iTalk2Learn

2013-06-30

Deliverable 4.1

Technical description of the first prototype of the iTalk2Learn platform

30 June 2013

Project acronym: iTalk2Learn

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

Work Package: 4

Document title: D4.1-Technical_description_of_the_first_prototype_of_the_iTalk2Learn_platform

Version: 2.0

Official delivery date: 30/06/2013

Actual publication date: 30/06/2013

Type of document: Report

Nature: Public

Authors: Jose Luis Fernandez, Sergio Gutierrez-Santos

Version	Date	Sections Affected
1.0	07/06/2013	Initial version
1.1	_____	Review comments processed
1.2	_____	Updated to reflect changes in the implementation
1.3	_____	Updated to reflect changes in the implementation
2.0	30/06/2013	Final version

Executive Summary

Italk2learn will provide a strong platform with high level integration (scalability) with new technologies and services; and aims at simple maintenance for future development.

It will be based on the Service Oriented Architecture (SOA) paradigm to achieve our goals. It will have an interface developed in HTML5 at the presentation layer, to make it available to most popular devices nowadays.

Table of Contents

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
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
LDAP	Lightweight Directory Access Protocol

List of Figures

1. Introduction

The iTalk2learn platform will be 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 up to the latest version without any effort from final users or school administrators.

This document describes the first prototype of the iTalk2Learn platform. The current prototype is going to be used to collect an initial corpus of child/tutor interaction for WP1, 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, 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 main goal of the work reported in this document has been to define the interfaces for the different elements of the system and their interplay. Additionally, necessary infrastructure ---i.e. data storage and knowledge representation--- is also described. The overall goal of WP4 is to provide a flexible and scalable infrastructure for the elements of the system developed in the other WPs in such a way that those elements can be exchanged independently from each other and, eventually, be replaced by other elements in the future providing a test bed for future technologies. On top of that, another important goal of the iTalk2Learn platform is to achieve great scalability, exposing the services that are necessary,

permitting reutilization and adding new services and content as necessary in the future with minimal overhead, even beyond the lifetime of the project. This goal has informed most of our design decisions.

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. Section 4 describes some strands for future work and improvement. The report closes in Section 5 with come concluding remarks.

2. Components diagram

A diagram with the main components of the Italk2learn architecture is presented in Figure 2.1. The following sections will explain each layer and its components in detail, as well as the technologies involved.

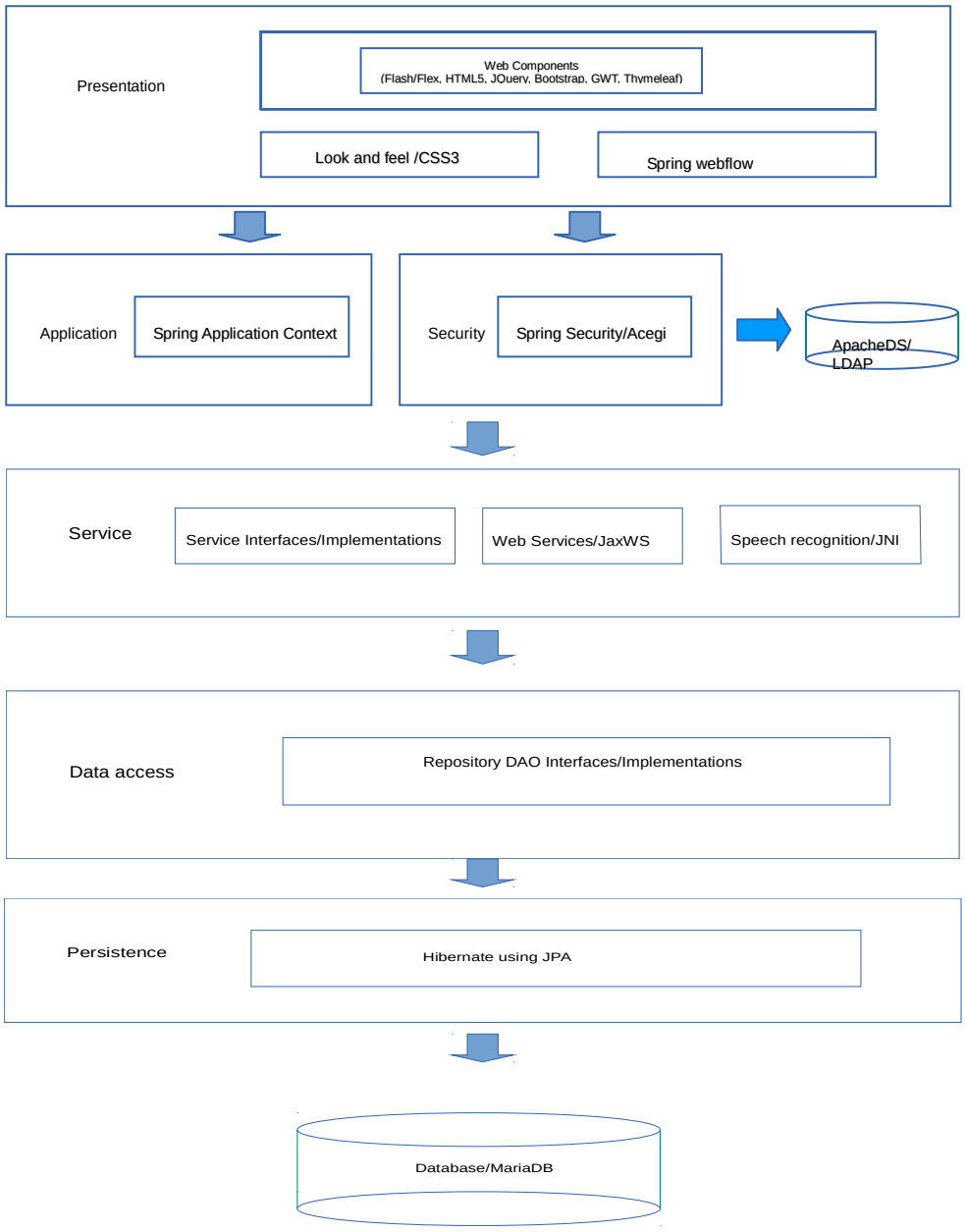

Figure 2.1. iTalk2Learn architecture main diagram

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 was 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.

Our Presentation Layer is cleanly divided according to a Model-View-Controller (MVC) pattern. This simplifies maintenance and, in particular, will facilitate 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 (e.g. authentication, sequencing and display of exercises) to make it easier to add functionality and keep a clear separation of concepts.

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.

- **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.

- **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 HTML is Javascript (sometimes referred to as JS or as per its standard name, ECMAScript [4])*. JavaScript is an interpreted computer programming language. It was originally 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 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 popular every day.

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

** For our understating we will consider JavaScript and ECMAScript as the same thing.*

- **Twitter Bootstrap**

Twitter Bootstrap [5] is a free collection of tools for creating websites and web applications (it is a HTML, CSS, and JavaScript toolkit). It contains HTML and CSS-based design templates for typography, forms, buttons, charts, navigation and other interface components, as well as optional JavaScript extensions.

We use some components and templates provided by Bootstrap in order to get a consistent and good-looking look-and-feel, and to ease the interaction of the user with the platform. It offers a convenient combination of visual components to use. As an added benefit, these components will be easy to port to tablets and mobile devices in the future.

- **GWT**

Google Web Toolkit [6] is an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java. GWT emphasizes reusable, efficient approaches to common web development tasks, namely asynchronous remote procedure calls, history management, bookmarking, UI abstraction, internationalization, and cross-browser portability.

Similarly to Twitter Bootstrap, we use some of the Javascript widgets provided by GWT in order to make the user's interaction as seamless as possible.

- **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.

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 maintain.

- **Thymeleaf**

Thymeleaf [8] is a Java XML/XHTML/HTML5 template engine that can work on both, the 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 worsens 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**

JQuery [9] is a multi-browser JavaScript library designed to simplify the client-side scripting of HTML. We use the functionality that provides JQuery about AJAX. 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**

External Interface [10] is an application programming interface that enables direct communication between Action Script and Flash 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). Using External Interface, the system can execute an Action Script function from within the Flash runtime using normal JavaScript calls from the HTML page. The executed ActionScript 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 code would

be running on two completely independent platforms (namely, the web browser and the Flash runtime environment) and would not be able to communicate.

Our interface implements a simple API to communicate with the exercises (Flash). It processes input and output data (Request-Response objects) in order to be aware of what happens inside the exercises. This enables the communication with Flash 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 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 did not provide the required credentials (normally username and password) access to the platform and its services would be denied.

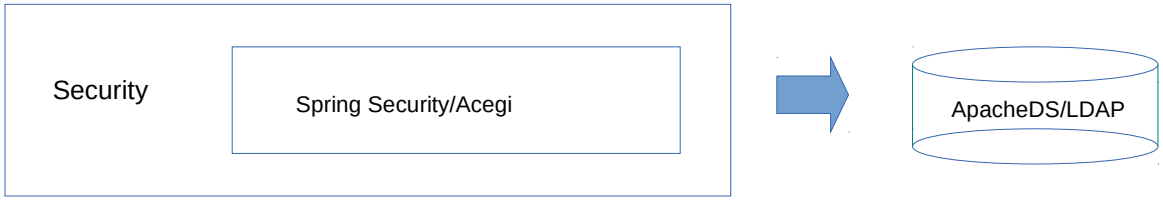

Figure 2.2.1. iTalk2Learn security diagram

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.

2.3 Application Layer

Inside this layer all exposed services are placed (interfaces only). The layer is divided in business units using a Restful architecture model to expose. 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.

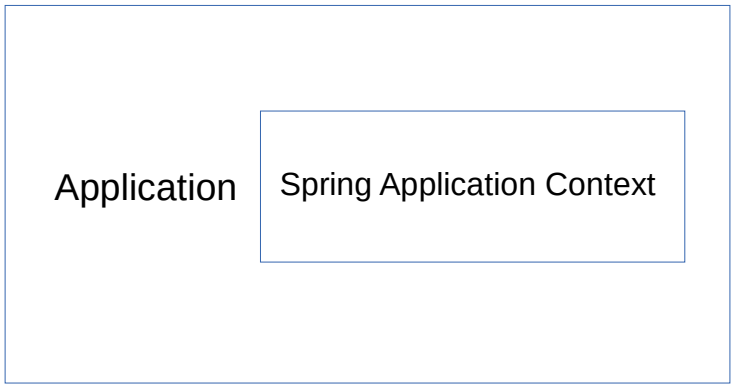

Figure 2.3.1 iTalk2Learn application context diagram

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 a certain tasks from its 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

2.4 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).

2.5 Service Layer (Recommender system and Speech recognition integration)

This layer will have all services implementations or business logic mentioned in the application layer.

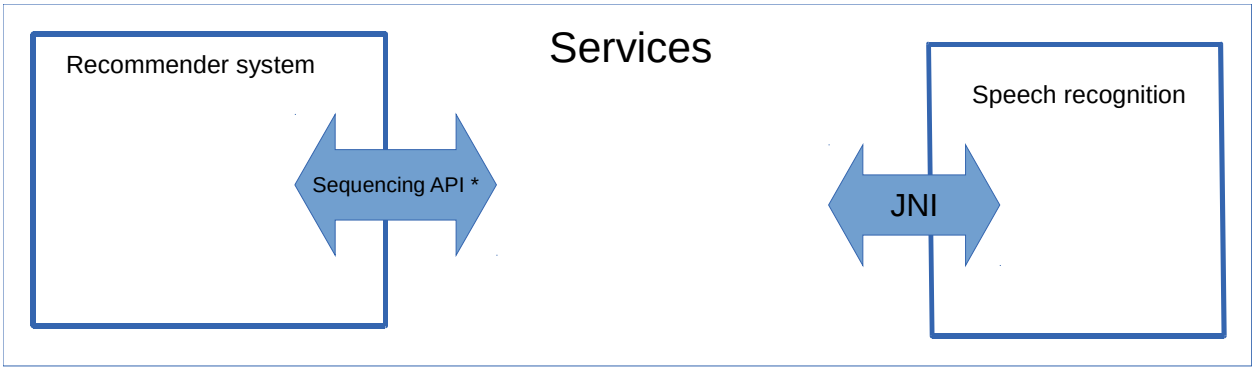

Figure 2.5.1 iTalk2Learn services layer diagram

This is the layer where the implementation of several important modules of iTalk2Learn will be placed. Examples of services whose implementations will be placed here include the algorithms that calculates the next exercise to be presented to the user, the management of exercises' scores, the code that transforms a sound waveform into a transcription of speech, as well as any other functionality to be added in the future.

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 in WP3 and the recommender system (sequencer) developed by UHi in WP2. Integration and data interchange with these modules is explained in more detail later in this document.

2.6 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 (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.

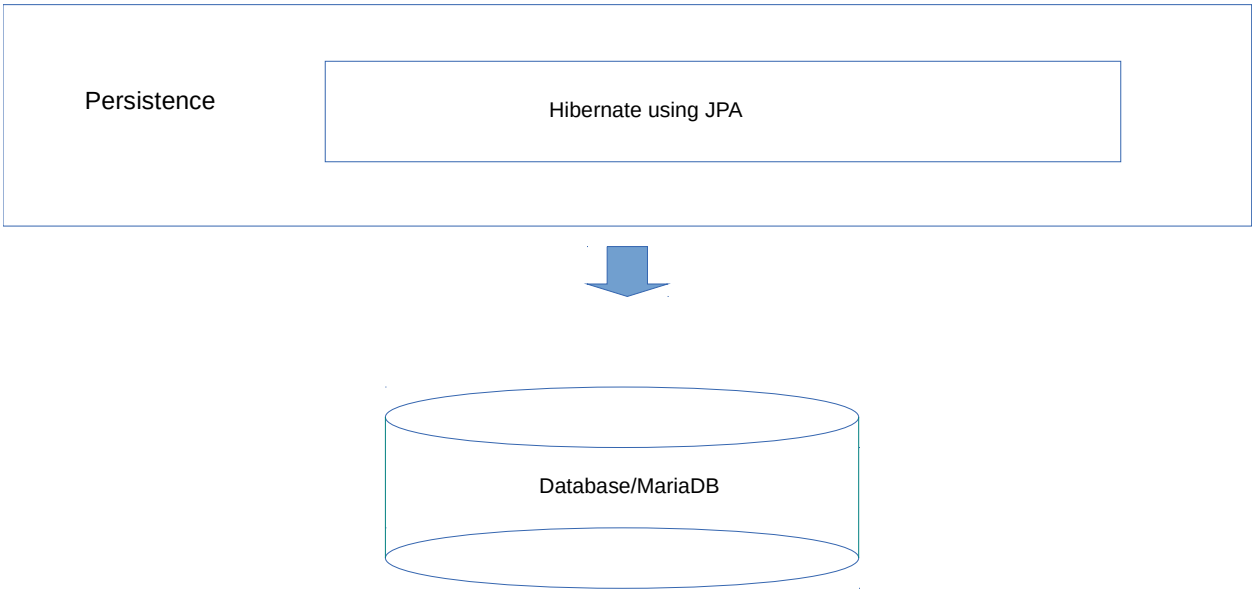

Figure 2.6.1. iTalk2Learn data access layer main diagram

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: it 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.

2.7 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 a continuation of MySQL, a well-known open-source database. MariaDB is a project 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. In the future we will add all necessary entities and relations in order to collect all data necessary. The current data model describes only users and sequences of exercises, as depicted in the following entity-relationship model:

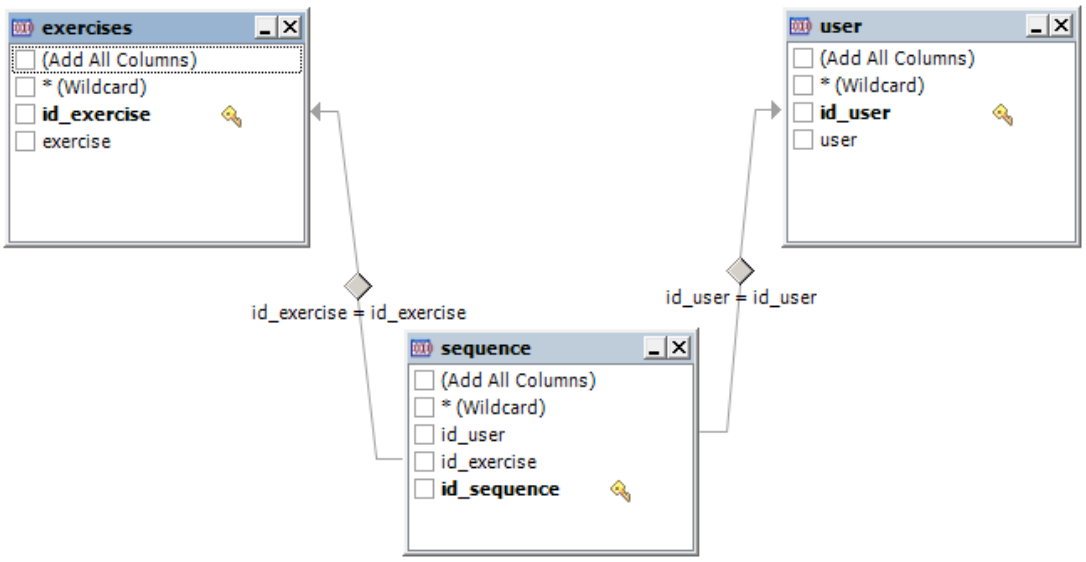

Figure 2.7.1. iTalk2Learn initial storage diagram

This model contains two entities and one relationship.

Exercise (Entity):

- id_exercise: It is the exercise identifier. Each exercise will have a distinct identifier to identify the rest properly.
- exercise: It is the name of exercise. For example, whether we have a Whizz exercise in html5, it will be called WhizzHTML5

User (Entity):

- id_user: It is the user identifier. Each user will have a distinct identifier to identify the rest properly.
- user: It is the User's name. It will be the same user that it's stored on LDAP for example whether we entry with the identifier "jkeats" on the application, it will be our user. With this name we get the exercise sequence and related information.

Sequence (Relationship):

- id_exercise: It will serve as to relation with the *exercise entity* to get the exercise and it can associate with the user.

- id_user: It will serve as a relation with the *user entity* to get the user and it can associate with the exercise.

3. Communication with other systems

This section explains in detail how the iTalk2Learn platform communicates with the other subsystems. At the learning content level, there are three main sources of learning content in iTalk2Learn: the Fractions Tutor, MathsWhizz, and the exploratory activities. The latter are not developed yet, so this document described only how the Fractions Tutor and the Whizz exercises have been integrated in the system. This is one important challenge in this project as both types of exercises were developed a long time ago, with a completely different context in mind, and using different (and, to a point, incompatible) technologies. The exploratory activities do not present such a challenge because they will be created from scratch and can, therefore, be implemented in collaboration between the partners and in line with the technology that we are using in the integration platform.

At the system level, there are two main technologies that need integration at this point: on the one hand, the speech-to-text module that recognises speech and provides a transcription; on the other hand, the recommender system that decides on the sequence of exercises (sequencer).

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.

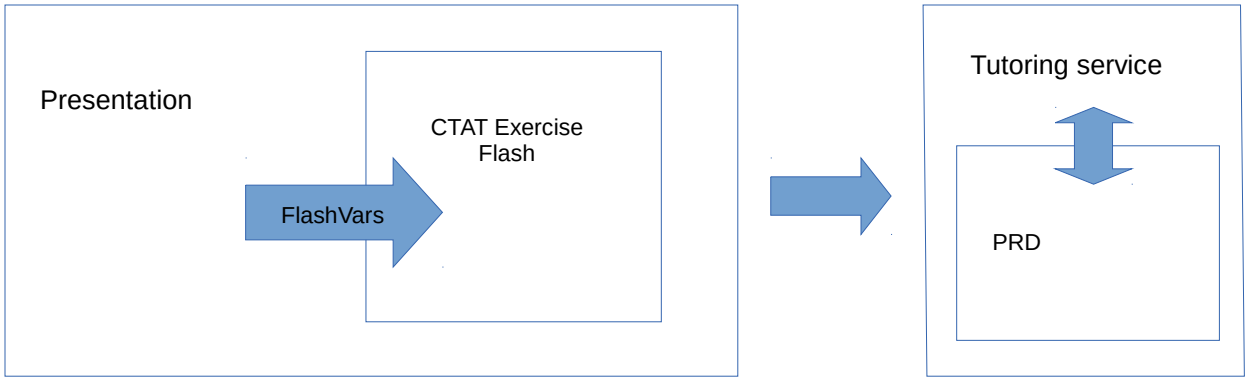

Figure 3.1.1 iTalk2Learn fractions tutor process diagram

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 communicate with CTAT exercises we have used External Interfaces.

3.2 Whizz 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, the current percentage, 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).

3.3 Speech recognition (SAIL software)

3.3.1 Integration with iTalk2Learn

There are two main components involved in speech recognition:

- A master program that starts, stops, and communicates with ASREngine, listens for client connections, and sends results to the client (e.g. the in our case, the rest of the iTalk2Learn platform).
- ASREngine. It is responsible of the conversion from a soundwave into text (i.e. speech recognition).

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 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.

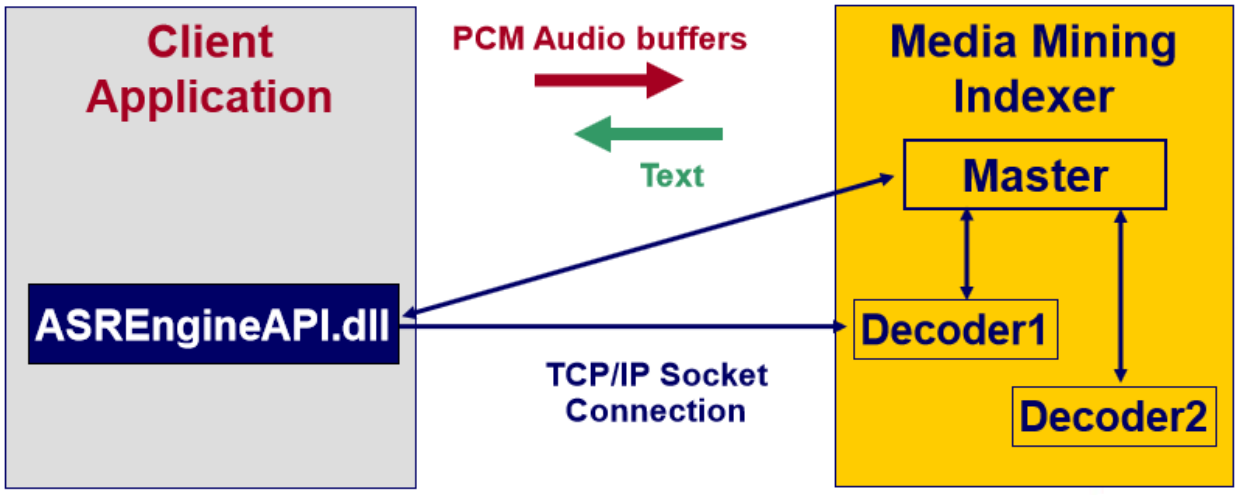

Figure 3.3.1 iTalk2Learn speech recognition process diagram

We have a client application developed in Java and allocated on the services layer (see description in former section). This client application communicates with the ASREngine library and it in turns communicates with the master.

The way data flows 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 as an audio file (wav) to the speech-to-text component. A transcription of the speech contained in the soundwave is returned to the client (the iTalk2Learn module) as an XML file. This XML file is then parsed and the speech's transcription is ready to be sent to interested components (e.g. the sequencing engine, the AI components that provide support to students).

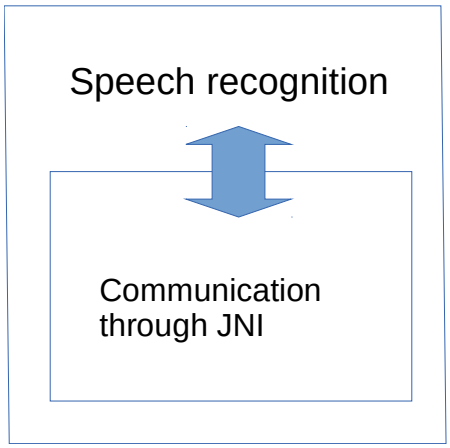

Figure 3.3.2 iTalk2Learn speech recognition communication diagram

Via the API it is possible to send buffers of audio data directly instead of recording a wav-file, and the engine will send back live the result for each speech segment. A more tight integration in the future via the API is desirable to get the latency* down to an acceptable level.

* *This is the time the speech recognizer needs from the end of speaking until the result is delivered. This ranges from subsecond to a few seconds, depending on the length of the utterance, never longer than the audio. Crucial here is the timely detection of the end of speech, parameters in the recognizer can be adjusted to get an answer quicker, but there is a trade-off between speed and accuracy.*

3.3.2 Speech Recognition requirements

In order to get the best recording performance and use Speech Recognition properly, we have the following requirements:

- Use a single channel, monaural (Mono)
- At least sampling rate of 16kHz

- 16 bits per Sample
- PCM encoding

In order to reduce the error rate of the speech recognizer, we will utilize background knowledge about the expectation of what a speaker will talk about. To make use of that knowledge it will be necessary to build a suite of different specific vocabularies and associated language models, which need to be selected depending on the state of the dialog.

Currently the speech recognition engine needs to be restarted to be able to work with a different model, which takes too long (depending on the size of the models up to a few minutes), or multiple instances of the engine need to be loaded in advance. We will extend the capabilities of the speech recognition engine to be able to deal with multiple vocabularies and language models, to enable the platform to quickly select the appropriate model without the need to restart the engine.

3.4 Sequencing engine (Recommender System)

The recommender system collects information about student actions, building student profiles and adaptive control for single-student tutoring. The Recommender system 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 a client program allocated in the services layer to communicate with the recommender system following this API. It contains three methods to drive the sequencing. A fourth method is described to be implemented in the future if needed, but it is not implemented at the moment.

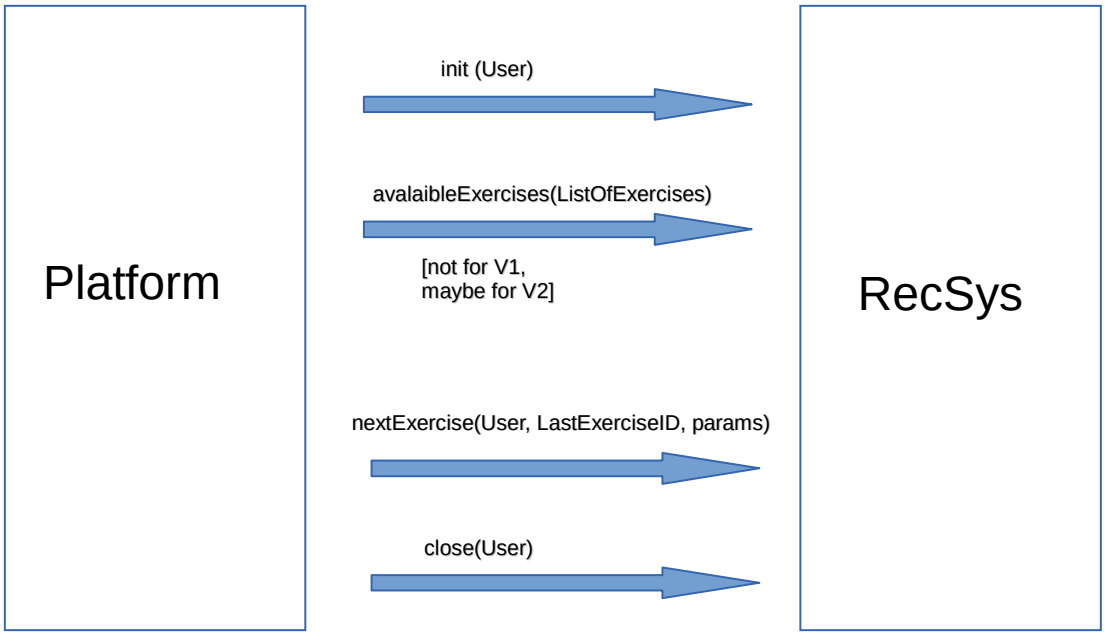

Figure 3.4.1 iTalk2Learn recommender system sequence engine diagram

- init: initialises the user on the recommender system
- nextExercise: gets the next exercise depending of the user and its past history (including the last exercise done and the score on that exercise)
- close: close the session with the recommender system
- avalaibleExercises (optional): it retrieves a list of exercises available for the user

4. Future work

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

The most clear functionality that is missing from the current prototype is the integration of exploratory activities, as these are not created yet. However, BBK has already started to draft some prototypes of exploratory activities in collaboration with partners IOE and TL, based on Flash and HTML5 technologies. It is not envisioned that integrating exploratory activities will be a major challenge because ---unlike Whizz and CTAT exercises--- they are developed from scratch in the context of iTalk2Lean and we have control over them. The next research step, once we have integrated all learning content sources in the platform, is the design and implementation of a common interface that enables all learning sources to communicate in an uniform manner with the rest of the system. This will facilitate the extension of the system with new sources of content and its intercommunication with external systems.

We are considering several technologies and frameworks for this purpose including IMS LTI [16] and inBloom [17].

We have plans to test different technologies in the context of the integration platform to study their suitability for the project. Although the current prototype is based on Java technology, we have plans of investigating alternative technologies that show potential in terms of rapid development (e.g. Rails, Django) or scalability and performance (Scala). In the specific case of authentication, we are considering the possibility of authenticating against other well-known platforms such as Facebook, Google+, and Twitter, by means of OAuth2. This is an authentication protocol that retrieves a token indicating whether credentials are accepted or not, and is gaining support among many sites.

Via the API of Speech Recognition Software is possible to send buffers of audio data directly instead of recording a wav-file, and the engine will send back live the result for each speech segment. We have plans to change the process that we have now, recording a wav-file and parsing the resulting XML file. So, we will get faster interaction with the platform in order to improve the usability and the user experience.

Finally, we would like to make the work in 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, this will require close collaboration with the content providers.

5. 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.

This integration platform will be used to answer 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.

References

[1] HTML5, http://www.w3.org/TR/htlml5

[2] Differences between HTML4 and HTML5, http://www.w3.org/TR/2011/WD-html5-diff-20110405/

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

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

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

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

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

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

[9] jQuery, http://jquery.com/ (last access: June 2013)

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

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

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

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

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

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

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

[17] inBloom, https://www.inbloom.org/ (last access: June 2013)