Chapter 5

A GENERIC APPROACH FOR PEN-BASED USER INTERFACE DEVELOPMENT

Sébastien Macé and Éric Anquetil
IRISA - INSA, Campus Universitaire de Beaulieu, F-35042 Rennes Cedex (France)
E-Mail: {sebastien.mace, eric.anquetil}@irisa.fr – Web: www.irisa.fr/imadoc
Tel.: + 33 2 99 84 {7546, 7238} – Fax: +33 2 99 84 71 71

Abstract Pen-based interaction is an intuitive way to realize hand drawn structured documents, but few applications take advantage of it. Indeed, the interpretation of the user hand drawn strokes in the context of document is a complex problem. In this paper, we propose a new generic approach to develop such systems based on three independent components. The first one is a set of graphical and editing functions adapted to pen interaction. The second one is a rule-based formalism that models structured document composition and the corresponding interpretation process. The last one is a hand drawn stroke analyzer that is able to interpret strokes progressively, directly while the user is drawing. We highlight in particular the human-computer interaction induced from this progressive interpretation process. Thanks to this generic approach, three pen-based system prototypes have already been developed, for musical score editing, for graph editing, and for UML class diagram editing.

Keywords: Hand drawn symbol recognition, Pen-based interaction, Structured document interpretation, User interface development.

1. INTRODUCTION

As pen-based interfaces are in wide expansion, there is a lack of applications taking advantage of this intuitive and ergonomic way to draw structured documents, such as electronic figures, plans, musical scores, tables, diagrams, etc. Fig. 1 presents an example of a tablet PC with such an interaction: the user composes musical scores in a traditional way by drawing the symbols on the screen: the use of a pen and a touch screen reproduces the "pen-paper" metaphor. This editing application is an example of system developed thanks to the methodology we present in this paper.

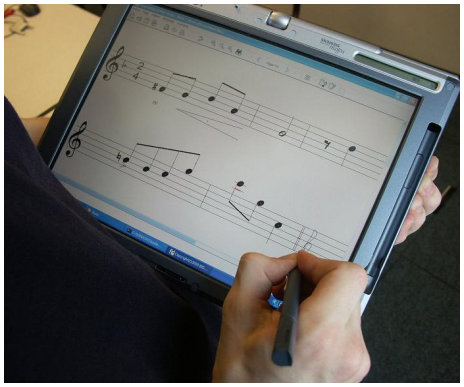

Figure 1. Example of a tablet PC with a pen-based interface.

This paper deals with the interpretation of hand drawn structured documents. The interpretation of the user *strokes*, which are the sequences of points captured by the touch screen between a pen-down and a pen-up, is called *on-line interpretation*. It is a complex problem of pattern recognition due to the variability of handwritten styles. This interpretation process is even more complicated in the context of structured document analysis, because these documents are constituted of many elements of various natures. Moreover, the same stroke can have different meanings according to the context in which it has been drawn, which must therefore be taken into account [1,9]. This justifies why few pen-based systems have been developed.

On-line interpretation can be either *lazy* [12] (*i.e.* occurring only when explicitly requested by the user) or *eager* [5] (*i.e.* occurring directly while the user is drawing). Lazy interpretation offers the advantage of not intruding into the user's creative phase during the document realization. Nevertheless, once the recognition process applied at once on all of his strokes, he has to examine the entire document to look for possible incorrect interpretations. Besides, it turns out that lazy systems are so far not robust enough and make too many mistakes, which reduces their usability. We believe that lazy recognition is a promising approach to offer unconstrained understanding of ink, but the difficulties to design automatic parsing coupled with a robust hand drawn shape recognition system show that it remains an open problem. Eager interpretation constitutes then another way to consider on-line structured document analysis. Every time the user draws a stroke, the system interprets it immediately; it then has to deal with documents in the process of realization and to make a decision as quickly as possible so that the user does not have to wait to continue his drawing. Moreover, it is possible to exploit the interaction with the user, who is aware of the system answers, and can then validate or reject them progressively. We believe that eager interpretation is a pertinent compromise, since these systems are often more ro-

bust and more efficient than lazy ones, which makes them more usable. The work we present in this paper is in the eager interpretation context: it aims at defining generic methods to interpret structured documents.

Most of the existing systems are adapted to analyze the elements of documents from a given domain, *i.e.* to only one kind of documents, and can not be generalized [3,6,13]. Some of them are based on existing libraries for pen-based applications [8], which aim at proposing generic graphical and pen-based editing functions in order to facilitate the development of such systems and to only focus on the analysis of the user strokes. Our goal is to go even further by formalizing the composition conventions of a document in order to eagerly analyze its elements. The idea is that from one pen-based system to another the needs are often the same, such as driving the recognition process and taking the relative positions of the document elements and the interaction with the user into account. Few generic systems for eager interpretation have been proposed [1,7,10]. Hammond *et al.* [7] present a hierarchical shape description language and Alvarado *et al.* [1] propose an approach based on the coupling of this formalism with dynamically constructed Bayesian networks, but, on the one hand, they only use local context of the shape to interpret it, and, on the other hand, they do not enable the representation of too complex elements such as symbols, drawings or text, which can be recognized by powerful hand drawn shape recognizers based, for instance, on neural networks [4] or hidden Markov models. We previously [9] presented an approach to design systems to eagerly interpret hand drawn structured documents using classical shape recognizers. We introduced a formalism which is an extension of the classical context-free grammars. In this paper, we present an evolution of this formalism, which makes it possible to model the interpretation of complex symbols in a more intuitive way. It is also easier to manipulate.

The main contribution of this paper is the specification of a system exploiting a generic approach to design pen-based systems with an eager interpretation process. To interpret the document elements, we use a g*lobal vision* of these elements, to take their document context into account, and a *local vision*, to recognize their shape. This knowledge is coupled with the use of hand drawn shape recognition systems. We propose a formalism, based on *interpretation rules*, that allows modeling the composition conventions of documents of various domains, and by extension how the system must interpret the document elements and interact with the user. As this method is generic, it can be applied to various natures of structured documents, such as musical scores, graphs and class diagrams in the Unified Modeling Language (UML), which are the domains we present more in detail in this paper.

The following section introduces the architecture of the system we developed. Section 3 presents one particular component, a flexible rule-based

formalism to model structured document interpretation. Section 4 highlights the consequence of the eager interpretation process on the human-computer interaction. Then, Section 5 describes three existing pen-based systems based on the approach. Finally, we summarize our future work.

2. ARCHITECTURE OF THE SYSTEM

The system is illustrated by Fig. 2. It is based on a framework constituted of three main components:

1. A set of graphical functions and pen-based editing functions (1), which are domain-independent and can be exploited by any pen-based system.
2. A formalism to model on-line structured document interpretation (2), which defines document composition conventions in a given domain.
3. A hand drawn stroke analyzer (3), which exploits the knowledge modeled by a formalism to eagerly interpret handwritten structured documents.

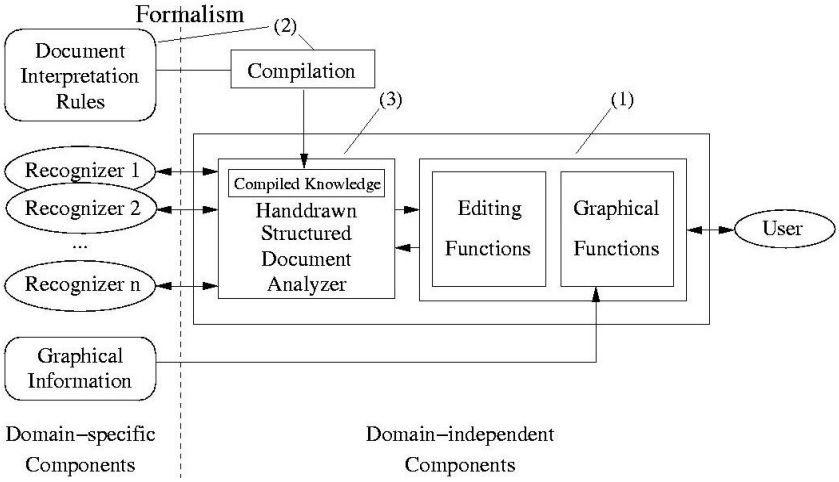

Figure 2. Architecture of the system.

The graphical and pen-based editing functions and the hand drawn stroke analyzer are *domain-independent components*; they do not need to be rewritten for each pen-based system. The development of such systems with our approach only requires realizing the *domain-specific components*: writing the interpretation rules (embedded under a compiled form), designing the necessary hand drawn shape recognizers (which can be reused from one system to another) and specifying the graphical information (for instance visual aspect of the elements). The components of our system are well separated and can be modified and adapted independently of the others. We present these components more in detail in the next subsections.

2.1 A Set of Graphical and Pen-based Editing Functions

The first component of the system deals with the user interface and the display of the document. It exploits graphical information, such as images of the symbols, which are externalized because they are domain-dependent. This component also deals with pen-based interaction and editing: the user can draw *graphical gestures*, which are interpreted by the system as editing actions. He can for instance select graphical elements, move them to another part of the document, delete them, *etc.* It is also possible to undo the last element, zoom in or out, save or load a document. This offers an alternative to classical menu and button-based interaction. Some graphical gestures are presented in Subsection 4.4. Although we have developed our own graphical and pen-based editing functions, this component is not the main contribution of this paper, because other authors have proposed similar functions [8]. The originality of our work consists in the two following components.

2.2 A Formalism to Express Interpretation Rules

The second component of the system is a rule-based formalism that describes the structured document interpretation process. It models composition conventions (for instance chronological information: which element can or must be drawn before another one). It also represents physical information, such as the spatial structure of the document. Finally, it drives the use of hand drawn shape recognizers depending on the structural context of a stroke, *i.e.* depending on its location in relation to the other elements of the document, which is its main originality. The document interpretation rules for one domain are externalized from the system: they are easily modifiable, independently of the other components. It just needs to be transformed by the generic compiler of the system. This formalism is presented more in detail in Section 3.

2.3 A Stroke Analyzer

The last component of the system is a hand drawn stroke analyzer which is able to deal with documents during their realization. Thanks to the use of the knowledge modeled by the formalism, it analyzes the strokes directly while the user is drawing. It is able to call the pertinent hand drawn shape recognizers (for example, handwritten character recognizer, geometrical shape recognizer, *etc.*) depending on the structural context of the stroke, and then update the document contexts that will help recognizing the following strokes. This parser is generic which means it does not need to be adapted to each specific domain. In the next section, we focus on the modeling of the interpretation process and the exploitation of this knowledge by the analyzer.

3. A FORMALISM FOR EAGER INTERPRETATION OF STRUCTURED DOCUMENT

We have seen that our approach is based on a formalism which aims at defining an eager interpretation of the elements of structured documents from various domains. The goal is then to propose a formalism which is as generic as possible. We previously [9] defined the four basic concepts that are, to our opinion, associated to the modeling of on-line structured document composition and eager interpretation: the *management of the chronological information*, the *representation of the document spatial structure*, the *driving of the recognition process by the document context analysis* and the *pen-based human-computer interaction*. The formalism we propose takes all these concepts into account. It is composed of interpretation rules which define the generating of the element which name they bear. Several rules can have the same name, which makes it possible to model different ways to compose the same element.

A rule takes a set of elements as parameters and returns a new one that can replace them; the parameters are the components of the new element. A parameter can be either a stroke or an already interpreted element, which allows a hierarchical shape description. The structure of a rule is composed of four blocks: a document context verification (*DCV*) block, a shape context verification (*SCV*) block, a shape recognition (*SR*) block and a document context creation (*DCC*) block, as follows:

SymbolName *(Parameter 1 , ... , Parameter n)*
Document Context Verification (DCV) block.
Shape Context Verification (SCV) block.
Shape Recognition (SR) block.
Document Context Creation (DCC) block.

DCV and *DCC* blocks enable a global vision of the document in order to define in which document contexts an element must be situated. A *DCV* block specifies the document structural contexts that have to be verified by a rule to try it, whereas a *DCC* block indicates the contexts that are created due to the recognition of an element. The *SCV* and *SR* blocks enable, given a document context, a local vision of the element to recognize; it distributes the recognition process among local constraints, formalized in the *SCV* block, and recognizers, formalized in the *SR* block.

The analysis mechanism is illustrated by Fig. 3. When a new stroke, or more generally a new element, is analyzed, the goal is to find the rule to apply. Only the rules dealing with the new element are activated. An existing document is constituted, on the one hand, of the already interpreted elements it contains and, on the other hand, of its structural contexts. This knowledge

defines a global vision of the document which must be exploited to analyze new elements. The analyzer identifies the elements of the document that can be associated to the new element to form a more complex one; this is modeled by the parameters of the rule. The analyzer also verifies the coherence of the new element with the structural context in which it is located; this is modeled by the *DCV* block. Only the rules that satisfy both of these criteria remain.

Once the analyzer has verified the coherence of the new element with the document context thanks to a global vision, it exploits a local vision to analyze the structural arrangement of the parameters of the rule; this is modeled by the *SCV* block. Then, if these constraints are satisfied, a shape recognition system can be used on the parameters of the rule in order to identify the corresponding new element. The rules that satisfy these blocks remain: they are applicable.

As more than one rule can be applicable, a *Rule Selection* component is exploited to make a decision. We give more information about the decision making process in Subsection 4.2.

Once interpreted, the new element is created; it replaces the parameters of the applied rule in the document. New structural contexts are created to help interpreting the following elements; this is modeled by the *DCC* block. Once a rule is applied, the current iteration of the analysis process is finished. Then, a new iteration begins, trying to eventually apply a rule on the new element, and so on until stability; as a consequence, a stroke can imply a sequence of transformations. If no rule can be applied on a stroke (*i.e.* if the first iteration does not succeed), it is rejected and disappears from the editing window.

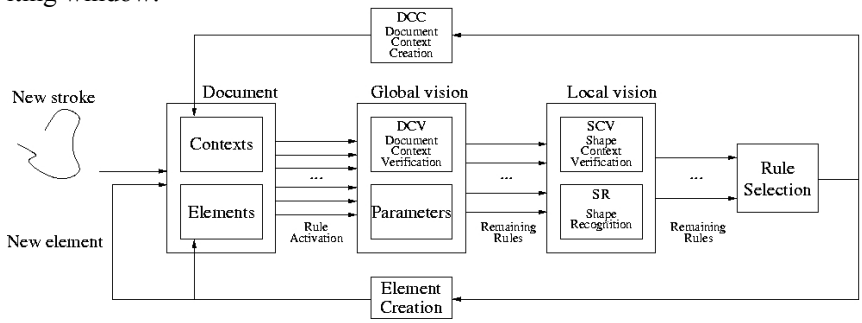

Figure 3. Illustration of the analysis mechanism.

To illustrate the definition of each block, we present hereafter two examples of interpretation rules in the context of musical score editing: interpretation of quarter-notes (***QuarterNote*** interpretation rule) and interpretation of sharps (***Sharp*** interpretation rule). We also present on Fig. 4 some illustrations of these interpretation rules.

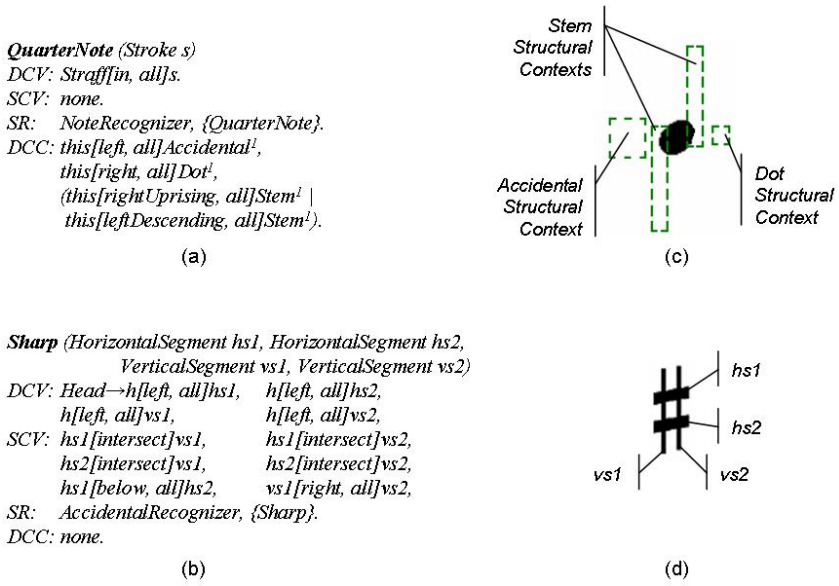

Figure 4. Two interpretation rules for quarter-note (a) and sharp (b) interpretation, the corresponding created document structural contexts (c) and sharp constituting elements (d).

A quarter-note is typically drawn with only one stroke, whereas a sharp is drawn with four of them, each one being a horizontal or a vertical segment. This explains the parameters of the corresponding interpretation rules. Moreover, as it can be seen in the rules, a block can be empty.

To understand the formalism, we present the syntax of each of the interpretation rule blocks. The *DCV block* is a list of structural contexts which have to be verified by the parameters of a rule; its syntax is as follows:

$$DCV: R_1[position_1, part_1]A_1\ , \ldots , R_n[position_n, part_n]A_n.$$

This means that at the relative position $position_i$ (*e.g.* on the left, above, *etc.*) of a reference R_i, the part $part_i$ (*e.g.* one, all, the first, the highest points, *etc.*) of an analyzed element A_i exists. R_i and A_i do not have to be part of the parameters. For example, in the ***Sharp*** interpretation rule presented in Fig. 4, the *DCV* block specifies that the four segments constituting a sharp must be on the left of a same head. A saving operator, denoted "→", is used to associate an element to an identifier and reuse it afterwards.

The SCV block models local constraints about the rule parameters to recognize an element constituted of several sub-elements: it enables to identify which part of the new element corresponds to each parameter. The syntax is the same as in the *DCV* block, but this time A and B must be parameters. By default, there is no constraint on the drawing ordering of the elements; if

necessary, a chronological operator can be used.

For example, in the ***Sharp*** interpretation rule, the *SCV* block specifies a spatial arrangement that the segments constituting a sharp must satisfy. In this rule, the structural context *[intersect]* is an alias for the context *[in,one]*. Fig. 4d illustrates how these elements must be located in relation to each other for the block to succeed. These local constraints are not necessarily enough to totally interpret the element. So we may want to exploit classical shape recognizers: this can be done with the *SR* block.

The SR block corresponds to the call to a hand drawn shape recognizer. As a consequence, only the relevant recognizers are invoked, depending on the context of an element. It is essential to increase the interpretation process robustness, since the less symbols a recognizer must interpret, the more efficient and the more reliable it is. The *SR* block syntax is:

SR: SymbolFamily, {AcceptedAnswers}.

This expression means that the recognizer of *SymbolFamily* is called, with the rule parameters as input; if its answer is included in *AcceptedAnswers* the recognition process is a success. Actually, the order of the elements presented to the recognizer is the order of the parameters in the declaration of the rule; so it is always the same. As a result, its work is relieved, because it has to interpret the elements always in the same order.

For example, in the ***Sharp*** interpretation rule, we exploit an accidental recognizer with as input the four segments the rule takes as parameters, always in this sequence; the expected recognizer answer is a sharp.

The DCC block is a list of document structural contexts that are created due to the recognition of an element. Its syntax is:

$$DCC: R_1[position_1, part_1]A_1^{m\_1}, \ldots, R_n[position_n, part_n]A_n^{m\_n}$$

This means that at the relative position $position_i$ of an element R_i, the part $part_i$ of an element A_i can exist. The current element is referenced as "*this*". The number m_i indicates how many A_i can exist in this context and is * if there is no limit. For example, in the ***QuarterNote*** interpretation rule, the *DCC* block specifies that once a quarter-note exists, it is possible to draw an accidental on its left, a durational dot on its right, and a stem, either uprising or descending. A disjunction operator, denoted "|", is used to define an alternative between structural contexts.

Fig. 4c illustrates the position of the corresponding structural contexts in relation to the quarter-note. More information about the visualization of these structural contexts is given in Section 4.1. Aliases are defined in the ***QuarterNote*** and ***Sharp*** interpretation rules to specify that a head can be a whole-note, a half-note or a quarter-note, and that an accidental can be a flat, a sharp or a natural.

4. EAGER INTERPRETATION AND USER INTERACTION

In this section, we present the human-computer interaction in the context of eager interpretation. The concepts are independent of the domain of the documents that are being drawn. We first explicit how the system can guide the user in the drawing process. Then, we highlight the importance of the interaction with the user within the interpretation process.

4.1 Drawing the Document Elements

To help the user to have reference marks, rectangles giving an indication of the document structural contexts that are generated in *DCC* blocks can be displayed. Whereas these rectangles are strict, structural contexts are not: the user does not have to draw the elements exactly in the rectangles. To lighten the editing area, a context is visible provided that it is not already filled with an element. Thus, as presented in Fig. 5, it is possible to switch between a *novice mode*, in which empty contexts are visible, and an *expert mode*, in which they are not. It is also possible to show only the boxes that are in the context of the pen, once again to lighten the editing area: it corresponds to a *contextual mode*. The experience shows that this last mode seems to be the most user-friendly, since it permits to only focus on potentially interesting structural contexts according to the pen position. We would like to note that the quarter-note structural contexts correspond to those declared in the ***QuarterNote*** interpretation rule defined in Fig. 4.

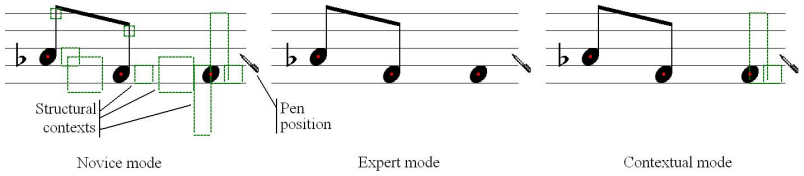

Figure 5. Three different visualization modes.

4.2 Dealing with Ambiguities

Naturally, more than one rule can sometimes be applied at the same time to an element. In order to limit this, we firstly determine to which degree an element satisfies the contexts that are defined in an interpretation rule; it is then possible to make a decision between rules if the difference between their degrees is significant. We secondly use hand drawn shape recognizers with reject options, *i.e.* which do not give an answer unless their confidence is high enough [11]. The advantage is to filter possible interpretations, and to prevent from displaying an answer with a high probability of being wrong.

4.3 User Validation

As the recognition process is eager, the result of the analysis is displayed directly as the user is drawing. We can then exploit the human-computer interaction and integrate the user in the interpretation process, because he can validate or reject its results. Thus, if after the display of the answer, the user goes on with the drawing of the document, he implicitly accepts and validates it; on the contrary, if he does not agree, he can delete the new element with a deletion gesture and so explicitly reject it. The main consequence of this process is that it is not necessary for the analyzer to question a decision made beforehand, since it has been validated by the user. We believe that it is pertinent because it could be perturbing for the user to see the interpretation of an element changing after drawing another one. It is a major advantage of the eager interpretation process on the lazy one: indeed, the user limits the ambiguities, which makes the system more robust and more efficient, increasing its usability. The system is also faster, since it just has to interpret the last stroke.

4.4 Editing the Document

Every element of the document has a selection dot, which is a small red anchor point. In order to select an element, the user has to surround its selection dot. This way, it is not necessary to draw a stroke as big as the element. Several elements can be selected or other elements can be added to the selection in the same manner. When an element is selected, the document elements associated to it are also selected; they actually correspond to the elements which have been drawn in the different contexts it has created. Once selected, elements can be moved to another part of the document by pointing to one of them and moving the pen to the appropriate place. It is also possible to move an element directly by pointing at its selection dot and moving the pen (by drag and drop). To delete an element, the user can, for instance, move it outside the editing window. These mechanisms are illustrated by Fig. 6: on the left, the user draws a stroke around the selection dot of a half-note; in the middle, the note and its associated elements (*i.e.* its natural and its stem) are selected, and the user moves them with the pen; on the right, he raises the pen to drop the elements.

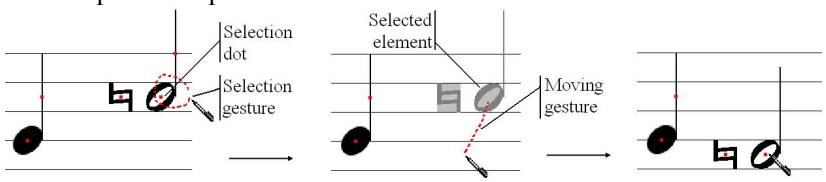

Figure 6. Selection and moving mechanisms: example on a half-note.

5. SYSTEMS DEVELOPED THANKS TO THE METHODOGY

Thanks to the generic method we present in this paper, we have already been able to develop three different prototypes, one for musical score editing, one for graph editing and one for UML class diagram editing. As mentioned in Section 2, we just needed to:

- Write the interpretation rules.
- Use existing recognition systems [2] or design new ones, here based on Radial Basis Function Networks (RBFN) classifiers [4,11].
- Specify the graphical aspects of the document elements.

We first developed a pen-based musical score editor, which is an evolution of the prototype introduced previously [9], with more available symbols. Fig. 1 presents this system, whereas Fig. 5 and Fig. 6 explicit some of the available symbols. Fig. 7 presents two screenshots of the system, on the left the interpreted symbols, and on the right the corresponding hand drawn strokes. In order to design a user-friendly system adapted to musician needs, it is developed in collaboration with professional musicians from the MIAC (Music and Image: Analysis and Creation) of the Rennes 2 University.

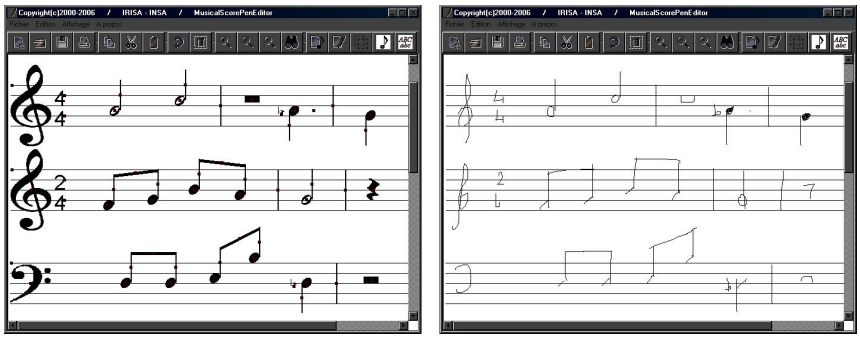

Figure 7. Screenshots of the pen-based musical score editor prototype.

The second domain we developed is graph editing: it allows the drawing of nodes, represented by geometrical shapes (*e.g.* rectangles, triangles or circles), and of arcs. Each of these components can be drawn with one or several strokes. We then focused on one particular graph-based domain, which is UML class diagram editing. The corresponding system makes it possible to draw classes, represented by rectangles, and to write their name, attributes and functions. Aggregation and inheritance are one of the already available symbols. Fig. 8 shows screenshots of the graph editor (above) and of the UML class diagram editor (below). As in Fig. 7, interpreted symbols are on the left and the corresponding handwritten strokes are on the right. We have not yet associated our UML class diagram editor with a handwritten character recognizer: strokes corresponding to handwritten text are detected but not

yet interpreted and so not changed. We would like to emphasize that although it is not visible on Fig. 7 and Fig. 8, the interpretation is eager: the user strokes are replaced with their corresponding symbols progressively.

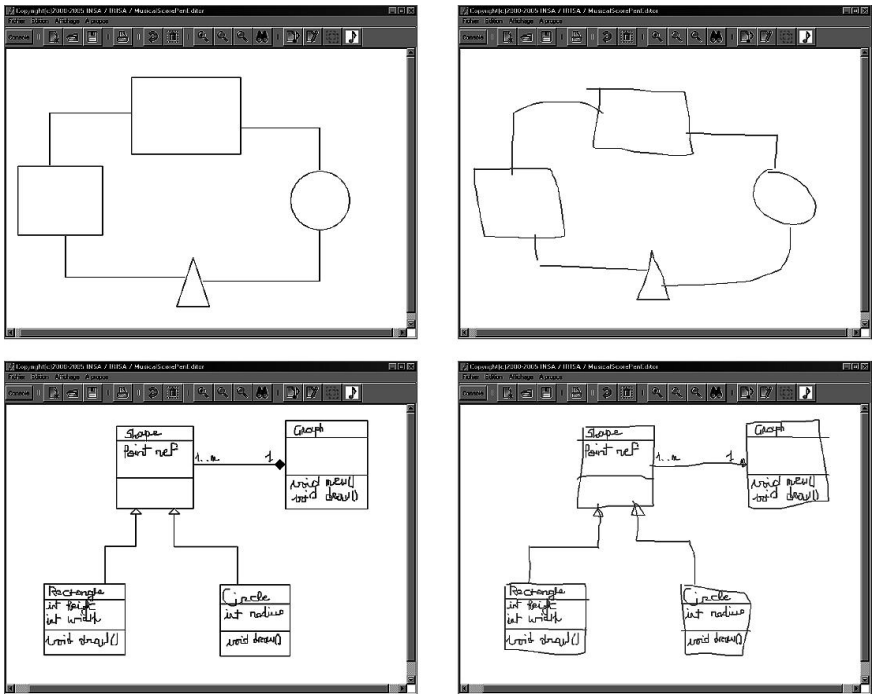

Figure 8. Screenshots of the graph and UML class diagram editor prototypes.

6. CONCLUSION

In this paper, we presented a new approach for pen-based system development. The aim is to interpret hand drawn structured documents eagerly, *i.e.* directly while the user is drawing. We emphasized the separation of the various components of our system, which can therefore be modified independently from the others. In particular, a major component is a rule-based formalism that models structured document interpretation; it can be adapted to various domains, such as musical score editing, graph editing and UML class diagram editing. We exploit human-computer interaction to integrate the user in the recognition process: he can validate or reject an answer given by the system. Future work will aim at defining test protocols at a larger scale in order to go on with improving the interaction with the user. The aim is to design pen-based systems that are as usable and intuitive as possible. In order to validate the genericity, we believe that the next challenge will be to apply this methodology to a new complex domain: electric diagram editing.

ACKNOWLEDGEMENT

Authors would like to thank Guy Lorette from the IMADOC team and Bruno Bossis from the MIAC team for their contribution to this work. This project benefits from the financial support of the Brittany Region.

REFERENCES

[1] Alvarado, C. and Davis, R., *SketchREAD: a Multi-domain Sketch Recognition Engine*, in Proc. of the 17th Annual ACM Symposium on User Interface Software and Technology UIST'2004 (Santa Fe, 24-27 October 2004), ACM Press, New York, 2004, pp. 23-32.

[2] Anquetil, É. and Bouchereau, H., *Integration of an On-line Handwriting Recognition System in a Smart Phone Device*, in Proc. of the 16th IAPR Int. Conf. on Pattern Recognition ICPR'2002 (Quebec, 11-15 August 2002), 2002, pp. 192-195.

[3] Anstice, J., Bell, T., Cockburn, A. and Setchell, M., *The Design of a Pen-Based Musical Input System*, in Proc. of the 6th Australian Conf. on Computer-Human Interaction OZ-CHI'96 (Hamilton, 24-27 November 1996), IEEE Comp. Soc. Press, 1996, pp. 260-267.

[4] Bishop, C.M., *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.

[5] Blostein, D., Lank, A., Rose, A., and Zanibbi, R., *User Interfaces for On-Line Diagram Recognition*, in Proc. of the 4th IAPR International Workshop on Graphics Recognition Algorithms and Applications GREC'2001 (7-8 September 2001), Lecture Notes in Computer Science, Vol. 2390, Springer-Verlag, Berlin, 2001, pp. 92-103.

[6] Donaldson, A.F. and Williamson, A., *Pen-based Input of UML Activity Diagrams for Business Process Modelling*, in Proc. of the 1st Workshop on Improving and Assessing Pen-Based Input Techniques (Edinburgh, 5 September 2005), 2005, pp. 31-38.

[7] Hammond, T. and Davis, R., *LADDER: A Language to Describe Drawing, Display, and Editing in Sketch Recognition*, in Proc. of the 18th Int. J. Conf. on Artificial Intelligence IJCAI'03 (Acapulco, 9-15 Aug. 2003), M. Kaufmann, San Francisco, 2003, pp. 461-467.

[8] Lank, E.H., *A Retargetable Framework for Interactive Diagram Recognition*, in Proc. of the 7th Int. Conf. on Document Analysis and Recognition ICDAR'2003 (Edinburgh, August 2003), IEEE Computer Society Press, Los Alamitos, 2003, pp. 185-189.

[9] Macé, S., Anquetil, É., and Coüasnon, B., *A Generic Method to Design Pen-Based Systems for Structured Document Composition: Development of a Musical Score Editor*, in Proc. of the 1st Workshop on Improving and Assessing Pen-Based Input Techniques (Edinburgh, 5 September 2005), 2005, pp. 15-22.

[10] Mas, J., Sanchez, G., and Llados, J., *An Incremental Parser to Recognize Diagram Symbols and Gestures represented by Adjacency Grammars*, in Proc. of the 6th IAPR Int. Workshop on Graphics Recognition GREC'2005, 2005, pp. 229-237.

[11] Mouchère, H., Anquetil, É., and Ragot, N., *Étude et gestion des types de rejet pour l'optimisation de classifieurs*, in Actes du 15ème Congrès francophone Reconnaissance des Formes et Intelligence Artificielle RFIA'2006 (Tours, January 2006), 2006.

[12] Nakagawa, M., Machii, K., Kato, N., and Souya, T., *Lazy Recognition as a Principle of Pen Interfaces*, in Proc. of the ACM Conf. Companion on Human Factors in Computing Systems INTERCHI'93 (Amsterdam, 24-28 April 1993), ACM Press, 1993, pp. 89-90.

[13] Toyozumi, K., Mori, K., Suenaga, Y. and Suzuki, T., *A System for Real-time Recognition of Handwritten Mathematical Formulas*, in Proc. of the 6th Int. Conf. on Document Analysis and Recognition ICDAR'2001 (Seattle, 10-13 Sept. 2001), pp. 1059-1064.