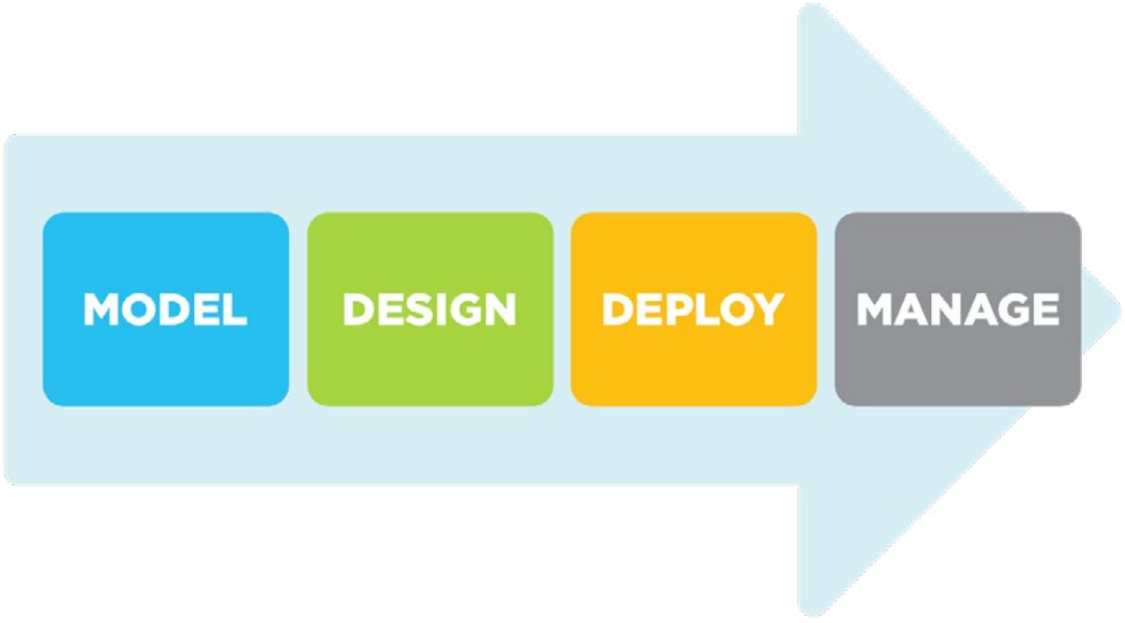
GenRocket has developed a structured methodology to guide testers through a four-step process for designing and generating real-time synthetic test data. It's part of a flexible and adaptive self-service approach that supports functional, non-functional and regression testing at all levels of the *Agile Software Test Pyramid.*



SOFTWARE TEST PYRAMID

	Functional	Non-Functional	Regression
UI Level	System Testing	Acceptance Testing	Regression Testing
Blend Production Data With Rules-Based Synthetic Data to Test Complex Workflows			
Service Level	API and Integration Testing	Performance and Security Testing	Regression Testing
Test API's and Complex Data Feeds using any Format, Variety or Volume of Data			
Unit Level	Unit Testing (Functional)	Unit Testing (Non-Functional)	Regression Testing
Rapid On-Demand Provisioning of Test Data by QA or Developers for Unit Testing			

Whether synthetic data is needed for testing at the Unit, Service or UI level, the *GenRocket Methodology* helps testers follow a consistent process for generating the most effective test data in the most efficient manner.



GENROCKET METHODOLOGY

The *GenRocket Methodology* starts with defining the data **MODEL** to be used for synthetic data generation. GenRocket self-service modules allow the tester to import a database schema and quickly validate the parent-child-sibling relationships for the data tables used by test cases. Alternatively, data can be modeled by uploading a DDL file to define the structure or stream of synthetic data to be generated.

Once the data is modeled, the tester is ready to **DESIGN** the exact data needed for testing. This article provides an expanded explanation of the DESIGN stage of the *GenRocket Methodology* in the paragraphs below.

After the design stage, the tester is ready to **DEPLOY** the test data by creating Test Data Scenarios and Test Data Cases that contain instructions for the volume and variety of data to be generated by the *GenRocket Engine*. GenRocket's 4th key component, the Receiver, is used to receive generated test data and deploy it into the test environment. More than 70 different Receivers are available to meet just about any deployment requirement.

The fourth stage of the *GenRocket Methodology* is when organization admins and testers **MANAGE** all of their test data projects. The G-Analytics dashboard provides deep insights into GenRocket use. Team Permissions provides granular control over access and permissions. And advanced GenRocket features intelligently automate common tasks so that testers can quickly organize, copy, and share their test data projects with other testers.

DESIGNING TEST DATA FOR MAXIMUM COVERAGE

Now let's take a closer look at the DESIGN stage of the GenRocket Methodology. In this stage, testers determine what kind of test data they need to fully meet the objectives of their test plan. They create the ideal test data design that will maximize test coverage.

For most QA professionals, this is a new and different concept. Conventional thinking has taught us that test data is either in the form of a production data subset or an Excel file with columns and rows.

This paradigm can be hard to unlearn.

For many, the realization that test data can be DESIGNED during an Agile sprint, integrated with a test case and then generated on-demand by a test script is an *Aha Moment.* However, once this light bulb is turned on, the paradigm begins to shift from traditional *Test Data Management* to modern *Test Data Automaton.*

UNDERSTANDING THE CHARACTERISTICS OF TEST DATA

As a foundation, let's begin our discussion of *Test Data Design* with an understanding of potential test data qualities and characteristics. Ask yourself, what data characteristics do you need for a given test? Here are some items to consider.

- **Controlled:** When we use the term controlled to describe test data, we are talking about the ability to specify its patterns, permutations, combinations, and boundary conditions – for both positive testing and negative testing scenarios.
- **Realistic:** This refers to synthetic test data that looks and acts just like real-world production data. Customer names should represent appropriate countries or ethnicities. Data fields like calendar dates, credit card account numbers and social security numbers should be represented in a valid format and have authentic values.
- **Accurate**: Sometimes the data used for testing has to accurately represent a value that is central to a test case. In healthcare for example, accurate medical procedure codes are necessary to test the approval or denial of benefits for a claims processing system.
- **Secure**: When testing software in regulated industries like healthcare and financial services, it's often necessary to replace sensitive patient or customer information with synthetic test data to ensure its security and privacy.
- **Stateful**: When testing workflows, data may need to reflect state changes. If after 3 login attempts an account is locked, the code should not allow a 4th attempt. Rules can be applied to generate the appropriate test data based on the current workflow state.
- **Scalable**: Test data should scale to billions of rows to effectively simulate real-world traffic conditions and transaction loads in order to properly stress-test an application for meeting performance goals and to understand its behavior under extreme conditions.
- **Unique**: When testing new account opening workflows, many new and unique accounts are needed; accounts already in production can't be used for this testing.

Depending on the level of integration for the software under test, a combination of these data characteristics may be required for a given test procedure. With GenRocket's TDA platform, a tester has the ability to generate synthetic data having any or all of these qualities and characteristics.

SOLVING THE TEST DATA CHALLENGE

Defining the test data challenge for a given test case depends on the nature of the test. Functional testing is a broad category that may have specific data requirements for *Unit Testing*, additional requirements for *Integration Testing*, and highly complex requirements for end-to-end *System Testing*.

Non-functional testing can also have simple or complex data requirements for different levels of security testing, compatibility testing or performance testing. Regression testing can reuse many of these functional and non-functional tests. Remember it's important for test data to be returned to its original state during regression testing so identical testing is used to ensure new code works well with previously tested software. With GenRocket, there is no need to worry about data refresh as synthetic test data is generated from scratch during each test procedure.

As you design your test case, specify the test data needed to fully exercise the code with data-driven testing. Document your test *data challenge* as stories or epics. Then translate them into *Test Data Scenarios* and *Test Data Cases* during the DEPLOY stage of the *GenRocket Methodology*.

Here's a checklist of the more prominent **Test Data Challenges** encountered when designing test data and a resource link to learn more about how GenRocket solves each one.

- **Patterns:** Synthetic data can be generated as patterned data. There are a number of patterns that may be appropriate for your test case such as ordered, sequenced, and random data. View this solutions video to learn more about patterned data.
- **Permutations**: Generate permutations to test all combinations of data for a given set of values. This allows exhaustive testing to be performed on the code. GenRocket provides a data permutation generator that makes overcoming this challenge easy to achieve.
- **Combinatorial**: Combinatorial testing can be a difficult challenge when code contains multiple inputs, each with multiple values, resulting in a combinatorial explosion. GenRocket's Pairwise solution can help solve this tough test data challenge.
- **Boundary**: Conducting boundary value analysis requires test data at or near the edge case values for a given data element. GenRocket's edge case generator provides the tester with a comprehensive solution for meeting this challenge.
- **Rules-Based**: To properly test the business logic in a given application, test data should conform to that logic and allow data-driven testing of positive and negative scenarios. Use GenRocket's Test Data Rules self-service module to generate rules-based test data.
- **Queries**: When test data must accurately reflect key values such as a customer ID number or a credit card prefix, a production database can be queried, and the results blended with synthetic data to achieve accuracy with control. GenRocket's Test Data Queries self-service module enabling a tester to meet this test data challenge.
- **Masked**: Instead of simply masking sensitive data for testing purposes, GenRocket allows testers to intelligently replace sensitive production data with controlled synthetic data to ensure privacy while providing control over the data needed for testing. This solutions video shows the value of GenRocket's synthetic data replacement capability.

- **Dynamic**: When testing complex workflows, test data must be dynamic and stateful so that appropriate data is used for changing workflow stages and conditions. View the solutions video Testing a Bank ATM Web Services Workflow to learn how GenRocket's intelligent data generators and flexible API make dynamic test data possible.
- **Formats**: Formatting test data can be as simple as configuring a SQL or XML output format or as complicated as building a complex, nested EDI X12 transaction set for testing health insurance claims processing. There are more than 260 formatting options for testers to choose from in the GenRocket library of supported output formats.
- **Volume**: Generating a massive amount of test data is an easy challenge to solve with GenRocket. Operating at 10,000 rows per second, the platform is able to meet the synthetic data generation needs of large data lakes and warehouses and the GenRocket Partition Engine allows performance to scale to meet virtually any volume requirement.

All of these test data challenges can be solved using the GenRocket platform. It's simply a matter of following the *GenRocket Methodology* step-by-step and provisioning the precise data needed for your test plan.

Remember these 4 steps: First **MODEL** your data environment. Then **DESIGN** your test data to meet the specific test data challenges presented by your test case. Then use the GenRocket self-service modules to **DEPLOY** your test data design using *Test Data Scenarios* and *Test Data Cases* integrated with your test automation tools. Finally **MANAGE** your TDA environment to flexibly share, adapt and apply your test data designs to all levels of Agile testing.

If you would like to discuss your specific test data challenges, request a live demonstration of the GenRocket platform with one of our *Test Data Automation* experts.

REQUEST A LIVE DEMO