DocMining: A Cooperative Platform for Heterogeneous Document Interpretation According to User-Defined Scenarios

Eric Clavier[1], Gérald Masini[2], Mathieu Delalandre[3], Maurizio Rigamonti[4], Karl Tombre[2], and Joël Gardes[1]

[1] France Télécom R&D, 2 Avenue Pierre Marzin, 22307 Lannion Cedex, France
rntl.ball001@rd.francetelecom.com, Joel.Gardes@rd.francetelecom.com
[2] LORIA, B. P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France
Gerald.Masini@loria.fr, Karl.Tombre@loria.fr
[3] PSI Laboratory, Université de Rouen, 76821 Mont Saint Aignan, France
Mathieu.Delalandre@univ-rouen.fr
[4] DIUF, Université de Fribourg, Chemin du musée, 31700 Fribourg, Switzerland
Maurizio.Rigamonti@unifr.ch

Abstract. The DocMining platform is aimed at providing a general framework for document interpretation. It integrates document processing units coming from different sources and communicating through the document being interpreted. A task to be performed is represented by a scenario that describes the units to be run, and each unit is associated with a contract that describes the parameters, data and results of the unit as well as the way to run it. A controller interprets the scenario and triggers each required document processing unit at its turn. Documents, scenarios and contracts are all represented in XML, to make data manipulation and communications easier.

1 Introduction

The design of document analysis systems requires the integration of various components and algorithms (image processings, character and symbol recognition, interpretation tasks, etc.), especially when it has to deal not only with textual components but also with graphics and images. If the aim is the design of a versatile system, the components must be able to cooperate in a flexible way and the domain knowledge must be easy to integrate into the system.

The classical way to process a textual document consists in segmenting the document into homogeneous zones to apply character recognition methods followed by "linguistic" post-processing steps: Dictionary lookup, application of language models, etc. There is good know-how in building such systems [2], as well as systems, like smartFIX, dedicated to more general "business documents" [7].

Mature systems are available for specific domains, such as check processing [10], recognition of tables [17], or recognition of forms [13]. However, especially when graphics-rich documents are concerned, most of the working systems designed up to now appear to be specific to a given application area. For example, the system by IBM Italy for the Italian land register maps encapsulates all the domain knowledge in its core [3]. Arias *et al.* also describe a system for the analysis of manhole drawings for a telephone company [1], whereas Dosch *et al.* propose an interactive system for architectural drawings analysis [9]. Some systems are designed to analyze drawings made of symbols and connecting lines [19], and the MAGELLAN system exploits the connections of legends to extract information from maps [16]. Only very little work has been done to build generic tools. In most cases, this implies that the system only offers low-level tools, or that it supplies complex and exhaustive knowledge representation schemes. The work by Pasternak [14], based on a hierarchical and structural description coupled with triggering mechanisms, and the DMOS method, based on syntactical techniques [4], belong to this category.

In this paper, we present an approach that may seem less ambitious, as we do not aim at representing all the domain knowledge, but that is probably more realistic when the purpose is to be able to process a large number of heterogeneous documents and to allow users to define their own knowledge. This approach is based on strategic knowledge acquisition and instrumentation. The knowledge describes the tools to be used to extract objects from documents and the chaining relations between these tools [15]. Scenarios, *i.e.* sequences of calls to *document processing units* (DPU's), may subsequently be constructed and executed, using available libraries of such units.

More precisely, the paper describes the DocMining platform that provides a general framework for document interpretation. The platform architecture is plug-in oriented, so that users can conveniently integrate new processing units. Document visualization and manipulation tools are designed according to the same principle, so that a user is able to fully customize the interactions with the platform. DPU's communicate through the document to be processed, that contains not only graphical objects and data, but also traces of the execution of the DPU's, in order to avoid the problems of data scattering usually met in classical document processing chains. Running a scenario results in collecting some user's experience, that becomes part of the scenario itself. The scenario may then be transformed into a new DPU corresponding to a higher-level granularity.

As a user can create his own objects, integrate his own DPU's into the platform, design his own interfaces and define his own scenarios, all the users share and exchange knowledge through the platform. In this way, the platform may be used for various purposes:

- Domain specific platform design: Document processing units available for a particular kind of document can be integrated into the platform as plug-in's. The domain knowledge is taken into account through domain specific scenarios and may be manipulated by interfaces especially defined by users.
- Evaluation of the results quality and of the influence of parameters and thresholds tuning, thanks to the trace recorded during the execution.
- Evaluation of the efficiency of a processing unit according to the nature of the input documents.

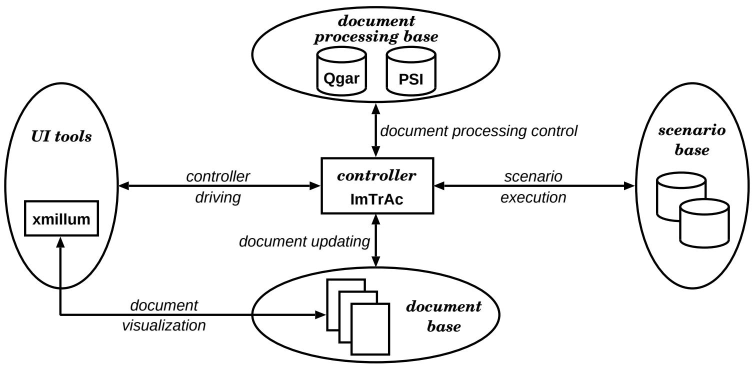


Fig. 1. The DocMining platform architecture.

- Benchmarking: Such a task consists in a predefined scenario describing a full processing chain. Each process to be experimented is associated with a so-called contract and each corresponding algorithm is implemented according to the specifications of its contract. When the scenario is run, the efficiency of the processing units can be directly measured.
- Experimentation of various implementations of a same algorithm: As users share the same integration framework, they can exchange processing units. They do not have to re-implement units already implemented by partners.

This project is supported by the DocMining consortium, including four academic partners, PSI Lab (Rouen, France), Project Qgar (LORIA, Nancy, France), L3I Lab (La Rochelle, France), DIUF Lab (Fribourg, Switzerland), and one industrial partner, GRI Lab from France Télécom R&D (Lannion, France). It is partially funded by the French Ministry of Research, under the auspices of RNTL (*Réseau National des Technologies Logicielles*).

2 Overview of the DocMining Platform

The platform architecture includes five main components (Fig. 1). The document base contains images of documents associated with metadata, the so-called document structures. A document structure represents the structure of the corresponding document at a given time, as well as information expressing how this structure has been produced. Users can define their own document structures, according to their needs.

The document processing base provides the ability to (re)use document processing units from various (remote) libraries. The platform makes them interoperate by providing a common integration framework based on the notion of contract, defined between each DPU and the document structure. Any user interface connected to the platform may access and manipulate the whole document structure or part of it, as all the users' interactions with the document are defined according to the same plug-in approach.

An interpretation task is represented by a scenario expressing the user's strategic knowledge, *i.e.* the parameters of the processing units, and the objects and data manipulated by the different units. The controller is responsible for running scenarios, updating the document structure, simulating processings, and so on. Its main task consists in controlling how processing units and scenarios access the document structure. It selects the required document elements to be processed at each step of a scenario, and updates the global document structure when each process is completed. When a processing unit is run, the controller transmits the required document elements and parameters to the unit. On the one hand, using a controller guarantees that all changes in a document structure are recorded and then may be utilized as a basis for building new scenarios. On the other hand, this ensures that a processing unit has no direct access to the document structure. The parts of the document structure that the unit has to access are specified by its contract.

Concerning document processing design, our contract-based approach has the advantage of being implementation independent, as users can change DPU implementations without modifying the corresponding contracts. Moreover, since the contract of each DPU is defined in a normalized way, all the platform users communicate with DPU's in the same way, especially to provide them with their parameter values.

3 The Platform Implementation

The platform implementation is based on a Java/XML architecture and relies on four major components.

The PSI Library integrates different research works of the PSI Laboratory and includes chains of processing units using statistical and/or structural approaches [5]. It is divided into three sub-libraries dedicated to image processing (mathematical morphology, blob coloring, feature extraction, etc.), classification (similarity search, model reconstruction, etc.), and XML data management. They are written in different languages (Java, C/C++, as well as XSLT stylesheets or Quilt query scripts) and can be run on Windows or Linux operating systems.

The Qgar software system[5] is developed by the same-named project for the design of document analysis applications [8] and includes three parts written in C++. QgarLib is a library of C++ classes implementing basic graphics analysis and recognition methods. QgarApps is an applicative layer, including high-level applications (text-graphics separation, thick-thin separation, vectorization, etc.). Applications may be designed and run using an interactive graphical interface called QgarGUI, that incorporates data manipulation and display capabilities. Any application, either designed apart from Qgar or not, may be integrated into the system by simply wrapping it with a predefined C++ class, without recompiling any part of the system itself.

[5] http://www.qgar.org/

```
<object_doc source="icdar2001.pdf" type="PdfDoc">
    <object_position h="842" w="595" x="0" y="0"/>
    <object_doc type="TextLine">
        <object_position h="17" w="309" x="144" y="104"/>
        <object_data>
            <ascii_data>Web-Based Cooperative Document Understanding</ascii_data>
        </object_data>
    </object_doc>
    <object_doc type="TextLine">
        <object_position h="12" w="225" x="62" y="267"/>
        <object_data>
            <ascii_data>This paper presents our ongoing work on the design of</ascii_data>
        </object_data>
    </object_doc>
    ...
</object_doc>
```

Fig. 2. The structure of a textual (PDF) document: Lines of text are represented by `TextLine` objects, provided with a location and a line content (as extra data).

xmillum (XML illuminator) is a framework for cooperative and interactive analysis of documents developed by the DIVA group at the University of Fribourg [11]. It is based on the philosophy that a document recognition system does not work in a fully automatic way, but cooperates with the user to incrementally learn from its feedback and to adapt itself to different document properties[6]. This approach implies that xmillum offers complex visualization and editing capabilities. Moreover, xmillum is independent from a particular document recognition system, as it does not require any specific language. There is a unique constraint: The original data must be represented in XML. Using an XSLT stylesheet, it is subsequently transformed into a xmillum specific structure represented with objects (defining how to visualize a data element), styles (specifying the rendering properties of the objects), handlers (encapsulating the interactions with the objects), selectors (indicating to handlers which objects are involved in an action), tools (that are specific modules extending xmillum capabilities, such as multiple views of a document), and layers (that are the data containers). The xmillum editor interprets this structure, but does not contain any implementation. It uses external modules to visualize or edit data, which makes xmillum freely extensible.

At last, the ImTrAc package, developed by the GRI Lab at France Télécom R&D Lannion, provides a processing engine, to control process running, and a scenario engine, to control scenario running, as well as tools to integrate new processing units in the system and to create scenarios.

3.1 Documents

A document is represented by an XML tree constructed according to an XML schema (Fig. 2). Basic elements are graphical objects defined by their type

[6] The so-called "CIDRE" (Cooperative and Interactive Document Reverse Engineering) philosophy is supported by the Swiss National Fund for Scientific Research, code 2000-059356.99-1

(`Binary Image`, `Connected Component`, `Text Block`...), their source (the document they come from), and their location in the image. We did not try to elaborate a complete predefined taxonomy of possible types: A user can define his own graphical object types, when necessary. A graphical object includes intrinsic data describing the way the object is physically represented, using an XML schema defined from basic data types such as `Freeman Chain`, `Feature Vector`, etc. Just as previously, a user can define his own data types.

However, a document is more than a simple description in terms of graphical objects and data. Its structure also contains information (name, parameters...) about processing units that have been applied to the document and that have yielded the objects. Objects included in the document structure are visualized using xmillum, thanks to XSLT stylesheets that define which objects may be visualized, how they are visualized, and how events involving objects (mouse clicks, for example) are handled. Each object is associated to a Java class that performs the rendering.

3.2 Document Processing Units

As previously said, a DPU has no direct access to the document structure and cannot modify it unless a so-called contract, defined according to an XML schema, has been established with the document. This contract describes the unit behavior: The way the processing modifies the XML document structure (by adding, removing, or updating nodes), the kind of graphical objects it produces, as well as parameters that do not require access to the document structure. The objects that a DPU may modify or access are defined by specifying a "trigger" node (that enables the execution of the corresponding DPU) and "updated" nodes (that are modified by the DPU). This means that the node that launches a DPU is not necessarily the one that is modified by the DPU. This distinction gives more flexibility to the design of DPU's, as a user may associate conditions with launching nodes (to require the presence of a particular kind of object, for example) and updated nodes. At the present time, various separate applications from the QgarApps and PSI libraries can be run by the DocMining platform.

3.3 Scenarios

In order to achieve interpretation tasks, users can interactively construct scenarios, that are defined as structured combinations of DPU's. There are two ways of creating a scenario. One is based on the contracts: As objects input and output by units are specified in the corresponding contracts, it is possible to determine which unit can feed a given unit and then to combine them.

The other way relies on a xmillum component that has been especially developed for the DocMining platform. It provides means to interact with the processing engine (ImTrAc) and to visualize the structure of a document. For each object of the document, the engine is able to supply the list of DPU's that may be applied. Once the user has chosen a DPU, the engine supplies the parameter list of the unit to the user, so as to get the corresponding values and to be able to launch the DPU. When the DPU terminates, ImTrAc updates the document structure and the user can then interact with the newly created objects. Each user action on the document is recorded in the scenario, that may later be applied to another document.

4 A Scenario for Mixed Text/Graphics Documents

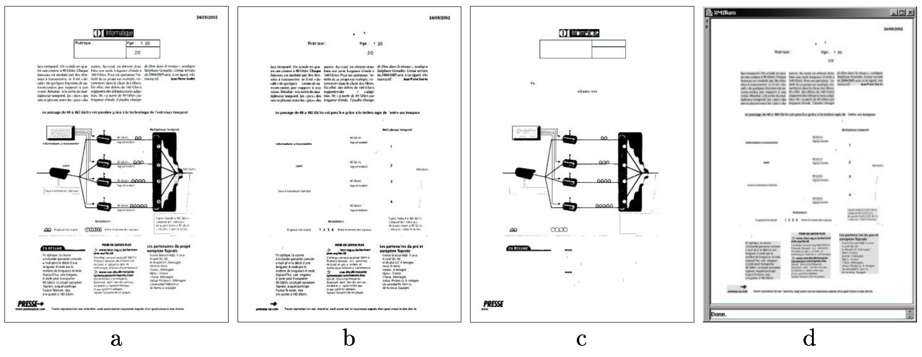
The DocMining platform has already been used to produce scenarios dedicated to various domains, like page segmentation evaluation or production of ground-truth data sets. The scenario presented here in detail performs the analysis of documents with a mixed text-graphics content and may be used for different purposes. It may become part of a more general document interpretation system, and be used to evaluate the robustness of an algorithm in case of noisy input images (leading to text-graphics separation errors). It may also be used as a benchmarking tool: When a developer implements a particular step of the scenario, he may run the different available processing units to evaluate the efficiency of his implementation.

4.1 Scenario Overview

In fact, the aim of the scenario is the separation of the graphical part of a document from its textual part, which can be briefly described as three main steps: Firstly, separate the graphical layer from the textual layer; secondly, perform a segmentation into text blocks on the textual layer, and then apply an OCR process on the resulting blocks to produce ASCII texts; finally, perform a vectorization on the graphical layer, to get graphical primitives represented in SVG or DXF format. The efficiency of such a scenario deeply depends on the efficiency of the text-graphics separation step. If some of the image components are mislabeled or not labeled during this step, further processing units cannot be correctly performed. It is therefore necessary to add an extra step dedicated to the correction of labeling errors. The final steps are ordered as follows:

1. Binarize the image.
2. Perform text-graphics separation.
3. Perform segmentation into text blocks on the textual layer.
4. Perform segmentation into connected components on the graphical layer.
5. Perform segmentation into connected components on the parts that have not been classified as graphical or textual parts.
6. Correct text-graphics separation errors by a cross-examination of the results of these three segmentations.
7. Perform OCR on the text blocks of the resulting textual layer.
8. Perform vectorization on the resulting graphical layer.

Figure 3 shows the different steps and the objects manipulated during the execution. This kind of scenario differs from classical sequential processing chains, that do not include switches like those of steps 2 and 6.



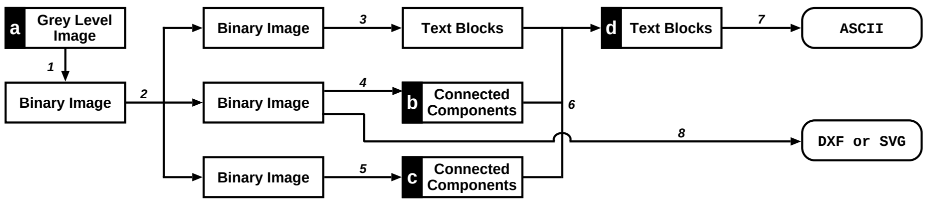


Fig. 3. The scenario for text-graphics separation. A number corresponds to a scenario step, a rectangle to the object that is produced, and a letter to an image: Initial image (a), graphical (b) and textual (c) layers resulting from text-graphics separation, final text blocks visualized by xmillum (d).

4.2 The Processing Units

Processing units involved in the scenario are taken from the libraries connected to the platform. *Binarization* may be performed using one of four available units. The PSI library implements two standard algorithms for automatic computation of binarization thresholds. One is histogram-based, the other is clustering-based [12]. The QgarApps library supplies an adaptive algorithm proposed by Trier and Taxt with some minor adaptations [18]. The last processing unit implements a raw method: The binarization threshold is given by the user.

Text-graphics separation is based on a method initially proposed by Fletcher and Kasturi and implemented by the Qgar Project, with an extra absolute threshold for the size of a text component [18].

For *text block segmentation*, two processing units are available at the present time: One is based on a classical top-down approach, the other on an hybrid approach [10]. The latter works by detecting pieces of text lines in small overlapping columns. These pieces are then merged by a bottom-up algorithm to form complete text lines and blocks of text lines.

The method for *connected component segmentation* is provided by the PSI image processing (sub-)library, that includes some applications based on a standard blob coloring algorithm: Image labeling, component analysis, and occlusion extraction [5, 6].

The purpose of the last processing unit, *correction of text-graphics separation errors*, is to adequately assign mislabeled or unlabeled connected components of the graphical layer to the textual layer. Each connected component of the graphical layer located inside a text block of the textual layer is transfered to the textual layer.

4.3 Results

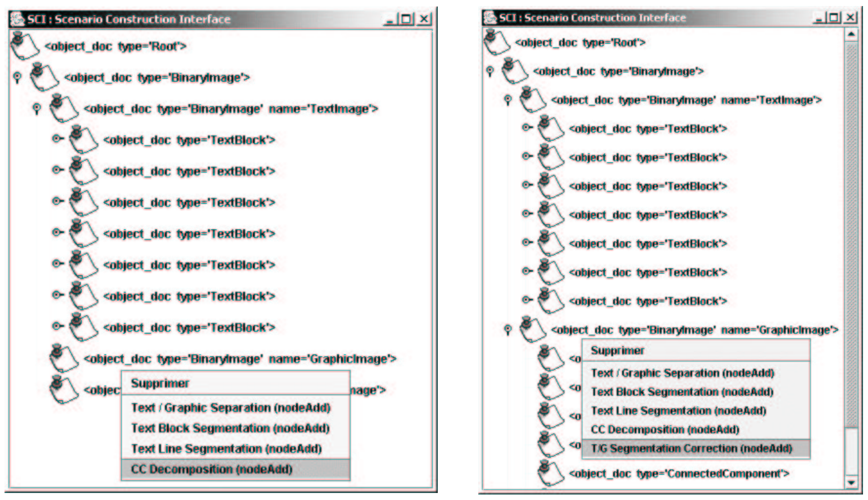
The execution of the scenario is partly illustrated by Figure 3. From an initial gray level image (cf. Fig. 3a), a text-graphics separation method (step 2) provides three new images representing the graphical layer (cf. Fig. 3b), the textual layer (cf. Fig. 3c), and the layer of unclassified parts (not shown here). The cross-examination of the text-blocks extracted from the textual layer and of the connected components extracted from the two other layers (step 6) allows a complete identification of the text blocks present in the initial image (cf. Fig. 3d). OCR (step 7) and vectorization (step 8) tasks are then respectively applied to the text blocks and to the graphical layer to get a full segmentation of the initial image into graphical and textual primitives.

5 Constructing the Scenario

The construction of a scenario is performed with an interactive tool called SCI (Scenario Construction Interface), that has been developed to drive the ImTrAc controller. The construction relies on a structural combination of DPU's belonging to the processing base. As explained in section 3.3, the combination is made possible by associating each DPU with a contract describing its behavior. Thanks to this very notion of contract, the elaboration of a scenario does not require any processing unit to be run.

Figure 4 shows screenshots displaying a partial view of a document tree structure: The popup menu shows authorized processing units for the currently selected node (*i.e.* the activation node). We can see that the correction of text-graphics separation errors is not possible (the corresponding processing unit does not appear in the popup menu of Fig. 4a) unless a connected component segmentation has been performed before (cf. Fig. 4b: This time, the processing unit appears in the popup menu). This implies that the coherence of a scenario is always validated during the construction itself of the scenario.

Figure 5 shows the contract defined for the DPU that performs the corrections of text-graphics separation errors. The noticeable aspects of the contract, as explained in section 3.2, can be seen: The service (tag `<service/>`), the produced objects (tag `<produced_object/>`), and the handled objects (tag `<xpath_exp/>`). The contract is not just an XML translation of a classical configuration file. In fact, it includes "processing instructions" as XPath expressions, like the one on line 8 (in italic shape), which indicates which elements the DPU needs. The ImTrAc engine interprets the Xpath expression, extracts the resulting element set from the document and transmits it to the DPU. The notion of


SCI : Scenario Construction Interface
Supprimer
Text / Graphic Separation (nodeAdd)
Text Block Segmentation (nodeAdd)
Text Line Segmentation (nodeAdd)
CC Decomposition (nodeAdd)
SCI : Scenario Construction Interface
Supprimer
Text / Graphic Separation (nodeAdd)
Text Block Segmentation (nodeAdd)
Text Line Segmentation (nodeAdd)
CC Decomposition (nodeAdd)
T/G Segmentation Correction (nodeAdd)

```
<object_doc type='Root'>
  <object_doc type='BinaryImage'>
    <object_doc type='BinaryImage' name='TextImage'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
    <object_doc type='BinaryImage' name='GraphicImage'>
    <objec                                nage'>
```

a

```
<object_doc type='Root'>
  <object_doc type='BinaryImage'>
    <object_doc type='BinaryImage' name='TextImage'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
      <object_doc type='TextBlock'>
    <object_doc type='BinaryImage' name='GraphicImage'>
      <o
      <o
      <o
      <o
      <object_doc type='ConnectedComponent'>
```

b

Fig. 4. The correction of text-graphics separation errors (b) is possible if and only if it is preceded by a connected component segmentation (a).

contract thus allows the delegation of part of a processing unit and increases the flexibility of the platform architecture. It provides a method for adapting users' data to processing units rather than adapting processing units to users' data. DPU's may be triggered from any node, may modify any node of the structure, and may take any node as a parameter, with very precise selection criteria.

The SCI tool, that drives the ImTrAc engine, has actually been integrated as a component of xmillum to be used as an interface to visualize and manipulate the document structure, as well as to construct and run scenarios. In particular, users can conveniently apply selected processing units to the document structure and, having visualized the results, tune the parameters of the units to improve the results quality. The way documents and document structures are visualized depends on styles described by XSLT scripts (cf. Fig. 3d for an example).

6 Conclusion

DocMining is not an interpretation system but a framework. It is a first step towards the definition of a flexible and reusable system to be used as a basis to develop applications related to specific domains or to design an interpretation system able to run complex scenarios requiring to combine processing units possibly coming from different sources (*e.g.* research projects). Among its main advantages, it lets users free in their choices about the information structure manipulated by the system as well as about the implementation of the processing units. These ideas has been successfully experimented and validated using scenarios to segment documents into textual and graphical parts.

```
<process_property class_name="docmining.sox.extern.OrphanMerger">
  <service name="nodeAdd">
    <handled_object>
      <xpath_exp>./object_doc[@type="ConnectedComponent"]</xpath_exp>
      <process_config>
        <param type="Input" name="TextImage" support="ObjectDoc"
          param_value="//object_doc[@name="TextImage"]" />
        <param type="ParamIn" name="Orphans" support="ObjectDoc"
          param_value="./object_doc[@type="ConnectedComponent"]"/>
        <param type="ParamIn" name="ThreshFactor" support="Data" param_value="2"/>
      </process_config>
      <produced_object>
        <object_doc type="TextBlock">
          <object_doc type="ConnectedComponent"> <object_info type_list='yes'/> </object_doc>
          <object_info type_list='yes'/>
        </object_doc>
      </produced_object>
    </handled_object>
  </service>
</process_property>
```

Fig. 5. The contract of the processing unit that performs the corrections of the text-graphics separation errors.

Acknowledgments The authors wish to especially thank Oliver Hitz and Rolf Ingold (DIUF Lab, Université de Fribourg), Jean-Marc Ogier (L3I Lab, Université de La Rochelle), Florent Le Dret (Université de La Rochelle), Jan Rendek (Qgar Project, LORIA, Nancy), Yves Lecourtier and Eric Trupin (PSI Lab, Université de Rouen), as well as all the other participants in the DocMining Consortium for their contribution to the work described by this paper.

References

1. J.-F. Arias, C.P. Lai, S. Surya, R. Kasturi, and A.K. Chhabra. Interpretation of telephone system manhole drawings. *Pattern Recognition Letters*, 16(1):355–359, 1995.
2. H.S. Baird. Anatomy of a versatile page reader. *Proceedings of the IEEE, Special Issue on OCR*, 80(7):1059–1065, 1992.
3. L. Boatto, V. Consorti, M. Del Buono, S. Di Zenzo, V. Eramo, A. Esposito, F. Melcarne, M. Meucci, A. Morelli, M. Mosciatti, S. Scarci, and M. Tucci. An interpretation system for land register maps. *IEEE Computer Magazine*, 25(7):25–33, 1992.
4. B. Coüasnon. DMOS: A generic document recognition method. application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems. In *Proceedings of 6th International Conference on Document Analysis and Recognition, Seattle (USA)*, pages 215–220, 2001.
5. M. Delalandre, S. Nicolas, E. Trupin, and J.-M. Ogier. Symbols recognition by global-local structural approaches, based on the scenarios use, and with a XML representation of data. In *Proceedings of 7th International Conference on Document Analysis And Recognition, Edinburgh (Scotland)*, 2003.
6. M. Delalandre, Y. Saidali, J.-M. Ogier, and E. Trupin. Adaptable vectorization system based on strategic knowledge and XML representation use. In *Proceedings*

of 5th IAPR International Workshop on Graphics Recognition, Barcelona (Spain), 2003.

7. A.R. Dengel and B. Klein. smartFIX: A requirements-driven system for document analysis and understanding. In D. Lopresti, J. Hu, and R. Kashi, editors, *Proceedings of 5th IAPR International Workshop on Document Analysis Systems, Princeton (New Jersey, USA)*, volume 2423 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, Berlin, 2002.
8. Ph. Dosch, C. Ah-Soon, G. Masini, G. Sánchez, and K. Tombre. Design of an integrated environment for the automated analysis of architectural drawings. In S.W. Lee and Y. Nakano, editors, *Document Analysis Systems: Theory and Practice*, volume 1655 of *Lecture Notes in Computer Science*, pages 295–309. Springer-Verlag, Berlin, 1999.
9. Ph. Dosch, K. Tombre, C. Ah-Soon, and G. Masini. A complete system for analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, 3(2):102–116, 2000.
10. N. Gorski, V. Anisimov, E. Augustin, O. Baret, and S. Maximov. Industrial bank check processing: The A2iA CheckReader. *International Journal on Document Analysis and Recognition*, 3(4):196–206, 2001.
11. O. Hitz, L. Robadey, and R. Ingold. An architecture for editing document recognition results using XML. In *Proceedings of 4th IAPR International Workshop on Document Analysis Systems, Rio de Janeiro (Brazil)*, pages 385–396, 2000.
12. A. Lassaulzais, R. Mullot, J. Gardes, and Y. Lecourtier. Segmentation d'infrastructures de réseau téléphonique. In *Colloque International Francophone sur l'Écrit et le Document, Québec (Canada)*, pages 188–197, 1998.
13. D. Niyogi, S.N. Srihari, and V. Govindaraju. Analysis of printed forms. In H. Bunke and P.S.P. Wang, editors, *Handbook of character recognition and document image analysis*, pages 485–502. World Scientific, 1997.
14. B. Pasternak. *Adaptierbares Kernsystem zur Interpretation von Zeichnungen*. Dissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.), Universität Hamburg, 1996.
15. Y. Saidali, S. Adam, J.-M. Ogier, E. Trupin, and J. Labiche. Knowledge representation and acquisition for engineering document analysis. In *Proceedings of 5th IAPR International Workshop on Graphics Recognition, Barcelona (Spain)*, 2003.
16. H. Samet and A. Soffer. MAGELLAN: Map Acquisition of GEographic Labels by Legend ANalysis. *International Journal on Document Analysis and Recognition*, 1(2):89–101, 1998.
17. J.H. Shamilian, H.S. Baird, and T.L. Wood. A retargetable table reader. In *Proceedings of 4th International Conference on Document Analysis and Recognition, Ulm (Germany)*, pages 158–163, 1997.
18. K. Tombre, C. Ah-Soon, Ph. Dosch, A. Habed, and G. Masini. Stable, robust and off-the-shelf methods for graphics recognition. In *Proceedings of 14th International Conference on Pattern Recognition, Brisbane (Australia)*, pages 406–408, 1998.
19. Y. Yu, A. Samal, and S. C. Seth. A system for recognizing a large class of engineering drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):868–890, 1997.