U.S. Department of Transportation
Office of the Assistant Secretary for Research and Technology
Volpe Center

Voices
Virtual Open Innovation Collaborative Environment for Safety

# VOICES Concept of Operations and Capabilities

## 1. Introduction

The purpose of this document is to provide the system goals and concept of operations for the Virtual Open Innovation Collaborative Environment for Safety (VOICES) platform. VOICES enables effective and efficient testing in a safe and secure environment to accelerate the deployment of cooperative driving automation (CDA) systems on U.S. roadways. This novel platform intends to reduce the development time and improve the safety and reliability of CDA systems, and provides a trusted testing and validation environment that is capable of examining such complex systems.

VOICES allows for different levels of testing, including:

- Subsystem level testing to ensure that software modules are working per design and the data flow between the different software modules is working properly.
- System level testing to verify that the system functions per design requirements and to validate that the system meets the operational needs of stakeholders.
- System to system interoperability testing to validate that the system exchanges data correctly with surrounding vehicles, nearby infrastructure, and the Cloud.

VOICES is a traffic scenario based system that integrates a set of  tools chain. With the expected built-in capabilities, such as emulating and simulating infrastructure; integrating live data from real devices; integrating a set of driving; traffic and vehicle simulators; replaying collected real data; etc., complex driving scenarios may be constructed and diversified using VOICES. Generating such a set of driving scenarios enhances the performance of the system to address a larger set of real-world scenarios.

VOICES is capable of hosting and integrating different testing environments to enable system validation under various set of scenarios. For example, one can connect a real device such as traffic signal controller to a traffic simulator in order to control signal phase and timing information at an intersection. In parallel, one can have a set of simulators to simulate pedestrian walk, vehicle motion, and traffic control devices at an intersection. VOICES also allows for blended live, virtual, and constructive (LVC) testing that links live platforms with manned simulators in virtual environments that can add constructive tools.[1]

VOICES facilitates a distributed synthetic test environment (DSTE) that connects various resources (within a test facility and among different remote test sites) to enable the interaction of concurrent tests in synchronous or asynchronous mode. As a result, developers from different entities can use VOICES to collaborate in creating a joint test bed environment, performing cooperative testing, and validating system-to-system interoperability.

VOICES is a scalable test platform with a built-in database of traffic scenarios and transportation systems management and operations use cases. As the testing grows in size and complexity, VOICES becomes more valuable to the various stakeholders categories. It is envisioned that over the time, the

[1] https://en.wikipedia.org/wiki/Live,_virtual,_and_constructive

collaborated efforts among different users will lead to a comprehensive and valuable set of test scenarios to both developers and evaluators. These test scenarios will cover both typical and edge cases.

## 2. Concept of Operations

This document describes the concept of operations for VOICES by illustrating test examples of two use cases or scenarios (tester/evaluator's perspective). To provide a context for this description, this document first provides an overview of the main blocks of an automated driving system (ADS), followed by general ADS testing needs. CDA enables communication and cooperation between properly equipped vehicles, ADS-equipped vehicles, infrastructure, and other road users. Prior to detailing the test examples, this document presents an overview of VOICES and its basic architecture.

### 2.1 Overview of an Automated Driving System

Figure 1 shows an ADS overview that consists of the following five basic modules:

1. Sensing module - Senses the surrounding environment of the host vehicle (HV). It uses a combination of various sensors such as camera, radar, lidar, global navigation satellite system (GNSS), and vehicle-to-everything (V2X) in order to perform this functionality with adequate accuracy. Examples of sensing subsystem outputs are images from a camera, point cloud from a lidar, signal returns/object file from radar, GNSS position, and basic safety message (BSM), signal phase and timing (SPaT), traveler information message (TIM), and personal safety message (PSM) from the V2X subsystem.
2. Perception module - Detects, tracks, classifies, and characterizes all objects of interest that are sensed by the sensing module. Objects of interest include, but not limited to, vehicles, pedestrians, cyclists, traffic signs, traffic lights, and road attributes. Examples of perception module output are objects map list with detailed characterization (e.g., object 1 classification, down range, longitudinal velocity, lateral velocity cross range, width, lane assignment, etc.).
3. Planning module - Establishes a scene interpretation based on the object map received from the perception module, and consequently decides on the desired HV motion profile with respect to the road as function of time. It outputs a well-defined trajectory data.
4. Control module - Uses a set of control algorithms and HV dynamic/engine model to calculate the required steering and throttle/braking in order to achieve the trajectory specified by the planning module.
5. Actuators enable drive-by-wire systems, which aim to achieve their commanded inputs received from the control module.

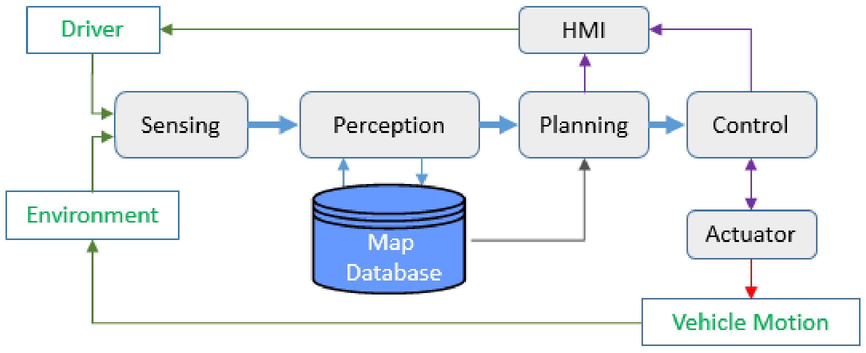

Figure 1. ADS Overview

## 2.2 ADS Testing

To test an ADS, one needs to check if the system is capturing the objects of interest on the road with adequate accuracy, and the HV is executing the correct behavior (motion profile or action) based on the traffic scenario and environment conditions. Therefore, there are different options for testing an ADS in a simulation environment:

1. Test the whole ADS as black box. Using this test methodology makes it hard to analyze the source of system failure or degraded performance.
2. Perform separate tests for ADS perception and ADS behavior:
    a. Testing ADS perception in a full simulation environment is not a trivial task due to the effort required to build the sensor modules with high fidelity and to represent enough scenes that are practically close to real-world scenes during driving. Therefore, perception testing is typically performed via bench tests using real data captured during driving for millions of miles on different road classes, road conditions, lighting conditions, weather conditions, etc.
    b. Testing ADS behavior can be fully executed using a simulation environment by bypassing the perception module and injecting simulated object maps into it.
3. Test the integrated ADS by providing separate libraries for each of the subsystems (library for perception and library for behavior). This way, both modules can be debugged when testing the integrated ADS. To overcome the difficulty of simulating the sensing module, captured real-world data are fed to the perception module. By this, perception can be tested, and at a minimum, behavior decision can also be tested.

Based on above discussion, the ADS simulation can be envisioned as shown in Figure 2:

1. The Scenario Database (library) can be built from sensor data collected during driving vehicles on real roads, collision database, pre-simulated data, and basic unavoidable scenarios found in some of the standards and literature.
2. Objectives of Test Case Formation are to:
    a. Perturb an existing scenario to test the sensitivity of the ADS to small changes in the area where the designer is focusing on some of the ADS weaknesses.

   b. Create fresh new scenario or construct scenario by combining the data of two scenarios or using part of the data of specific scenario.
3. The Simulation Specification function is to set up the simulation by parameterizing the object behavior and environment. Examples include pedestrians, other vehicles, road layout, weather conditions, initial positions, etc.
4. The Host Vehicle block contains the sensor models that the HV is using, the HV dynamic model, and the ADS proprietary model that is similar to the one shown in Figure 1.
5. The LVC Simulator can combine LVC environments to form the simulation scenario and execute it. It uses the simulation setup data along with the behaviors data of the HV's ADS to execute the simulation scenario while continuously outputting the simulated environment data to the ADS proprietary model.
6. The ADS proprietary model establishes a scene interpretation based on the simulated data from the LVC Simulator, determines the behavior, and feeds this behavior in the form of planned trajectory to the LVC simulator.

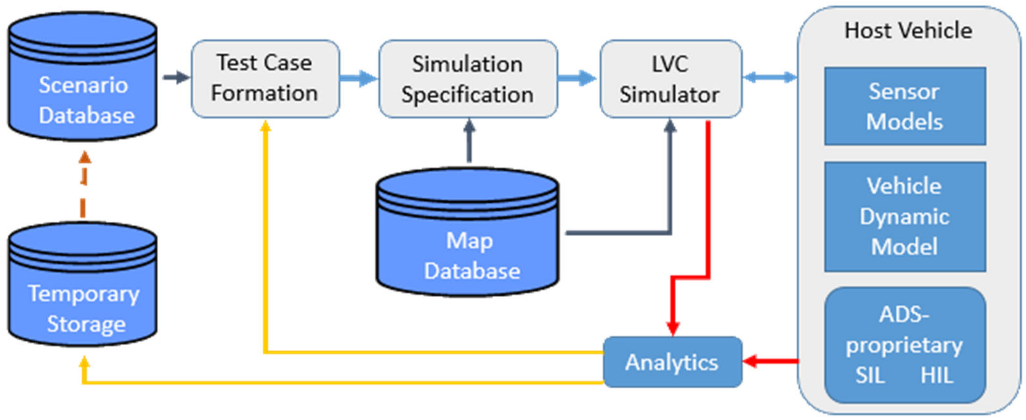

Figure 2. ADS Simulation Structure

Figure 3 illustrates a multiple users' simulation system, in which all users can have access to the simulator capabilities and features at the same time.

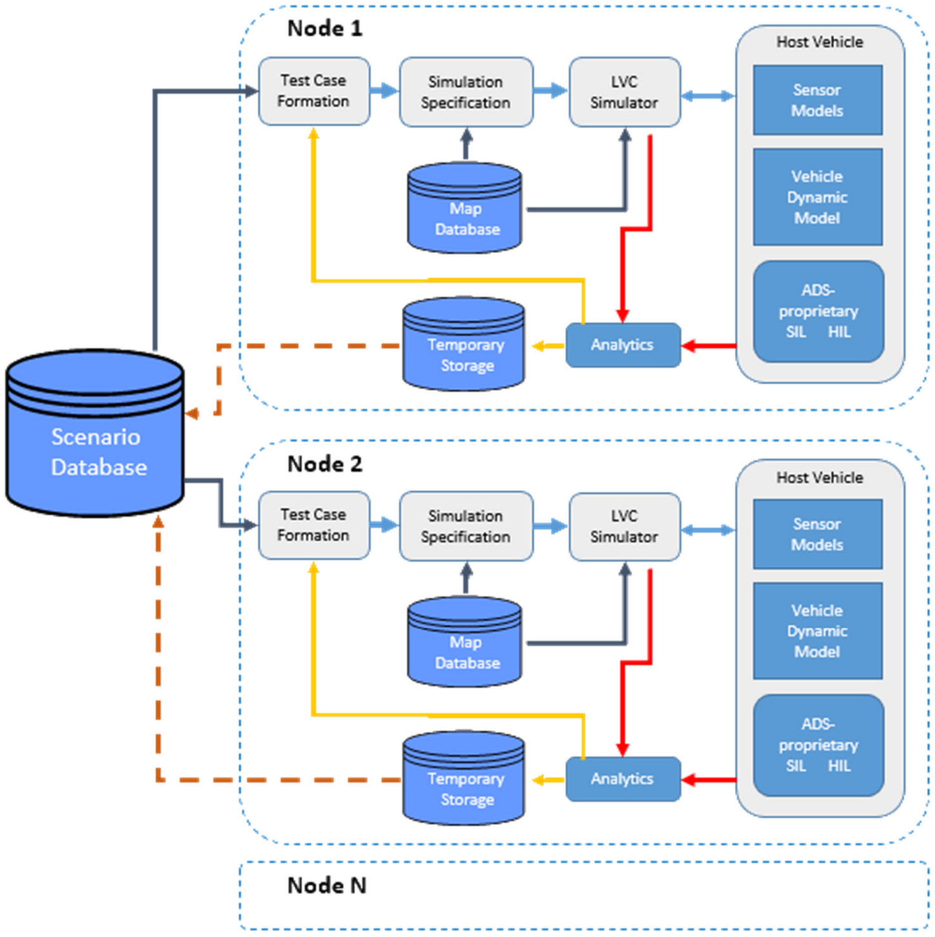

Figure 3. Multi-Users on Simulation System

## 2.3 VOICES Overview

The structure of VOICES platform is similar to the one illustrated in Figure 3. VOICES adopts the Department of Defense Test and Training Enabling Architecture (TENA) to facilitate the execution of a simulation system that integrates multiple simulation platforms with different semantics, and regulation of complex interactions between diverse simulation models. The core of TENA is its Common Infrastructure, including:

- TENA Repository that contains all the relevant TENA information such as the software library containing the TENA Middleware, standardized resource interface definitions, and executable versions of the TENA utilities and tools, along with related documentation.
- TENA Object Models that define the common data and interfaces shared by the integrated applications, and enable semantic interoperability among resource applications by encoding all the information that needs to be communicated among those applications.
- TENA tools, utilities, and gateways that assist the user in creating and managing an integration of different resources, as well as in optimizing the TENA Common Infrastructure. The core set of

tools is a suite of key software applications for planning, configuring, controlling, monitoring, and analyzing synthetic and test results.

TENA, through its data archive, stores and provides for the retrieval of test results and other injected data.

## 2.4 VOICES Examples

The following two examples provide a background or context to help understand VOICES system goals and concept of operations.

### 2.4.1 Example 1: CACC Testing

Cooperative Adaptive Cruise Control (CACC) system is an enhancement to the Adaptive Cruise Control (ACC) system by the addition of wireless communication with preceding vehicles and/or the infrastructure to augment the ACC active sensing capability. It uses active sensing data such as ranging to forward vehicle, HV data, over-the-air data from other surrounding vehicles and from infrastructure, and driver input to longitudinally control the vehicle via throttle and brake controls (see Figure 4). With CACC system, vehicles are able to maintain shorter gap between each other without compromising safety.

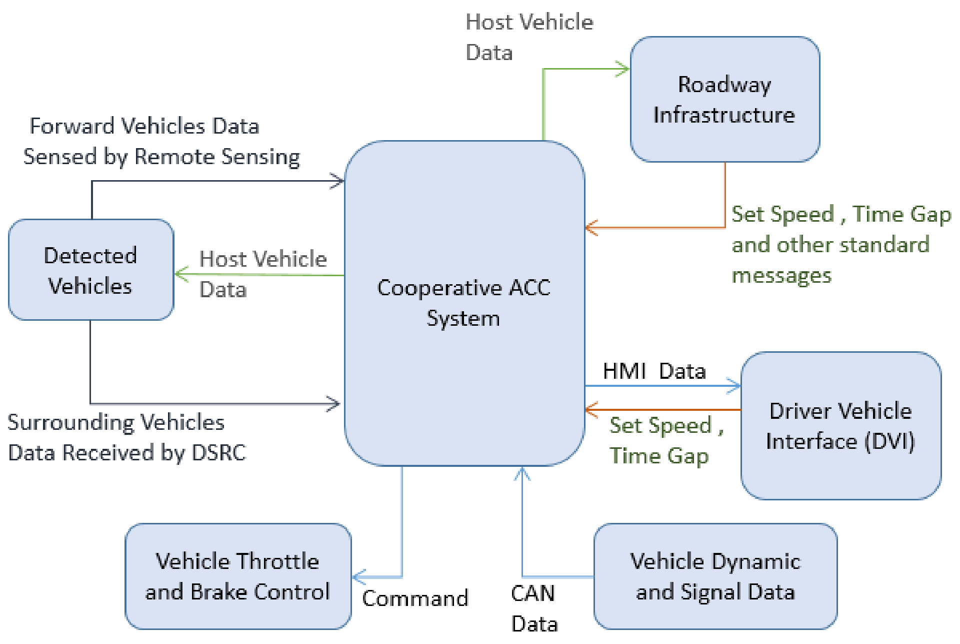
Host Vehicle Data
Roadway Infrastructure
Forward Vehicles Data Sensed by Remote Sensing
Set Speed , Time Gap and other standard messages
Detected Vehicles
Host Vehicle Data
Cooperative ACC System
HMI Data
Surrounding Vehicles Data Received by DSRC
Set Speed , Time Gap
Driver Vehicle Interface (DVI)
Vehicle Throttle and Brake Control
Command
CAN Data
Vehicle Dynamic and Signal Data

Figure 4. CACC Depiction Diagram

This example illustrates how live and constructive environments can be integrated together using the VOICES platform. In CACC, the HV is supposed to follow the lead vehicle (LV) while maintaining a short gap. For the purpose of the experiment illustration, let us assume the following:

1. HV's control module for throttle and brake is integrated in the VOICES platform, located at some location X (constructive).

2. LV is driven by a human driver on a test track located at some location Y (live).

The tester would like to evaluate the performance of the control algorithm for different LV driving behaviors. The idea is to test proprietary controller integrated in the VOICES environment using the live data originating from driving the LV at the test track. To accomplish this experimental task, the following will be required to happen, as depicted by the experiment functional architecture in Figure 5:

1. LV is equipped with an on-board unit (OBU) that broadcasts BSMs using a short-range communication device.
2. Single or multiple roadside units (RSUs) are installed at the test track, which receive the BSMs and relay them to the VOICES network.
3. The VOICES platform needs to recover and decode these BSMs, and send all or their part of their contents to the simulation environment per the implemented Application Programming Interface (API).
4. The 'Target and Sensing Simulator' integrated in the VOICES environment receives the data, runs its simulation, and sends the following to the control module using the implemented API: LV data extracted from BSMs and HV initial position, speed and orientation. Examples of the final LV data are down range, cross range, velocity, heading, accuracy measure, etc.
5. The 'Control Module' integrated in the VOICES environment receives the data from the 'Target and Sensing Simulator' and calculates the required acceleration command. The acceleration data or its equivalent of brake or throttle is sent to a 'HV Engine/Dynamic Simulator'. The function of this simulator is to receive the acceleration command from the controller (Control Module) and generate an updated state of the HV: updated acceleration, updated velocity, updated position, updated orientation, etc.
6. This data is then passed to the 'Target and Sensing simulator', and then Steps 4 and 5 and this step are repeated. Also, the 'HV Engine/Dynamic Simulator' may feed the updated vehicle state data back to the 'Control Module'.

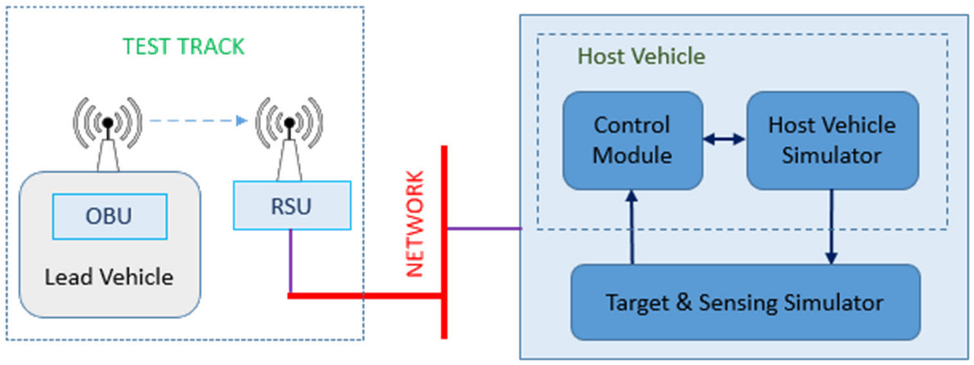

Figure 5. Functional Architecture of Live-Constructive CACC Testbed

This example demonstrates the integration of multiple simulator platforms, inclusion of software in the loop (SIL) for the propriety controller, and integration of live and constructive environments to achieve the experiment goal.

This experiment scope can be expanded to test human drivers in the loop using a driving simulator, similar to the Highway Driving Simulator (HYSIM) at Turner-Fairbank Highway Research Center.[2] Test subjects will be able to ride in the driving simulator and evaluate some human factor parameters that are directly impacted by the control behavior (e.g., gap suitability, gap dynamics, jerk). In this test setup, the driving simulator will be at a different location and is mainly driven by the control module output through closed-loop feedback control. Figure 6 explains this test setup. By having the driving simulator in the simulation loop, VOICES integrates virtual environment along with the live and constructive environments introduced earlier.

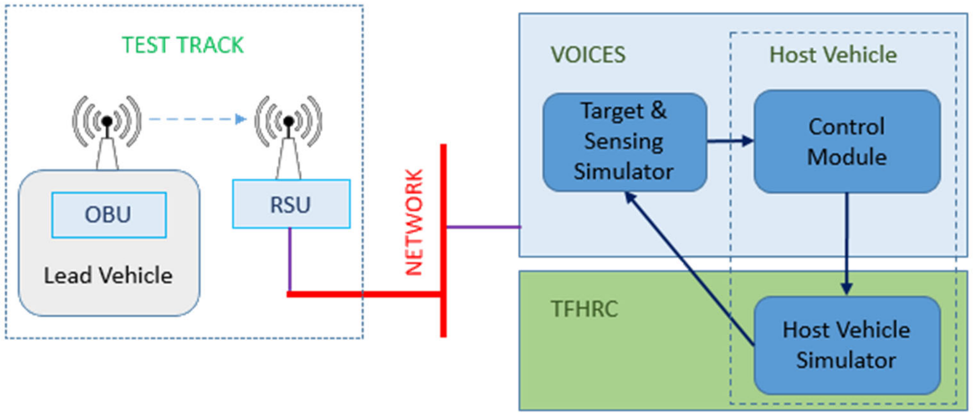

Figure 6. Driving Simulator in the Loop but at Different Site

### 2.4.2 Example 2: Cooperative Perception, ADS Testing, Platooning Testing: ADS Vehicle Turning Right at Intersection Conflicting with Pedestrian Crossing

This example attempts to highlight and illustrate few aspects of VOICES operation:

1. Role of TENA
2. Distributed and collaborative VOICES test environment
3. Integration of a combined live and constructive environment
4. Integration of different simulation tools
5. Access of scenario database
6. Integration of different user proprietary modules (SIL)
7. Integration of CARMA Streets module into VOICES[3]
8. VOICES replay capabilities

Figure 7 illustrates the scenario of interest and the 4-way intersection equipped with:

1. Traffic controller controlling signal heads to regulate the approaching traffic on lane level

[2] https://www.fhwa.dot.gov/publications/publicroads/94winter/p94wi19.cfm

[3] CARMA Streets is a downloadable, infrastructure-based application supporting CDA participation of the infrastructure and interactions with other participants to engage in CDA. The platform enables communication between infrastructure and various modes, vehicles, and road users to optimize transportation system management, operations, and safety. This new capability will enable the use of edge computing to support such activities as cooperative perception and future signal control strategies, such as vehicle reservations.

2. Camera systems monitoring the four legs of the intersection
3. CARMA Streets platform detecting and tracking pedestrians on crosswalks
4. RSU broadcasting some of the SAE J2735 messages such as SPaT, MAP and PSM (used for pedestrians).

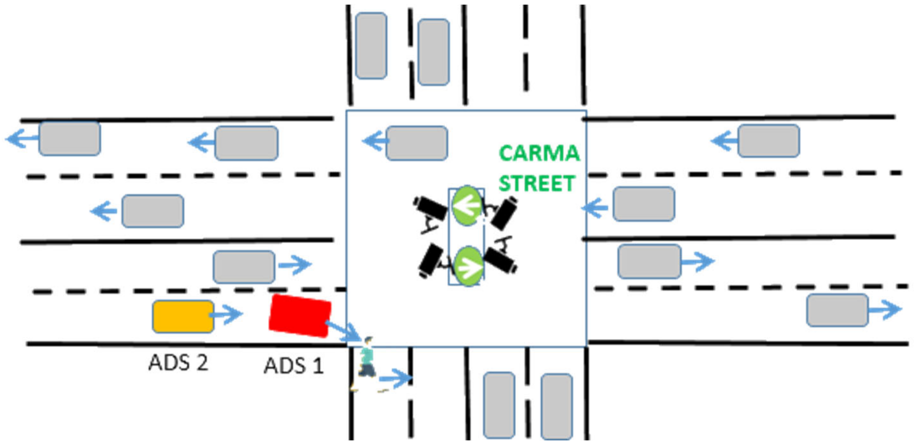

Figure 7. ADS Vehicle Conflicting with Pedestrian Crossing while Turning Right at Intersection

Within the traffic environment of a 4-way intersection, two ADS vehicles (ADS1 and ADS2) are driving on the right most lane of one of the approaches. ADS1 is the closest in-lane vehicle to ADS2. The signal head controlling this lane is green for the right and the straight movement status. At the time when ADS1 is about to start turning right at the intersection, a pedestrian starts to move on the crosswalk in the direction shown Figure 7. Due to lack of line of sight between the ADS vehicles and the pedestrian, neither ADS1 nor ADS2 is able to detect the pedestrian using their integrated onboard sensors. On the other hand, CARMA Streets detects and tracks the pedestrian and sends the pedestrian data such as position, heading, and speed to the RSU and to the traffic controller. As a result, two events happen in response to this pedestrian information:

1. Traffic controller enables the 'Pedestrian Bit' as if the pedestrian is pushing the pedestrian button on the pole holding the signal head. The traffic controller then updates its SPaT signal to set the 'Pedestrian Bit'.
2. RSU constructs the PSM message and broadcasts it along with the SPaT and MAP messages.

This setup is mounted on some busy urban intersection. In this test example, it represents the live element of the simulation environment by providing live pedestrian, SPAT, and MAP data in the J2735 format. The map is the local map of the intersection as described in J2735. If ADS1 and ADS2 are driving live on the intersection, they will receive the broadcasted PSM, SPaT, and MAP messages, and perform lane-level matching to determine which SPaT signal corresponds to the lane they occupy. If it happens that a conflict with a pedestrian is predicted and imminent, ADS1 is expected to brake to stop to avoid running over the pedestrian. ADS2 is also expected to develop a response to emergency braking by ADS1 and to the received PSM. Based on the distance between both vehicles, ADS2 will automatically apply the appropriate level of braking.

This scenario is very hard to test for in a real environment. The probability of having this kind of driving conflict for one specific vehicle with a pedestrian is very low. Therefore, the simulation environment in the test setup becomes a powerful mechanism to test this important safety scenario. The simulation environment can keep the ADS vehicles *repeating turning right* on the intersection for *the whole duration of live data*. Also, a better scenario control can be performed by recording the live data and then replaying it. This way, the tester can have more control on the test initial conditions especially the initial position and speed of the ADS vehicles.

Implementation of this crucial scenario in simulation requires the following:

1. Objects for vehicles, pedestrian, traffic light, and road layout and the needed motion trajectory along with sensor data.
    a. CARLA simulator can create the object specifications and sensor data, modify weather and road conditions but not suitable for lane-level SPaT message.
    b. SUMO simulator is used to simulate the SPaT message.
    c. ROADRUNNER is suitable for road layout and pedestrian walk; therefore, it is used for map and pedestrian simulation.
    d. PTV Vissim is known for driving simulation and it can provide the needed traffic simulation.
    e. CARSIM is known for vehicle dynamic and engine simulation; therefore, it is used for the ADS vehicles simulation.
2. An architecture (in this case TENA) to:
    a. Integrate the different types of simulators. Integration is done by two important features:
        i. Adaptor implementations to enable passing messages around between the different simulators
        ii. Scenario Manager to orchestrate the simulation execution according to the simulation setup/configuration inputs.
    b. Integrate the live data into the simulation.
    c. Integrate the proprietary SIL and hardware in the loop (HIL), by building and executing the proper API.
    d. Access the scenario database without significant delay.
    e. Store the data of interest.
    f. Visualize the data of interest.
    g. Replay the data of interest.

Figure 8 represents the functional architectural diagram of VOICES to execute this test. In this diagram, TENA is receiving the data from different simulators and live input through the designed and implemented adaptors. The adaptors translate the data to TENA format. As a result, this data will be sent to all applications that require particular data via Publish/Subscribe methodology. TENA finalizes its execution manager based on the simulation specification received from the simulation setup module. The SIL/HIL adaptor will combine the needed data of and feed it into to the proprietary SIL/HIL through the designed API as appropriate. The proprietary module calculates the behavior of the ADS vehicle and sends it back to TENA through the API. Simulation data can be:

1. Passed to analytic module for performance analysis.
2. Stored for future replay and/or for future move to the scenario database.

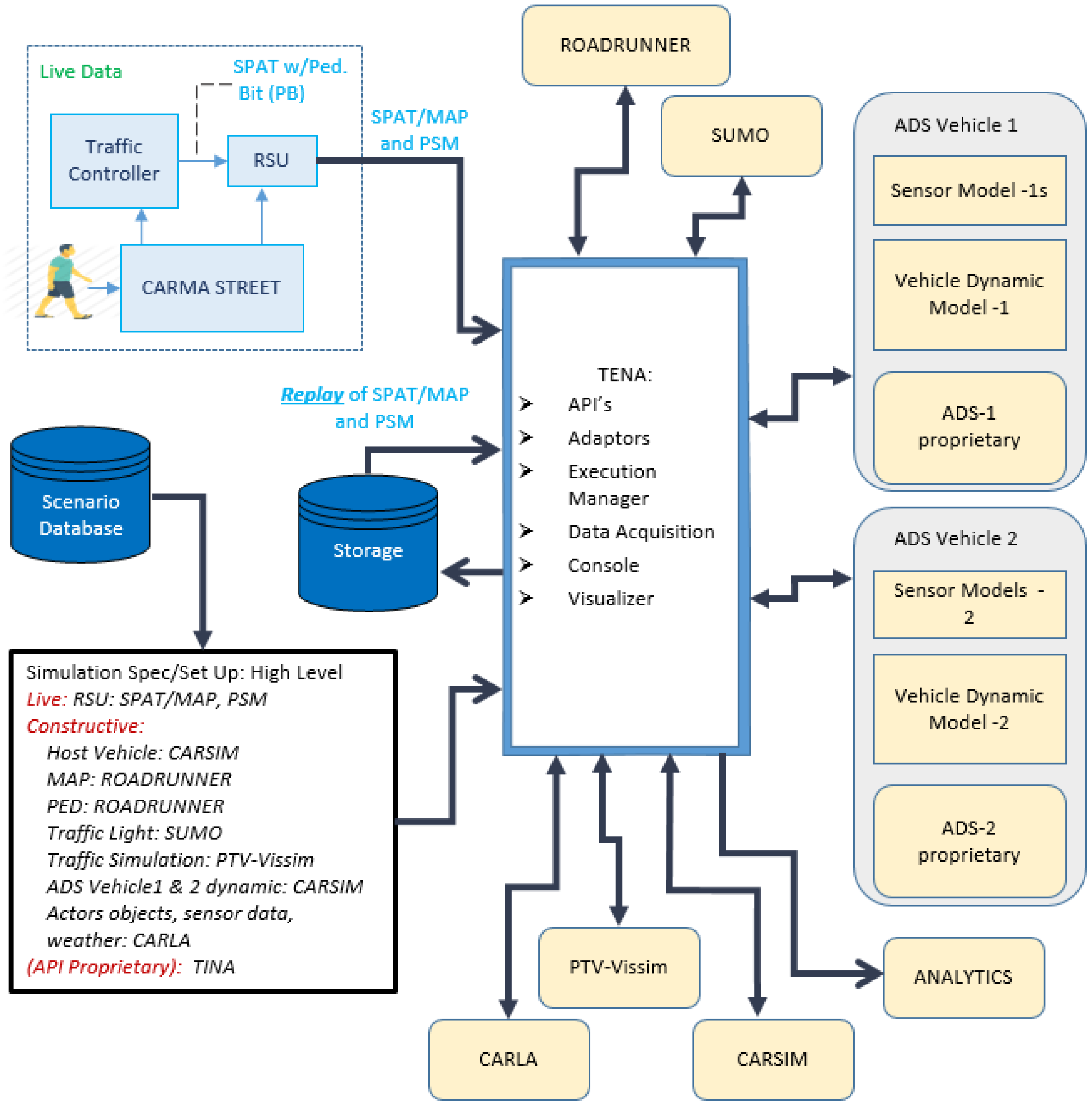

Figure 8. VOICES Managed and Executed Using TENA: ADS Vehicle Conflicting with Pedestrian Crossing

## 3. System Capabilities and Enabling Requirements

Based on the concept of operations as illustrated in the two test examples in Section 2.4, the VOICES system should provide capabilities to:

1. Construct a virtual environment that resembles a real-world driving environment:
    a. Integrating multiple simulation platforms with different semantics, and regulating complex interactions between diverse simulation models.
    b. Simulation platforms shall be interoperable.
    c. Selected simulation tools shall be capable of creating (for example):
        i. On road static and moving on road objects such as vulnerable road users, vehicles, lane marking, traffic signs, traffic lights, stop lines, curbs, work zone structure, etc.
        ii. Different types of roads, road attributes and road conditions
        iii. Moving objects dynamic models
        iv. Different weather conditions
        v. Road surrounding building structure
        vi. Subject vehicle engine and dynamics
        vii. Event, etc.
2. Protect the intellectual property (black box of individual systems) and ensure data privacy and test confidentiality of individual or group of testers.
3. Be secure.
4. Be scalable.
5. Be a connected system
    a. Be accessed remotely and receive data wirelessly.
    b. Integrate cloud connectivity and V2X communication.
6. Include a replay capability.
7. Include a capability to visualize data of interest.
8. Include a mechanism to feed the scenario database with a new or modified scenario.
9. Combine LVC testing. It shall:
    a. Be able to run different simulation modules at different simulation time steps. This is a simulation only requirement.
    b. Synchronize the LVC data.
    c. Have the required interfaces to exchange data with different LVC modules.
    d. Any simulations that are needed for an LVC test shall be able to run one to one with wall clock time and must not fall behind in time.
10. Set up and execute integrated tests from different users. The architecture shall allow exchanging data between different testing nodes.
11. Implement a mechanism to time synchronize the data coming from different sources.
12. Run a stressed simulation scenario without any significant intra delay. This delay shall not scale up significantly as the number of nodes increases. Example of stressed simulation scenario will

include different type of simulators, different users, complex test scenario (high number of actors in complex environment), and integration of different live elements.

13. Estimate with sufficient accuracy the latency in the data at different junctions in the data flow path.
14. Report the estimated latency to the SIL/HIL platform.
15. Accommodate throughput and processing to handle high data flow and multiple tests.
16. Host multiple tests at the same time. The architecture shall allow for executing multiple tests concurrently.
17. Execute stored and runtime scenarios from an integrated use case database.
    a. VOICES architecture shall enable simulated scenarios or some real-world scenarios to be specified or selected via user interface.
    b. Allow access to the database without any significant delay.
    c. Provide the scenario data to the proprietary model via API and without any significant delay.
    d. There may be room to change the motion data of other vehicles impacted by the disturbance of the HV motion.
18. Process raw and processed collected data. To handle the raw data, VOICES should have the needed framework to integrate a marketplace repository that includes a default AI-based perception module. The perception module is pluggable and therefore can be replaced with the user module if needed. The perception module shall be able to detect, track, classify, and characterize different types of vehicles, pedestrians, cyclists, road attributes, traffic signs, road signs, traffic lights, free space, etc.
19. Construct a sensing environment resembling sensors mounted on the vehicle and have pluggable sensor models for different types, including radar, lidar, and video camera.
20. Provide localization and electronic horizon (Look Ahead) data.
    a. Have access to different levels and types of map databases.
    b. Have models for different levels of GNSS receivers.
    c. Have access to point cloud maps for LIDAR localization.
21. Integrate the SIL and provide the needed API.
22. Run the test with system HIL and provide the needed interface and API.
23. Integrate and emulate the different V2X messages in SAE J2735.
24. Provide the needed interface for two-way data exchange with real devices such as traffic light controller, RSUs, and OBUs.
25. Store the testing results in a format configured by the tester.
26. VOICES test Environment should be controllable for test repeatability.
27. Integrate CARMA ecosystem capabilities into VOICES.

## 4. Development of Detailed Implementation Requirements for Proof-of-Concept Demonstration

There are two main sets of implementation requirements that are required in order for the VOICES project to move forward toward its demonstrable proof of concept:

1. Requirements for TENA adaptors.
2. Requirements for the TENA scenario manager.
3. Market Place Repository
4. Containerization
5. Some level of time management.
6. Requirements for Scenario definitions that are sufficient to execute a test.
7. Requirements for map accuracy and positional accuracy that is sufficient to integrate live and constructive systems into a shared real-time environment.
8. Requirements for accessing the environment and the data within it.

Development of these requirements requires working closely with simulator vendors, especially on the input data set to the simulators. The scenario manager needs to orchestrate all the simulation and produce the final integrated simulation correctly.

One approach in developing the requirements is to start with a single scenario use case, then expand the requirements as more cases need to be tested. This incremental development model will be helpful in moving the project moving forward towards its goal, and will provide some direction where these requirements will be heading.

Finally, it must be recognized that, at the end, the simulation needs to provide the HV proprietary module with its desired data. Therefore, it is important that implementers, system engineers, simulator vendors, users, etc. need to work together in this requirement development phase.

## 4.1 Requirements Examples

### 4.1.1 Voices Scenario Manager

Figure 9 shows the Voices high-level architecture.

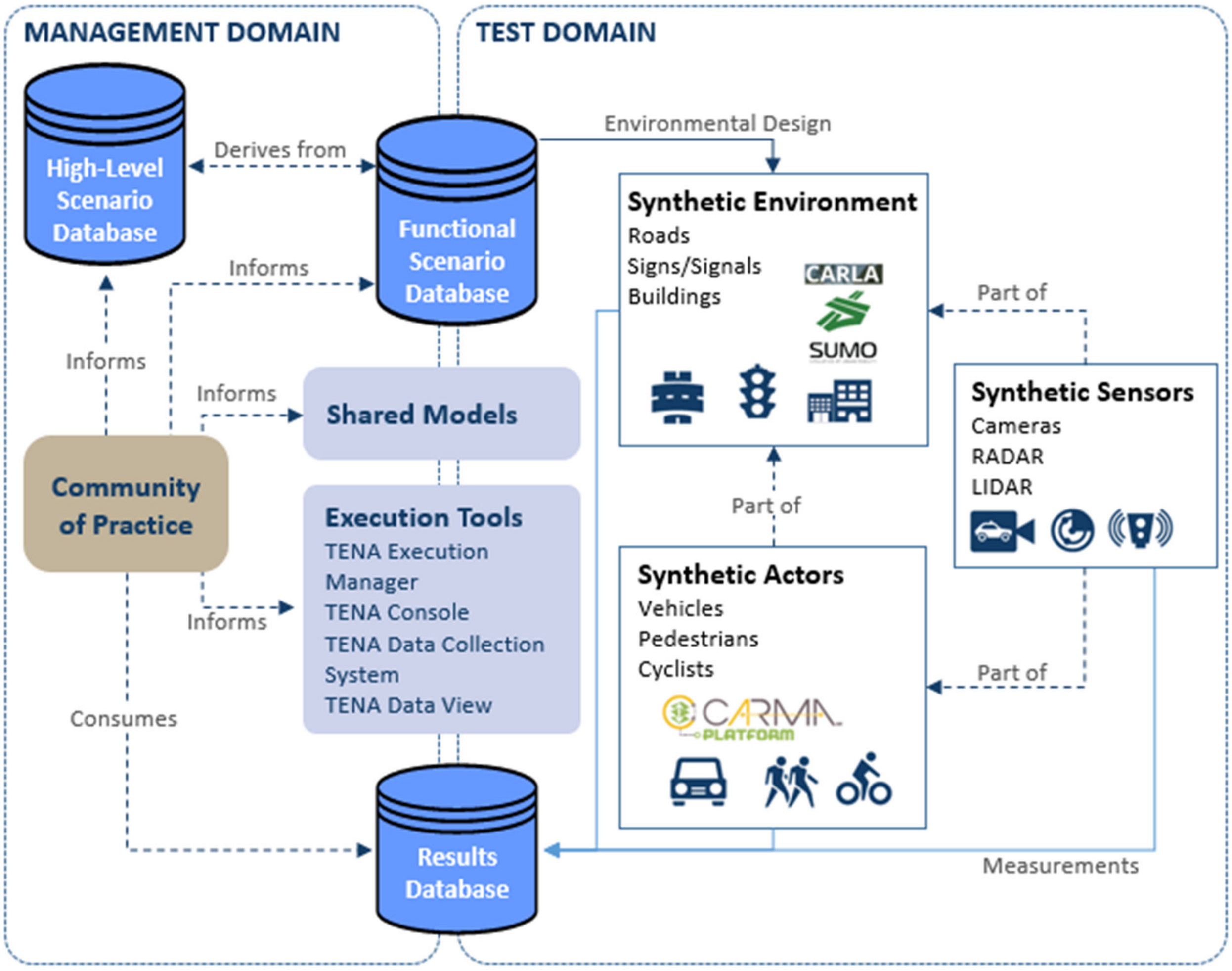

Figure 9. VOICES High-Level Architecture

The Scenario Manager is responsible of making the test happens according to the simulation specification from the simulation setup/configuration. It is basically the interface between the management domain and the test domain in VOICES architecture. Figure 10 illustrate how to use the Scenario Manager to begin a test.

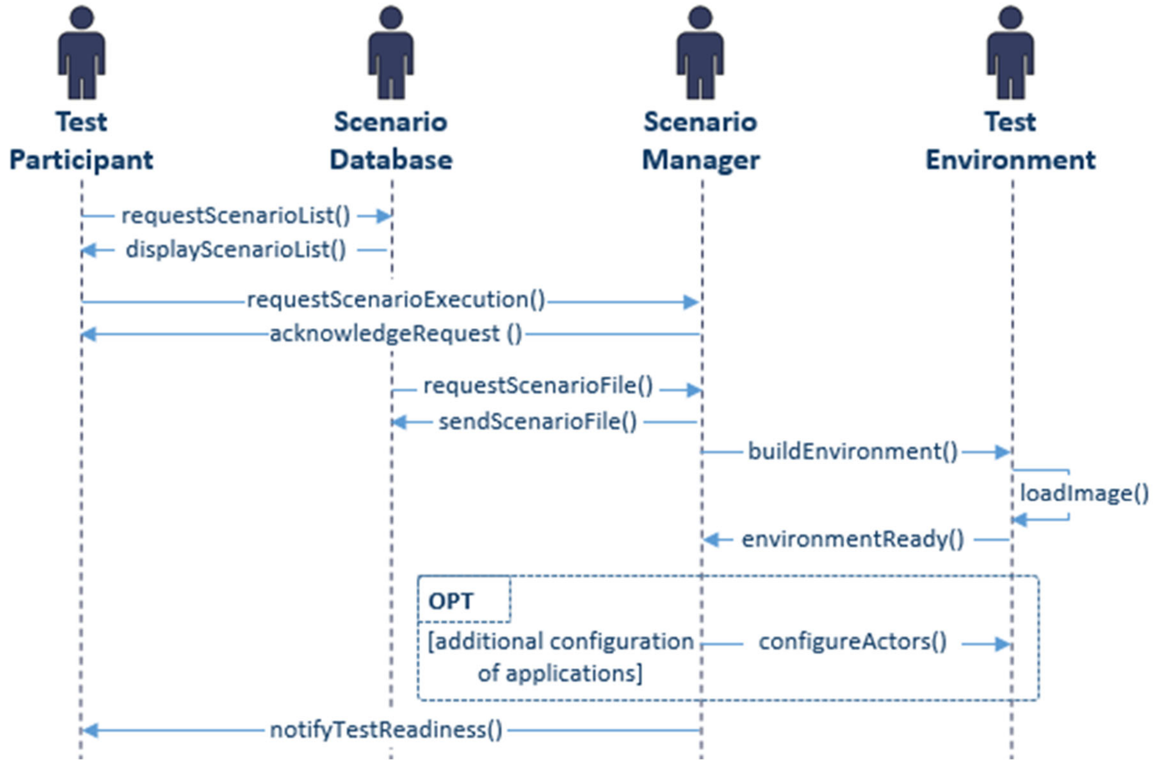

Figure 10. Using Scenario Manager to Begin a Test

### 4.1.2 Marketplace Repository

The system will provide the ability for users to select virtualization scenarios from a marketplace-style repository. The repository will provide the building blocks of test scenarios for users to build the virtual environment for customized virtualized testing environment. The initial instance of the marketplace will provide for the selection of preconfigured scenarios. The final goal will be the full modular marketplace for user customized testing.

There will be three types of users in the marketplace, as shown in Figures 11 and 12:

1. Test users who will have the ability to select components and configure variables within the components, or fully defined scenario. This user will also be able to run the simulation once all components are selected and configured.
2. Test contributors who will be able to submit new modules, scenarios, and test components for evaluation and inclusion in the marketplace. The modules and components will be evaluated by a board of administrators before inclusion for functionality and security prior to being available on the marketplace.
3. Administrators who will have the ability to add/remove/modify modules and components, participate in evaluation boards, remove components, and perform functions for adding, removing, elevating, or demoting privileges for other users and contributors.

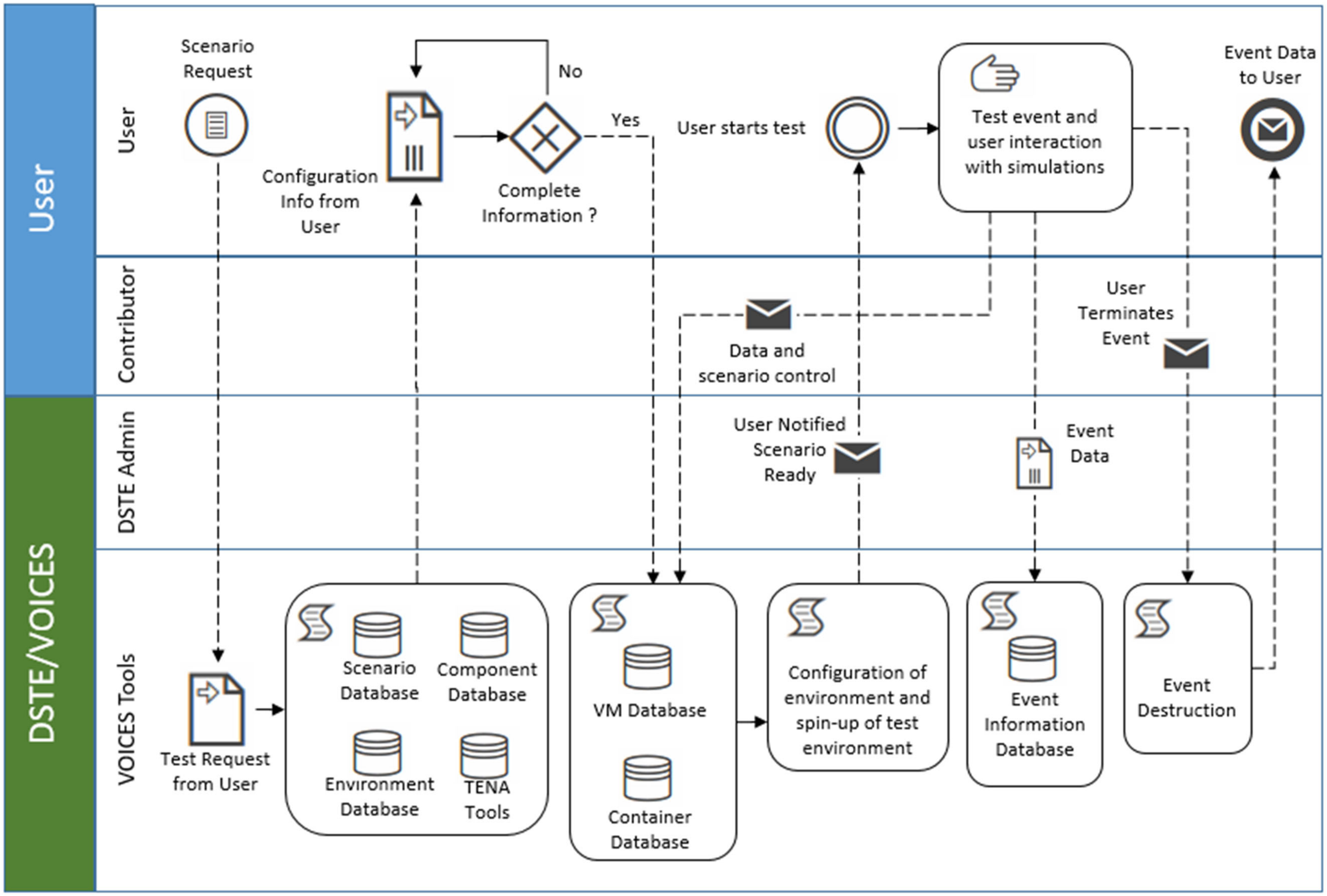

Figure 11. Marketplace Repository: Functionality

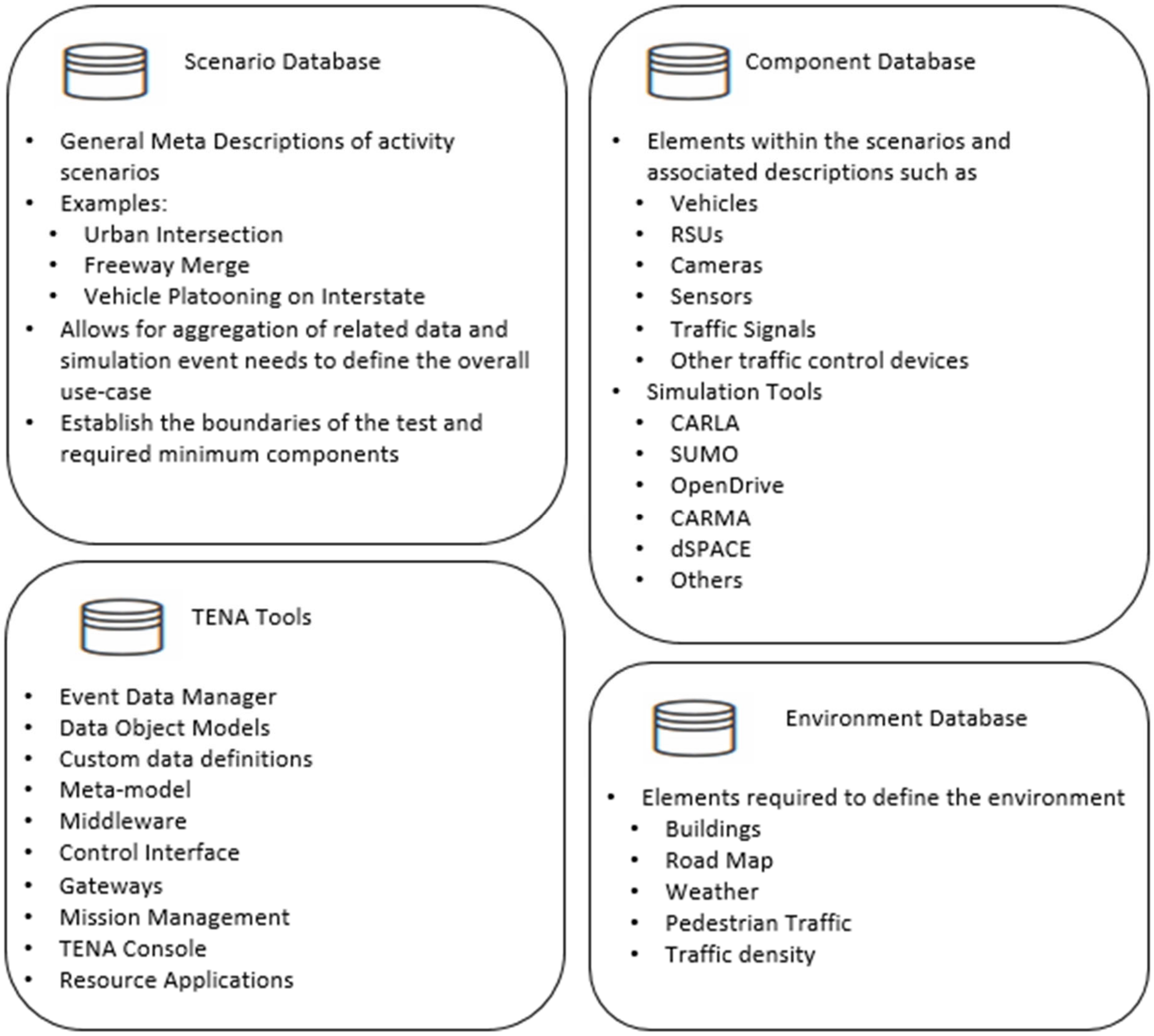

Figure 12 Marketplace Database

### 4.1.3 Containerization

The DSTE infrastructure for the VOICES environment will consist of containerized applications and virtual machines to provide pre-configured and secured test environments for individual test and experimentation scenario environments. The containerization technology and virtual machines will be determined and implemented by the VOICES DSTE administrators. The solution will be selected based on high availability, open-source licenses, and security features available to provide an environment that is low in requirements for administrators’ time and user learning curves.

### 4.1.4 Time Management

The VOICES testbed shall seamlessly provide multiple modes of operation to assist in testing algorithms ranging from immersion in a fully simulated environment with thousands of entities from multiple simulations to embedding in a real-time system for testing in an LVC distributed system.

## List of Acronyms

| | |
|---|---|
| **ACC** | Adaptive Cruise Control |
| **ADS** | Automated Driving System |
| **API** | Application Programming Interface |
| **BSM** | Basic Safety Message |
| **CACC** | Cooperative Adaptive Cruise Control |
| **CDA** | Cooperative Driving Automation |
| **DSTE** | Distributed Synthetic Test Environment |
| **GNSS** | Global Navigation Satellite System |
| **HIL** | Hardware In the Loop |
| **HV** | Host Vehicle |
| **LV** | Lead Vehicle |
| **LVC** | Live, Virtual and Constructive |
| **OBU** | On-Board Unit |
| **PSM** | Personal Safety Message |
| **RSU** | Road Side Unit |
| **SIL** | Software In the Loop |
| **SPaT** | Signal Phase and Timing |
| **TENA** | Test and Training Enabling Architecture |
| **TIM** | Traveler Information Message |
| **V2X** | Vehicle-to-Everything |
| **VOICES** | Virtual Open Innovation Collaborative Environment for Safety |