POLITECNICO DI MILANO
Scuola di Ingegneria dell'Informazione

POLO TERRITORIALE DI COMO

Master of Science in
Computer Engineering

TAILORED BOOK: A DISTRIBUTED PLATFORM FOR AUTHORING AUGMENTED COMMUNICATION BOOKS WITH LANGUAGE CORPORA-REFERENCED ANNOTATIONS

Supervisor: Prof. Thimoty Barbieri

Master Graduation Thesis by: Alexey Khlebtsov
Student Id. number: 751892

Academic Year 2011/12

POLITECNICO DI MILANO
Scuola di Ingegneria dell'Informazione

POLO TERRITORIALE DI COMO

Corso di Laurea Specialistica in
Ingegneria Informatica

LIBRO SU MISURA: UNA PIATTAFORMA DISTRIBUITA PER LA CREAZIONE DI LIBRI CON COMUNICAZIONE AUMENTATIVA UTILIZZANDO ANNOTAZIONI BASATE SU CORPORA LINGUISTICI

Relatore: Prof. Thimoty Barbieri

Tesi di laurea di: Alexey Khlebtsov
Matr: 751892

Anno Accademico 2011/12

Abstract

This project is an endeavor to facilitate the development of communication and language for children with complex disabilities by means of tailored books. The tailored book or “su misura” book[10] is the well-known tool invented in Italy used to help children at a very early stage in all situations where there is at least the slightest doubt of possible difficulties in the future development of communication.

The process of tailored books’ production is not easy and time-consuming, because there is no specifically adapted software available for this purpose. In this project I would like to propose a system that allows the creation, modification, visualization and distribution of such books.

The system is composed of the desktop and mobile applications that are called “Tailored Book Editor” and “Tailored Book Reader” accordingly. “Tailored Book Editor” is responsible for the creation, editing, and publication of such books. “Tailored Book Reader” is responsible for visualization of tailored books created by “Tailored Book Editor” application.

“Tailored Book Editor” takes as input a .txt file and a set of tagged images and produces a tailored book. The “Tailored Book Editor” contains also “Image Tagger” module which allows tagging images with the reference to lexical database.

The creation of a tailored book is carried out from a text file that is then tagged using the POS tagger for assigning the part of speech to each word. The couple of the word and its part of speech is then sent to the lexical database WordNet that is used to find all synonyms for each couple. These synonyms or in other words tags are then assigned to each word of initial text file for further image assignment. The difference between text tagging and image tagging is that the image tagging requires supervision while the text tagging is done automatically. The design and implementation of such system are described below in this project.

Sommario

Questo progetto è un tentativo di facilitare lo sviluppo della comunicazione e del linguaggio per bambini con disabilità complesse per mezzo di libri su misura. Il libro su misura [10] è il ben noto strumento inventato in Italia che viene utilizzato per aiutare i bambini in una fase molto precoce in tutte le situazioni in cui c’è almeno il minimo dubbio di difficoltà nello sviluppo futuro della comunicazione.

Il processo di produzione di libri su misura non è facile e richiede tempo, perché non c’è alcun software specificamente adattato per questo scopo. In questo progetto vorrei proporre un sistema che permette la creazione, la modifica, la visualizzazione e la distribuzione di tali libri.

Il sistema è composto dalle applicazioni desktop e mobile che si chiamano rispettivamente “Tailored Book Editor” e “Tailored Book Reader“. “Tailored Book Editor” è responsabile per la creazione, la modifica e la distribuzione di tali libri. “Tailored Book Reader“ è responsabile per la visualizzazione di libri su misura creati da applicazione “Tailored Book Editor”.

“Tailored Book Editor” prende come input un file .txt e un set di immagini marcate e produce un libro su misura. “Tailored Book Editor” contiene anche “Image Tagger”, un modulo che permette di etichettare le immagini, con il riferimento al database lessicale.

La creazione del libro su misura viene effettuata da un file di testo che poi viene etichettato con POS tagger per assegnare le parti del discorso ad ogni parola. La coppia della parola e la sua parte del discorso viene poi inviata al database lessicale WordNet, che viene utilizzato per trovare tutti sinonimi di ogni coppia. Questi sinonimi - o in altre parole i tag - vengono quindi assegnati a ogni parola del file di testo iniziale per ulteriore l’assegnazione delle immagine. La differenza tra il tagging dei testi e il tagging dell’immagine è che l’annotazione delle immagine richiede la supervisione, mentre l’annotazione dei testi è fatto automaticamente. La progettazione e l’attuazione di tale sistema è descritto di seguito in questo progetto.

Index

I. Background

1. Context

1.1. What is AAC?

For most people, communication involves the spoken language. However, for some, the ability to hear speech or to speak is compromised. There are children who cannot use speech because they have difficulty moving the required muscles, developmental disabilities, or hearing impairments. For these children particular speech communication strategies, called augmentative and alternative communication (AAC) systems, may be useful. AAC systems can augment existing communication skills or provide an alternative to speech [1].

The use of the AAC was initially limited to children with severe mobility problems and normal cognitive development, however over the years it has been gradually extended to many different disabilities affecting intentional communication (autistic disorder), the expressive components of language and motor skills (cerebral palsy, spinal muscular atrophy, muscular dystrophy), language comprehension (some dysphasia) or more often with mixed components (Angelman syndrome, Down syndrome and other genetic syndromes, mental retardation, severe dysphasia, progressive neurological diseases, etc.)[2].

For some students with autism, cerebral palsy, Down syndrome, and other disabilities, speech may not fulfill all of their communication needs. For these students with complex communication needs (CCN), the use of Augmentative and Alternative Communication (AAC) can have important benefits [3].

There is a wide range of AAC techniques available, including:

- Picture, symbol, alphabet, and word boards
- Signs and gestures
- Speech-generating devices (including specialized computer systems)

The use of AAC systems can support participation for students at any age in a wide range of academic and social activities [3].

1.2. Children with special needs - who are they?

Over the past 20 years, the number of students with disabilities has been steadily increasing at a faster rate than both the general population and school enrollment [4]. Children who use AAC have needs similar to the needs of other students in the classroom. They require educational programs that are rigorous and relevant and that promote positive relationships [3].

There is a fair quota of children with different types of disability that temporarily or permanently have serious problems of linguistic communication within surrounded environment. The ability to establish a form of communication, and then get in touch with others is essential to the relational and cognitive development as well as to ensure the acquisition of a social role in the life. Epidemiological data shows an area of substantial need [5]. About 2% of the Italian populations before 18 are disabled, and at least one quarter of them has communication disorders, transient or permanent, for a total of at least 50,000 children and teenagers [6].

1.3. Speech processing

The timely interventions of AAC can play an important role in preventing both further symbolic and cognitive impoverishment and the appearance of minor psychiatric disorders.

For Children and adolescents with complex communication disabilities it is necessary to enhance the existing communication strategies through Augmentative and Alternative Communication, in order to bypass the communication deficit and reactivate interactions and development. Symbol or photo systems may be used, digital recorders that may “lend” the voice when necessary, assistive technology, or systems that allow functional reading and writing for people that have not reached alphabetical access or cannot use a pen. The system has to be multimodal in nature, flexible and sized for each individual; it has to follow the changing needs of the child due to growth and actively involve family and life context, according to a participated model.

Almost all children with speech and language difficulties have associated literacy problems. However, school-age children whose speech difficulties persist beyond 5 years of age are most at risk for associated difficulties in reading, spelling and sometimes maths. Let's first consider the nature of persisting speech difficulties with reference to a simple psycholinguistic model of speech processing (see Figure 1).

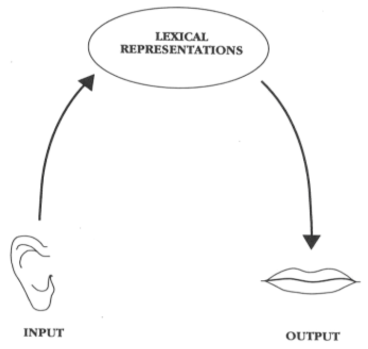


Figure 1 Psycholinguistic model of speech processing

Figure 1 illustrates that we receive spoken information through the ear (input). The information is then processed as it goes up the left hand side of the model and is stored at the top in a word store (lexical representations). When we want to speak we can access stored information and program it for speaking on the right hand side of the model (output). Some children with speech difficulties have difficulties with speech input (e.g. differentiating between similar sounding words); others have imprecise or "fuzzy" storage of words which makes it difficult to access them (as in word finding difficulties) or to program a clear production of them because of missing elements in the word store; while others have a difficulty pronouncing speech at an articulatory output level (on the right hand side of the model) even though they know the words involved perfectly well. Children with persisting difficulties, however, may well have pervasive problems which involve all of these aspects of speech processing: input, representations and output. Where this is the case they may also have language difficulties (comprehension and/or expression).

The speech processing system, as illustrated above, is not only the basis for speech and language development but also the foundation for literacy development; "written language" being an extension of 'spoken language'. For example, if a child has delayed understanding of spoken language he will find it very hard to access meaning from the printed word even though he may be able to decode the letters perfectly well. Sometimes, children with comprehension or "semantic-pragmatic" difficulties are described as "hyperlexic"; this term indicates that a child can read print mechanically better than they can understand it. Other children, particularly those with persisting speech difficulties, have a problem with the mechanics of reading and are more likely to be described as "dyslexic" or as having 'specific' reading and spelling problems. This suggests a problem at one or more levels in the speech processing system depicted above.

In summary, children's speech difficulties arise from problems at one or more points in their underlying speech processing system. This system is the foundation for their written language as well as their spoken language skills. If this foundation is unstable, additional support will be needed to enable a child to use the strengths he has to develop phonological awareness skill and letter knowledge. This is tough but not insurmountable [7].

1.4. Tailored books or books “Su Misura”

An AAC system is a kind of “ongoing immediate decoder” between the child’s language and our own, and vice versa. The systems of symbols or images can be used to overcome the communication deficit and reactivate and enhance interactions and development.

In this system every framed figure is composed of the image or symbol, and the word it represents written above or below, to make it understandable to the communication partner (see Figure 2).

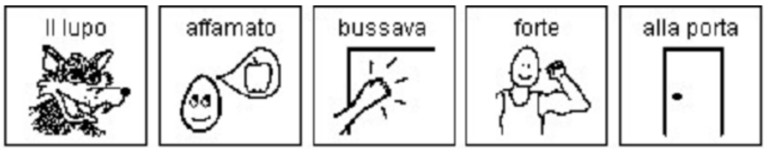


Figure 2 Text in symbols

This is not only a rehabilitation technique, but also may be applied to build a flexible system tailored to each person and put in place in all contexts; since the communication is a fundamental right of the individual, and for each of us it is necessary and essential in every moment, and not just in the therapy room.

These interventions are very popular abroad, particularly in English-speaking countries that have always been very attentive to the issue of the rights and quality of life, but still very present in Italy. Also due to cultural lack of communication in situations of disability, there is an unfortunate persistence of prejudices that interfere with the success of interventions (for example, the fantasy that the use of AAC tools prevents the structuring of a verbal language rather than facilitate it).

A fair chunk of small users simultaneously presents a number of problem areas that require continuous interaction of different professional specialties (child psychiatrist, psychologist, speech therapist, psychomotor system developmental therapist, physical therapist, professional educator, information technology, etc.) in a global, complex and detailed context.

After a lot of debate, the literature has now agreed that there are no minimum prerequisites needed in the children (so there is no a low cognitive level, or severity, or age below which it is not advisable to start), there are instead the minimum specifications of rehabilitation services and the environment that are essential [8-9]. In the context in which the child psychiatry or rehabilitation services are missing or do not have professional training in augmentative communication, it is very difficult to put in place a system of CAA since the local technical support are missing in the school and family.

Also it is not enough to have the best and most refined technical assessment or the best rehabilitation service to succeed the interventions. Since the role of the environment has a lot to do with it, so the intervention must permeate all stages of a child’s life: active collaboration at school, family, etc.

There are two areas of particular importance in AAC intervention: the initial communication and the constant updating of vocabulary and in both areas the active role of the living environment is absolutely crucial. The pediatrician’s knowledge is important to adequately support the ongoing interventions from outside, and in the case of the initial communication it also have to raise awareness among children’s families in the direction of the interaction procedures that facilitate the development of communication and language.

The basic facility which is recommended to use is called “tailored” books [10], which are helpful at very early stage (in the first or second year at most) in all situations where there is at least the slightest doubt of possible difficulties in the future development of communication.

A customized book or “tailored” book is a book “on measure for a specific child”. It is a particularly important tool for children with complex disabilities, who without specific attentions and adaptations would not be able to have access to the book-sharing. Through the experience of making tailored books for individual children, it has become possible to create group “laboratories” in partnership between parents, teachers and health professionals and then to organize a specific dedicated section of the town library, to act on participation barriers in the community [11].

The production of tailored books or books “su misura” is specific for each child and is very burdensome in terms of time for families and professionals.

The presence of tailored books in public libraries and nursery schools also could go to support two other populations, whose access to the book is generally limited: children with language disorders and children of migrants. The translation of the text into symbols which are present in the tailored books allows children with disabilities following much better the text, improving greatly the attention and language comprehension, as well for the children who have no cognitive problem but, for example, have a disorder of language development [13].

1.5. The translation of text into figures

The text is the part that requires major adaptations, both the structure of sentences, which must be simplified without losing its liveliness and content, and in the subsequent translation of the text into figures. In some cases it is then necessary to change the number of pages, their consistency and assembly. The first books for a child with complex disabilities cannot be built entirely at his measure, and consequently require the working together with parents, teachers and practitioners, that through special training workshops can learn how to create customized books.

A personalized or tailored book is a book built from scratch for a single child on a specific topic that has for him a high motivational engagement (a significant event in a positive or negative: a holiday, a celebration, an event, or otherwise in a shelter hospital).

It can also be very short, very few pages, and it is generally the book which a very small child with disabilities or with special difficulties (with autism, with difficulties in language comprehension or mental retardation among moderate and severe) should be able to grab for the first time.

In the construction of the book it is important to pay attention to the correctness of the translation of the text into figures: all text should be written in figures, in black and white or in color, squared, respecting the structure of the sentences. The completeness of the text in figures, along with the modeling of the adult allows the children to follow better the story they are listening, supporting language comprehension and attention.

It is also possible to increase gradually the level of richness of sentences and complexity of the used figures, adding functors (small parts of speech that modulate the words) and morphosyntactic elements as the child’s attention to the nuances of speech increases.

1.6. Many figures or a few figures?

At first the text of the books should contain a few figures in order not to confuse the children with too many not yet known ones. It is like trying to use only a few words with young children who are starting to listen to the language. Also it was realized that using a few figures may complicate things instead of simplifying them for those children who listen to parent’s reading [39].

If there are only a few figures and you do not understand the words you hear, you are in the same situation of a person who can neither understand nor read Russian, and sees and hears the following phrase (see Figure 3):

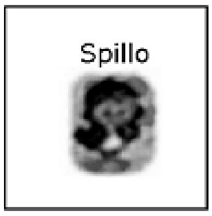
Я не могу петь рыдала 

Figure 3 Many figures or a few figures, example 1

It might mean anything that concerns Spillo: e.g. it is happy or sad, it is fine or sick, it is hungry or ate too much, and it goes out or comes back, etc. (see Figure 4)

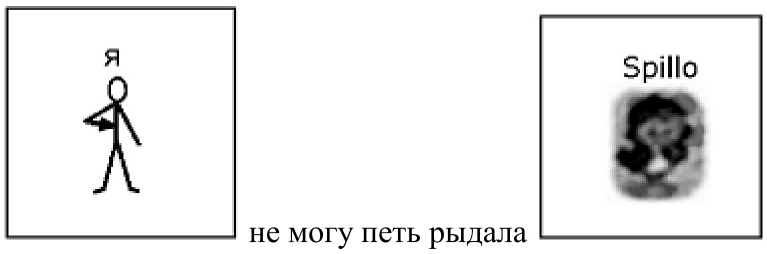


Figure 4 Many figures or a few figures, example 2

Even the addition of the subject does not improve the situation. Me and Spillo. Are we friends or enemies? Do we do something together or we do not want to do something together? (See Figure 5)

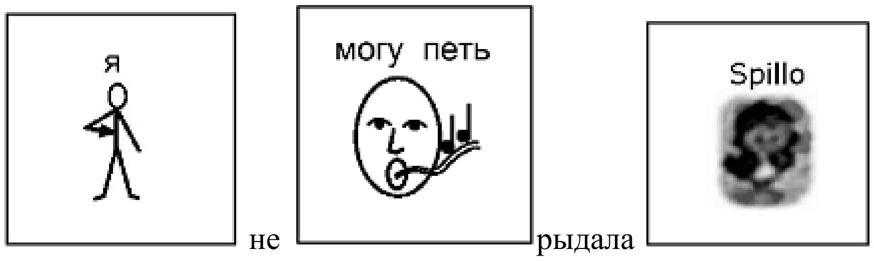


Figure 5 Many figures or a few figures, example 3

Right now it seems a bit better, with the addition of the verb, but there are still many doubts. Spillo is a bird, maybe he will sing or I will sing for Spillo. (See Figure 6)

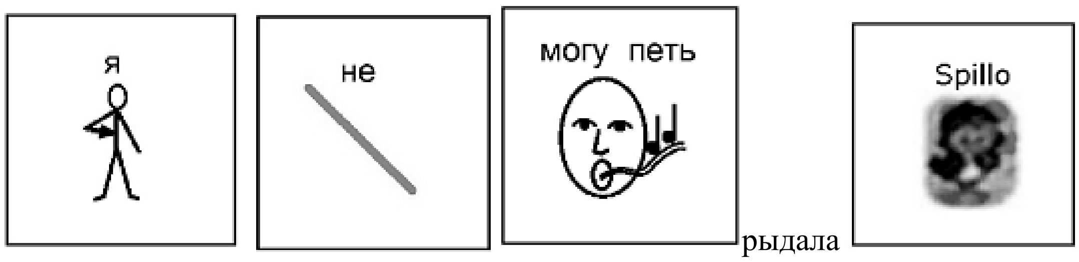


Figure 6 Many symbols or a few symbols, example 4

The missing part was an essential piece which changes the meaning of the phrase completely: I cannot sing. (See Figure 7)

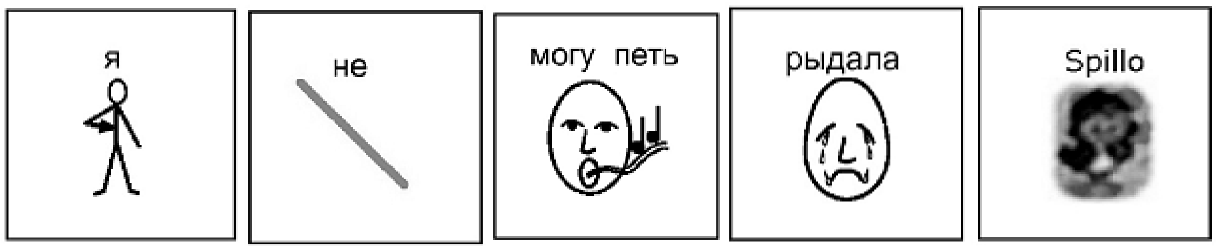


Figure 7 Many figures or a few figures, example 5

Here, finally, the whole sentence. I cannot sing cried Spillo. The sight of figures helps a lot to who do not understand Russian.

Especially the children with communication disorders can find it hard to decipher the nuances of language. Also it is really helpful to read aloud to children and indicate the figures while reading. With the text entirely in figures, the process of reading and indicating figures becomes incredibly natural. It is enough to see it only once as it should be done, and adults and peers are able to do it spontaneously [12].

1.7. The parts of speech

Initially every little part of speech should not be represented by a figures right away, because people who are at the beginning of learning a language focus mainly on the meaning and do not pay attention to the nuances and parts of speech.

At the beginning the phrase should be formed by the key elements necessary to understand its meaning (see Figure 8):

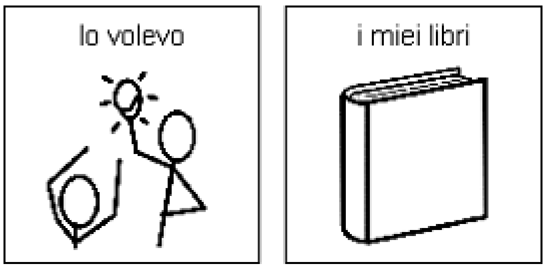


Figure 8 The parts of speech, example 1

Then gradually, the complexity can be increased by introducing the subject. (See Figure 9)

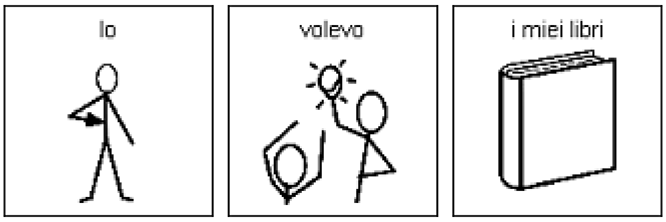


Figure 9 The parts of speech, example 2

Afterwards the possessive can be separated. (See Figure 10)

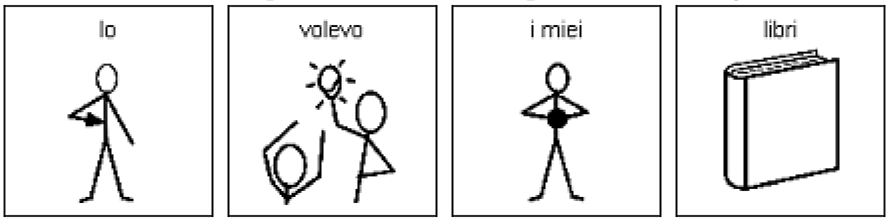


Figure 10 The parts of speech, example 3

Then the article and the plural can be added as a separate symbol. (See Figure 11)

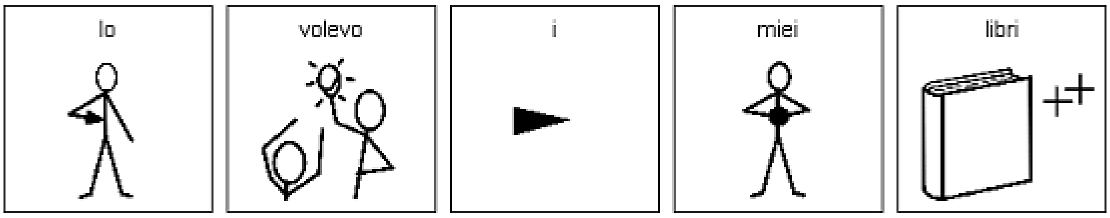


Figure 11 The parts of speech, example 4

And finally, the tense of the verb can be highlighted.

The introduction of morphosyntactic elements in writing symbols supports the language comprehension of children, while exposure to over-simplified language and low morphosyntactic elements cannot fully master the language [13].

1.8. Framing

The framing is an important element as well. The presence of the external frame in fact defines the unit of meaning that binds the written word and symbol together, facilitating the assignment of meaning. It is especially important for children who have difficulty in understanding the language. Here is the same text in symbols with (see Figure 12) and without (see Figure 13) external frame [13].

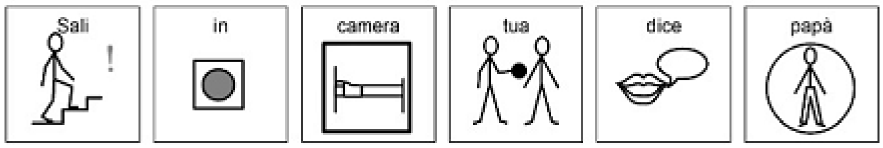


Figure 12 The text in figures with external frame

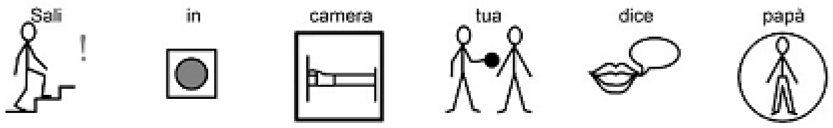


Figure 13 The unframed text in figures

2. Technologies

2.1. Classification of ACC Systems

Advances in computer technology have led to the creation of specialized devices called augmentative and alternative communication (AAC) devices, which help make it possible for individuals with no speech, or individuals with poor speech, to overcome their communication problems.

Augmentative devices are designed to support or enhance the speaking capability of a person. Alternative devices, on the other hand, replace speech as a means of communication. There are a variety of electronic AAC devices on the market, ranging from very low tech to very high tech, and ranging in price from a few hundred dollars to several thousand dollars. Some devices are “dedicated”, that is, their only purpose is to provide a means of communication. Other devices have been designed to work in conjunction with a computer that plays multiple roles (such as word processing or calculations). In addition, existing computers can now be modified for use as an AAC device through the addition of special communication software and hardware. These modifications are often less expensive and more flexible than many custom-built AAC devices [14].

AAC systems vary in terms of their portability, complexity, input method, vocabulary representation format, and means of output delivery. Selecting an appropriate system must be tied to the needs and capabilities of the student. For example, students with physical or mental disabilities who cannot use a standard keyboard can use alternative input devices, such as touch-sensitive pads, selection switches, or optical pointing devices.

For students who have difficulty with vocabulary, AAC systems have been developed to allow communication through word selection devices or even devices using pictures and graphics.

To assist students with disabilities in delivering a message, various speech and print output devices have been developed.

Today, many communication devices incorporate either synthetic or digital speech output. Synthetic speech is artificially generated by the computer, while digital speech is an actual recording of human speech stored in the memory of the device. Written output can be provided by printers that are built into the communication device or attached externally, but this option is cumbersome because of the large amount of paper required. As a result, some devices use liquid crystal displays (LCDs) to show students' messages—some displaying a single line of text at a time, some displaying multiple lines of text, and some using both the LCD and speech output together. AC systems are categorized in several ways (Heller 2004). Below is common terminology:

No technology (no tech): These AAC systems involve only the individual's body. Some systems involve formalized languages, such as American Sign Language (ASL). A simpler system may involve gestures unique to a particular child that the caregiver understands (for example, when a child touches her cheek to ask for a hug).

Low technology (low tech): Low-tech systems are nonelectronic but involve materials outside the child's body. They include non-electronic communication boards and graphic symbols. The technology involved in the use of these tools can be either pointing-based (i.e., individual points to a symbol in order to communicate) or exchange-based (i.e., individual hands over a graphic symbol in exchange for an object or activity delivered by the communication partner). The popularity of an exchange-based approach, the so-called Picture Exchange Communication System (PECS), led to further increased acceptance of low-tech AAC approaches with the population.

High technology (high tech): The application of high-tech special-purpose AAC hardware and software emerged for adoption. These tools were used primarily for expressive communication purposes. They use an integrated circuit and were developed for the special needs population [15]. High-tech systems include the use of electronic communication boards and/or computerized speech synthesizers. Electronic communication boards have a display of communicative messages using photographs, line drawings, phrases, words, or letters. The message is activated by touch or laser beam to produce a printout or synthesized/digitized speech.

As the use of AAC for persons with disabilities became more widespread, it expanded beyond simply its traditional use for expressive communication alone, and was used to augment comprehension and to provide organization of time and sequences.

Despite these advances, the AAC systems described above are often expensive, cumbersome, and time-consuming to program and personalize. Often, they serve to stigmatize the user. These barriers may prevent many individuals from using their systems on a regular basis.

More recently, however, the relatively expensive, dedicated hardware platforms and software programs have been subject to competition from less costly, consumer-level tools developed for the general market. For some individuals, communication needs can be met by using consumer-level hardware (e.g., personal laptop computer, tablet computer) to run special-purpose software designed to support communication. The examples of consumer-level hardware running special-purpose software include personal laptop computers with Speaking Dynamically Pro®, tablet computers running Clickit! and personal laptops with Pogo Boards.

For others, special-purpose software is not necessary, and communication needs can be met through a combination of consumer-level hardware and general-purpose software. Examples of consumer-level hardware running general-purpose software include tablet computers running Microsoft Word and personal desktop or laptop computers running Microsoft Powerpoint to present dynamic displays.

Low-cost, widely available peripheral devices (e.g., cameras, camcorders, DVD players) offer even more opportunities for AAC users to create, edit, store and present content in ways that do not rely on expensive, dedicated tools. Thus, relatively affordable, consumer-level hardware, software and peripherals now afford opportunities for greater flexibility in selection and design of AAC systems for users with disabilities [15].

2.2. Speech-generating devices

There are many varieties of AAC devices available as communication supports. The availability of "speech" from a device is often very motivating to a student with disabilities, and is especially useful for communicating in a group or with partners who are not familiar with other methods of AAC. These devices typically contain both pre-programmed vocabulary as well as vocabulary important to that particular student (e.g., names of family members, the vocabulary needed for school activities, etc.) that has been programmed into the device by an adult who knows the student well. Some devices speak the programmed vocabulary aloud using synthesized speech, while others make use of "digitized" speech or short recordings of human voices.

Pictures, symbols, letters, and words can be represented and organized in many different ways according to the user.

Students also can operate AAC devices in a number of ways. In addition to touching a keyboard or screen, students can use their eyes to control a device with "eye-tracking" technology. They can also use switches to scan pictures, words, and letters on an AAC device or a computer. An occupational therapist can help identify the most effective way in which people with severe physical disabilities can operate AAC technology.

2.3. Barriers to effective use of technology for students with disabilities

Many technologies previously described are readily available for use by individuals with different types of disabilities and are already providing many students with special needs an opportunity to be educated alongside their nondisabled peers.

However, several barriers inhibit more widespread use of these applications and devices, especially inadequate teacher training and cost.

Lack of adequate teacher training has an especially strong impact on students with disabilities because technology is often a critical component in planning and implementing an educational program for these students.

In addition, use of technology for multimedia projects, for example, can be very motivating for students with disabilities. But classroom teachers must have a deep understanding of what they are trying to accomplish and how technology can help them achieve their goals [16]. Thus, to meet the needs of students with disabilities within regular classrooms, all teachers, both those in regular education and those in special education programs, need training in how technology can be used, and the technical skills to carry out a plan of action [17].

The cost of the technology needed to help students with disabilities participate in regular classroom settings, especially the computer systems needed for students with more severe disabilities, is also a serious consideration for all schools. Such systems often must be tailor-made for each student and can be quite expensive, costing tens of thousands of dollars.

2.4. AAC and acceptance

There is little published information about why individuals with disabilities accept, reject, or abandon AAC strategies. There is some evidence that psychosocial factors may influence outcomes. Individuals with disorders may experience frustration, depression, denial and loss of autonomy [18]. Some people may find it difficult to accept that their natural speech has not returned [19] and may feel anxious about communicating in public [20]. Others may feel that AAC methods are too effortful to use or that their partners do not encourage AAC methods. In a case report, Lasker and Bedrosian [21] found that a person with disability was reluctant to use a device in public because he felt ashamed about his communication and feared that people would think he was abnormal. Fox and Sohlberg [22] found that people were more motivated to use AAC strategies when they conversed about personally relevant social roles.

Speech pathologists in medical settings often train individuals to use low-technology communication aids to help them communicate basic needs and make choices. Interventions may deemphasize other equally important communicate functions, such as social closeness and opinions. This type of clinical decision making may inadvertently foster rejection of AAC devices and cause feelings of isolation and partner dependence.

2.5. Emergence of mobile devices

With the current widespread availability of general-purpose portable hardware (e.g. Apple iPhone, Google Android) running specialized AAC applications, new opportunities now exist for AAC users. In fact, the adoption of the new portable hardware and software may suggest a significant paradigm shift in AAC.

Not only the devices themselves have become smaller and multi-functional, but the number of communication Apps and tablet platforms is increasing more rapidly than AAC hardware or software ever did. Up to now there are hundreds or more Apps that can meet the needs of individuals with disabilities.

Mobile touch screen devices cost much less, are readily available, and what is more important for children is that they are socially acceptable. Mobile technologies offer a broad spectrum of communication options as well as other functions.

As a more affordable addition or alternative to PC-based AAC devices, mobile devices provide a much larger market of consumers who can afford AAC technologies, compared to users who may have relied on third party funding. Some families no longer need to wait for lengthy insurance reviews, denials and appeals to determine eligibility. And like the shift to PCs before, mobile devices offer a universe of non-AAC applications such books, photos, games, movies, music, the Internet, and educational and personal productivity software, among others.

Many of the apps designed for these devices (e.g., Proloquo2go, MyTalk) may serve as full AAC systems, similar in many ways to the specialized systems described above, while others (e.g. Steps, First-Then, MyChoiceBoard, PicCalendar) may also provide support for the organization and enhance the efficiency of simple functions such as choice-making. Apps are often easily obtainable, affordable, customizable, and user-friendly [22].

II. Motivation and Goals

The goal of the project is to assist the development of communication and language for children with complex disabilities by means of tailored books.

Unfortunately, there are some difficulties with the creation and distribution of such books:

1. So far, to create tailored books, the standard office applications, which are not specifically adapted for this purpose, have been used. As a result, the process of tailored book creation is hard and time-consuming.
2. There are also no interactive applications for their visualization, because tailored books are available in hard copy, which in turn complicates and slows down the learning process.

Thus, the relevance of my thesis is in detailed examination of the above issues, the development and implementation of software solutions for these problems, which in turn facilitates and accelerates the education process for children with disabilities.

In the project I set the main objective to design, develop and implement a system that allows the creation, modification, visualization and distribution of tailored books.

The main objective can be split in the following sub objectives:

1. Design and implementation of a specialized desktop application to simplify the process of creating, editing, and distribution of such books.
2. Design and implementation of an interactive mobile application to visualize such books on smartphones and tablets with OS Android.

This system should kill two birds with one stone, because firstly it facilitates the creation process which by-turn attracts more and more people to the creation of such books and therefore attracts attention and the public interests to children with special needs; secondly, for the visualization of tailored books mobile devices such as smartphone and tablet are used to make the book interesting and entertaining. The app is also capable to read books aloud or pronounce selected words using synthesized speech, which is often very motivating to a student with disabilities.

Moreover mobile devices are socially acceptable and widely available in all sectors of society including people with special needs. Mobile devices with Apps are increasing more rapidly than specialized AAC systems and being readily adopted by people with disabilities, because of the affordability and the accessibility.

This work appeals to families of very young children, individuals with special needs who may not traditionally have thought about AAC.

III. Design and Implementation

1. Overview of the system

The proposed system is capable of creating, editing, visualizing and distributing of tailored books. In the Figure 14 the overview of the system is depicted.

The system is composed of two parts implemented by me in this project: a desktop application and a mobile application. The desktop application is called “Tailored Book Editor” and is devoted to the creation, revision and export of tailored books. The published or exported tailored book consists of the book itself in the JSON format and the folder with the images associated with this book.

The “Tailored Book Editor” aggregates two modules implemented by me; one is called “Tailored Book Creator” and another one is called “Image Tagger”. The “Tailored Book Creator” is responsible for creating tailored books. It uses dialogs (pop up windows) to collect information about the new book and leads the user through the process of tailored book creation. The “Tailored Book Editor” uses the text file of a book as input. The text is then tagged using the Stanford POS tagger for assigning the parts of speech to each word. The Stanford POS tagger integrated in this project will be described in detail in the section “Tailored Book Editor” back-end. The couples of the word and its part of speech are then sent to the lexical database WordNet, for retrieving all synonyms for each couple. The detailed description of the utilized interface to the WordNet and the lexical database itself will be given in the section “Tailored Book Editor” back-end. The synonyms, or in other words tags, are then assigned to each word of initial text file for further image assignment. In order to create a tailored book from the text the “Tailored Book Creator” requires the tagged images. Here comes the “Image Tagger” by means of which the images can be tagged with the reference to the WordNet.

The “Tailored Book Reader” is a mobile application dedicated to visualizing tailored books created by the “Tailored Book Editor”. The “Tailored Book Reader” is able to download tailored books from the server and save them on the SD card of the mobile device if it is available or in the built-in storage. The books can be put directly on the SD card and be read by the “Tailored Book reader”.

The design and implementation details of the system are described below in this chapter.

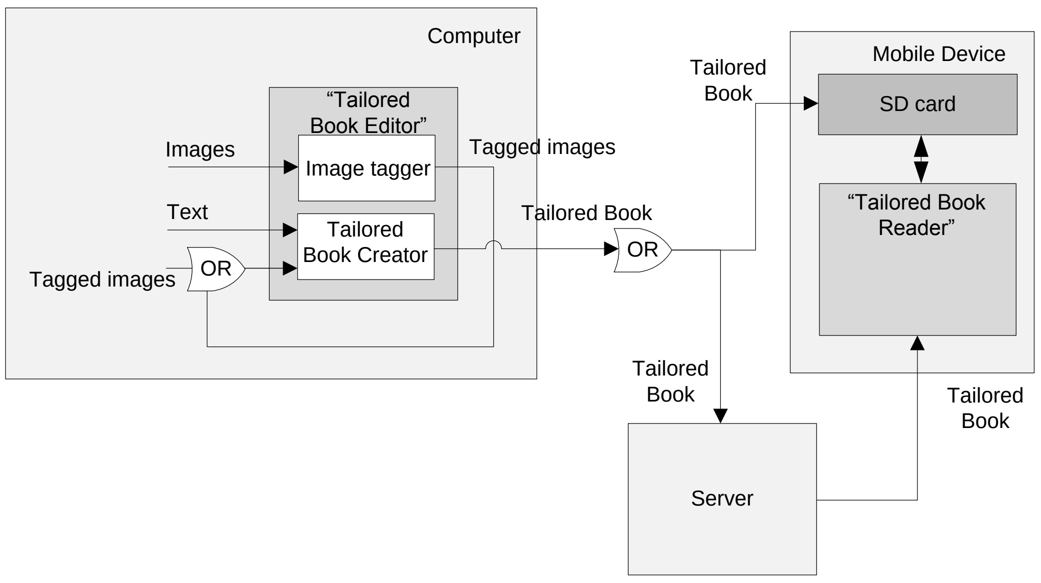


Figure 14 Overview of the system

2. "Tailored Book Editor"

2.1 "Tailored Book Editor" front-end

2.1.1. "Tailored Book Creator" features

A tailored book creator dialogs are responsible for creating a new book. The first dialog displays when the user goes to the main menu: File → New Book and it requires the user to insert information about a new book. Its functionality is encapsulated within the BookDetails class. The functionality of other dialogs: for selecting book .txt file and setting the directory for a "trained" file for Stanford POS tagger and setting the directory for images for further assignment is encapsulated within the Editor class. The description of BookDetails and Editor classes will be given below in the "Tailored Book Editor" back-end section. The "trained" file has the extension ".tagger" and is used for the part of speech tagging. The detailed description of this file will be provided below in the "Tailored Book Editor" back-end section.

The tailored book creator dialogs allow the user inserting a title and an author of the book and a logo. The logo will be displayed in the "Tailored Book Reader" application on the "Download Books", "My Books" screens. The detailed description of these screens will be given in the "Tailored Book Reader" front-end section.

The "Tailored Book Editor" is in charge of checking the correctness of inserted symbols using the regular expressions. After clicking on the «Create» button another dialog appears which allows selecting .txt file of the future tailored book for further processing. After selecting the .txt file another dialog appears for selection of the "trained" file for a Stanford POS tagger. This dialog is shown only once before the creation of a first tailored book, and then the "trained" file is saved and becomes the default option. The next dialog that allows setting the image source directory is prompted. The image source directory is used during tailored book creation when for every figure the appropriate image is assigned. The assignment is occurred if there is a match of tags between image and the item. Below in the section called System usage workflow I tell in detail about Image source directory. In the Figure 15 a mockup of a tailored book creator dialog is shown.

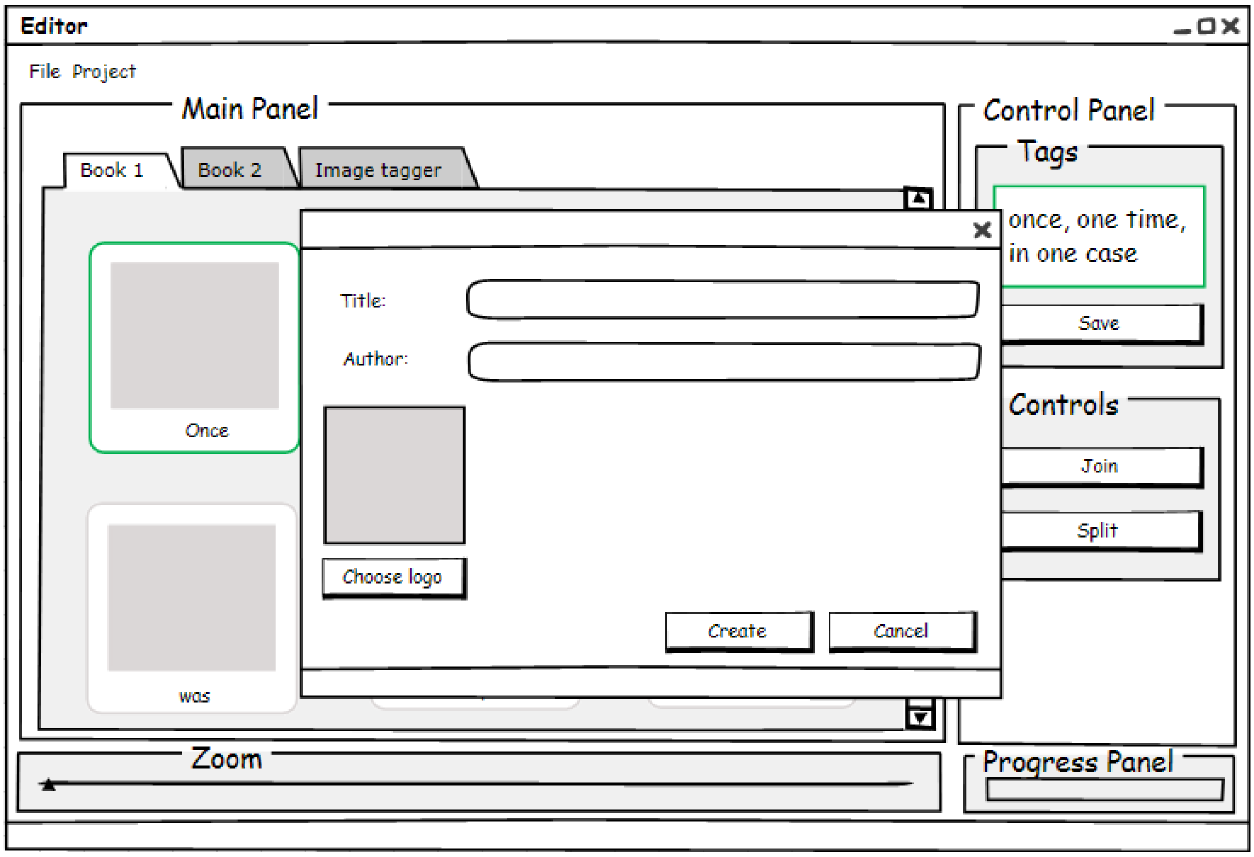


Figure 15 Mockup of the "Tailored Book Creator" dialog

2.1.2. "Tailored Book Editor" main interface features

After the book is automatically created and the images are assigned a new tab with the name of the book is added to the tabbed pane. A mockup of the "Tailored Book Editor" interface shown in the Figure 16 is designed for revising a book. The main screen of the "Tailored Book Editor" is composed of four parts: Main Panel, Control Panel, Zoom Panel and Progress Panel. On the Main Panel the tabbed pane is placed. The Control Panel is grouped in two sub panels: Tags and Controls panels. The book editor functionality is encapsulated within the Editor class. The description of the Editor class will be given below in the "Tailored Book Editor" back-end section.

The progress panel of the "Tailored Book Editor" displays the progress of tailored book creation. A tabbed pane allows placing more than one book and switch between them. On the tabbed pane the tailored book figures or items are displayed (white squares with image and text). I implemented the functionality to edit tailored books by joining or splitting these figures. So the tabbed pane allows selecting one or more figures. When one tailored book figure is selected the Tags sub panel of the Control Panel displays the tags associated with the figure. A button «Save» of this sub panel allows modifying the tags assigned during automatic tailored book creation and it provides the great flexibility for tailored book revision. When two or more figures are selected the «Join» button of the Controls Panel becomes active and permits combining selected items into one item. When the figure content is composed of two or more words and this figure is selected, the «Split» button becomes active and permits splitting the item. This gives the freedom for customizing a tailored book. In other words, it is up to the user to decide if he wants, for example, prepositions and articles to be inside the figure with the corresponding noun or that every part of speech has its own figure.

A zoom panel that is placed bellow of the Tabbed Panel allows zooming in/out tailored book figures, what in turn facilitates the book revision.

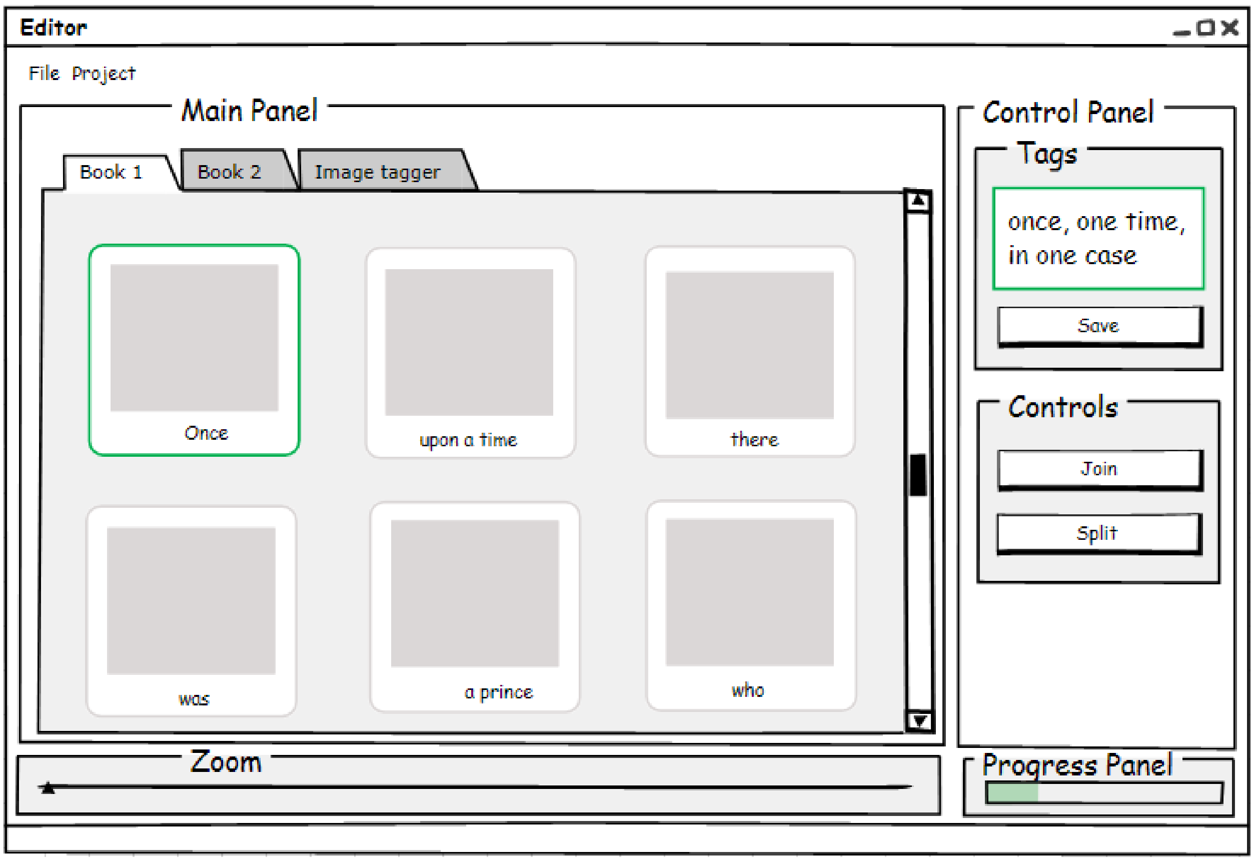


Figure 16 Mockup of the “Tailored Book Editor” interface

2.1.3. “Image Tagger” features

An Image Tagger is the module of the “Tailored Book Editor” that is used for preparing images, which the tailored book creator uses for the image assignment. To start the image tagging process the user should go to the main menu: File → New Image Tagger. On the Figure 17 the “Image Tagger” interface is shown. After clicking on menu item the dialog will appear. This dialog permits the user selecting the appropriate images. The user can open one or more images with .jpg extension. When the images are chosen a new tab is added to the tabbed pane. The “Tailored Book Editor” uses the same tabbed pane for displaying tailored books and image taggers together. The common tabbed pane allows placing and using more than one Image Tagger.

The Image Tagger functionality is encapsulated within the Editor class. The description of this class will be given below in the “Tailored Book Editor” back-end section.

The most important point implemented by me for the “Image Tagger” is the synchronization between images and the book figures. The synchronization is done in both directions. The following example should clarify this concept. For example, if a tag of an image is changed by the “Image Tagger”, the image will be automatically reassigned to the appropriate book figure at the moment when the user switches to the tailored book tab (direction: image → tailored book). And on the contrary, if the user changes a tag of a figure, automatically assigned during the tailored book creation, the appropriate image that has the same tag will be assigned to this figure (direction: tailored book→ image).

When the image is selected the Tags sub panel of the Image Tagger displays tags, assigned to the image, or an empty text view if the image has not been tagged yet. A button «Save» of the Tags sub panel allows writing tags to the image metadata. After clicking the «Save» button the assigned tags will appear in the figure under the image. The inserted tags can be also seen in the default OS tool that allows displaying properties of the image. The Image Tagger is also compatible with images tagged using the default OS tool and is able to recognize the tags. That means that the images can be tagged using external default OS tool and that the “Tailored Book Editor” is able to use the images tagged by the external image taggers. The Image tagger developed by me has a reference to the WordNet. In the Tag panel, when the comma is typed after the written tag, a list view with synonyms appears if there is an association with the tag (word form) in WordNet database. The Tags panel as well displays the parts of speech available for the inserted tag. By clicking on one of the parts of speech radio buttons the list view gets updated with appropriate content. By clicking on the one of the suggested tags, it will be inserted into the Tags text view.

A Zoom Panel allows zooming in/out image figures similar to zooming book figures.

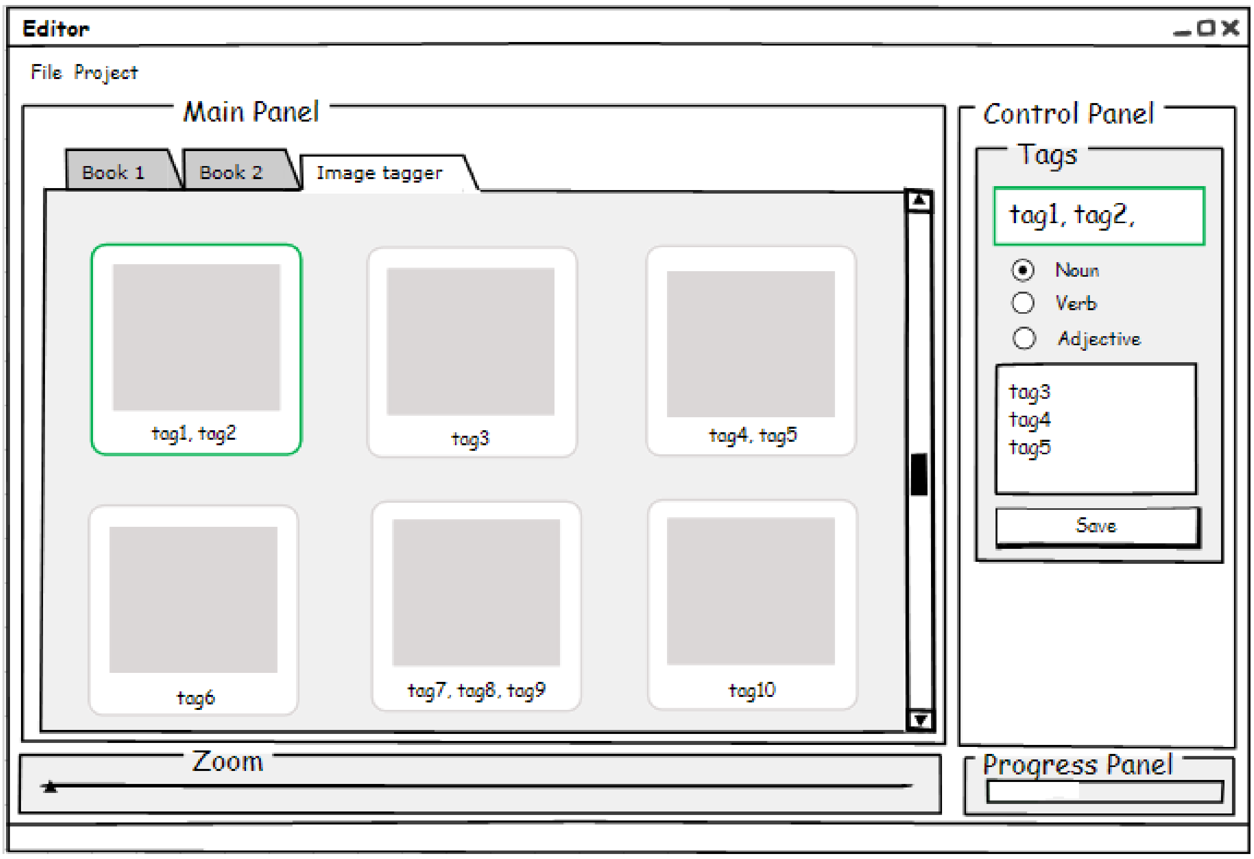


Figure 17 “Image Tagger” interface

2.1.4. Selection of Java GUI

When developing a Java program it is important to select the appropriate Java Graphical User Interface (GUI) components. There are two basic sets of components that can be used to create user interface. These two groups of components are called the Abstract Window Toolkit (AWT) and Swing. Both of these groups of components are part of the Java Foundation Classes (JFC). In general, AWT components are appropriate for simple applet development or development that targets a specific platform (i.e. the Java program will run on only one platform). For most any other Java GUI development Swing components should be used [40]. In this project Swing components are chosen, since the desktop application should run on different operating systems.

2.3. “Tailored Book Editor” back-end

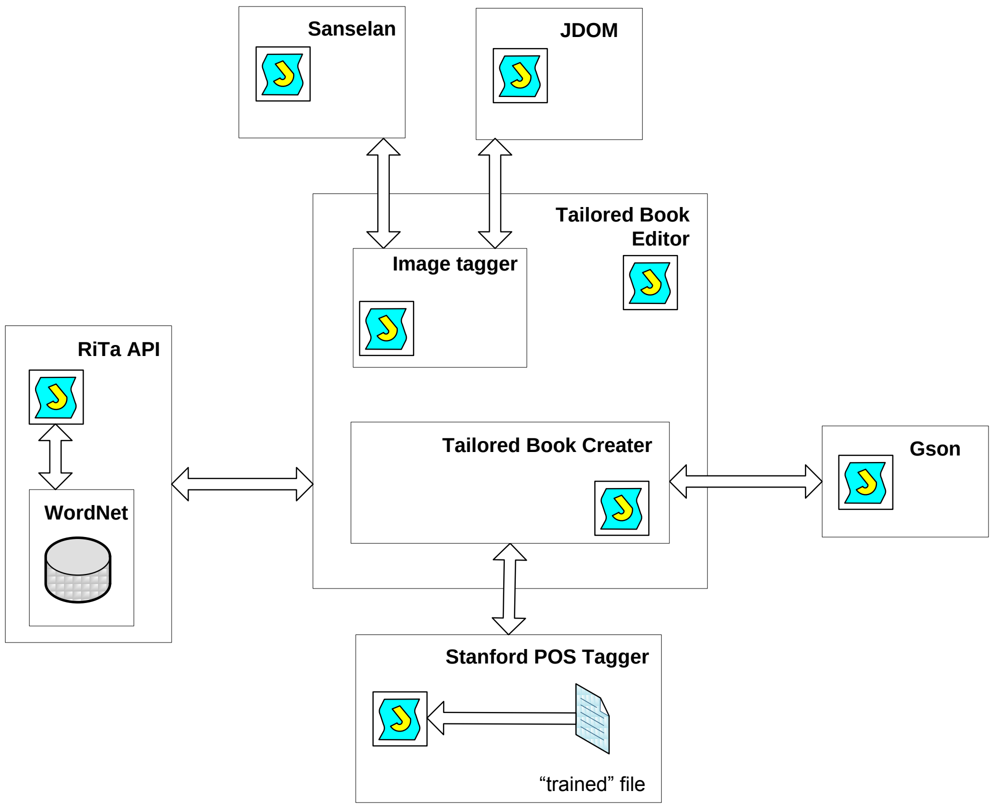


Figure 18 The “Tailored Book Editor” back-end design

The “Tailored Book Editor” aggregates several third-party Java libraries. In the Figure 18 the back-end design is shown. As you can see that “Tailored Book Editor” implemented by me contains two antecedent modules “Image tagger” and “Tailored Book Creator”, which were also implemented by me in this project. Each of these modules utilizes the libraries implemented by third party developers.

The “Image Tagger” uses Sanselan and JDOM Java libraries. The Sanselan is an open source library that is used in this project for reading (writing) an image metadata from (to) the JPEG image. This metadata contains the tags, which user assigns to the image using “Image Tagger”. The metadata is encoded in XML like format that “Image tagger” module should be able to parse and modify. Here comes the JDOM open source library, which allows accessing, modifying and outputting XML data.

The “Tailored Book Creator” utilizes Gson and Stanford POS Tagger Java libraries. Gson is used in this project for serializing the object of the class Book and transmitting it between a server and the “Tailored Book Reader” mobile application developed by me. Also JSON format is used for saving serialized objects. All the exchange file formats defined by me utilize this library. All file formats used in the project are described in detail below in the section Files Formats.

Both modules utilize the WordNet lexical database of the English language where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets). In order to access the database the RiTa interface is used. This interface allows retrieving all the synonyms from the database associated with the word. For finding all synonyms for the given word the WordNet requires two arguments: the word itself and its part of speech. I have already mentioned in the section “Tailored Book Editor” front-end that the “Image tagger” is capable of suggesting the synonyms for the typed word but it requires the user to select the part of speech using radio buttons. So the “Image tagger” finds all synsets for all parts of speech and displays them to the user and it is up to user to choose what part of speech he wants. On the contrary, the “Tailored Book Creator” is able to find all synonyms for each word of the text itself without interference of the user, because it utilizes the Stanford POS Tagger library. The Stanford POS Tagger is a Java library that for each word of the text finds its part of speech (noun, verb, adjective, etc.).

The detailed descriptions of the most important libraries are provided below.

2.3.1. WordNet

WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions.

First, WordNet interlinks not just word forms - strings of letters - but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated.

Second, WordNet labels the semantic relations among words, whereas the grouping of words in a thesaurus does not follow any explicit pattern other than meaning similarity [35].

Structure

The main relation among words in WordNet is synonymy, as between the words shut and close or car and automobile. Synonyms - words that denote the same concept and are interchangeable in many contexts - are grouped into unordered sets (synsets). Each of WordNet's 117 000 synsets is linked to other synsets by means of a small number of "conceptual relations." Additionally, a synset contains a brief definition ("gloss") and, in most cases, one or more short sentences illustrating the use of the synset members. Word forms with several distinct meanings are represented in as many distinct synsets. Thus, each form-meaning pair in WordNet is unique [35].

Relations

The most frequently encoded relation among synsets is the super-subordinate relation (also called hyperonymy, hyponymy or ISA relation). It links more general synsets like {furniture, piece_of_furniture} to increasingly specific ones like {bed} and {bunkbed}. Thus, WordNet states that the category furniture includes bed, which in turn includes bunkbed; conversely, concepts like bed and bunkbed make up the category furniture. All noun hierarchies ultimately go up the root node {entity}. Hyponymy relation is transitive: if an armchair is a kind of chair, and if a chair is a kind of furniture, then an armchair is a kind of furniture. WordNet distinguishes among Types (common nouns) and Instances (specific persons, countries and geographic entities). Thus, armchair is a type of chair, Barack Obama is an instance of a president. Instances are always leaf (terminal) nodes in their hierarchies.

Meronymy, the part-whole relation holds between synsets like {chair} and {back, backrest}, {seat} and {leg}. Parts are inherited from their superordinates: if a chair has legs, then an armchair has legs as well. Parts are not inherited "upward" as they may be characteristic only of specific kinds of things rather than the class as a whole: chairs and kinds of chairs have legs, but not all kinds of furniture have legs.

Verb synsets are arranged into hierarchies as well; verbs towards the bottom of the trees (troponyms) express increasingly specific manners characterizing an event, as in {communicate}-{talk}-{whisper}. The specific manner expressed depends on the semantic field; volume (as in the example above) is just one dimension along which verbs can be elaborated. Others are speed (move-jog-run) or intensity of emotion (like-love-idolize). Verbs describing events that necessarily and unidirectionally entail one another are linked: {buy}-{pay}, {succeed}-{try}, {show}-{see}, etc.

Adjectives are organized in terms of antonymy. Pairs of “direct” antonyms like wet-dry and young-old reflect the strong semantic contract of their members. Each of these polar adjectives in turn is linked to a number of “semantically similar” ones: dry is linked to parched, arid, dessicated and bone-dry and wet to soggy, waterlogged, etc. Semantically similar adjectives are “indirect antonyms” of the contral member of the opposite pole. Relational adjectives (“pertainyms”) point to the nouns they are derived from (criminal-crime). There are only few adverbs in WordNet (hardly, mostly, really, etc.) as the majority of English adverbs are straightforwardly derived from adjectives via morphological affixation (surprisingly, strangely, etc.) [35].

Cross-POS relations

The majority of the WordNet’s relations connect words from the same part of speech (POS). Thus, WordNet really consists of four sub-nets, one each for nouns, verbs, adjectives and adverbs, with few cross-POS pointers. Cross-POS relations include the “morphosemantic” links that hold among semantically similar words sharing a stem with the same meaning: observe (verb), observant (adjective) observation, observatory (nouns). In many of the noun-verb pairs the semantic role of the noun with respect to the verb has been specified: {sleeper, sleeping_car} is the LOCATION for {sleep} and {painter} is the AGENT of {paint}, while {painting, picture} is its RESULT [35].

Applications

WordNet has been used for a number of different purposes in information systems, including word sense disambiguation, information retrieval, automatic text classification, automatic text summarization, and even automatic crossword puzzle generation.

A project at Brown University started by Jeff Stibel, James A. Anderson, Steve Reiss and others called Applied Cognition Lab created a disambiguator using WordNet in 1998 [O. Malik. How google is that?. Forbes, 10.04.1999]. The project later morphed into a company called Simpli, which is now owned by ValueClick. George Miller joined the Company as a member of the Advisory Board. Simpli built an Internet search engine that utilized a knowledge base principally based on WordNet to disambiguate and expand keywords and synsets to help retrieve information online. WordNet was expanded upon to add increased dimensionality, such as intentionality, people and colloquial terminology more relevant to Internet search (i.e., blogging, ecommerce). Neural network algorithms searched the expanded WordNet for related terms to disambiguate search keywords (Java, in the sense of coffee) and expand the search synset (Coffee, Drink, Joe) to improve search engine results. [P. J. Hane. Beyond Keyword Searching - Oingo and Simpli.com Introduce Meaning-Based Searching. InfoToday, Posted On December 20, 1999.] Before the company was acquired, it performed searches across search engines such as Google, Yahoo!, Ask.com and others.

Another prominent example of the use of WordNet is to determine the similarity between words. Various algorithms have been proposed, and these include considering the distance between the conceptual categories of words, as well as considering the hierarchical structure of the WordNet ontology [36].

2.3.2. Java APIs for WordNet

JAWS

The Java API for WordNet Searching (JAWS) is a high-performance interface that provides Java applications with the ability to retrieve data from the WordNet database. It is a simple and fast API written entirely in Java and is compatible with both the 2.1 and 3.0 versions of the WordNet database files and can be used with Java 1.4 and later. The JAWS was created and is maintained by Brett Spell, who is currently an adjunct member of the faculty in the Computer Science and Engineering (CSE) department at Southern Methodist University.

JWNL

Java WordNet Library (JWNL) is an API for accessing WordNet in multiple formats, as well as relationship discovery and morphological processing. It's compatible with WordNet versions 2.0 to 3.0, and is a full Java implementation that allows developers to use more easily Java for building NLP applications. JWNL is released under the BSD License, and is used by many commercial applications [35].

RiTa WordNet API

RiTa WordNet is a friendly WordNet interface that allows Java applications to retrieve data from the WordNet database. This Java API written entirely in Java comes with WordNet dictionary itself so that there is no need to install WordNet, thereby RiTa WordNet API requires only adding two JAR files to Eclipse build path: ritaWN.jar and supportWN.jar. The advantage of using RiTa WordNet API over other APIs such as JAWS or JWNL (Java WordNet Library) is that the final program can be launched on any device without WordNet Database being installed, so that the process of final product deployment is considerably simplified. Since this API perfectly fits the project it was chosen. This software is covered under the GPL.

2.3.3. Part-of-speech tagging

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech (noun, verb, adjective, etc.) to each word. The tagger is licensed under the GNU General Public License (v2 or later)

Part-of-speech tagging is harder than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times, and because some parts of speech are complex or unspoken. This is not rare in natural languages (as opposed to many artificial languages), a large percentage of word-forms are ambiguous. For example, even "dogs", which is usually thought of as just a plural noun, can also be a verb:

The sailor dogs the barmaid.

Performing grammatical tagging will indicate that "dogs" is a verb, and not the more common plural noun, since one of the words must be the main verb, and the noun reading is less likely following "sailor" (sailor !→ dogs). Semantic analysis can then extrapolate that "sailor" and "barmaid" implicate "dogs" as 1) in the nautical context (sailor→<verb>←barmaid) and 2) an action applied to the object "barmaid" ([subject] dogs→barmaid). In this context, "dogs" is a nautical term meaning "fastens (a watertight barmaid) securely; applies a dog to".

"Dogged", on the other hand, can be either an adjective or a past-tense verb. Just which parts of speech a word can represent varies greatly.

Trained linguists can identify the grammatical parts of speech to various fine degrees depending on the tagging system. Schools commonly teach that there are 9 parts of speech in English: noun, verb, article, adjective, preposition, pronoun, adverb, conjunction, and interjection. However, there are clearly many more categories and sub-categories. For nouns, plural, possessive, and singular forms can be distinguished. In many languages words are also marked for their "case" (role as subject, object, etc.), grammatical gender, and so on; while verbs are marked for tense, aspect, and other things.

In part-of-speech tagging by computer, it is typical to distinguish from 50 to 150 separate parts of speech for English, for example, NN for singular common nouns, NNS for plural common nouns, NP for singular proper nouns (see the POS tags used in the Brown Corpus). Work on stochastic methods for tagging Koine Greek (DeRose 1990) has used over 1,000 parts of speech, and found that about as many words were ambiguous there as in English [22].

The Stanford POS Tagger lets "tag" the words in the string. That is, for each word, the "tagger" gets whether it's a noun, a verb, etc. and then assigns the result to the word. For example for the following string: "Once upon a time" the result will be: Once/RB upon/IN a/DT time/NN. There is a tag assigned to each word is added after slash. Below the table with description of tagsets used for Stanford POS Tagger is shown.

Penn Treebank Tagset

Tag	Description
CC	Coordinating conjunction Example: and,but,or...
CD	Cardinal Number
DT	Determiner
EX	Existential *there*
FW	Foreign Word
IN	Preposision or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List Item Marker
MD	Modal Example: can, could, might, may...
NN	Noun, singular or mass
NNP	Proper Noun, singular
NNPS	Proper Noun, plural
NNS	Noun, plural
PDT	Predeterminer Example: all, both ... when they precede an article
POS	Possessive Ending Example: Nouns ending in 's
PRP	Personal Pronoun Example: I, me, you, he...
PRP$	Possessive Pronoun Example: my, your, mine, yours...
RB	Adverb Most words that end in -ly as well as degree words like quite, too and very
RBR	Adverb, comparative Adverbs with the comparative ending -er, with a strictly comparative meaning.
RBS	Adverb, superlative
RP	Particle
SYM	Symbol Should be used for mathematical, scientific or technical symbols
TO	*to*
UH	Interjection Example: uh, well, yes, my...
VB	Verb, base form subsumes imperatives, infinitives and subjunctives
VBD	Verb, past tense includes the conditional form of the verb to be

VBG	Verb, gerund or persent participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner Example: which, and *that* when it is used as a relative pronoun
WP	Wh-pronoun Example: what, who, whom...
WP$	Possessive wh-pronoun
WRB	Wh-adverb Example: how, where why

Table 1 Penn Treebank Tagset

Punctuation Tags	Description
$	Dollar Example: $ -$ --$ A$ C$ HK$ M$ NZ$ S$ U.S.$ US$
''	Closing quotation mark Example: ' ''
(	Opening parenthesis Example: ([{
)	Closing parenthesis Example:)] }
,	Comma
.	Sentence terminator Example: . ! ?
:	Colon or ellipsis Example: : ; ...
``	Opening quotation mark Example: ` ``

Table 2 Punctuation Tagset

In order to do this, the tagger has to load a “trained” file that contains the necessary information for the tagger to tag the string. This “trained” file is called a model and has the extension “.tagger”. There are several trained models provided by the Stanford NLP group for different languages.

2.3.4. JDOM

JDOM is a Java library that provides a complete, Java-based solution for accessing, manipulating, and outputting XML data from Java code.

JDOM is available under an Apache-style open source license, with the acknowledgment clause removed. This license enables developers to use JDOM in creating new products without requiring them to release their own products as open source.

2.3.5. Sanselan

Sanselan is a pure Java library, which reads and writes a variety of image formats, including fast parsing of image info (size, color space, etc.) and metadata. This project is Open Source and available under the ASF (Apache) License.

2.3.6. Gson

Gson (also known as Google Gson) is a Java library that is used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson is an open-source project and available under Apache License 2.0 [41].

Features:

- ✓ Gson can handle collections, generic types and nested classes (including inner classes, this cannot be done by default though)
- ✓ When deserializing, Gson is navigating the type tree of the object, which is being deserialized. This results in ignoring extra fields present in the JSON input.
- ✓ The user is allowed to write a custom serializer and/or deserializer, so that he can control the whole process and even (de)serialize instances of classes which has not accessible source code.
- ✓ The user has the possibility to write an InstanceCreator which allows him to deserialize instances of classes without defined no-args constructor.

Gson is highly customizable, you can specify:

- ✓ Compact/pretty printing (whether you want compact or readable output)
- ✓ How to handle null object fields - by default they are not present in the output
- ✓ Rules of what fields are intended to be excluded from (de)serialization
- ✓ How to convert Java field names

2.3.7. RDF

The Resource Description Framework (RDF) is a flexible technology, capable of addressing a wide variety of problems. RDF's purpose is to provide a means of recording data in a machine-understandable format, allowing for more efficient and sophisticated data interchange, searching, cataloging, navigation, classification, and so on. It forms the cornerstone of the W3C effort to create the Semantic Web, but its use isn't restricted to this specific effort.

RDF can be integrated into the architecture of a tool from the ground up, or RDF and RDF/XML can be incorporated into their existing applications. Adobe, a major player in the publications and graphics business, takes the second way. Its RDF/XML strategy is known as Extensible Metadata Platform. According to the Adobe XMP web site, other major players have agreed to support the XMP framework, including companies such as Microsoft [42].

2.3.8. Metadata and XMP

Metadata is any data that helps to describe the content or characteristics of a file. Some basic metadata can be seen and edited through the File Info or Document Properties box found in many software applications and some operating systems.

Adobe's Extensible Metadata Platform (XMP) is a labeling technology that allows you to embed data about a file, known as metadata, into the file itself. With XMP, Adobe has taken the "heavy lifting" out of metadata integration, offering content creators an easy way to embed meaningful information about their projects and providing industry partners with standards-based building blocks to develop optimized workflow solutions.

With an XMP-enabled application, information about a project can be captured during the content-creation process and embedded within the file and into a content-management system. Meaningful descriptions and titles, searchable keywords, and up-to-date author and copyright information can be captured in a format that is easily understood by you as well as by software applications, hardware devices, and even file formats.

Best of all, as other workgroup members modify files and assets, XMP-encoded metadata can be edited and updated in real time during the normal course of the workflow [43].

By providing a standard, W3C-compliant way of tagging files with metadata across products from Adobe and other vendors, XMP is a powerful solution enabler. As an open-source technology, it is freely available to developers, which means that the user community benefits from the innovations contributed by developers worldwide. Furthermore, XMP is extensible, meaning that it can accommodate existing metadata schemas; therefore, systems don't need to be rebuilt from scratch. A growing number of third-party applications already support XMP.

XMP was first published around September 2001. It stores the RDF in the APP1 chunk of JPEG and it adds a magic string ("W5M0MpCehiHzreSzNTczkc9d") at the start of the RDF to help distinguish XMP from other things that might be present in the JPEG file. XMP embedding is also defined for other formats than JPEG, such as TIFF and PDF. Adobe recommends the Dublin Core schema and offers additional schemas for recording version history, image manipulations, etc [37].

An important consideration with these embedded techniques is that there is no adverse impact on the file, nothing that impacts on the visibility of a JPEG or a PNG graphic or prevents an HTML file from loading into a browser [44].

2.3.9. Dublin Core Schema

Dublin Core [37] is an initiative to create a digital "library card catalog" for the Web. Dublin Core is made up of metadata elements (data that describes data) that offer expanded cataloging information and improved document indexing for search engine programs.

The two most common forms of Dublin Core are *Simple Dublin Core* and *Qualified Dublin Core*. Simple Dublin Core expresses elements as attribute-value pairs using just the base metadata elements from the Dublin Core Metadata Element Set. Qualified Dublin Core increases the specificity of metadata by adding information about encoding schemes, enumerated lists of values, or other processing clues. While enabling searches to be more specific, qualifiers are also more complex and can pose challenges to interoperability. In other words, Simple Dublin Core gives basic information. However, if more information is required, The Qualified Dublin Core schema can be used.

Below there is an interpretation of the Simple Dublin Core properties, applied to a photo.

Title
a short description of the photo. Example: *help*
Subject
a set of keywords to describe the photo. Example: *help, assist, aid*
Description
a longer description of the photo. Example: *help*
creator
the photographer, as a URL that can be further described with other schemas.
Publisher
the person or institution making the photo available, often the same as the creator.
Contributor
a person who contributed in some way, e.g., the person who digitized the photo; may be a URL or a name.
Date
the date and time the photo was taken, conforming to ISO format. The year is required, everything else can be omitted. The default time zone is UTC.
Type

	always "image"
Format	how the resource is presented. Always "image/jpeg"
identifier	a number for the photo that is meaningful to the publisher. This is not the URL of the photo and it does not have to be globally unique.
Source	where the content originally derived from
Language	in what language the content is written
Relation	identifies a series: the event or topic for a series of photographs. Can be a URL or a string.
Coverage	the location shown on the photo. Example: *Como*
Rights	copyright statement, or the URL for one.

Table 3 Interpretation of the Simple Dublin Core properties, applied to photo material

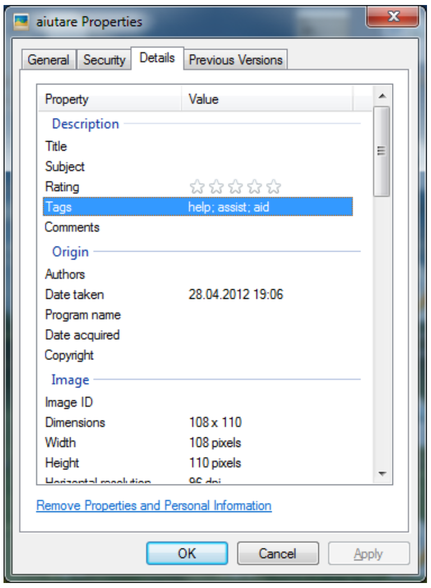
In this project only the subject property defined by the Dublin Core Schema is used. The other properties can be added to the image but are ignored by the "Image tagger" module.

Below a metadata written in the .jpg image is shown. "Image tagger" is capable to write key words in the image.

```xml
<?xpacket begin='?' id='W5M0MpCehiHzreSzNTczkc9d'?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description xmlns:xmp="http://ns.adobe.com/xap/1.0/" rdf:about="uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b">
      <xmp:CreateDate>2012-04-28T19:06:27</xmp:CreateDate>
    </rdf:Description>
    <rdf:Description xmlns:dc="http://purl.org/dc/elements/1.1/" rdf:about="uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b" />
    <rdf:Description xmlns:dc="http://purl.org/dc/elements/1.1/" rdf:about="uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b">
      <dc:title>
        <rdf:Alt>
          <rdf:li xml:lang="x-default" />
        </rdf:Alt>
      </dc:title>
      <dc:description>
        <rdf:Alt>
          <rdf:li xml:lang="x-default" />
        </rdf:Alt>
      </dc:description>
      <dc:subject>
        <rdf:Bag>
          <rdf:li>help</rdf:li>
          <rdf:li>assist</rdf:li>
          <rdf:li>aid</rdf:li>
        </rdf:Bag>
      </dc:subject>
    </rdf:Description>
    <rdf:Description xmlns:MicrosoftPhoto="http://ns.microsoft.com/photo/1.0/" rdf:about="uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b" />
    <rdf:Description xmlns:MicrosoftPhoto="http://ns.microsoft.com/photo/1.0/" rdf:about="uuid:faf5bdd5-ba3d-11da-ad31-d33d75182f1b">
      <MicrosoftPhoto:LastKeywordXMP>
        <rdf:Bag>
          <rdf:li>help</rdf:li>
          <rdf:li>assist</rdf:li>
          <rdf:li>aid</rdf:li>
        </rdf:Bag>
      </MicrosoftPhoto:LastKeywordXMP>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
<?xpacket end='w'?>
```

Figure 19 Metadata of .jpg file

The key words or image tags can be also seen from the properties of the image (See Figure 20).



Property	Value
Description	
Title	
Subject	
Rating	☆☆☆☆☆
Tags	help; assist; aid
Comments	
Origin	
Authors	
Date taken	28.04.2012 19:06
Program name	
Date acquired	
Copyright	
Image	
Image ID	
Dimensions	108 x 110
Width	108 pixels
Height	110 pixels

Figure 20 Image details in OS Windows

2.4. Text processing

In this section the process of creation items from the text is described. Every framed item or figure is composed of the image and the word it represents written below.

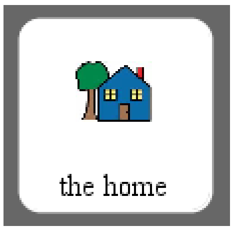


Figure 21 Example of framed item or figure

1) Let’s take the text of the famous Brothers Grimm’s fairy tale Little Red Riding Hood and consider it as the initial text. All excess spaces are removed from the text.

Once upon a time in the middle of a thick forest stood a small cottage, the home of a pretty little girl…

2) After applying Stanford POS tagging the parts-of-speech tags were added to every text item:

Once/RB upon/IN a/DT time/NN in/IN the/DT middle/NN of/IN a/DT thick/JJ forest/NN stood/VBD a/DT small/JJ cottage/NN ,/, the/DT home/NN of/IN a/DT pretty/RB little/JJ girl/NN

3) For facilitating extraction of the words themselves and parts-of-speech tags the text is split in the array list:

Once/RB
upon/IN
a/DT
time/NN

in/IN
the/DT
middle/NN
of/IN
a/DT
thick/JJ
forest/NN
stood/VBD
a/DT
small/JJ
cottage/NN
,/,
the/DT
home/NN
of/IN
a/DT
pretty/RB
little/JJ
girl/NN

4. Then each item of the array list is split in two parts: word and tag.

Word	Tag
Once	RB
upon	IN
a	DT
time	NN
in	IN
the	DT
middle	NN
of	IN
a	DT
thick	JJ
forest	NN
stood	VBD
a	DT
small	JJ
cottage	NN
,	,
the	DT
home	NN
of	IN
a	DT
pretty	RB
little	JJ
girl	NN

Table 4 word and tag table representing array list

5. Since Stanford POS tagger output tagsets are more detailed than input tagsets of WordNet we need to convert parts-of-speech tagsets of Stanford POS tagger into WordNet parts-of-speech.

Valid WordNet parts-of-speech include (noun="n", verb="v", adj="a", and adverb="r"). These can be specified either as literals or using the String constants:

RiWordnet.NOUN
RiWordnet.VERB
RiWordnet.ADJ
RiWordnet.ADV

RiWordnet.NOUN	RiWordnet.VERB	RiWordnet.ADJ	RiWordnet.ADV
NN, NNP, NNPS, NNS, PDT	VB, VBD, VBG, VBN, VBP, VBZ, MD	JJ, JJR, JJS	RB, RBR, RBS

Table 5 Conversion of tags of POS tagger to WordNet tags

All the rest parts-of-speech tags of Stanford POS tagger which are not included in the table (e.g. punctuation, preposition tags) won't be sent to WordNet.

6. Each word with recognized in the previous step part of speech is then sent to WordNet in order to get all Synsets of the word. When the Synsets are found the object of class Item is going to be constructed. The class Item has the constructor with the three following fields:

Item item = new Item (synonyms, word, tag);

Where the first argument: synonyms are the Synsets found in the WordNet, the second argument: word is the word itself and the third one: tag is the parts-of-speech tag of a Stanford POS tagger. If there are no found Synsets for the particular word, the word itself takes place of the first parameter of the constructor.

For those words with parts-of-speech tags of Stanford POS tagger (PRP, PRP$, DT, IN, CD, TO and so on) which were not matched to Wordnet parts-of-speech (RiWordnet.NOUN, RiWordnet.VERB, RiWordnet.ADJ, RiWordnet.ADV) the procedure of construction of items is similar to the words which are not found in the Wordnet. In other words the word itself will take the first parameter of the item constructor.

During the item construction the instance variable position is calculated based on the third argument of the Item constructor. The type of position is enum, whose fields consist of a fixed set of constants: PREFIX, INFIX and POSTFIX.

The calculation of the position is based on the following logic: all the prepositions, articles, and personal pronouns usually take place in front of the word therefore these items will get position set as PREFIX. All punctuation like comma, semicolon, colon, point, question mark and so on are usually placed after the word and so the position will be set as POSTFIX. Noun, verb, adverb and adjective will get the INFIX position.

The assignment of parts of speech to the position (PREFIX, INFIX, POSTFIX) is predefined and is considered to be the default option in this project. In the future release of the "Tailored Book Editor" this will not be the case and users will get the right to modify the default option and decide if they want to have prepositions, articles, pronouns and etc. to be together in one item with other parts of speech or to be separated in two items. In the Table 6 the default assignment of parts of speech to the position is shown.

Word	Tag	Position
Once	RB	INFIX
upon	IN	PREFIX
a	DT	PREFIX
time	NN	INFIX

in	IN	PREFIX
the	DT	PREFIX
middle	NN	INFIX
of	IN	PREFIX
a	DT	PREFIX
thick	JJ	INFIX
forest	NN	INFIX
stood	VBD	INFIX
a	DT	PREFIX
small	JJ	INFIX
cottage	NN	INFIX
,	,	POSTFIX
the	DT	PREFIX
home	NN	INFIX
of	IN	PREFIX
a	DT	PREFIX
pretty	RB	INFIX
little	JJ	INFIX
girl	NN	INFIX

Table 6 Position assignment

7. All the created in the previous step items are collected to an array list. The array list is then processed based on the following rule. If the item has the position PREFIX its word is added to the next item and the item with position PREFIX is removed from the list. If the item has the position POSTFIX its word is added to the previous item and the POSTFIX item is then removed from the list. As the result we get the reduced list of items with the position INFIX.

Word	Tag	Position
Once	RB	INFIX
upon a time	NN	INFIX
in the middle	NN	INFIX
of a thick	JJ	INFIX
Forest	NN	INFIX
Stood	VBD	INFIX
a small	JJ	INFIX
cottage ,	NN	INFIX
the home	NN	INFIX
of a pretty	RB	INFIX
Little	JJ	INFIX
Girl	NN	INFIX

Table 7 Processing of the positions

8. Then the images from the specified source directory are loaded. The image load is done recursively; therefore it is enough to specify the top folder. The array list of objects of class MyImage is created. The constructor of MyImage class has the following arguments:

```
MyImage img=new MyImage(name, tags, filePath);
```

Where the argument: name is the name of the image, the argument tags are the tags manually assigned to the image using Image Tagger and the parameter: filePath is the path to the image file.

Afterwards for each item the image is assigned to the item if there is the match among the tags of the item and the image. These items are then visualized in figures (See Figure 22 Result of text processing).

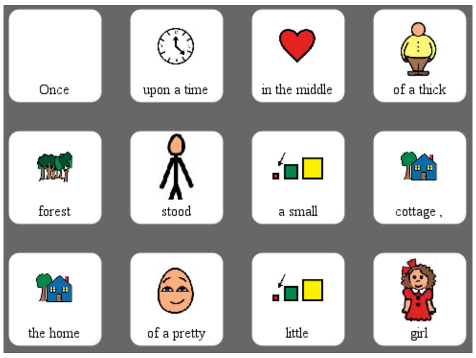


Figure 22 Result of the text processing

2.5. Files Formats

Image format:

The “Tailored Book Editor” supports only JPEG images, this is due to the fact that JPEG format is widely used and permits to read/write image Metadata.

“Tailored Book Editor” Settings format

The “Tailored Book Editor” saves the settings information in the file called defaults.khl. The content of this file is encoded in a JSON format. The JSON format is used for serializing an object of a class Defaults. The structure of the defaults.khl file is shown in the Figure 23. The Defaults class contains three properties: current directory, current image directory path and current tagger path. The “Tailored Book Editor” creates this file or updates it every time when the user closes the application. The currentDirectory property of the class Defaults contains the latest directory opened by the user during working with the “Tailored Book Editor”. The currentImageDirectoryPath property contains the path selected by the user for an image source directory; for the directory that application will use for the image assignment. So the system requires selecting the image source directory only once and then stores it and uses as default option. The directory then can be change manually by going to the main menu Project → Image source directory. Also it is possible to change the directory directly in the defaults.khl file but it is not recommended. The currentTaggerPath property contains the path to the .tagger file used by the Stanford POS tagger. Similar to the image source directory, the system requires to set this file only ones and then does not bother the user.

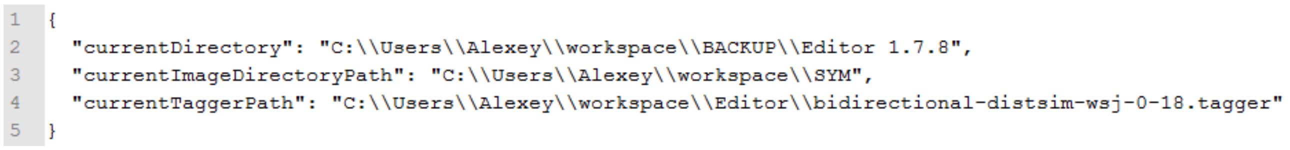
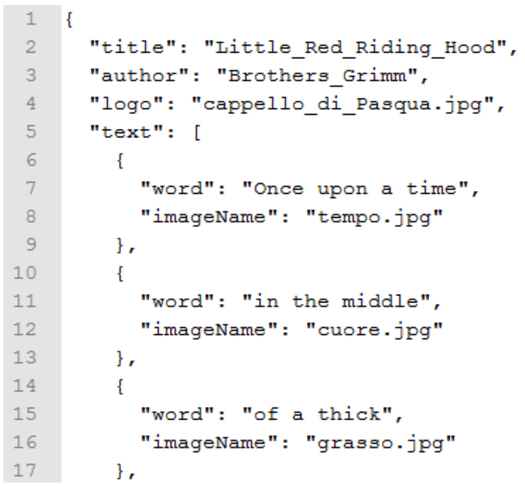

```
{
  "currentDirectory": "C:\\Users\\Alexey\\workspace\\BACKUP\\Editor 1.7.8",
  "currentImageDirectoryPath": "C:\\Users\\Alexey\\workspace\\SYM",
  "currentTaggerPath": "C:\\Users\\Alexey\\workspace\\Editor\\bidirectional-distsim-wsj-0-18.tagger"
}
```

Figure 23 Structure of default.khl file

Exchange Book Format

The “Tailored Book Editor” publishes the book in the format .khl and folder with JPEG images. The content of the book file .khl is encoded in a JSON format. The JSON format is used for serializing the object of the class Book and transmitting it between a server and the Book Reader application. The example of the published Little Red Riding Hood book is shown below.

Files published by the "Tailored Book Editor" are the following: the tailored book itself called Little_Red_Riding_Hood.khl and a folder called Little_Red_Riding_Hood with assigned images. The structure of Little_Red_Riding_Hood.khl file is shown in the Figure 24 below:



```
{
  "title": "Little_Red_Riding_Hood",
  "author": "Brothers_Grimm",
  "logo": "cappello_di_Pasqua.jpg",
  "text": [
    {
      "word": "Once upon a time",
      "imageName": "tempo.jpg"
    },
    {
      "word": "in the middle",
      "imageName": "cuore.jpg"
    },
    {
      "word": "of a thick",
      "imageName": "grasso.jpg"
    },
```

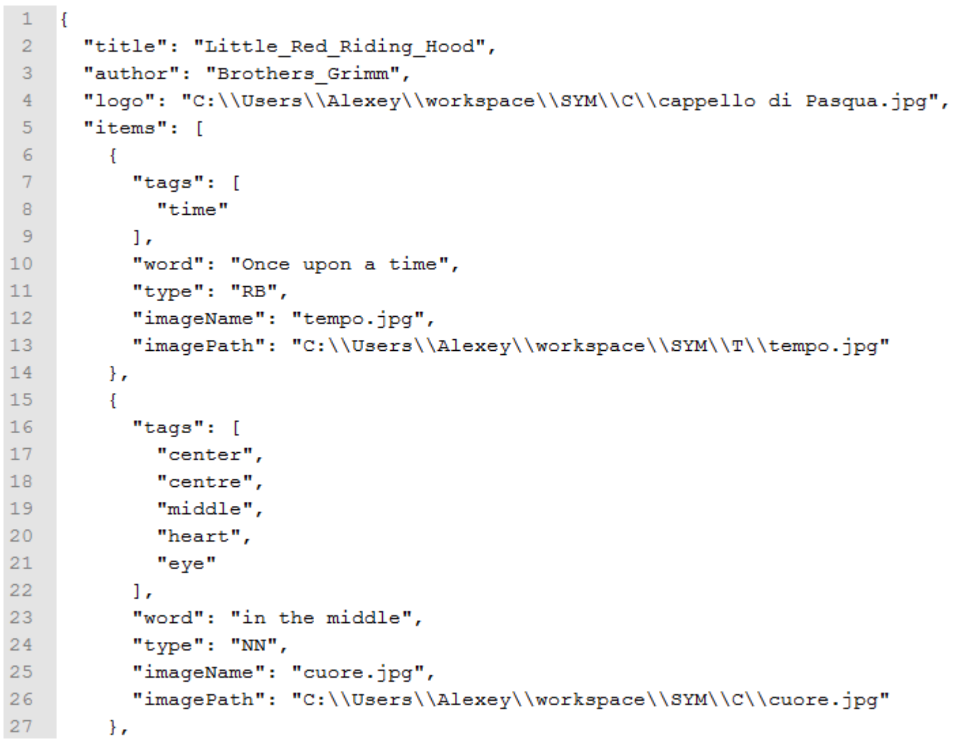
Figure 24 Structure of Little_Red_Riding_Hood.khl file

The folder Little_Red_Riding_Hood contains the images: cappello_di_Pasqua.jpg, tempo.jpg, cuore.jpg, grasso.jpg etc.

"Tailored Book Editor" Save Book Format

The Book Editor is able to open and save books in the format .bookkhl. The content of the file is encoded in the JSON format. The JSON format is used here for serializing the object of class BookSaveFormat that extends Book.

Structure of Little_Red_Riding_Hood.bookkhl file:



```
{
  "title": "Little_Red_Riding_Hood",
  "author": "Brothers_Grimm",
  "logo": "C:\\Users\\Alexey\\workspace\\SYM\\C\\cappello di Pasqua.jpg",
  "items": [
    {
      "tags": [
        "time"
      ],
      "word": "Once upon a time",
      "type": "RB",
      "imageName": "tempo.jpg",
      "imagePath": "C:\\Users\\Alexey\\workspace\\SYM\\T\\tempo.jpg"
    },
    {
      "tags": [
        "center",
        "centre",
        "middle",
        "heart",
        "eye"
      ],
      "word": "in the middle",
      "type": "NN",
      "imageName": "cuore.jpg",
      "imagePath": "C:\\Users\\Alexey\\workspace\\SYM\\C\\cuore.jpg"
    },
```

Figure 25 Structure of Little_Red_Riding_Hood.bookkhl file

“Tailored Book Editor” Input Format

An input file loaded in the “Tailored Book Editor” must be in .txt format. After processing the text file a tailored book is created and then can be both saved for further editing in the format .bookkhl and published in .khl format.

Server Side Index File Format

A tailored book published by the “Tailored Book Editor’ can be either uploaded on the server or put directly on SD card of the mobile device. On the server side there is an index file called books.khl. It can be seen from the URL: http://tbreader.is-great.net/BOOKS/books.khl

This file book.khl is accessed by the “Tailored Book Reader” application when the user goes to the “Download Books” screen. The data of this file is encoded in JSON format. The file books.khl contains the titles and authors of all available books for download on the server and has the following structure:

```
[{
  "title": "Little_Red_Riding_Hood",
  "author": "Brothers_Grimm",
  "logo": "cappello_di_Pasqua.jpg",
  "text": []
},
{
  "title": "The_Lion_and_the_Mouse",
  "author": "Unknown",
  "logo": "leone.jpg",
  "text": []
}
]
```

Figure 26 Structure of books.khl file

The structure of this file reminds a tailored book .khl file with the exception of some differences. The first difference is that books.khl file contains an array of books in format .khl, and the second one is that the .khl files are used without book content (“text” field is an empty array).

2.6. “Tailored Book Editor” class diagram

For the “Tailored Book Editor” I designed and implemented the following classes: Editor, Tagger, DirectoryReader, Book, Item, BookItem, BookSaveFormat, MyImage, BookDetails, ImageFilter, MyFilter, TaggerFilter, TextFileFilter, BookFileFilter, PicViewRenderer, ButtonTabComponent, RadioButton.

In the Figure 27 the “Tailored Book Editor” class diagram is shown. Since the system utilizes a variety of file types both standard .txt, .jpg, .tagger and defined by me: .khl, .bookkhl there is a need for accepting a suitable file type at every step during the process of tailored books creation.

When the system asks the user to locate a file it makes use of the instance of an appropriate class: ImageFilter, MyFilter, TaggerFilter, TextFileFilter, BookFileFilter in order to eliminate a possibility to select the wrong file by hiding all the files that don’t correspond to the required type. All the properties and methods of these classes are shown in the Figure 29.

The object of the Item class represents a figure (See Figure 21 Example of a framed item or a figure) in the “Tailored Book Editor” application. The Item class contains the following properties: imageName, imagePath, tags, type, word. The imageName and imagePath stores the name and the path of the image assigned to the figure. The tags property contains an array of tags found in the WordNet database for a word stored in the word property. The type property contains the word type (tag) assigned by the Stanford POS tagger (See Table 1 Penn Treebank Tagset). The Item class contains all the information necessary for visualization of the figure and assignment of an appropriate image to it. The most important method of the Item class is called assignImage. This method takes as an argument an array of objects belonging to MyImage Class. The instance of the MyImage class represents a figure of the "Image Tagger" module. The MyImage class contains the property tags similar to the Item class. These tags are assigned by the user when he tags the image using the "Image Tagger". So previously mentioned method (assignImage) takes the array of MyImage objects and tries to find the intersection between tags of the Item class instance and tags of the MyImage class instances. Once found the method assigns the name and the path of the found image to the Item class properties (imageName, imagePath).

An instance of the Book class corresponds to a published tailored book. Just that instance will be encoded in JSON format and exported with the extension .khl for further usage in the "Tailored Book Reader" application. The main properties of this class are: title – title of the book, author – author of the book, text – content of the book. In the Figure 28 all properties and methods of the Book class are shown. The content of the book represents an array of the BookItem instances. An instance of the BookItem class corresponds to a figure of the "Tailored Book Reader". In the BookItem class two properties are defined: imageName and word. The imageName property is a name of the image associated with the figure and the word is a text (contains one or more words) inside the figure.

The BookSaveFormat class extends Book class and was introduced to enable saving a tailored book for further editing. The difference between that class and the Book class is that the content of a tailored book is stored in the array that contains the instances of Item and BookItem classes respectively. Since an instance of the Item class allows storing all the information about the figure and not only the image name and the word, an instance of the BookSaveFormat after the serialization and saving with the extension .bookkhl can be read by the "Tailored Book Editor".

The DirectoryReader class is responsible for finding all the images under the given directory. The most important method of this class is called Process. This is a recursive method that is able to find all the images in the selected directory and all its sub directories. This is greatly facilitates the image source directory setting. Once done, the "Tailored Book Editor" scans all the beneath directories for the images that will be then used for image assignment during a process of tailored book creation.

The Tagger class allows initiating the Stanford POS Tagger and performs a part of speech tagging of a given string.

The BookDetails class extends JDialog class and represents a dialog which appears when the user intends to create a new tailored book (See Figure 15 Mockup of the "Tailored Book Creator" dialog). This dialog is responsible for checking the correctness of the user input and notifying users about missing fields and wrong characters. The verification of the user input is implemented using regular expressions.

The ButtonTabComponent extends JPanel class and was developed to be used as tab component in the "Tailored Book Editor" application. This component displays an icon in every tab (the icon of the book if user works with the tailored book and the icon of the image if the user utilizes "Image Tagger" module) and a close button. Also a tab for the "Image Tagger" contains the name of the folder where the image(s) is(are) located. A tab with a tailored book contains the title of the book.

In order to display the book figures in a grid layout I used the grid component for Java Swing [38] available under Apache License 2.0. I designed the PicViewRenderer that implements the interface GridCellRenderer. The PictureViewRenderer displays a figure with the instance of class Item and MyImage in the "Tailored Book Editor" and "Image Tagger" respectively.

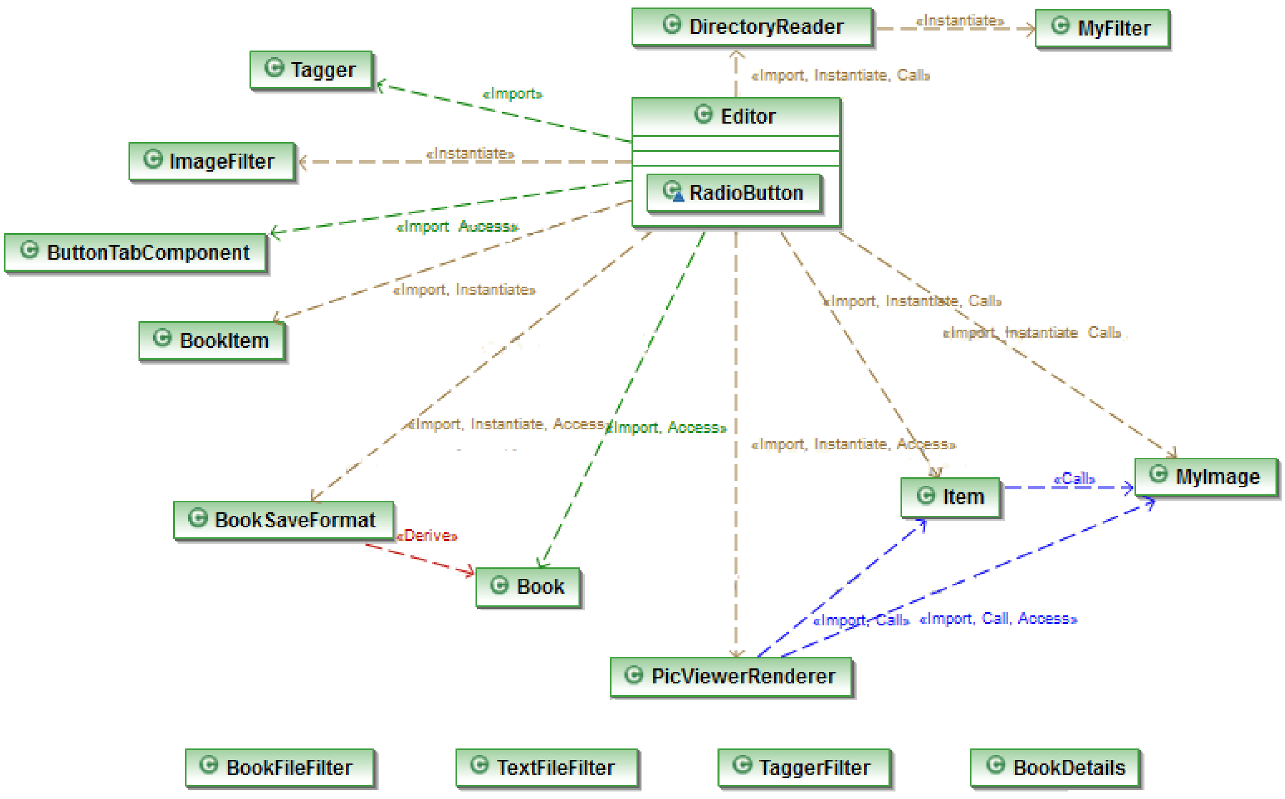


Figure 27 The "Tailored Book Editor" Class diagram

The RadioButton class that extends JRadioButton was implemented. This class is used to display the radio buttons with the parts of speech found in the WordNet database for a given word in the "Image Tagger". The item (part of speech) can be selected or deselected and depending on the selection the JList component is updated with the appropriate suggested synonyms.

The Editor class is the core class of the "Tailored Book Editor" and embraces all the functionality of the desktop application previously described. All the properties and methods of this class are shown in the Figure 30.

2.7. "Tailored Book Editor" implemented classes

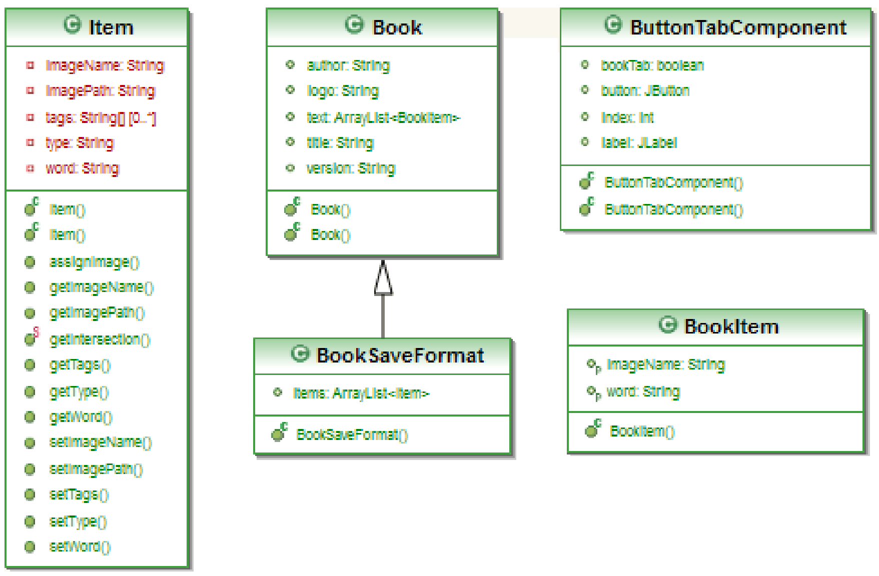


Figure 28 The "Tailored Book Editor", implemented classes: Item, Book, BookSaveFormat, ButtonTabComponent, BookItem

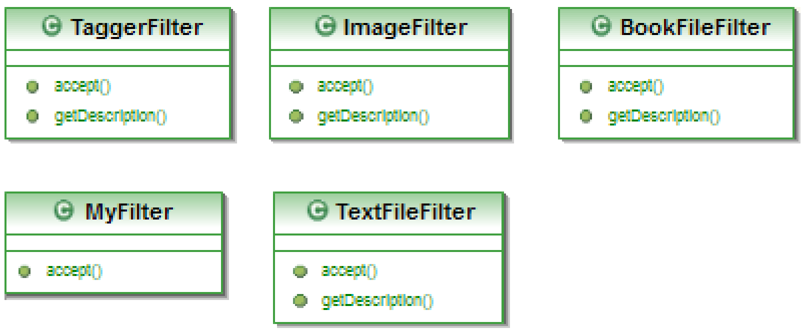


Figure 29 The “Tailored Book Editor” Implemented classes: TaggerFilter, ImageFilter, BookFileFilter, MyFilter, TextFileFilter

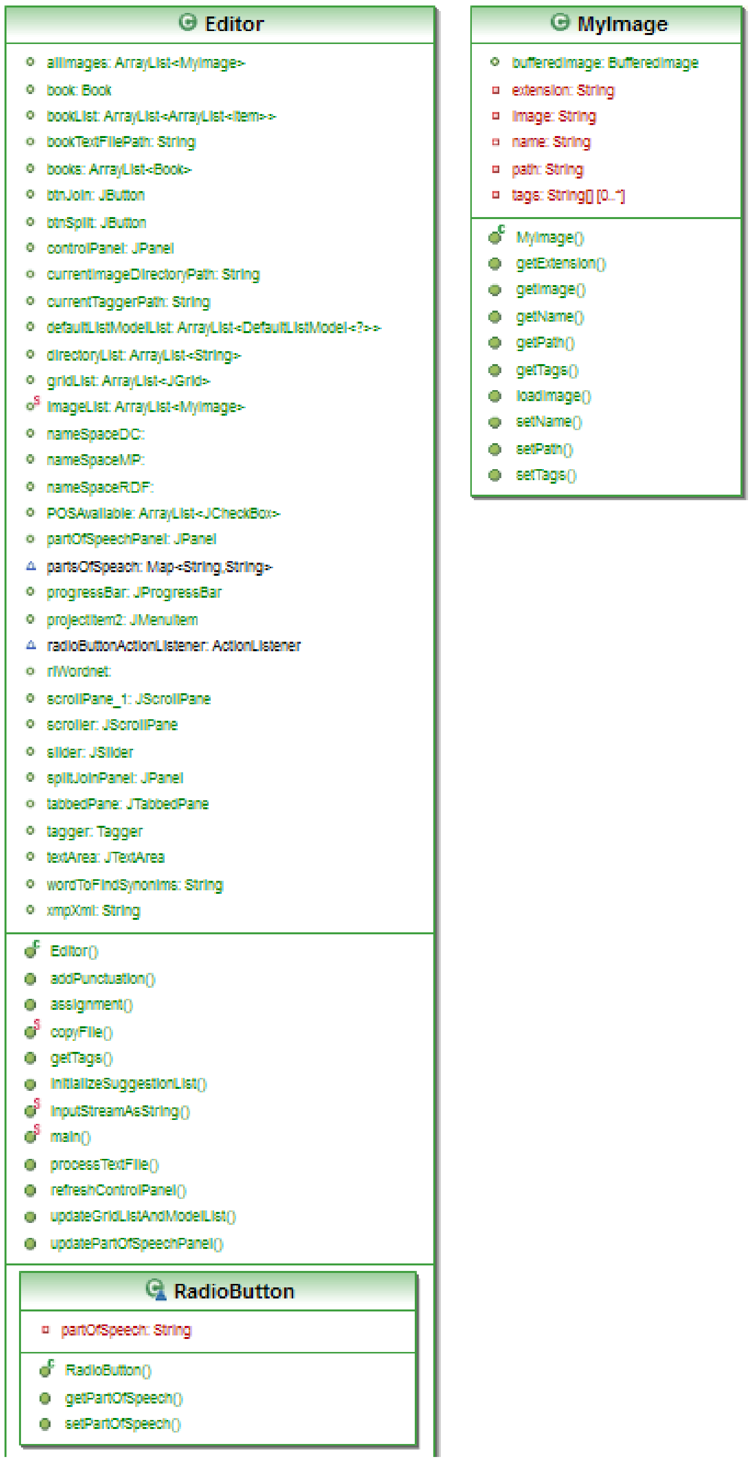


Figure 30 The “Tailored Book Editor” Implemented classes Editor, MyImage, RadioButton

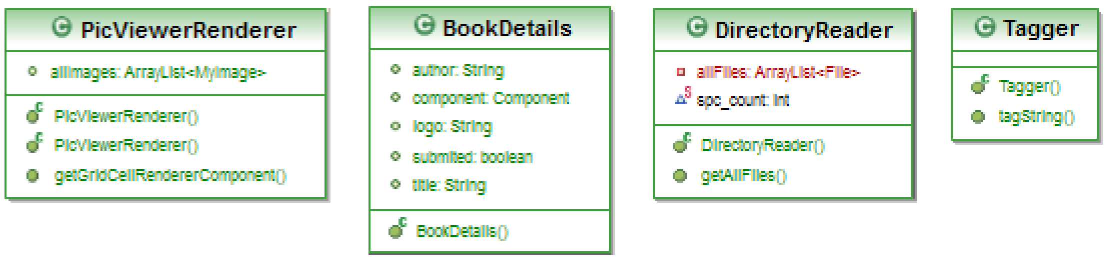
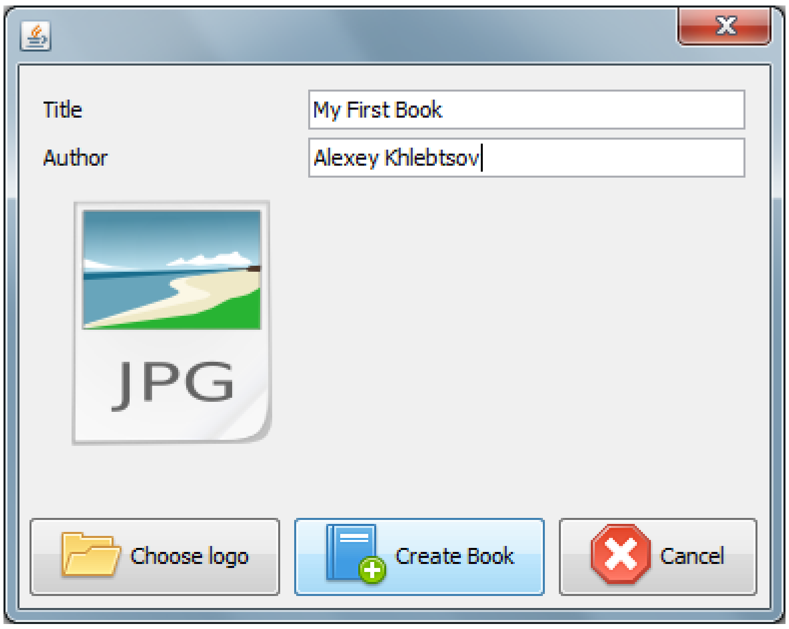


Figure 31 The "Tailored Book Editor" Implemented classes: PicViewRendere, BookDetails, DirectoryReader, Tagger

2.8. "Tailored Book Editor" screens

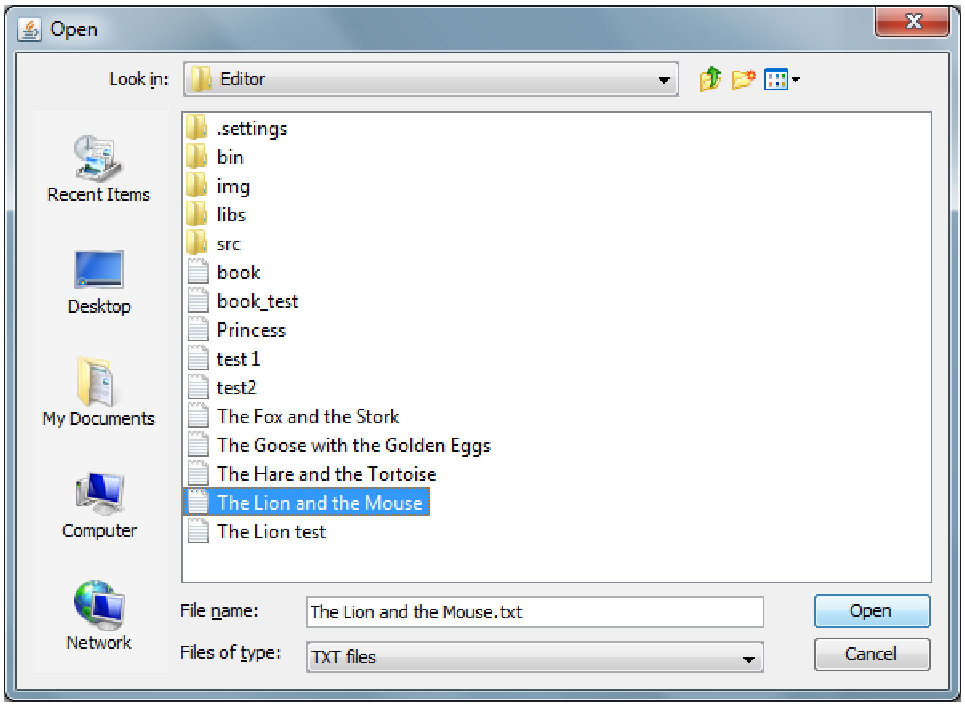
In the Figure 32 a screenshot of a dialog for a tailored book creation implemented by me for the "Tailored Book Editor" application is shown. This is the first dialog that appears to the user when he starts creating a tailored book. In this window the user should fill in two obligatory fields: title and author of the book. Also the system requires a logo of the book to be chosen; this logo will be displayed in the list of books in the "Tailored Book Reader" application.



Title: My First Book
Author: Alexey Khlebtsov

Figure 32 A screenshot of the "Tailored Book Creator" dialog

In the Figure 33 a screenshot of the next dialog for a .txt file selection implemented by me for the "Tailored Book Editor" application is shown. This dialog allows only picking the files in the text format.

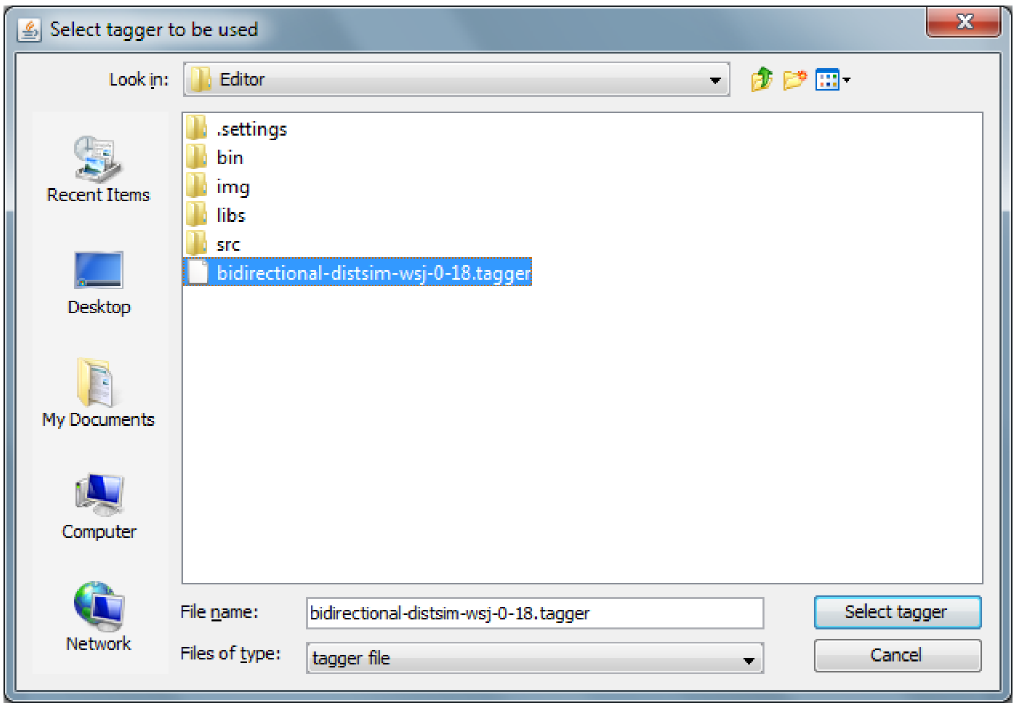


.settings
bin
img
libs
src
book
book_test
Princess
test 1
test2
The Fox and the Stork
The Goose with the Golden Eggs
The Hare and the Tortoise
The Lion and the Mouse
The Lion test

File name: The Lion and the Mouse.txt
Files of type: TXT files

Figure 33 A screenshot of a dialog for .txt file selection

In the Figure 34 a screenshot of a dialog for a .tagger file selection is shown. This dialog was implemented by me for the “Tailored Book Editor” application. This dialog accepts only the files in the .tagger format.

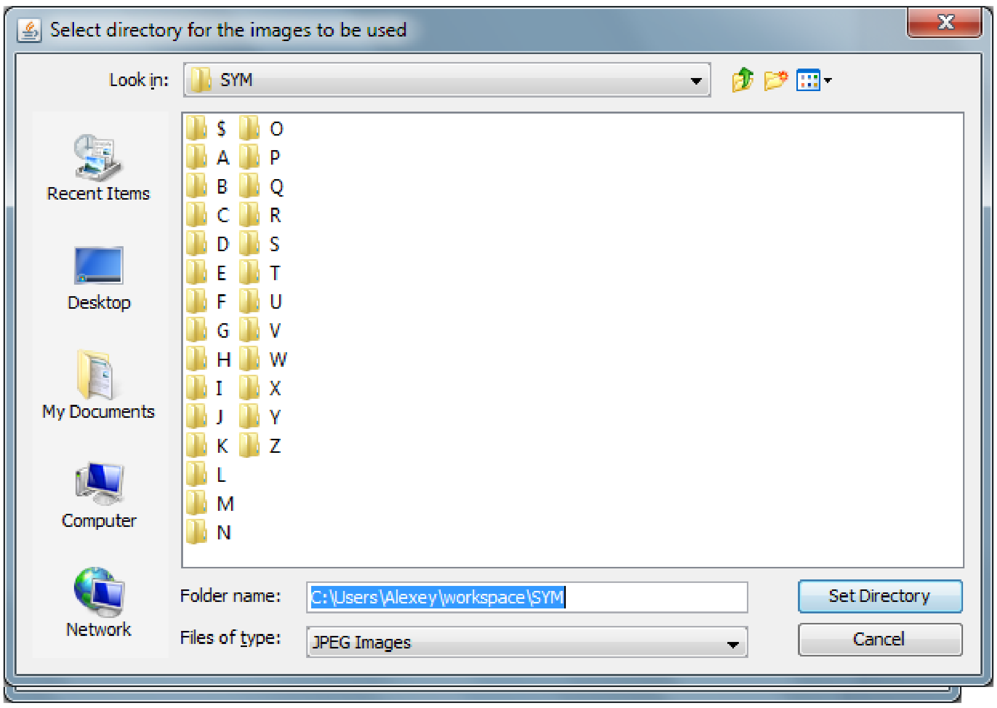


.settings
bin
img
libs
src
bidirectional-distsim-wsj-0-18.tagger

File name: bidirectional-distsim-wsj-0-18.tagger
Files of type: tagger file

Figure 34 A screenshot of a dialog for .tagger file selection

A screenshot of a dialog for selecting the image directory is shown in the Figure 35. This dialog was implemented by me for the “Tailored Book Editor” application and it allows only the selection of images in JPEG format.


Select directory for the images to be used
Look in: SYM
$ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Recent Items
Desktop
My Documents
Computer
Network
Folder name: C:\Users\Alexey\workspace\SYM
Files of type: JPEG Images
Set Directory
Cancel
Figure 35 A screenshot of a dialog for the image directory selection

A screenshot of the "Tailored Book Editor' interface developed and implemented by me is shown in the Figure 36. In this screenshot you can see a selected figure (green frame). In the Control Panel you can see the tags associated with this figure. Inside of this figure there is a text which is composed of two words. Since the number of words in the figure is more than one, a button "Split" of the Control Panel is active. Therefore the figure can be split into two figures.

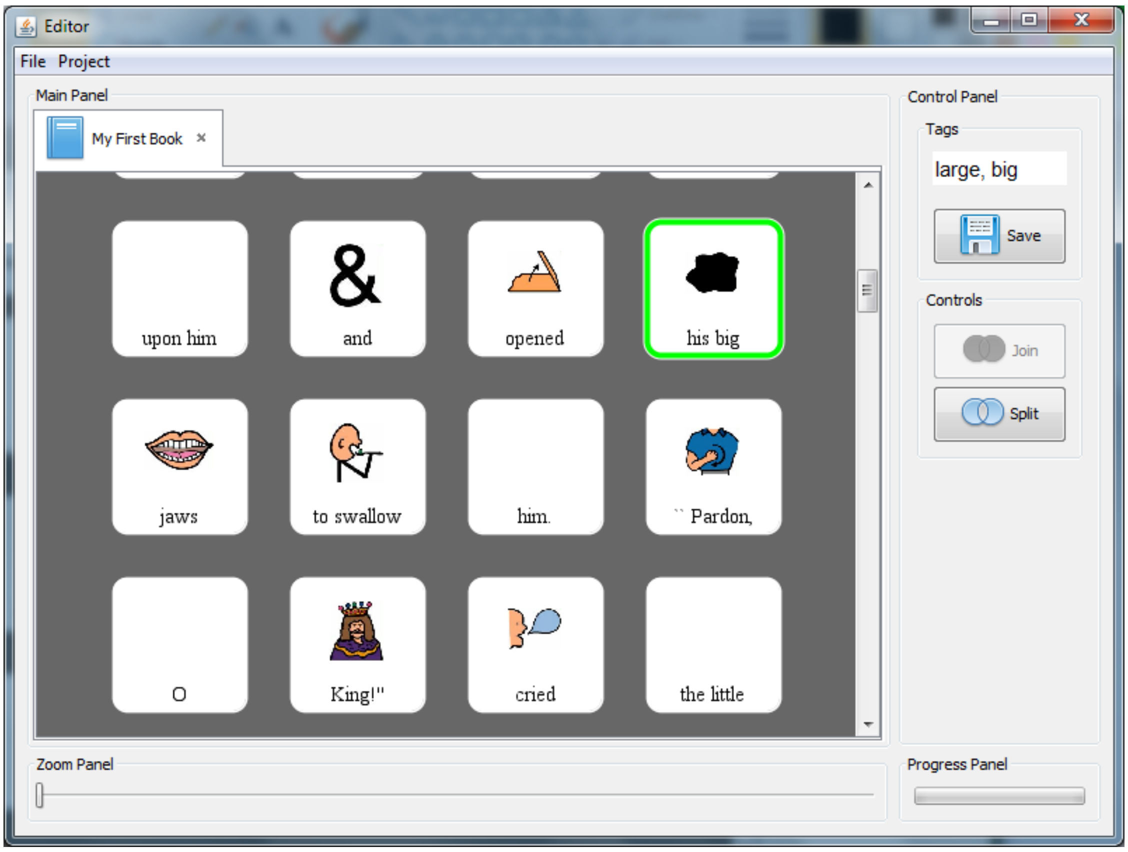


Figure 36 A screenshot of the "Tailored Book Editor" interface; single selection of the figure for further split

In the Figure 37 you can see the same interface of the “Tailored Book Editor” that was shown in the Figure 36 but in this case the previously selected figure has been already split into two figures. After splitting the system assigns the tags for each item. For example the figure “his big” has the tags: large, big; and after splitting we get one figure “his” with tag his and the second figure “big” with tags: large, big. In the Figure 37 you can see that when more than two figures are selected the “Join” button of the Control Panel becomes active. This button allows joining the figures into the single figure. In this case the tag of the first figure is assigned to the newly combined figure. The tag can be easily changed using the Control Panel of the “Tailored Book Editor” application.

In the Figure 38 a project menu of the “Tailored Book Editor” is shown. Using this menu the user can set the image source directory and publish the currently opened tailored book.

In the Figure 39 a main menu of the “Tailored Book Editor” is shown. Using this menu the user can open the new image tagger; start the process of new book creation; open and save the tailored book.

In the Figure 40 a screenshot of the “Image Tagger” interface is shown. In this screenshot you can see the grid of images the user opened for further tagging. The selected image (green frame) has been already tagged by the user with the tags: help, assist, aid.

In the Figure 41 a screenshot of the “Image Tagger” interface created by me is shown. As you can see an image with the plane is selected. In the Control Panel the user has typed the tag plane and when he has put a comma the panel has been expanded. Now the Control Panel contains the suggestion list of tags and above it there are three radio buttons with parts of speech available for the word the user typed. The list of suggested words is taken from the WordNet database. This functionality was designed and implemented by me in this project.

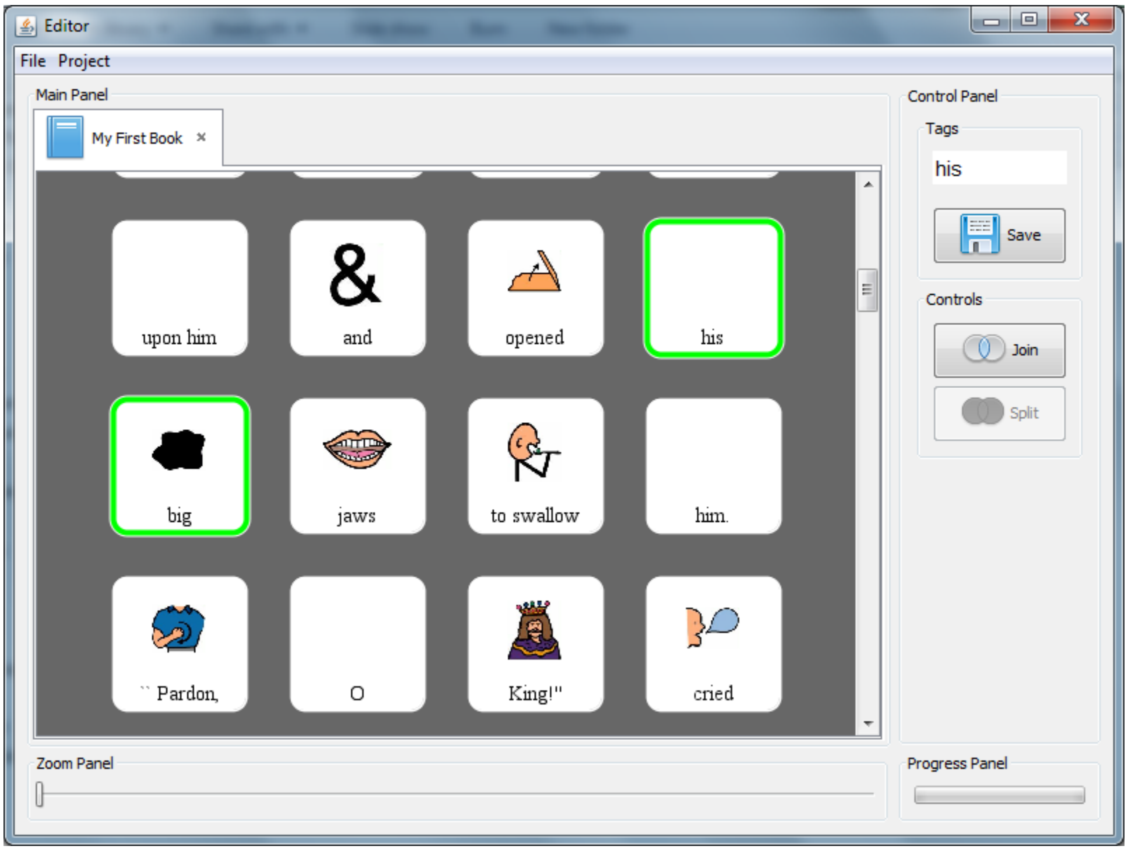


Figure 37 A screenshot of the “Tailored Book Editor” interface; two figures selection for further join

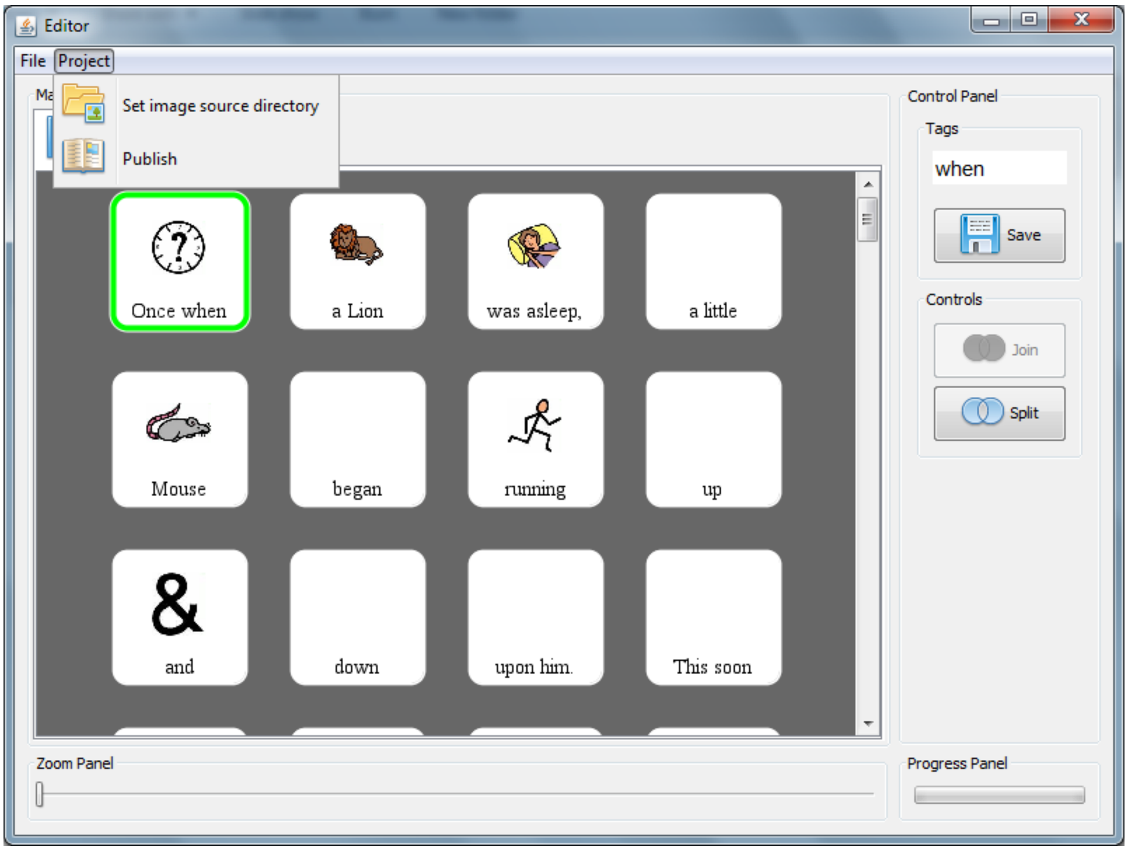


Figure 38 A screenshot of the “Tailored Book Editor” interface; Project menu

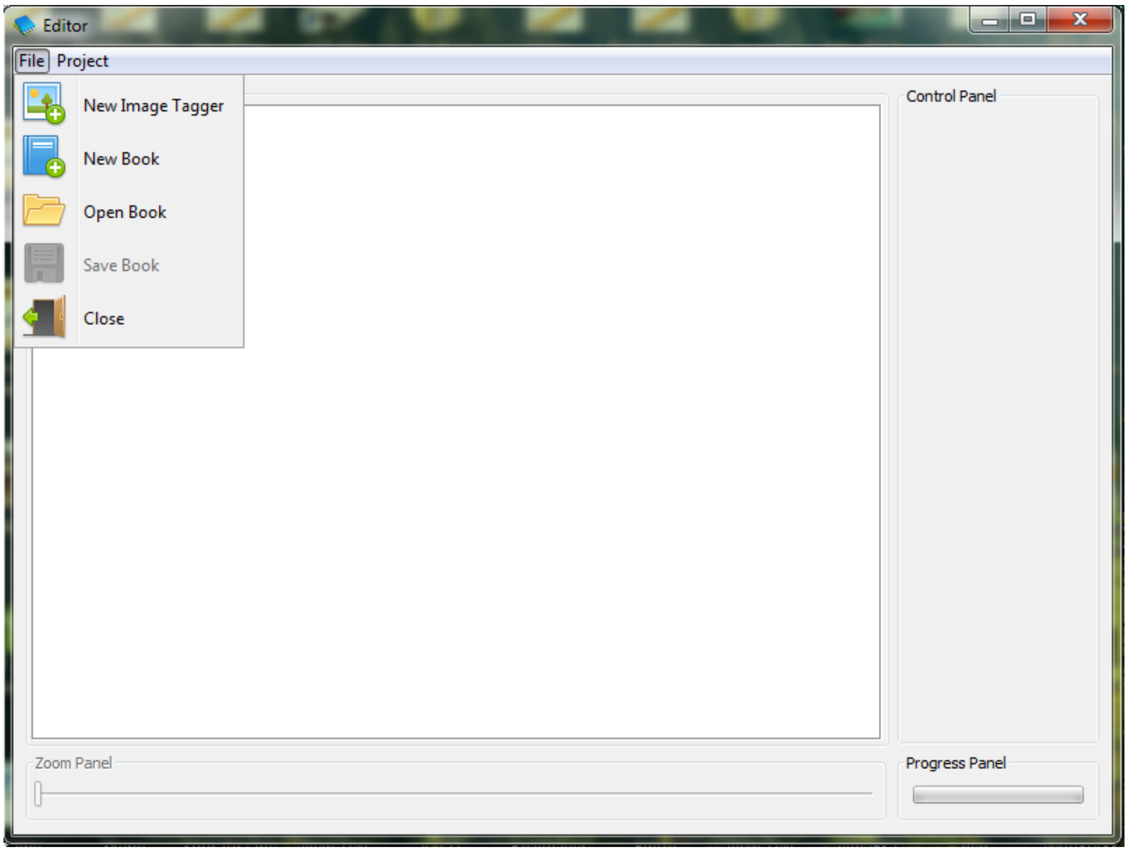
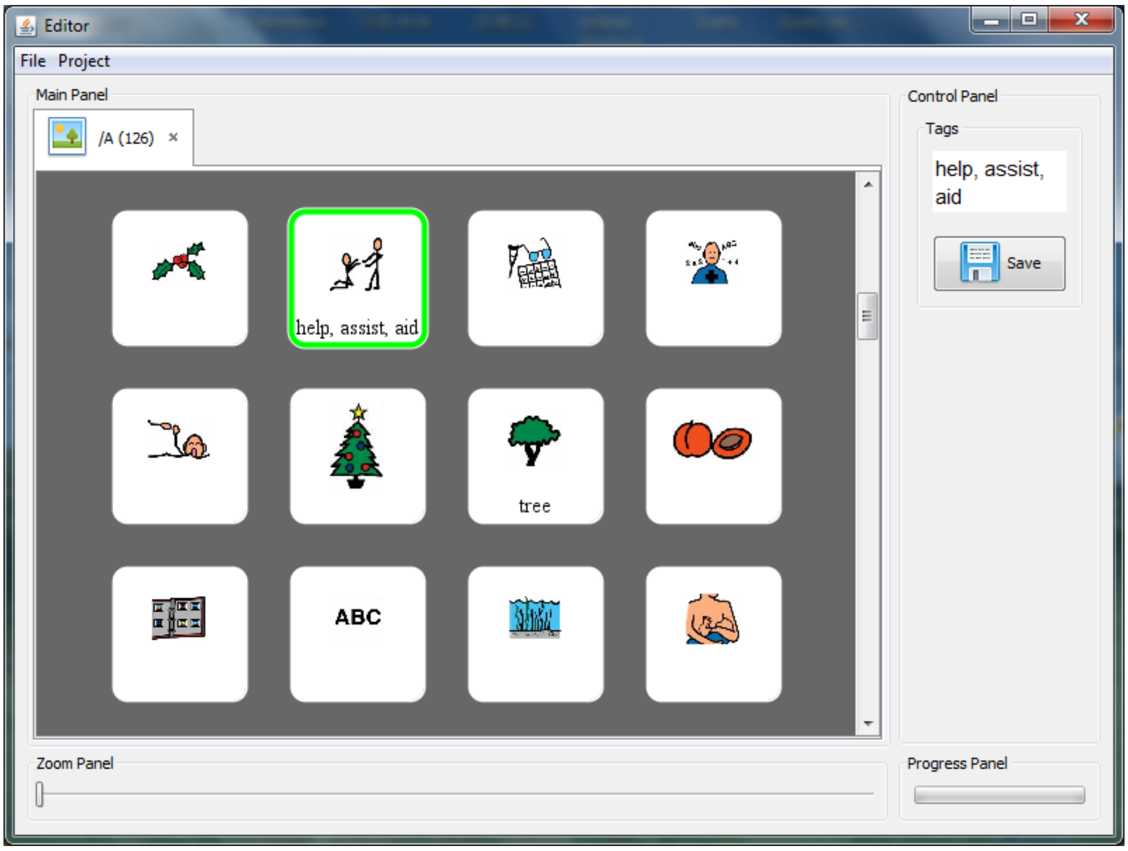
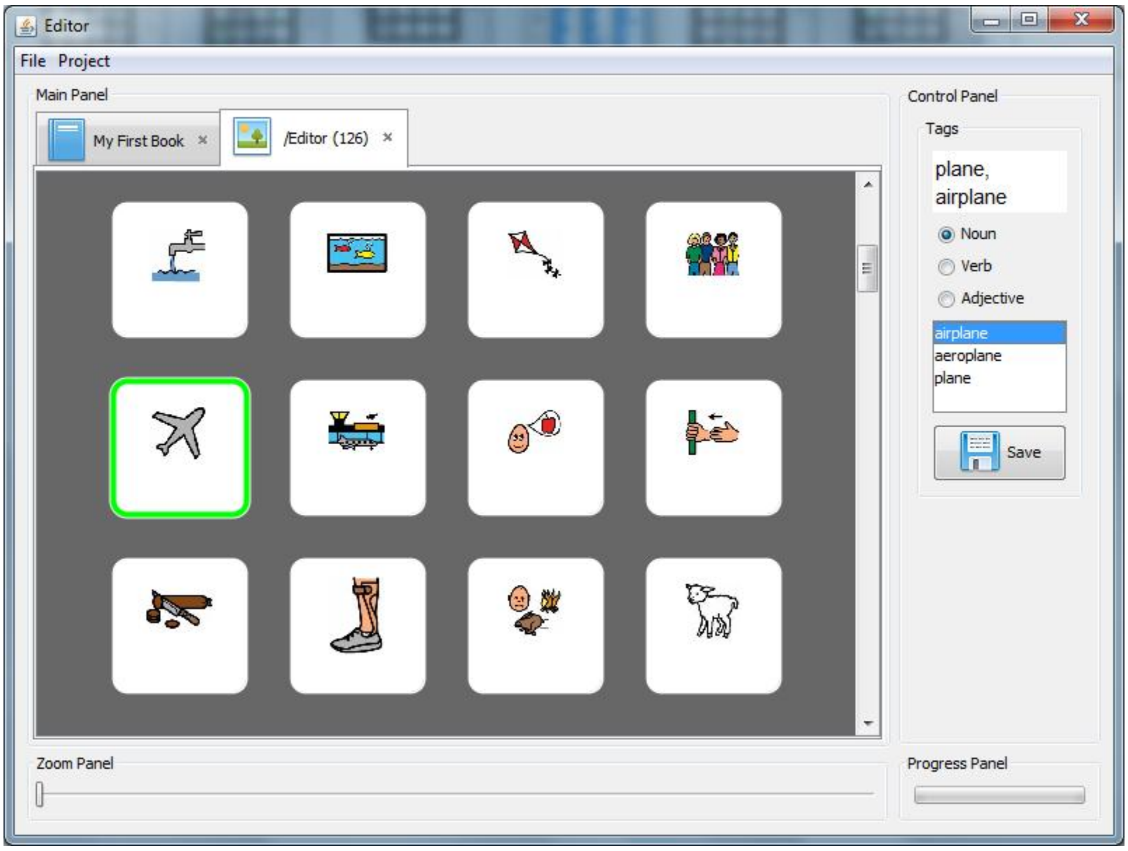


Figure 39 A screenshot of the “Tailored Book Editor” interface; File menu



Tags: help, assist, aid

Figure 40 A screenshot of the “Image Tagger” interface



Tags: plane, airplane

(● Noun) ○ Verb ○ Adjective

airplane
aeroplane
plane

Figure 41 A screenshot of the “Image Tagger” interface; Suggestion list

3. “Tailored Book Reader”

3.1. “Tailored Book Reader” front-end

3.1.1. “Menu” screen features and layout

A “Menu” screen serves as the main navigation screen in the application. This is the first screen shown to the user when the app starts and requires the user to choose where to go next. Its functionality is encapsulated within the ViewPagerMenuActivity class. The description of this class will be given below in the “Tailored Book Reader” back-end section. This screen should allow user to choose “Download Books”, “My Books” or “Info” screens. The Figure 42 shows a mockup of the main menu screen.

A layout of the “Main” screen contains the following elements: a LinearLayout, a TextView and a ListView. The LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. In our case the orientation of the LinearLayout is vertical and it occupies the whole screen. The vertical orientation of the LinearLayout means that all other elements or childs added to this layout are placed in vertical order, one after another, from top to bottom.

The first added element is the TextView that serves as a header of the screen. It displays the screen name: “Menu”. The second element added to the LinearLayout is called the ListView. The ListView displays a list of items in a vertically scrolling list.

The “Menu” screen and the described below “Download Books” and “My Books” screens have the uniform layout. The layout for all three screens is shown in the Figure 43.

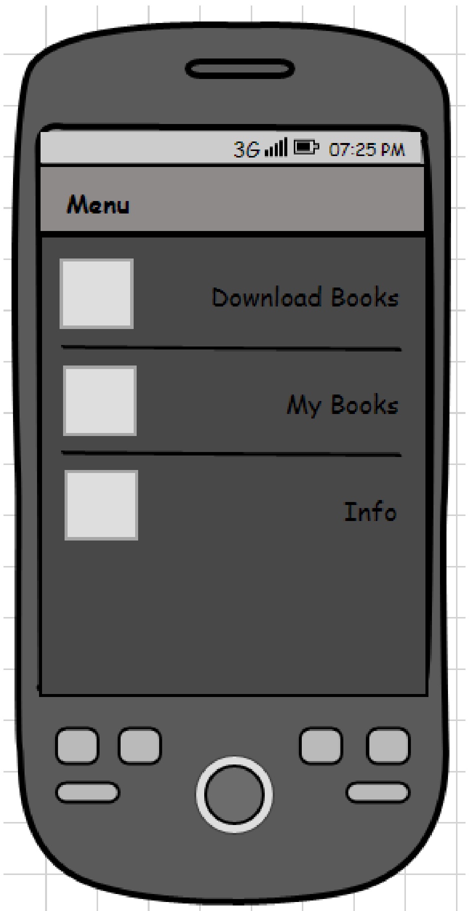


Figure 42 Mockup of the “Menu” screen

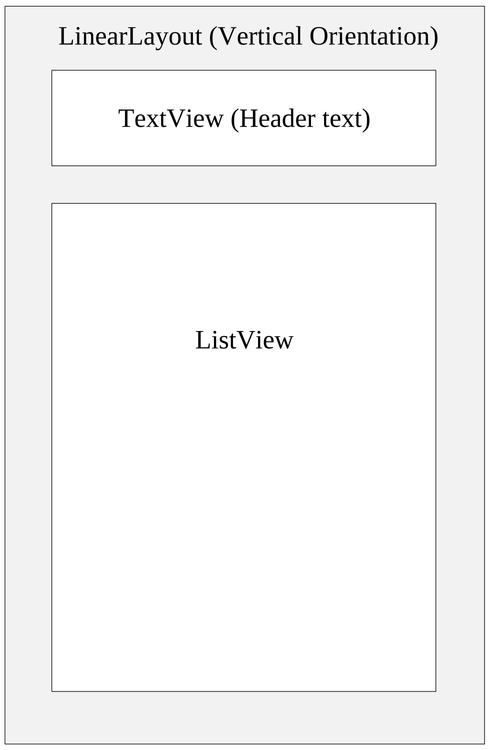


Figure 43 Layout design for “Menu”, “Download Books” and “My Books” screens

3.1.2. “Info” screen features and layout

An “Info” screen explains the user what the “Tailored Book Reader” application is about and how to use it. Its functionality is encapsulated within the ViewPagerInfoActivity class. The description of this class will be given below in the “Tailored Book Reader” back-end section. This screen displays to the user a text about the application and enables the user to scroll up and down through the text. Figure 44 shows a mockup of the “Info” screen.

A layout of “Info” screen is very similar to the layout described above for “Menu” screen. The only difference is that the ListView element is replaced by the TextView one. The first TextView like in previous case serves as a header; while the second TextView is used to display the content of the info page.

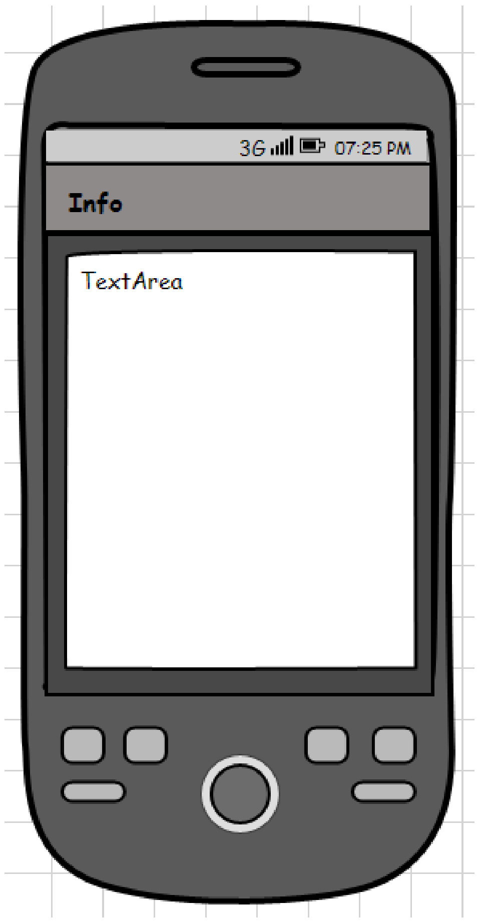


Figure 44 Mockup of the “Info” screen

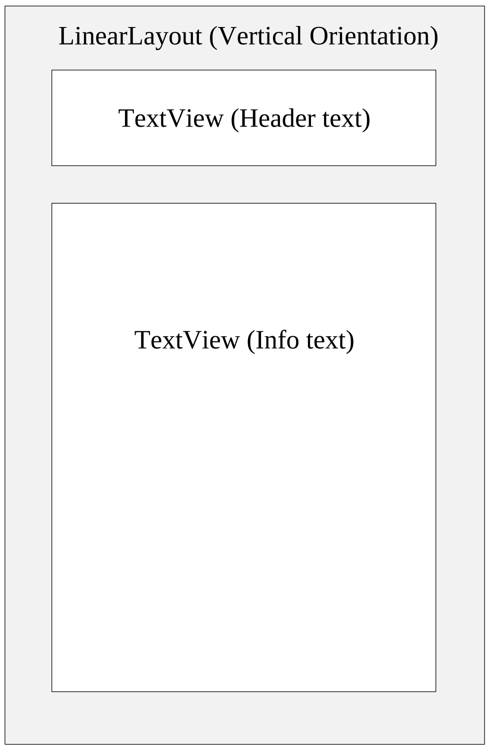


Figure 45 Layout design for “Info” screen

3.1.3. “Download Books” and “My Books” screen features and layouts

A “Download Books” screen displays a list of tailored books available for download on the server side. Its functionality is encapsulated within the ViewPagerDownloadActivity class. The description of this class will be given below in the “Tailored Book Reader” back-end section. When this screen is accessed by the user the “Tailored Book Reader” tries to connect to the server and loads the content of the index file called books.khl, which contains the information about the book currently located on the server. The list row contains a tailored book title and logo. By clicking on a row the user can download an appropriate book. This screen also displays the current download status using a progress dialog which shows the name of the book being downloaded and the number of images that has been downloaded so far over the total number. The “Tailored Book Reader” downloads a tailored book with extension .khl and all images associated with the book and saves them all on SD card of the mobile device. Figure 46 shows a mockup of the Download Books screen.

A "My Books" screen shows all the books available on the user device. Its functionality is encapsulated within the ViewPagerMyBooksActivity class. The description of this class will be given below in the "Tailored Book Reader" back-end section. The "Tailored Book Reader" reads tailored book from the SD card of the mobile device and displays their titles and logos as a list. Since all the books downloaded from the server side are saved on the SD card, this list displays all tailored books: both books put on the SD card by user and downloaded from the server. When the user selects an appropriate book the consecutive activity responsible for book visualization is loaded. At this moment the "Tailored Book Reader" is responsible for preparation of the selected book for further visualization on the "Book" screen. Figure 47 shows a mockup of the "My Books" screen.

The layouts of both "Download Books" and "My Books" screens are the same as a layout of the "Menu" screen. The description of the "Menu" screen layout was written before in the section Menu Screen Features and Layout. The layout itself is depicted in the Figure 43 Layout design for "Menu", "Download Books" and "My Books" screens.

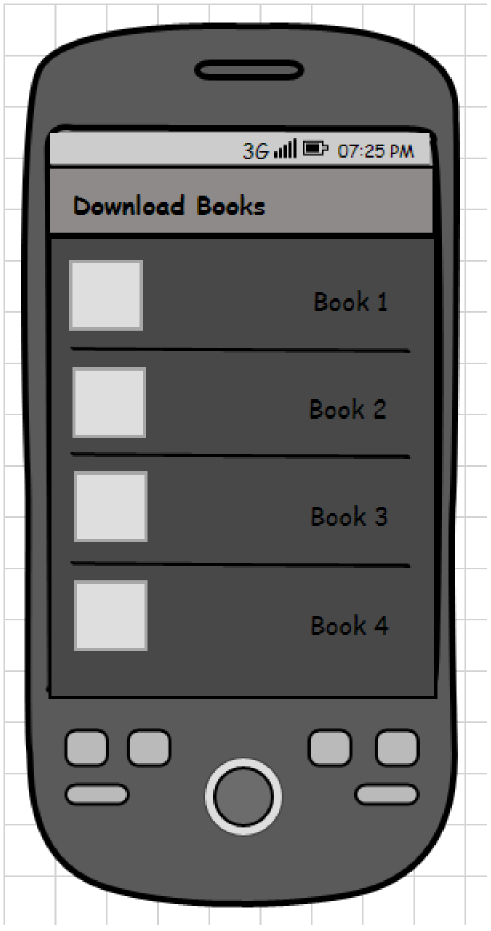


Figure 46 Mockup of the "Download Books" screen

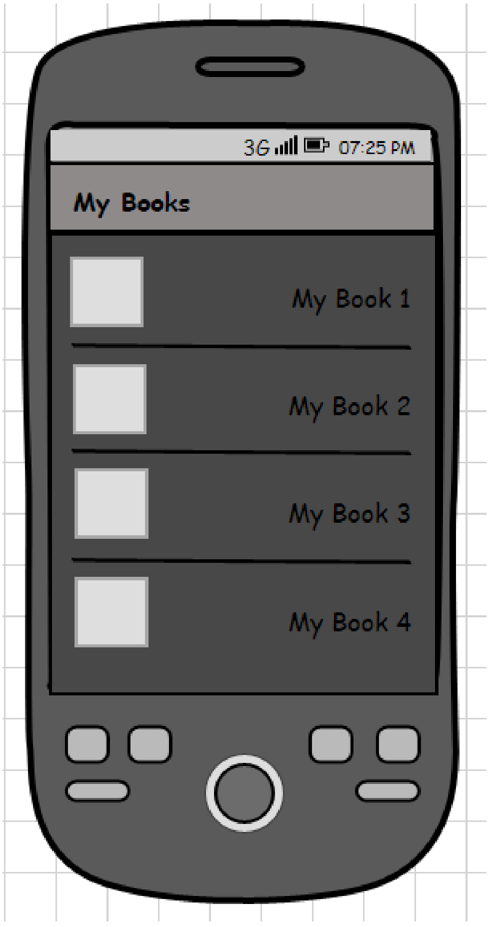


Figure 47 Mockup of the "My Books" screen

3.1.5. "Book" screen features and layout

A "Book" screen is the most important screen of the "Tailored Book Reader" application, because this screen displays the content of the selected tailored book and user will spend most of the time on this screen while reading, listening to the book or practicing the pronunciation. The functionality of this screen is encapsulated within the ViewPagerActivity class. The description of this class will be given below in the "Tailored Book Reader" back-end section.

The “Tailored Book Reader” loads tailored books from the SD card and displays them to the user. Android OS provides a feature (from Android 1.6) called Text-to-Speech. It is capable of synthesizing a speech from text for an immediate playback. I implemented the TextToSpeech listener and “The Tailored Book Editor” can pronounce the words inside the figure on click using TTS. The “Book” screen contains the options menu with two menu items: study and normal mode. The user can reveal the options menu panel by pressing the Menu button. The study and normal modes are the modes of reading the page aloud using TTS technology. The both modes have been implemented by me. The normal mode reads the current page of the book aloud with the normal speed observing punctuation. The study mode reads the current page at a slow pace and highlights the figure of the current word being spoken. The implemented “Book” screen navigation enables user to go first/last page and next/previous using buttons in the footer. Also the user can use swipe gesture to go to next/previous page. Figure 48 shows a mockup of the “Book” screen.

A “Book” screen layout contains the following elements: three RelativeLayouts, a ViewPager, two ImageButtons, two Buttons and two TextViews. The Figure 49 shows the layout design for “Book” screen. IB stands for the ImageButton and B for the Button.

The first RelativeLayout occupies the whole screen. The RelativeLayout allows specifying how child views are positioned relative to each other. On top of it there are other two RelativeLayouts. The second relative layout contains a text view and a view pager; the third one - image buttons, a text view and buttons. The TextView of the RalativeLayout serves as a header of the screen. The second element is the ViewPager. The View Pager is a layout manager that allows the user to flip left and right through pages of data. In other words The ViewPager is responsible for visualizing other layouts organized in pages and enables the user to go through the pages by horizontal swiping. The third relative layout is dedicated to a navigation bar. Two image buttons represent controls for loading first or last page of the ViewPager and two normal buttons are used to go to next or previous pages. The text view of third relative layout is used to display current page of the ViewPager.

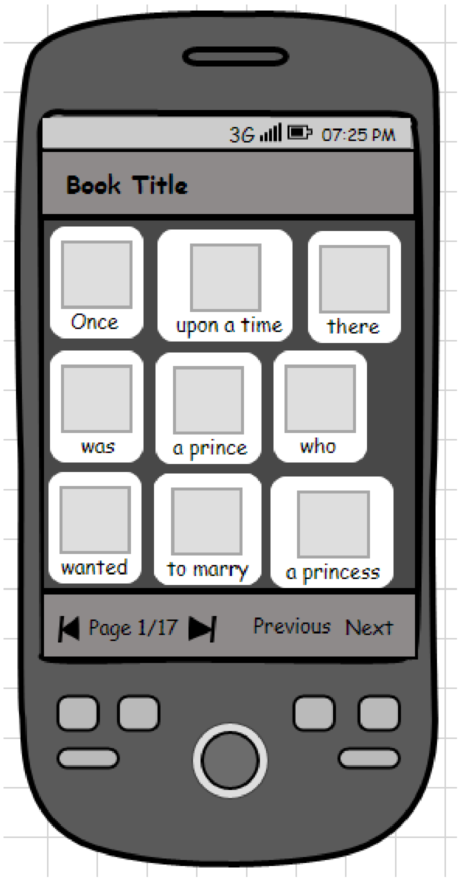


Figure 48 Mockup of the “Book” screen

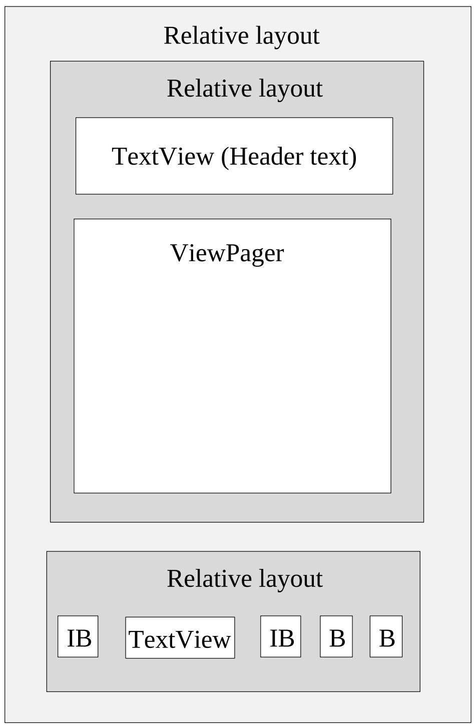


Figure 49 Layout design for “Book” screen

In the Figure 50 a layout design for a Box item is shown. The Box item layout represents a figure or a tailored book item. This is the layout for a custom component called Box, specially developed for a tailored book. The Box item is composed of two LenearLayouts, TextView and ImageView. Box items are displayed on the ViewPager Layout Page.

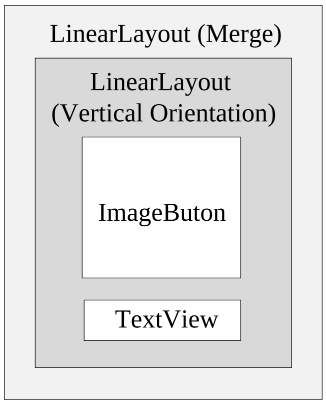


Figure 50 Layout design for Box item

3.2. “Tailored Book Reader” back-end

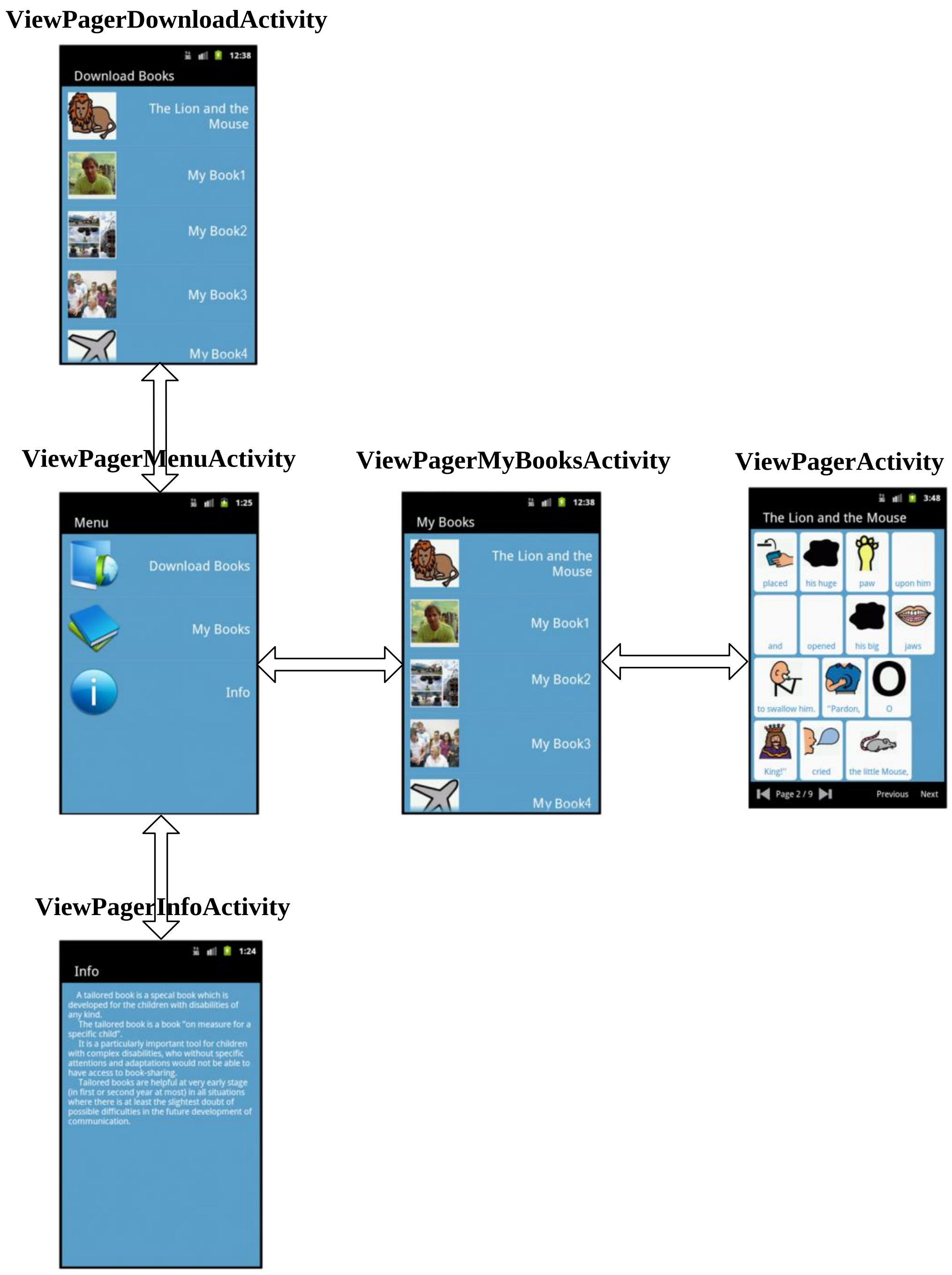
3.2.1. Activity workflow

An Activity is an application component that provides a screen with which users can interact in order to do something (e.g. clicking a menu item to go to another activity). “Tailored Book Reader” application is composed of five activities that are bound to each other: ViewPagerMenuActivity, ViewPagerInfoActivity, ViewPagerMyBooksActivity, ViewPagerDownloadActivity, ViewPagerActivity.

Each activity is given a window in which to draw its user interface: ViewPagerMenuActivity represents the main menu screen; ViewPagerInfoActivity represents the info screen; ViewPagerMyBooksActivity represents the user downloaded books screen; ViewPagerDownloadActivity represents the books available on the server; ViewPagerActivity represents the book screen. The activity workflow is shown on the Figure 51.

Typically, one activity in an application is specified as the “main” activity, which is presented to the user when launching the application for the first time. As you can see from the activity workflow, ViewPagerMenuActivity is considered to be main activity. From the main activity you can go to the Download Books screen, My Books screen and Info screen by clicking the appropriate menu item. When the action is performed the corresponding activity is launched. ViewPagerMyBooksActivity allows launching ViewPagerActivity or coming back to the main Menu using Back button of the device. ViewPagerActivity is launched by selecting the appropriate book from the list of available book previously downloaded from ViewPagerDownloadActivity or the books manually placed on SD card of the device.

Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the “back stack”). The back stack uses the basic “last in, first out” stack mechanism, so, when the user is done with the current activity and presses the Back button, it is popped from the stack and the previous activity resumes.



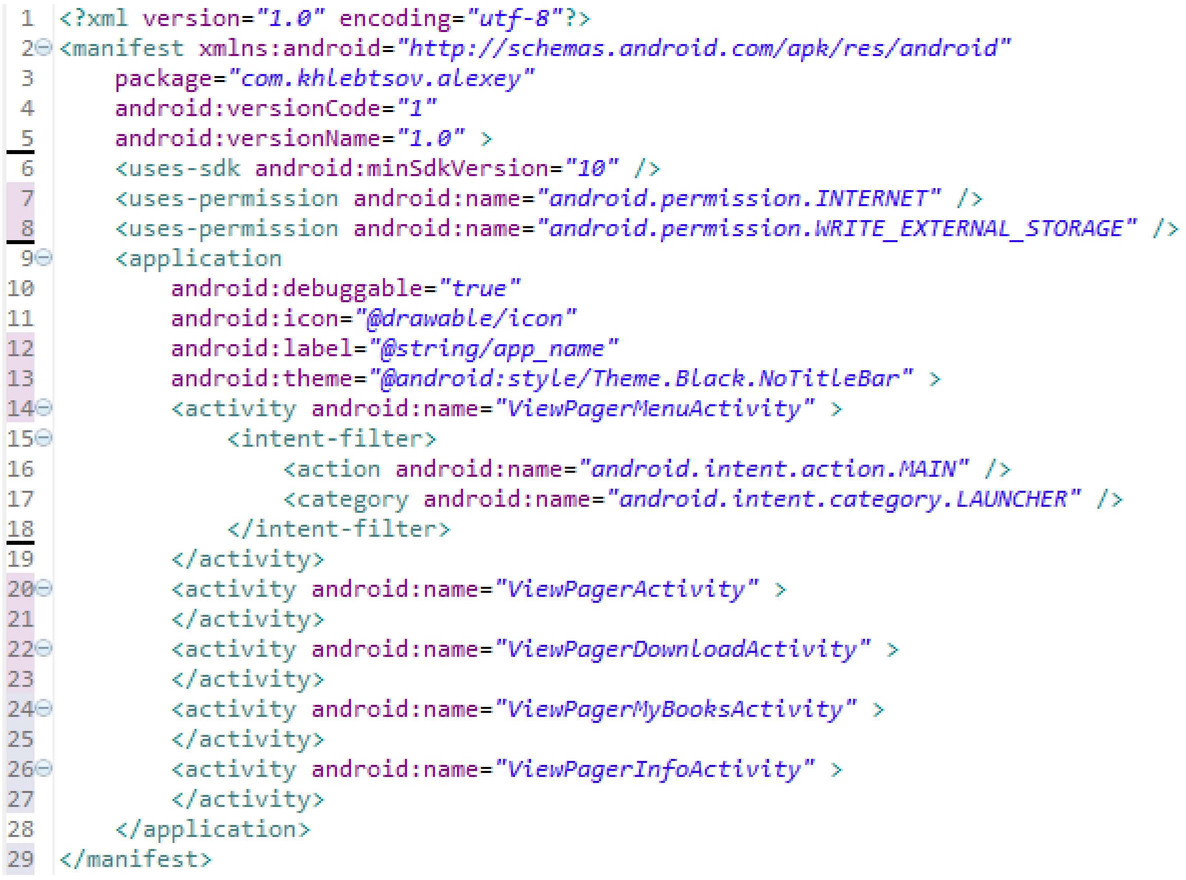
A tailored book is a specal book which is developed for the children with disabilities of any kind.
The tailored book is a book "on measure for a specific child".
It is a particularly important tool for children with complex disabilities, who without specific attentions and adaptations would not be able to have access to book-sharing.
Tailored books are helpful at very early stage (in first or second year at most) in all situations where there is at least the slightest doubt of possible difficulties in the future development of communication.

Figure 51 The "Tailored Book Reader" activity workflow

3.2.2. "Tailored Book Reader" configuration file

Activities are defined in a central configuration file of the "Tailored Book Reader" mobile application. This file is called AndroidManifest.xml. In the manifest file an Intent filter is used to designate an activity as the primary entry point of the application (See Figure 52).

Among activities, the manifest file also declares which permissions the application must have in order to access protected parts of the API and interact with other applications; the minimum level of the Android API that the application requires; application name and icon; interface theme and other fields. For "Tailored Book Reader" the following permissions were set: INTERNET and WRITE_EXTERNAL_STORAGE. The user is in charge of granting the permission to an application that requests it. For the normal functioning of the application the user should grant these permissions. The INTERNET permission allows the application to connect the internet for downloading books available online. The WRITE_EXTERNAL_STORAGE permission is used to save the books downloaded from the internet to the SD card of the user mobile device.



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.khlebtsov.alexey"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:debuggable="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.Black.NoTitleBar" >
        <activity android:name="ViewPagerMenuActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="ViewPagerActivity" >
        </activity>
        <activity android:name="ViewPagerDownloadActivity" >
        </activity>
        <activity android:name="ViewPagerMyBooksActivity" >
        </activity>
        <activity android:name="ViewPagerInfoActivity" >
        </activity>
    </application>
</manifest>
```

Figure 52 The central configuration file of the “Tailored Book Reader” application

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform. For “Tailored Book Reader” the minimum SDK version is set to 10, which means that the application will run on all devices starting from the Android 2.3.3. Deciding which platform to target is a trade-off between new features and the device coverage. For example newer versions of Android have more features, but targeting an older version allows more devices (including older devices) to use the application. With API 10 the application covers 84,2% of the devices distributed on the market (See table 27).

Version	**Codename**	**API**	**Distribution**
1.5	Cupcake	3	0.1%
1.6	Donut	4	0.3%
2.1	Eclair	7	3.1%
2.2	Froyo	8	12%
2.3 - 2.3.2	Gingerbread	9	0.3%
2.3.3 - 2.3.7		10	53.9%
3.1	Honeycomb	12	0.4%
3.2		13	1.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	25.8%
4.1	Jelly Bean	16	2.7%

Table 8 Current Device Distribution in November 1, 2012

The Android platform provides a large collection of styles and themes that can be use in the applications. The reference of all available styles is in the R.style class. For “Tailored Book Editor” application the ThemeBlackNoTitleBar theme is applied (See Figure 52 The central configuration file of the “Tailored Book Reader” application): "@android:style/Theme.Black.NoTitleBar".

3.2.3. “Tailored Book Reader” class diagram

As was mentioned before the ViewPager is a layout manager that allows the user swiping left and right through pages of data. The ViewPager needs a data adapter to determine and load the appropriate content for each page the user swipes to. The MyPagerAdapter extends the PagerAdapter class, and implements several key methods.

Firstly, the size of the paging range should be defined. In our case, the number of pages depends on the screen size and they are calculated during runtime while the book is being prepared to display on the screen. Thereby on the devices with different screen sizes the number of pages will be different.

Secondly, the instantiateItem() method should be implemented. This method inflates the appropriate layout resource file, depending on the user’s swipe position. The farthest page to the left is in position 0, the next page to the right is position 1, and so on. The instantiateItem() method uses the LayoutInflater service to inflate the specific layout and add it to the collection used by the ViewPager. During this method the custom GUI elements called Boxes (designed and implemented by me) are added to the inflated layout. The number of Boxes per page is calculated during initialization of MyPagerAdapter.

The last important method that should be implement is the destroyItem() method, which removes the specific layout from the collection used by the ViewPager when it is no longer being displayed.

The initial position of the pager by default is set to 0 (the first page). In order to change the page the user can use the swipe left right gesture, moreover the navigation system is implemented, which allows to go to last/first and next/previous page. The corresponding buttons of the control system use the ViewPager method called setCurrentItem() to change the page programmatically.

Book class contains all information about the book and the book content. The properties of the Book class are shown on Figure 57.

For real-world mobile applications, the default look and feel of the Android ListView is not very attractive. It only renders a simple String in every ListView row using the internal TextView control. For this application, I'd like to create an interface that is more graphically rich and visually pleasing to the user. The ListView is very powerful control and with the help of custom item layouts, it can be customized. This Book class is also used to create a custom ArrayAdapter and to bind the objects with the ListView. The values of book title and book author of the Book class properties are displayed on the ListView. A custom ArrayAdapter called BooksAdapter inherits the Android ArrayAdapter class and overrides the getView method. The constructor of the BooksAdapter class takes three parameters. The first parameter is the Context and we can pass the reference of the activity in which we will use the BooksAdapter class. The ViewPagerMyBooksActivity and the ViewPagerDownloadActivity use the BooksAdapter class. The second parameter is the resource id of the layout file that is used for displaying each ListView item. I pass the resource id of the layout row.xml for this parameter. The third parameter is an array of Book class objects that will be used by the Adapter to display data. The BooksAdapter class getView method is overridden. This method is called for every item in the ListView to create views with their properties. The getView method is also using a temporary holder class called BookHolder that is declared inside the BooksAdapter class. This class will be used to cache two TextViews so they can be reused for every row in the ListView and this provides a great performance improvement when recycling the same two views with different properties and there is no need to find TextViews for every ListView item.

The MenuAdapter also extends the ArrayAdapter class and its implementation is similar to the BooksAdapter class. The MenuRow class represents menu item and has title and image properties. This MenuRow class is used to create a custom MenuAdapter and to bind the objects (array of MenuRow objects) with the ListView. A temporary holder class called MenuHolder declared inside the MenuAdapter class is similar to the BookHolder class.

Both the ImageFileFilter and the BookFileFilter implement the Java FileFilter interface. The ImageFileFilter class matches filenames that end with image file extensions .jpg and .JPG. The key thing about implementing a FileFilter is that a method “Accept” should be implemented. This method returns true for JPEG files and false otherwise. By analogy the BookFileFilter filters all files except tailored book .khl file.

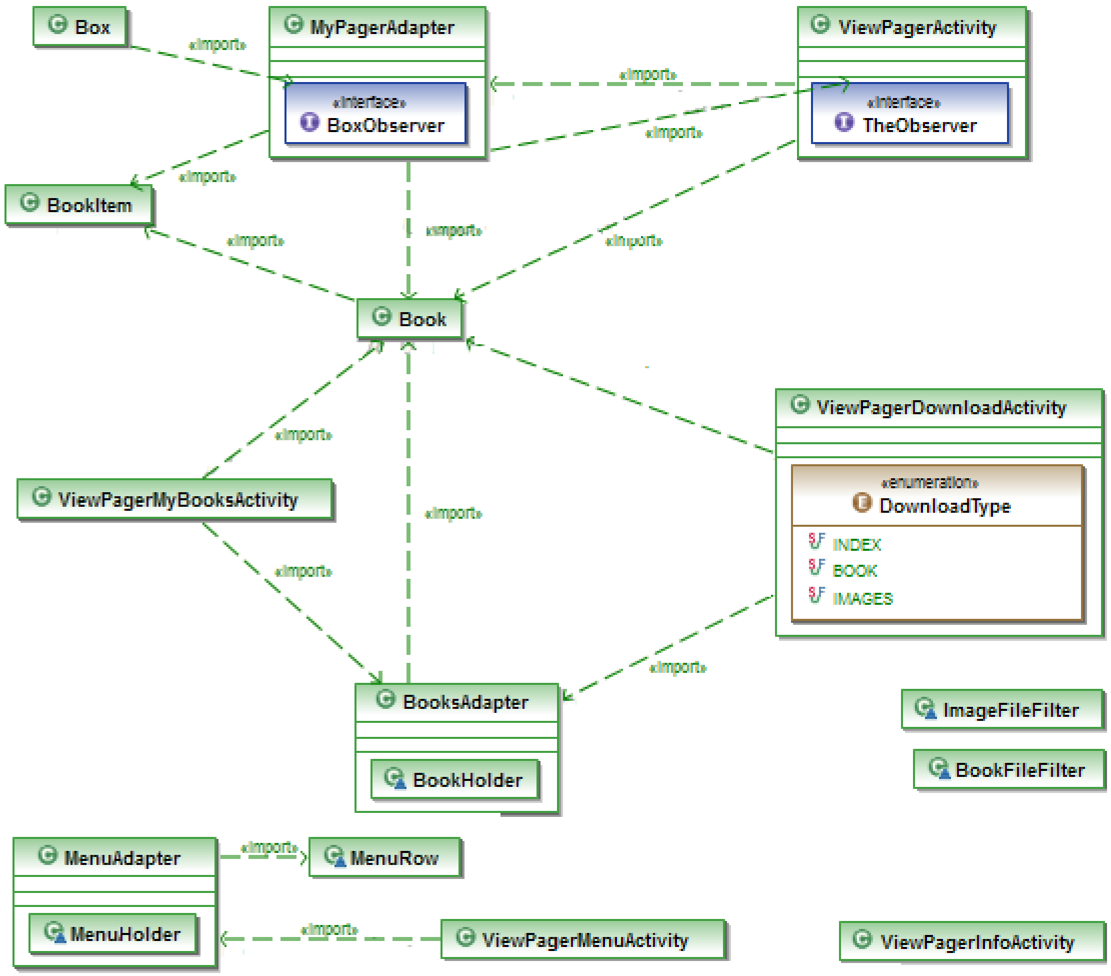


Figure 53 The “Tailored Book Reader” Class diagram

3.2.4. “Tailored Book Reader” implemented classes

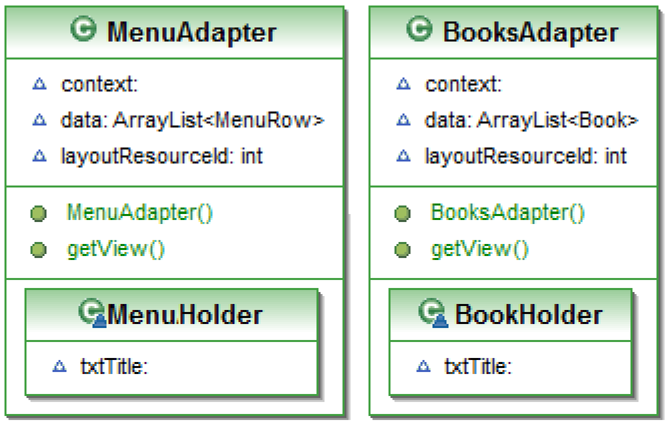


Figure 54 The “Tailored Book Reader”, implemented classes: MenuAdapter, BookAdapter

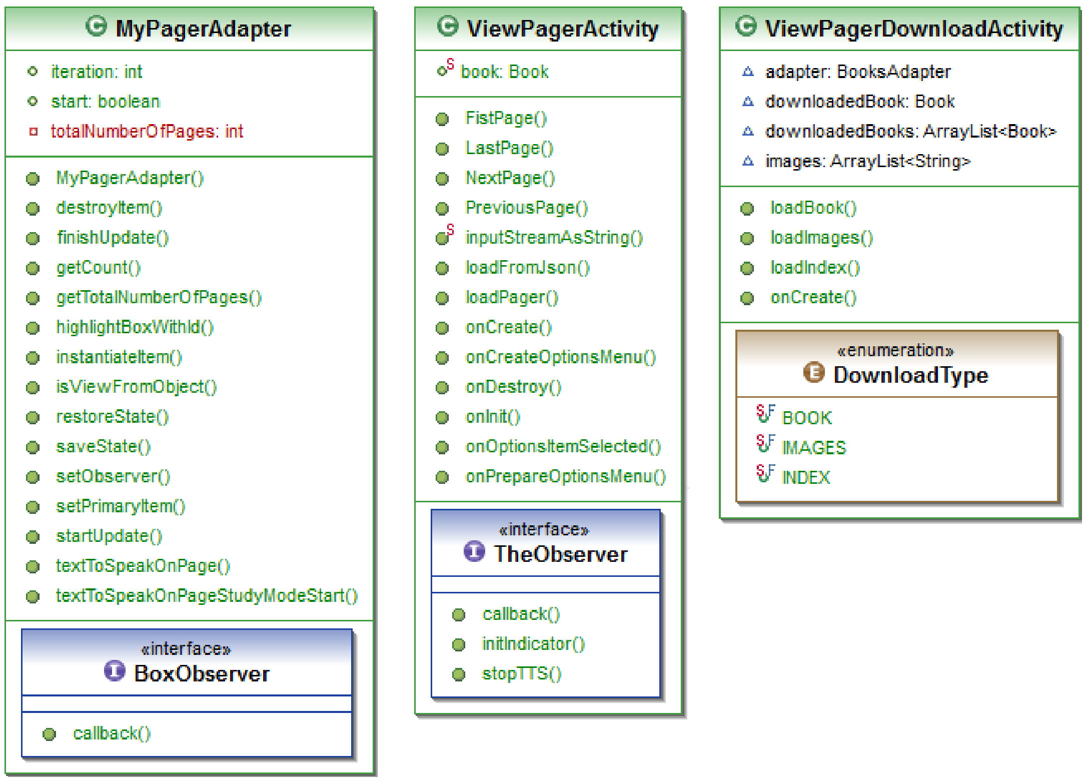


Figure 55 The“Tailored Book Reader” implemented classes: MyPagerAdapter, ViewPagerActivity, ViewPagerDownloadActivity

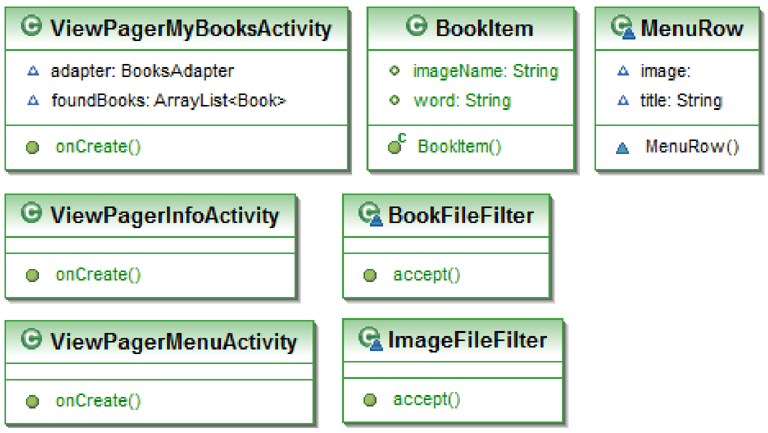


Figure 56 The “Tailored Book Reader”, implemented classes: ViewPagerMyBooksActivity, BookItem, ManuRow, ViewPagerInfoActivity, BookFileFilter, ViewPageMenuActivity, ImageFileFilter

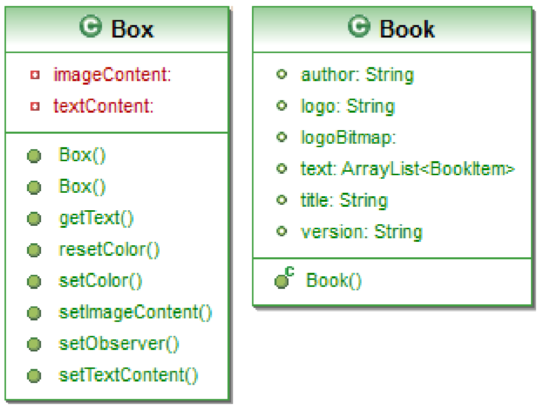


Figure 57 The “Tailored Book Reader”, implemented classes Box, Book

3.3. “Tailored Book Reader” screens

In the Figure 58 the main menu screenshot of the “Tailored Book Reader” mobile application is shown. From the main menu the user can go to the following screens: “Download Books” to see the list of books available on the server and download them; “My Books” to see the list of downloaded books; and “Info” to read about the application.

In the Figure 59 the “Download Books” screen is shown. In this screen you can see the list off books available for download. The user should click on the appropriate book to start downloading. When the book is downloaded it can be found in the “My Books” section accessible from the main menu.

In the Figure 60 a screenshot of the “My Books” screen of the “Tailored Book Reader” mobile application is shown. This list of my books contains both tailored books downloaded from the server and ones put on the SD card by the user.

Figure 61 shows the screenshot of the “Tailored Book Reader” “Info” screen. This screen is introduced in order to give some information about tailored books and this mobile application.

In the Figure 62 two screenshots of the “Book” screen are shown. On the left screenshot you can see the interface of the most important screen of this application. The “Book” screen displays the content of a tailored book. On the top part of the screen a title of the book is placed; on the bottom part there is a navigation bar. The middle part occupies the content of the book. On the right screenshot you can see the “Book” screen with the options menu, which contains two menu items: study and normal mode. The options menu panel can be revealed by pressing the Menu button.

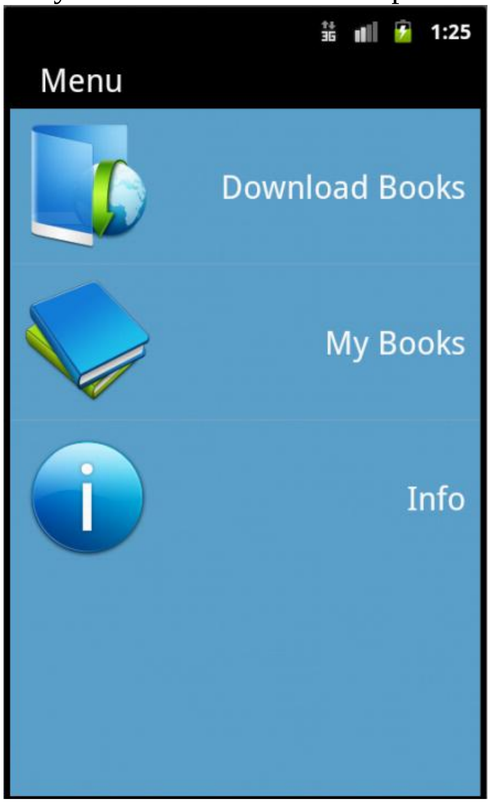


Figure 58 A screenshot of the “Tailored Book Reader” mobile application; “Menu” screen

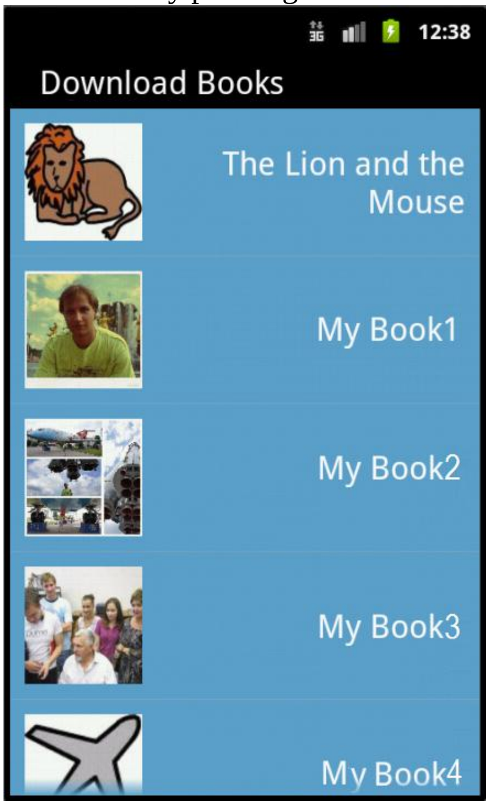


Figure 59 A screenshot of the “Tailored Book Reader” mobile application; “Downloaded Books” screen

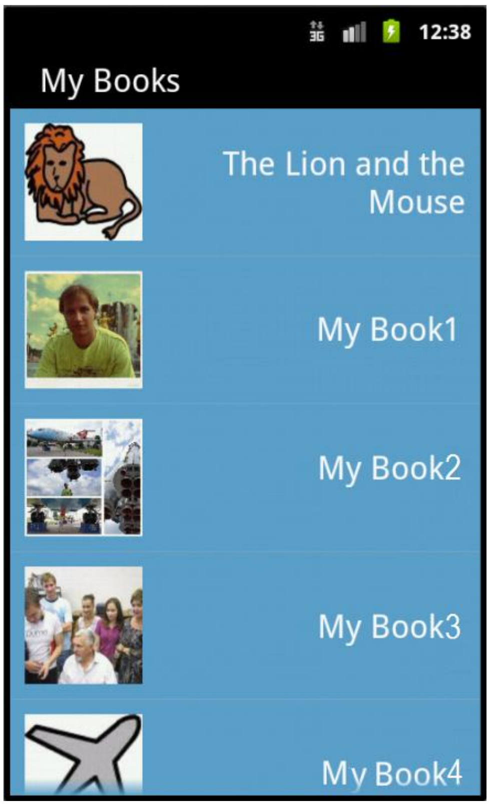
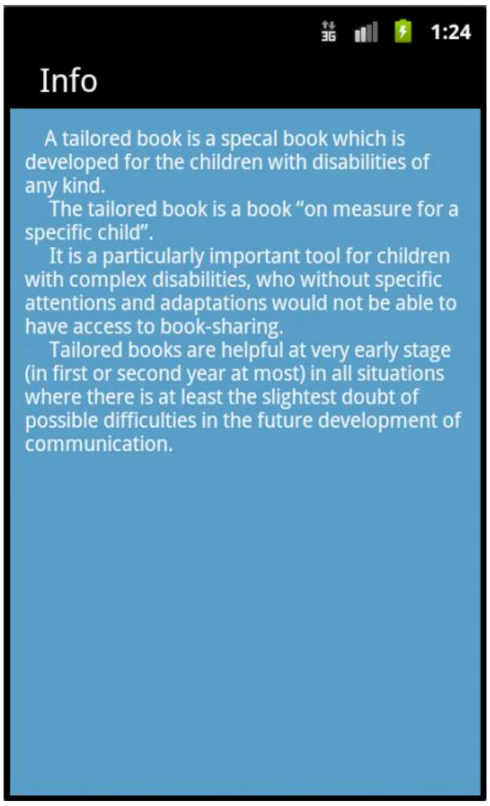
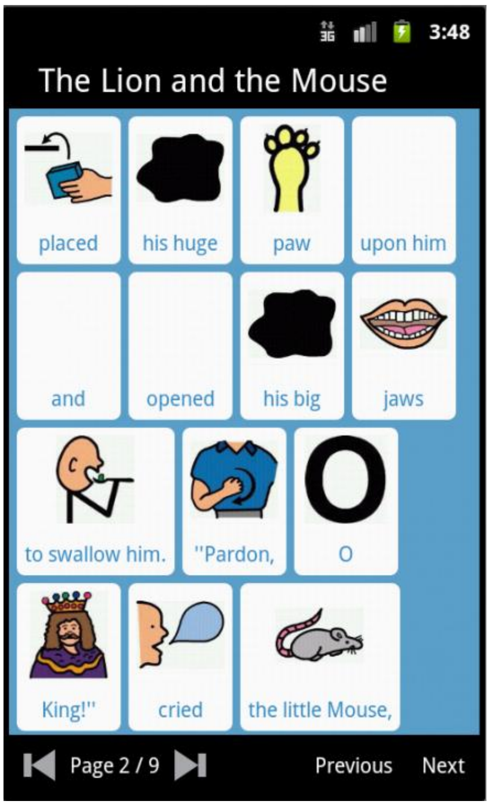


Figure 60 A screenshot of the "Tailored Book Reader" mobile application; "My Books" screen


Info

A tailored book is a specal book which is developed for the children with disabilities of any kind.
The tailored book is a book "on measure for a specific child".
It is a particularly important tool for children with complex disabilities, who without specific attentions and adaptations would not be able to have access to book-sharing.
Tailored books are helpful at very early stage (in first or second year at most) in all situations where there is at least the slightest doubt of possible difficulties in the future development of communication.

Figure 61 A screenshot of the "Tailored Book Reader" mobile application; "Info" screen



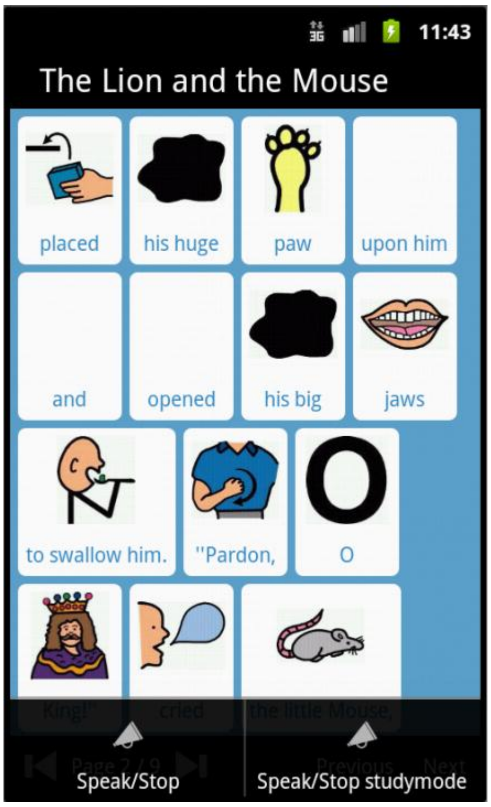


Figure 62 A screenshots of the "Tailored Book Reader" mobile application; "Book" screens

4. System usage workflow

As has been previously mentioned the designed system is composed of the desktop and mobile applications called “Tailored Book Editor” and “Tailored Book Reader” accordingly. Below on the Figure 63 the system usage workflow is shown for both applications correspondingly.

As you can see, in the first activity diagram that represents “Tailored Book Editor” after running the application the user can go two ways depending on the specified condition (Images are ready?).

If the images are available and tagged the user can go to the next paragraph, which describes the procedure of tailored book creation. Here instead I’d like to give some comments related to the procedure of image tagging. As has been said earlier in the project the necessary requirement for creation tailored book is the availability of images, because the book is in some sense the combination of text and images. Any images in format JPEG can be used. I’d like to stress it again that the system works only with this format since it supports writing and reading image metadata, in which the tags, assigned to the image, are stored. The “Tailored Book Editor” supports image tagging, for this reason the special module was developed. To start the image tagging the user should go to the main menu, placed on the top bar of the application and click File → New Image Tagger. After clicking, a new file picker is popped up. The file picker allows selecting one or several images. When the images are tagged using the “Image Tagger” the user can set the image source directory. From this directory the “Tailored Book Editor” takes images that then will be assigned to the items. In order to set the image source directory you should go to main menu item Project → Set image source directory. It is not obligatory to set it explicitly, because before the procedure of image assignment during tailored book creation, the system will ask to set the image source directory if it was not done beforehand.

After the image source directory is set the user should prepare the .txt file using standard text editing tool e.g. Notepad. This text file should contain the text of the future tailored book.

When the text file is ready the user can start creating the tailored book by clicking on main menu item File → New Book. After clicking the system will display the dialog with the following fields: book title, book author and book logo. Note in order to proceed to next step all required information should be inserted. The system should warn you if some fields are empty or id there is some unaccepted symbols. The system accepts numbers and digits. When there are no warnings in the input field, click the button “Create Book”. After clicking “Create Book” the system will launch file picker which should be used to locate the previously created .txt file. This file picker accepts only files with .txt extension. Then the system will prompt the picker to select the POS tagger if it has not set before. This file is provided with the application. After locating the file the system will start automatic creation of tailored book. When the book is created the user can start editing the tailored depending on his/her needs. When the book is ready, it can be published by clicking on main menu Project→Publish. The system will pop up the destination picker. The published book is composed of two files: the tailored book itself in the format .khl and the folder of supported images.

The second Activity diagram shows the procedure of tailored book setup for “Tailored Book Reader” application. Firstly the tailored book produced by the “Tailored Book Editor” can be loaded to the server or put directly to the SD card root directory.

On the server side there is an index file called books.khl. It can be seen from the URL: http://tbreader.is-great.net/BOOKS/books.khl

In order to make book downloadable from the server the two steps should be taken:

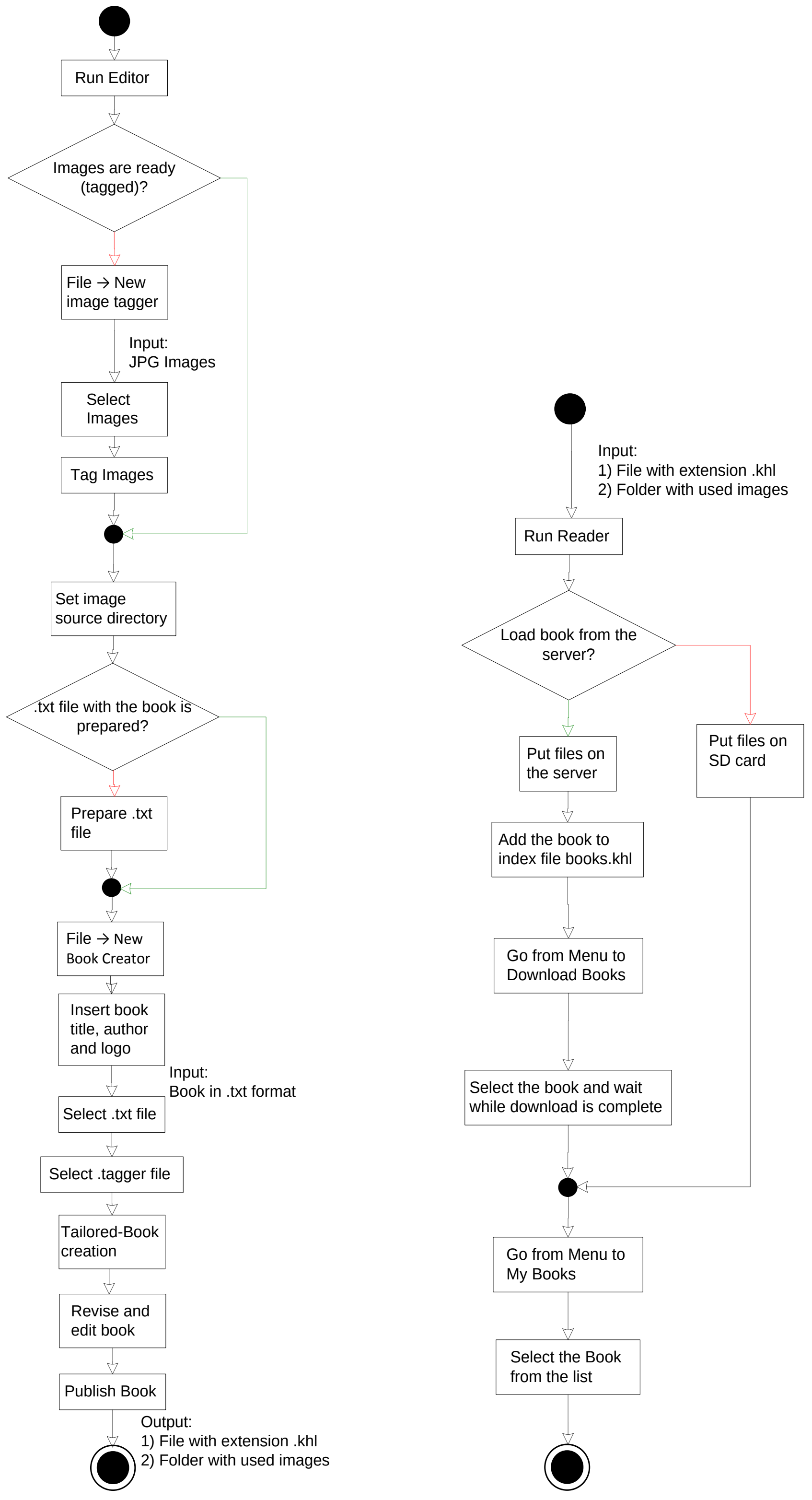
1) Download the book .khl with the folder of images on the server
2) Modify the books.khl file

The structure of the books.khl file was explained before in the Files Formats section.

In order to add a new tailored book to the books.khl file the content of the tailored book .khl file should be copied and pasted in the books.khl file after the existing book and then you should replace the content of “text” field with the square brackets.

When these steps are done, you should launch the "Tailored Book Reader" and from the main menu go to the "Download books" screen. The application will load the list of available books from the server. To download the book you should click on appropriate one. The application will load .khl file at first and then all the images. During download the progress dialog is shown. After successful download the book will be placed on the SD card.

To access the book from the SD card the user should go from main menu to "My Books" Screen. On this screen the list of all available on the SD card books are displayed. To start reading the book you should click on the appropriate book.



Run Editor
Images are ready (tagged)?
File → New image tagger
Input: JPG Images
Select Images
Tag Images
Set image source directory
.txt file with the book is prepared?
Prepare .txt file
File → New Book Creator
Insert book title, author and logo
Input: Book in .txt format
Select .txt file
Select .tagger file
Tailored-Book creation
Revise and edit book
Publish Book
Output: 1) File with extension .khl 2) Folder with used images
Input: 1) File with extension .khl 2) Folder with used images
Run Reader
Load book from the server?
Put files on the server
Put files on SD card
Add the book to index file books.khl
Go from Menu to Download Books
Select the book and wait while download is complete
Go from Menu to My Books
Select the Book from the list

Figure 63 System usage workflow

IV. Evaluation

In order to evaluate the users' satisfaction, the Goal Attainment Scale Method is adopted. Before explaining the basics of this method I'd like to tell a bit about the testing conditions. For the evaluation phase the two tailored books were created using the desktop application called "Tailored Book Editor" developed by me in this project. The books are called "Little Red Riding Hood" and "The Lion and the Mouse" respectively. These tailored books were placed on the server side. The mobile application "Tailored Book Reader" developed by me in this project was installed on the Samsung Galaxy S II device and given for testing to a girl and a boy at the age of 6 and 8 respectively.

Goal attainment scaling or GAS is a method of scoring the extent to which patient's individual goals are achieved in the course of intervention. In effect, each patient has their own outcome measure but this is scored in a standardized way as to allow statistical analysis.
Traditional standardized measures include a standard set of tasks (items) each rated on a standard level. In GAS, tasks are individually identified to suit the patient, and the levels are individually set around their current and expected levels of performance.

An important feature of GAS is the "a priori" establishment of criteria for a "successful" outcome in that individual, which is agreed with the patient and family before intervention starts so that everyone has a realistic expectation of what is likely to be achieved, and agrees that this would be worth striving for. Each goal is rated on a 5-point scale, with the degree of attainment captured for each goal area:

- If the patient achieves the expected level, this is scored at 0.
- If they achieve a better than expected outcome this is scored at: +1 (somewhat better) or +2 (much better)
- If they achieve a worse than expected outcome this is scored at: -1 (somewhat worse) or -2 (much worse)

Goals may be weighted to take account of the relative importance of the goal to the individual, and/or the anticipated difficulty of achieving it.

Overall Goal Attainment Scores are calculated by applying a formula:

$$GAS = 50 + \frac{10\sum w_i x_i}{\sqrt{(1-\rho)\sum w_i^2 + \rho(\sum w_i)^2}}$$

where:
w_i - the weight assigned to the ith goal (if equal weights, $w_i = 1$)
x_i - the numerical value achieved (between –2 and + 2)
ρ - the expected correlation of the goal scales

For practical purposes, according to Kirusek and Sherman, ρ most commonly approximates to 0.3, so the equation simplifies to:

$$GAS = 50 + \frac{10\sum(w_i x_i)}{\sqrt{0{,}7\sum w_i^2 + 0{,}3(\sum w_i)^2}}$$

In effect, therefore the composite GAS (the sum of the attainment levels x the relative weights for each goal) is transformed into a standardized measure or T score with a mean of 50 and standard deviation of 10. If goals are set in an unbiased fashion to that results exceed and fall short of expectations in roughly equal proportions, over a sufficiently large number of patients, one would expect a normal distribution of scores and the GAS thus performs at interval level.

Calculation of Goal Attainment Scoring

The GAS method was a tested on two children with the following characteristics: the smart and brilliant girl at the age of 6 that attends the English lessons at primary school; and the good 8 years old boy diagnosed with Asperger syndrome, attention problems, difficulties in reading and writing and without knowledge of English.

To each child the developed "Tailored Book Reader" mobile application was given, after some time they were interviewed to collect the results and experiences. The goals and attainment score levels were defined in the following table:

Goals:		Attainment score levels:				
		Most unfavorable treatment outcome thought likely	Less than expected success	Expected level of success	More than expected success	Best anticipated success
		-2	-1	0	1	2
1.	A child is able to navigate through the application	A child doesn't understand how to use the app and how to navigate within it	A child partly understands what app is needed for and how to navigate it	A child understands the app and knows how to use it	A child fully understands the app and knows how to use it	A child understands the complete structure of the app and knows where to go to get what he needs
2.	A child is able to select an appropriate book and launch it	A child is not able to select an appropriate book and launch it	A child is not able to choose a book without any help	A child is able to choose a book with some assistance	A child is able to choose a book himself	A child is able to choose books himself
3.	A child is able to download new books	A child is not able to download new books	A child is able to download new books but doesn't know how to launch it	A child is able to download new books and knows where it was saved to launch it	A child downloaded many books and launched them	A child is interested in downloading new books and reading them
4.	A child is able to turn over the pages using scroll mode	A child is unable to go through the book using scroll mode	A child cannot go to the next/previous page without the help	A child is able to page the book using scroll mode	A child is able to page the book using scroll mode without any help	A child pages the book easily using scroll mode and understands at what page he is and how to go to next page or go back

5.	A child is able to turn over the pages using next/previous buttons	A child is unable to go through the book using buttons	A child cannot go to the next/previous page without the help	A child is able to page the book using buttons	A child is able to page the book using buttons without any help	A child pages the book easily using the buttons and understands at what page he is and how to go to next page or go back
6.	A child is able to click on the item of the book and repeat aloud the word being pronounced	A child is not able to do it at all	A child is only able to select the word he wants to be pronounced	A child is able to select the word and repeat it within at least three attempts.	A child is very involved in selecting the words and repeating them.	A child is interested in this exercise and became very fluent in repeating the words.
7.	A child is able to turn on/off normal/study modes of reading the book	A child doesn't understand how to switch between two modes	A child changes the modes unwillingly and is not interested	A child understands how to switch between the modes and how to stop pronunciation of the words and can do it with some help	A child is able to switch between different modes without any help	A child is able to choose his favorite mode of reading the book and to switch easily between different modes
8.	A child follows the highlighted items of the book while being spoken aloud (study mode)	A child is not able to follow the words	It is very hard for a child to follow the words	A child is able to follow the words but sometimes he is lost or confused	A child can follow the words and seems very concentrated	A child follows the words with ease and he is very interested in an exercise
9.	A child is able to follow the words while being spoken aloud (normal mode)	A child is not able to follow the words	It is very hard for a child to follow the words	A child is able to follow the words but sometimes he is lost or confused	A child can follow the words and seems very concentrated	A child follows the words with ease and he is very interested in an exercise
10.	A child is able to read the book aloud and understand the content without it being spoken	A child is not able to read the book himself	It's very difficult for a child to read the book himself and understand it	A child is able to read the book aloud himself and partly understand the content	A child is able to read the book and understand the content completely	A child is able to read the book himself with interest without any assistance

Table 9 The goals and attainment score levels

In the first column of the table the goals are stated, the rest columns show attainment score levels achieved by the children. The expected level of success with the value equals to 0 is considered to be an expected outcome of the application usage.

A weight was assigned to each goal using the table below. Weight is the product of importance and difficulty. Importance and difficulty is rated on a 4 point scale:

Importance	Difficulty
0 - not at all (important)	0 - not at all (difficult)
1 - a little (important)	1 - a little (difficult)
2 - moderately (important)	2 - moderately (difficult)
3 - very (important)	3 - very (difficult)

Table 10 Importance and difficulty scale

If the goal is “not at all” important it will not be selected and if “not at all” difficult it has presumably already been achieved. If a weighting system is not used, a value of “1” is simply applied to weight in the Overall Goal Attainment Scores formula. In the table below the calculated weight is shown for each goal. The weighting and attainment score levels achieved by two children are shown in the table below.

Goals:	Importance	Difficulty	Weight	Attainment score levels	
				I child	II child
1	3	3	9	2	2
2	2	1	2	1	1
3	1	1	1	0	0
4	2	1	2	2	2
5	2	1	2	-1	-1
6	3	3	9	1	0
7	2	1	2	-1	-1
8	3	2	6	0	0
9	2	3	6	-1	-2
10	1	3	3	-1	-2

Table 11 The weighting and attainment score levels achieved by two children

I child Overall Goal Attainment Scores:

$$GAS = 50 + \frac{10\sum(w_i x_i)}{\sqrt{0{,}7\sum w_i^2 + 0{,}3(\sum w_i)^2}}$$
$$= 50 + \frac{10(18+2+0+4-2+9-2+0-6-3)}{\sqrt{0.7(81+4+1+4+4+81+4+36+36+9)+0{,}3\cdot 1764}}$$
$$= 50 + \frac{10\cdot 20}{26.7} = 50 + 7{,}5 = 57{,}5$$

II child Overall Goal Attainment Scores:

$$GAS = 50 + \frac{10\sum(w_i x_i)}{\sqrt{0{,}7\sum w_i^2 + 0{,}3(\sum w_i)^2}}$$
$$= 50 + \frac{10(18+2+0+4-2+0-2+0-12-6)}{26{,}7} = 50 + \frac{10\cdot 2}{26{,}7}$$
$$= 50 + 0{,}75 = 50.8$$

The overall data can be considered positive because all the children reached the maximum goal attainment level (+2) for two goals and the positive goal attainment levels (0, +1, +2) for 6 of 10 goals. The overall Goal Attainment Scores for the first and the second child are 57.5 and 50.8 respectively.

The range of GAS scores calculated using the expected correlation of the goal scales (ρ) approximated to 0.3 goes from 18.54 to 81.46. The low bound is calculated assuming that for all the goals a child scores the most unfavorable goal attainment levels (-2). The upper bound is calculated assuming that for all the goals a child scores the most fortunate goal attainment levels (+2). When the attainment score levels for all goals are assumed to be zero we get the expected results equals 50. Thereby for these two children, GAS confirms better than expected result.

Also from the provided feedback we found out that the children have a poor interest in reading books in English and they want books in Italian presumably due to a weak level of English language skills.

The second reason for the lack of strong interest is supposedly limited physical access to the tested application “Tailored Book Reader”. This may be because they had very little training. In our experiment the application was installed on a single device that was given to the children for the limited time.

As well the children were not very interested in reading due to competing interests around. The latest supposition is based on the fact that at the time of testing the multiple distracting factors were activated such as TV, computer games, board games, music, etc., which favors the creation of the situation mostly close to the reality.

Since the small Italian users on which the system was tested have a strong will to read books in Italian the support of the Italian language will be added in the future release.

I would like to draw attention to the fact that all the materials used for the test has been created by me using the desktop and mobile applications (“Tailored Book Editor” and “Tailored Book Reader” respectively) designed and implemented in this project; which, in turn, based on the synergy effect increases the absolute value of the obtained results.

V. Conclusions and Future Work

The diffusion of mobile technologies has greatly changed the state of affairs for individuals with AAC. Touch screen smartphones, tablets, powerful mobile processors, reasonable price and user-friendly interfaces are now widely available to all sectors of society. Mobile technologies are also being readily adopted by people with disabilities and there is no doubt that technology has the potential to free many children from their disability in a way that allows them to achieve their true potential.

In this project a system for the creation, modification, visualization and distribution of tailored books was designed and implemented. This system helps to improve reading skills and development of communication for children with special needs, who are not able to read the standard books without its specific modification, adaptation and interactive presentation.

As a result the following objectives have been achieved:

1. Design and implementation of a specialized desktop application to simplify the process of creating, editing, and distribution of tailored books.
2. Design and implementation of an interactive mobile application to visualize tailored books on smartphones and tablets with OS Android.

The desktop application “Tailored Book Editor” designed and implemented by me in this project helps to achieve the objective number one. The application “Tailored Book Editor” has the following characteristics:

- ✓ loads text from .txt file and construct the tailored book automatically using WordNet database
- ✓ assigns automatically the images to the words if there is a match of tags
- ✓ allows assigning manually the image to the word
- ✓ allows editing the generated tailored book (e.g. join or split items, change tags of book items or images to achieve appropriate result)
- ✓ allows tagging images with reference to WordNet database
- ✓ allows saving and opening the tailored book for further editing
- ✓ allows publishing the book to the selected directory on the hard disk
- ✓ synchronization between tabs of books and images(e.g. if image tag is changed the image assigned to the book item is updated)
- ✓ remembers user preferences (e.g. last selected path, image source directory, .tagger file)

The mobile application "Tailored Book Reader" designed and implemented by me in this project helps to achieve the goal number two. The application "Tailored Book Reader" has the following characteristics:

- ✓ reads books from SD card
- ✓ downloads books (file with extension .khl and folder with images) from the server
- ✓ pronounce words aloud on click with TTS
- ✓ read the whole page with TTS (normal mode)
- ✓ highlighting the word while being spoken (study mode)
- ✓ navigation: first/last page button, next/previous button, next/previous page with a scroll gesture

As all systems have some limitations the developed system is not an exception. Below I'd like to mention the directions of further improvements.

For "Tailored Book Editor" the essential extension will be a module for creating tailored books in Italian, Russian and German languages. Another main concept is to create a community between people who produce tailored books and people who want to read them. For this reason the module which allows automatic publication of tailored books on the server to share with others should be introduced. On the server part the system that manages all the books uploaded by the users and allows searching an appropriate book should be designed.
Another important improvement for "Tailored Book Editor" is to add the possibility to set the custom options before tailored book creation, e.g. to allow the users to decide if they want to have prepositions, articles, pronouns and etc. to be together in one figure with other parts of speech or to be separated in two figures.

For "Tailored Book Reader" I'd like to offer the extra functionality to the user. For example it's worth adding a possibility to set the number of figures to be displayed on the screen. So far the number of displayed figures is calculated depending on the dimensions of the screen. Also it would be useful to add a possibility to set the size of the figure and the font. Also as a complement to the future community a possibility to rank the book (from 1 to 5 stars) in the section «My Books» and «Download Books» should be added. The vote will be used to calculate the mean rank for the book on the server. Also user should be able to share books stored on the SD card with others. Below I list some features for organisation of books I plan to add in the future release for "Tailored Book Reader" application:

- ✓ sorting books both in the «My Books» and «Download Books» sections
- ✓ possibility to organize the books by drag and drop in the section «My Books»
- ✓ possibility to mark the book as favourite in the «My Books» section; all the favourite books will be placed on the top of the list

- ✓ possibility to remove book from the section «My Books»

The developed system can also be used outside of the ACC context for example for learning foreign languages.

VI Methods description

1. “Tailored Book Editor” methods

class Editor extends JFrame

Method	Description
public String [] getTags(String filePath)	Returns a string array of tags obtained from the image, using Sanselan library for reading image metadata and JDOM library for accessing and parsing XML data.
public void initializeSuggestionList()	Initializes suggestion list of similar tags for Image Tagger
public void updatePartOfSpeechPanel(String[] pos)	Updates panel with suggestion list when the part of speech control (radio button) is changed
public Editor()	Constructs a new frame that is initially invisible
public ArrayList<Item> addPunctuation(ArrayList<Item> items)	Returns ArrayList of items which is usually smaller than the initial ArrayList if it contains punctuation symbols. Idea: If the item tag of the initial array of items is punctuation symbol, its value is added to previous item and then the item is removed from the ArrayList.
private void loadGrid(ArrayList<MyImage> newImageList, final JGrid grid)	Initialization of the grid for Image Tagger
private ArrayList<MyImage> chooseDirectory()	Image picker which allows choosing images for Image Tagger. This method create instance of JFileChooser with the filter ImageFilter and returns the list of selected image paths.
private String chooseTxtFile(FileFilter filter)	Txt file picker for choosing book text for further processing.
private void saveBook()	Shows save dialog which assist to save Tailored Book which can be opened then by Editor.
private BookSaveFormat createBookSaveFormat()	Converts Tailored Books to special format for further saving on the hard disk
private Book createBook()	Shows book creation dialogs and then generates Tailored Book.
private void gatherImages(JFileChooser fc, Book book)	Outputs all the images used in Tailored Book in the separate folder for further export in Reader
private void publishBook()	Outputs Tailored Book with the folder of images.
private ArrayList<MyImage> setImageSourceDirectory(boolean setDefault)	Returns ArrayList of MyImage instances which is created from all the images under selected directory
public void assignment(ArrayList<Item> itemsList)	Assigns image to the item from ArrayList if there is tags intersection
public void updateGridListAndModelList(String	Synchronization between Image Tagger and

name, MyImage im)	Book Editor
public ArrayList<Item> processTextFile(String tagged, boolean postfixSufixProcess)	Part of Speech Tagging and finding synonyms in WordNet database. Returns array of items ready for visualization in the grid
private void loadBookGrid(ArrayList<Item> newItemList, final JGrid grid)	Initialization of the grid for Book Editor

Table 12 The "Tailored Book Editor", methods description of Editor class

class RadioButton extends JRadioButton

public String getPartOfSpeech()	Returns part of speech.
public void setPartOfSpeech(String partOfSpeech)	Sets part of speech.
public RadioButton(String string, boolean b, String partOfSpeech)	Constructs radio button.

Table 13 The "Tailored Book Editor", methods description of RadioButton class

public class Book

public Book()	Create Tailored Book without title, author, text and logo.
public Book(String title, String author, ArrayList<BookItem> text, String logo)	Constructs Tailored Book with title, author, text and logo.

Table 14 The "Tailored Book Editor", methods description of Book class

public class BookDetails extends JDialog

public BookDetails()	Constructs book details Dialog which is used to insert title, author and select book logo.
private ImageIcon normalizeImage(Image src)	Change size of the image in order to fit allotted space.

Table 15 The "Tailored Book Editor", methods description of BookDetails class

public class BookFileFilter extends FileFilter

public boolean accept(File pathname)	Whether the given file is accepted by this filter. Accept files with format .bookkhl
public String getDescription()	The description of this filter.

Table 16 The "Tailored Book Editor", methods description of BookFileFilter class

public class BookItem

public BookItem(String word, String imageName)	Constructs book item. Book items represent text of Tailored Book.

Table 17 The "Tailored Book Editor", methods description of BookItem class

public class BookSaveFormat extends Book

public BookSaveFormat(String title, String author, ArrayList<Item> text, String logo)	Constructs the book in the format the suitable for saving and opening the book later with the Editor.

Table 18 The "Tailored Book Editor", methods description of BookSaveFormat class

public class ButtonTabComponent extends JPanel

public ButtonTabComponent(boolean bookTab, int index)	Constructs custom tab at position index
public ButtonTabComponent()	Constructs custom tab

Table 19 The "Tailored Book Editor", methods description of ButtonTabComponent class

private class TabButton extends JButton implements ActionListener

public TabButton()	Creates a tab button.
protected void paintComponent(Graphics g)	Paint the cross on the tab.

Table 20 The “Tailored Book Editor”, methods description of TabButton class

public class ComboBoxRenderer extends JTextArea implements ListCellRenderer<Object>

public ComboBoxRenderer()	Constructs ComboBoxRenderer
public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean isSelected, boolean cellHasFocus)	Return a component that has been configured to display the specified value.

Table 21 The “Tailored Book Editor”, methods description of ComboBoxRenderer class

public class DirectoryReader

public DirectoryReader(File aFile)	Construct DirectoryReader
private void Process(File aFile)	Recursively examine all image files and add it to ArrayList
public ArrayList<File> getAllFiles()	Returns all image files as ArrayList

Table 22 The “Tailored Book Editor”, methods description of DirectoryReader class

public class ImageFilter extends FileFilter

public boolean accept(File pathname)	Whether the given file is accepted by this filter. Accept files with format .jpg
public String getDescription()	The description of this filter.

Table 23 The “Tailored Book Editor”, methods description of ImageFilter class

public class Item

public Item(String[] tags, String word, String type)	Constructs Item with tags, word and part of speech type.
public Item()	Constructs empty Item.
public String[] getTags()	Returns array of tags.
public void setTags(String[] tags)	Sets array of tags.
public String getWord()	Returns word of the Item.
public void setWord(String word)	Sets word of the Item.
public String getType()	Returns part of speech type.
public void setType(String type)	Sets part of speech type.
public void assignImage(ArrayList<MyImage> images)	Assigns image to Item
public static Set<String> getIntersection (Set<String> x, Set<String> y)	Returns intersection of two sets
public String getImageName()	Returns assigned image name
public void setImageName(String imageName)	Sets image name
public void setImagePath(String imagePath)	Sets image path
public String getImagePath()	Returns image path

Table 24 The “Tailored Book Editor”, methods description of Item class

public class MyFilter implements FileFilter

public boolean accept(File pathname)	Whether the given file is accepted by this filter. Accept files with format .jpg

Table 25 The “Tailored Book Editor”, methods description of MyFilter class

public class MyImage

public MyImage(String name, String[] tags, String path)	Constructs MyImage with name, tags and path.
public String[] getTags()	Returns tags.
public void setTags(String[] tags)	Set Tags.
public String getName()	Returns name.
public void setName(String name)	Sets name.
public void loadImage()	Loads image as BufferedImage.
private byte[] imageToByteData(BufferedImage image)	Transforms BufferedImage to bytes.
public String getPath()	Returns image path
public void setPath(String path)	Sets image path
public String getImage()	Returns image in bytes
public String getExtension()	Returns image extension

Table 26 The "Tailored Book Editor", methods description of MyImage class

public class PicViewerRenderer extends JComponent implements GridCellRenderer

public PicViewerRenderer(ArrayList<MyImage> allImages)	Constructs PicViewerRenderer with all images
public PicViewerRenderer()	Constructs PicViewerRenderer
public Component getGridCellRendererComponent(JGrid grid, Object value, int index, boolean isSelected, boolean cellHasFocus)	Initialize grid cell with appropriate text and image
protected void paintComponent(Graphics g)	Draw grid cell

Table 27 The "Tailored Book Editor", methods description of PicViewerRenderer class

public class Tagger

public Tagger(String path)	Constructs Tagger with tagger path
public String tagString(String st)	Returns POS tagged string

Table 28 The "Tailored Book Editor", methods description of Tagger class

public class TaggerFilter extends FileFilter

public boolean accept(File pathname)	Whether the given file is accepted by this filter. Accept files with format .tagger
public String getDescription()	The description of this filter.

Table 29 The "Tailored Book Editor", methods description of TaggerFilter class

public class TextFileFilter extends FileFilte

public boolean accept(File pathname)	Whether the given file is accepted by this filter. Accept files with format .txt
public String getDescription()	The description of this filter.

Table 30 The "Tailored Book Editor", methods description of TextFileFilter class

2. "Tailored Book Reader" methods

public class Book

public Book()	Create Tailored Book without title, author, text and logo.
public Book(String title, String author, ArrayList<BookItem> text, String logo)	Constructs Tailored Book with title, author, text and logo.

Table 31 The "Tailored Book Reader", methods description of Book class

public class BookItem

public BookItem(String word, String imageName)	Constructs book item. Book items represent text of Tailored Book.

Table 32 The “Tailored Book Reader”, methods description of BookItem class

public class BooksAdapter extends ArrayAdapter<Book>

public BooksAdapter(Context context, int layoutResourceId, ArrayList<Book> downloadedBooks)	Constructs BooksAdapter which is used for list of books visualization
public View getView(int position, View convertView, ViewGroup parent)	Returns view of every book with title and logo

Table 33 The “Tailored Book Reader”, methods description of BooksAdapter class

public class Box extends LinearLayout

public Box(Context context)	Constructs Box
public Box(Context context, AttributeSet attrs)	Constructs Box with atributes
private void setupViewItems()	Initializes instances of ImageButton and TextView
public void setObserver(BoxObserver boxObserver)	Sets BoxObserver
public void resetColor()	Sets box color to white color
public void setColor()	Highlights box with green color
public void setTextContent(String text)	Set text of the box
public String getText()	Returns text of the box to play TTS
public void setImageContent(String bookTitle,String name)	Sets image of the box

Table 34 The “Tailored Book Reader”, methods description of Box class

class MenuRow

MenuRow(String title, Drawable image)	Constructs menu row with text and icon

Table 35 The “Tailored Book Reader”, methods description of MenuRow class

public class MenuAdapter extends ArrayAdapter<MenuRow>

public MenuAdapter(Context context, int layoutResourceId, ArrayList<MenuRow> menuRows)	Constructs MenuAdapter which is used for main menu visualization.
public View getView(int position, View convertView, ViewGroup parent)	Returns view of every menu item with title and logo

Table 36 The “Tailored Book Reader”, methods description of MenuAdapter class

public class MyPagerAdapter extends PagerAdapter

public MyPagerAdapter(Activity activity, Book book)	Constructs MyPagerAdapter with book
public int getTotalNumberOfPages()	Returns number of pages for book visualization. Number of pages depends on device screen size.
public String textToSpeakOnPage(int page)	Returns string to play TTS. The string is formed by concatenation of all words from

	boxes on the current page.
public void textToSpeakOnPageStudyModeStart(boolean value)	Starts to play TTS in study mode by highlighting every box from which the word is being pronounced
private int dpToPx(int dp)	Converts density-independent pixels to actual pixels on the screen
public void setObserver(TheObserver observer)	Sets observer TheObserver
public void highlightBoxWithId(int id, int curentPage)	Highlights box with the green color while playing TTS in study mode or when the box is pressed.
public Object instantiateItem(View collection, int position)	Creates and adds new box item to the layout. The length of the item depends on the length of the word in the box.
public void setPrimaryItem(ViewGroup container, int position, Object object)	Called to inform the adapter of which item is currently considered to be the "primary", that is the one show to the user as the current page.
public void destroyItem(View arg0, int arg1, Object arg2)	Remove a page for the given position. The adapter is responsible for removing the view from its container, although it only must ensure this is done by the time it returns from finishUpdate(ViewGroup).
public void finishUpdate(View arg0)	Called when a change in the shown pages has been completed.
public boolean isViewFromObject(View arg0, Object arg1)	Determines whether a page View is associated with a specific key object as returned by instantiateItem(ViewGroup, int). This method is required for a MyPagerAdapter to function properly.
public void restoreState(Parcelable arg0, ClassLoader arg1)	Restore any instance state associated with this adapter and its pages that was previously saved by saveState().
public Parcelable saveState()	Save any instance state associated with this adapter and its pages that should be restored if the current UI state needs to be reconstructed.
public void startUpdate(View arg0)	Called when a change in the shown pages is going to start

	being made.

Table 37 The “Tailored Book Reader”, methods description of MyPagerAdapter class

public interface BoxObserver

void callback(String word)	Calls the method callback(String word, int id) of the instance of class TheObserver to sent text to speak.

Table 38 The “Tailored Book Reader”, methods description of BoxObserver interface

public interface TheObserver

void callback(String word, int id)	Calls method speakOut to pronounce the word of the box with id.
void stopTTS()	Terminates TTS.
void initIndicator()	Initializes page indicator.

Table 39 The “Tailored Book Reader”,   methods description of TheObserver interface

public class ViewPagerActivity extends Activity implements TextToSpeech.OnInitListener

public Book loadFromJson(Activity activity)	Returns Book item from Json
public void onCreate(Bundle savedInstanceState)	Called when the activity is first created. View creation, header and footer setup.
protected void onPostCreate(Bundle savedInstanceState)	Called when activity start-up is complete (after onStart() and onRestoreInstanceState(Bundle) have been called).
protected void onStart()	Called when the activity is becoming visible to the user. MyPagerAdapter and TTS initialization.
public void loadPager(Book book)	Creates new instance of MyPagerAdapter.
protected void speakOut(String word, int id)	Pronounce the word of the box.
public void onInit(int status)	OnUtteranceCompletedListener is implemented which is used to know when the TTS finishes pronunciation of the word.
public void onDestroy()	Shutdowns TTS
public boolean onCreateOptionsMenu(Menu menu)	Initialize the contents of the Activity's standard options menu. Menu options: speak in normal mode and speak in study mode.
public boolean onPrepareOptionsMenu (Menu menu)	Prepare the Screen's standard options menu to be displayed. This is called right before the menu is shown, every time it is shown. This method is used to enable/disable items. For example when the study mode is on, normal mode is disabled until study mode is stoped.
public boolean onOptionsItemSelected(MenuItem item)	This hook is called whenever an

	item in your options menu is selected. Activates one of two modes study or normal.
public void NextPage()	Loads next page of the book
public void PreviousPage()	Loads previous page of the book
public void FistPage()	Loads first page of the book
public void LastPage()	Loads last page of the book

Table 40 The “Tailored Book Reader”, methods description of ViewPagerActivity class

public class ViewPagerDownloadActivity extends Activity

public void loadIndex()	Loads index of books available on the server side and displays it.
public void loadBook(String bookTitle)	Load book from the server
public void loadImages(ArrayList<String> images)	Load all images associates with the book.
public void onCreate(Bundle savedInstanceState)	View creation, header and footer setup.

Table 41 The “Tailored Book Reader”, methods description of ViewPagerDownloadActivity class

class ImageFileFilter implements FileFilter

public boolean accept(File pathname)	Whether the given file is accepted by this filter. Accept files with format .jpg

Table 42 The “Tailored Book Reader”, methods description of ImageFileFilter class

private class AsyncDownloadBook extends AsyncTask<Void, Void, Boolean>

private AsyncDownloadBook(DownloadType downloadType, String urlString)	Creates a new asynchronous task.
private String getBook(String bookUrl)	Returns book json string downloaded from the server.
private Bitmap getBitmap(String bitmapUrl)	Downloads logo from the server.
protected void onPreExecute()	Initialization of ProgressDialog
protected Boolean doInBackground(Void... params)	Perform background book downloading and saving on SD card.
protected void onPostExecute(Boolean result)	Creates ArrayList of images found in the book including logo for further downloading.

Table 43 The “Tailored Book Reader”, methods description of AsyncDownloadBook class

private class AsyncDownloadImages extends AsyncTask<Void, Integer, Boolean>

private Bitmap getBitmap(String bitmapUrl)	Downloads image from the server.
protected Boolean doInBackground(Void... arg0)	Perform background downloading of image and saving it on SD card.
protected void onPreExecute()	Initialization of ProgressDialog
protected void onPostExecute(Boolean result)	Dismiss ProgressDialog.
protected void onProgressUpdate(Integer... values)	Updates ProgressDialog by incrementing number of downloaded images.

Table 44 The “Tailored Book Reader”, methods description of AsyncDownloadImages class

public class ViewPagerInfoActivity extends Activity

public void onCreate(Bundle savedInstanceState)	Info activity is starting.

Table 45 The “Tailored Book Reader”, methods description of ViewPagerInfoActivity class

public class ViewPagerMenuActivity extends Activity

public void onCreate(Bundle savedInstanceState)	Main menu activity is starting.

Table 46 The “Tailored Book Reader”, methods description of ViewPagerMenuActivity class

private class MyAsyncTask extends AsyncTask<Void, Void, Boolean>

private MyAsyncTask(Class<?> activityClass)	Creates a new asynchronous task
protected void onPreExecute()	Initialization of ProgressDialog
protected void onPostExecute(Boolean result)	Dismiss ProgressDialog.
protected Boolean doInBackground(Void... params)	Loads activity which was selected. Ex: ViewPagerMyBooksActivity, ViewPagerDownloadActivity, ViewPagerInfoActivity

Table 47 The “Tailored Book Reader”, methods description of MyAsyncTask class

public class ViewPagerMyBooksActivity extends Activity

public void onCreate(Bundle savedInstanceState)	MyBooks activity is starting. Loading list of books from SD card.

Table 48 The “Tailored Book Reader”, methods description of ViewPagerMyBooksActivity class

class BookFileFilter implements FileFilter

public boolean accept(File pathname)	Whether the given file is accepted by this filter. Accept files with format .khl

Table 49 The “Tailored Book Reader”, methods description of BookFileFilter class

VII. Figures and tables index

VIII. Bibliography

1. Augmentative and Alternative Communication in the Early Childhood Years, July 2007
2. L’intervento di Comunicazione Aumentativa e Alternativa (CAA) in età evolutiva Quaderni acp 2007
3. AAC Augmentative and Alternative Communication resource guide for teachers.
4. Office of Special Education Programs. To assure the free appropriate public education of all children with disabilities: Twenty-first annual report to Congress on the implementation of the Individuals with Disabilities Education Act, 1999. Washington, DC: OSEP, U.S. Department of Education, April 10, 2000.
5. Matas JA, Mathy-Laikko P, Beukelman DR,Legresley K. Identifying the nonspeaking population: a demografic study. Augmentative and Alternative Communication 1985;
6. Ministero dell’Istruzione, dell’Università e della Ricerca; Direzione Generale per i Sistemi Informativi. Sedi, alunni, classi, dotazioni organi-che del personale della scuola statale, AS 2005/2006.
7. Children with Apraxia and Reading, Writing, and Spelling Difficulties by Joy Stackhouse, Ph.D. 2008
8. Beukelman DR, Mirenda P. Augmentative and Alternative Communication: management of severe communication disorders in children and adults (2nd ed). Baltimore: Paul H Brookes, 1998.
9. Schlosser RW. The efficacy of augmentative and alternative communication: Toward evidence based practice. New York: Academic Press, 2003
10. Costantino MA, Marini M, Bergamaschi E, Lanzini L. Dal “libro su misura” alla “biblioteca di tutti”. Quaderni acp, 2006;
11. L’intervento di Comunicazione Aumentativa e Alternativa (CAA) in età evolutiva, 2007
12. Piacevoli esperienze di letture al bambino. Centro sovrazonale di comunicazione aumentativa Milano – Verdello
13. Libri “su misura” e “Nati per Leggere”, 2009
14. Cochran, P.S. Technology for individuals with speech and language disorders. In Technology and exceptional individuals. 3rd ed. J.D. Lindsay, ed. Austin, TX: Pro-Ed, 2000
15. Applying Technology to Visually Support Language and Communication in Individuals with Autism Spectrum Disorders 2011
16. White, B.Y., and Fredrickson, J.R. Inquiry, modeling, and metacognition: Making science accessible to all students. Cognition and Instruction,1998
17. Mack, C.G., Koenig, A.J., and Ashcroft, S.C. Microcomputers and access technology in programs for teachers of visually impaired students. Journal of Visual Impairments and Blindness, 1990 Impairment. London: Whurr; 1999.
18. Fox L, Sohlberg M.M, Fried-Oken M. Effects of conversational topic choice on outcomes of augmentative communication intervention for adults with aphasia. Aphasiology. 2001;

19. Lasker J, Bedrosian J. Promoting acceptance of augmentative and alternative communication by adults with acquired communication disorders. Augmentative Altern Commun. 2001;
20. Parr S, Byng S, Gilpin S. Talking About Aphasia. Bristol, PA: Open University Press; 1997.
21. Fox L, Sohlberg M. Meaningful communication roles. In: Beukelman D, Yorkson K, Reichle J, eds. Augmentative Communication for Adults with Neurogenic and Neuromuscular Disabilities. Baltimore: Paul H. Brookes; 2000
22. Wikipedia, Part-of-speech tagging, http://en.wikipedia.org/wiki/Part-of-speech_tagging
23. Mobile devices & communication apps An AAC-RERC White Paper
24. Java Reference Library, By David Flanagan, O'Reilly, 1997
25. Java I/O, Elliotte Rusty Harold, O'Reilly, 1999
26. Java Swing, Robert Eckstein, Mare Loy and Dace Wood, O'Reilly & Associates, 1998
27. Android UI Fundamentals Develop and Design, Jason Ostrander
28. Learn Java for Android Development, Jeff Friesen, 2010
29. Android Application Development, Lauren Darcey, Shane Conder, 2010
30. Head First Android Development, Jonathan Simon, 2011
31. Android Application Development, Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike, 2009, O'Reilly Media
32. Goal Attainment Scaling (GAS) in Rehabilitation. A practical guide. Herbert Dunhill Chair of Rehabilitation, King's College London.
33. Autonomamente: Using Goal Attainment Scales to Evaluate the Impact of a Multimodal Domotic System to Support Autonomous Life of People with Cognitive Impairment, Thimoty Barbieri, Piero Fraternali, Antonio Bianchi, and Clarissa Tacchella, 2010
34. Goal Attainment Scaling: A General Method for Evaluating Comprehensive Community Mental Health Programs, Thomas J. Kiresuk, Ph.D. Robert E. Sherman, M.S., 1968
35. Princeton University, What is WordNet? http://wordnet.princeton.edu/
36. Wikipedia, Wordnet, http://en.wikipedia.org/wiki/WordNet
37. Yves Lafon, Bert Bos, Describing and retrieving photos using RDF and HTTP, W3C Note 19 April 2002
38. JGrid - Grid Component for Java Swing http://code.google.com/p/jgrid/
39. Piacevoli esperienze di letture al bambino, riassunto. Centro sovrazonale di comunicazione aumentativa, Milano Verdello
40. AWT vs Swing, Josh Fletcher, 2009
41. Google-gson, A Java library to convert JSON to Java objects and vice-versa http://code.google.com/p/google-gson/
42. Semantic Search, Wijekoon, University of Moratuwa, 2008 http://asanga86.wordpress.com/2008/08/
43. Extensible Metadata Platform (XMP) http://www.adobe.com/products/xmp/overview.html
44. Practical resource description framework (rdf), Etutorials.org http://etutorials.org/Misc/Practical+resource+description+framework+rdf/