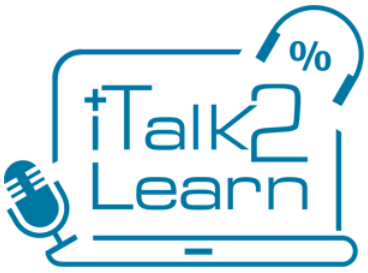


iTalk2Learn
2014-04-30

Deliverable 4.2.1

Initial technical report

30th April 2014

Project acronym: iTalk2Learn

Project full title: Talk, Tutor, Explore, Learn: Intelligent Tutoring and Exploration for Robust Learning

Work Package:	4
Document title:	Initial technical report
Version:	4.0
Official delivery date:	30/04/2014
Actual publication date:	30/04/2014
Type of document:	Report
Nature:	Public

Authors:	Jose Luis Fernandez (BBK), Sergio Gutierrez-Santos (BBK) with contributions from Norbert Pfannerer (SAIL), Carlotta Schatten (UHi), Ruth Janning(UHi), Matteo Romagnoli (Testaluna) and Beate Grawemeyer(BBK)
Internal Reviewers:	Matteo Romagnoli (Testaluna), Norbert Pfannerer (SAIL), Carlotta Schatten (UHi)

Version	Date	Sections Affected
1.0	19/03/2014	Initial version
2.0	16/04/2014	Second version: Schemes colored and new structure
3.0	24/04/2014	Third version: Contains more detailed explanation about speech recording process and some clarifications in other points.
4.0	28/04/2014	More info added to communication with subcomponents

Executive Summary

This deliverable describes the integration platform that is responsible of bringing together all components in the iTalk2Learn system as well as the two different kinds of learning content (structured vs exploratory). The integration platform integrates both kinds of content, developed using different technologies (Flash, HTML5, Unity) to provide a seamless learning experience to students.

This document includes an installation manual.

The technical innovation of this platform lies not only on the fact that it allows the integration of both structured and exploratory learning activities, but also in integrating them with the intelligent components in the system: the use of speech (recognition and synthesis), the definition of different sequencing strategies, and the provision support and feedback for students. Additionally, its open service-oriented architecture facilitates future extensibility.

As general features, it is based on the Service Oriented Architecture (SOA) paradigm. It has an interface developed in HTML5 at the presentation layer, to make it available to most popular devices nowadays and support an easy integration with other features.

Table of Contents

Executive Summary 3

Table of Contents 4

List of Abbreviations 6

List of Figures 6

1. Introduction 7

2. Components diagram 9

2.1 Presentation Layer 10

2.2 Speech production 14

2.2.1 Technologies involved 14

2.2.2 Speech production process 14

2.3 Security Layer and LDAP 15

2.4 Application Layer 16

2.5 Aspects 17

2.6 Service Layer (Machine learning, Audio-based Difficulty Classifier, and Speech recognition integration) 18

2.7 Data access layer and ORM 19

2.8 Database (Maria DB) 20

3. Communication with sub-components 23

3.1 Fractions Tutor (CTAT - Cognitive Tutors Authoring Tools) 23

3.2 MathWhizz Exercises 25

3.3 Fractions Lab (for exploratory activities), task dependent support and task independent support 25

3.4 Sequencing engine (Machine Learning) 26

3.5 Audio-based Difficulty Classifier 28

3.6 Speech recognition (SAIL software) 28
4. Wizard-of-Oz tools 30
5. Future work 31
6. Conclusions 32
References 33
Appendix A - iTalk2Learn Installation Manual 34

List of Abbreviations

SOA	Service-oriented architecture
ORM	Object-relational mapping
DAO	Data access object
RIA	Rich Internet application
HTML5	Hypertext marked language version 5
AJAX	Asynchronous JavaScript And XML
HQL	Hibernate query language
JNI	Java native interface
DTO	Data transfer object
JSON	Java script object notation
ECMAScript	Scripting language standardized by Ecma International in the ECMA-262 specification and ISO/IEC 16262.
CSS	Cascading Style Sheets
XML	Extensible Markup Language
XHTML	Extensible HyperText Markup Language
XMPP	Extensible Messaging and Presence Protocol
ELE	Exploratory Learning Environment
JAX-WS	Java API for XML Web Services
LDAP	Lightweight Directory Access Protocol

List of Figures

Figure 2	iTalk2Learn architecture main diagram
Figure 2.1	iTalk2Learn presentation layer diagram
Figure 2.2.2	iTalk2Learn speech production process
Figure 2.3	iTalk2Learn security diagram
Figure 2.4	iTalk2Learn application context diagram
Figure 2.6	iTalk2Learn services layer diagram
Figure 2.7	iTalk2Learn data access layer main diagram
Figure 2.8.1	iTalk2Learn model to get exercises
Figure 2.8.2	iTalk2Learn model to storage exercises
Figure 3.1	iTalk2Learn fractions tutor process diagram
Figure 3.4	iTalk2Learn machine learning sequence engine diagram
Figure 3.6.1	iTalk2Learn speech recognition process diagram
Figure 3.6.2	ITalk2Learn speech recognition communication engine

1. Introduction

The iTalk2Learn platform is a web-based platform that allows the deployment of a robust tutoring system. Web applications are very popular in all settings, but in particular in educational contexts due to three factors: (a) they are more secure than applications installed on the operating system, (b) they do not require any installation or maintenance, and (c) they can be kept updated to the latest version without any effort from final users or school administrators.

iTalk2Learn is a learning platform for children aged between 8 and 12 years old. It is an open source platform which integrates structured practice and exploratory, conceptually-oriented learning. It facilitates interaction in different modalities including speech, as well as multiple representations which can be manipulated and reasoned with when learning fractions.

This document describes the initial technical report of the iTalk2Learn platform. The current prototype has been used to collect an initial corpus of child/tutor interaction for WP1, and to prove how the components developed in WP2 (i.e. recommender system for sequencing, intelligent support for exploratory activities) and WP3 (speech recognition) can interact together. It currently being used to collect additional data for WP1 and to inform decisions that relate to the evaluation of the learning outcomes of the project (WP5).

This initial prototype has been designed and developed with the aim of providing a generic and flexible robust learning platform able to deploy structured and exploratory learning activities to students. It is based on a Service Oriented Architecture (SOA), allowing a high level integration of all different components oriented to consume exposed services. This approach simplifies future maintenance while allowing for:

- accessing channels to provide interaction within the system,
- gathering student—tutor and student—student interactions, use of exercise descriptors that associate problems and activities (exploratory or not) with competencies involved in their solution and general metadata,
- use of solution strategy descriptors that associate problems with typical steps employed by humans working towards a solution (correct ones as well as misconceptions),
- development of learner model builders that aggregate student/tutor interactions to build learner models, and adaptive control of single-student tutoring, e.g., selecting the next problem/activity to pose that allows the student to learn most effectively, selecting interventions to support the student most in his/her current situation.

The platform combines different technologies, including speech recognition and speech production, with a range of learning systems, such as Whizz, Fractions Tutor, and the exploratory learning environment Fractions Lab. The main aim is to define the interfaces for the different elements of the platform and their interplay. These elements include the learning content developed within the different learning systems; the intelligent components that will be developed in our project, including the recommender and the intelligent support; and the intuitive interaction features, including speech. Additionally, data storage of the relevant information is included in the platform. The overall goal is to provide a flexible and scalable infrastructure for the elements of the platform, which are being developed in such a way that those elements can be exchanged independently of each other and, eventually, be replaced by other elements in the future, providing a test bed for future technologies.

The rest of this report describes the components of the platform and the technologies employed at each level (Section 2) as well as how the different components of the iTalk2Learn system are integrated (Section 3), including the different learning content sources from WP1, the intelligent components from WP2, and the speech recognition system from WP3; additional facilities (the wizard-of-oz tools) are described on Section 4. Section 5 describes some strands for future work and improvement. The report closes in Section 6 with come concluding remarks. Additionally, an installation manual is added as an appendix.

2. Components diagram

A diagram with the main components of the iTalk2Learn architecture is presented in Figure 2. The following sections will explain each layer, technologies and its components in detail.

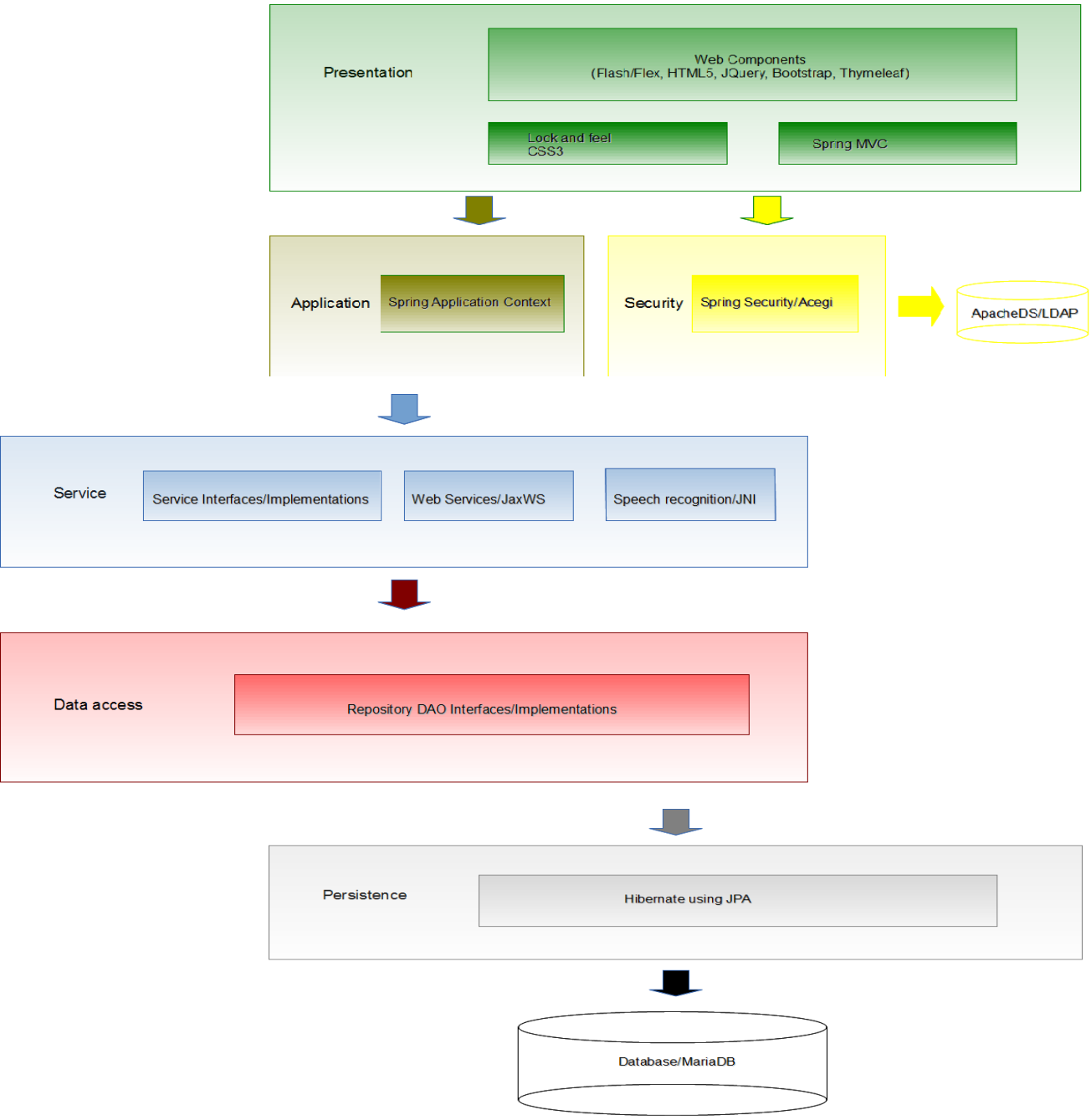


Figure 2. italk2learn architecture main diagram

2.1 Presentation Layer

The user interface of this initial prototype is based on HTML5 (Hyper Text Marked Language v5) [1] and Flash [3]. Although the use of Flash technology introduces some limitations into the system (e.g. working with tablets), this has been impossible to avoid because of two reasons: (a) on the one hand, Flash technology is used by several of the learning content elements that are part of iTalk2Learn, and this could not be changed; (b) on the other hand, HTML5 technology is not as developed as Flash for some specific use cases (e.g. voice recording). Our future plan aims at gradually reducing our reliance on Flash, as described later in this document.

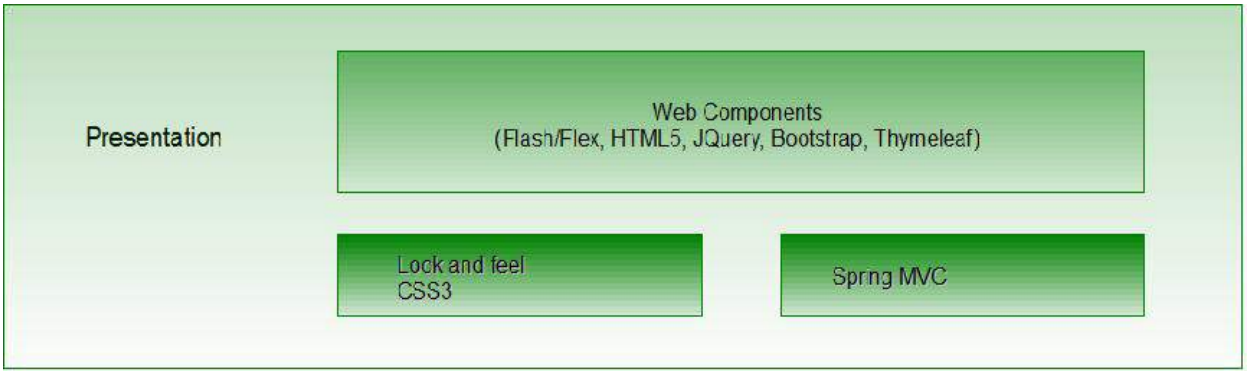


Figure 2.1 iTalk2Learn presentation layer diagram

Our Presentation Layer is cleanly divided according to a Model-View-Controller (MVC) pattern. This simplifies maintenance and, in particular, facilitates interaction with project partner Testaluna to produce the final user facing components of the iTalk2Learn system*. A view requests from the model the information that it needs to generate an output representation to the user, while the model notifies its associated views and controllers when there has been a change in its state. This notification allows views to produce updated output and controllers to change the available set of commands. A controller sends commands to its associated view to change the view's presentation of the model. It can also send commands to the model to update the model's state. We have a controller for each view to make it easier to add functionality and keep a clear separation of concepts. There are currently two student-related views (authentication, sequencing and display of exercises) and an additional view for the wizard-of-oz tools (see below).

In order to integrate all different facets of the user interface we have had to use a series of different technologies. These technologies, and the role they play in the UI layer of iTalk2Learn are described below.

** Process described in WP3, Task 3.7 by Testaluna*

- **Adobe Flash (AS2)**

Adobe Flash is a multimedia and software platform used for the authoring of vector graphics, animation, games and Rich Internet Applications (RIAs) which can be viewed, played and executed in Adobe Flash Player. It provides tools to facilitate the development of exercises and creates a better interaction with the user. Action Script (sometimes referred to as AS) is the programming language used to develop applications with Flash. The most recent version of the language is Action Script 3 (AS3).

The main language used by Whizz exercises is Flash / Action Script 2 (AS2). Most of their exercises are based on Flash technology (versions 5 and 6). For this reason the platform integrates Flash.[1]

- **HTML5**

HTML5 is a combination of markup languages and associated technologies for structuring and presenting content for the World Wide Web. It is a core technology of the Internet today and its importance grows stronger every day. HTML5 is the fifth revision of HTML standard (created in 1990 and standardized as HTML 4 as of 1997) [2]. The effort for HTML5 aims at improving the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices (web browsers, parsers, etc).

One of the main purposes of the iTalk2Learn platform is to provide a container of exercises. This container is developed in HTML5 (Hyper-Text Marked Language, version 5) and CSS (Cascading Style Sheets), and provides a simple wrapper around the exercises presented to students enabling them to communicate with the rest of the system.

- **JavaScript**

A crucial part of HTML5 is Javascript (sometimes referred to as JS or as per its standard name, ECMAScript [4])[2]. JavaScript is an interpreted computer programming language. It was originally developed at Netscape and implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. Nowadays it is running on every web-browser and enables web browsers to become application development platforms like any operating system in the past; however, the security model is

1 *At the time of writing, Whizz has started to generate new exercises based on HTML5. BBK and Whizz are working closely to integrate these new exercises into iTalk2Learn.*

2 *For the purposes of this document, we will consider JavaScript and ECMAScript as synonims.*

more restrictive (and therefore applications are more secure) and users are not required to install (or maintain) anything on their computers, which makes web applications more very popular.

To add extra functionality to HTML5 that enriches the user experience we have used JavaScript, as well as several associated technologies, which are explained below.

- **Unity3D**

Unity is a multi-platform game engine and authoring system developed by Unity Technologies. It is used to develop video games and 3D graphics applications for web, desktop, consoles and mobile devices. The graphics engine uses Direct3D (Windows, Xbox 360), OpenGL (Mac, Windows, Linux), OpenGL ES (Android, iOS), and proprietary APIs (consoles). The game engine's scripting is built on Mono, the open-source implementation of the .NET Framework. Programmers can use UnityScript (a custom language with ECMAScript-inspired syntax, referred to as JavaScript by the software), C#, or Boo (which has a Python-inspired syntax).

The technology used by the Exploratory Learning Environment (ELE) Fractions Lab is Unity3D, which is also used to develop the Task-Dependent Support for exploratory tasks running on Fractions Lab to have straightforward communication with the Fractions Lab interface. Both the ELE itself and the support for the tasks are developed in C# on top of the Unity Engine API. For this reason the platform integrates Unity3D. (More information about the support for exploratory activities can be read on D2.2.1).

- **Apache Tiles**

Apache Tiles [7] is an HTML templating framework based on the "Composite" paradigm. It allows for the HTML page to be broken up into multiple pagelets, called Templates, Definitions and Composing pages. At run time the pagelets are stitched together to generate the final HTML that is sent to the client browser.

In order to get the same look-and-feel in all views we reuse standard layouts and use property inheritance. For example, as the header, footer and options bar presented by all exercises is common, the system inherits them from a common template rather than defining them on every page. This makes the system easier to manage and more maintainable.

- **Thymeleaf**

Thymeleaf [8] is a Java XML/XHTML/HTML5 template engine that can work on both web (servlet-based) and non-web environments. It is better suited for serving XHTML/HTML5 at the view layer of MVC-based web applications, but it can process any XML file even in offline environments. It provides full Spring Framework integration.

In web applications Thymeleaf aims to be a complete substitute for Java Server Pages (JSP), and implements the concept of Natural Templates: template files that can be directly open in browsers and that still display correctly as web pages. JSP has been a very popular technology for integrating web content

and programming functionality, but it has two serious limitations. First, it works synchronously, meaning that any change in the page requires reloading it from the server (which can be a serious disadvantage with weak internet connectivity and has a detrimental effect on the user's experience). Additionally, it does not cleanly separate programming code (Java) from presentation code (HTML). By using Thymeleaf we are introducing a clearer separation and pushing the view layer back to the HTML side where it belongs, allowing us to use our programming templates more actively for customer validation and allowing for an easy update without ever losing the information needed to process them dynamically at runtime.

- **JQuery Ajax (Javascript)**

JQuery [9] is a multi-browser JavaScript library designed to simplify the client-side scripting of HTML. We use the functionality provided by JQuery about Ajax (AJAX, Asynchronous JavaScript and XML). With Ajax, the platform can send data to -and retrieve data from- a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

Using JQuery in our main container view (the view that displays the exercises to the user) allows us to implement an interface that sends procedure calls to a services API at the server in order to retrieve an exercise sequence and put the next exercise on the view container, all of this happening seamlessly and without any apparent modification in the behaviour of the system from the learner's point of view. This was an important requirement given that the exercises to be used in the context of iTalk2Learn have been developed in different contexts and using different technologies. It is crucial that this does not affect the users' experience (and therefore, have a negative effect on their learning).

- **External interface (Javascript)**

External Interface [10] is an application programming interface that enables direct communication between Javascript, Flash, and Unity3D movies (in our case, exercises). In iTalk2Learn the main container is placed on an HTML page with JavaScript that uses Flash Player to display a *swf* file (e.g. a Whizz exercise) and Unity3D player to display a *unity3d* file (e.g a Fractions Lab exercise). Using External Interface, the system can execute a function from within the Flash/Unity3D runtime using normal JavaScript calls from the HTML page. The executed function will return a value and JavaScript receives it immediately as the return value of the call. It is important to note that without the External Interface the Javascript code and the Action Script or C# code (depending on the platform that is executing the code) would be running on two completely independent platforms (namely, the web browser and the Flash/Unity3D runtime environment) and would not be able to communicate.

Our interface implements a simple API to communicate with the exercises (Flash/Unity3D). It processes input and output data (Request-Response objects based on JSON) in order to be aware of what happens inside the exercises. This enables the communication with Flash/Unity3D exercises to retrieve information such as time spent in the exercise and number of correct answers. This information can be forwarded to the sequencing engine as described later in the document.

- **Spring MVC**

Spring [11] MVC provides a clear separation of layers inside the iTalk2Learn platform. This makes its source code more structured and organized so it will be easier to maintain and extend it according to the project future needs.

2.2 Speech production

2.2.1 Technologies involved.

In order to produce voice through text messages or strings of characters, iTalk2learn uses the following technologies:

- Google (Text to Speech)

Google (Text to Speech) is a technology provided by Google to convert written words into audio sending a string of characters through a HTTP petition. As a result of this petition it receives a chunk of audio that can be embedded on a html5 audio player.

To set the GET petition we have the following parameters:

q: The query string to convert to audio

tl: Translation language, for example, ar for Arabic, or en-us for English

ie: Encoding format, use default UTF-8

- HTML5 (Audio component)

One of the features provided by HTML5 is the audio component. It provides methods, properties, and events allow you to manipulate <audio> and <video> elements using JavaScript.

Using it is quite straightforward. The URL of the petition is received and then a very simple API can be used to reproduce songs. The main methods involved are:

load() Re-loads the audio/video element

play() Starts playing the audio/video

2.2.2 Speech production process

The following diagram depict in more detail how the process works:

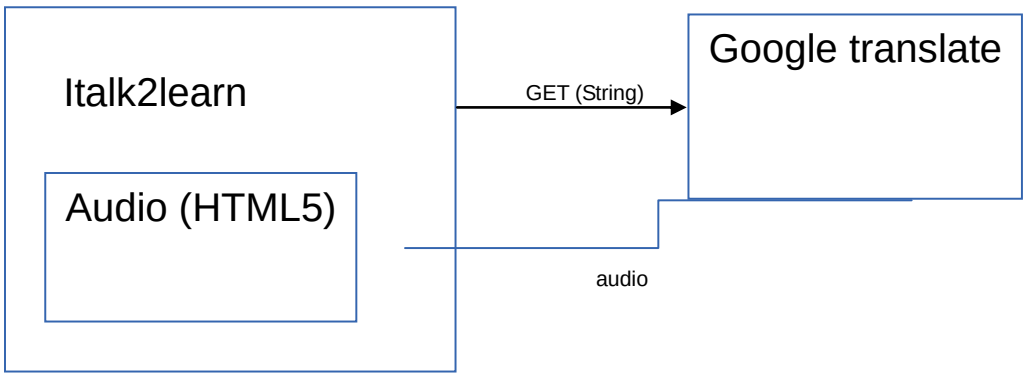


Figure 2.2.2 iTalk2Learn Speech production process

1. The iTtalk2learn sends a petition to Google translate service, e.g.:

http://translate.google.com/translate_tts?ie=UTF-8&q=hello&tl=en_gb&total=1&idx=0prev=input

The q parameter defines the string that will be converted in audio, in this case "hello".

2. The result is embedded on an audio component, which is contained inside of the iTalk2learn client layer. Using the javascript API, the audio is played.

Currently, there are two kinds of messages read aloud by the iTalk2Learn system:

- Task descriptions
- Support messages during the tasks, in particular XMPP messages from the wizard-of-oz facilities described later on Section 4.

Additional information about the use of speech production as part of the intuitive interface interaction in iTalk2Learn (including contingency plans for the unlikely risk that Google's service becomes unavailable before the project ends) is available on D3.4.1.

2.3 Security Layer and LDAP

The iTalk2Learn platform has a security layer that ensures that final users get access to the platform in a secure mode. It implements credential support. In the case that the user does not provide the required credentials (normally username and password) access to the platform and its services is denied.

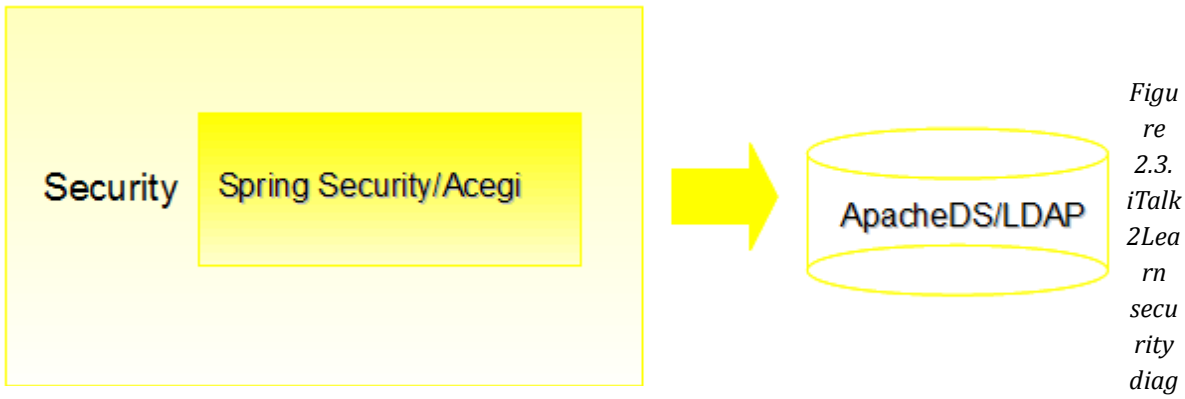


Figure 2.3. iTalk2Learn security diagram

To construct the security layer we have used LDAP (Lightweight Directory Access Protocol, a commonly used database oriented to users) and Spring security. On the security layer we can set different levels of security or view access depending on group, organization, and role of the user. We can restrict calls to services to ensure that a user without the right credentials will not be able to compromise data on the database.

The technologies we have used are:

- **ApacheDS and LDAP**

We have used ApacheDS [12] as the implementation of LDAP to be used in iTalk2Learn. LDAP is an application protocol for accessing and maintaining distributed directory information services.

Directory services may provide any organized set of records, often with a hierarchical structure, such as a corporate email directory or telephone directory. In our case, we store information about users' credentials such as identifier (ID), password, organization, and role (e.g. teacher, student).

- **Spring Security**

Spring [11] Security is a Java/Java EE framework that provides authentication, authorization and other security features for enterprise applications. We will use it to retrieve and manage the users' credentials on the platform, including tasks such as logging information interchange in the platform (between the different components) and showing different parts of a view.

2.4 Application Layer

Inside this layer all exposed services are placed (interfaces only). The layer is divided in business units exposed using a Restful architecture model. Every business using or service implements a CRUD (Create, Read, Update, Delete) interface. While this model is in a way more restrictive than a pure service oriented approach (e.g. RPC), the fact that it is simpler to understand makes it more adequate in our case, as we aim for an approach that facilitates development and integration of services developed by other partners in the project and in the future.

This layer is responsible for ensuring that the Spring core works adequately. It contains database initialization scripts, data sources descriptions, JNDI, Spring MVC settings, Apache Tiles settings, services initialization, controllers, and data access objects initializations.

Figure 2.4 iTalk2Learn application context diagram

This layer serves to uncouple the code with other layers (i.e. implementation services and view). By reducing the overall coupling of the system, we make sure that future scalability is not compromised by past development, that is, that adding new functionalities in the future is easier and faster. The layer is developed using Spring and it provides an initialization container for dependency injection. This process is called inversion of control, and aims at four goals:

- Decoupling the execution of certain tasks from their implementation
- Allowing modules to focus on what they are designed for
- Modules do not need to make any assumptions about what other systems do internally (just rely on their contracts, i.e. exposed interfaces)
- Replacing modules has no side effect on other modules

2.5 Aspects

This is an intermediate layer that standardizes structures and works as a shell of the Service Layer to add extra functionality. As we will work with input/output structures (i.e. Request – Response Objects) it is paramount to guarantee that these structures are not inconsistent and data is properly received. In other words, this layer acts as a common shell for all services or business units.

Request/Response objects are used to standardize data interchange in the platform and externally. These data structures have a header with common data of all services, to store e.g. user, score, time, etc; and a log to store whether there was an error on the platform (facilitating tracing errors back to the source).

2.6 Service Layer (Machine learning, Audio-based Difficulty Classifier, and Speech recognition integration)

This layer contains all services' implementations or business logic described in the application layer.

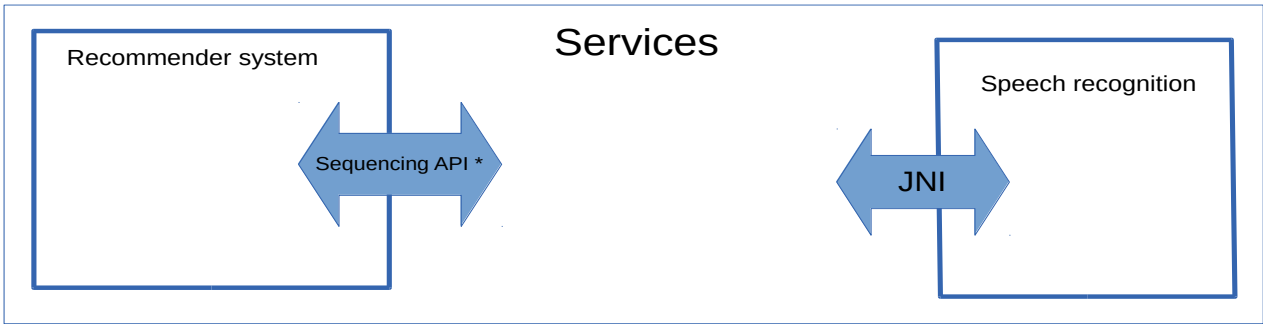


Figure 2.6 iTalk2Learn services layer diagram

This is the layer where the implementation of several important modules of iTalk2Learn is placed. Examples of services whose implementations are placed here include the sequencer (implementing the algorithms that calculate the next exercise to be presented to the user, more information on D2.2.1), the management of exercises' scores, the Audio-based Difficulty Classifier, the speech recognition software (that transforms a sound waveform into a transcription of speech, more information on D3.4.1). In the future, any other functionality to be added will be here.

This layer is responsible for converting data structures to be stored on the database (DTO) and the data structures to be shown on views (i.e. value objects). This design pattern is called T*ransfer Object Assembler* and is used to improve network performance. This pattern is most useful when the client needs to obtain data for the application model or part of the model. The Transfer Object Assembler pattern builds a composite transfer object and returns it to the client (as a value object).

Within the current prototype, the main client functionality (programs/libraries) to be placed on this layer are the speech transcription module (developed by SAIL, WP3), the recommender system that acts as a sequencer (Uhi, WP2) and the Audio-based Difficulty Classifier (Uhi, WP3). Integration and data interchange with these modules is explained in more detail later in this document.

Main technologies involved in this layer, which use java as the main language are:

- **JaxWS/ Webservices**

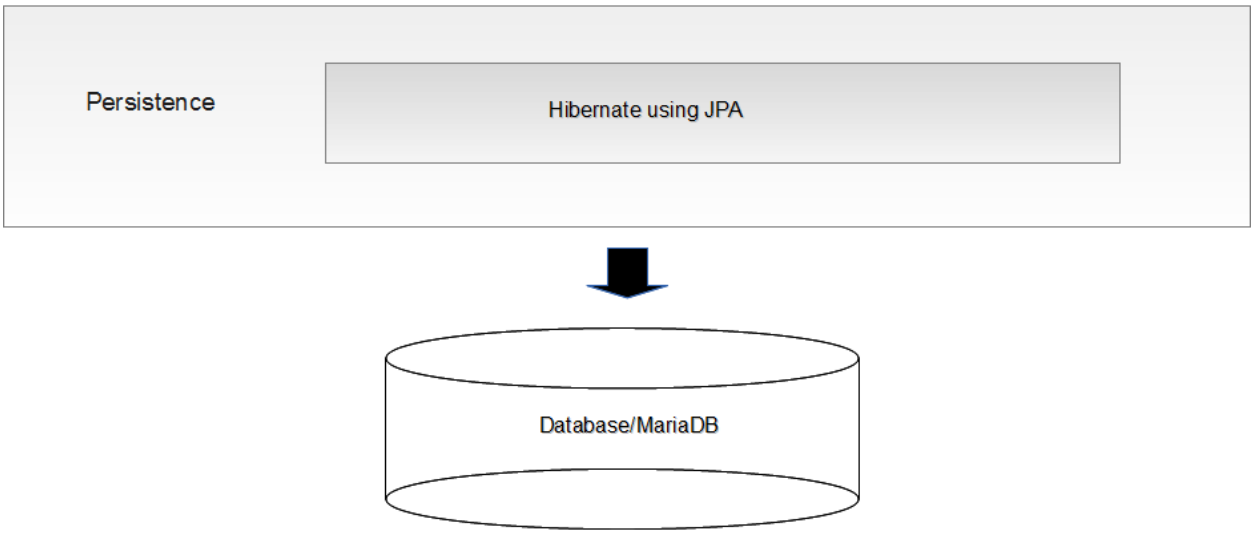
To provide straightforward communication with the recommender system UHi provides a web service. It requires a communication implementation from the services layer in order to manage input/output data between both places (iTalk2Learn-UHi). This implementation is done using JaxWS, which is a Java API for creating and communicating web services.

- **JNI**

Java Native Interface provides an API to communicate Java with applications based on C++. Speech recognition engine is developed in C++, so iTalk2Learn provides this communication mechanism.

2.7 Data access layer and ORM

This layer contains all implementations for the creation of new data, access, modification, or deletion of data on the database (based on MariaDB). For this purpose we have used an API developed in Java called JPA (Java Persistence API) in combination with an ORM (Object-Relational Mapping)



framework. An ORM creates a "virtual object database" that can be used from within the layer.

Figure 2.7. iTalk2Learn data access layer main diagram

For this proposal we will use the next technologies:

- **JPA**

JPA [14] is a Java programming language framework to manage relational data in applications using Java. The main features include: expanding object-relational mapping functionality, adding support for collections of embedded objects, linking the ORM with a many-to-one relationship, adding multiple levels of embedded objects, order lists, providing a combination of access types, and a criteria query API to standardize queries.

- **Hibernate (ORM)**

Hibernate provides an open source object-relational mapping framework for Java. It also provides a template with JPA methods to access the database that we will extend with our own access methods.

2.8 Database (Maria DB)

In order to collect and manage the data required for this research project, the platform uses a database layer based on MariaDB. All data storage of iTalk2Learn will be on this layer. MariaDB is the continuation of the MySQL project, a well-known open-source database. MariaDB is the project that forked out from MySQL when Oracle bought MySQL out of concerns about the future of the project (Oracle flagship product is a direct competitor or MySQL), and has improved and extended MySQL's features; nowadays, most of the core developers of MySQL have moved into MariaDB. It combines perfectly with Hibernate using a MySQL dialect.

The database collects all data in the system, including information about all exploratory learning activities and structured exercises (e.g. the score of an exercise, duration of exercise), and the sequence of exercises. This information can be used by any component of the system, from the sequencer to the intelligent support subsystems. The current data model describes users and sequences of exercises, as depicted in the following entity-relationship model:

- Model to to get sequences of exercises

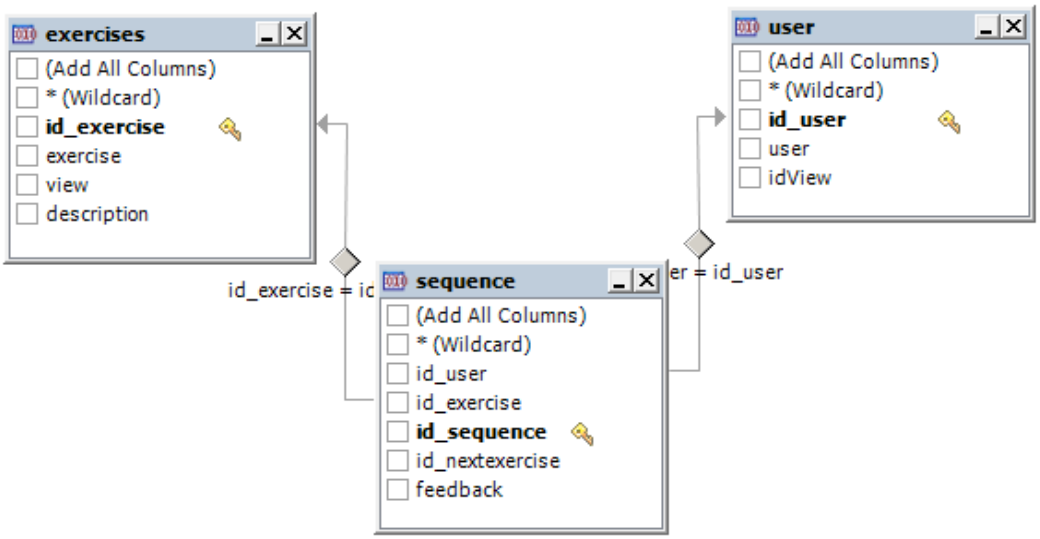


Figure 2.8.1. iTalk2Learn model to get exercises

This model contains two entities and one relationship.

Exercise (Entity):

- id_exercise: the exercise identifier. Each exercise will have a distinct identifier to identify and distinct of other exercises.
- exercise: the name of exercise.

User (Entity):

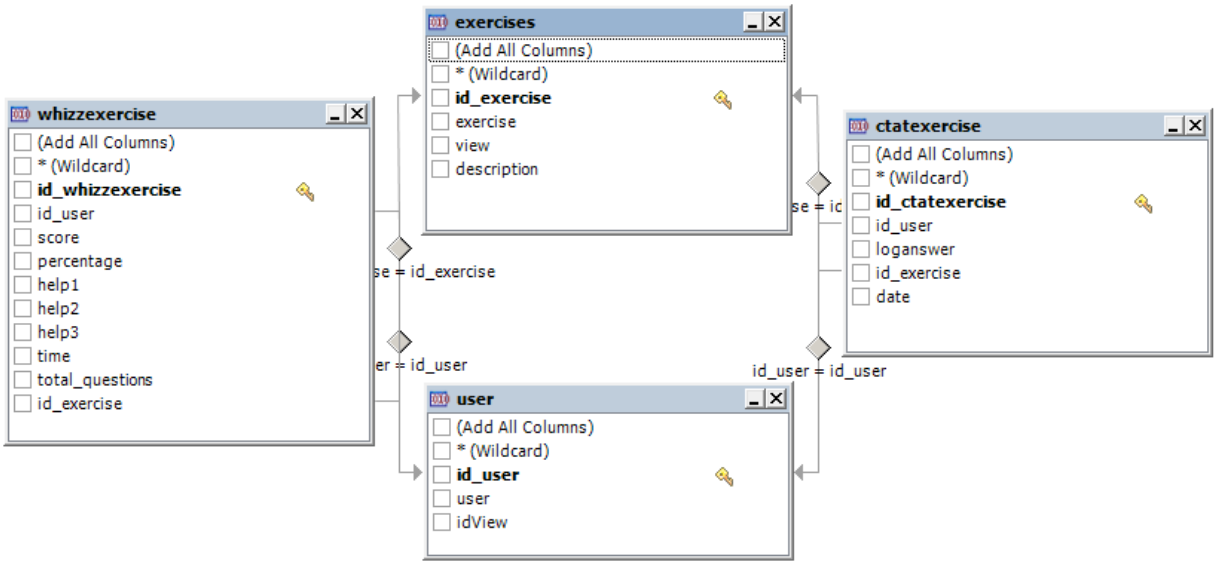
- id_user: the user identifier. Each user will have a distinct identifier to identify and distinct of the other users.
- user: the user's name, the same user as stored on LDAP.

Sequence (Relationship):

- id_exercise: points to an *exercise entity* to get the exercise.
- id_user: points to a *user entity* to get the user.

- Model to store data

The iTalk2Learn platform stores information from both the structured and exploratory tasks for



posterior analyis. Data storage for structured tasks is based on the following model:

Figure 2.8.2. iTalk2Learn model to storage exercises

WhizzExercise (Relationship):

- id_exercise: point to an *exercise entity.*
- id_user: points to a *user entity.*
- parameters: the main data that can be extracted from the exercise.
- date: date on which the user tried the exercise (task).

CTATExercise (Relationship):

- id_exercise: point to an *exercise entity.*
- id_user: points to a *user entity.*
- loganswer: log that contains data related to the exercise
- date: date on which the user tried the exercise (task).

3. Communication with sub-components

This section explains in detail how the iTalk2Learn platform communicates with the other components and subsystems at both the content and the system level.

At the learning content level, there are three main sources of learning content in iTalk2Learn: the Fractions Tutor from Carneggie Learning, MathsWhizz from partner Whizz, and the Exploratory Learning Environment (ELE) Fractions Lab (for exploratory activities) developed in WP3, Task 3.6 by Testaluna. This document will describe how the Fractions Tutor, the Whizz exercises and Fractions Lab have been integrated in the system. This was one important challenge in this project as Fractions Tutor and the Whizz exercises were developed a long time ago, with a completely different context in mind, and using different (and, to a point, incompatible) technologies.

At the system level, there are three main modules that need to be integrated: the speech-to-text module that recognises speech and provides a transcription, the recommender system that decides on the sequence of exercises (sequencer), and the Audio-based Difficulty Classifier.

3.1 Fractions Tutor (CTAT - Cognitive Tutors Authoring Tools)

The Fractions Tutor is one of several tutors developed with the Cognitive Tutors Authoring Tools (CTAT). The visual exercises of CTAT are developed using either Flash or Java-AWT.

In order to be able to show these exercises in our platform, we have used *swfobject* (developed in JavaScript). Swfobject offers a JavaScript API that aims to provide a complete tool set for embedding SWF files and retrieving Flash Player related information. This allows us to embed the CTAT Flash Movie Clip (swf) in our HTML view.

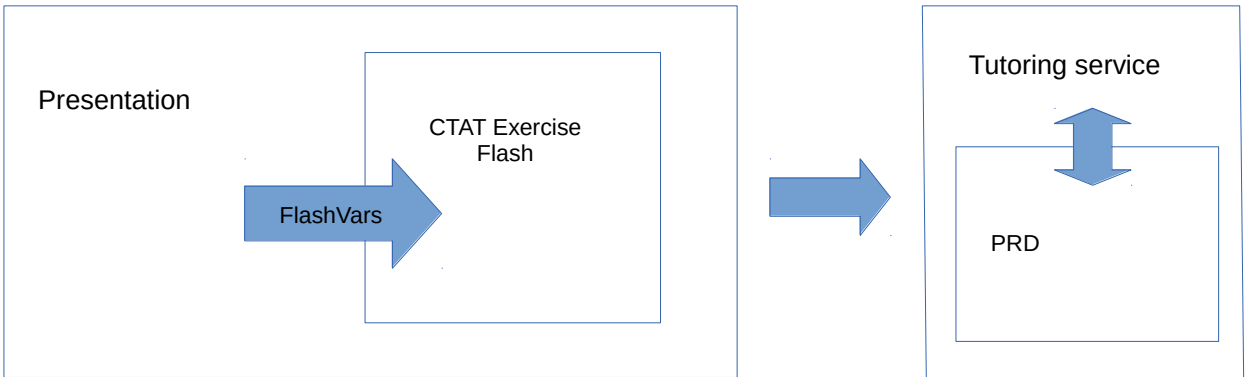


Figure 3.1. iTalk2Learn fractions tutor process diagram

In order to CTAT to work properly it is necessary to start a service called Tutoring Service. Tutoring Service is developed in Java, and acts as an interpreter of PRD files. PRD files are XML files that describe the exercise behavior, how it works, and how a user can interact with this. Tutoring Service can read PRD files, interpret their behavior and then open a socket to communicate with the exercise; at that point, the exercise knows what needs to be shown, the hints, the components, the solution, etc.

Given that the exercises are provided "as is" at this point, and they cannot be modified, we must treat them as black boxes without changing their logical behaviour.

To extract information/data from the exercise, CTAT provides a logging system named Tutor Message v4 to evaluate, implement, or update. The platform gets the logs from the exercises, which contain all information related to the exercise (e.g. hints, current state, exercise submitted).

The behaviour of CTAT exercises showed in the PRD file works as a state machine. Depending on the state that we are in the moment, we can extract four distinct types of log:

- Attempt: a student's try at a step in the tutor, with the selection, action, input details of the student's entry
- Result: the tutor's response to the attempt, with action evaluation (Correct or Incorrect)
- Hint request: the student's request for a hint, sent when the student presses the Hint button
- Hint message: the tutor's response to a hint request, containing the text of the hint

Depending of the type of log we can get the following information:

- Type of log
- Date time of the exercise
- Name of the exercise
- Action of the exercise (ex. done button pressed)
- Result of the exercise (ex. correct or incorrect)
- Number of the step in the exercise (ex step 17)

All of this information is stored on the iTalk2Learn database, including date and user that was performed the exercise. This information is available for further study later in the project.

3.2 MathWhizz Exercises

Whizz exercises are developed in Flash and HTML5. Most exercises are developed using Flash AS2, while the newer ones are developed with HTML5 in an effort to extend the company's market to new devices that do not accept Flash (e.g. tablets).

As it was the case with the Fractions Tutor, the exercises from the MathsWhizz tutoring service are provided "as is" at this point, and they cannot be modified. Therefore, we must treat them as black boxes without changing their logical behaviour. Project partner Whizz provided a shell to enable communication with their exercises, based on the shell that the MathsWhizz system uses. The shell is developed in Flash and it is basically a container of exercises with a bit of extra functionality. The shell handles the loading of exercises and tests and provides event handlers for the lesson data. Data that can be accessed from inside the exercises include: the current score of the exercise, the current percentage of solved questions, the number of help requests (levels 1, 2, and 3), the current elapsed time, the total number of questions in the exercise, and the current question number. Whizz exercises all contain a variable number of questions, usually between 5 and 15.

The shell is embedded in the iTalk2Learn's platform main container by means of *swfobject*, *as* a Flash Movie Clip (swf) as described previously on CTAT process. This Javascript API contains methods to display correctly the exercise.

We use external interfaces described previously in client layer point to call methods sending data in an intercommunication from the exercise (Flash) and Javascript (our main container is developed in HTML5, which include Javascript support). The Javascript code running on the browser processes this information and sends it to the platform in order to be stored according to the project needs. The data is stored on the iTalk2Learn database to be available for further study.

3.3 Fractions Lab (for exploratory activities), task dependent support and task independent support

One important aspect of Talk2Learn is the integration of exploratory, conceptually-oriented tasks. In task 3.6 of WP3 has been developed an Exploratory Learning Environment (ELE) called Fractions Lab (WP3, task 3.6) on which to perform these tasks. This software is developed on Unity3D (previously discussed in point 2.1).

In order to be able to show Fractions Lab exercises in our platform, we have used unityobject2 (developed in JavaScript). The unityobject2 component offers a JavaScript API that aims to provide a complete tool set for embedding unity3D files and retrieving Unity3D Player related information. This allows us to embed the Fractions Lab Unity Object (unity3D) in our HTML view. This process is quite similar to Flash integration processing.

To get more straightforward communication between the ELE running the exploratory tasks and the task-dependent support[3] for such tasks, both are developed on top of the same framework (Mono, an open source framework based on .NET technology). In other words, both Fractions Lab and the support for the exploratory tasks are developed in C#.

We use external interfaces (described previously at the client layer point) to to allow the communication between the environment (Unity3D) and Javascript. The current methods of the developed interface include the possibility to send internal messages to the exercise and it is displayed as "high" or "low" messages inside the exercise*. To send this data we use JSON (Java Script Object Notation); the information is sent as a pair of key-values containing and string with the displayed message and the corresponding key with the parameter to show (High/Low).

Exploratory tasks are described by a Task Information Package (TIP), that includes information about such as the name and description of the task, as well as information about misconceptions, goals, and rules that are needed from the task-dependent support. When the next task is an exploratory one, TIP are retrieved from the database and sent to the FractionsLab Unity3D object through external interfaces, using key-value JSON parameters.

** This process is described in D3.4.1*

3.4 Sequencing engine (Machine Learning)

The machine learning[4] collects information about student actions, building student profiles and adaptive control for single-student tutoring. The machine learning will expose an API of services in order for the rest of the iTalk2Learn system to communicate with it. Implementation of services will be developed in Java (Service Layer) and will be exposed on the application layer as a Restful API.

As per the current prototype, the Sequencing API is as described in the following diagram. There is an interface allocated in the services layer to communicate with the machine learning following this API. The API is exposed through a Web Service, which is currently allocated on a server in UHi. It contains one method to drive the sequencing.

In order to do so, at the services layer we will use JaxWS (as discussed in Section 2.6) to communicate with the web service. JaxWS is a Java programming language API for creating and communicating web services. Being a Java API can be fully integrated on iTalk2Learn platform.

The following diagram shows how iTalk2Learn communicates with the machine learning and the main methods of the interface:

3 More information on how task-dependent and task-independent support for exploratory activities has been designed and implemented can be found on D.2.2.1.

4 More details about the API can be found in [17].

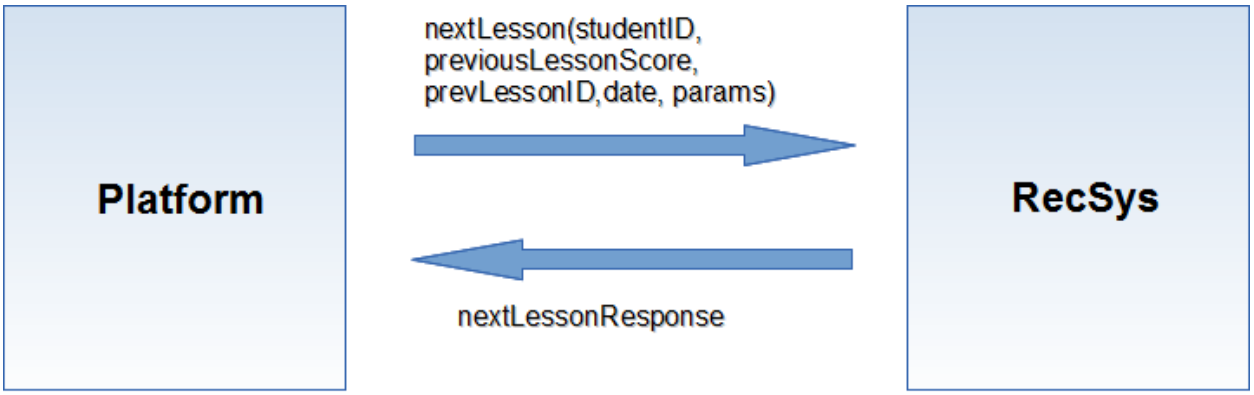


Figure 3.4. iTalk2Learn machine learning sequence engine diagram

- nextLesson: Check the students exists, updates the DB with the old data and it loads the required information from the DB according to the possible contents.
- nextLessonResponse: returns the next lesson.

Each method contains a structure with request/response objects as web service demands. Request object for nextLesson of current web service deployed in UHi, contains the following parameters:

- studentId: User identifier
- prevStudentScore: Previous student score of the exercise
- prevLessonId: Previous lesson identifier
- timestamp: Date and time that was performed the exercise.
- whizzLessonSuggestion: Suggested lesson by Whizz exercise.

The response object contains an output string with the result of nextLesson.

3.5 Audio-based Difficulty Classifier

The audio-based Difficulty Classifier is a piece of software module in charge of analyzing possible emotions detectable on the audio stream. It will extract some features from the wav files of students speech input. The output of this approach applied to the features will be an input for the sequencing engine.

This software is currently starting its development (WP3, task 3.4). It will be developed in Java and it will be fully integrated in the services layer of iTalk2Learn platform. No special efforts will be

needed to integrate it. It will be constructed as a part of services layer using all potential and characteristics that the platform provides, similarly to what has been described in the former section.

3.6 Speech recognition (SAIL software)

There are two main components involved in speech recognition process:

- ASREngine. It is responsible of the conversion from a soundwave into text (i.e. speech recognition). It is developed in C++.
- A client program that orchestrates the communication with ASREngine, starts/stops the engine, managez results on real time, and sends data to other components.

The software for speech recognition is developed in C++ while the rest of the iTalk2Learn system is based on Java and related technologies (with the exception of Flash, Unity3D and Javascript but only at the UI level). For that reason we need an interface to communicate with both layers. We have chosen Java Native Interface (JNI). This is a powerful Java framework that enables Java code running in a Java Virtual Machine (JVM) to call, and to be called by, native applications implemented (and compiled) in other programming languages (to a specific hardware and operating system platform). The JNI framework lets a native method use Java objects in the same way that Java code uses these objects. A native method can create Java objects and then inspect and use these objects to perform its tasks. A native method can also inspect and use objects created by Java application code.

JNI wraps the communication with the engine (C++) to be used as an API of Java. In our case we are using the following API:

- initSpeechRecognitionEngine: Starts the engine.
- sendNewAudioChunk: Send an audio chunk encoded as a ByteArray, meanwhile it's receiving data at real time.
- close: Closes the communication with the engine and retrieves the whole transcription as a XML file.

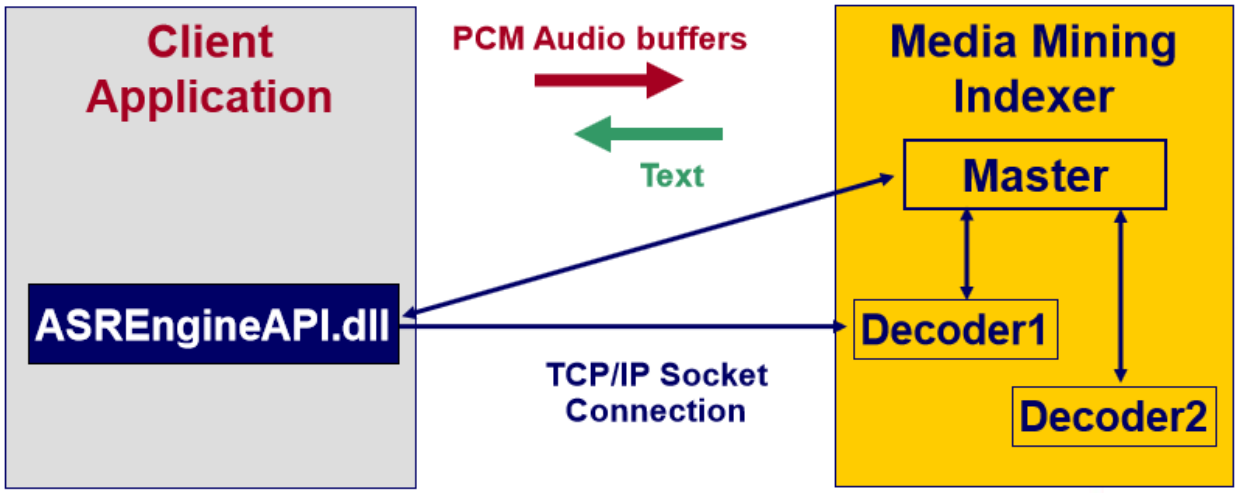


Figure 3.6.1 iTalk2Learn speech recognition process diagram

We have developed a client class in Java and it is allocated on the services layer (see description in former section). It will attend petitions of client layer through a service previously called. This java class uses JNI to communicates with the ASREngine library (.dll or .so) which in turn communicates with the ASR module.

The data flow in the system is as follows. A sound wave is recorded from the end user (i.e. student) at the UI, and this is sent from our module at the service layer encoded as a ByteArray to the speech-to-text component. Results of the transcription are received on real time, without need to terminate the whole process. These results or transcribed words at real time are ready to be sent to interested components (e.g. the sequencing engine, the AI components that provide support to students, task independent support) and stored on the database to be subject of study.

Finally, when the whole process finishes, a transcription of the speech contained in the soundwave is returned as an XML file. This XML file is then parsed and stored on the database.

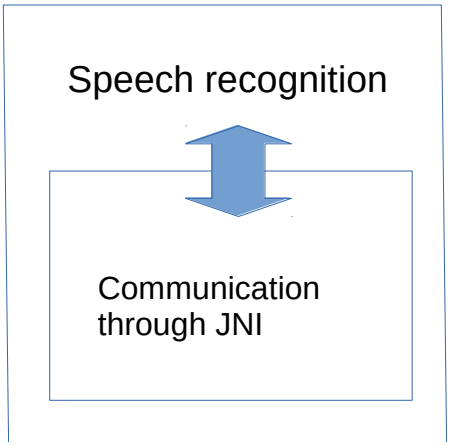


Figure 3.6.2 iTalk2Learn speech recognition communication diagram

4. Wizard-of-Oz tools

As part of the iterative development of the intelligent components of the iTalk2Learn system (speech recognition, adaptive sequencing, and intelligent support for exploratory activities) several Wizard-of-Oz studies have been scheduled in the context of iTalk2Learn in which human operators simulate the role of those components. The iTalk2Learn platform integrates a set of tools that provide real time communication between the student machine and the machine of a "wizard" making these studies possible. These tools enable wizards to change next exercise of a sequence of a determined user as well as sending text and voice messages to the students' machine, fully integrated on the activities that the student is performing in the moment.

To make possible these studies we have integrated the following technologies:

- XMPP server

Extensible Messaging and Presence Protocol (XMPP) is a communications protocol for message-oriented middleware based on XML (Extensible Markup Language). The protocol was originally developed for near real-time, instant messaging (IM), presence information, and contact list maintenance.

This technology is used to send messages that the student can receive in real time, fully integrated on task that it's performing in that moment. Wizards send messages by using a chat component on an interface with special tools. To send XMPP messages we use a javascript library called strophe.js as an XMPP client and OpenFire as the XMPP server. Both components are open-source and have had stable releases for a long time.

- Google (Text to Speech)

Google (Text to Speech) is a technology provided by google to converts written words into audio. A string of characters is sent via an HTTP request and a chunk of audio is received in responsel; this audio can be embedded on a html5 audio player.

This is used to read aloud the messages sent to the student by the wizard. The messages are listened as voice adapted to the student's actions, prompting the student to believe that he is working with an intelligent machine.

5. Future work

The current prototype implements enough functionality to support the rest of the project partners in the design of more advanced functionalities and the design of the pilot and evaluation studies to take place during the second year of the project. However, there are several strands of improvement that will be followed by project partner BBK (in some cases, in collaboration with other iTalk2Learn partners).

One of the main strands for future work is on integration of new sources of learning content. We plan to extend the capabilities of the platform to enable content based on several *de facto* standards including SCORM and inBloom.

We have plans to include a visual interface to provide teacher facilities, which include among other things:

- Providing access to information/data from speech transcriptions
- Providing access to information/data from exercises
- Create student accounts
- Upload / delete / replace contents
- Build a curriculum as "bag of contents plus choice of sequencer".
- Configure sequencers in curricula, e.g., set a fixed sequence
- Assign students to curricula
- See a summary of the data collected so far and download it.

We would like to make iTalk2Learn available for tablets users, which means that we will need to gradually reduce our dependency on Flash technology over time, given that the main reason for using Flash in iTalk2Learn are the exercises themselves. We will make those experiments for the sake of testing different exploitation paths.

Finally, we plan to make iTalk2Learn a portable platform that can be executed from a pendrive, further reducing the technical expertise needed to use an iTalk2Learn system in schools and favouring its dissemination and exploitation.

6. Conclusions

The iTalk2Learn project provides a modern and multifunctional platform with high level integration of different systems and sources of learning content. The platform uses a combination of different technologies to achieve these goals as described in the present document. This platform occupies a crucial position in this research project, so every effort has been made to achieve a highly robust system that does not compromise future scalability and changes to the system as the research progresses. Although there was a risk at the beginning of the project that not all components could have been integrated, at the time of writing all the different parts (learning content and subsystems) have been successfully integrated with the only exception of the Audio-based Difficulty Classifier which is currently under development. The risk of not being able to integrate this last component is negligible as it uses the same technology as other components already integrated in the system.

Being placed at the center of this research project, this integration platform enables most of the research questions that iTalk2Learn is investigating, from the right balance between structured and exploratory learning to the influence of speech in the learning process. The platform collects data from end users, including among others speech and answers to exercises, and distributes it to the different components in the system to ensure that the users' experience is seamless and does not interfere with their learning. The platform enables and is used for the different studies taking place in year 2 of the project, both in Germany and in the UK.

References

[1] HTML5 (2013). Retrieved June, 2013, from http://www.w3.org/TR/htlml5

[2] Differences between HTML4 and HTML5 (2014). Retrieved April, 2014, from http://www.w3.org/TR/2011/WD-html5-diff-20110405/

[3] Adobe Flash (2013). Retrieved June, 2013, from http://www.adobe.com/uk/products/flash.html

[4] ECMAScript (2013). Retrieved June, 2013, http://www.ecma-international.org/publications/standards/Ecma-262.htm

[5] Twitter Bootstrap (2013). Retrieved June, 2013, from http://twitter.github.io/bootstrap/

[6] Google Web Toolkit (2013). Retrieved June, 2013, from https://developers.google.com/web-toolkit/

[7] Apache Tiles (2013). Retrieved June, 2013, from http://tiles.apache.org/

[8] Thymeleaf (2013). Retrieved June, 2013, from http://www.thymeleaf.org/

[9] jQuery (2013). Retrieved June, 2013, from http://jquery.com/

[10] External Interface (2013). Retrieved June, 2013, http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/external/ExternalInterface.html

[11] Spring (2013). Retrieved June, 2013, from http://www.springsource.org/

[12] ApacheDS, Apache´s LDAP implementation (2013). Retrieved June, 2013, from http://directory.apache.org/apacheds/

[13] JPA (Oracle´s documentation), http://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html (last access: June 2013)

[14] Hibernate from JBoss (2013). Retrieved June, 2013, from http://www.hibernate.org/

[15] MariaDB (2013). Retrieved June, 2013, from http://www.hibernate.org/

[16] IMS Learning Tools Interoperability (2013). Retrieved June, 2013, from http://www.imsglobal.org/toolsinteroperability2.cfm

[17] Schatten C., Wistuba M., Schmidt-Thieme, L. and Gutiérrez-Santos S. (2014) Minimal Invasive Integration of Learning Analytics Services in Intelligent Tutoring Systems, 14th IEEE International Conference on Advanced Learning Technologies - ICALT2014.

Appendix A - iTalk2Learn Installation Manual

This appendix will guide you through the installation of the integration platform of iTalk2Learn, including several technological dependendencies (other software that is needed for iTalk2Learn to run). You may find it convenient to create a directory (preferably named 'italk2learn') and place all the required components inside; of course, this is not mandatory.

Java

The compilation of the source code requires a 32-bit Java compiler. Consequently, the executables require the corresponding JVM. The first step is to install the latest 32-bit JDK.

Windows: you can download the current JDK from the following page:

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

Linux: most distributions contain a java package including the word "java" or "jdk" (for Java Develpment Kit) that can be installed using the usual package manager (e.g. apt, yam, synaptic).

Web Application Server

The application server that should be used for iTalk2Learn is Apache Tomcat 7.0.

Windows: You can download the required version from the following page:

http://tomcat.apache.org/download-70.cgi

There is no installer for this software for Windows. The files have to be decompressed and placed in a directory. The server will then be ready for immediate execution. Since the software is a Java application it needs to know where the JVM is so that it can direct itself to it. The environment parameter JAVA_HOME needs to be set for that purpose. The following two lines of code have to be inserted into the file startup.bat. This file exists in the bin folder under the server's home directory.

```
rem set JAVA_HOME if not defined
if not defined JAVA_HOME set "JAVA_HOME=C:\ProgramFiles\Java\jdk1.7.0_51"
```

The path given in the above command will probably have to change and reflect the path you decided to install the software to. Also note that these lines have to precede the script that starts the catalina server. The recommended location is line 19 or 20.

Linux: distributions contain a package named "tomcat7" or similar.

Common:

The next step is to define roles and users in tomcat. Amend the file tomcat-users.xml in the conf folder that is located under the tomcat home directory and add the following lines as the content of the <tomcat-users> tag.

```
<role rolename="admin-gui"/>
<role rolename="manager-gui"/>
```

```
<role rolename="manager"/>
<role rolename="admin"/>
<role rolename="manager-script"/>
<user password="script" roles="manager-script" username="script"/>
<user username="admin" password="admin" roles="admin,manager,admin-gui,manager-gui"/>
```

Maven

This is the software that prepares the iTalk2learn application and installs it into tomcat.

Windows: You can download the latest version from the following page (choose the binary version):

http://maven.apache.org/download.cgi

There is no installer for this software either. The files have to be decompressed and placed in a directory. The application is written in Java and therefore there is a dependency on JVM.

Create a file (preferably) named maven.bat and place it in the project main directory. Then add the following lines to it:

```
rem set JAVA_HOME if not defined
if not defined JAVA_HOME set "JAVA_HOME=C:\Program Files (x86)\Java\jdk1.7.0_51"
rem set MAVEN_HOME if not defined
if not defined MAVEN_HOME set "MAVEN_HOME=C:\italk2learn\apache-maven-3.2.1"
rem update path
set PATH=%PATH%;%MAVEN_HOME%\bin
```

The paths used are for illustration purposes only. They have to reflect the actual locations that the software is installed to. This file can be used to test and execute maven. It has to be executed from the command line prior to maven so that the system can find maven and maven can find the JVM.

The following command can be executed after this file in order to test maven:

```
mvn –version
```

Linux: the package name is usually “maven”; the installer takes care of everything.

Database Management Server

The DBMS used for this project is MariaDB 10.0.

Windows: The software can be downloaded from the following page:

https://downloads.mariadb.org/

You can use the following command to install (silently) the DBMS under the current (home) directory:

```
msiexec /i mariadb-5.5.36-win32.msi INSTALLDIR=%CD%\mariadb SERVICENAME=italk2learn
PASSWORD=italk2learn /l marialog.txt /qn
```

You can then create the necessary database objects and populate them with test data using the following command:

```
"%CD%\mariadb\bin\mysql.exe" "--defaults-file=%CD%\mariadb\data\my.ini" -uroot --
password=italk2learn < italk2learn.sql
```

This command has to be executed from the directory that is immediately above the DBMS home directory (its parent directory). It also presupposes the existence of the file italk2learn.sql in the same directory. This file contains all the required SQL scripts for creating and populating the database and creating the user that will be accessing it.

Directory Server

The LDAP service used for this project is ApacheDS. The software can be downloaded from the following page:

http://directory.apache.org/apacheds/downloads.html

Installation in this case has to be done in interactive mode and relates only to the service. The service cannot be administered without a complementary software called Apache Directory Studio. This software can be downloaded from the following page:

https://directory.apache.org/studio/downloads.html

Again, installation has to be done in interactive mode. After installation the studio can be used to import the file it2l.ldif that contains account details of the users that correspond to the sample database created in the previous stage but before that is done the LDAP server needs to be created and configured.

Execute the directory studio and do the following:

Create an ApacheDS 2.0.0 server and activate it using the default ports.

Right click on the server and select create a connection. That will create a connection that can be used to access the server.

Then go to LDAP Browser and right click the node DIT. Select import à LDIF import… and use the dialog window that appears to import the file it2l.ldif

iTalk2Learn

The software for this project can be downloaded or pulled from the following git repository:

https://github.com/it2l/it2l-platform

The project configuration file has to be replaced with the following:

```
database.driverClassName=com.mysql.jdbc.Driver
database.url=jdbc:mysql://localhost:3306/italk2learn
database.username=italk2learn
database.password=password
```

```
database.jndi=jdbc/dsItalk2learn

jdbc.pool.initialPoolSize=1
jdbc.pool.minPoolSize=1
jdbc.pool.maxPoolSize=2
jdbc.pool.checkoutTimeout=5000
jdbc.pool.maxStatements=5000
jdbc.pool.testConnectionOnCheckin=true
jdbc.pool.idleConnectionTestPeriod=600

node.tree.cron.expression = 00 17 09 * * ?
sync.obralia.cron.expression = 00 38 11 * * ?

ws.base.address=http://localhost:9999/

ldap.url=ldap://localhost:10389/o=italk2learn
ldap.password=secret
ldap.jndi.t3=t3://localhost:7001
```

The location of this file in the project is given in the following path:

\src\main\resources\

Before compilation the gienah maven repository needs to be in place.

Then the code has to be compiled and the resulting package has to be installed into Tomcat. Both actions can be carried out by MVN.

The following command has to be executed from the project home directory.

```
mvn clean compile package tomcat:deploy
```

This command compiles the package and installs it into tomcat. Please note that tomcat needs to be operational during this process.

If this is not the first time you compile the project then maven may need to be updated so that the local repository contains the latest dependency files. In this case the following command must be executed before the compilation:

```
maven update
```

The final step is to open a web browser and use the system.

http://localhost:8080/italk2learn/