DILVI – A Platform to build Language Training Simulations

Dilunika J. K.[#1], Wimalarathne S. P.[#2]
[#]*University of Colombo School of Computing*
[1]kasundilunika@gmail.com, [2]spw@ucsc.cmb.ac.lk

Abstract — ***This paper introduces DILVI: an abstract OSGI based framework which enables the developers and researchers to focus more on advanced pluggable modules such as Speech Recognition, Automatic Language Assistance and Student Evaluations rather than concentrating on the underlying framework while developing language training simulations. The evaluation of DILVI has proven to be a successful as it has shown 70% reduction of development efforts significantly.***

Keywords— **Simulation, Framework, Language Training**

I. INTRODUCTION

It is a well-known fact that practice leads to perfection [1] which is also proven in the task of learning a foreign language, than in any other task. Foreign language teachers try out different tactics to make that environment available for students.

Tactical Language Training Systems are a good answer; they are designed to help learners quickly acquire basic communication skills in foreign languages and cultures. Many numbers of tactical languages training simulation systems have been developed so far [2] [3] [4].

As AI technologies, especially Natural Language Processing techniques are still in their development stages, tactical language training systems also have the capacity to be improved more. There are various open research areas to be touched on. When someone needs to start such a research in the area of tactical language training simulation development, he or she has to start from the scratch, most of the times with the graphics engine. This is a burden which costs considerable amount of time only to build the base for the research when it compared with the total research effort. Especially, this paper presents a framework which can be used for building English language training simulators with the focus of following goals:

- ✓ Facilitate API developers with boiler-plate implementations of the graphical functionalities as a wrapper to the graphics engine.
- ✓ Provides a modular (plug-in based) architecture to develop and deploy NLP components as 'in-VM (Virtual Machine) services'.

The remainder of the paper is structured as follows. Section 2 presents related work in the area of tactical language training simulator environments. Section 3 presents the DILVI architecture and implementation of the components it is comprised of. Section 4, Evaluation, describes the testing and validation process. It explains how the experimental set up is arranged and the tests are carried out on the prototype built. Finally, section 6 states the conclusion and future developments of the research area.

II. RELATED WORK

The research efforts to introduce a framework which abstracts some of the boiler plate implementations in Language Training Simulation development domain. A large number of researchers have focused on developing such training simulations in different perspectives while lesser number of researchers has tried to build a common framework to some extent which prevents some of the hard initiatives.

Modern day Computer Supported Language Learning (CSLL) applications are not just simple 2D games. If the CSLL domain is analysed, basically it can be divided into two sections: CSLL applications with, gaming technologies and non-gaming technologies.

Second Life [5] is one of the most popular non-game 3D multi-user virtual environments in the CSLL domain. Although it is free to most users, Second Life is in fact a commercial endeavour. As a platform SL is capable of supporting not only traditional pedagogical modes such as lectures and discussions, but also experimental learning experiences that would be difficult or impossible to achieve in real life, such as complex scientific or historical simulations with which users can interact directly. Since Second Life is a commercially available platform, authors more concerned about the OpenSimulator [6], which is an open source multi-platform, multi-user 3D application server. More interestingly, to deal with OpenSimulator, a user does not need to be a smart 3D application developer. As all of these platforms (SL and OpenSim) are rich and very high-level, they are not providing reasonable control to their users. As OpenSimulator is a Multi User Virtual Environment (MUVE) it creates avatars per user, simply avatar for just connected OpenSim client. Hence it was very heavy and painful task when it comes to scenarios that are having in this proposed framework. In our scenarios, there are few avatars in the simulation, but only one should be there as a connected user. All others should be simulator managed avatars. To have above described feature, it is required to do a huge hacking to the OpenSimulator environment.

The proposed framework here requires some more control over the animation framework and the pedagogical agent. As discussed previously, to build up this proposed framework, we need an underlying framework which provides control in a lower level rather than the abstract level above frameworks provides. So it is decided to step down to another level rather than using high-level abstracted CSLL frameworks.

The idea of employing a game engine for non-game applications is not new. 3D game engines have found their

2013 International Conference on Advances in ICT for Emerging Regions (ICTer) 978-1-4799-1276-6/13/$31.00 ©2013 IEEE

way to a number of non-game applications ranging from CAVE™ installations to simulating context-aware services [7] [8][9].

Our analysis is thoroughly done on three game engines, OGRE [10], UnrealEngine 3[11] and jMonkey[12] before the conclusion is made for our selection. UnrealEngine 3 is leading with its great graphics, loads of features it provides and the rendering speed it has. Therefore it is perfect for game designing. But, as we stated in the OpenSimulator critique in the previous section discussion, this engine also has a high level abstraction that prevents us having more control on its core which is more required when designing a framework like we are proposing here.

When the OGRE is concerned, although it is not a rich game engine like UnrealEngine[13], it provides such sufficient features which are required and also the level of control on its core needed for our framework design. This proposed framework is designed to integrate with bunch of open source NLP libraries which most of them are written in Java and Python. As OGRE is purely written in C++, there is a considerable performance cost and effort when it is come to integrating non C++ NLP libraries.

After considering the facts above described, JMonkey engine is selected as our underlying graphics engine. JMonkey engine is slow in rendering when it is compared to UnrealEngine 3 and OGRE, but it is not a great issue when we consider the complexity of the graphics we are willing to use with this framework.

It is obivious that there have been many attempts made to use VR technologies and NLP concepts in teaching ESL successfully with lucrative features such as correction supporting, pedagogical support, improved graphics, etc. However, all of them are applications which provide minimal support or allowance for enhancements or changes to them. A researcher who wishes to enhance certain features would require developing a system from the scratch as a result of the rigidness of the existing applications. Hence, the need of an abstract framework which would provide a good stepping stone for both researches and developers who wish to use NLP and VR techniques is inevitable.

III. DILVI FRAMEWORK

When it is analyzed functionalities of a tactical language training simulator, following set can be recognized as major features.

- Parse the script file and build the role-play model.
- Create System Characters to interact with student with the audio dialogs.
- Generate the pedagogy system for the dialogs to be practiced.
- Capture the response provided by the student.
- Evaluate the responses of the student.

From the above functionalities, first two are not getting changed most of the time once they have been properly developed, while latter three are more likely to be changed in its implementation. The "DILVI" framework is targeting to provide base implementations to those mostly stable functionalities and keep extension points for features which are likely to be changed.

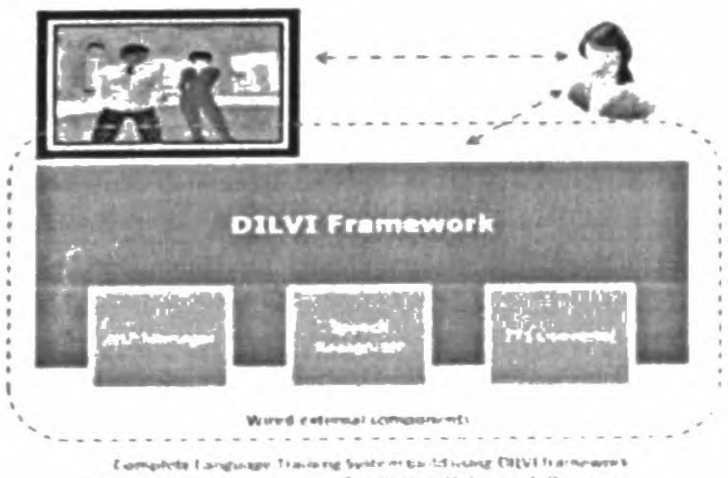

Fig 1 Architecture of a DILVI based System

As illustrated in figure 1, DILVI provides the solution for this stalking issue of not having a framework to motivate and speed up the application developers and researches who are involved in Computer Aided Language Learning.

A. Architecture and Design

DILVI composed with following modular architecture illustrated in figure 2.

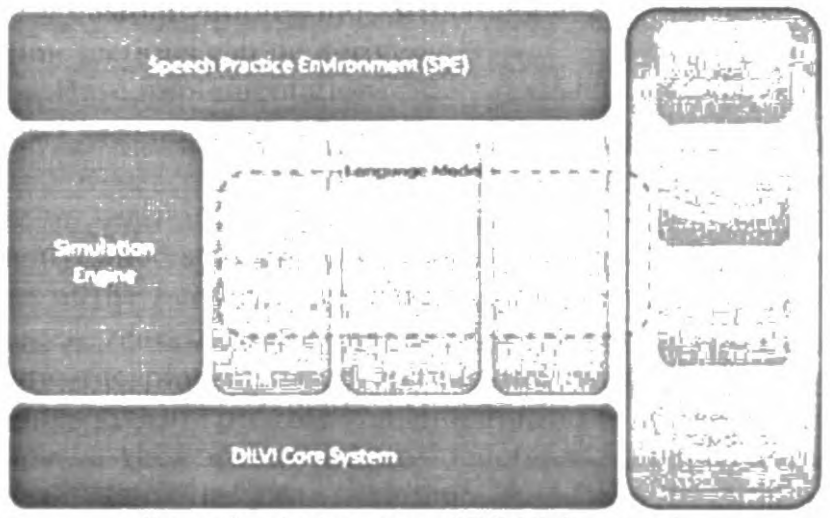

Figure 2 – Architecture of DILVI framework

DILVI's unique architecture can be discussed in several perspectives as depicted bellow.

Developers Perspective: DILVI framework can be seen with respect to the developer's viewpoint of the areas which requires customization and which does not. Hence the following sections can be derived.

- Base implementations
- Extension points to be used for other functionalities

Plug-in Perspective: Module based development is one of the highly interested areas in modern software engineering, because of the overhead created by large complex monolithic software systems during its maintenance. DILVI framework is following the plug-in based architecture as the frameworks' main idea is to provide an integration platform to the developers who are trying to develop language learning simulations. In order to facilitate the plug-in life-cycle management, DILVI uses OSGi container.

978-1-4799-1276-6/13/$31.00 ©2013 IEEE

B. Implementation

A prototype system was built for the purpose of demonstration and the proof of the concepts presented. The framework is designed to be developed in Java [14] platform. Java Standard Edition 1.7 has been used as the Java technology in the DILVI framework. Eclipse Equinox [15] is the container which is used as the implementation of the OSGi specification [16]. Spring framework is used with its Dependency Injection [17] and Dynamic Modules features. The top level layered architecture can be illustrated as in Figure 3.

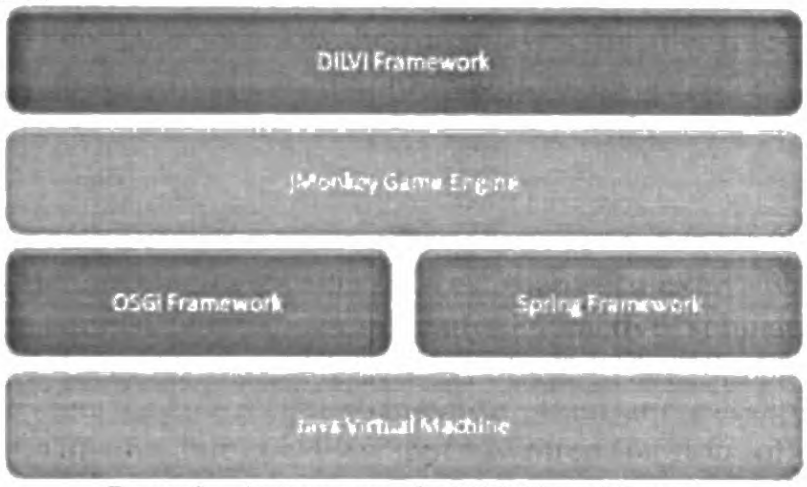

Figure 3 – Architecture of a DILVI based System

DILVI framework consists of six main modules: Role-Play Director, Simulation Engine, Speech to Dialog, Dialog to Speech, Pedagogical Agent and Progress Monitor. Except the 'Progress Monitor' module, all other modules have been implemented in the prototype system.

1) Role-Play Director

Role-Play Director takes the role-play script file, role-play name and description as inputs and generates a RolePlay object which is the main processing data structure for the rest of the modules

2) Simulation Engine

Simulation Engine is the place where RolePlay object model is translated to a graphically interactive simulation. Real life stage drama concept is applied to the Simulation Engine's design in order to achieve the translation described above. This module is a wrapper written on top of the jMonkey game engine and defines the contracts pertinent to attain the translation between RolayPlay data model and Drama data model.

3) Dialog to Speech (DTS)

Primary objective of this module is to make audio clips for the System Characters using related text dialogs. DTS module is one of the extensible modules in DILVI framework which signifies, the framework itself only provides a well defined API with the required contracts. Nevertheless, the framework is shipped with at least single implementation plug-in.

4) Speech to Dialog (STD)

Capturing user response is a critical part of DILVI framework. Speech to Dialog module is responsible to handle response capturing. Generally, it can be divided into two sub tasks: record what user speaks and recognize what user speaks, where latter is the toughest challenge in the framework. STD module is also an extendible module in DILVI framework, users of the framework can hook outside module into the framework just by adhering the STD API contract while having any underlying implementation.

5) Pedagogical Agent

Pedagogical agent module is responsible to provide the assistance to the student while he is practicing the simulated role-play. It can be achieved in two steps, first tagging the dialogs with tags based on the occasion it is used and secondly, showing those tag phrases as hints for the practicing dialog while student is engaging with the role-play simulation.

Tagged Phrases are kept as a XML corpus as shown in figure 4. Each time while a role-play is created, if there is a new phrase, phrase is tagged and added to the corpus.

```
<PhraseCorpus>
        <phraseSet>
                <tag>MeetingSomeone</tag>
                <phrase>
                        <text>Hello, after long time</text>
                        <rank>1</rank>
                </phrase>
                <phrase>
                        <text>Hey,</text>
                        <rank>1</rank>
                </phrase>
        </phraseSet>
</PhraseCorpus>
```

Figure 4 – Structure of the XML Phrase corpus

Pedagogical Agent is also an extension point in the DILVI framework where org.dilvi.pedagogi.api is the API plug-in for this module.

6) Integration of the Modules

Extension points are the access points to the framework. If the application developer is not satisfied about the default implementation which framework contains, he or she can extend the functionality of the framework by adding a new plug-in developed as an extension module. The process that user required to follow in order to implement an extension module will be briefly discussed in the developer guide with an example.

7) Proof of Concept Application

In order to prove the usability of the DILVI framework, a Proof of Concept (POC) application has been developed based on framework with some minimal functionality. POC contains DTS implementation plug-in based on the freetts open source library which is capable of generating audio files for the dialog conversations provided in the script file. Although we were unable to include a complete implementation of STD API due to the limited time, we have provided a basic implementation of STD API which is capable of recording the student's responses as audio files for further manual evaluation.

POC is capable of accepting a script file with dialogs in the format as shown in table 1 and building a role-play based on the content of the script. Figure 5 shows a screen shot of the script pointing window.

Amal:Hello, after long time~
Kamal: Yes, after awhile.~
Amal: So, how are you?~
Kamal:I am quite well, thank you.~

Table 1 – Format of a sample script

2013 International Conference on Advances in ICT for Emerging Regions (ICTer) 978-1-4799-1276-6/13/$31.00 ©2013 IEEE

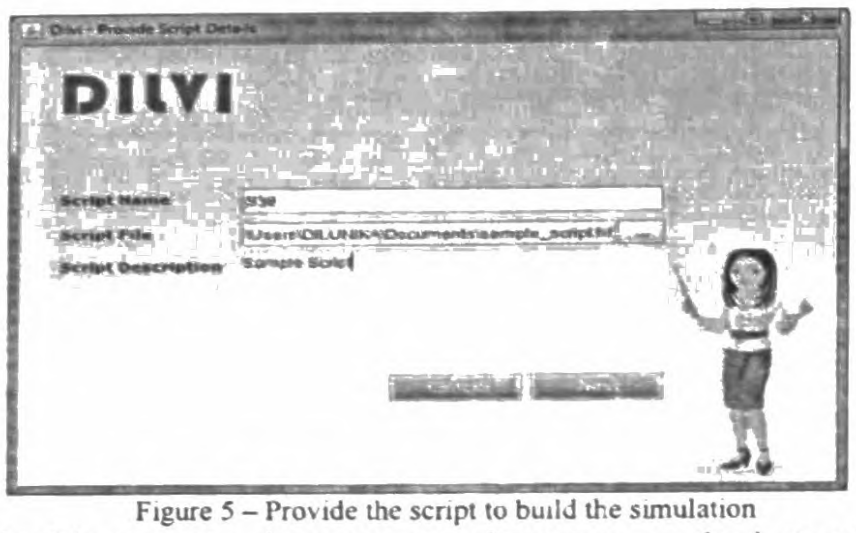

Figure 5 – Provide the script to build the simulation

In this example, there are two characters, so the instructor is prompted to select which is the student character. Based on the selection, system generates audio clips for the system character and again prompts the instructor to insert the hints for the dialogs of the user character. If the particular dialog phrase has already been tagged, the category is shown, otherwise instructor is asked to enter the nearly matching category to the phrase and also some supporting hints for that phrase. Figure 6 shows a screenshot of the dialog hint configuration window.

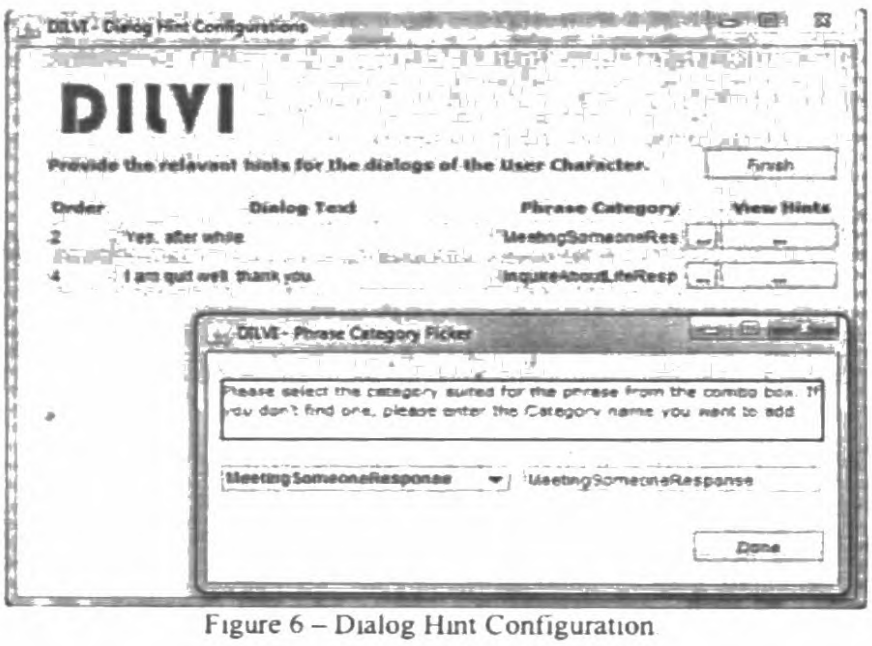

Figure 6 – Dialog Hint Configuration

After the meta-data is fed to the system addition to the script, DILVI builds a data structure called RolePlay and persist with the system. Stored RolePlay models can be loaded as simulation to be practiced. When a student selects a persisted RolePlay, system builds another data structure called Drama and uses Jmonkey Engine features to run the Drama as a simulation. Figure 7 shows such loaded simulation.

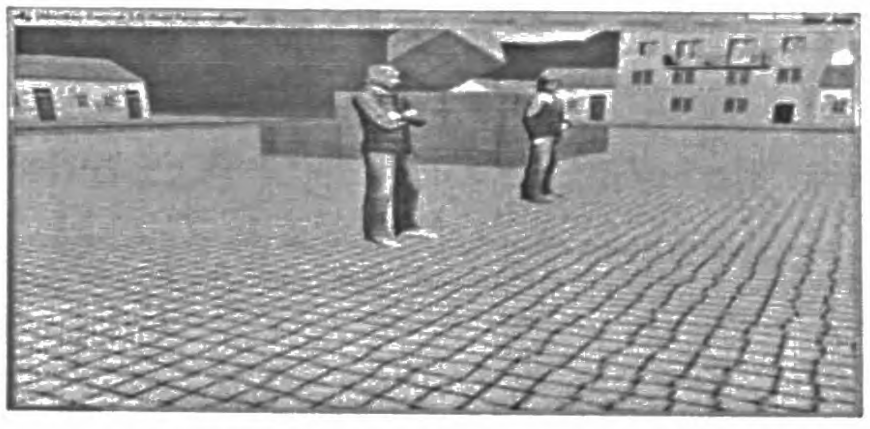

Figure 7 – Loaded role-play simulation.

As it is mentioned earlier, this script contains two characters, student and system, so the dialogs belong to system character are played automatically and student is prompted to input his or her dialog parts to the system through a microphone. Meanwhile, student is provided the hints related to the conversation which was entered during the role-play setup. Figure 8 shows such popup window which contains some speech part tips.

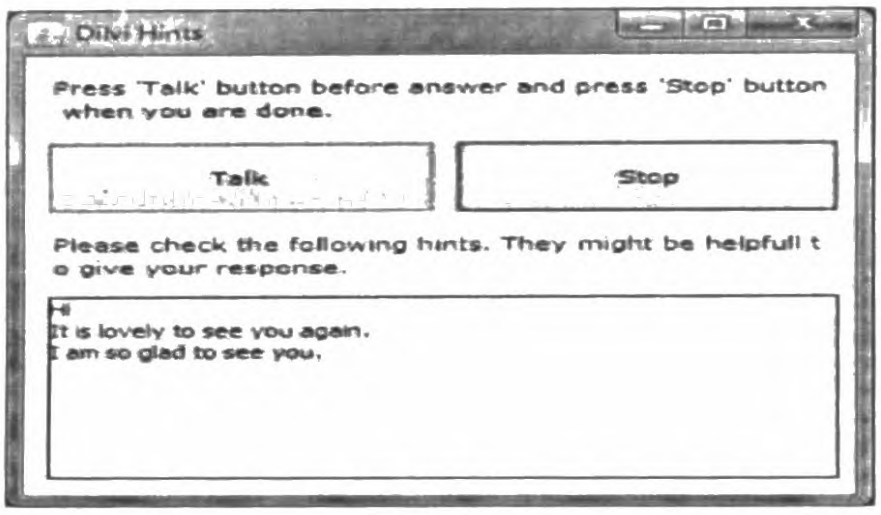

Figure 8 – Capturing student feedback after providing hints

IV. EVALUATION

Evaluation for the DILVI framework is carried out in four different perspectives: Effort Estimation, API quality, performance and user interaction.

A. Effort Estimation

A survey was planned in order to identify the effectiveness of DILVI framework in reducing the overhead of developing common components in simulation applications. This was done using a group of ten API users and the user group was selected in a manner which would capture developers ranging novice to expert.

The requirement case study was then presented and explained to the team which was required to be built. Once the users were completely aware of the requirement the team was then divided in to two groups of five each; group A and B. Group A was provided with a comprehensive overview and introduction about the DILVI framework while group B was provided an insight on the JMonkey and java technologies with respect to the requirement. However, group B was not provided with any insights on DILVI framework. Subsequently, both teams were asked to provide approximate figure for individual effort estimation on developing the said.

When it is analyzed given estimate values, it can be noticed that, introducing the DILVI framework has cut down the development cost roughly by seventy six persent (76%).

Average development effort without DILVI framework
= 51.0 (Man Days)

Average development effort with DILVI framework
= 12.4 (Man Days)

Effort cut down = ((51.0 – 12.4) /51.0) * 100
= **75.6 %**

978-1-4799-1276-6/13/$31.00 ©2013 IEEE

B. API Quality

As this is a qualitative analysis, authors decided to carry out a survey based on the three qualitative measurement concepts have been discussed in the book, "Practical API design", written by Jaroslav Tulach[18].

- *Comprehensibility* : Understandability of the API
- *Consistency*: If the users of an API have to invest time to learn a concept, it's important that the concept be applied consistently across the entire API
- *Discoverability:* It is important to create a single place that can serve as a starting point and can send people in the directions that solve their problems

The survey is carried out with the same set of users who participated in the previous survey. Before the evaluation, all the API users were given an introductory session about the framework.

Most of the users found it is more convenient about the concept of abstracting the game engine by applying the stage drama model which is more familiar than graphics terminologies, but requested improved documentation than provided. Code samples provided with the documentation has been use full while understanding the framework. But, users who were not having better knowledge about OSGi technology, found it is bit difficult while understanding the framework. Consistency of the code of the framework is not in a satisfied level. So it is highly required to have refactoring cycle before it has been released as alpha or beta release.

C. User Interaction

In order to prove the concept of the framework, a Proof of Concept (POC) application is developed using the DILVI framework. The POC application implements three extension points of the framework: Dialog to Speech module (DTS), Pedagogical Agent module and Speech to Dialog module (STD). Therefore, the POC is capable of parse a script file and build a simulated role-play with pedagogical hints and capture the student response and record it as an audio file while executing the generated role-play simulations.

A group of eighteen students have been selected as the sample group of evaluators. The group can be categorized into three sets based on their English proficiency in speaking as follows:

Category I: The selected set of students is good in writing and listing in English, however not confident and accurate enough when it comes to speaking.

Category II: Users who have never worked in an English speaking environment; however they are capable to operate the system with basic introduction provided.

Category III: Understanding and reading skills are also not proficient enough to operate the system; hence, it is required to guide them to drive the system while giving instructions in Sinhala (native language).

Experiment was carried out targeting one student at a time. Initially they were provided with an introduction about the importance and the requirement of such systems.

Major conclusion which can be inferred from the survey results is that majority of the students found the new software comfortable to speak to rather than normal role-play sessions which they attend generally. Most students felt less shy and quite comfortable to use the system and interact freely rather than being worried about whether others would laugh at you or not. Also a majority believes NLP software would provide a good stepping stone for them to learn.

However, the students did not find the quality of the speech smoothing as much as they would have expected. Even though, majority of the students found the use of hints as useful, it was observed from the results that the students who were familiar with the use of English language found it supportive while the students who had no experience in interacting English found the hints less useful. Finally, there were lots of complaints related to the HCI technologies used in the system.

D. Performance Analysis

Performance of the DILVI based POC application is analyzed in two different aspects: hardware usage and quality of graphics rendering. Experiment is planned for a role-play which contains twenty dialogs having the average length of 8 words. Twenty five iterations were considered and calculated the average to be presented as the summarized results.

Processor	Intel® Core™ i3 CPU M380 @ 2.53GHz
Installed Memory (RAM)	3.87 GB usable
System Type	64-bit Windows 7
Graphics Card	ATI Mobility Radeon HD 5650

Table 2 – Machine configuration used for performance test

Table 2 lists hardware configurations of the computer which is used for the analysis. It is observed that the systems' maximum usage of CPU and the RAM was while loading the simulation. During that time it uses 13% of the CPU and 38MB from the Memory. But, in average, system never uses CPU than 5% and memory than 20MB.

At the moment, with the prototype application developed, DILVI does not contain any advanced graphics, so the analysis showed that its FPS (Frames Per Second) rate is a very low value, averagely 3 and average number of triangles used in rendering was 21857.

V. Conclusion and Future Work

The research was carried out with the motivation of identifying and establishing an abstract framework which would include the fundamental boiler-plate implementation to develop language speech practice simulations. Figures which were derived using the survey results shed the insights about the time effectiveness, usefulness of DILVI.

2013 International Conference on Advances in ICT for Emerging Regions (ICTer) 978-1-4799-1276-6/13/$31.00 ©2013 IEEE

The system could be enhanced in the future in many ways which would allow ESL tactical learning system developers better equipped with an affluent framework as a stepping stone in creating an enthusiastic learning environment for students.

The quality of the graphics which was used in the framework required to be higher as the students found the low quality graphics was a main concern. However, the underlying jMonkey engine is capable of facilitating much more advanced graphical features than the DILVI currently possess. In future, it is planned to include few more scenery setups like class room, a shopping complex and a ground which would provide more 3D illustration to the user.

Another future work which was important is the availability of tips in native languages. As the survey suggested, most of the students who were weak in English failed to grasp the activity properly.

After putting enormous effort on sphinx framework, it was removed from the scope from the initial phase as it did not yield the expected accuracy. So, it is necessary to improve the accuracy of the speech recognizing module as it is one of the major targets of the framework.

REFERENCES

[1] "Practice (learning method) - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Practice_(learning_method). [Accessed: 10-Nov-2011].

[2] J. Jia, "CSIEC (Computer Simulator in Educational Communication): A Virtual Context-Adaptive Chatting Partner for Foreign Language Learners," in *Proceedings of ICALT*, NY, 2004, pp. 690-692.

[3] Y. Shih, Y. Lin, and M. Yang, "The Development of an Online Virtual English Classroom: VEC3D," vol. 2, no. 2, pp. 61-68, 2007.

[4] J. Xiaoluo, J. Chang, and C. Lizhi, "Implementation of a project-based 3D virtual learning environment for English language learning," presented at the Education Technology and Computer (ICETC), 2010 2nd International Conference, Shanghai, 2010, vol. 3, pp. 281-284.

[5] "Virtual Worlds, Avatars, free 3D chat, online meetings - Second Life Official Site." [Online]. Available: http://secondlife.com/. [Accessed: 11-May-2011].

[6] "Main Page - OpenSim." [Online]. Available: http://opensimulator.org/wiki/Main_Page. [Accessed: 03-May-2011].

[7] G. Lepouras and C. Vassilakis, "Virtual Museums for all: Employing Game Technology for Edutainment," *Journal - Virtual Reality*, vol. 8, no. 2, Sep. 2004.

[8] J. Jacobson, "Using 'CaveUT' to Build immersive Displays With the Unreal TournamentEngine and a PC Cluster," presented at the ACM Symposium on Interactive 3D graphics, 2003.

[9] T. Rhyne, "Computer games and scientific visualization," in *Communications of the ACM*, vol. 45, 7 vols., 2002, pp. 40-44.

[10] "OGRE – Open Source 3D Graphics Engine." [Online]. Available: http://www.ogre3d.org/. [Accessed: 11-May-2011].

[11] "Unreal Technology." [Online]. Available: http://www.unrealengine.com/. [Accessed: 11-May-2011].

[12] "jMonkeyEngine.org | Home." [Online]. Available: http://jmonkeyengine.org/. [Accessed: 11-May-2011].

[13] M. Lewis and J. Jacobson, "Game Engines in Scientific Researshs," in *Communications of the ACM*, vol. 45, 2002.

[14] "Java SE Overview - at a Glance." [Online]. Available: http://www.oracle.com/technetwork/java/javase/overview/index.html. [Accessed: 09-Nov-2011].

[15] "Equinox." [Online]. Available: http://www.eclipse.org/equinox/. [Accessed: 09-Nov-2011].

[16] "OSGi Alliance | About / OSGi Technology. " [Online]. Available: http://www.osgi.org/About/Technology. [Accessed: 06-Nov-2011].

[17] "Chapter 3. The IoC container." [Online]. Available: http://static.springsource.org/spring/docs/2.5.x/reference/beans.html. [Accessed: 09-Nov-2011].

[18] J. Tulach, *Practical API Design: Confessions of a Java Framework Architect*. Apress.

978-1-4799-1276-6/13/$31.00 ©2013 IEEE