future internet

MDPI

Article

A Model for Creating Interactive eBooks for eLearning

Antonio Sarasa-Cabezuelo

Department of Computer Systems and Computing, School of Computer Science, Complutensian University of Madrid, 28040 Madrid, Spain; asarasa@ucm.es

Received: 24 October 2020; Accepted: 2 December 2020; Published: 7 December 2020

Abstract: In recent decades, electronic books have revolutionized the publishing world. In this sense, an area of application is education, where electronic books can be used as educational resources to implement learning strategies about content and in eLearning environments. For this, it is necessary to introduce interactive elements in the electronic books that turn the reader into an active actor in the reading process. However, ebooks have a limitation regarding their creation process. In this sense, the tools can be user-oriented or programmer-oriented. The former are intuitive to use and have user-friendly interfaces, but they offer a reduced number of functionalities to add to books. The second are aimed at programmers, allowing for the implementation of any functionality, but limiting the number of content creators who can use them. The main motivation of this work is to propose an intermediate solution that offers a wide number of functionalities while not requiring deep programming knowledge to use them. In this sense, the solution of this article is novel since it proposes the use of extensible markup language (XML) documents to specify the structure of the electronic book in such a way that its processing will lead to the electronic book.

Keywords: eLearning; ebooks; XML; language-driven development

1. Introduction

Electronic books have allowed the digitization of books and have revolutionized the field of publishing. These resources offer many advantages to the reader with respect to the physical book, such as the possibility of having a huge number of digital books in an electronic reading device, a lower price with respect to the physical book, and other possibilities with respect to the process of reading, such as the possibility of looking up words in online dictionaries, electronic annotations, navigation by levels, etc. However, there are other areas that have been influenced by electronic books, such as the educational world. An ebook can be used as an educational resource in which a learning strategy can be implemented. For this, it is essential to add interactive resources to the electronic book that transform the reader of the book into an active agent in the reading process. The main problem that electronic books present is their creation process, since it must be carried out with specialized applications. In this sense, there are applications oriented towards a content creators with a basic user profile in terms of programming notions, which offer a limited number of functions to add to an electronic book. Moreover, other types of editing applications are oriented towards content creators with programming knowledge, allowing for the implementation of any type of function to an electronic book.

The main motivation of the work presented in this article is to propose an intermediate solution with respect to the two extremes represented by the described tools: a tool that allows one to implement a large number of functionalities in an electronic book and that does not require deep programming knowledge.

In this sense, the tool that is proposed in this article is a novelty with respect to the existing tools, both in the requirements to be able to use it and in the approach that is proposed for its creation. This approach is based on the following ideas:

(a) Use of extensible markup language (XML) documents to describe the structure of an electronic book.
(b) Representation of the functionality of the electronic book using attribute grammars.
(c) Generation of the electronic book by processing the XML document using the attribute grammar.

The creator of the book would be responsible for carrying out step (a), while steps (b) and (c) would be performed semi-automatically using programming language processing tools that should be implemented only once.

The objective of this paper is to describe a proposal in the field of interactive fiction electronic book production oriented to its use in eLearning environments and that constitutes a compromise solution regarding the technological knowledge required of content creators. To do this, a solution based on language-directed development is proposed.

2. Related Work

The use of new technologies in the book industry has produced as the main novelty [1] the appearance of electronic books. The concept of the electronic book [2] is a combination of hardware and software that tries to emulate the physical book. In this sense, an electronic book is composed of a hardware device called an eReader on which special software is executed that is capable of executing and displaying electronic documents encoded according to a public or proprietary standard such as epub [3] that receives the name of ebooks [4]. However, mostly, the concept of the electronic book is associated with ebooks [5], since they are the ones that actually contain the book. The eReader is exclusively one of the possible means to view them, since an ebook can be viewed using other devices such as smartphones, tablets, or computers (the only thing necessary to read an ebook is the possession of software [6] capable of reading the ebook encoding format, which is device-independent).

The advent of electronic books represented a momentous change for both writers and readers in many ways [7]. In the case of the writer, the production of the book becomes an electronic edition of content [1], for which they can use anything from a text editor to more sophisticated editors oriented to specific ebook-encoding formats. In the case of the reader, the electronic format of ebooks opens up a range of possibilities for interaction [8] with the contents, which were not possible in the traditional physical format. In physical books, the interaction is static since it is limited to reading the printed text in the book. However, in an electronic book, the digitized text is complemented by other resources, such as multimedia elements (videos, images or audio), interactive elements (hyperlinks, animations, etc.) and other artifacts that enrich the reading process.

In the context of ebooks, the so-called interactive fiction arises [9]. This is a line of work that aims to take advantage of the new interaction possibilities offered by electronic books to change the role that readers play in the reading process. In traditional books, the role of the reader is passive [10] with respect to the content, since the only action that the reader can take is the reading of the book itself. However, the technology of the digital world opens the possibility of carrying out a huge variety of interaction actions [11] with the content of a book, such as [12] being able to view a video at a given moment of reading, listening to an audio file related to the content being read, participating in a game, expanding information on a topic described in the book, making annotations, etc. In this sense, interactive fiction aims to turn the reader into an active actor [10] in the reading process, so that it is the reader who builds the story he is reading based on his interactions with the content. As a consequence, not all readers will experience the same content and in the same way. To achieve this objective, different interaction strategies are implemented in electronic books, such as [13] (1) the existence of decision points where the reader must choose how to continue the story they are reading, (2) resolution of problems of various nature (puzzles, logic problems, etc.) that allow for the accession of new content

that is hidden or that provides clues on how to continue reading the book, (3) personalization of the story with data provided by the reader, and (4) use of gamification techniques, among others.

From a technical point of view, an electronic book is a digital document that, in addition to text, contains certain encodings [14] that implement different types of interactions. Thus, a writer essentially has two options to create an electronic book with these characteristics [15]. The first alternative consists of direct coding, for which you must have the necessary programming knowledge. The main advantage of this option is that any interaction strategy with the level of detail and configuration as desired can be implemented. However, its disadvantage [16] is the requirement of having the necessary programming knowledge, which is not usually the case for a writer.

The second option is to use a content editor that conceals the complexity of the programming and offers a user-friendly interface to the writer. In this sense, in recent years, different developments have emerged with this objective, such as the free source interactive fiction editors [17] Twine [18], Quest [19], Squiffy [20], TADS [21], Ren'Py [22], and others under licenses such as Inform [23]. In general, these tools have [24] the same structure as a normal text editor, except that in addition to offering standard text formatting elements (bold, underlined...), they also facilitate the possibility of including interaction elements within the content in a user-friendly way. The main advantage of this type of editor is that it is within the reach of most writers, since only user knowledge is required to be able to use them. However, its main limitations [25] are the reduced variety of interactive elements that can be used, the limited possibilities of adaptation, configuration or modification that it offers, and the impossibility in most cases to be extended with new interaction elements. Likewise, in general, the lack of a defined interactive fiction model is observed in all of the tools [26].

Interactive fiction has a direct application in the field of eLearning [27]. In this, there are different works that show the possibilities offered by interactive fiction for its use in the teaching–learning process [28]. These works highlight some of the benefits of interactive fiction for this purpose [29–31]: (1) The gamification characteristics that can be implemented in these resources positively influence the student's motivation to read and understand the content [32]. (2) The different types of interaction available can be used to implement different learning styles [33]. (3) The variety of information sources integrated into a single resource makes a more complete and comprehensive knowledge about each topic dealt with in the contents available to the student [34]. (4) It provides the teacher with the implementation of different learning paths according to the dynamic interactions of the students with the contents [35]. (5) It provides the teacher with the updating of the contents in a simple and dynamic, so that it is possible to speak of a "living book" [33], since it will never be completely closed. This will be modified throughout its useful life with new content or updates to existing ones. (6) It makes it easier for the teacher to evaluate students using some of the interaction elements available to implement different types of assessment tests [36,37]. Finally, (7) it facilitates the implementation of collaborative learning models through the use of content annotations [38].

3. Materials and Methods

3.1. Language-Driven Development

Language-driven development is a development paradigm based on applying the principles, techniques, and tools used in the design and implementation of computer languages. In this sense, it proposes the following [39]:

- The explicit definition of specific languages to describe different aspects of a software application.
- The implementation of said languages to automate the production of said aspects.

This approach has the following main advantages [40]:

- It facilitates development and maintenance. This is achieved because the aspects addressed by the different languages are specified and maintained at a high level of abstraction and close to the domains of the problems that general-purpose languages cause.

- It facilitates the active participation of experts in the domain of the application in the development process. Since the languages that are designed are focused on mastering the problem, it is not necessary for experts to have technical knowledge of software development. This allows experts to understand, and even modify, the specifications.
- It allows one to configure and apply sophisticated and advanced design patterns and algorithms in final implementations. This is because in this approach, some of the code is generated automatically without intervention from the developers.
- It promotes reuse. Once a language has been defined for one type of application, it can be reused to implement similar applications.

3.2. Language-Driven XML Document Processing

XML (extensible markup language) [41] is a specification that was proposed in the late 1990s by the World Wide Web Consortium (W3C) as an evolution of the SGML (Standard Generalized Markup Language) standard [42]. In recent decades, it has become a de facto standard used for the representation of electronic documents. XML, however, does not regulate how documents should be processed in a particular computer application. Processing of documents transcends the purpose of XML, and must be carried out using other means. In this way, since the appearance of XML, multiple technologies have also been proposed to address the problem of processing XML documents [43]. Regardless of their nature, many of these technologies share a common characteristic: understanding XML documents as data structures (e.g., XML documents as trees, XML documents as sequences of events, XML documents as strings of information elements, etc.). However, in addition to this data-oriented view, there is also a linguistic view of XML. In this sense, for a certain type of XML document, a specific markup language defined by means of a formal linguistic model (document grammar) can be proposed, and the XML documents can be processed by means of a processor for the markup language that defines such documents.

Therefore, the ideas of language-driven development can be applied when specific XMLs are used to describe certain aspects of an application, giving rise to the so-called documentary paradigm, which proposes the following:

- Defining specific markup languages that allow for the description of different aspects of the application to be developed by means of documentary grammars (DTD (a formal mechanism to represent the grammar of an XML language) or XML schemas).
- Description of the most relevant aspects of an application using XML documents that conform to the documentary grammars.
- Linguistic description of the processing to be carried out on said documents.
- Provision of processors for the defined languages, which, together with an appropriate application framework, allow the application to be automatically generated from the documents that describe it. These components are integrated into a core of the application.
- Generation of the application by processing the XML documentation. This phase can be seen as an effective instantiation of the application core to give rise to the application described by the documents.

3.3. Attribute Grammars for Processing XML Documents

The linguistic description of the processes to be carried out on an XML document supports several approaches. One approach consists of carrying out the description using translation schemes of the type supported by ANTXR [44], an environment built on the ANTLR tool, and RelaxNGC [45], an extension of the RelaxNG document schema language that allows automatic generation of descending recursive translators. Likewise, there is another higher-level approach that consists of using attribute grammars, a formalism of specification of the semantics of computer languages proposed in the late 1960s by the American scientist Donald D. Knuth [46,47]. These grammars are extensions of the BNF grammars

normally used in the description of the syntax of computer languages, which add semantic attributes and semantic equations to said grammars, in order to represent the processing of the sentences of the languages generated by them. In this way, the processing model associated with attribute grammars is based on computing the attribute values in such a way as to respect the dependencies established between the attributes by the semantic equations: before computing the value of an attribute, it is necessary to have computed the values of all the attributes on which it depends. Outside of this basic restriction, the order in which the attributes are computed is irrelevant, and in no case is it prefixed in the attribute grammar. This fact leads to a processing model directed by the dependencies between attributes, which is highly modular, since it is possible to add new attributes and equations without affecting the existing ones: the computation style directed by dependencies is in charge of automatically reorganizing the order in which the different processing steps must be carried out. In this sense, attribute grammars constitute a higher-level formalism than translation schemes, since in the latter, it is necessary to specify the order of execution of semantic actions.

4. Results

4.1. XLOP (XML Language-Oriented Processing)

XLOP is an environment that uses attribute grammars to describe how to process XML documents. In this sense, it follows the language-driven paradigm to develop XML document processing programs. Thus, these programs are understood as language processors, and the development process is understood [48] as the process of building and maintaining said processors. In this way, using XLOP, such processors can be automatically generated from high-level specifications expressed as attribute grammars.

XLOP consists of a specification language and a runtime environment. The specification language is a metalanguage that allows for the description of attribute grammars for XML markup languages (the syntax of the language is shown in Figure 1).

XLOPSpecification ::= {*Rule*}+
Rule ::= *NonTerminal* '::=' { *SintacticElement* }* '{' { *Equation* }* '}'
SintacticElement ::= *NonTerminal* | **#pcdata** | XML*Element*
XML*Element* ::= *OpeningTag* { *SintacticElement* }* *ClosingTag* | *EmptyElementTag*
Equation ::= *AttributeRef* '=' *SemanticExpression*
SemanticExpression ::= *Function* '(' (*SemanticExpression* { ,*SemanticExpression* }*)? ')' |
AttributeReference
AttributeReference ::= *Attribute* **of** (*NonTerminal* | **#pcdata** | *OpeningTag*) ('(' *OccurrenceNumber* ')')?

Figure 1. Extensible markup language (XML) language-oriented processing (XLOP) specification language syntax.

In an XLOP specification, the underlying uncontextual grammar represents the logical structure of a type of XML document. Moreover, for its definition, non-terminal symbols and terminal symbols are used. The latter can be of two types: (1) the #pcdata symbol to represent a fragment of textual content in the processed document, and (2) opening and closing tags that will depend on the XML language to be processed (in particular, they must be properly nested).

On the other hand, the required processing is represented by semantic equations that establish dependency relationships on semantic attributes associated with the symbols of the grammar. Thus, in the case of the #pcdata symbol, it supports only one lexical attribute, text, whose value refers to the particular textual content represented by the symbol. Regarding the opening tags, they can also have lexical attributes that correspond to the attributes that have been explicitly defined in the elements of the type of XML document to be processed. Finally, non-terminal symbols can have associated both synthesized and arbitrary inherited attributes (it is not necessary to make explicit in the grammars what type they are, since it is inferred by the use made of them in the equations). Note that the same

symbol can appear more than once in a production; in this case, to disambiguate the specific symbol to which the attribute belongs, an order number is used in the attribute reference.

Finally, note that XLOP does not offer support to define the semantic functions that are used in semantic equations, thus, the functions must be defined externally as Java methods of a semantic class.

The other element of XLOP is the generator. It is a component that takes as input a grammar described according to the XLOP specification language and generates an implementation of the associated processor written in CUP (a system for generating LALR parsers from simple specifications). In this implementation, it should be noted that the evaluation of the attributes is embedded within the syntactic analysis process. To do this, a delayed execution mechanism is used that causes the evaluation of the semantic equations to be carried out depending on the availability of values of the attributes that intervene in the corresponding semantic expressions.

On the other hand, observe that the execution of the generated processors also requires other additional elements:

- The semantic class that implements semantic functions as Java methods (mentioned above).
- The classes that make up the application-specific logic.
- The classes that configure the XLOP runtime environment. Among them is a generic lexical analyzer whose function is to go through the logical structure of XML documents and obtain a sequence of lexical components that will serve as input to the generated processor. Furthermore, this component connects with an API to XML documents based on SAX (simple API for XML, which is an event-driven online algorithm for parsing XML documents).

Thus, the use of XLOP allows one to define a life cycle for the production of XML processors outlined in Figure 2.

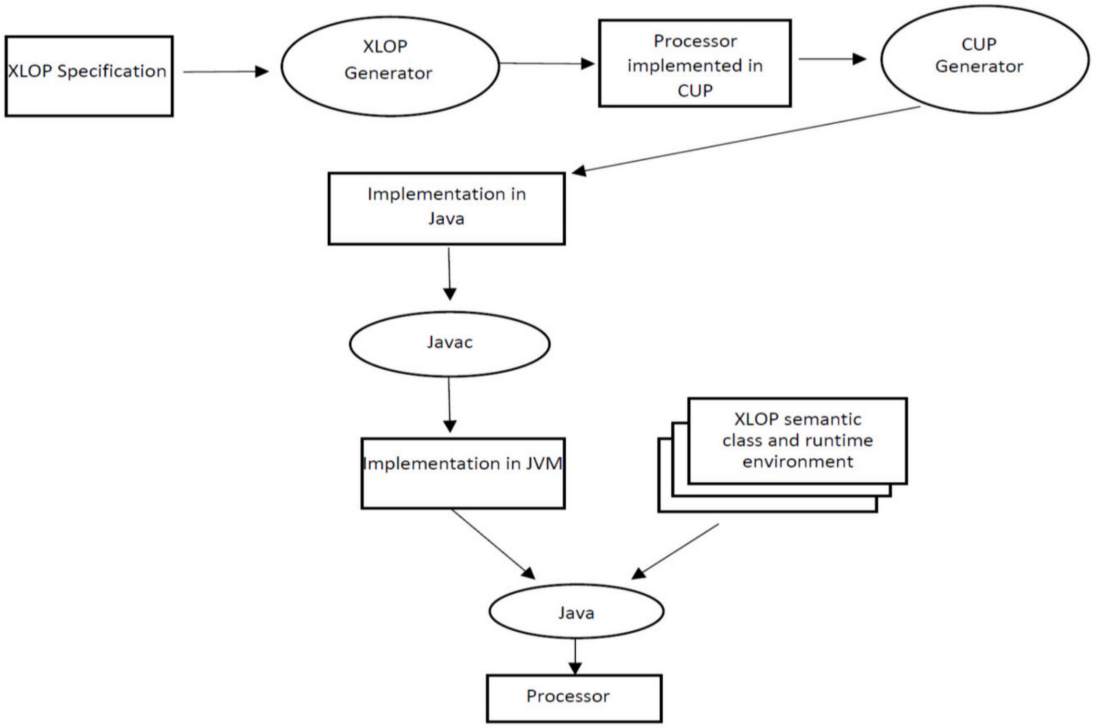

Figure 2. Life cycle for the production of XML processors.

This life cycle involves a set of productive activities:

(1) Writing a context-independent grammar specific to the processing task that characterizes the structure of the documents to be processed.

(2) Writing an attribute grammar that specifies the processing to be performed using attributes and semantic equations (XLOP specification).
(3) Implementation of the semantic functions used in the attribute grammar as Java methods (semantic class). This programming task will require specific application logic.
(4) Building of the processor.
(5) Testing of the generated processor with different documents. In the case of any modification being necessary, return to the previous activities.

Note the following:

- Processor maintenance, tuning or refinement tasks are simpler than if the processor had been implemented directly with a programming language. This is because the processor is defined at a higher level of abstraction (attribute grammar or XLOP specification).
- The approach advocated by XLOP allows one to structure XML processing applications in two layers connected by the semantic class: on the one hand, the application-specific logic layer (application-specific framework), and on the other hand, the linguistic layer of XML processing specified as an attribute grammar (XLOP specification that is translated into an executable implementation using the XLOP generator).

4.2. Proposal for the Production of the Interactive Fiction Electronic Book

The proposal for the interactive fiction electronic book production model for eLearning using language-oriented techniques would consist of the following activities:

- Analysis of electronic book technologies. The objective of this activity is to analyze the main technologies that are currently being used for the development of interactive fiction electronic books.
- Definition of an XML language to describe different types of electronic books with elements of interactive fiction. The objective of this activity is to obtain a markup language defined as a DTD or an XML schema that allows one to describe the content and structure of an interactive fiction electronic book.
- Transformation of the DTD or XML schema of the language to an equivalent non-contextual grammar. From the definition of the language obtained in the previous section, it will be transformed into an equivalent uncontextual grammar.
- Implementation of an attribute grammar that reflects how to process the description in the form of an XML document of an interactive fiction electronic book. For this, XLOP will be used as an auxiliary tool since it provides the necessary infrastructure to define an attribute grammar to process XML documents and generate the corresponding processor. In this sense, the attribute grammar will be implemented from the uncontextual grammar defined in the previous activity using the XLOP specification language. For this, a formalization process will be carried out following the established patterns of the logical process of an electronic book, adding the appropriate semantic equations and attributes to the non-contextual grammar that represents it and identifying the required semantic functions.
- Implementation of the specific logic necessary for the electronic book. During this activity, the corresponding semantic functions that arise during the previous activity will be implemented in Java, as well as all of the Java classes of the specific logic.
- Integration of the specific logic and the XML processing component generated from the attribute grammar. During this activity, the components created in the previous activities are integrated to produce the electronic book.

The scheme of the activities within the production process is shown in Figure 3.

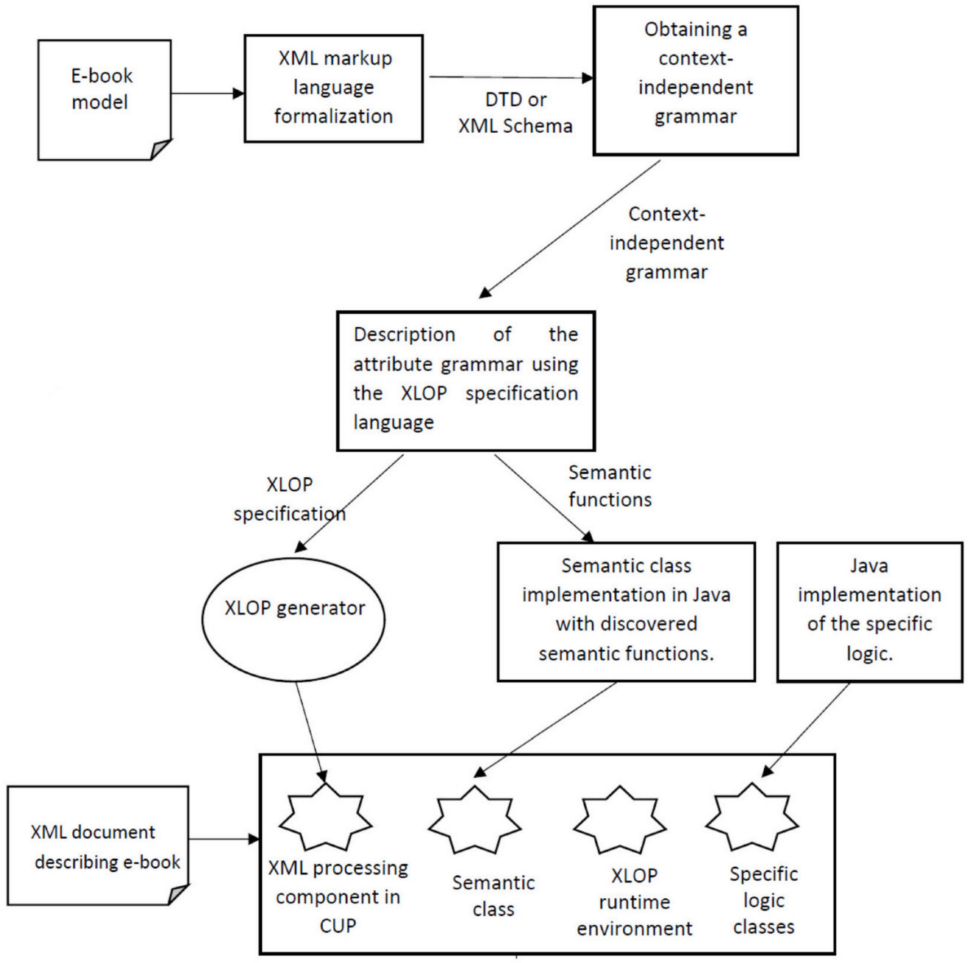

Figure 3. Electronic book production cycle.

Finally, Figure 4 shows an example of an interactive book created using the process described.

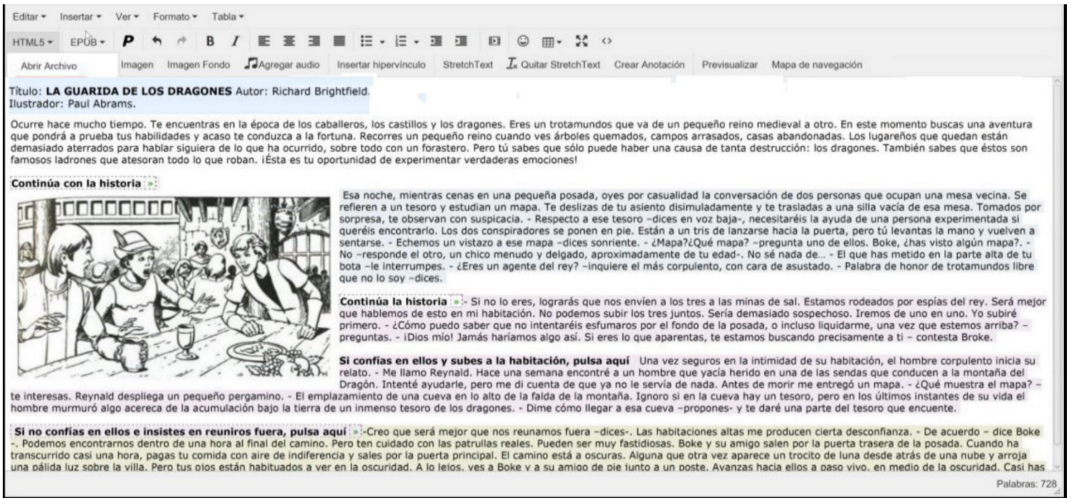

Figure 4. Example of an ebook.

4.3. Analysis

The solution proposed in this article is a novelty with respect to the tools that are currently used to create interactive electronic books due to the requirements that are asked of the content creators and

the approach used for its development. A content creator does not need to have deep programming knowledge, it only is necessary to describe the structure of the ebook using a markup language that will be provided. Next, a programmer will need to configure the tool appropriately with the components of application-specific logic, so that automatic processing of the XML document will result in the specified ebook. In this sense, if it is compared with the classic tools, it constitutes an intermediate solution in terms of user requirements and possibilities of functionality that it offers. A weak point of the proposal is that requires the collaboration of two types of roles: on the one hand, the writer, who will be in charge of structuring the book and providing the content, and the programmer who will be in charge of implementing the behaviors and the interaction components (the specific logic of the application that represents the different types of functions that can be added to electronic books). Another weak point is the complexity of the representation of some types of non-sequential interactions between the reader and the book, through a grammar.

5. Conclusions

From the point of view of the interactive fiction electronic book production process, the main novelty of this proposal is to offer a language-directed strategy for their development. According to this strategy, an interactive fiction electronic book can be represented using a domain-specific XML language, and its functionality can be described declaratively using a formalism widely used in the definition and processing of computer languages: grammar attributes. In this way, the generation of the book is translated into the processing of the XML document by a processor explicitly generated for this type of document.

The metalinguistic approach to development, which allows for the generation of interactive fiction ebooks from high-level declarative specifications, is therefore a unique distinguishing feature from other possible solutions.

The main limitations of the proposed approach are found in the implementation of the specific logic of the application that is indicated in the production process of the books. Likewise, another limitation is in the representation of certain complex behaviors of the interaction of readers with the content. Thus, non-linear behaviors could make the used attribute grammar too complex.

The main line of future work consists of the implementation of a component for generating learning objects in the Shareable Content Object Reference Model (SCORM) 2004 for electronic books. To do this, a program will be encoded that, taking as input an electronic book generated according to the previous model and an XML document that represents the book's metadata in LOM-ES (an application profile of LOM (learning object metadata)), will result in a learning object in the SCORM 2004 (a collection of standards and specifications for web-based electronic educational technology) format labeled with the LOM (learning object metadata—a data model used to describe a learning object and similar digital resources used to support learning).

Funding: This work has been supported by the Research Project CetrO+Spec (TIN2017-88092-R).

Conflicts of Interest: The author declares no conflict of interest.

References

1. Henke, H. The global impact of eBooks on ePublishing. In Proceedings of the 19th Annual International Conference on Computer Documentation, Santa Fe, NM, USA, 21–24 October 2001; pp. 172–180.
2. Robinson, A. Reading I didn't See Coming How eReaders Change Reading. *Kill Your Darlings* **2011**, *7*, 31.
3. Lin, L.C.; Tsai, T.P.; Lin, J.; Li, J. Some Useful ePUB3-based Contents Delivery Functions. In Proceedings of the 5th International Conference on Information and Education Technology, Tokyo, Japan, 10–12 January 2017; pp. 49–52.
4. Loebbecke, C. The Emergence of ebooks: Just Another Media Industry Joining the Converging Digital World? An Explorative Study on User Preferences and Industry Structure Changes. 2010. Available online: https://mtm.uni-koeln.de/team-loebbecke-publications-conf-proceedings/Conf-142-2010-TheEmergenceOfeBooks.pdf (accessed on 3 December 2020).

5. Vassiliou, M.; Rowley, J. Progressing the Definition of "e-Book". 2008. Available online: https://pdfs.semanticscholar.org/41b4/82071541e1cf2a8ec6f625d705eba00b3d33.pdf?_ga=2.102763904.1406676833.1606966126-1370911419.1602810527 (accessed on 3 December 2020).
6. Koh, H.; Herring, S.C. Ebooks, ereaders, and ebook device design. In *Encyclopedia of Information Science and Technology*; IGI Global: Hershey, PA, USA, 2015; pp. 2278–2287.
7. Colombo, L. An Approach to the Evaluation of eBooks from a User Experience Perspective. 2013. Available online: http://i3.fbk.eu/sites/i3.fbk.eu/files/ibooc2013-v2_0.pdf#page=40 (accessed on 3 December 2020).
8. Zambarbieri, D.; Carniglia, E. Eye movement analysis of reading from computer displays, eReaders and printed books. *Ophthalmic Physiol. Opt.* **2012**, *32*, 390–396. [CrossRef] [PubMed]
9. Ronn, M.L. *Interactive Fiction*; Ursabrand Media: New York, NY, USA, 2015.
10. Smith, S.; Bates, J. *Towards a Theory of Narrative for Interactive Fiction*; Department of Computer Science, Carnegie-Mellon University: Pittsburgh, PA, USA, 1989.
11. Wyatt, C.S. The Natural Accommodation of Interactive Fiction: How Text-Based Games Remove Barriers to Participation. In Proceedings of the Annual Computers and Writing Conference, Fairfax, VA, USA, 24–27 May 2018; pp. 27–28.
12. Ciesla, R. The (Ancient) Art of Interactive Fiction. In *Game Development with Ren'Py*; Apress: Berkeley, CA, USA, 2019.
13. Saffer, D. *Designing Gestural Interfaces: Touchscreens and Interactive Devices*; O'Reilly Media, Inc.: Sevastopol, CA, USA, 2008.
14. Bold, M.R.; Wagstaff, K.L. Marginalia in the digital age: Are digital reading devices meeting the needs of today's readers? *Libr. Inf. Sci. Res.* **2017**, *39*, 16–22. [CrossRef]
15. Koenitz, H.; Ferri, G.; Haahr, M.; Sezen, D.; Sezen, T.İ. *Interactive Digital Narrative. History, Theory and Practice*; Routledge: Abingdon, UK, 2015.
16. Batista, C.R.; Ulbricht, V.R.; do Valle Filho, A.M. A Model and Guidelines for the Interface Design Process for Adaptive Web Applications (IDPAWA). In Proceedings of the International Conference on Human-Computer Interaction, Heraklion, Greece, 22–24 June 2014; pp. 387–398.
17. Green, D.; Hargood, C.; Charles, F. Contemporary issues in interactive storytelling authoring systems. In Proceedings of the International Conference on Interactive Digital Storytelling, Bournemouth, UK, 5–8 December 2018; pp. 501–513.
18. Ford, M. Writing Interactive Fiction with Twine. 2016. Available online: http://ptgmedia.pearsoncmg.com/images/9780789756640/samplepages/9780789756640.pdf (accessed on 3 December 2020).
19. Salter, A. *What Is Your Quest? From Adventure Games to Interactive Books*; University of Iowa Press: Iowa, IA, USA, 2014.
20. Fatemi, S.S. Rewrite: An Experimentation in the Field of Interactive Fiction. 2014. Available online: http://www.biotica.org/tweaver/student_work/edpx5700_student_work/fatemi_ma_edpx5700.pdf (accessed on 3 December 2020).
21. Ciesla, R. Working in Ren'Py, Twine, and TyranoBuilder. In *Game Development with Ren'Py*; Apress: Berkeley, CA, USA, 2019; pp. 109–143.
22. Ciesla, R. Deeper Down the Dungeon. In *Game Development with Ren'Py*; Apress: Berkeley, CA, USA, 2019; pp. 145–187.
23. Reed, A. *Creating Interactive Fiction with Inform 7*; Nelson Education: Toronto, ON, Canada, 2010.
24. Linehan, C.; Kirman, B.J.; Reeves, S.; Blythe, M.A.; Tanenbaum, J.G.; Desjardins, A.; Wakkary, R. Alternate endings: Using fiction to explore design futures. In Proceedings of the CHI'14. Human Factors in Computing Systems, Toronto, ON, Cananda, 26 April–1 May 2014; pp. 45–48.
25. Amory, A.; Govender, D. *Interactive Fiction: Model Development and an Example Created with DHTML and Microsoft Agent*; AACE: Waynesville, NC, USA, 2000; pp. 1241–1242.
26. Martens, C.; Iqbal, O. Villanelle: An Authoring Tool for Autonomous Characters in Interactive Fiction. In Proceedings of the International Conference on Interactive Digital Storytelling, Snowbird Ski & Summer Resort, UT, USA, 19–22 November 2019; pp. 290–303.
27. De Diego, I.R.; Igado, M.F. Elige tu propio aprendizaje: Ficción interactiva y pedagogía. *Edutec. Rev. Electrónica Tecnol. Educ.* **2013**, *44*, a242. [CrossRef]
28. Aguilar, D.; Morón, A.C. Multimedia en educación. *Comunicar* **1994**, *3*, 81–87.

29. Flynn, S.; Hardman, M. The Use of Interactive Fiction to Promote Conceptual Change in Science. *Sci. Educ.* **2019**, *28*, 127–152. [CrossRef]
30. Howard, K.T.; Donley, R. Using Ink and Interactive Fiction to Teach Interactive Design. In Proceedings of the International Conference on Interactive Digital Storytelling, Snowbird Ski & Summer Resort, UT, USA, 19–22 November 2019; pp. 68–72.
31. Yu, Y.; Chen, H.; Wu, P. Using Brainwave Characteristics for Exploring the Effect of Integrating Graduated-Prompting Strategy into Interactive e-Books on Students' Learning Attention. In Proceedings of the 7th International Congress on Advanced Applied Informatics (IIAI-AAI), Yonago, Japan, 8–12 July 2018; pp. 330–333. [CrossRef]
32. McLellan, H. Digital storytelling in higher education. *J. Comput. High. Educ.* **2007**, *19*, 65–79. [CrossRef]
33. Mayfield, M.; Mayfield, J. Enhancing Student Learning and Engagement with Interactive Fiction Using Twine, by Chris Klimas. *Acad. Manag. Learn. Educ.* **2019**, *18*, 639–640. [CrossRef]
34. Sarasa-Cabezuelo, A.S. El uso de anotaciones como herramienta de aprendizaje. In Proceedings of the IV Congreso Virtual Internacional Sobre Educación, Innovación y TIC, Madrid, Spain, 18–19 December 2019; pp. 704–719.
35. Diez-Sanmartín, C.; Gayoso-Cabada, J.; Sarasa-Cabezuelo, A. Use of Critical Annotation and Interactive Fiction for the Creation of Digital Educational Content. *Int. J. Emerg. Technol. Learn.* **2020**, *15*, 231–244. [CrossRef]
36. Gayoso-Cabada, J.; Sarasa-Cabezuelo, A.; Sierra-Rodríguez, J.L. A review of annotation classification tools in the educational domain. *Open Comput. Sci.* **2019**, *9*, 299–307. [CrossRef]
37. Troussas, C.; Krouska, A.; Sgouropoulou, C. Towards a Reference Model to Ensure the Quality of Massive Open Online Courses and E-Learning. In *Brain Function Assessment in Learning*; Lecture Notes in Computer, Science; Frasson, C., Bamidis, P., Vlamos, P., Eds.; Springer: Cham, Switzerland, 2020. [CrossRef]
38. Temprado-Battad, B.; Sierra, J.L.; Sarasa-Cabezuelo, A. An Online Authoring Tool for Interactive Fiction. In Proceedings of the 23rd International Conference Information Visualisation (IV), Paris, France, 2–5 July 2019; pp. 339–344.
39. Abelson, H.; Sussman, G.J. *Structure and Interpretation of Computer Programs*, 2nd ed.; MIT Press: Cambridge, MA, USA, 1996.
40. Sierra, J.L. Language-driven Software Development. In Proceedings of the 3rd Symposium on Languages, Applications and Technologies, Bragança, Portugal, 19–20 June 2014; pp. 3–12.
41. Goldfarb, C. *The SGML Handbook*; Oxford University Press: Oxford, UK, 1991.
42. Birbeck, M. *Professional XML*, 2nd ed.; WROX Press: Birminghan, UK, 2001.
43. Sierra, J.L.; Fernández-Valmayor, A.; Fernández-Manjón, B.; Navarro, A. ADDS: Una Aproximación Documental al Desarrollo de Software. In Proceedings of the III Jornadas de Programación y Lenguajes, Alicante, Spanish, 12–14 November 2003; pp. 100–112.
44. Stanchfield, S. ANTXR: Easy XML Parsing based on The ANLR Parser Generator. Java Due.com, Hillcrest Comm. & FGM. Available online: javadude.com/tools/antxr/index.html (accessed on 10 October 2020).
45. Okajima, D. RelaxNGCC. *Bridg. Gap Schemas Programs* **2002**, *8*, 72–82.
46. Knuth, D.E. Semantics of Context-free Languages. *Math. Syst. Theory* **1968**, *2*, 127–145. [CrossRef]
47. Paaki, J. Attribute Grammar Paradigms—A High-Level Methodology in Language Implementation. *ACM Comput. Surv.* **1995**, *27*, 196–255. [CrossRef]
48. Rodríguez-Cerezo, D.; Sarasa-Cabezuelo, A.; Sierra-Rodríguez, J.L. Implementing attribute grammars using conventional compiler construction tools. In Proceedings of the Federated Conference on Computer Science and Information Systems, Szczecin, Poland, 18–21 September 2011; pp. 855–862.