Extraction of Information from Born-Digital PDF Documents for Reproducible Research

Jacek Siciare and Bogdan Wiszniewski
Department of Intelligent Interactive Systems, Faculty of ETI, Gdansk, Poland
Email: siciarek@gmail.com, bowisz@eti.pg.gda.pl

***Abstract*—Born-digital PDF electronic documents might reasonably be expected to preserve useful data units of their source originals that suffice to produce executable papers for reproducible research. Unfortunately, developers of authoring tools may adopt arbitrary PDF generation strategies, producing a plethora of internal data representations. Such common information units as text paragraphs, tables, function graphs and flow diagrams, may require numerous heuristics to handle properly each vendor specific PDF file content. We propose a generic Reverse MVC interpretation pattern that enables to cope with that arbitrariness in a systematic way. It constitutes a component of a larger framework we have been developing for making executable papers out of PDF documents without injecting in the PDF file any extra data or code.**

***Index Terms*—information retrieval models, content mining, executable papers, user interfaces**

I. Introduction

A born-digital PDF document, generated by an authoring or typesetting tool from the electronic original, is a data structure combining text, vector and bitmap graphics – composed of data objects from a small set including numbers, strings, arrays, dictionaries and streams, among others. Extracting useful information from such a document requires predefined interpretation patterns that underlie the data contained in its objects. Unfortunately, the internal logical structure of the original (source) document is severely degraded after being exported to the PDF format, so that only its visual layout and placement of objects may indicate basic relationships between different logical units of the document content. Two PDF files looking identical on a computer screen may have their internal structure of data objects arranged in the entirely different way. This is because developers of authoring tools may use different strategies to generate PDF files, making the internal document data structure *vendor sensitive*. In consequence, instead of literally parsing the PDF content in terms of formal grammar rules or data containers delimited by tags, visual interpretation must be performed in order to retrieve any useful information from it.

We propose for that purpose the *Reverse MVC* (*rMVC*) reconstruction pattern, which intentionally refers to the *Model-View-Controller* (*MVC*) design pattern–well known in programming Web applications.

In the paper we demonstrate its use for extracting information from tables, function graphs and flow diagrams within the context of reproducible research.

Information we focus on in the paper are: data series extracted from columns or rows of tables, as well as from line graphs, data records extracted from table rows, and data flows extracted from block and workflow diagrams.

A. Reproducible Research

The *reproducible research* principle requires that conclusions reported in a scientific paper can be reproduced independently by different readers using the content of its tables, graphs, diagrams, and other information units. Augmenting the original document with data and services may turn it into the *executable paper*, suitable for implementing such reproducible research scenarios, as simple exercises with the paper's functionality when tracing the paper's content from data contained in the paper to conclusions presented by the author, repeating experiments reported in the paper with alternative or voluminous data and computational resources provided by the paper's publisher, or interactive design of experiments that can combine data and services provided by the paper with some third party data or services.

The Executable Paper Grand Challenge launched by Elsevier in 2011 has identified a set of key attributes of executable scientific papers including executability, short and long-term compatibility, ability to validate data and code, conveying work done on large-scale computers, management of large size files and tracking of reader's actions, among others [1].

Numerous initiatives have been started to achieve the objectives listed above–some of them well before the challenge was launched by Elsevier.

One is *Amrita* capable of converting PDF documents into interactive entities with embedded JavaScript, for which the PDF viewer constitutes their computing environment [2]. One disadvantage of this approach is its strong dependence on the Rich Media technology–a solution that can persist for as long as supported by Adobe.

The Sweave tool can integrate in Latex documents various information units in the form of text, figures, tables, etc., based on the data analysis performed by

Manuscript received July 15, 2014; revised December 19, 2014.

©2016 Engineering and Technology Publishing

doi: 10.12720/joams.4.3.238-244

special chunks of R code embedded in the source document [3]. Owing to that the final PDF document will always be up-to-date, and the information necessary to trace back all steps of the data analyses available for inspection–if only the author makes the data and the relevant code available.

The *Planetary* system provides a Web-based authoring environment enabling authors to semantically annotate LaTeX source documents before transforming them into XHTML [4]. One advantage of this approach is that no system-level programming skills are required from authors. It narrows, however, the authoring process to just one source document format.

IPOL is an on-line journal publishing algorithms for image processing and analysis [5]. Authors must submit with their manuscripts the implementation software in C/C++, and the relevant datasets as Web pages. Components of the submissions are evaluated by referees and upon acceptance the code is distributed under the free software license. While making the IPOL papers executable is supported by the journal's technical staff and does not require authors to be skilled Web interface designers, distribution of their software remains outside of their control. An alternative would be providing to the paper readers some external Web service for executing the code outside of the actual paper location, under control of the paper's author.

The *Collage Authoring Environment* enables authors to augment traditional paper content with computational experiments combining code snippets and data files [6]. These experiments can be repeatedly executed by readers using a common Web browser, with the support of the publisher's server providing the required execution capability. There is practically no limitation on the programming language of the submitted code snippets, data formats or the paper theme, so practically any on-line journal of interactive publications may be created. Although the interactive elements of code are embedded in the publication interface, it remains executable for as long as it is located on the publisher's server and the interested reader stays connected.

B. *Interactive Open Document Architecture (IODA)*

We have proposed an Interactive Open Document Architecture (IODA) that does not require authors to prepare their papers with any specific authoring or typesetting tools in order to make them executable; it also enables augmentation of existing (already published) papers with the execution capability, without any need for prior arrangements nor interfering with the overall paper generation process [7].

The IODA document consists of three layers, as shown in Fig 1.

The bottom *data layer* contains the original document file in a printable form, accompanied by other textual or binary data and document services. The services may be implemented as *embedded*–with the executable binary code or scripts, or specified as *local* or *external* ones. Local services may be performed at the client's side upon copying data from the data layer, whereas external services, which the document may want to call, are provided from outside of the current document location.

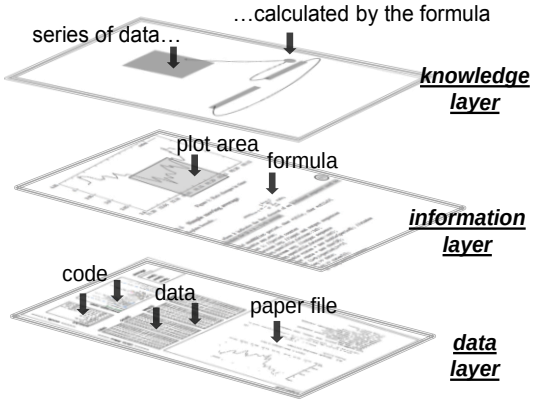

Figure 1. IODA layers

The middle *information layer* contains interpretation patterns for extracting logical information units from data provided by the data layer, whereas the top *knowledge layer* combines these units with services of the data layer into contexts enabling interactive experimentation of the readers with the document content.

Throughout the rest of the paper we concentrate on services embedded in the IODA document data layer that can support implementation of interpretation patterns provided by its information layer. In particular we will address the following two problems:

Problem 1: Extraction from the original PDF document file of any data that may constitute meaningful units of information, despite of its logical structure degradation when compared to its source original.

Problem 2: Recognizing and interpreting the relevant data objects in the PDF document file, despite of their alternative representations generated by various vendor specific PDF generators.

II. Logical Structure Degradation

The extent to which logical structure of the original document is degraded in the PDF document is specific to the particular PDF generator used. For example, the data object representing the text *Lorem ipsum* in the PDF file generated by the LibreOffice Writer tool will look like:

```
Tf[<01>2<020304>-6<05>2<0607>2<0809>-2<0A>-7<05>10<0B>]TJ
```

But when generated by the PDFLatex tool will look quite different:

```
Td[(Lorem)-333(ipsum.)]TJ 154.421 -615.691 Td [(1)]TJ
```

It may be seen that LibreOffice encoded each character of the example text with its respective key in the font dictionary (placed in the generated output PDF file as a separate data object), whereas PDFLatex represented the two component words directly as PDF strings, but eliminated the middle space by replacing it with a numerically determined visual offset -333.

MOST WIEDZY Downloaded from mostwiedzy.pl

This example indicates that attempting a universal procedure for retrieving logical units of information from the PDF document would be unrealistic, given the variety of possible strategies for generating "identically looking" PDF documents.

A. Plain Text

Continuity of sentences forming a paragraph may be broken into lines, words, or single characters, depending on the placement and/or orientation of the text in the original document. Text related data objects in the PDF file specify its size, position in the page and the font dictionary. Its component characters may be represented directly by strings or indirectly by keys in the font dictionary. When the text unit is rotated or its characters are scattered over some area even positions of individual characters may be specified separately.

B. Tables

Relations between cells, their content and borders, row labels and column headings, may be lost by the PDF generator, since they are free to assume any particular order of data objects inside the produced file.

Tables are usually represented as text lines (cell content) surrounded by cell borders. Labels and headings can be recognized if its font parameters differ from parameters of the cell content. Borders are line segments of a given length, so analysis of position and relations between text lines and graphical elements is necessary to retrieve tabular data.

Sometimes degradation inflicted by the PDF generator may go beyond the capacity of any analysis, especially when the original table lacked any graphical components. For example LibreOffice puts data objects with graphical elements of the table in the output PDF file before the objects with the textual content, whereas PDFLatex puts graphical and textual data objects describing each single table cell side by side.

C. Function Graphs

The common form of function graphs in research papers are line plots, which are usually approximated with line segments of the length measured in points (pt). Accuracy of the approximation depends strongly on the segment length assumed by the generator used to produce the output file. The respective pieces of information, including axes, the grid, tics (if any), value (text) labels of tics, and other graphical elements of data series, are split in numerous data objects of the document file. Before extracting values of data series, e.g. by sampling, curves must be reconstructed from the line segments by searching and analyzing all respective data objects in the file. Consider Fig. 2 illustrating the problem.

Positions of all segments of the line plot, e.g. $((x_1,y_1)(x_2,y_2))$ and $((x_2,y_2)(x_3,y_3))$, must be retrieved and their respective ends matched before sampling to extract the data series, e.g. $...,d_{i-1},d_i,d_{i+1},...$, could be applied.

In order to do that properly it is needed to determine the range of the plot and calculate the sampling rate proportional to that range or the distance of tics. Accuracy of the so extracted data depends on the size of segments approximating the line plot in the analyzed PDF document.

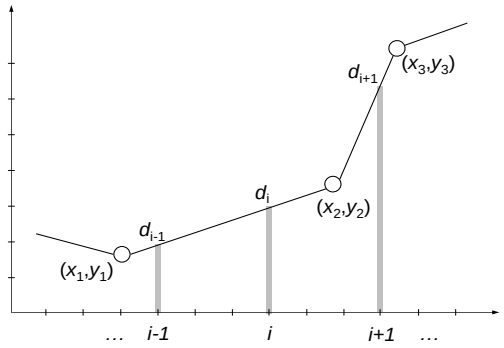

Figure 2. Interpretation of a function graph.

Extraction of multiple data series is possible if individual color or line pattern attributes are explicitly specified by the analyzed data objects, for otherwise multiple plots could not be distinguished.

D. Flow Diagrams

Nodes and arcs, visualized as various meaningful shapes, like boxes, circles, arrows, etc., are usually represented in the output PDF file as *paths* to be plotted by the underlying graphics engine, using elementary operations: *moveTo*, *lineTo*, *curveTo* and *closePath*. While reconstruction of these shapes is possible by mimicking the plotting operations, relations between shapes, e.g. links, are lost.

Arcs in turn are lines with optional arrows at their ends. Problems occur when retrieving arcs described by dashed or dotted lines. While horizontal and vertical lines can be retrieved easily–if they only have their respective (dotted or dashed) patterns indicated in the corresponding PDF data objects, skewed lines are usually composed of line segments with some empty space between them, complicating their recognition.

Other elements of interest may be textual or iconic descriptions placed all over the page area containing the diagram–with arbitrary positions, in general hard to associate with shapes.

III. Related Work

Extraction of logical information units from PDF files has been attracting attention of the document analysis and recognition research community for over a decade, in the context of content mining and understanding of electronic documents published on the Web–with tables and diagrams considered the most informative components of scientific and technical publications.

For example, the AIDAS tool was developed to extract content from technical manuals for a database of training materials [8]. The underlying idea of AIDAS was to sort data objects of each PDF document page on their coordinates and then run a set of grammars on them to filter out specific elements of the logical document structure, such as sections, lists, paragraphs, tables and figures. The approach was incremental, i.e. domain

MOST WIEDZY Downloaded from mostwiedzy.pl

©2016 Engineering and Technology Publishing

specific grammars could be added to recognize various classes of content, and would not fail for the unknown content. However, adopting this approach for extracting data for reproducible research would be problematic. Discovery of the logical structure of the PDF document concerned in fact only the presentation level–sufficient for partitioning the document content in chunks and store them in the database, but demanded more analysis to generate series of data from function graphs or identify flows in diagrams. Grammars capable of achieving that would have to capture specificity of PDF generators, thus developed separately for each specific vendor.

An impressive amount of effort has been spent by researches on automatic detection and recognition of tables and graphics in PDF documents. Although the particular approaches may vary in detail, e.g. some attempt machine learning techniques to separate diagrams in classes [9], classify tables with regard to the use of ruling lines based on the analysis of the significantly large set of document corpora [10], develop heuristics for aligning basic content elements and grouping them by considering their spatial features [11], or providing grouping rules to obtain higher level graphic and text objects from primitive objects, like line segments, rectangles, etc. [12], their common denominator is the bottom-up approach. In general it involves analysis of various PDF data objects in the document page and clustering them to form more complex structures up to the point where the table or figure is finally *detected* and *localized* in the page.

Recognition of tables, graphs and diagrams are all considered in the related literature very difficult problems. The reason is probably subscribing by researchers to the *bottom-up* philosophy and striving for uniformity of recognition rules, which based on our experience with executable papers, make the recognition task unnecessarily complex. The bottom-up approach is certainly justified for analyzing raster images of scanned documents, where attempts to build uniform recognition rules based on clustering of pixels are more straightforward than in the case of graphical primitives occurring in PDF files [13].

We propose instead a *top-down* approach. Firstly, no automatic segmentation of logical units of the document structure is needed if users can interact directly with the IODA document via its knowledge layer, as shown in Fig. 1. Note that the respective page and area containing the table or the diagram of interest may be just indicated or marked on the screen by the human reader.

Secondly, PDF generators used by individual authoring tools are quite stable in what graphical primitives they produce, and how they combine these primitives to render complex structures. Therefore, instead of analyzing thousands of PDF documents in order to classify these primitives, as well as the ways they may be combined, we preferred to find out what PDF data objects the most popular tools could generate. Information on the particular make and version of the PDF generator used to produce the paper is present in the file and can be easily retrieved!

Finally, especially in the case of table recognition, certain formatting rules are imposed by the publisher, so instead of searching document repositories for similar looking documents it might be wiser to analyze the template or style sheet used to generate the given document of interest.

The above observations have led us to the concept of the generic interpretation pattern driven by the set of vendor specific recognition rules. Its core is the rMVC pattern.

IV. Reverse MVC Pattern

Our rMVC interpretation pattern, like its MVC design pattern counterpart, separates the *Model* component holding units of data, from the *Controller* component with the control logic, and the *View* component providing presentation of the model data. In MVC the Model notifies the View on any change of the state of its data, but does not know details of the presentation. Only upon being notified by the Model the View retrieves the necessary data from it to produce the actual visual output. The Controller intercepts user generated events and sends requests to both: the View, to change presentation of the Model data, and the Model, to update their states accordingly.

Since MVC provides a template solution for graphical interpretation of units of data (from the Model to the View), reversing the process (from the View to the Model) should get the units of data when interpreting their visual presentation. In other words, rMVC can provide a template solution for extracting information units from the PDF document, being nothing less than visual interpretation of data dispersed inside the document file.

Conceptually, extraction of document data in the form enabling their use in various reproducible research scenarios, corresponds in fact to the *reconstruction* of the Model. This idea is outlined in Fig. 3.

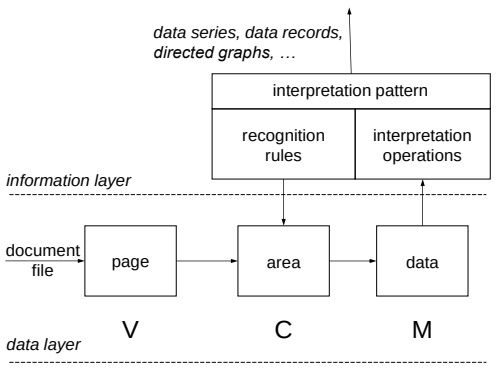

Figure 3. Reconstruction with Reverse MVC.

We propose a generic interpretation pattern, combining *recognition rules* and *interpretation operations*. Recognition rules provide the relevant logic for the Controller (C) component shown in Fig. 3, whereas interpretation operations are used to process data of the Model (M) component. The extraction process is started by the View (V) component, which gets a single page of the PDF document, filters out and decodes its textual data

MOST WIEDZY Downloaded from mostwiedzy.pl

©2016 Engineering and Technology Publishing

streams, separates metadata to obtain the document generator make and version info, and transfers them to C.

Based on that info C chooses the appropriate recognition rules for the page area to which the interpretation pattern is applied. These rules involve heuristics matching the particular vendor, or follow explicit specifications of the publisher, as in the case of the Sweave package mentioned before.

Recognition rules are used by C to analyze streams positioned in the area indicated by the reader, whereas interpretation operations extract units of data, create data structures in the internal rMVC application format and transfer them to M.

In our current implementation M provides tabular data as series, and function graphs as samples, both in JSON [14], and flow diagram data as a collection of nodes and arcs in XML [15].

The interpretation pattern in Fig 3 may be applied repeatedly to the same page area, if more then one logical unit of information is expected, e.g. the page contains a table and a function graph located side by side or embedded one in another.

V. Interpretation Patterns

While it is relatively easy to analyze individual data objects in the PDF file, e.g. with regular expressions, their visual context needs vendor sensitive algorithms capable of structuring them accordingly to human perception of patterns. That is why we propose to take an advantage of rMVC with the generic interpretation pattern augmented with recognition rules capturing vendor specific heuristics. They enable separation of various logical units of information in the document content, e.g. aligning coordinates of text elements to graphical elements when recognizing table cells.

Thus identified units of data can be processed next with interpretation operations to make them usable, e.g. exporting them to the required data format, like CSV or XML.

Below we present example interpretation patterns for tables, function graphs and flow diagrams that we have implemented already with our generic interpretation pattern. In each case the specified rules had to be *redefined* for each specific vendor, in order to capture its specific PDF generation strategy, whereas the interpretation operations remain *invariant* for all vendors.

A. Table Patterns

Recognition rules for tabular data in PDF files generated by the LibreOffice generator are the following:

- The page area contains vertical and horizontal objects with equal *x* and *y* coordinates and text elements are surrounded by these segments;
- Fonts of table headers differ from the rest of the table. If not, additional subrules have to be applied to detect them;
- The table is in the normal form, i.e, it does not contain joined cells. Detection and splitting of joined cells requires the additional subrule;
- Tabular data are lists of records, where values of table headers are field names.

Interpretation operations to be performed on data identified with the table recognition rules enable to:

- **Sort** tabular data by the given field name in the ascending or descending order;
- **Filter** with the use of regular expressions;
- **Partition** the specified set of records into subsets with regard to various attributes;
- **Aggregate** data with functions like *sum*, *mean*, *max*, etc.;
- **Export** data to any specified format, e.g. CSV or XML;
- **Redirect** extracted data to specific external interpretation services.

B. Graph Patterns

Recognition rules for function graphs generated by Gnuplot would be the following:

- The page area includes the border, axes, grid and tick segments, along with tick value labels;
- Adjacent line segments, optionally distinguished by color or line pattern attributes, represent the line plot;
- Relative positions of tick segments and their value labels determine the range of x and y coordinates;
- Line plot segments determine a series of samples for the assumed resolution (sampling rate and quantization levels).

Interpretation operations for data extracted with the function graph recognition rules make it possible to:

- **Show** and **hide** the specified data series;
- **Cut off** line plot sections to provide the specified range;
- **Export** extracted data series to any specified textual or graphical format;
- **Redirect** extracted data to specific external interpretation services.

C. Diagram Patterns

Surprisingly in our experiment with BPMN diagrams, both LibreOffice Draw [16] and Xfig/transfig [17] generated PDF files that could be handled with the same set of rules specified below:

- The biggest rectangular shape in the page area is the BMPN diagram boundary;
- Data objects inside the boundary are graphical and textual components of the diagram;
- Graphical objects are shapes of the predefined set of node elements, textual and iconic descriptions, relations and arrow heads;
- Graphical objects are shapes of the predefined set of node elements, textual and iconic descriptions, relations and arrow heads;
- Relative positions of textual descriptions and graphical objects determine relations between components of the flow;
- Node components are vertices and relation components are edges of the directed graph. Other elements are metadata.

MOST WIEDZY Downloaded from mostwiedzy.pl

©2016 Engineering and Technology Publishing

Interpretation operations for data extracted with the flow diagram recognition rules make it possible to:

- **Generate** the adjacency matrix and perform basic graph operations, e.g. finding the shortest path in the graph or building its spanning tree;
- **Extract** the specified flow for further analysis;
- **Redirect** processing of extracted data to the specified external interpretation services;
- **Export** the adjacency matrix to various data formats.

VI. Implementation

A prototype implementation of the rMVC pattern outlined in Fig. 3 concerned first extraction of tabular data from PDF documents. The objective was to exploit regular expressions to filter out lines with definitions of graphical and textual objects from textual PDF stream objects. It, however, turned out to be quite onerous, as implementation of regular expressions in Java required considerable effort. When compared to just a few characters in Perl, in Java the significant portions of code had to be written to get the regular expression with a similar functionality. Later we switched to Inkscape [18], the tool capable of converting PDF to SVG. Owing to that the content of document pages was much easier to be searched for various vector graphics objects with popular XML tools.

We have analyzed several vendor specific PDF generation strategies in order to compare them and to define their relevant recognition rules. They included extraction of data series from the LibreOffice Writer and the Microsoft Word tables [14], extraction of data series from Gnuplot function graphs, and extraction of business logic from BPMN flow diagrams generated with LibreOffice Draw [16], Xfig [17] and PDFLatex [19]. Substantial discrepancies between PDF generation strategies have been observed. Although their detailed presentation is well beyond the size limit for this paper, one conclusion is obvious – there is no chance whatsoever to get a set of PDF generation rules widely accepted by all major vendors of authoring and/or typesetting tools. In consequence it will not be able to provide one common set of recognition rules for basic information units that may be found in research papers published as PDF files.

VII. Conclusion

The reproducible research principle states that conclusions reported in a scientific paper can be reproduced independently by different readers using the paper content. If not provided otherwise, the data must be properly interpreted and extracted from the document file and specific execution services provided to process them.

We prefer to address the above problem with a "lightweight" approach, and propose two major innovations beyond the current state-of-the-art – one in the area of information extraction from the PDF document content, and another in the area of paper executability.

Information extraction: we focus on extracting just as much information as possible from the area explicitly indicated by the paper's reader and in the top-down fashion, with predefined interpretation patterns, instead of combining graphical elements to see what can be found and where in the paper. This approach is more efficient, as not all information extracted from the paper content is useful for reproducible research. As indicated before we are interested in extracting data series from tables and function graphs, and flows from diagrams.

Paper executability: we enable the executable paper to be self-sufficient, i.e. executable outside of the publisher's server. The multilayered IODA architecture supports that by clearly separating document data (the data layer) from their underlying interpretation patterns (the information layer) and the reproducible research scenarios (the knowledge layer). This separation significantly eased the effort on extracting information from born-digital PDF documents and enabled us to successfully address all the key attributes of the executable paper postulated by Elsevier.

Acknowledgment

This work was supported in part by the National Science Center grant no. DEC1-2011/01/B/ST6/06500.

References

[1] Elsevier. (2011). The Executable Paper Grand Challenge. [Online]. Available: http://www.executablepapers.com.

[2] J. Quirk. (2012). Executable Papers–The Day the Universe Changed. [Online]. Available: http://www.amrita-ebook.org/doc/amp/2011.

[3] F. Leisch, "Sweave: dynamic generation of statistical reports using literate data analysis," in *Proc. Comp. Statistics COMPSTAT'02*, Berlin, Germany, 2002, pp. 575-580.

[4] M. Kohlhase, "The planetary project: Towards eMath3.0," in *Proc. 11th Int. Conf. on Intelligent Computer Mathematics CICM'12*, Bremen, Germany, 2012, pp. 448-452.

[5] N. Limare, L. Oudre, and P. Getreuer, "IPOL: Reviewed publication and public testing of research software," in *Proc. IEEE 8th Int. Conf. on E-Science (e-Science)*, Chicago, IL, USA, 2012, pp. 1-8.

[6] E. Ciepiela, D. Harężlak, M. Kasztelnik, J. Meizner, G. Dyk, P. Nowakowski, and M. Bubak, "The collage authoring Environment: from proof-of-concept prototype to pilot service," *Procedia Computer Science*, vol. 18, pp. 769-778, 2013.

[7] J. Siciarek and B. Wiszmiewski, "IODA–An interactive open document architecture," *Procedia Computer Science*, vol. 4, pp. 668-677, 2011.

[8] A. Anjewierden, "AIDAS: Incremental logical structure discovery in PDF documents," in *Proc. 6th Int. Conf. on Document Analysis and Recognition*, Seattle, WA, USA, 2001, pp. 374-378.

[9] R. P. Futrelle, M. Shao, Ch. Cieslik, and A. E. Grimes, "Extraction, layout analysis and classification of diagrams in PDF documents," in *Proc. 7th Int. Conf. on Document Analysis and Recognition, ICDAR'03*, Edinburgh, Scotland, UK, 2003, pp. 1007-1014.

[10] T. Hassan and R. Baumgartner, "Table recognition and understanding from PDF files," in *Proc. 9th Int. Conf. on Document Analysis and Recognition*, Parana, 2007, pp. 1143-1147.

[11] E. Oro and M. Ruffolo, "PDF-TREX: An approach for recognizing and extracting tables from PDF documents," in *Proc.*

MOST WIEDZY Downloaded from mostwiedzy.pl

10th Int. Conf. on Document Analysis and Recognition, ICDAR'09, Barcelona, Spain, 2009, pp. 906-910.
[12] A. Gabdulkhakova and T. Hassan, "Document understanding of graphical content in natively digital PDF documents," in *Proc. the 2012 ACM Symp. on Document Engineering* (DocEng '12), Paris, France, 2012, pp. 137-140.
[13] W. Szwoch and M. Mucha, "Recognition of hand drawn flowcharts," in *Image Processing and Communications Challenges 4, Advances in Intelligent Systems and Computing*, vol. 184, R. S. Choras, Ed, Springer, 2013, pp. 65-72.
[14] J. Siciarek, "Semantics driven table understanding in born-digital documents," in *Image Processing and Communications Challenges 5, Advances in Intelligent Systems and Computing*, vol. 233, R. S. Choras, Ed, Springer, 2014, pp. 153-160.
[15] P. Pieniążek, "Automatic extraction of business logic from digital documents," in *Proc. 6th Int. Conf. in Image Processing & Communications*, Sept. 10-12, 2014, Bydgoszcz, Poland (in press).
[16] I. Vignoli, "LibreOffice: State of the project," presented at the LibreOffice Conference, Milano, Italy, Sept. 25-27, 2013.
[17] T. Sato and B. V. Smith. (2013). Xfig user manual. [Online]. Available: http://xfig.org/userman/authors.html
[18] T. Bah, *Inkscape: Guide to a Vector Drawing Program*, 4th ed. Prentice Hall, 2011.
[19] K. Höppner, "Strategies for including graphics in LATEX documents," *The PracTEX Journal*, No. 03, Rev. 2005-07-15, pp. 1-11, 2005.

Jacek Siciarek got his BSc in Computer Science from Gdansk University (UG) in 2005 and MSc in Computer Science from Gdansk University of Technology (GUT) in 2008 (with honours). He started his PhD studies in 2010 at the Department of Intelligent Interactive Systems at GUT and is expected to complete them by the end of 2014. He is currently employed as a software architect in a private company in Gdansk and as an investigator in the project "MENAID – methods and tools for next generation document engineering", funded by the Polish National Science Centre. His research interests include document engineering, advanced human-computer interfaces and Web programming.

Bogdan Wiszniewski got his MSc in Electronics in 1977 from Gdansk University of Technology (with honours), PhD in 1984 and DSc in 1998, both in Computer Science and from Gdansk University of Technology. In 2006 was granted the Professor academic title by the President of Poland. Since 2000 he is the Head of the Department of Intelligent Interactive Systems at Gdansk University of Technology. He coordinated several national and international projects funded by national (Polish) and international funds, from the areas of distributed processing, software engineering, document analysis and document engineering. His publication record of seven books and about 200 papers, includes international and national books, book chapters, journal articles and conference papers. The current research focus of prof. Wiszniewski is on advanced human-computer interaction, document-centric virtual collaboration, intelligent agents and automated negotiation.

MOST WIEDZY Downloaded from mostwiedzy.pl

©2016 Engineering and Technology Publishing