Infrastructure of an Adaptive Multi-agent System for Presentation of Mathematical Expression to Visually-Impaired Users

Ali Awde, Chakib Tadj, Yacine Bellik

Abstract - **The infrastructure of our adaptive system is aimed at correctly presenting a mathematical expression to visually impaired users. Its design is based on a multi-agent system that determines the appropriate presentation format based on the given interaction context (i.e. combined user, environment and system contexts) and the expression complexity as well as the user preferences. The architecture of our infrastructure is layered, thus encapsulating the components of the various layers. The system design is intended to be adaptive, fault tolerant and is capable of self-adaptation under varying conditions (e.g. missing or defective components). In case of failure of media device, our system is capable of replacing the faulty component with a new one (if a replacement is available). If replacement is not possible, the system re-determines the new modality and presentation format apt for the new configuration. The human intervention is greatly reduced in our system as it is capable of self-configuration, and learning. In our work, agent communication simulation has been carried out on the Java Agent Development Framework (JADE[*]) platform. This work is an original contribution to the ongoing research in helping the visually-impaired users to become autonomous in using the computing system. Our aim is to improve the computing productivity of a visually-impaired user.**

Index Terms— **Adaptive system, fault-tolerant system, mathematics for visually-impaired users, multi-agent system.**

I. Introduction

MATHEMATICS is a fundamental foundation of science. Understanding science is impossible without knowing mathematics. Mathematics for visually-impaired users, however, is a challenging task due to the following reasons: First, the visual mathematical representation is bi-dimensional and the interpretation of a mathematical expression is related to one's knowledge of the expression's individual spatial components. Second, the conversion of a multi-dimensional structure to a non-visual representation is a difficult problem. For example, the representation of a mathematical expression in Braille requires supplementary information to denote some components in order for the blind users to read the expressions easily. Also, the conversion of mathematics into an audio format is often ambiguous. Third, the vocabulary terms used by sighted people in a mathematical document are quite large compared with the amount of data accessible by a visually-impaired user. For example, in a standard 6-dot Braille[†], we can encode 64 characters. This number of symbols is, however, insufficient to represent all frequently used mathematical symbols. Also, large number of symbols is a big challenge to blind users. For instance, Braille characters are often embossed into a paper which is a static media; hence, user will not be able to manipulate the data easily. Indeed, some flexible methodologies are needed to allow blind users to access and navigate terms in a mathematical expression easily.

This paper presents the challenges that underlie in designing such infrastructure that provides presentation of mathematical expressions to visually impaired users, and how we address the problem cited above by proposing an intelligent multimodal computing system that interacts with the user, capable of choosing modalities and media devices based on a given interaction context. The format for the presentation of a mathematical expression is selected based on available media devices, user's preferences and context of the mathematical expression. The proposed infrastructure is a multi-agent system. The design of this multi-layered infrastructure enables every layer's calculation and decision making be hidden from other layers. In this way, the possible propagation of ripple effect during any stage of software life cycle becomes limited and restricted only within the boundaries of the concerned layer. Various layers communicate among themselves via parameters passing. In this work, we discuss the design of each agent and its responsibilities. Also, we

Manuscript received February 15, 2009. This work has been made possible through the funding provided by LIMSI laboratory, LATIS laboratory and the Décanat aux études of École de technologie supérieure (ÉTS).

A. Awde is with École de technologie supérieure, Montréal, Canada. Phone: 514-396-8800; fax: 514-396-8684; Email: ali.awde.1@ens.etsmtl.ca.

C. Tadj is with École de technologie supérieure, Canada. Email: ctadj@ele.etsmtl.ca.

Y. Bellik is with Université Paris Sud XI, France. Email: Yacine.bellik@limsi.fr.

[*] JADE: http://jade.tilab.com/

[†] http://6dotbraille.com

present our fault tolerant system design and a JADE simulation of agent communication.

This work is our contribution to make mathematics more accessible to visually-impaired users. It is aimed at providing some autonomy to blind users when dealing with mathematical expressions. The rest of this paper is structured as follows. Section 2 provides a brief review of the state of the art related to our work; Section 3 lists down the technical challenges in this work and essays our approach to address each one of them. Section 4 presents the individual components of our adaptive multimodal computing system. In Section 5, the principles used for the knowledge acquisition of our learning agent are discussed. Examples are provided in Section 6 while Section 7 presents some formal specifications related to the functionalities of the system. The future works and conclusion are presented in Section 8.

II. Review of the State of the Art

To a visually-impaired user, understanding mathematical expression requires repeated passage over the expression, sometimes skipping some secondary information, only to revert back to it again and again until the user fully grasps the expression. A complicated task like this is detailed in [1]. Some tools, however, have been developed to lessen the complexity of performing similar task, among them being the Mathtalk [2], Maths [3], DotsPlus [4], EasyMath [5], and AudioMath [6, 7]. MathTalk and Maths convert a standard expression into audio information. In Maths, the user can read, write and manipulate mathematics using a multimedia interface containing speech, Braille and audio. Raman developed Aster [8], a program that takes a Latex document and reads it loud using several audio dimensions that make up the different components of the expression. VICKIE [9] and BraMaNet [10] are transcription tools that convert mathematical document (written in Latex[‡], MathML[§], HTML, etc.) to Braille representation. DotsPlus is a tactile method of printing documents that incorporates both Braille and graphic symbols (e.g. $\prod$, $\sum$, etc.) In EasyMath, regardless of using Braille or overlay keyboard, its main focus is to keep the general structure of mathematical expressions intact.

None of these tools, however, is complete. Studies have been made for evaluating these tools based on users' needs [5, 11]. Results indicate that users are neither independent nor able to do their homework (in case of students) without the help of sighted people. Indeed, each tool has its own set of usage limitations. For example, Aster transforms only a LaTex document into an output suitable for speech while AudioMath transforms only a MathML input document into a speech output. Our approach, therefore, is to get the strength of each tool, integrate each one of them into our work in order to build a system that (1) broadens the limits of utilization, (2) provides the user with opportunities to access as many document types as possible, and (3) presents data output in as many suitable formats as possible after considering user situation and the special symbols within the expression. This work is an essential contribution because we offer all types of data presentation formats yet requires minimum intervention from the user.

HOMERE [12] is a multimodal system that allows visually-impaired users to use haptic/touch and audio modalities to explore and navigate virtual environments. In comparison, our approach is better because there are no pre-defined input-output modalities; the selected modalities are chosen according to their suitability to user's context. To visually-impaired users, multi-modality is even more important as it provides them equal opportunities to use informatics like everybody else. In determining the appropriate modality, the user situation plays an important role. In our work, the notion of user context includes additional handicaps and user preferences on the priority rankings and parameter settings of media devices and presentation formats.

An agent is some software that senses its environment and is capable of reaction, proactivity, and social interaction. A group of agents in a system forms a Multi-Agent System (MAS) [13]. Agents and MAS [14, 15] have been widely used in many applications, from relatively small systems such as email filters up to large, open, complex, mission-critical systems such as air traffic control [16]. Generally, it is preferred over traditional techniques (i.e. functional or object-oriented programming) because the latter is inadequate in developing tools that react on environment events. Significant works on MAS for visually impaired include [17, 18]. For example, Tyflos [17] could help a visually impaired user to be partially independent and able to walk and work in a 3-D dynamic environment. Our work, in contrast, uses agents to detect user context, and other data in order to assist the system to determine the media and modalities that are appropriate for the user.

III. Technical Challenges

In this section, we resume the problems in designing a system that will present a mathematical expression to visually impaired users, pose specific technical challenges that need to be solved and describe our

[‡] L. Lamport, LaTeX: The Macro Package, http://web.mit.edu/texsrc/source/info/latex2e.pdf, 1994
[§] MathML, http://www.w3.org/Math

approach to address those challenges.

In our proposed infrastructure, we envision a system that is rich in media devices and modality selection, data formats, techniques in converting one data format to another, and an adaptive interface that allows user to manipulate mathematical data. To design such a system, a solution must address key requirements, cited below:

Requirement 1: *Provide a multiagent system that coordinates all its components in an orderly manner, providing autonomy to the user and is able to adapt automatically to a given interaction context.* How do we design a multi-layered, multiagent system that satisfies the system requirement? What mechanisms must be adopted to make the system tolerant from faults?

Requirement 2: *Provide a mechanism that allows selecting the appropriate media device supporting the selected modality.* How do we design a system that satisfies the user preferences? How the media device will be detected? Which media device will be activated to support a specific modality?

Requirement 3: *Provide an infrastructure for analysis and conversion of a mathematical expression, embedded within a document (in MathML format) into its corresponding encoded format and then into its presentation format.* How to convert an expression written in MathML format into an expression in encoded format and into an expression using a presentation format such as the Braille, DotsPlus, EasyMath and audio?

Requirement 4: *Provide a mechanism that allows visually impaired users to manipulate terms in a mathematical expression.* How do we design a system component that allows the user to add, modify and delete mathematical terms, and to search the expression for a term in random manner?

The rest of this paper addresses the technical challenges by providing specific solutions to the system requirements cited above.

IV. The Components of an Adaptive Multimodal Computing System

A. Our Adaptive Multimodal Computing System for Mathematical Expression Presentation to visually impaired users

Fig. 1 shows the layered view of an adaptive multimodal computing system that presents mathematical expressions to visually-impaired users. The layers and their roles are as follows: (1) *Physical Layer* – contains all the physical entities of the system, including devices and sensors. The raw data from this layer are sampled and interpreted and forms the current instance of interaction context. (2) The *Context Gathering Layer* – here, the interaction context (of tuple <user, system, environment>) is detected; (3) The *Control and Monitoring Layer* – it controls the system, coordinating the detection of user's interaction context, the mathematical expression, its presentation and/or manipulation; (4) The *Data Analysis Layer* – here, the presentation format of the mathematical expression is selected based on available resources and user's situation; (5) The *Data Access Layer* – in this layer, mathematical expression may be searched or edited by the user; (6) The *Presentation Layer* – here, the mathematical expression is presented through an optimal presentation format.

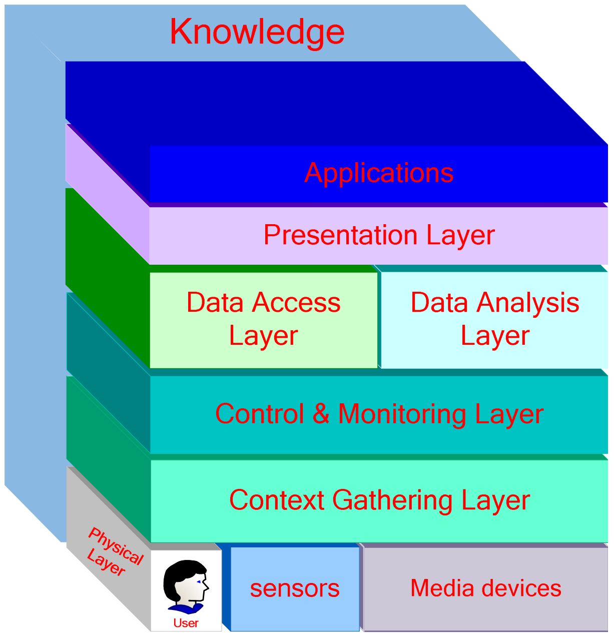


Fig. 1. The architectural abstraction of a generic MM computing system for visually-impaired users.

Fig. 2 shows our model of a multi-agent adaptive multimodal system for visually-impaired users. The agents' functionalities in the system's layers are detailed in the next sections.

B. Modality and Media

In our work, we adopt the concepts of media and modality that are defined by Bellik in [19].

1. *Modality* is defined by the information structure as it is perceived by the user (e.g. text, speech, Braille, etc.).

2. *Media* is defined as a device used to acquire or deliver information or data (e.g. screen, terminal Braille, mouse, keyboard, etc.)

Here, *Vocal* and *Tactile* modalities are possible since we address visually impaired users. Also, in general, interaction is possible if there exists at least one modality for data input and at least one modality for data output. Given a modality set M = {V_{in}, T_{in}, V_{out}, T_{out}} wherein V_{in} = *vocal input,* V_{out} = *vocal output,* T_{in} = *tactile input and* T_{out} = *tactile output* then interaction is possible under the following condition:

$$Interaction\ Possible = (V_{in} \vee T_{in}) \wedge (V_{out} \vee T_{out}) \qquad (1)$$

where the symbols $\wedge$ and $\vee$ denote logical AND and OR, respectively. There are usually more than one media that support a specific modality (see section C-3).

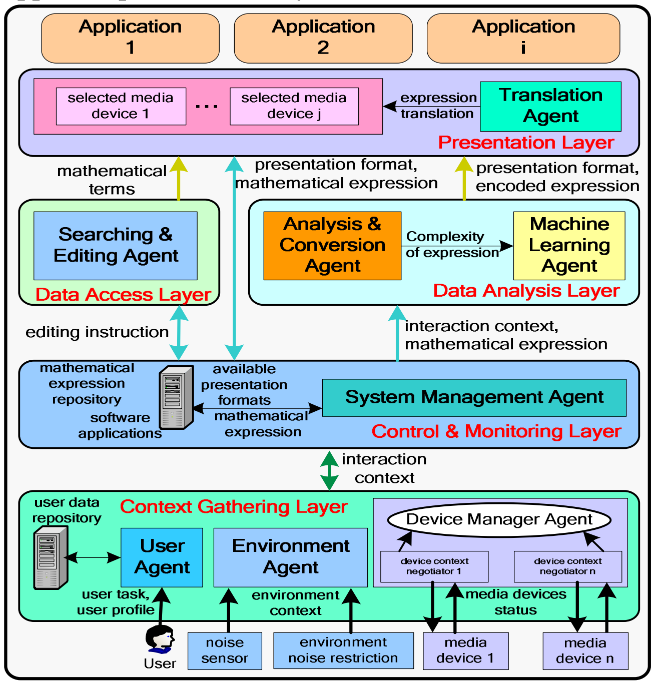


Fig. 2. Architectural layer view of the multimodal computing system

C. *The Context Gathering Layer*

In this layer, the user's interaction context is identified. There are three agents that obtain the context of the *user*, the *environment*, and the *system* which collectively form the interaction context. Here, we present briefly the interaction context, for more details see [20].

1. The User Context

In this work, the user context (UC) is a function of user profile (including any handicap) and preferences. The *user agent* (UA) detects the user's profile and preferences. We have a *user data repository* where user's task (i.e. the mathematical expression) is stored. In such repository, there is a *user profile* (UP) for the user, which contains, among others, the user's username, password, his computing devices and identifications (i.e. IP addresses) and his special needs (i.e. handicap). This information is useful for determining the suitable modalities. For example, being mute prevents the user from using vocal input modality.

2. The Environment Context

The *environment's context* (EC) detected by Environment Agent (*EnvA*), is the assessment of a user's workplace condition. In this work, EC is based on the following parameters: (1) the workplace's *noise level* – identifies if it is quiet/acceptable or noisy, and (2) the *noise level restriction* – identifies whether a workplace imposes mandatory silence or not. For example: in a library where silence is required, sound producing media (e.g. speaker) needs to be muted or deactivated.

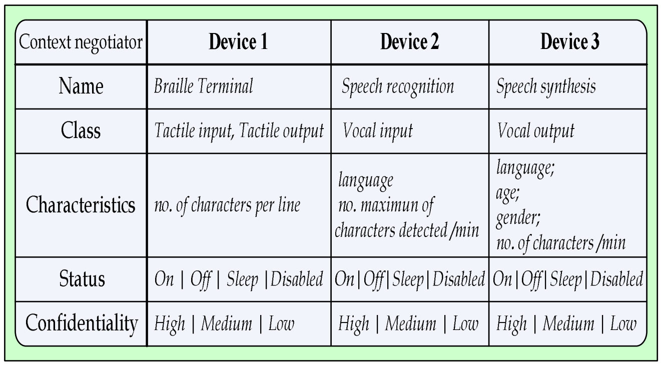
The noise level is interpreted by EnvA from the sampled raw data of a sensor. In our work, *50 dB or less* is considered "*acceptable*" while *51 dB or more* is considered "*noisy*".

For environment noise restriction, we have a database of pre-defined places (e.g. library, park) and their associated noise restrictions (e.g. library: silence required, park: silence optional). User can update and modify some database records.

3. The System Context

In this work, the *system context* (SC) implies the user's computing device and the available media devices. SC is managed by the *Device Manager Agent* (DMA). The computing device (e.g. PC, laptop, PDA, cellular phone) affects the modality selection. For example, using a PDA or cell phone prevents user from using tactile input or output modality. On the other hand, some of the most commonly-used media devices suitable for blind users are: (i) *Keyboard*; (ii) *Microphone*; (iii) *Speech Recognition* (iv) *Speech Synthesis* (v) *Speaker*; (vi) *Headset*; (vii) *Braille Terminal* (viii) *Overlay or Concept Keyboard* (ix) *Tactile Printer or Embosser.*

Every media device has its own *device context negotiator* (DCN). A DCN is an agent that detects the media device's context; it is a link between the actual media device and the DMA. Every DCN has the following attributes (see Fig. 3): its *name, class, characteristics, status*, and *confidentiality*. *"Name"* identifies the media device it manages (e.g. "Braille context negotiator" detects the context of a Braille terminal). *"Class"* identifies its form of modality. *"Characteristics"* identify the unique features of the device. *"Status"* identifies if the device is on, off, sleep or disabled. *"Confidentiality"* is the perceived reliability of the device and is denoted as high, medium or low.



Context negotiator	**Device 1**	**Device 2**	**Device 3**
Name	*Braille Terminal*	*Speech recognition*	*Speech synthesis*
Class	*Tactile input, Tactile output*	*Vocal input*	*Vocal output*

Characteristics	*no. of characters per line*	*language*
		no. maximun of characters detected /min
			language;
			age;
			gender;
			no. of characters /min

Status	*On | Off | Sleep |Disabled*	*On | Off | Sleep | Disabled*	*On | Off | Sleep | Disabled*
Confidentiality	*High | Medium | Low*	*High | Medium | Low*	*High | Medium | Low*

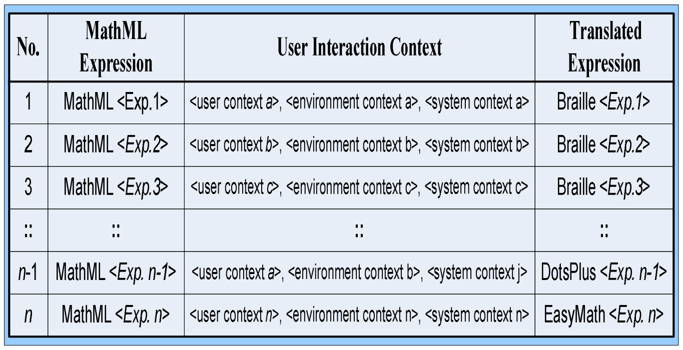
Fig. 3: Attributes of device context negotiator for Braille terminal, speech recognition and synthesis

D. The Control and Monitoring Layer

System Management Agent (SMA) monitors and reports on the system configuration and application activity. When a fault occurs, SMA should deal with errors and find solution to keep system working (see section V).

Upon detection of interaction context, the control and monitoring layer via its SMA examines the mathematical expression in the user's task. It determines the available presentation formats by sensing the presence of conversion software in the computer. It also determines if the current mathematical expression and interaction context are already listed in the system's *mathematical expression repository* (MER). If so, then the expression is sent to Presentation Layer for its presentation. Otherwise, the system must learn how to present this expression by sending all the available information to the Data Analysis Layer.

The MER is a *private* database that keeps all mathematical expressions that have already been encountered. All expressions are stored in a tabular form; each table entry contains (1) the original expression in MathML format, (2) the user interaction context, and (3) the translated expression. Fig. 4 shows the MER contents in generic format. Hence, an entry in MER prevents the unnecessary repetition of calculations and analysis that were done when the condition was first encountered. The MER is scalable. Its contents are time-bounded (i.e. similar to electronic mail); its information is periodically updated, and very old records are deleted.



No.	MathML Expression	User Interaction Context	Translated Expression
1	MathML <Exp.1>	<user context *a*>, <environment context a>, <system context a>	Braille <*Exp.1*>
2	MathML <*Exp.2*>	<user context *b*>, <environment context b>, <system context b>	Braille <*Exp.2*>
3	MathML <*Exp.3*>	<user context *c*>, <environment context c>, <system context c>	Braille <*Exp.3*>
::	::	::	::
n-1	MathML <*Exp. n-1*>	<user context *a*>, <environment context b>, <system context j>	DotsPlus <*Exp. n-1*>
n	MathML <*Exp. n*>	<user context *n*>, <environment context n>, <system context n>	EasyMath <*Exp. n*>

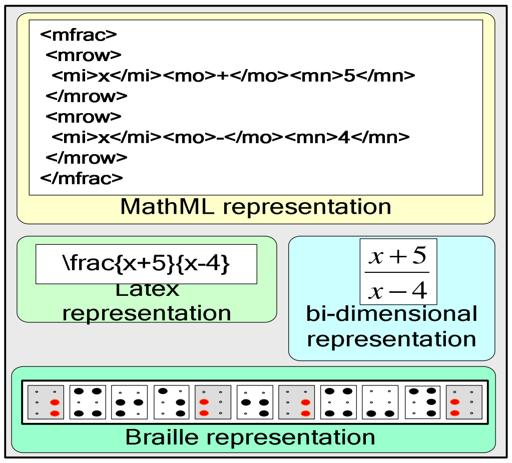
Fig. 4: The mathematical expression repository, in generic format.

E. The Data Analysis Layer

Here, we present the analysis and learning methods that are invoked in determining the optimal presentation of a mathematical expression based on a given interaction context.

1. The Visually-Impaired User's View of a Mathematical Expression

To a visually-impaired user, a *simple* mathematical expression (e.g. quadratic equation) becomes complex due to the presence of elements such as the *subscript*, *exponent*, *mathematical symbols* (e.g. Π, ⤖, etc.), and the expression's *dimension* (i.e. complex numerator, denominator). In informatics, a mathematical expression is generally written using the syntax of *MathML* or LaTex which are inappropriate format for visually impaired users. For example, a simple fraction in Fig. 5 is shown with its equivalence in MathML, LaTex, Braille and its linear representations.



```
<mfrac>
 <mrow>
  <mi>x</mi><mo>+</mo><mn>5</mn>
 </mrow>
 <mrow>
  <mi>x</mi><mo>-</mo><mn>4</mn>
 </mrow>
</mfrac>
```

```
\frac{x+5}{x-4}
```

$$\frac{x+5}{x-4}$$

Fig. 5: A fraction in bi-dimensional form and its corresponding equivalent in LaTex, MathML and Braille.

2. Representation of Mathematical Operation in Different Formats

Presentation formats use different methods to represent a mathematical operation. Using *Braille*, there is a unique symbol for every operation. Using *speech*, an operation is uttered using a specific word (e.g. "+" is "add", "-" is "minus", etc). Using *DotsPlus*, an operation is represented by a unique symbol in a tactile form. Using *EasyMath*, every basic operation (e.g. +, -, ×, ÷, etc.) is represented by a unique symbol similar to its Braille representation. For special operation (e.g. ⤖, Π, log, ∫, ∬, ∭, etc), however, the representation is in tactile form. Note that the representation of special operations in DotsPlus and EasyMath are not the same. For example, the + operation symbol is represented by a Braille symbol whereas in DotsPlus it is represented as "+" in embossed tactile form.

3. The expression complexity

The complexity of the expression affects the choice of the format of presentation. In case of simple expressions (see Fig. 6, expression (b)), the user will choose simple presentation format such as Braille or audio. Note that when the expression is complex, user has to choose more complex presentation format such as DotsPlus's presentation (e.g. expression (a) in Fig. 6). Hence, the complexity of the expression is important for determining the suitable presentation format. In [21], authors proposed a method to determine the complexity of the expression based on: the depth of syntax tree, number of operands and operators.

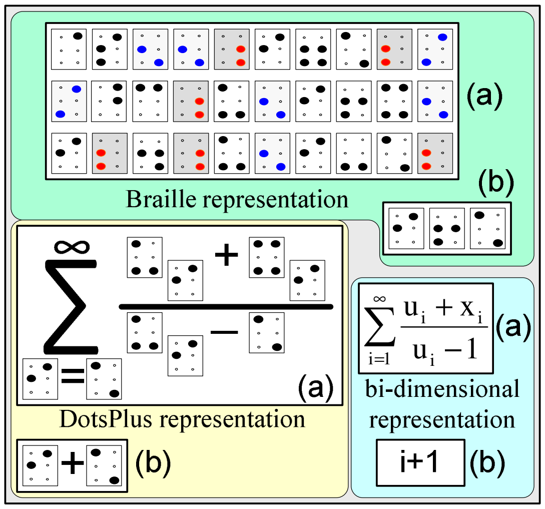

(a)
(b)
Braille representation
(a)
DotsPlus representation
(b)
$\sum_{i=1}^{\infty} \frac{u_i + x_i}{u_i - 1}$ (a)
bi-dimensional representation
i+1 (b)

Fig. 6: Two sample expressions in bi-dimensional form and its corresponding equivalent in Braille and DotsPlus format.

4. The Analysis and Conversion Agent

Fig. 7 shows the functionalities of Analysis and Conversion Agent (ACA). It receives a mathematical expression (in MathML format) from SMA. Using *grammar rules and dictionary*, the expression is analyzed lexically. The result yields a list of lexemes. A lexeme is a parameter within an expression which may be an operand or an operator. Given the lexemes, the *parser* analyzes the expression parameters (i.e. operands, operators and syntax tree) then sends parameters to the *Expression Evaluator* to determine its complexity. The *parser* sends then the expression to the *Expression Encoder* to be translated into its encoded format.

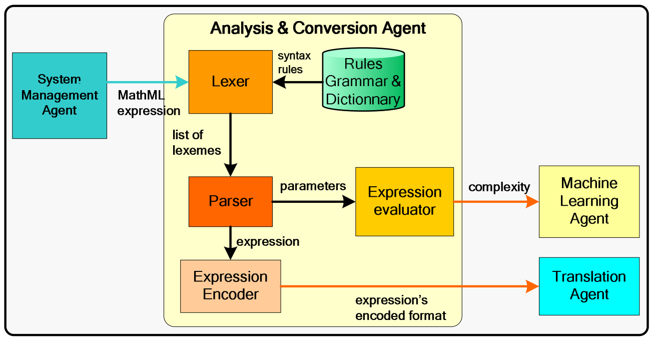


Fig. 7: The Analysis and Conversion Agent.

As an example, Fig. 8 shows the fraction defined in Fig. 5, as a specimen expression. In (step 1), the expression is sent to the Lexer. In (step 2), using the XML grammar, the expression is decomposed into a list of lexemes; the list is then sent to the parser. In (step 3), the operations and operands in the expression are sent to expression evaluator and encoder. Together, in (step 4), the evaluator deduces the complexity of the expression (e.g. simple) while the encoder produces the encoded expression. Finally, in (step 5), the MLA is informed about the complexity of the expression, while the encoded expression is forwarded to the Translation Agent (TA).

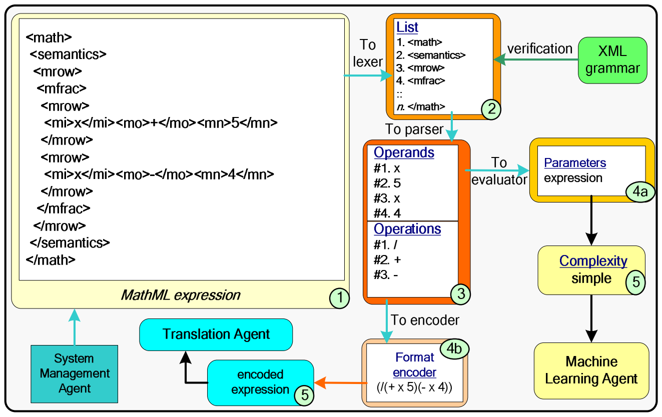
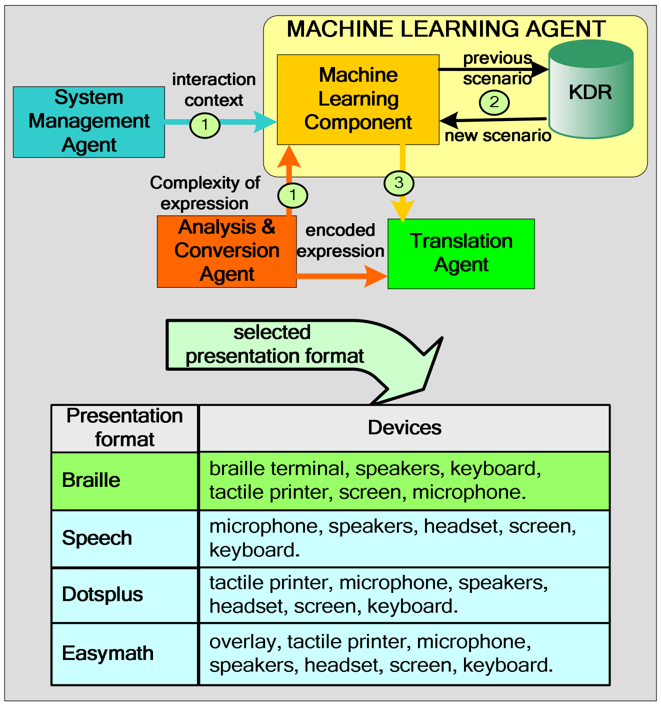


Fig. 8: A sample analysis of a specimen fraction.

5. Determining a Mathematical Expression's Presentation Format

The MLA selects the presentation format based on interaction context and the complexity of the expression. The selection of the appropriate presentation format and the learning process are detailed in [20]. Here, we present briefly the functionalities of MLA that are depicted in Fig. 9.



Presentation format	Devices
Braille	braille terminal, speakers, keyboard, tactile printer, screen, microphone.
Speech	microphone, speakers, headset, screen, keyboard.
Dotsplus	tactile printer, microphone, speakers, headset, screen, keyboard.
Easymath	overlay, tactile printer, microphone, speakers, headset, screen, keyboard.

Fig. 9: The Machine Learning Agent and its interaction with other system components.

In (step 1), MLA receives interaction context from SMA. The ACA informs the MLA of the complexity of the expression as discussed in previous section. The interaction context and the complexity of the expression input to the *machine learning component* (MLC) forms the pre-condition scenario. In (step 2), information about the corresponding post-condition is searched. If it is empty (i.e. it is a new scenario) then the MLA determines the modality that is appropriate for the interaction context. Next, it determines which available media devices support the chosen modality. Using the complexity of the expression, the MLA finally decides the optimal presentation format. The chosen presentation format becomes the post-condition scenario and is recorded in the *scenario repository* (SR). Otherwise, the user condition is not a new scenario; hence the MLA simply retrieves the entry in the post-condition scenario. In (step 3), the chosen presentation format (e.g. Braille) is forwarded to the TA. The suitable media devices also are activated as shown in the bottom of Fig. 9.

F. The Presentation Layer

Here, we present a mathematical expression presentation in a chosen format.

1. The Translation Agent

Using the chosen presentation format sent by MLA, the *Translation Agent* (TA) converts the encoded expression into its final presentation. As shown in Fig. 10, the TA controller forwards the encoded expression to format translator(s) (i.e. *Braille*, *EasyMath*, *DotsPlus*, and *Speech*). See Fig. 11 , for the translation of specimen fraction into each of the 4 formats. The translated expression is then forwarded to the selected media devices and to SMA. The SMA then updates its mathematical expression repository by storing the new information (i.e. MathML expression, user interaction context, and the translated expression), implying that a new scenario (i.e. the current one) is encountered.

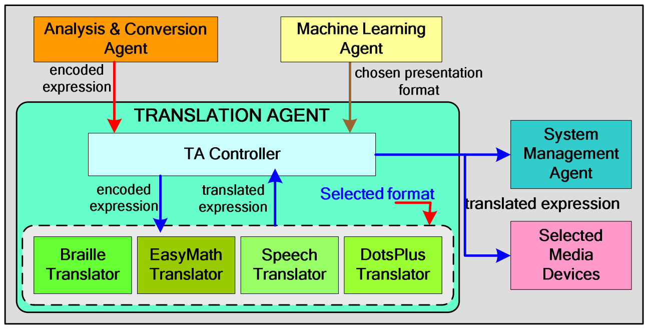


Fig. 10: The Translator Agent and its components.

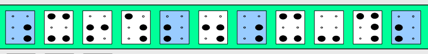
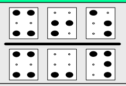
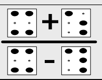
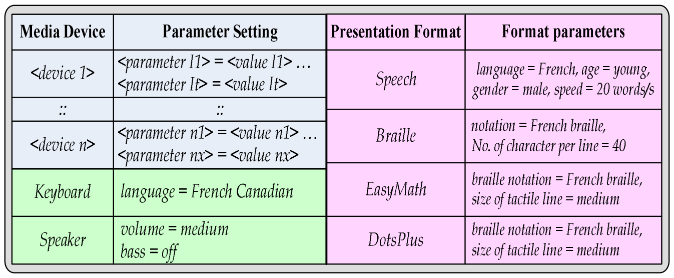
Encoded Expression	(/(+ x 5)(- x 4))
Presentation format	**Translated expression**
Braille	
EasyMath	
Speech	Open parenthesis x plus five close parenthesis divided by Open parenthesis x minus four close parenthesis
DotsPlus	

Fig. 11: The translation of a specimen fraction into 4 formats.

2. Parameters Setting of Presentation Formats

TA must know the parameters of every presentation format if it is to produce a correct translation of a mathematical expression. In general, values of these parameters are set by the user himself. For example, for speech presentation, parameters such as language, gender, and speed of audio message are presented as per user specification. In Fig. 12 (Right), the parameters of 4 presentation formats (and their sample values) are shown. The presentation format's parameter settings are part of user preferences as contained in the user profile.



Media Device	Parameter Setting
<device 1>	*<parameter I1> = <value I1> …* *<parameter It> = <value It>*
::	::
<device n>	*<parameter n1> = <value n1> …* *<parameter nx> = <value nx>*
Keyboard	*language = French Canadian*
Speaker	*volume = medium* *bass = off*

Presentation Format	Format parameters
Speech	*language = French, age = young, gender = male, speed = 20 words/s*
Braille	*notation = French braille, No. of character per line = 40*
EasyMath	*braille notation = French braille, size of tactile line = medium*
DotsPlus	*braille notation = French braille, size of tactile line = medium*

Fig. 12: Parameter settings for presentation formats and some media devices.

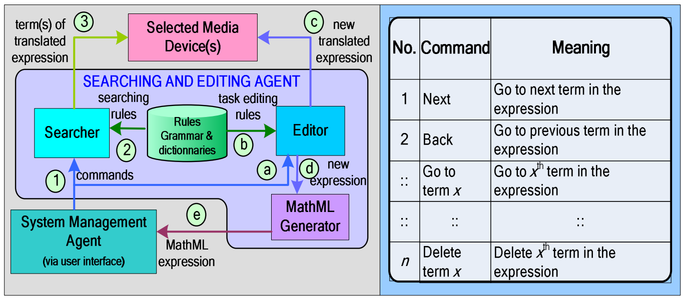
3. Parameters Setting of Selected Media Devices

The selected media devices can also be configured so that their settings suit the user's needs. Fig. 12 (Left) shows the generic format of media devices. As an example, the parameters of keyboard and speaker are set, as shown. During system activation, the parameters of each media device are set according to its media setting record by the device context negotiator. These settings, a part of user preferences, are relayed to the *Control Panel* of the operating system to effect the changes.

G. The Data Access Layer

1. The Searching and Editing Agent

The *Searching and Editing Agent* (SEA) allows navigation in and manipulation of a mathematical expression, controlling how mathematical terms are read and edited. SEA is composed of the following components: (1) the *Searcher*, (2) the *Editor*, and (3) the *MathML Generator*. SEA allows either sequential or random term access. Vocal commands for accessing mathematical terms are adopted because of their proven efficiency (i.e. cases of MathTalk and Meditor [19]). Via vocal commands, the user directly accesses an object, and perceives the modifications on the object via tactile and/or sound feedback. Fig. 13 illustrates the functionalities of SEA. Using user interface, the user can issue a vocal command (e.g. *go to <term 5>*). In (step 1), SMA sends user command to the Searcher. In (step 2), the Searcher verifies the validity of the command using grammar rules. If valid, the command is executed, and the result is sent to selected device(s) for presentation (step 3). Otherwise, no command is executed.


term(s) of translated expression
3
Selected Media Device(s)
c
new translated expression
SEARCHING AND EDITING AGENT
searching rules
task editing rules
Searcher
Rules Grammar & dictionnaries
Editor
2
b
1
commands
a
d
new expression
System Management Agent (via user interface)
e
MathML expression
MathML Generator

No.	Command	Meaning
1	Next	Go to next term in the expression
2	Back	Go to previous term in the expression
::	Go to term *x*	Go to x^{th} term in the expression
::	::	::
n	Delete term *x*	Delete x^{th} term in the expression

Fig. 13: Searching agent and editor agent cooperates with Device manager agent.

When the user edits an expression, the command (e.g. *Delete <term x>*) is sent by SMA to the Editor (step a). Upon command validation (step b), expression editing is executed and result is presented through selected device(s) (step c). This modification produces a new expression (step d). The MathML Generator produces the new MathML code and sent it to SMA for presentation in the user interface (step e). In Fig. 13 (Right), sample SEA commands are shown in tabular form. Note that when the presentation format is DotsPlus, search and modification of mathematical terms are not possible because the data are all embossed on paper (i.e. static media).

V. FAULT TOLERANT SYSTEM

A. *General Principle of a Fault-tolerant system*

In general, a fault-tolerant system is about the ability of a computing system to continue operating properly in the event of the failure of some of its components. It is designed to be able to handle several possible failures that may occur both in software-related faults such as communication between components or hardware-related faults such as input or output device failures.
In a fault-tolerant system, we have to proceed in three steps: 1) determine the set of faults to be tolerated; 2) choice of techniques which provide solution to the identified faults; and 3) test or experiment the efficiency of adopted techniques.

B. *Our Multi-agent Fault-Tolerant System*

The architecture of our system is designed to resist failure. When one or more faults (an agent or device is missing or defective) occur, the system would resist failure by self-reconfiguring. SMA reacts immediately to replace the failed component based on learned knowledge and users preferences (media devices priorities). Here, we can support 2 possible sources of failure: agent failure and defective or missing device.

1. A failed agent

In case of agent, SMA can perform various actions to prevent system crash. These actions are inspired from CONIC [22] and AAA [23]. The state of the agent determines action nature to be executed by the system. An agent can be in 1 of 4 states (see Fig. 14): 1) *Idle* – the agent is ready to execute actions. 2) *Running* – it is active and ready to accept and react with others queries. 3) *Disconnected* – when the agent is still alive but it does not reply correctly. 4) *Stopped* – the agent is missed and can not be repaired.

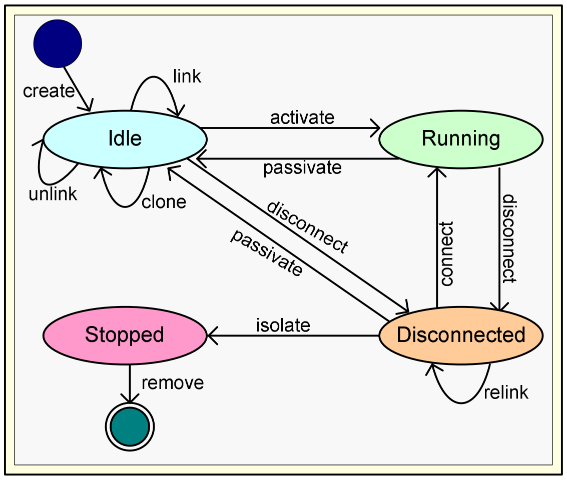


Fig. 14: All possible states of agent and primitives in our system.

In our system, the configuration is realised by using some primitives as shown on Fig. 13: 1) *create* – it is the primitive that allows adding an agent to the configuration, 2) *clone* – it is used to clone an agent in the configuration, 3) *link* – it creates a connection between 2 agents, 4) *unlink* – it removes the connection between 2 agents, 5) *re-link* – it reconstitutes the connection between 2 agents, 6) *isolate* – when an agent is not repairable, it is isolated to be deleted, and 7) *remove* – it destroys the stopped agent.

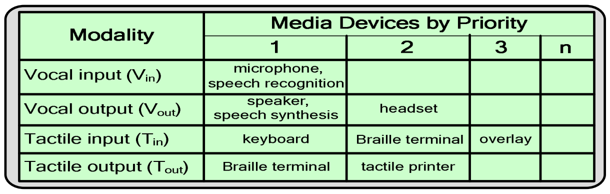
For example, if the agent A is failed, SMA tries to repair it based on its state. If there is a problem of communication between agents (i.e. agent is disconnected), primitives such as link, unlink and re-link are useful. If problem persists SMA assigns the works to the duplicate of the agent A that is already added to configuration at the beginning (i.e. using clone primitive). So the failed agent is isolated and then it is removed from the configuration.

2. A missing or defective device

Usually, when a media device is malfunctioning or absent (i.e. *failed*), the system searches the device (that is classified in the same group of modality) which is next in priority. If it is found, the replacement device is activated and the search is over. Otherwise, the system keeps searching for a replacement through *priority ranking order*.

Let there be a *media devices priority table* (MDPT) (see Table I) containing media devices grouped according to the modality they support and arranged by priority ranking. When our system implements a modality, it selects the media device(s) that is/are ranked top in priority. It is also through the MDPT that the system searches for a replacement to a failed media device.

TABLE I. A SAMPLE MEDIA DEVICES PRIORITY TABLE (MDPT)



Modality	Media Devices by Priority			
	1	2	3	n
Vocal input (V_{in})	microphone, speech recognition			
Vocal output (V_{out})	speaker, speech synthesis	headset		
Tactile input (T_{in})	keyboard	Braille terminal	overlay	
Tactile output (T_{out})	Braille terminal	tactile printer		

When a *new media device* $\mathbf{d_{new}}$ is *added* or *introduced* to the system for the *first time*, the device is associated to a modality and is given a priority ranking **r** by the user. What happen to the rankings of other devices $\mathbf{d_i}$, ($1 \leq i \leq n$, and n = number of media devices) which are in the same modality as $\mathbf{d_{new}}$ in the MDPT? Two things may happen, depending on the user's selection. The first possibility is that after having the new device's priority **Priority($\mathbf{d_{new}}$)** set to r then the priority of the other device **i**, ($1 \leq i \leq n$) denoted **Priority($\mathbf{d_i}$)**, remains the same. The second possibility is the priority rankings of all media devices ranked **r** or lower are adjusted such that their new priority rankings are one lower than their previous rankings. Formally, in Z [24], this is specified as: $\forall \mathbf{i}, \exists \mathbf{r}: \mathbb{N}; \forall \mathbf{d_i}, \exists \mathbf{d_{new}}: \textbf{Devices} \mid (\textbf{Priority}(\mathbf{d_{new}}) = \mathbf{r} \wedge \textbf{Priority}(\mathbf{d_i}) \geq \mathbf{r}) \Rightarrow \textbf{Priority}(\mathbf{d_i})' = \textbf{Priority}(\mathbf{d_i}) + \mathbf{1}.$

When a media device fault is detected and replaced by another one, the selection of the appropriate presentation format may be affected. Then ML must search the optimal presentation format based on the new situation. The process of selection of presentation format is discussed in [20].

VI. EXAMPLE OF SIMULATION WITH JADE

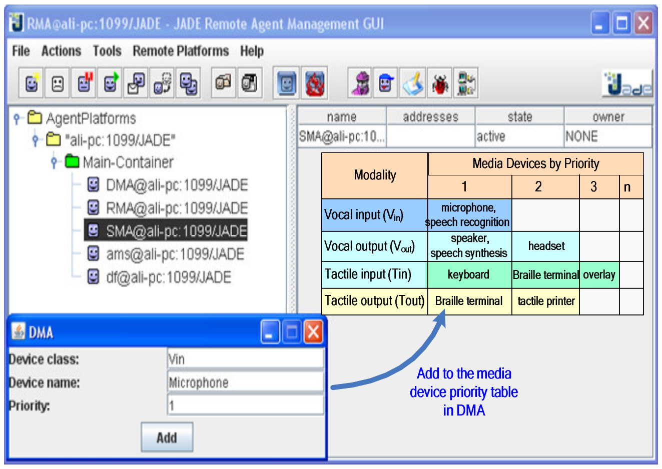
JADE [25] (Java Agent Development framework) is a software framework and middle-ware aimed at developing multi-agent applications conforming to FIPA standards. JADE is an Open Source project and has been coded in Java. Using this framework, a programmer should code his agents in Java. JADE provides an implementation for the following components:

- Agent Management System (AMS). This agent is responsible for controlling access to the platform, authentication and registration of participating agents.
- Directory Facilitator (DF). This agent provides a yellow page service to the agents in the platform.
- Agent Communication Channel (ACC). This agent provides a white page service. It also supports inter-agent communication and inter-operability within and across different platforms.

When a JADE platform is launched, AMS and DF are immediately created and ACC module is set to permit communication between agents by set of messages. The agent platform can be distributed on several hosts. AMS and DF live in the main-container that is an agent and it contains the RMI registry used internally by JADE. The other agents created should be connected to the main-container.

The components of a multi-agent system implemented with JADE communicate with each other using flexible and efficient messaging services. According to the FIPA specification, agents communicate via asynchronous message (i.e. Agent Communication Language ACL messages) and the communication between agents involves an exchange of ACL messages. ACL is conceived in a formal language that avoids any ambiguity.

Fig. 15 shows a Graphic User Interface (GUI) generated by the general management console for a JADE that is called RMA (Remote Management Agent). RMA provides control of all registered agent within platform, acquires information about the platform and executes GUI commands as create new agent, kill agent, etc. Through the RMA, a sniffer agent which is an important tool of JADE for monitoring and checking ACL messages exchanged among agents. When we sniff one agent or more, every message incoming/outgoing to/from agent is tracked and displayed in the Sniffer Agent's GUI in a diagram similar to UML sequence diagrams. There are other useful tools in JADE (Dummy Agent, Introspector) for monitoring and debugging the multi-agent systems. In this example, we present a simple simulation of 2 agents of our system: SMA and DMA.



name	addresses	state	owner
SMA@ali-pc:10...		active	NONE

Modality	Media Devices by Priority			
	1	2	3	n
Vocal input (V_{in})	microphone, speech recognition			
Vocal output (V_{out})	speaker, speech synthesis	headset		
Tactile input (Tin)	keyboard	Braille terminal	overlay	
Tactile output (Tout)	Braille terminal	tactile printer		

Device class: Vin
Device name: Microphone
Priority: 1

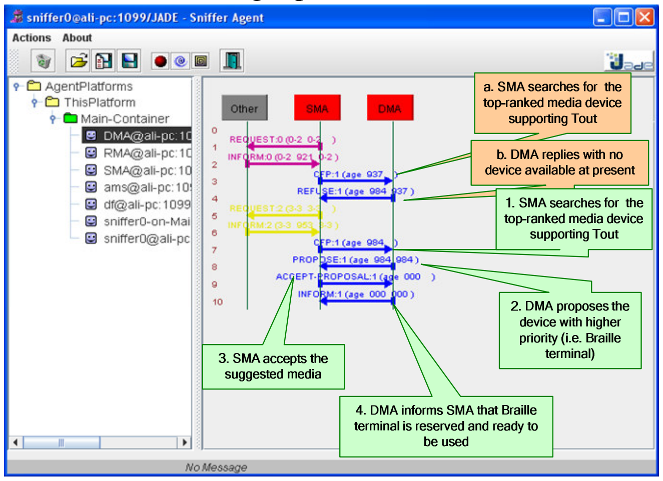
Fig. 15: A simple Jade simulation of SMA and DMA, also a sample MDPT.

In our simulation, we demonstrate the communication and the coordination between SMA and DMA. Usually, DMA communicates with all negotiators that are responsible to determine the status of each device and fills the MDPT. User must only specify its preferences.

However, we use a user interface for DMA in order to fill the MDPT as shown in Fig. 15.

At the beginning, SMA has a modality and searches to select the media device that is top-ranked in priority. To do that, SMA communicates with DMA as shown on Fig. 16.

Messages (a and b) represent the case when there are no device of modality T_{out} available in the MDPT. Messages (1,2,3 and 4) present a typical scenario when DMA has at least one device of the modality searched by SMA (i.e. here it is T_{out}). First, SMA looks to select the best available media device supporting T_{out} (a or 1) to DMA. If it is found, as in our example, it replies with a proposal message (2) that contains name (Braille Terminal) and rank (10) of the found media. Otherwise, it replies with a refuse message (b) to inform SMA that no device is available supporting T_{out}. In case of proposal message, SMA replies with an acceptance (3) and DMA informs (4) SMA that the media device (i.e. Braille Terminal) is ready to be used. Also, these steps are executed in finding replacement to a failed device.



0 REQUEST:0 (0-2 0-2)
1 INFORM:0 (0-2 921 0-2)
2 CFP:1 (age 937)
3 REFUSE:1 (age 984 937)
4 REQUEST:2 (3-3 3-3)
5 INFORM:2 (3-3 953 3-3)
6 CFP:1 (age 984)
7 PROPOSE:1 (age 984 984)
8 ACCEPT-PROPOSAL:1 (age 000)
9 INFORM:1 (age 000 000)
10

a. SMA searches for the top-ranked media device supporting Tout

b. DMA replies with no device available at present

1. SMA searches for the top-ranked media device supporting Tout

2. DMA proposes the device with higher priority (i.e. Braille terminal)

3. SMA accepts the suggested media

4. DMA informs SMA that Braille terminal is reserved and ready to be used

Fig. 16 sniffer agent monitors and checks ACL messages exchanged among agents SMA and DMA.

VII. CONCLUSION

Our ongoing research is focused on providing computing infrastructure to visually-impaired user through multimodality. One area of such domain is the infrastructure for mathematical presentation to blind users which this paper addresses. In this work, we presented an infrastructure supporting the presentation of mathematical expressions. Our multi-agent system considers the interaction context (i.e. combined user's, environment's and system's contexts) as well as the nature of the mathematical expression itself and of the user's preferences.

In this paper, we have presented the architecture of our adaptive multi-agent system. Also, we have presented the agents' functionalities in the system's layers. For each layer, we have shown the agents and their behaviour.

The architecture of our infrastructure is layered, thus encapsulating the components of the various layers. It is adaptive that it is capable of determining the best configuration (modality, media, and presentation) for the user. In case of failure of media device, our system is capable of shutting down the faulty component and replacing it with a new one (if a replacement is available). If replacement is not possible, the system re-determines the new modality and presentation format apt for the new configuration. The human intervention is greatly reduced in our system as it is capable of self-configuration, and learning. This system feature promotes autonomy to visually-impaired users, thus enhancing their information processing productivity.
In order to demonstrate the behaviour of the agents, a simulation has been carried out on the Java Agent Development Framework (JADE) platform. This simulation has confirmed the efficiency of our system design.

This work is our continuing contribution to advance research on making informatics more accessible to handicapped users. Our future works involve the prototyping of this infrastructure and simulating its performance using several computing platforms. Such prototype will also be tested on visually-impaired users with other varying interaction contexts.

REFERENCES

1. Stöger, B., K. Miesenberger, and M. Batusic, *Mathematical Working Environment for the Blind Motivation and Basic Ideas*, in *ICCHP*. 2004, Springer. p. 656-663.
2. Edwards, A.D.N. and R.D. Stevens. *A Multimodal Interface for Blind Mathematics Students*. in *INSERM*. 1994. Paris, France.
3. Cahill, H., et al., *Ensuring Usability in MATHS*, in *The European Context for Assistive Technology*. 1995, IOS Press: Amsterdam. p. 66-69.
4. Preddy, M., et al. *Dotsplus: How-to make tactile figures and tactile formatted math.*
5. Podevin, A., *Accès aux formules mathématiques par des personnes non voyantes : étude et définition d'une méthode adaptée*. 2002, Université de CAEN.
6. Ferreira, H. and D. Freitas, *Enhancing the Accessibility of Mathematics for Blind People: The AudioMath Project*, in *9th International Conference on Computer Helping People with Special Needs (ICCHP)*. 2004, Springer Lecture Notes in Computer Science (LNCS): Paris, France. p. 678-685.
7. Ferreira, H. and D. Freitas, *AudioMath: Towards Automatic Readings of Mathematical Expressions*, in *Human-Computer Interaction International (HCII)*. 2005: Las Vegas, Nevada, USA.
8. Raman, T.V., *Audio System for Technical Readings*. Vol. 1410. 1998, Berlin, Germany: Springer-Verlag.
9. Moço, V. and D. Archabault, *VICKIE: A Transcription Tool for Mathematical Braille*, in *7th European Conference for the Advancement of Assistive Technology in Europe (AAATE)*. 2003, IOS press: Dublin, Ireland.

10. Schwebel, F. and R. Goiffon. *BraMaNet: Quelques règles simples à connaître pour qu'un aveugle puisse lire vos documents mathématiques et vos pages web*. in *Journées nationales Caen*. 2005. Caen, France.
11. Garlini, P. and F. Fogarolo, *LAMBDA: Linear Access to Mathematics for Braille Device and Audio Synthesis - Analysis of the User's Needs*. 2003, University of Padova: Padova, Italy.
12. Lécuyer, A., et al. *HOMERE: a Multimodal System for Visually Impaired People to Explore Virtual Environments*. in *Proceedings of the IEEE Virtual Reality* 2003. Washington, USA: IEEE Computer Society.
13. Wooldridge, M., *An Introduction to Multiagent Systems*. 2002, Chichester, England: Wiley.
14. Weiss, G., *Multiagent systems.* MIT-Press, 1999.
15. Ferber, J., *Les systemes multi-agents*, ed. V.u.i. collective. 1995, Paris: InterEditions.
16. Jennings, N.R. and M.J. Wooldridge, *Applications of Intelligent Agents*, in *Agent Technology: Foundations, Applications, and Markets*, N.R. Jennings and M.J. Wooldridge, Editors. 1998, Springer-Verlag: Heidelberg, Germany. p. 3-28.
17. Bourbakis, N.G. and D. Kavraki. *An Intelligent Assistant for Navigation of Visually Impaired People*. in *the 2nd IEEE International Symposium on Bioinformatics and Bioengineering Conference*. 2001.
18. Awde, A., C. Tadj, and Y. Bellik, *Un système multi-agent pour la présentation d'expressions mathématiques à des utilisateurs non-voyants*, in *21ième Conférence Canadienne de génie électrique et génie informatique*. 2008, IEEE Canada: Niagara Falls, Ontario, Canada.
19. Bellik, Y., *Interfaces multimodales : concepts, modèles et architectures.*, in *LIMSI*. 1995, Université de Paris-Sud XI Orsay: Paris.
20. Awde, A., et al., *An Adaptive Multimodal Multimedia Computing System for Presentation of Mathematical Expressions to Visually- Impaired Users*, in *Journal of Multimedia (JMM)*. to be published.
21. Awde, A., Y. Bellik, and C. Tadj, *Complexity of Mathematical Expressions in Adaptive Multimodal Multimedia System Ensuring Access to Mathematics for Visually Impaired Users.* International Journal of Computer and Information Science and Engineering, 2008. **2**(2): p. 103-115.
22. Kramer, J. and J. Magee, *Analysing Dynamic Change in Software Architectures: A Case Study*, in *Proceedings of the International Conference on Configurable Distributed Systems* 1998, IEEE Computer Society: Washington, DC, USA.
23. Kumar, S. and P.R. Cohen. *Towards a Fault-Tolerant Multi-Agent System Architecture*. in *The fourth international conference on Autonomous agents* 2000: ACM Press.
24. Lightfoot, D., *Formal Specification Using Z*. 2nd ed. 2001: McMillan Press.
25. Bellifemine, F.L., G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. 2007: Wiley. 300.

Ali AWDE is currently a PhD student at the École de technologie supérieure (ÉTS). He obtained his master degree in Computer Science in 2003 from Université de Montréal. His research interests include multimodal multimedia for visually-impaired users, machine learning, and mathematics for the blind. Email: ali.awde.1@ens.etsmtl.ca.

Chakib TADJ is a professor at ETS, Canada. He received his PhD degree from ENST Paris in 1995. His main research interests are automatic speech recognition, human-machine interface, multimodal and neuronal systems. Email: *ctadj@ele.etsmtl.ca*.

Yacine BELLIK is an assistant professor at LIMSI-CNRS (*Laboratoire d'informatique pour la mécanique et les sciences de l'ingénieur*). He holds a PhD and HDR in Computer Science. His research interests concern multimodal human-computer interaction, aid for the blind and ambient intelligence. Email: *Yacine.Bellik@limsi.fr*.