ISITEP
D5.5.3–SEMANTIC/SYNTACTICTRANSLATOR ENGINE DESIGN DESCRIPTION

Document Manager:	George Mitsopoulos	NETFI	Editor

Programme:	Inter System Interoperability for Tetra-TetraPol Networks
Project Acronym:	ISITEP
Contract Number:	312484
Project Coordinator:	FINMECCANICA
SP Leader:	RM3

Document ID N°:	ISITEP_D5.5.3_20160204_V1.1	**Version:**	V1.1
Deliverable:	D5.5.3	**Date:**	04/02/2016
		Status:	Approved

Document classification	**PUblic**

Approval Status	
Prepared by:	George Mitsopoulos (NETFI)
Approved by(WP Leader):	George Mitsopoulos (NETFI)
Approved by(SP Leader):	Alessandro Neri (RM3)
Approved by(Coordinator)	Paolo Di Michele (FNM)
Security Approval (Advisory Board Coordinator)	Etienne Lezaack (BFP)

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

CONTRIBUTING PARTNERS

Name	Company/Organization	Role/Title
Claudia OLIVIERI	FNM	Contributor
Federica BATTISTI	RM3	Contributor

DISTRIBUTION LIST

Name	Company / Organization	Role / Title
All Companies Project Managers	All involved companies	
Elina MANOVA	EC DG REA	EC Programme Officer
General Public	NA	NA

REVISION TABLE

Version	Date	Modified Pages	Modified Sections	Comments
V0.1	20/03/15	All	All	Initial version
V0.2	29/08/15	All	All	Slight adaptations
V1.0	31/08/15	All	All	Final release
V1.1	04/02/16	All	All	Updated according to the remarks of the Commission after the 2nd Annual Review

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

Publishable extended abstract

In this document, the Semantic and Syntactic translator is introduced, which will provide TETRA/TETRAPOL end users the ability to communicate in emergency conditions. Our application attempts to bridge the language gap between teams that do not share any common means of communications. Acting as an intermediate node between TETRA terminals, the Semantic and Syntactic translator will comprise a pipeline, which will translate a message to the native language of the target device's owner.

This deliverable focuses on design as well as implementation details of the translator tool. It provides the System Overview. Initially, the involved actors and the use cases are illustrated. The functional, as well as non-functional requirements of the system are analysed, while insights related to the system architecture are also given. In addition, it provides all the implementation details, in relation to the database design, the MySQL usage for the implementation, the Database Schema, the development language, as well as a use-case flowchart that explains further the afore-mentioned information. Last but not least, this deliverable reports on the test methodology that was followed, i.e. the database testing process, as well as, load, stress and trigger testing methods are discussed.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

CONTENTS

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

1 INTRODUCTION

During natural disasters, emergencies or times of crisis, multinational teams may be assembled and asked to cooperate, towards the accomplishment of a common goal. In such cases, effective communication between the individuals is of paramount importance for the successful completion of the undertaken effort. Obviously, not sharing a common language creates an insurmountable barrier, which prohibits the successful completion of the effort.

The Semantic and Syntactic translator introduced in this document is an effective tool that will provide TETRA/TETRAPOL end users the ability to communicate in emergency conditions. Our application attempts to bridge the language gap between teams that do not share any common means of communications. The Semantic and Syntactic translator will act as an intermediate node between TETRA terminals. It will comprise a pipeline, which will translate a message to the native language of the target device's owner.

Semantic and Syntactic translator is distributed across two network architecture components, namely:

- TETRA/TETRAPOL communication server.
- TETRA/TETRAPOL terminals.

More details regarding the design and deployment of the Semantic and Syntactic translator are provided in the following.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

2 DOCUMENT SCOPE

This deliverable (D5.5.3) reports on the Design Description of the Semantic /Syntactic Translator. More specifically, it reports on the following:

- Section 3 includes the definitions and abbreviations that are useful for the document
- Section 4 provides the System Overview. Initially, the involved actors and the use cases are provided. Afterwards, the functional, as well as non-functional requirements of the system are analysed. In the last sub-section of section 4, insights related to system architecture are given.
- Section 5 provides all the implementation details, in relation to the database design, the MySQL usage for the implementation, the Database Schema, the development language, as well as a use-case flowchart that explains further the afore-mentioned information.
- Section 6 analyses the test methodology that was followed. The database testing process is provided in the beginning, while afterwards, load, stress and trigger testing methods are discussed.
- Section 7 concludes the document
- Section 8 provides the document's references

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

3 DEFINITIONS AND ABBREVIATIONS

3.1 Definitions

This section is intended to capture the definitions of some key terms used in the document for the purpose of increased consistency:

Black box testing: Testing method, in whichemphasis is given in the “behaviour” and responsiveness of the database or software to external stimuli.

Command: This is the text to be translated. It mainly refers to the predefined set of translated texts/commands.

Source Language: The language that the user wants to translate.

Stress testing:also sometimes referred to as torturous testing as it stresses the application under test with enormous loads of work such that the system fails. This helps in identifying breakdown points of the system.

Target Language: The language, into which the user wants his request to be translated.

Trigger testing: very important in order to address a black box system testing. Trigger testing enables the identification and verification of system behavior and the level of compliance with the system behavioral requirements.

White box testing: Testing method, in whichemphasis is given in the internal functionality and operation of the developed database or software

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

Acronym	Definition
API	Application Programming Interface
DB	Database
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
RDBMS	Relational Database Management System
REST	Representational State Transfer
SQL	Structured Query Language
SW	Software
TCP	Transmission Control Protocol
TETRA	Trans-European Trunked Radio

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

4 SYSTEM OVERVIEW

In the context of this paragraph we will provide an overview of the designed solutions. We commence the presentation by outlining the actors that we envisage as involved in conjunction with a number of use cases. Afterwards we detail a number of functional and non functional requirements and conclude the section with the system architecture and the message sequence.

4.1 Involved Actors and Use Cases

The Semantic and Syntactic translation service will be developed as a web service. Towards this direction, a Graphical User Interface (GUI) will provide a point of interaction between TETRA Terminal and the translation service. The use cases that will be supported include:

- Translation of a specified command to a target language.
  - The system supports the provision of dynamic and non-dynamic translations. Non-dynamic translation procedure is triggered in case the requested translation for a command is available in the database. Otherwise, the system will dynamically produce a translation for the command. In order to do so the system will contact external online translation services so as to acquire a recommended translation of the unknown COMMAND. In both dynamic and non-dynamic translation procedures, the translated command is sent to the terminal.
  - A translation request is sent by a client to the translation service. The translation request may contain a predefined command that has been previously translated to a set of target languages or an unknown command so as to trigger the dynamic translation functionality
- Propose alternative translation.
  - A request containing an alternative translation for an already translated command is sent by a client. The alternative translation is stored in the database giving the ability to the administrators to process it and probably eventually update the initial official translation.
- Score provided translation.
  - A request containing a score for a translated command is sent by a client. A translation score is an evaluation mean of the provided translation given to the users. Translation scores are stored in the database. Administrators then use the score to evaluate and filter out or update probably poor quality translations.
- Update content.
  - Administrators use the web interface of the translation in order to update/edit existing translations or to extend the translator's knowledge base.

Following the aforementioned use cases, we will now present the main actors of our system. Table 1 that follows provides more details:

Table 1 System actors

Actor	Description
TETRA end users (non-authorized users)	End users that utilize the Semantic and Syntactic translator in order to translate commands.
Authorized Users	Selected end users that can contribute to the translating mechanism by scoring and suggesting corrections for already translated commands.
Administrators	Users with the authority to add remove and modify the actual database contents, presumably based on the received suggestions and scoring the translations that the authorized Users have performed.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

4.2 Functional and Non Functional Requirements

The use cases of the previous paragraph in conjunction with the project scope imply a number of functional and non-functional requirements that are presented in the context of this paragraph. Essentially these requirements drive our implementation effort and the selection of employed technologies.

- Clients will communicate with the server using a simple API
- Users should be provided with a simple and user friendly GUI that will require minimum learning time.
- The application and overall system must be easy and simple to maintain.
- The application must implement a modular and extensible architecture facilitating easy deployment and update by means of including new modules without imposing any burden on the users.
- Inter-component communication should take place with simple, fast, robust and well known communication protocols. Towards this direction, HTTP appears as potential solution in case it is supported by the terminal.
- If HTTP cannot be used then a custom TCP solution should be available.
- The application must be software and if possible hardware independent. The latter will minimize operating system and device hardware restrictions. Towards this direction, a minimal Python implementation seems like a viable option.
- Information should be easily updated with a simple way by the administrators. A web application facilitating management of the stored information should be designed.

4.3 System Architecture

Following the use cases and the requirements of the previous sections we envisage a 3-tier client server application consisting of

- TETRA/TETRAPOL communication server,
- A Database for permanent information storage and,
- ISITEP enhancedTETRA/TETRAPOL terminals.

Regarding the first component of the architecture, Semantic and Syntactic translator is considered as an online service running on top of a TETRA/TETRAPOL communication server. Its purpose is to serve translation requests as those are provided from TETRA/TETRAPOL terminals. Such translation requests are of the form <COMMAND, SOURCE LANGUAGE, TARGET LANGUAGE(S)>.When a new translation request is received, Semantic and Syntactic translation service will initially check the existence of the requested COMMAND. A query to the database placed in TETRA/TETRAPOL communication server is executed so as to examine whether a translation for this COMMAND in the considered SOURCE LANGUAGE exists or not. In case it exists, the translated COMMAND to the TARGET LANGUAGES is sent back as a response to the translation request. However, when a miss occurs, the translator will automatically invoke external online translation services (e.g. online University translation service, Google translate, etc.) so as to be consulted with an initial arbitrary translation of the unknown COMMAND.

Therefore, Semantic and Syntactic translation service in TETRA/TETRAPOL communication server provides two functionalities: i) static translation of predefined COMMANDS, ii) dynamic translation of newly introduced COMMANDS. Concentrating on the static functionality of the translator, the majority of the incoming translation requests, as these are generated from TETRA/TETRAPOL end users, regards predefined commands known a-priori (during the design and development of the system). Those COMMANDS and their equivalent translated counterparts are manually fed to the system as an offline process. This offline process reassures the correctness of translated terms and the

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

elimination of mistakes that would cause deficiencies in the communication during emergency conditions. Furthermore, the static functionality of the translator can be easily updated offline, anytime, (e.g. during maintenance of the server), in order to upgrade the system and extend translators service capabilities.

Apart from the static translation of predefined COMMANDS, translator also provides dynamic translation of unknown commands through the utilization of open access online translators. Every unknown COMMAND and its suggested translation to TARGET LANGUAGES will be stored temporarily in server database. Thus, translation service is capable of extending its functionality with newly introduced commands, while avoiding gaps in communication of TETRA/TETRAPOL terminals. Proposed caching of temporal translated COMMANDS is developed in a way to be easily updated and also permits the removal of old entries, when for instance the time of temporal storage expires.

Proceeding on to the second component of the architecture that Semantic and Syntactic translator is deployed on, translator provide TETRA/TETRAPOL terminal end users with the ability to both translate COMMANDS to a specific set of TARGET LANGUAGES and, if they are authorized, to vote for a quality score in already translated COMMANDS. This functionality (i.e. voting for a quality score) is granted to end users through a web interface. In addition, end users have the opportunity to propose alternative translation to existing translated COMMANDs. Quality scores gives administrator of the Semantic and Syntactic translation service a straightforward way to enhance poor quality translated COMMANDS, especially in cases where dynamic translation of newly introduced COMMANDS occurs. Following the same approach, end user proposals can be evaluated offline by the Semantic and Syntactic translation service administrator and support them improve existing translations. Considered functionality will be accessible to a set of privileged TETRA/TETRAPOL terminal end users through a web interface.

Finally, a Graphical User Interface (GUI) will provide a point of interaction between TETRA Terminal and the translation service. The use cases that will be showcased, include: i) a use case where a translation request contains a predefined COMMAND to a set of TARGET LANGUAGES and ii) a use case where a translation request contains an unknown COMMAND so as to trigger the dynamic translation functionality.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

5 IMPLEMENTATION DETAILS

The semantic and syntactic translator implements a three tiered architecture. This type of SW architecture provides a way to theoretically categorize the components of an application, providing abstract modularity. In essence the model is a client-server architecture which is composed of three layers; the presentation tier, the functional process logic tier and the data storage tier. These three tiers communicate with each other in a strictly defined way, which allows the developer to independently maintain each tier.

- In most practical applications of the model, the presentation layer includes all forms of user interaction with the service. The process logic tier encompasses the application's functionality, while the data storage tier handles the flow of the information from and to the data source, which in our case is a database.
- Presentation layer: The presentation tier in the translator is composed of a network API (HTTP or custom TCP-based) which will provide access to the translator functionality (translate command/text, update translation etc.) and a set of web pages that utilize a REST API[1] (web pages providing an interface for administrators and privileged users to manage the translator database).
- Application logic layer: The application logic tier is divided into two parts that each serve its own purpose; the base application logic and database connectivity. The base application logic is a group of functions that handles the incoming user requests. The database connectivity provided easy access and management of the stored data.
- Data storage layer: The third and last layer of the translator is responsible for the data management of the whole service. For the Data storage layer we use a relational database management system (RDBMS), namely MySQL ([2]).MySQL, is a free, open source cross-platform tool, and a popular choice of database for use in web applications.

The architecture is shown in Figure 1.

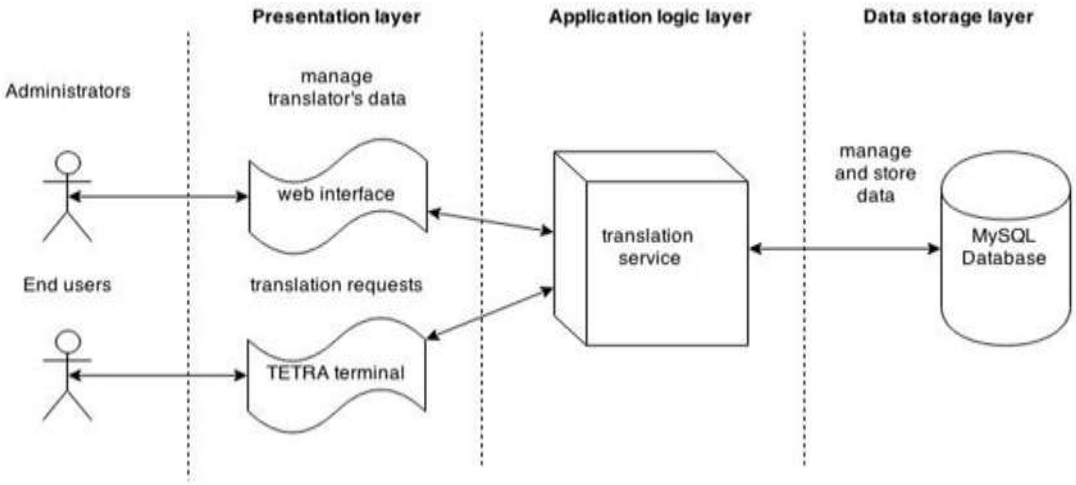

Figure 1: Software Architecture.

5.1 Database

During the deployment of the TETRA/TETRAPOL translation service, the translator will contain a set of predefined commands in English, varying from 50 to 100 records. Those commands will be translated offline to a specific set of languages (Italian, French, English, Swedish, Dutch, Spanish) and will be stored in a database. The database constitutes the knowledge base of the translator.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

5.1.1 MySQL

In terms of database we opted for MySQL. MySQL is the most widely used open-source RDBMS and deployed in many high-profile, large-scale applications and websites (i.e. Twitter and Facebook). Some of the MySQL's benefits are:

- *High Performance*: It provides a unique storage-engine, which can be easily configured to achieve amazing performance results. It achieves the most demanding performance expectations from high-speed transactional systems to heavy-load web sites, which serve a tremendous number of queries per day.
- *Multi-Platform*: It is easy to install and runs in many different operating systems such as Linux, Windows and Mac OS X.
- *Easy application development*: It offers many development interfaces thus application developers can build solutions in any of the many supported languages (PHP, Python, Java, etc.).
- *Open Source*: It is freely available from the MySQL web site. Its open source community provides trusted software and great support.

The afore-listed benefits make MySQL a valid database choice for the translation service.

5.1.2 Database Schema

The translator's database schema is shown in Figure 2. The database consists of three tables: *Language*, *Command* and *CommandExtras*. Inter-table associations *Command-Language* and *Command-CommandExtras* are one-to-one relationships. The tables in more details:

Language

This table contains the supported, by the translator, languages. Every language is mapped to a unique *Language_Id*, which is used to join and access the *Command* table.

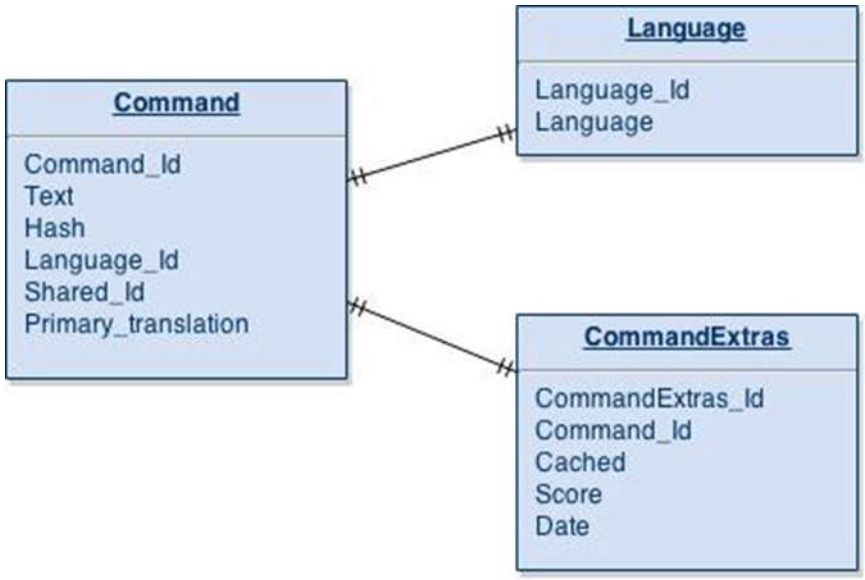

Figure 2: Database tables

Command

This table represents the translated commands. It contains all the information needed for querying incoming translating requests. It contains the following fields:

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

- *Command_Id*: unique id for every stored command in the database
- *Text*: this is the command's full text
- *Hash*: A SHA-1 hash value of the Text field
- *Language_Id*: this id is used for joining the Command table with the Language table
- *Shared_Id*: it is the id that same commands share across different languages.
- *Primary_translation*: a flag that indicates whether this translation is primary or not. If it is not a primary then it is an alternative translation. Only primary translations are given as response to a translate request.

Generally big string/text searching in a database is not always very efficient especially in the translator's case where string/text searching is a very common task. In order to assure efficiency instead of searching for a command via its full text, we first hash the command's text and then we query the database using the hash value. This way the string/text used for querying is much smaller (in length) than the original command's length thus the searching is faster.

CommandExtras

This table contains all the additional info needed for each command.

- *CommandExtras_Id*: unique id
- *Command_Id*: this is used in order to join the Command table with the CommandExtras table
- *Cached*: this is a flag which indicates whether this Command_Id comes from dynamic translation (and thus is cached) or not.
- *Score*: the translation's score (if any)
- *Date*: The date when the command/translation was stored in the database.

Predefined and dynamically translated commands are stored under the same table *Command* but can be distinguished through the *Cached* flag. The fields *Score*, *Cached* and *Date* are used for filtering out/deleting commands from the database. Translations with score under a specific threshold can be easily removed through a simple query. Field *Cached* in conjunction with field *Date* can be used to remove old cached entries.

The above design's simplicity makes querying and managing the database easy and straightforward.

5.2 Development Language

The Semantic and Syntactic translation service will be implemented in Python programming language[3]. Python is one of the most popular and fastest-growing interpreted languages. It 'is very strong, stable and it is used in the industry for a lot of different usages. Here are the reasons why we choose Python for the implementation:

- *Python is efficient and fast*: As a high level language it provides a strong automatic memory management which makes even complex data processing straightforward and easy. Although it is an interpreted language, tense development in the language's core has been done over the past years to improve and assure Python's high performance.
- *Cross-platform*: Python is hardware independent. It is easy to install and run on all major operating systems.
- *Many third-party libraries*: Although the language itself comes with a huge set of features and libraries there are also many third-party libraries which extend even more the language's capabilities.
- *Flexible and easy to use*: Python's simple and clear syntax encourages good programming habits which lead to fewer software errors and faster development. The code is more readable and easier to extend and maintain.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

- *Open source*: Python is an open source programming language which immediately reduces up-front project costs.

5.3 Use –case Flowchart

In the previous sections we presented the translator's architecture and implementation details. Here we present the basic use-case, i.e. translating request, in more details. The translation process is shown step by step in Figure 3.

First the user makes a translation request. The translator checks if there is a translation in the database, which satisfies the received request. If it exists then the translator immediately responds back to the user with the answer.

In case the translation request contains an unknown command then the translator invokes an external translating service and dynamically translates the request command. The translated command is stored in the database (cached for future reuse) and the response is sent back to the user.

In this section we have presented a high-level view of the steps of the translating request process. Further details on this use case as well as other communication scenarios (i.e., unsuccessful translation, score submission and DB update) are provided in **Errore. L'origine riferimento non è stata trovata.** where the SST is decomposed to its Functional Components and detailed analysis of the information exchanged among the Functional Components is given.

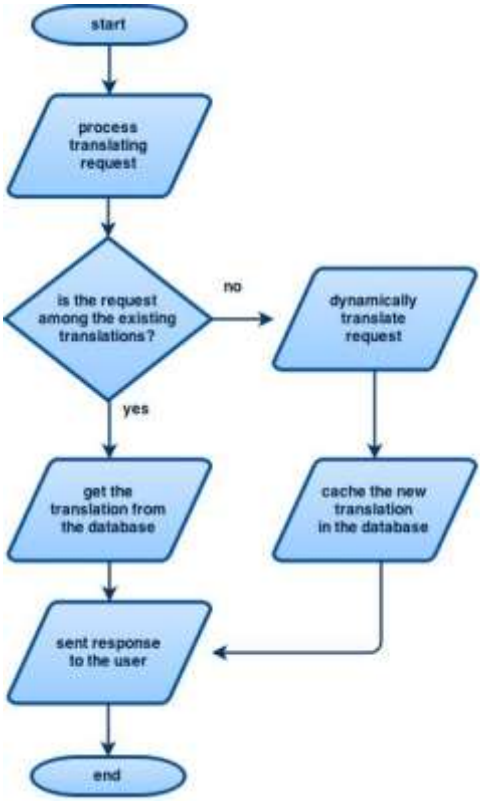

Figure 3: Translation use-case flowchart

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

6 TEST METHODOLOGY

Computer applications are more complex these days with the use of new Operating Systems, like Android, and with the spreading of a large number of smart phone apps. The more complex the front ends, the back ends are even more intricate. Especially in TETRA applications and mission critical environments, it is apparent that testing is required in order to assess the effectiveness and responsiveness of the DB.

The most common approaches to testing are basically two:

White box testing: emphasis is given in the internal functionality and operation of the developed DB or SW.

Black box testing: emphasis is given in the "behavior" and responsiveness of the DB or SW to external stimuli.

For the SST and DB operation testing, we focus on the black box testing approach, since it is more suitable to test the validity and correspondence to certain KPIs related to load and response time.

6.1 Database Testing Process

The general test process for DB testing is not very different from any other application. The following are the steps:

Step #1) Prepare the environment: Prepare SST engine to receive translation requests. Set up the server and infuse predefined data to the DB schema.

Step #2) Run a test: Execute queries from the ISITEP enhanced terminals that require either non-dynamic, or dynamic translation

Step #3) Check test result: Measure response time of the SST engine

Step #4) Validate according to the expected results: Expected results are based on the mean response time of the mysql and transmission time of the data.

Step #5) Report the findings to the respective stakeholders.

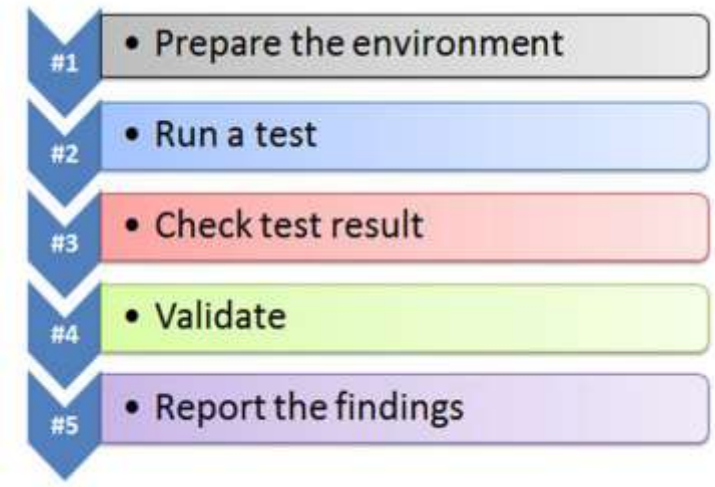

Figure 4: Database testing process

6.2 Load testing

The purpose of any load test should be described in use cases or scenarios intended for the testing.

The following types of configurations are common for load testing.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

1. The most frequently used user transactions have the potential to impact the performance of all of the other transactions if they are not efficient.
2. It is recommended that one non-editing user transaction to be included in the final test suite, so that performance of such transactions can be differentiated from other more complex transactions.
3. The more important transactions that facilitate the core objectives of the system should be included, as failure underload of these transactions has, by definition, the greatest impact.
4. The observation of the optimum response time under big numbers of virtual users for all the prospective requirements.
5. The observation of the effective times for fetching of various records.

6.3 Stress testing

Stress testing is also sometimes referred to as torturous testing as it stresses the application under test with enormous loads of work such that the system fails. This helps in identifying breakdown points of the system.

Most common occurring issues during database testing

1. Significant amount of overhead could be involved in order to determine the state of the database transactions.
2. Solution: The overall process planning and timing should be organized so that no time and cost based issues appear.
3. An SQL generator is required to transform SQL validators in order to ensure the SQL queries are apt for handling the required database test cases.
4. Solution: Maintenance of the SQL queries and their continuous updating is a significant part of the overall testing process, which should be part of the overall test strategy.
5. The afore-mentioned prerequisites ensure that the set-up of the database testing procedure could be costly as well as time consuming.
6. Solution: There should be a fine balance between quality and overall project schedule duration.

6.4 Trigger testing

Trigger testing is very important in order to address a black box system testing. Trigger testing enables the identification and verification of system behavior and the level of compliance with the system behavioral requirements.

1. Whether the required coding conventions have been followed during the coding phase of the Triggers.
2. Check whether the triggers executed for the respective transactions have fulfilled the required conditions.
3. Whether the trigger updates the data correctly once they have been executed.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

7 CONCLUSIONS

In this deliverable, the description of the SST system and data base architecture has been presented, detailing the functional part and the respective operations. Initially, in the first part of the document an overview of the system was presented, illustrating the involved actors and use cases. The requirements of the system were discussed and insights regarding the system architecture were provided.

In the next part, the details of the system implementation were provided. The data base schema has been illustrated and analyzed. Moreover, the basic approach and methodology for the system and Database testing has been presented and discussed.

More important for the next steps is the use of the described methodology, which is state of the art approach to system and database testing, in order to define the test scenarios and use cases and to validate the most important aspects of the data base and SST operations, which are related to stress and load testing. The main consideration for the testing approach is the use of the system as a black box and focus on the behavior of the system and DB operation.

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

8 REFERENCES

[1] [REST API tutorial, [Online],http://www.restapitutorial.com

[2] MySQL website, [Online], https://www.mysql.com

[3] Python programming language website, [Online], https://www.python.org

[4] ISITEP Deliverable D5.5.1 – “Semantic/Syntactic Translator Database Population”

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee

This document is produced under the EC contract 312484. It is the property of the ISITEP Parties and shall not be distributed or reproduced without the formal approval of the ISITEP Steering Committee