Interactive and Personalized Activity eBooks for Learning to Read: The iRead Case[⋆]

Nick Deligiannis[1], Dionysis Panagiotopoulos[2], Panagiotis Patsilinakos[3], Chrysanthi Raftopoulou[2], and Antonios Symvonis[2]

[1] Patakis Publishers, Athens, Greece
`ngdeligiannis@gmail.com`
[2] School of Applied Mathematical and Physical Sciences, National Technical University of Athens, Athens, Greece
`dionisis.panag@gmail.com, crisraft@mail.ntua.gr, symvonis@math.ntua.gr`
[3] School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece
`patsilinak@mail.ntua.gr`

Abstract. In this paper, we present a full system that provides a personalized educational service by making use of the new ePub3 eBook standard. The system has been developed in the framework of the iRead Project [7] which aims to develop tools for personalized learning of reading skills. Based on a structural representation of the task "learning to read in a specific language" which captures linguistic expert knowledge, a profile (for each user) which reflects the users competence in mastering the reading skill is created and maintained. Based on the user's profile as well as her reading history, a new personalized interactive activity eBook which contains educational content (ePub3 embeddable mini-games) appropriate for the current mastering level of the specific user is created on demand. We present in detail the main concepts behind this *"authorless"* activity book creation: *domain modeling* and *user profiling*, *learner profile adaptation*, *personalized content selection*.

Keywords: Interactive eBooks· Personalized eBooks· Activity eBooks· Domain modelling· User profile· ePub3· Mobile learning· iRead project.

1 Introduction

iRead is the acronym for the *"Infrastructure and integrated tools for personalized learning of reading skills"* European Union Horizon-2020 funded project [7]. iRead is a 4-year (2017-2020) project that aims to develop a software infrastructure of personalized, adaptive technologies (which include real-time user modeling and domain knowledge components) and a diverse set of applications for supporting learning and teaching of *reading* skills for children with different abilities and linguistic backgrounds. iRead targets primary school children aged 6-12

[⋆] This work forms part of the iRead project, funded under Grant Agreement No. 731724 from the European Union Horizon 2020 Research and Innovation Programme.

years old. As reading is a language dependent skill, iRead currently focuses on English, English as a Foreign Language (EFL), German, Greek and Spanish. In addition, for English and Greek it also covers children with dyslexia. In iRead, personalized learning is supported by two teaching tools: the *NAVIGO* literacy game [9, 17, 1] and the *AMIGO* eReader application [10]. A large number of iRead evaluation pilots is currently underway across six European countries and the result of the evaluation are expected to be available in December 2020. One of the individual tasks of iRead focuses on interactive personalized eBooks. Its objective is to establish, as a proof-of-concept, that we can develop mechanisms for the automatic creation of interactive personalized activity eBooks which can be used to test and improve the reading skills of children. In this paper, we describe the prototype system that provides this personalized educational service, focusing on the main concepts behind the automated activity eBook creation: *domain modeling* and *user profiling*, *user profile adaptation*, *personalized content selection.* Even though the developed concepts and mechanisms are capable in supporting learning activities in different applications areas, for simplicity of presentation, in the rest of the paper we only consider the case of the *"learning to read in a specific language"* skill, or simply *"reading skill"*.

Consider a typical paper activity-book that aims to help a learner develop her reading skill. Such a book may be full of activities of the form *"which (out of the given two or three options) is the correct way to spell a given word"*, *"fill the ending in the following words"*, or *"underline the spelling errors in the following text"*. We have all grown up with such books containing activities and exercises given as practice material. One characteristic of these activity books is that they are of a *static* nature in many aspects: They can be used only once, unless filled with a pencil and then erased. No matter how many times the reader goes over the book she always sees the same material. Usually the only feedback provided for a erroneous response by a reader is just the correct answer with no clues on what went wrong. Finally, the learning material is limited in size; she may need to practice more but no more activities are available.

Besides the above limitations and having decided to buy such an activity book, we face a more serious question: *"which is the best book to buy?"*. The answer to this question requires *"to know"* the intended reader. In our example, the parent has to select the book that matches the mastery level of the child's reading skill. But, is there a variety of alternatives available to a parent? The reading activity-books are all classified to a small set of categories (based on age or school-year) and the parent has to select one of them. Usually, there is little hope to locate a book composed of the right mixture of spelling activities: a (small) set of reinforcing activities on the items the child has mastered, a large set of activities on the items the child has made partial progress towards mastering, and a (small) set of activities on the "harder" items that are to follow. From the above discussion, it is obvious that we would like to move from the classical paper activity-books that are static and non-personal to *activity eBooks* that are *interactive* and *personalized.*

Besides the typical interactivity of an eBook (e.g., turning pages, adding bookmarks, changing the font-size), the latest IDPF eBook specification, *ePub3* [4], provides for support of JavaScript [12] which allows us to create and embed interactive content into eBooks [14]. This opens numerous possibilities for interactivity: we can include into eBooks engaging interactive activities (in iRead, we call them *mini-games*), we can observe and record the readers' responses, we can provide feedback and reinforcement, we can point out the activity parts that proved to be difficult to the reader, we can support guided navigation through the pages of the eBook. In a sense, since we have embedded in the eBook an executable JavaScript program, we can include any type of interactivity we can imaging, provided that we write code for it. Of course, there exist restrictions which are mainly imposed by the way different eBook readers implement the ePub3 standard in conjunction with security related issues.

One particular feature being present under the new ePub3 standard is the ability of an interactive eBook to communicate with a (remote) server and store to it the reader's responses/answers, i.e., the reader's *reading/playing history*, for further processing. Based on the user performance on specific activities (information present in the user's reading history), we can build a *learner profile* which reflects the current level of mastering of the reading skill. In turn, by also utilizing our knowledge on the learning material/activities the learner has been exposed to through her reading so far, we can prepare in an automatic way (i.e., *author-less*) a new personalized activity book which aims to advance the current level of the reading skill.

In order to be able to automatically capture the mastering level of the reading skill, we must identify the *features* that are important in mastering the reading skill and the relations among them. These features and their interrelationships constitutes a *domain model*, the construction of which require incorporating linguistic expert knowledge. Having identified the features present in a domain model, we can estimate the *mastering level*, also referred as *competence*, of a particular learner for each feature. This array of competence-values constitutes the *learner's profile.* So, assuming that small (large) competence values correspond to low (high) level of mastery, the goal of any reading skill acquisition program is, through interactive literacy activities, to bring the learner's competence for each domain feature to a level that it can be considered as being mastered. Our ability to update the learner's competence values based on the correctness of her responses to interactive activities is referred to as *learner profile adaptation.* Of course, all the above assume that we are also equipped with a way to *measure* the learner's competence.

In iRead, we have implemented six JavaScript literacy mini-games which, when complemented with appropriate literacy content, they form activities capable of supporting the learning of multiple domain features. Each page of an iRead interactive eBook is devoted to a specific domain feature and utilizes exactly one of the developed mini-games. In principle, the content of an iRead eBook is developed automatically and is always relevant to a specific learner. The selection of features to be targeted, the mini-games to be utilized in the activities and the literacy material to be used, is based on the learner's profile, the learner's reading/playing history, a set of language specific content selection rules and a set of language specific resources (dictionaries). The size (number of pages) of each eBook is typically user defined.

The rest of the paper is structured as follows: In Section 2, we provide an overview of the architecture of iRead's personalized eBook system and of the supported user work-flow. In Sections 3, we examine the domain/learner models employed by iRead's eBook system. In Section 4, we describe how the iRead eBook system utilized learner profiling in order to produce personalized activity eBooks. We conclude in Section 5.

2 iRead eBook System Architecture

In this section we review the architectural design of the iRead eBook system[4]. We start our description with a brief presentation of a typical supported user work-flow which highlights the services provided to its users.

2.1 A typical work-flow

We assume that the *learner* is a child and that some actions are taken on her behalf from her *parent*. From the system's point of view, these two entities are not distinguishable; they are both treated as *users* of the system operating the same *account*. The iRead eBook system supports the following user work-flow:

1. The user [*parent*] creates an account in the iRead eBook server.
2. The user [*parent*] chooses the type of activity eBook to be created by selecting one of the available language domain models (currently supported: English, Greek). A *profile* is created and initialized for the specified domain model.
3. On the eBook creator application, referred to as *eBook synthesizer*, the user [*parent*] specifies eBook attributes (number of pages; one activity per page, which mini-games to include, etc). Default values are available.
4. Based on the learner's profile and reading/playing history, the eBook synthesizer selects appropriate literacy content and generates a new activity eBook.
5. The user [*parent*] downloads the generated eBook from the iRead eBook server to her tabloid.
6. (a) The user [*learner*] opens and reads (plays) the activity eBook on her favorite ePub3 compliant eBook reader. During play, reading/playing actions (referred to as *progress*) are locally saved.
   (b) Locally saved data (progress) is synced to the iRead eBook server.
   (c) The learner-profile is updated according to the newly received progress.
7. The user [*parent*] can login to the iRead eBook server, to see her [*learner*] overall progress, saved eBooks etc.

Step-6 is repeated as long as the learner reads the eBook. Step-6a (data sync) and Step-6b (profile update) are transparent to the user.

[4] For a description of the architecture of the complete iRead system see [11].

2.2 The Architecture

Figure 1 presents a schematic representation of the architecture, showing both the components present at the eBook server as well as at the client side (tablet). For a more detailed description of the architecture, see [2].

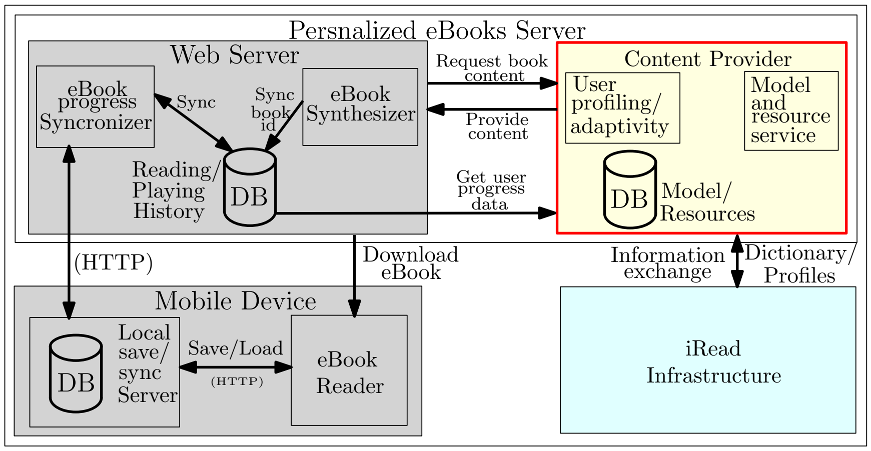

Fig. 1. The architecture of our interactive personalized eBook system.

Client-side Architecture (on Android Device) It is shown at the lower-left part of Figure 1 and it consists of the following parts:

– *eBook reader.* Any commercial eBook reader running on an android device should work. Our interactive activity eBooks have been successfully tested on the following readers: ePub Reader for Android [3], GitDen Reader [6], Kotobee Reader [13], Supreader [18].
– *Local Save/sync server app.* A learner must be able to read a downloaded activity eBook even when she is off-line. This entails a local saving functionality in order to allow the learner to partially read her eBook, to close and then re-open it without having lost her progress. Since ePub readers don't support a universal local saving mechanism, we use a specialized app on the user's device (tablet) as a local save/sync server. The app runs on the background of the user's device and functions as a basic web server, listening on some pre-selected port to which the eBook reader, through embedded code in the interactive eBook, sends HTTP requests. All data from eBooks used on a particular device are stored using this mechanism in the save/sync server's *local database*. An eBook can also use the same mechanism (HTTP request) to retrieve saved data from the local save/sync server, in order to load saved mini-game states.

The save/sync server app is also used to synchronize it's local database data with the eBook remote server's database which containing the progress of all users. A unique id embedded in every eBook upon construction, associates it with the user the eBook was created for.

Personalized eBook Server Architecture It is shown at the top part of Figure 1 and it consists of the following parts:

– *User interface web service.* It handles user account creation, account management, access to services and data etc. It is not shown in Figure 1.
– *eBook Synthesizer.* It is the component responsible for the generation of personalized eBooks. Through the *eBook synthesizer web interface* the user [parent] specifies the characteristics of the interactive eBook to be created (e.g., topic [domain model], number of pages, which mini-games). Typically, the eBook's content is generated automatically based on the learner's model and her playing/reading history, however, the user also has the option to manually specify the activities that populate the eBook's pages. Then, the eBook synthesizer requests, trough an appropriate API, related content (according to the user's options) from the *content provider.* Based on the received content, it composes and generates the interactive eBook (according to the ePub3 standard). Finally, the eBook synthesizer informs the *user-data database* about the eBook-id of the generated eBook, associating it to the user who requested it.
– *eBook progress synchronizer.* It is the component which communicates with the *save/sync server* (running on the user's android device) in order to sync the learner's progress. In this way, a complete reading/playing history for each is maintained at the eBook server which, in turn, can be used to update the learner's profile.
– *User-data database.* It is used to store user-account info, the generated eBooks, and playing/reading history for all users.
– *Content provider.* The **content provider**, is responsible for generating the content to be included in a new personalized eBook. In doing so, it also maintains (stores and updates) the learner's profiles. It employs a local *Model/Resources database* which stores domain-models (topics), related resources and mini-games. Upon request, the content provider generates appropriate educational content and forwards it to the eBook synthesizer system. The automatic content selection is based on the user's profile and playing/reading history. Learner profile maintenance and update is the task of the *user profiling and adaptivity component* (see Section 4).

  The content provider can also utilize external source in order to receive user modeling services and/or relevant resources. In Fig. 1, the iRead project[7] plays the role of the external infrastructure and provides access to linguistic resources (specialized dictionaries).

3 Domain Modeling in iRead

Domain models constitute a central component of iRead which inform all subsystems (NAVIGO Game, AMIGO Reader, Text Classification, Personalized eBooks) and determine the individualized pedagogical design used for each child. The role of each domain model is to (a) specify the linguistic structures, referred to as *domain features*, that must be mastered when learning to read as well as *progression schemes*, in the form of *prerequisites*, indicating the order of teaching of those features, and (b) provide the basis for the formulation of individual learner models, which enable the iRead system to record each learners individual strengths and weaknesses. The linguistic information included in domain models was especially selected to address all the linguistic sub-skills that are relevant or necessary for reading development and covers awareness of speech sounds (phonology), spelling patterns (orthography), word meaning (semantics), grammar (syntax) and patterns of word formation (morphology) [8]. As "learning to read" is language dependent, a specific domain model has to be developed for each language. Table 1 presents information about the size of the basic iRead domain models. Differences is size (e.g., German vs Spanish model) reflect not only differences in the language but also in the level of granularity different linguistic experts approached the modelling task.

Domain models are represented and stored within the iRead infrastructure as *weighted directed acyclic graphs* (*DAGs* for short). Vertices of the graph correspond to language features while directed edges indicate prerequisites. Naturally, in such a representation, there can be no directed cycle, while the non-negative weight on the edges allows to express the fact that two different features can be both prerequisites of a third feature, but one of them is more important than the other. In the case where a language feature has prerequisites, the sum of the weights of its incoming edges must be equal to 1.

Using weighted DAGs allows to efficiently instantiate *learner models*, most commonly referred to simply as *profiles*, and store a learners progress. Profiles are instantiations of domain models, which can be personalized to each learner via their interactions with the interactive educational components (smart literacy games and interactive eBooks). Profiles are also represented as weighted DAGs (in fact, a copy of the corresponding domain model graph), where each vertex is equipped with an additional attribute, called *competence*. Competence is a non-negative integer value showing the current mastering of the underlying language feature. When a learner exercises a particular language feature, the value of the competence can increase or decrease, depending on her performance. The value of competence cannot vary arbitrarily: for every language feature we define its

	English	**Greek**	**German**	**Spanish**
# of **Features** (vertices)	279	446	316	326
# of **Prerequisite relations** (edges)	4,457	17,552	748	17,290

Table 1. Size of iRead domain models.

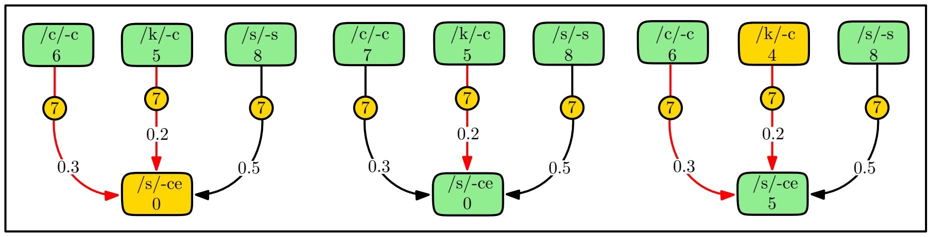

Fig. 2. Domain modeling: Unlocked and locked features. Unlocked features are shown in green, locked in yellow. Unlocked edges are drawn black, locked red. We assume that the *unlock_value* of all vertices is equal to 0.75 and the *competence_threshold* value for unlocking is 5. In the leftmost case, the bottom feature is locked as the sum of the unlocked incoming edges is equal to 0.5 (i.e., smaller than 0.75). In the middle configuration, the corresponding sum is equal to 0.8 (i.e., greater than 0.75) and becomes unlocked, while in the rightmost configuration, although the sum of the unlocked incoming edges is less than the *unlock_value* of 0.75, the competence of the feature is 5, equal to the threshold value. Thus, the feature becomes unlocked.

minimum and maximum value. By default, and unless otherwise specified within the domain model, we consider the minimum value to be equal to zero, and the maximum value equal to ten.

For a particular learner, not all language features are "available" for practicing. It is clear that a learner can practice a language feature if she has mastered the feature's prerequisites. In order to express the "availability" of a language feature, we equipped both vertices and edges of the graph with the *unlock_value* attribute. The availability status of a vertex (or edge) can be either *locked* or *unlocked*, i.e. available or non-available respectively. An edge becomes unlocked if the competence of its source is higher than the *unlock_value* of the edge, otherwise it is locked. Vertices can be also unlocked in three ways: (i) if a vertex has no prerequisites it is, by default, unlocked, or (ii) if the sum of the weights of their incoming unlocked edges is higher than the *unlock_value* of the vertex, or (iii) if the competence of the feature is higher than a *competence_threshold* value. As an example, consider the configurations shown in Figure 2: vertices are drawn as rectangular shapes and their label shows the corresponding *language feature* and the competence of the learner (e.g. the bottom vertex has competence 0 in the first two configurations and competence 5 in the third one). For the sake of simplicity, we assume that the *unlock_value* of all vertices is equal to 0.75 and the *competence_threshold* value for unlocking is equal to 5. The *unlock_value* for the edges is denoted within a yellow circle, while the weight of each edge is also present. An immediate consequence of the above unlocking procedure is that a language feature can become unlocked even if the learners competence on the prerequisites cannot unlock incoming edges. Hence, the sub-graph of the unlocked vertices and edges, is not necessary a connected graph. This property can prove very useful in the case where the initialization of a profile is based on an undertaken test; the domain model contains too many language features and it is almost impossible to test all of them. Still, the available unlocked vertices will allow the learner to practice different areas of the domain model in parallel.

4 Adaptivity and Personalizion

The ability of the iRead eBook System to provide personalized interactive activity books is based on its learner modeling capabilities. By maintaining learner profiles it can choose the proper language features to focus on and, in turn, choose appropriate activities relevant to these features to be included in the eBook. Finally, by consulting the playing/reading history of the learner, it can select from the available language resources the appropriate (for the specific learner) linguistic material to be used in conjunction with the chosen activities.

In the architecture of the iRead eBook system, services related to learner profiling and adaptivity as well as services responsible for maintaining domain models and related linguistic resources, are part of the *content provider* subsystem (see Fig. 1). Even though these services completely follow the logic employed in the iRead infrastructure, we have decided to provide a separate implementation as part of our eBook server. This allows for the incorporation of models (and their relevant resources) which are not covered by iRead (e.g., relevant to mathematics) and the production of personalized activity eBooks in the corresponding fields. However, as indicated in Fig. 1, it is possible to obtain this functionality from external resources, as the iRead infrastructure in our case, provided it is available. Actually, in our implementation, we utilize the iRead dictionary services by directly connecting to the iRead infrastructure.

Figure 3 provides a view of the architecture focusing on the components of the content provider as well as the flow of information that this relevant to profile adaptation and content selection, the topic of this Section. During the presentation, we should always keep in mind the typical work-flow of a learner: (a) she downloads an activity eBook, (b) she read it, i.e., she plays the activities in it and, when finishes reading/playing, (c) she orders a new activity eBook (more precisely, her parents do so). Also, keep in mind that when the learner is online, the *save/sync app* that runs on the tablet sends the learner's progress to the eBook server.

4.1 Adaptivity

When a new eBook is being ordered for a specific learner, we have to make sure that the learner's profile in accurate and reflects her reading skill state. If it is the first time that we generate an eBook for the learner in question, we have to initialize a new profile for her. This can be done by either adopting one of predefined initial profiles that fit the characteristics of the learner (e.g., age, school year, an informal classification) or by giving her a literacy test on the features of the domain model (currently, this option is not implemented in iRead). If we already have a profile for the user, we have to make sure that the profile is accurate. Assuming that the profile was last updated during the creation

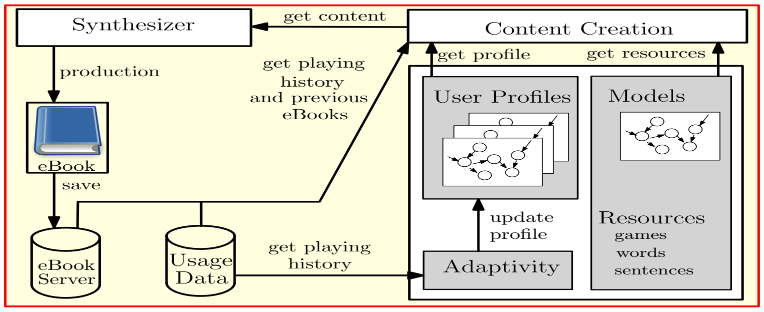

Fig. 3. The flow of information for the content provider.

of the last book ordered for the learner, we now have to *re-evaluate* it. It is the profile re-evaluation that guarantees that the activities and content included in the eBook are appropriate for the learner, i.e., the eBook is personalized.

The learner's responses for each activity in the eBook are all recorded in her reading/playing history. So, after reviewing the performance data for a specific activity, we should be able to decide whether the learner improved her competence on the feature targeted by the activity[5]. Typically, if a user has a high success rate over the few last played activities targeting the specific language feature, the competence is increased by one unit. In iRead (where the reading skill is improved by playing the NAVIGO literacy game, and the feature competence takes integer values in the range $[0 \ldots 10]$) we focus on the last three played activities and we request a success rate of 75% in all of them. In our activity eBook system we have adopted a lighter approach, that is, we allow feature competence to only take the values zero (0) and one (1) and we only consider the last two (2) played activities. The rational for doing so is based on our desire to (theoretically) allow the learner to master all language features in a reasonable amount of eBook pages. Assuming the user provides only correct answers, the English reading skill can be mastered by reading 558 pages[6]. In any case, these values can be easily adjusted. Note that, during reevaluation it makes sense to also reduce the feature competence if a learner provides too many wrong responses. This allows the learner to work again on improving her skills on language features she has not practiced recently and has forgotten them.

Updating the competence values for all features in the current reading/playing history of the learner completes the reevaluation process. However, we are not ready yet to proceed to the next stage; the selection of personalized content. Before we do that, we have to examine whether the changes in competence for some features results to unlocking more language features (which can be the

[5] Each activity aims to improve the competence of the learner in one specific language feature. It tries to achieve that by asking the learner to play an appropriate mini-game.

[6] The English model has 279 features. Playing 2 activities per feature requires $2 \times 279 = 558$ eBook pages in total.

topic of new activities). This process is triggered by the end of the reevaluation process. For each feature with an improved competence value, we check whether the feature that *depend on it* (i.e., they come immediately after it in the DAG representing the domain model) satisfy the conditions for unlocking them.

4.2 Personalized Content Selection

Each page of the activity eBook contains one activity. An *activity* is defined to be a triplet consisting of (a) a language feature, (b) a type of activity[7], and (c) a mini-game, together with appropriate educational content (e.g., words, sentences). Six mini-games used in the NAVIGO games have been adopted for use in the iRead eBook system and developed by iRead partners Patakis and PickaTale. A list of them together with a brief description is given in Table 2.

Mini Game	**Description**
Cleomatchra	The learner connects pairs of strings to form full words. PickaTale [16].
Perilous paths	The learner selects a string (word or sentence) over three possible options, to complete a sentence or answer a question. PickaTale [16].
Remove the runes	The learner selects all words from a given list that have a specific property. Patakis [15].
Sliceophagus	The learner splits compound words to stems. PickaTale [16].
Bridgyptian	The learner selects a number of string compounds from an available set and places them in an appropriate order so that she creates a word or a sentence. Some string compounds may have a predefined, fixed placement. PickaTale [16].
Raft rapid fire	Words appear sequentially on the learner's screen. The learner must "hit" all words that have a predefined property. Patakis [15].

Table 2. The Mini-games employed in iRead eBooks

Information about each activity is stored in a database and is accessed during the selection of personalized content. Note that the process of selecting the content required in order to play an activity may be quite elaborate. For example, we may be required to identify several words relevant to a feature (e.g., words starting with "b" as in "bag") but also *distractor words* relevant to another feature (e.g., words starting with "d" as in "dog"). These rules are specified as a collection of *filters* which are employed during content selection.

The *get_activity* service is responsible for determining the activities to be played. The selection of the activities is made by focusing only to those relevant to unlocked features in the domain model. Emphasis is given to unlocked features with competence below their threshold value. The reading/playing history database is also consulted. We want to avoid repetitions, we want to use as

[7] Possible activity types for literacy games are: *accuracy, automaticity and blending.*

many mini-games (relevant to the feature) as possible. The aim is to get a good mixture of activities so that we build an engaging eBook.

Having decided on which activities to include in the eBook, it remains to select content for each activity. In the case of iRead eBooks, this basically means selecting words and passing them through the filters mentioned above. Specialized dictionaries are used to complete this task. The dictionaries are specialized in the sense that each word is pre-processed in advance so that word selection is speed up and the application of the filters is made easier. An additional requirement is that the words have been carefully selected so that they cover all language features and they are appropriate to be presented to children.

The *get_content* service is responsible for returning content for each activity. The service identifies the words in the dictionary that are relevant to the activity's feature and pass all the filters. The word's *difficulty* (an attribute of each word stored in iRead's dictionary) is taken into account in the final word selection (the smaller the learner competence the easiest the words we select). In addition, we also consult the reading/playing history in order to avoid words that the user has seen too many times, to replay words that the user usually get wrong and, in general, get a good mix that avoids boring repetition.

In addition to the fully automatic activity selection made by *get_activity* we also provide a mechanism for the semi-automatic way to select the content of an activity eBook. It is semi-automatic in the sense that a teacher/parent can only specify the activity that appears in each page of the eBook[8]. The rational behind providing such a service is the need to allow teachers to build activity eBooks that reflect the specific teaching plan they use in class.

5 Conclusion

We presented the iRead eBook system that provides an integrated personalized learning service aiming to develop the learner's reading skills by engaging her in activities delivered through interactive eBooks. The system has been implemented in the framework of the iRead H2020 EU research project. There are several threads of research that are worth investigating:

– Experiment with different methods to select activities and content for each activity.
– Expand the available resources for the reading skill acquisition. More mini-games, more elaborate activities, larger dictionaries, more languages.
– Develop domain models for learning skills in other fields (besides "reading"). Mathematics may be an option.
– Consider potential privacy issues that may arise due to the use of profiling, for example, in the case that an eBook is addressed to a learner belonging in a specific group such as children with dyslexia. This question may be more relevant for services deployed and use in Europe due to the GDPR [5].

[8] Specifying both the activities and the content for each activity proved to be too complicated, especially when discractor words are also required) and too boring.

References

1. L. Benton, A. Vasalou, W. Barendregt, L. Bunting and A. Rvsz. Whats Missing: The Role of Instructional Design in Childrens Games-Based Learning In Proceedings of *CHI Conference on Human Factors in Computing Systems (CHI 2019)*
2. S. Droutsas, Pa. Patsilinakos and A. Symvonis. Interactive personalized ebooks for education. In *9th International Conference on Information, Intelligence, Systems and Applications, IISA 2018, Zakynthos, Greece, July 23-25, 2018*, pages 1–8, 2018.
3. ePub, "ePub Reader for Android", 2015, *http://www.graphilos.com/epub.*
4. Epub3, "Epub3, International Digital Publishing Forum", 2017, [online] *http://idpf.org/epub/30.*
5. GDPR, "General Data Protection Regulation. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC". *Official Journal of the European Union, L 119/1,* 4.5.2016.
6. "Gitden Reader", 2018, *http://gitden.com/gitden-reader.*
7. iRead: Infrastructure and integrated tools for personalized learning of reading skills. European Union Horizon 2020 funded project. https://iread-project.eu/
8. iRead deliverable: "D4.1 Dyslexia Domain Models (English and Greek)". Available online at *https://iread-project.eu.*
9. iRead deliverable: "D6.1 iRead Game Specification and Design". Available online at *https://iread-project.eu.*
10. iRead deliverable: "D7.1 iRead Reader app Integration and Visual Design". Available online at *https://iread-project.eu.*
11. iRead deliverable: "D8.1 System Architecture". Available online at *https://iread-project.eu.*
12. JavaScript, "JavaScript, MDN web docs", 2018, *https://developer.mozilla.org/en-US/docs/Web/JavaScript*
13. Kotobee, "kotobee Reader", 2018, *https://www.kotobee.com/en/products/reader*
14. S. Okuda and K. Emi, "Make once, play anywhere!: EPUB 3 interactive function enables us to make and play game software anywhere!," 2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE), Tokyo, 2013, pp. 381-384.
15. Patakis Publishing, *http://www.patakis.gr.*
16. Pickatale, *https://www.pickatale.com.*
17. Emma Sumner, Elisabeth Herbert, Laura Benton, Nelly Joye, Manolis Mavrikis, Mina Vasalou. "How critical engagements with design can identify gaps and create new evidence: A case study of the iRead project", Poster presented at the *1st European Literacy Network Summit (ELN 2018)*, Porto, Portugal, 2018.
18. Supreader, "ePUB EBook Reader", 2017, *https://play.google.com/store/apps/details?id=com.s2apps.reader.*