Graphic Design Principles for Automated Document Segmentation and Understanding

J. Fernando Vega-Riveros, fvega@ece.uprm.edu
Hector J. Santos Villalobos, hector.santos@ece.uprm.edu
Department of Electrical and Computer Engineering
University of Puerto Rico, Mayagüez

Abstract – When designers develop a document layout their objective is to convey a specific message and provoke a specific response from the audience. Design principles provide the foundation for identifying document components and relations among them to extract implicit knowledge from the layout. Variable Data Printing enables the production of personalized printing jobs for which traditional proofing of all the job instances could result unfeasible. This paper explains a rule-based system that uses design principles to segment and understand document context. The system uses the design principles of repetition, proximity, alignment, similarity, and contrast as the foundation for the strategy in document segmentation and understanding which holds a strong relation with the recognition of artifacts produced by the infringement of the constraints articulated in the document layout. There are two main modules in the tool: the geometric analysis module; and the design rule engine. The geometric analysis module extracts explicit knowledge from the data provided in the document. The design rule module uses the information provided by the geometric analysis to establish logical units inside the document. We used a subset of XSL-FO, sufficient for designing documents with an adequate amount complexity. The system identifies components such as headers, paragraphs, lists, images and determines the relations between them, such as header-paragraph, header-list, etc. The system provides accurate information about the geometric properties of the components, detects the elements of the documents and identifies corresponding components between a proofed instance and the rest of the instances in a Variable Data Printing Job.

1. Introduction

Today's printing technology has reached new frontiers with the arrival of the digital era. Politis defined Digital Publishing (DP) as printing-imaging processes where the film or plate making stages are eliminated, and where printing or imaging takes place after the pre-press process [8]. It is an end-to-end workflow where customers can obtain the desired service on-demand. In other words DP allows publishing anything, at anytime, anywhere, and by anyone. Digital Publishing (DP) empowers individuals to control all, or most of the publishing processes [10].

One of the advances that DP has promoted is Variable Data Printing (VDP). The essence of VDP is job personalization where a document layout yields a template whose content can be filled with different images, figure illustrations, or texts for each individual reader [11]. A job in DP remains digital until it is actually printed on paper allowing the modification of the job at any point along the workflow and the generation of multiple variants with minimal cost.

Print-shops collect basic information about the customer and the job in the Intent stage, which is used for quality control and tracking. Jobs must pass through a number of processes before actually sending the job to the press, of which preflight and proofing, are very relevant to our work. Preflight checks that the digital document contains all the elements required to perform

well in the production workflow. However, proofing, as currently conducted by human experts to ensure the quality of every job, raises a problem when dealing with VDP jobs. If the proofing expert wants to ensure that all the instances in a VDP job are ready for printing, each instance of the job should be verified; an impossible task to execute, given that a single VDP job could contain millions of different instances.

As a result of the variable data, some defects or artifacts may appear on some instances of the job. We have classified defects specific of VDP into two categories. On one hand we find defects that are evident, like a text overflow in a document field. On the other hand, we find style inconsistencies with regard to the original design of a particular VDP job, which are more subtle defects. For example, we can not determine that a given heading with typeface Times New Roman in a job instance is defective, until we know that the typeface used by the designer was Arial. We refer to this type of error as style-dependent artifacts[1]. The data required to detect such artifacts are implicit in the document context. Graphic designers use a flexible but consistent set of principles to avoid aesthetic flaws [9]. These principles act as a procedure to encode the message in a document. Hence, if we assume that the message is encoded using a set of rules that represent those principles, we can decode the document with the same rules and artificially extract the logical meaning of the components inside a page. Style-dependent artifacts cannot be detected by preflight and thus, based on the arguments above we developed a Knowledge-Based Artifact Recognition (KBAR) tool. KBAR is designed for document segmentation, layout understanding, artifact detection, and quality assessment. A detailed block diagram of the KBAR tool is shown in Figure 1. In this paper we describe our approach to document segmentation and understanding. For a detailed description of the KBAR tool refer to [5].

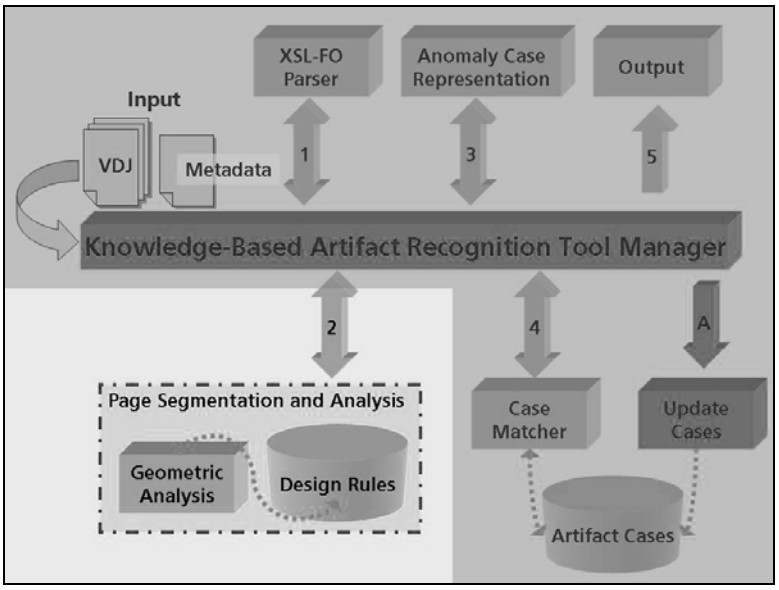

Figure 1: K-BAR System Architecture

[1] The term artifact stands for any type of defects in digital documents; it is used throughout this paper interchangeably with the terms defects and errors.

2. Design principles for document understanding

Variable Data Printing gives rise to fascinating novel printing services such as document personalization. These services bring along challenges such as maintaining the original document design intentionality, while the contents can vary widely. Thus, graphic design principles need to be incorporated as part of the processing required for detecting artifacts that may arise as a result of variable contents.

The most basic design principle is known as *readability*, which is a term that refers to the adequacy of an object to attract readers. Good readability makes the page comfortable to read. This term can not be confused with word legibility, which describes the adequacy of an object to be deciphered [6]. When designers develop a document layout or template, their objective is to convey a specific message and provoke a specific response from the audience [12]. The principle of readability establishes that the purpose of a design is to make the document understandable. This main principle is achieved by two general rules, the white space and unity rules.

White space, also known as negative space, has more significance than the usual value people give to it and is considered by many as the most important factor in document design. It provides the context or physical environment in which a message or form is perceived [6]. In addition it balances the context; giving the eyes a visual break [12]. Negative space helps the reader to navigate through the content and determine what part of the information is the most relevant. In addition a good amount of white space gives the design a kind of luxury, generosity, or classic simplicity. The most important thing about white space is that it does not appear as an obvious consequence of the positive space or content. It is deliberately used.

Unity is achieved by joining elements and exploiting their potential relationships and alignments [6] [12]. It is composed of four sub-principles: similarity; contrast; proximity and repetition.

The *similarity* principle refers to elements that share similar properties like size, color shape, position, or texture. *Alignment* is a sub-principle inside similarity. It is used to unify and organize the components in the page. There is a term known as *strong lines*. A strong line is an invisible line that runs at the edge or the center of the aligned components [7]. Designers make use of strong lines to define a relationship between components. In general the similarity principle tells viewers when some similar context share a logical meaning.

The *contrast* principle is the reverse of similarity. It gives different meanings to objects inside the layout [7]. For example, different sizes between objects lead to a different appreciation of importance. The job of contrast is to tell viewers when objects are different.

The principle of *proximity* establishes that elements physically close together are related, and less related if they are separated further apart [6] [7]. For example section headings should be closer to the paragraph that follows the heading than to the paragraph before. In this case the proximity rule gives viewers implicit information that connotes that a heading is more related to the paragraph below than to the paragraph above. Moreover the proximity principle has a dual functionality; on one hand, it is used to establish relationships between components, as the similarity principle does; and on the other hand it establishes the components level of importance, as the contrast principle does.

The *repetition* principle produces rhythm. When spectators see the same size, positioning, color or use of rules, background, and boxes; they expect equivalent meanings [12] [7] [4] [6]. This

principle should not be confused with the similarity principle. Similarity deals and exploits the correspondence between the properties of components, while repetition deals with replications of similarity, contrast, and proximity together. Moreover, repetition exploits the joint properties of components and their logical meaning inside the layout. A good example of the capability of repetition is found in newspapers. When a reader is looking for his/her favorite newspaper between several different ones in a stand, he/she can rapidly identify his/her favorite one because of its particular design; an impossible achievement if the newspaper administrators changed the design every week. Repetition gives an identity to the document and can be interpreted as consistency that will unify all parts of the design [7].

In the following sections we discuss the use of these graphic design principles for document segmentation and understanding.

3. Document Segmentation and Understanding

An accurate document defect analysis requires extracting information from what is implicitly stored in the document layout. As a first approach, a basic segmentation and analysis tool for single page documents was developed as a proof of concept.

We used a subset of XSL-FO for document formatting and developed a parser to extract the job information needed by the segmentation module to subdivide the document into logical units. The segmentation module is composed by two sub-modules: the geometric analysis and the design rules module.

The geometric analysis module extracts explicit knowledge from the data provided by the XSL-FO parser. The properties gathered from each document component are: position inside the page; dimensions; text size, leading, typeface and color; page margins; percentage of white space on the page; and page dimension.

The design rule module uses the information provided by the geometric analysis together with graphic design principles to establish logical units inside the document. It determines relations between components and obtains implicit knowledge from the document layout, such as headers, captions, footnotes, image-caption relations, etc. This module was implemented as a rule-based system and uses the design principles of repetition, proximity, alignment, similarity, and contrast.

4. The analysis cycle

It is assumed that one instance of the VDP job has been proofed by a human expert. We do not make any specific assumptions as to how this instance is chosen. This instance is referred to as the *approved instance*. Figure 2 shows a model of the sequence of tasks in the tool. Every instance of the VDJ has to pass through these processes including the approved instance, but the approved instance stops after the process *Assign Logical Types to Components*. All the information gathered about the approved instance at that point is stored in memory for later evaluation of the remaining instances.

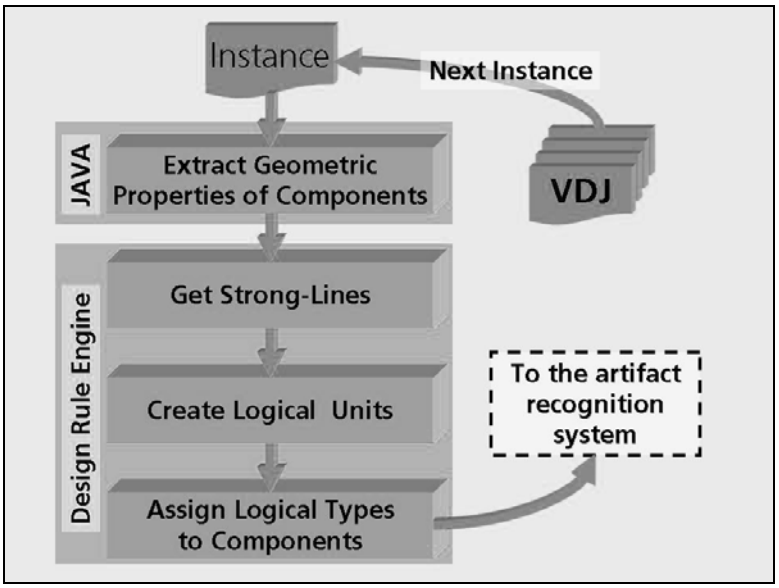

Figure 2: Flow Chart for Document Segmentation and Understanding

4.1. Gathering Geometric Properties

The first process in the geometric analysis is to extract the geometric properties of the document page layout and context from the XSL-FO file. We only used a small subset of the XSL-FO [2] but the subset allows designing documents with an adequate amount complexity to test our concepts.

We created Java Objects to represent each XSL-FO object. The extraction of geometric properties is done inside a recursive function since XSL-FO uses a tree structure. The recursive function receives each node and gathers all its children. The program enters a loop which will exit only when all the children of a node are processed. When a child is taken and the respective class loaded, the child is passed to the class Java Object which extracts all the information contained in the child and returns to the main loop. In the main loop, if the child has other children, the recursive function is called again but this time the child is sent as the input parameter, and the process repeats for this new node. When a node does not have any children but siblings, *i.e.* it is a terminal node, the program processes the remaining siblings and checks if any of the siblings have children. The geometric properties extraction finishes when there are no more children or siblings. All the extracted objects and their respective properties are stored in a general object called the *InstanceGeometricLayout*.

4.2. The similarity principle: detecting strong lines

The principle of similarity has as sub-principle, the alignment rule, and one of the alignment rules is the strong line. Figure 3 shows a business card that will be used to explain how strong lines in the document layout are found. Strong lines can exist vertically and horizontally but in this first implementation only vertical strong lines were considered. The horizontal strong line detection follows a similar process.

The first step is determining suitable coordinates through which potential strong lines may pass, such as the left edge, the center, or the right edge of a component. In Figure 3 this points are

marked with asterisks. The current system assumes the language has a left-to-right orientation, like English or Spanish. Notice that the logo on the left of the card has three suitable coordinates, because it was not explicitly aligned with any other component. However the text components on the right of the card are right-aligned.

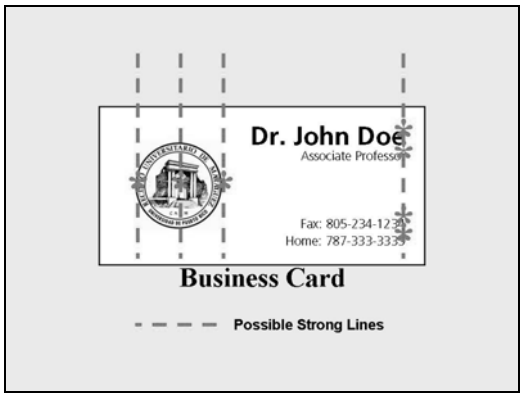

Figure 3: Example showing left, center and right coordinates and potential vertical strong lines

Strong lines are established if there are components with similar suitable coordinates. Figure 3 shows potential strong lines for the example. Components can be on multiple potential strong lines at the same time, and there is no restriction as to the number of components on a strong line. If there is more than one component through the trace of a potential strong line these components are grouped together.

The last step is the removal of invalid strong lines. An invalid strong line has less than two components on it. When two or more strong lines share the same component; the component is eliminated from the shorter strong lines. When two strong lines are very close to each other the system measures the distance between them and if the distance is below 2 points (1 point = 1/72 inch) the rule engine merges the components into a single strong line in the middle of the merging lines. In Figure 3, only the right most strong line remains after merging.

4.3. Creating Logical Units

Logical Units are created based on the proximity and contrast principles. Components that are placed purposely closer by the designer [6], [7], implicitly establish a strong relation among them.

Any isolated component is initially considered a potential unit. Components that belong to the same strong line are also considered a potential unit. Figure 4 shows the initial logical units for the example. The proximity and contrast rules may split units in groups of components with higher cohesion between them. Thus, the system splits these units in pairs as shown in Figure 5.

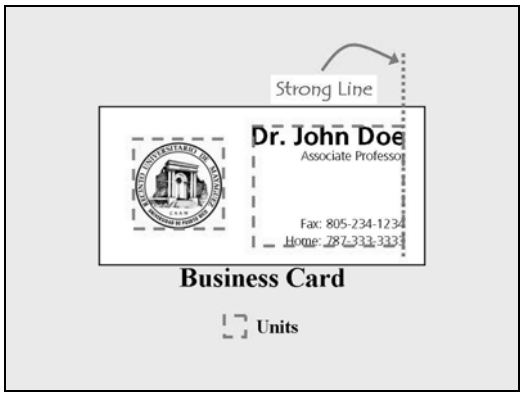

Figure 4: Initial logical units determined from strong lines

In the example in Figure 5 the unit on the right hand side, initially composed of four components, was split into three overlapping and smaller units of two components each. This task is done for every unit on the page with three or more components.

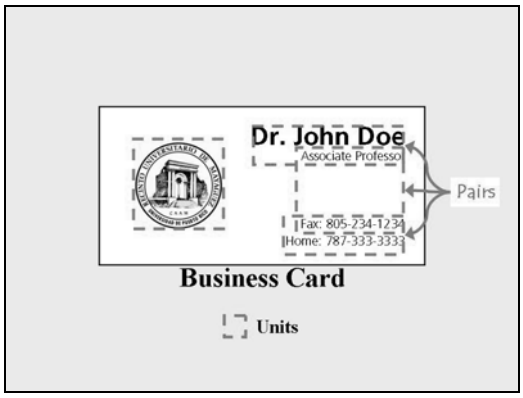

Figure 5: Potential Logical units after splitting

Next, the distance between components is measured. The system selects pairs of units that belong to the same strong line and share a common component. The distances between the common component and its partners are measured for both potential units. The unit with the smaller distance keeps the component and the other unit it is dissolved since a component cannot belong to more than one logical unit. If the calculated distances are equal, the units are merged together. Figure 6 shows the final units for the example.

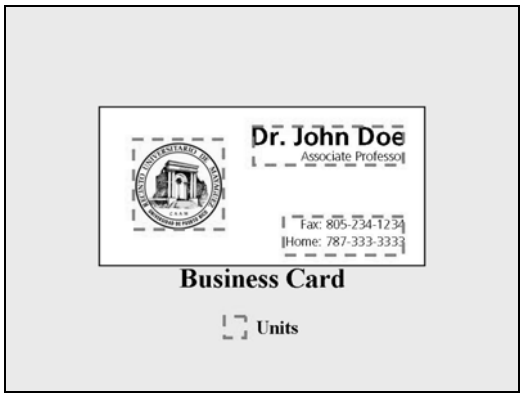

Figure 6: Final Logical units

The example business card is composed of five components that the system grouped in three different logical units and which humans can readily identify.

4.4. Assigning logical types to components

Designers, using the contrast principle, make components different to establish certain types of relations, giving different meanings and importance to components. For our proof of concept we constrained the assignment to headings, lists, and paragraphs, using logical units, patterns and contrast. An example of a document is shown in Figure 7.

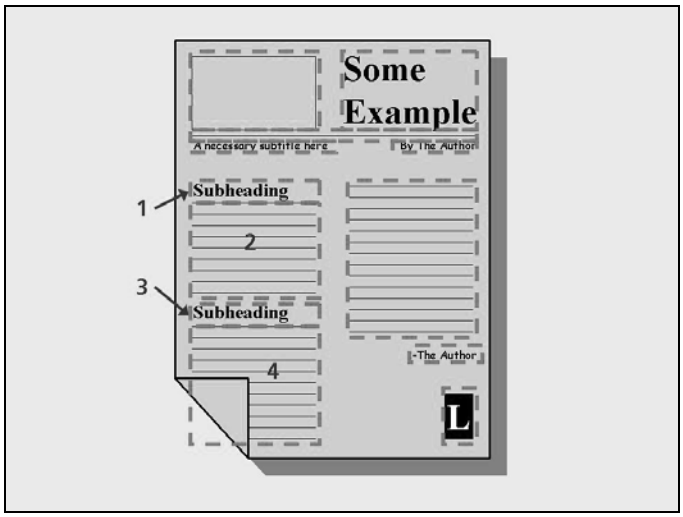

Figure 7: Logical type - document after logical unit assignments

The first step selects three units with the same strong line. Consider the units numbered 1 through 4 in Figure 7. The system can select two sets of three units each: [2, 3, 4] and [1, 2, 3]. Considering, for example, the set [2, 3, 4], the system detects in unit 3 a single line of text, with increased text size, and a change in typeface family and weight. In addition, the distance from unit 3 to unit 2 is larger than distance from unit 3 to unit 4. By the proximity principle, it can be assumed that unit 3 is more closely related to unit 4 than to unit 2. These design principles lead to establish that unit 3 is a heading for unit 4. The system tags the component in unit 3 as a heading and merge unit 3 and 4. The system continues to analyze the components in unit 4. If the components are single sentence components distributed vertically, the group of components is tagged as a list. If the components are composed of several sentences each component is tagged as a paragraph. This process exploits the properties of the repetition principle and is repeated for the remaining units. When there is no group left with three units the system analyzes groups of two units.

5. Results and Analysis

The document segmentation is an important piece inside the knowledge-based artifact recognition tool. Its analysis provides substantial data about the document layout to allow the case-base artifact recognition system (see Figure 1: K-BAR System Architecture) to match anomalies with known artifacts. The experiments done for the document segmentation and understanding module were aimed at verifying a correct operation of this module instead of benchmarking the performance of the system.

Strong lines were the point of reference to start the segmentation analysis. We tested 36 different layout samples to verify the proper identification of the strong lines in each layout. The samples were composed of a maximum of six components, with different alignments, and no overlapping between components. The system detected correctly the strong lines in every layout tested. Figure 8 shows four examples for the assignment of strong lines in three different layouts with specific component alignments.

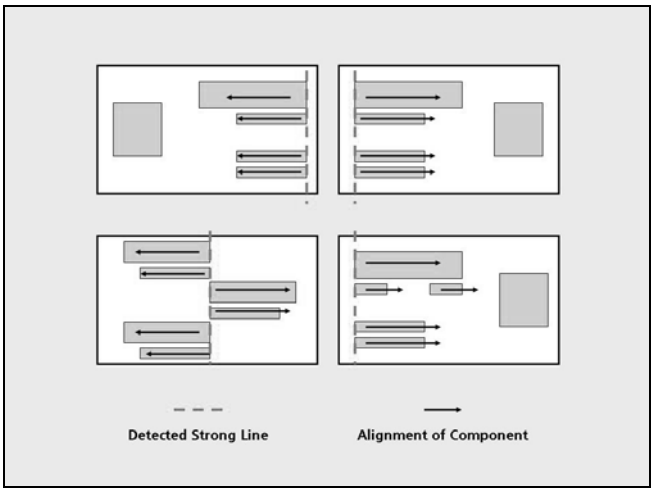

Figure 8: Representative examples of detected strong lines

To assure the correct assignment of logical units we used 43 test samples. These test samples are the same test set for the artifact recognition system. The samples consist of diverse types of jobs, signs, brochures, essays, bumper stickers, and business cards. From these job types the signs, brochures, and essays categories have the most complex layouts, because they contain paragraphs, lists, and headings. Figure 9 shows an example of an essay job type with a correct classification of the headings and paragraphs. A correct execution of this module will ensure a proper performance in the artifact recognition tool. The system detected effectively the headings, the paragraphs and the relation between them. In addition the system assigned effectively equally spaced components to the same unit. On other job types (e.g. business cards) when there is not enough information to classify the component as a heading or paragraph the system assigned a more general type such as text or image.

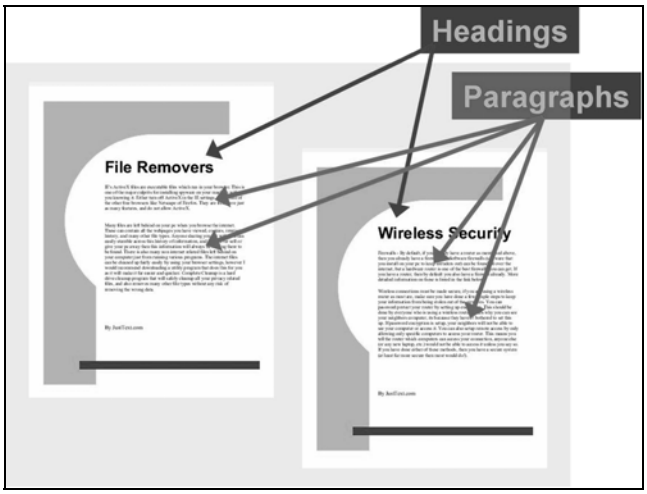

Figure 9: Example of logical type assignments in actual test sample

The document segmentation and understanding module needs additional enhancements to its capabilities in order to process real life document samples. For example the system should recognize horizontal strong lines and separate components in the foreground from those in the background. For the current test set all the strong lines were detected effectively and the components logical type and logical units were assigned correctly.

6. Conclusions

Design principles form an important frame of reference for document segmentation and understanding. This paper described a system with two modules. The first module carried out the extraction of the geometric properties of the document page layout and context from XSL-FO files. The second module, using a rule-based approach, applied graphic design principles to segment and merge document components into logical units. The final task of the segmentation unit was to identify logical types and establish relations such as heading-paragraph, figure-caption, etc.

This current system is part of an automated artifact recognition system for digital publishing where we have initially limited the system to a subset of XSL-FO but can be easily extended to larger and richer subsets of this language.

Design principles as applied in this work, reveal implicit knowledge inside the document. Future implementations and improvements should add additional operations to the geometric analysis and the rule engine to extend the functionality of the document segmentation and understanding tool. Additionally, the methodology proposed and implemented is a sound and novel technique for document segmentation and understanding. This technique can also be used and integrated to document recognition systems or information retrieval systems, where the logical units and types serve as the basic units of analysis for feature extraction, for example, for semantic annotation for digital libraries or applications in the Semantic Web.

7. Acknowledgements

This work is part of the Digital Publishing Research Program in Collaboration with Purdue University and is partly funded by Hewlett Packard.

8. References

[1] Friedman-Hill E., “Jess in Action: Rule-based Systems in Java”, Manning Publications, 2003, pages xxi, 15-16.

[2] Holman G. K., “Definitive XSL-FO”, Prentice Hall, 2003.

[3] Jackson P., “Introduction to Expert Systems”, Addison-Wesley, 1999, 3rd Edition, pages 4-8.

[4] Collier D., “Collier’s Rules for Desktop Design and Typography”, Addison-Wesley, 1991.

[5] Santos-Villalobos H. *Style-Dependent Artifact Recognition for Digital Variable Data Printing*. M.Sc. Thesis. Department of Electrical and Computer engineering, University of Puerto Rico. Mayagüez, Puerto Rico. July 2005.

[6] White A., “The Elements of Graphic Design”, Allworth Press, 2002.

[7] Williams R., “The Non-Designer’s Design Book”, Peachpit Press, 2nd Edition, 2004.

[8] Politis, A. Digital printing: finishing technologies making digitally printed documents professional. XML Europe 2000 Conference. 12-16 June 2000. Paris, France.

[9] S. J. Harrington, J. F. Naveda, R. P. Jones, P. Roetling, and N. Thakkar, “Aesthetic Measures for Automated Document Layout”, Proceedings of the 2004 ACM symposium on Document engineering, 2004, Milwaukee, Wisconsin USA, Pages: 109 – 111.

[10] Kleper M. L., “The Handbook of Digital Publishing”, Graphic Dimensions 2001, Prentice Hall, Vol. 1, Page XXVII – XXXVI.

[11] H. Chao and J. Fan, “Layout and Content Extraction for PDF Documents”, DAS 2004, IAPR Int. Workshop on Document Analysis Systems, Florence, Italy, 8-10 Sept. 2004.

[12] Topping S. M., “Graphic design and color in today’s office”, KODAK Publication, W-628, 1990.