A mind-mapping front-end for text writing

Sérgio F. Lopes
Centro Algoritmi
University of Minho
Guimarães, Portugal
sergio.lopes@algoritmi.uminho.pt

Renata Castro
School of Engineering
University of Minho
Guimarães, Portugal
pg32903@alunos.uminho.pt

Sílvia Araújo
Institute of Arts and Humanities
University of Minho
Braga, Portugal
saraujo@ilch.uminho.pt

Abstract — **Writing is a challenging task especially for graduate students, who are often required to produce scientific writing. Mind maps help to organize ideas and are an essential tool in the planning phase of writing. Software tools have been developed to support mind-mapping, but in our research we have not found tools offering specific support for writing activities. In this paper we introduce planTEXT, a mind-mapping tool for text writing and describe its front-end architecture. This tool has been assessed by students, and the results are presented.**

Keywords – scientific writing, mind maps; web apps, SVG, AngularJS.

I. Introduction

Writing is a challenging task especially for graduate students, who are often required to produce scientific writing. It involves the transfer of acquired knowledge into text form. Both teachers and students acknowledge that students have many difficulties, such as the repetition of ideas, poorly structured paragraphs and lack of or erratic flow/progress.

The writing process involves diverse cognitive processes and strategies that can be grouped in different activities. Reference [1] identifies three top-level activities: planning, the development of text and revision. Difficulties have been detected in all of these activities, but planning is critical. If it is not properly done, the other activities will become more difficult, compromising the resulting text.

Mind maps [2] are a method for organizing ideas, where one draws text boxes around a central topic, creating sub-topics in a tree-like hierarchy. They are particularly useful in the planning phase, helping students to structure ideas. The effectiveness of mind maps has been demonstrated in different learning contexts [3] [4].

Several software tools have been developed to support the construction of mind maps, ranging from desktop and mobile applications to web applications. A few examples of the latter are: iMindMap, Popplet, MindMeister and MindMup. However, these tools are generic and do not provide specific support for the writing process. In our research, we have not found mind-mapping tools capable of supporting writing activities other than planning.

In this paper, we introduce planTEXT, a tool that offers integrated support for the planning, writing and revision activities of the writing process, and describe the tool front-end. It provides a map perspective that is synchronized with the plain text view. It is a modern HTLM5 web application, which has been assessed through a survey completed by students. The contributions of this paper are:

- an architecture that can be used as reference to develop other text writing tools based on mind maps;
- a front-end with functionalities that support writing activities from an integrated perspective.

This paper has six more sections. Section II deals with the main activities of the writing process and with the students' difficulties. In section III, related work is reviewed. Sections IV and V, respectively, describe the tool design and front-end development. The survey and its results are presented in section VI and the conclusions are discussed in section VII.

II. Writing Process Theory

Writing is a complex activity that involves many cognitive processes and strategies [5]. According to [1], the cognitive skills involved in writing can be grouped in three components: planning, textualization, and revision. The authors call them processes, but we call them activities of the overall writing process.

The planning activity involves the development and refinement of goals (the top-level ideas). Thus, it is necessary to group lower-level ideas and form concepts that enable writers to structure the text coherently. Poorly structured texts are often the result of non-existent or incipient planning.

The textualization activity consists in creating linguistic sequences, organized in sentences, and paragraphs that follow the plan to achieve the intended goals. In a school setting, the difficulties are evident, with students not being able to follow progression and continuity principles, to avoid repetitions or to maintain logical coherence.

The revision activity consists in reading the text to evaluate it and make corrections/improvements, at micro and/or macro structural levels. Usually, students only consider spelling and punctuation errors, leaving aside the abovementioned aspects, which are also important.

Although planning is the natural start of the writing process, all activities are cyclic and overlap throughout it. For example, when a writer stops, revises, and then re-writes, she/he then adjusts the planning and/or plans what will follow.

III. Related work

There are mind-mapping tools for the major computing platforms and many for the web. Most of the analyzed web tools feature a common set of basic functionalities, such as: virtually unlimited number of nodes and levels, zooming, exporting in several formats (PDF, JPEG, SVG, etc.), different

TABLE I. FUNCTIONAL REQUIREMENTS

R#	Description
1	Public area with general information about the tool and on how to use it.
2	Registration using an e-mail address.
3	Private area for account update and text writing and management. [a]
4	Users have access and can manage only their own texts.
5	Set of mind map templates for text writing.
6	Texts with a two-level structure: sections and topics.
7	Mind-map editing area to create the text structure.
8	Text editing area to type the contents.
9	One-way synchronization from the map view to the text.
10	Automatically insert introduction and conclusion sections in the text view, which do not appear in the map.
11	Export of the map and text in well-known file formats.

a. All the following requirements.

node content (e.g., text, images) and appearance options, branch customization, undo and redo, move part/all of the map.

iMindMap is the tool designed by Tony Buzan (inventor of mind maps). It has a very rich set of features (different modes, integration with other applications, etc.) and its web version is based on Adobe Flash. It was used, for example, in teaching and accounting [7] and gesture-based authoring [8].

Popplet is another tool with an Adobe Flash web version. Its most distinctive feature is allowing users to share the map with other people and work on it simultaneously. Thus, it is especially suited for teamwork and has been used for collaborative learning [9].

MindMeister web version is HTML5 and it allows the map to be shared on the cloud, where all maps are stored. It features a history of changes and imports files from other tools. This application has been used for user-modelling and information retrieval [10].

Coggle is another HTML5 application that allows maps to be shared on the cloud and has a change history. Maps cannot be simultaneously edited, but it features a group chat [11]. It was used for in-class and home collaborative activities.

MindMup is an HTML5 application that supports offline operation and online map sharing and publishing, but its interface is less intuitive than those of other tools. It was used to monitor the students' learning of syllabus topics [12].

There are several web-based mind-mapping tools and their usage has been explored in a variety of fields. However, we have not found any targeting the field of writing.

IV. PLANTEXT DESIGN

All tools that we have found are general-purpose and therefore they do not provide specific features for text writing. For example, there are no ordering or sequence relationships between ideas at the same hierarchical level. Our aim is to build a tool that offers those kinds of features and helps with the textualization and revision activities.

TABLE II. NON-FUNCTIONAL REQUIREMENTS

R#	Description
1	To maximize browser compatibility.
2	Minimize hangs and any interruptions on the availability of the user interface.
3	Operate with and without internet connection.
4	Provide a clean and user-friendly user interface.

A. *Requirements Analysis*

The project involves people from the fields of linguistics, playing mainly the role of users, and software engineering, acting predominantly as developers. From the workshops of this multidisciplinary team, a set of requirements was defined. Tables I and II summarize the most important ones.

Functional requirements 1-4 and 11 are self-explanatory. Requirement 5 offers an easier way to start writing a text, by automatically creating a set of sections, which could be, for example, the well-known IMRaD (Introduction, Methods, Results and Discussion) structure for scientific writing. Requirement 6 stems from two goals: to develop a proof-of-concept tool that can be used for measuring the success of the approach, and to target medium-sized texts for which a two-level structure is adequate. Requirements 6-8 are essential to provide an integrated environment for the writing process (see section II). Specifically, the mind map and text editing areas support the planning and textualization activities, respectively. The automatic update of the text according to structural changes in the map provides support for the iterations between the two activities. A few examples of structural changes are: to reorder sections/topics, to move a topic from one section to another, and to promote/demote a topic/section. For this to be possible, the tool must remember the order/sequence of the text elements at the same hierarchical level. The idea behind requirement 10 is that most texts have an introduction and a conclusion, and therefore it is not necessary to include them in the map view. In addition, the tool can automatically list the sections' titles inside the introduction, helping the writer to describe the text structure. It does the same for the conclusion, helping the writer to write a summary of the main ideas in the text.

Non-functional requirement 1 means that the web application must use either standard HTML or a framework that transparently uses browser-specific features. Plugin technologies are not an option, since they are planned to be discontinued and their availability in browsers already faces significant restrictions. Due to requirement 2, the application cannot work based on requests of entire pages. Rather, it must use asynchronous communication with the server to update parts of the interface. Technically, it must be a Single-Page Application (SPA). Requirement 3 is very important for two reasons. Firstly, because internet connection is not always available, and the tool core functionalities do not need to depend on it. Therefore, the tool must be an offline web application that writers can use regardless of internet availability. Secondly, because whenever there is a connection breakdown, the work produced since the last save must not be lost. A consequence of this requirement is that the web

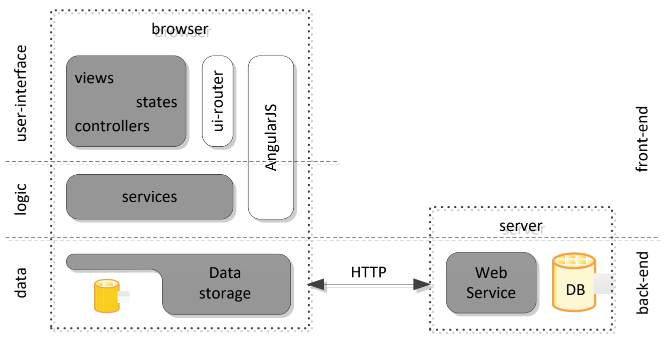


Figure 1. Example of a figure caption.

application must store data locally, in the browser. Requirement 4 is common and self-explanatory.

B. *System Architecture*

Fig. 1 presents the system architecture, which comprises a server and a client side. The server side is based on nodeJS (https://nodejs.org) and is composed of a database and a REST web service interfacing the database. The client side follows the traditional 3-tier architecture (data, logic and user interface). The data layer is a Javascript data persistence library for both local and remote storage. This means that the data layer is the consumer of the REST web service. The client data layer and the server side constitute the application back-end (this is outside the scope of this paper and will be addressed elsewhere). All the front-end sees is the API of the data layer.

The user interface (UI) is key to the success of software tools, and nowadays web applications feature interactivity, responsiveness and aesthetics levels that explain the widespread adoption of UI or web application frameworks. These frameworks facilitate development, especially if the application is large. Although our application is not large, it is expected to integrate further functionalities in the future (e.g., dictionaries and spellers). Therefore, we decided to base our tool on a front-end web application framework. Google's AngularJS (https://angularjs.org) was our choice, mainly because we had previous experience with it.

Being an application framework, AngularJS covers both layers of the front-end and therefore defines the structural elements of the application. In planTEXT, the logic layer has the form of AngularJS services, while the UI layer is composed of views, controllers and states. Since native AngularJS routing facilities (ngRoute) are limited (to a single view per page), we opted for a more flexible solution, specifically, the widely-used module of ui-router library (http://ui-router.github.io). Ui-router offers multiple views in a single page based on a hierarchy of nested states (and views).

C. *Front-end*

The front-end design is shown in Fig. 2. It comprises the perspectives that are offered (the rectangles), and the possibilities of navigation between them (the lines with direction arrows). (The term "perspective" is used to distinguish the external view from software artifact views.) The

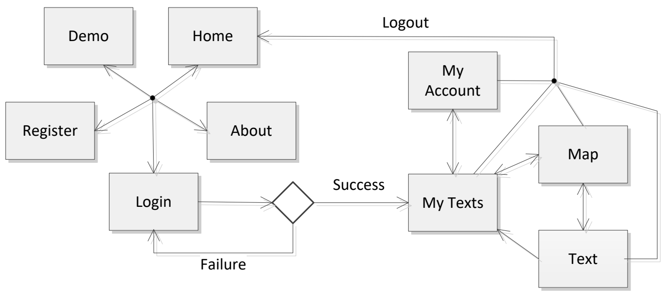


Figure 2. Application perspectives and navigation.

perspectives identify the main application features/contents. Fig. 2 separates public perspectives (on the left side) from the ones that require authentication. The publicly accessible perspectives are:

- Home – introduces the application with an identifying logo and a menu to access other perspectives;
- About – introduces the project team and the project itself (how it started, goals, novelty…);
- Demo – describes the functionalities provided by the tool and how they can be used;
- Register – enables the user to create an account for accessing the core functionalities;
- Login – identifies the user and gives access to her/his private area of text writing.

In the public area, the user can change between any two perspectives with a single click, as depicted by the star relationships between them in Fig. 2. In the Login perspective, if the user input matches an account's credentials, the application gives access to the authenticated area, which is composed of four perspectives:

- My Texts – lists the user's texts ordered by different criteria and supports text management;
- My Account – allows the user to manage her/his personal data;
- Map – supports the development of the text (section and topics) structure, i.e., creating the mind map;
- Text – displays the text in plain form and supports editing the content of each topic.

The My Texts perspective is the entry to the private area and is the only access to My Account. All authenticated perspectives allow the user to log out and return to the public area. When a text is selected in My Texts, the application changes to Map perspective. Both Map and Text perspectives allow the user to switch between each other and to return to My Texts.

V. FRONT-END DEVELOPMENT

A. *Interface design*

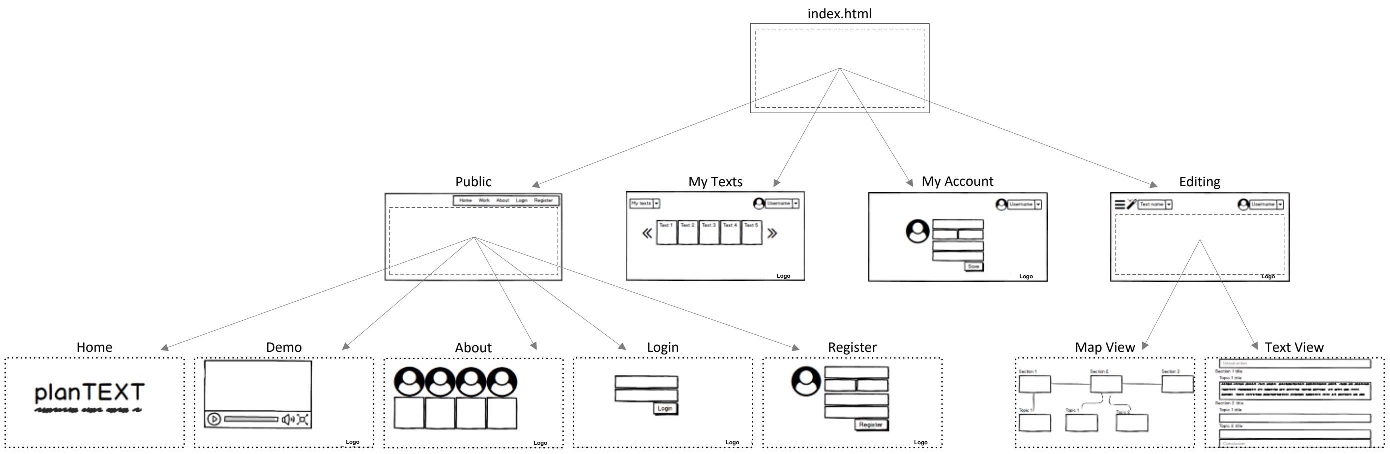


Figure 3. Interface design: graphical sketches and hierarchy of the different views.

The interface design comprises the graphical sketches of the different application perspectives (see previous subsection) and the structure of SPA views. These two aspects are described in Fig. 3, which shows the views layout, a small (for the sake of space) sketch of each view, and the hierarchical relationships between them.

The main page defines all that is common to all views of the application and a central content area. This area is filled by one of four different views: Public, MyTexts, Editing, and MyAccount. MyTexts and MyAccount views have perspective-specific content (i.e., they are leafs in the view hierarchy), while Public and Editing views have contents shared by a set of perspectives (i.e., they are structural views).

Both MyTexts and MyAccount views support the perspective with the same name (see Fig. 2). The Public view supports what is common to all public perspectives, namely, the navigation menu between them, and it defines a content area to be filled by one of its five perspective-specific sub-views. When the user navigates between any of the public perspectives (on the left side of Fig. 2), it is only this part of the interface (content area) that is updated. The same applies to Editing view, the Map and Text perspectives and the respective likewise-named sub-views. The application interface is thus structured in three layers.

B. *Back-end integration*

The data layer was developed in ECMAScript 6 (ES6) Javascript and consists of a set of classes, as shown in Fig. 4. The classes represent the different objects of the data layer (Users, Text, Section and Topics) forming a hierarchy of composition relationships. More specifically, they have the following roles:

- DB – represents the local and remote storage databases and supports user-related functions, namely, account management and login;
- User – represents a user and offers methods for managing the respective texts;
- Text – represents a text and supports the management of its sections;
- Section – represents a section and allows to manage its topics;
- Topic – represents a topic, simply storing its data.

To maximize browser compatibility and to be able to use data layer classes directly in AngularJS (version 1.4), we translated ES6 code to ES5 (in a process also known as *transpiling*). To carry out the translation, we used webpack (https://webpack.js.org), a bundler of JS modules with dependencies between them, which features a set of extensions (loaders), including an ES6-to-ES5 source-to-source compiler. The ES6 classes were then converted to ES5 objects.

C. *Structure*

The logical structure of the front-end is composed of views, controllers, states and services. The front-end implements the Model-View-ViewModel [13] pattern, instrumenting the views (with Angular's ngModel directive) to perform two-way data-binding with controllers.

With the exception of the top-level empty view in index.html, all views have a corresponding state. Consequently, Fig. 3 describes both the views and the states hierarchy. The application control flow can be described by Fig. 2, since its nodes correspond to leaf-states (see Fig. 3), and its arrows represent the transitions between states.

Fig. 4 shows the controllers and services, with the former in the UI layer and the latter in the logic layer. Each controller has the same name as its view/state. There are no controllers for Home, Demo and About views and states, because they implement non-interactive perspectives, with static content only. Controllers rely on services, as shown by the down arrows between them. The figure also shows the usage of data layer classes by services.

D. *Behaviour*

Controllers provide the functions supporting the respective view/state. The Register controller provides the functions for verifying username uniqueness and password safety, uploading and resizing the user photo, and adding a new user to the DB. The Login controller is the simplest one, containing only the authentication function.

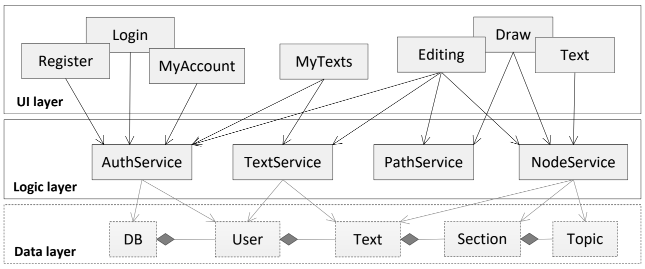


Figure 4. Controllers, services and back-end relationships.

The MyAccount controller provides functions to update the user profile. MyTexts includes functions to calculate the number of pages necessary to display all texts, list texts ordered by different criteria, add and remove texts, change text title and select a text (for editing).

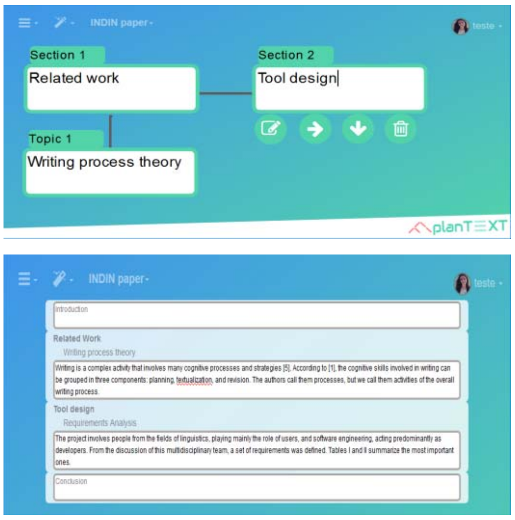
The Editing controller contains functions common to both Map and Text perspectives, namely, for logging out and downloading files with the map and the text. The Text controller is simple, since it supports only the editing of content in HTML <textarea> elements. In contrast, Map is the most complex controller, since it supports the creation of mind maps. It includes functions to add, delete, and move sections and topics. The paths between map nodes (i.e., section and topics) are automatically built. Move actions involve rebuilding all necessary paths between nodes. Furthermore, some of these drag-and-drop actions change the hierarchical and ordering relationships between two or more sections and topics. Examples are: swapping two sections/topics; promoting a topic to section; and, demoting a section to topic, if the former has no child nodes (topics).

The Map controller implements the map in SVG, which is the natural HTML choice to support diagram drawing. In addition, instead of using a 2D framework, we used HTML5 SVG native API directly, because it proved adequate for the task. To implement text editing directly in the diagrams, we used the foreignObject SVG element to include a <textarea> element inside the map nodes.

The services contain data and behaviors used by several states, namely:

- AuthService – supports adding a new user, managing user sessions, as well as obtaining and updating the profile of the currently logged user;
- Inage – supports the upload and resizing of the user photo;
- TextService – retrieves the list of texts (title and dates only) by different orders, manages texts and their titles, and manages data about the text being edited;
- PathService – manages the list of paths between all map nodes (sections and topics), computes new paths and rebuilds existing ones;
- NodeService – manages both section and topic nodes on the map, and retrieves the topics content.

Most controller dependencies on services are straightforward, so to be concise we will only refer to the noteworthy ones. Most controllers use AuthService to detect if a user is logged in or not. The Editing controller uses both PathService and NodeService because it not only needs node and path data to export the map diagram, but it also needs access to topic contents to export the text as a Word document. NodeService is especially important because it allows the sharing of data between Map and Text controllers, synchronizing the two views of the text.



Introduction

Related Work

Writing process theory

Writing is a complex activity that involves many cognitive processes and strategies [5]. According to [1], the cognitive skills involved in writing can be grouped in three components: planning, textualization, and revision. The authors call them processes, but we call them activities of the overall writing process.

Tool design

Requirements Analysis

The project involves people from the fields of linguistics, playing mainly the role of users, and software engineering, acting predominantly as developers. From the discussion of this multidisciplinary team, a set of requirements was defined. Tables I and II summarize the most important ones.

Conclusion

Figure 5. Screenshots of planTEXT: map view (top) and text view (bottom).

E. *Offline mode*

Storing data locally is not enough to support offline usage of the web application; the source files must also be available. With the Offline Web applications API, this is achieved by creating a cache manifest file listing all source files (in its cache section) and adding to the application <html> element a manifest attribute pointing to the manifest file.

Although all files are available offline, the application does not allow new users to register, nor does it allow current users to start a new text. This is because the central storage service is not available and unique identifiers cannot be assigned.

Finally, by defining an application cache the application also runs faster and saves network bandwidth by avoiding requesting to the server resources that have not changed.

VI. RESULTS

The resulting application can be found at http://plantext.dei.uminho.pt. Fig.5 shows what it looks like and its main features: at the top, a screenshot of the mind map underlying the present paper (before the topic of writing process theory was promoted to a section); and, at the bottom, part of the text view (without the footer), already with the first topic of the 3rd section. The order of sections and topics in the text view is the same as the respective nodes in the map view, where it is made explicit through section and topic numbering.

A survey was conducted with 32 students from the master's degree in Multilingual Translation and Communication of the University of Minho. Most subjects had experience in using mind-mapping tools. Some of the questions were:

TABLE III. SURVEY RESULTS

Likert scale	1	2	3	4	5	Avg.
Qu.gr. a (5x)		4	14	42	95	4.5
Question b			3	10	19	4.5
Question c		1	3	9	19	4.4
Qu.gr. d (7x)	1	5	12	63	140	4.5
Question e		1	1	2	28	4.8
Question f	No: 2, Yes: 30 (93.7%)					

a) Application look-and-feel, through five questions concerning colors, fonts, layouts, images and language;

b) The purpose of each perspective is clear;

c) Navigation is easy to understand and use;

d) Writing features, through seven questions concerning the various features for building maps and the two views;

e) Download options are useful and well placed;

f) Would recommend the tool to friend.

Questions a-e were answered on a Likert scale, from 1 to 5, including a no-opinion option. Question f was answered with yes/no. The results are shown in Table III. The most important questions are b-d, because they are the most relevant for a text-writing tool. All of those have a very good assessment. The score of question f demonstrates that users were highly satisfied with the application.

VII. CONCLUSION

The planTEXT mind-mapping tool for writing is introduced and its front-end is described. The technological choices and defined architecture proved to be quite feasible. The front-end was evaluated through a survey and the results are very encouraging. The thorough discussion of the tool features and interface by the multidisciplinary development team is thought to have been decisive for the results obtained in the survey.

We argue that the approach is successful and should be further improved. The tool will continue to be used by students for academic assignments, and in future work we expect to confirm that it will help them to significantly improve their writing skills. More functionalities will be added based on user feedback, but we already envisage the synchronization of the map from changes in the text view, offering more advanced support for revision activities.

ACKNOWLEDGMENTS

This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

REFERENCES

[1] L. Flower, and J. R. Hayes, “A Cognitive Process Theory of Writing,” *College Composition and Comm.*, vol. 32, no. 4, pp. 365–387, 1981.

[2] T. Buzan, *The Ultimate Book of Mind Maps*. HarperCollins Publishers, 2005.

[3] G. Boyson, *The Use of Mind Mapping in Teaching and Learning*. The Learning Institute, 2009.

[4] A. Buran, and A. Filyukov, “Mind Mapping Technique in Language Learning,” *Procedia - Social and Behavioral Sciences*, vol. 206, pp. 215-218, 2015.

[5] R. T. Kellogg, “Training writing skills: A cognitive developmental perspective,” *Journal of Writing Research*, vol. 1, 2008.

[6] L. F. Barbeiro, and L. Á. Pereira, *O Ensino da Escrita: A Dimensão Textual*. Lisboa: Ministério da Educação Direção-Geral De Inovação e de Desenvolvimento Curricular Lisboa, 2007.

[7] W. N. Wan Jusoh, and S. Ahmad, "iMindMap as an innovative tool in teaching and learning accounting: an exploratory study," *Interactive Technology and Smart Education*, vol. 18, no. 13, pp 71-82, Apr, 2016.

[8] M. Miyasugi, H. Akaike, Y. Nakayama and H. Kakuda, "Implementation and evaluation of multi-user mind map authoring system using virtual reality and hand gestures," in Proc. 2017 IEEE 6th Global Conference on Consumer Electronics, Nagoya, 2017, pp. 1-5.

[9] S. N. Sailin, and N. A. Mahmor, “Promoting meaningful learning through Create-Share-Collaborate,” in Proc. of the ICECRS, vol. 1, no. 1, Jan 2017.

[10] J. Beel, S. Langer, M. Genzmehr, and B. Gipp, “Utilizing mind-maps for information retrieval and user modelling,” in International Conference on User Modeling, Adaptation, and Personalization, Springer, Cham, Jul 2014, pp.301-313..

[11] I. Papushina, O. Maksimenkova, and A. Kolomiets, "Digital Educational Mind Maps: A Computer Supported Collaborative Learning Practice on Marketing Master Program", *Interactive Collaborative Learning*, Springer International Publishing, pp. 17-30, 2017.

[12] J. Pereira, “Leveraging chatbots to improve self-guided learning through conversational quizzes,” in Proc. of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality, Salamanca, Spain, ACM, 2016, pp. 911-918.

[13] J. Smith, "WPF Apps with the Model-View-ViewModel Design Pattern," MSDN Magazine, vol. 24, no. 02, Feb 2009. Accessed on: March, 18th, 2018. [Online]. Available: https://msdn.microsoft.com/en-us/magazine/dd419663.aspxJ