VOL. 1, NO. 2, Oct 2010 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

http://www.cisjournal.org

Using Multimodal Fusion in Accessing Web Services

Atef Zaguia[1], Manolo Dulva Hina[1,2], Chakib Tadj[1], Amar Ramdane-Cherif[2,3]

[1]LATIS Laboratory, Université du Québec, École de technologie supérieure
1100, rue Notre-Dame Ouest, Montréal, Québec, H3C 1K3 Canada
[2]PRISM Laboratory, Université de Versailles-Saint-Quentin-en-Yvelines
45, avenue des États-Unis, 78035 Versailles Cedex, France
[3]LISV Laboratory, Université de Versailles-Saint-Quentin-en-Yvelines
45, avenue des États-Unis, 78035 Versailles Cedex, France
Email : manolo-dulva.hina.1@ens.etsmtl.ca

ABSTRACT

In our days, the technology allows us to produce extended and totally human-controlled multimodal systems. These systems are equipped with multimodal interfaces allowing a more natural and more efficient interaction between man and machine. End users could take advantage of natural modalities (e.g. eye gaze, speech, gesture, etc.) to communicate or exchange information with applications. The use of multimodal applications in web services, integrated with natural modalities, is an effective solution for users who cannot use a keyboard or a mouse, on users who have visual handicap, on mobile users equipped with wireless telephone/mobile devices, on weakened users, etc. Our work presents an approach in which various input modalities (speech, touch screen, keyboard, eye gaze, etc.) are available at user's disposition in order to access web services. While current state-of-the-art uses two (on rare cases, three) pre-defined modalities, our approach allows an unlimited number of concurrent modalities using semantic level called "multimodal fusion". Such approach gives user the flexibility to use the modalities as he sees fit for his situation. The detailed description of the proposed approach as well as the application that has been developed that uses these multimodalities is presented in this paper.

Keywords: *multimodal fusion, web service, human-computer interface.*

1. INTRODUCTION

Ever since the computer was born, one of the biggest challenges in informatics has always been the creation of systems that allow transparent and flexible human-machine interaction (Sears and Jacko 2007; Aim, Alfredson et al. 2009). Since the 1960's, computer evolution has been rapid and as always, the goal has been to satisfy the needs of users and come up with systems that are intelligent, more natural and easier to use.

Various researches have been directed towards the creation of systems that facilitate communication between man and machine (Yuen, Tang et al. 2002) and allow a user to use his natural modalities (eye gaze, speech, gesture, etc.) in communicating or exchanging information with applications. These systems receive inputs from sensors or gadgets (e.g. camera, microphone, etc.) and make an interpretation and comprehension out of these inputs; this is multimodality (Ringland and Scahill 2003; Ventola, Charles et al. 2004; Carnielli and Pizzi 2008; Kress 2010). A well-known sample of these systems is that of Bolt's "Put that there" (Bolt 1980) where he used gesture and speech to move objects.

In our days, various multimodal applications (Oviatt and Cohen 2000; Yuen, Tang et al. 2002) have been conceived and are effective solutions for users who, for one reason or another, cannot use a keyboard or a mouse (Shin, Ahn et al. 2006), on users who have visual handicap (Raisamo, Hippula et al. 2006) , on mobile users equipped with wireless telephone/mobile devices (Lai, Mitchell et al. 2007), on weakened users (Debevc, Kosec et al. 2009), etc. These applications are integrated with web services which are software components that represent an application function or service. The web services can be accessible from another application (a client, a server or another web services) within the Internet network using the available transport protocols (Li, Liu et al. 2007). This application service can be implemented as an autonomous application or a set of applications. It pertains to a technology allowing applications to communicate with one another via Internet, independent of the platforms (i.e. Windows (Microsoft 2010), Linux (Red_Hat_Enterprise 2010), Mac OS (Apple 2010), etc.) and languages (i.e. Java (Bates and Sierra 2003), C (King 2008), C++ (Malik 2010), J2EE (Weaver and Mukhar 2004), etc.) that these applications are based.

Our approach is to develop a flexible system based on application, capable of manipulating more than two modalities. The approach consists of modules that detect the modalities involved, take into account each modality's parameters and perform the fusion of the modalities in order to obtain the corresponding action to be undertaken. This approach is more flexible than current state-of-the art systems that run on predefined two modalities. The fusion of modalities involved in this work is based on the modalities selected by the user as he sees suitable to his situation.

The rest of this paper is organized as follows. Section II takes note of other research works that are related to

ours. Section III discusses the modalities and multimodal fusion system, section IV is about the discussion on the components of the multimodal fusion and their specification. The paper is concluded in section V; it also takes note of other works that we intend to do in the near future.

2. RELATED WORK

Modality refers to the path or channel by which human and machine interact with one another. An impoverished traditional computing set-up uses mouse, keyboard and screen by which the user interacts with the computer and vice-versa. The concept of multimodality allows the use of other means, or modality, in sending user input to the machine as well as the machine sending output to the user. Such modalities include other gadgets and sensors, such as touch screen, stylus, etc. and man's natural modalities, such as speech, eye gaze and gestures. The invocation of multimodalities permits a more flexible interaction between user and machine as well as allowing users with temporary or permanent handicap to benefit from the advancement in technology in undertaking computing tasks. Multimodality allows the invocation of other means when some other modalities are not available or possible to use. For example, speech (Schroeter, Ostermann et al. 2000) can be a more effective input modality than a mouse or a keyboard if the user is on the go. Using multimodality in accessing user application is an effective way of accomplishing user's computing task.

Web service (Ballinger 2003) is a software component that represents an application function (or application service). It is a technology that allows applications to interact remotely via Internet, independent of platforms and languages on which they are based. The service can be accessed from another application (a client, server or another Web service) through Internet using transport protocols. Web services are based on a set of standardizing protocols. Theses protocols are divided in to four areas, namely: the transport layer, the XML messages, the description of services and the search service.

Accessing web services using various modalities has been done and implemented in many applications for various services. For example, the work of (Caschera, D'Andrea et al. 2009) presents an effective web-based multimodal system (Madani, Nigay et al. 2005) that can be used in case of disasters, such as earthquake. In that work, the user sends and receives information using his mobile device through voice, touch or eye gaze (Zhang, Imamiya et al. 2004). This multiple-modalities system ensures simplicity of use and instant access to information that are pertinent in providing emergency services in critical situations, and consequently, save many lives. The work of (Steele, Khankan et al. 2005) demonstrates the concepts of discovery and invocation of services. The concept is illustrated using an example, that of a passenger in an airport. Here, the passenger can use his cell phone to know the available services in the airport. Using voice and touch, the user can browse and select his desired services. This system is significant as it satisfies the user's urgent needs for easy and quick access to information. In (Pfleger 2004), the author presents a system commonly used in house construction, namely the bathroom design. The multimodal system interface (Oviatt 2002) spontaneously integrates speech and stylus inputs. The output comes in the form of voice, graphic or facial expressions of a talking head displayed on screen. Here the user interacts with the system to get an ongoing assistance during the bathroom design. (Giuliani and Knoll 2008) presents a case of a human-robot multimodal interaction. Here, the two-armed robot receives vocal and non-verbal orders to make or remove objects. The use of such robots with remote control can be very beneficial especially in places where access is dangerous for human beings. In (Wang, Zhang et al. 2006), the authors proposed a multimodal system that helps children learn the Chinese language through stylus and voice. The system presents a good way for educating children as it helps them learn while playing.

In a nut shell, it could be said that the above-mentioned multimodal systems are of great importance to the users in terms of efficiency. However, the very fact of their being based on only two (and on rare occasion, three) modalities makes the use of most of these systems challenging for users. This leads us to the creation of a multimodal system with an unlimited number of modalities in order to provide an easier interface access and simpler usage for the users.

3. MODALITIES AND MULTIMODAL FUSION SYSTEM

In this section, we describe the approach that we propose and all other modules involved in the implementation of such approach.

A. Architecture and Approach

Accessing a web service involves the use of four web-service modules. These modules need to be loaded on a computer, on a robot, or any machine that can communicate via Internet or social network. The architecture of our proposed system is shown in Figure 1.

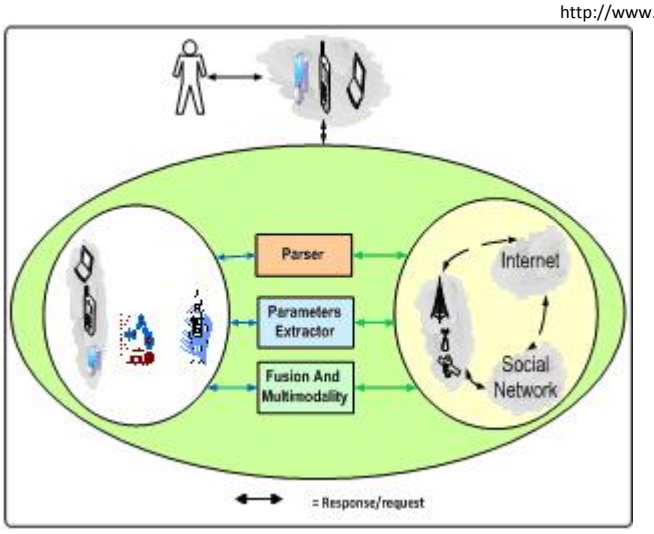

Figure 1. Architecture of multimodal fusion system for accessing web services

As shown in the diagram, the system consists of the following elements:

- *Parser* – it takes an XML file as input, extracts information from it and yields an output indicating the concerned modality and its associated parameters
- *Parameter Extractor* – the output from the parser serves as input to this module, it then extracts the parameters of each involved modality
- *Fusion and Multimodality* – based on the parameters involved in each modality as well as the time in consideration, it decides if fusion is possible or not
- *Internet/Social Network* – serves as the network by which the user and the machine/computer involved communicate
- *Computing Machine/Robot/Telephone* – this is the entity by which the user communicates with

The processes involved in the multimodal fusion are shown in Figure 2. Two or more modalities may be invoked by the user in this undertaking. Consider for example the command "*Replace this file with that file*" wherein the user uses speech and a mouse click to denote "*this file*" and another mouse click to denote "*that file*". In this case, the modalities involved are: *input modality 1 = speech* and *input modality 2 = mouse*. The processes involved in the fusion of these modalities are as follows:

- *Recognition* – this component converts the activities involving modalities into their corresponding XML files.
- *Parser Module*, *Parameter Extraction Module* and *Multimodal and Fusion Module* – same functionalities cited earlier
- *Action* – this involves the corresponding action to be undertaken after the fusion has been made. The resulting output may be implemented using output modality 1, 2, …, *n*. In the same case cited earlier, the *output modality involved is the screen*. It is also possible that the confirmation of such action may be presented using a *speaker*.
- *Feedback* – when conflict arises, a user receives a feedback from the system. For example, in the cited case, if "*this file*" and "*that file*" refer to the same entity, the user is informed about it via feedback.

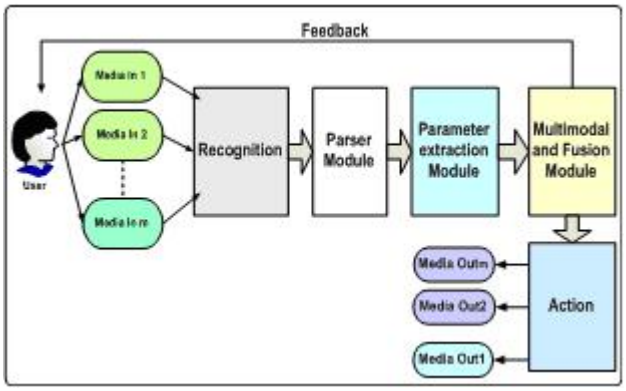

Figure 2. Framework of multimodal fusion

All of these modules may have been installed or are situated in any location within the network. The modules themselves communicate with one another in order to exchange information or do a task such as instruct a robot to do something.

B. Multimodal Fusion

Fusion (Pfleger 2004; Pérez, Amores et al. 2005; Lalanne, Nigay et al. 2009) is a logical combination of two or more entities, in this case two or more modalities. Modality signals are intercepted by the fusion agent and then combine them based on some given semantic rules.

Based on literature review, two sets of fusion schemes exist: the *early fusion* and the *late fusion* (Wöllmer, Al-Hames et al. 2009*). Early fusion* (Snoek, Worring et al. 2005) refers to a fusion scheme that integrates unimodal features before learning concept. The fusion takes effect on signal level or within the actual time that an action is detected (Oviatt, Cohen et al. 2000). On the other hand, *late fusion* (Mohan, Dhananjaya et al. 2008) is a scheme that first reduces unimodal features to separately learned concept scores, and then these scores are integrated to the learned concepts. The fusion is effected on semantic level. In this work, the fusion process used is the late fusion.

Assume for instance the arrival of modality A, along with its parameters (e.g. time, etc.) and another modality B with its own parameters (e.g. time, etc.), then the fusion

agent will produce a logical combination of A and B, yielding a result, we call C. The command/event C is then sent to the application or to the user for implementation. The multimodal fusion can be represented by the relationship f: C = A + B.

Figure 3 describes the steps involved in the fusion process. As shown, an *n* number of XML files involving modalities serve as input to the system. Recall that each XML file contains data related to modalities as well as their corresponding parameters. The steps undertaken are as follows:

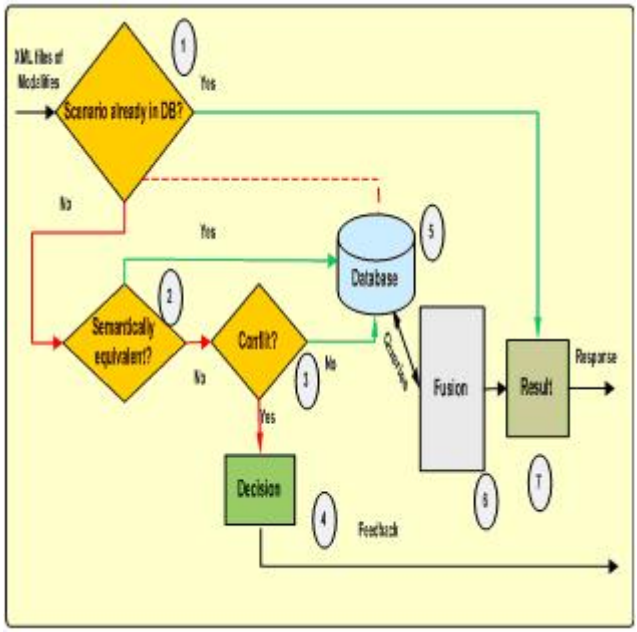

Figure 3. The detailed fusion process

- *Step 1:* The system begins asking if the scenario (i.e. the condition) is already in the database. If so, we already know the result (also stored in the database); it is retrieved and is to be implemented. Otherwise, step 2 is performed.
- *Step 2:* This indicates that the scenario is new to the system. The system asks if the semantic of the operations just performed or about to be performed using the involved modalities are equivalent. Consider for example a case in which, using speech, the user says: "*Write 5*" and also using keyboard, he types "*5*". Technically, the two actions mean the same and are therefore semantically equivalent. The result means that there is no fusion to be made. The information will be stored in the database for reference in case such event happens again in the future. The desired action is that only one of these commands – one or the other – will be implemented.
- *Step 3:* There is a conflict if two or more commands are in contradiction with one another. For example, if the user, using speech, says: "*Write 5*" and using stylus, for example, he writes "*4*".
- *Step 4:* A feedback is sent to the user to correct a detected conflict. If the user takes no action, the system continues on sending user a feedback for the resolution of the problem. The user needs to decide which of the two commands (or *n* commands, in general) – one or the other – must be considered by the system for implementation.
- *Step 5:* If the scenario is completely new and there is no conflicting data and action involved, all the corresponding data will be stored in the database.
- *Step 6:* Queries are sent to the database by the fusion agent to retrieve information involving modalities. The fusion is then performed and the result is also stored in the database.
- *Step 7:* Result refers to the desired action to be performed using the modalities involved.

As shown in Figure 4, the fusion agent is composed of three sub-components, namely:

- *Selector* – it interacts with the database in selecting the desired modalities. It retrieves 1 .. *m* modalities at any given time.
- *Grammar* – verifies the grammatical conditions and all the possible interchanges between the modalities involved.
- *Fusion* – this is the module that implements the fusion function.

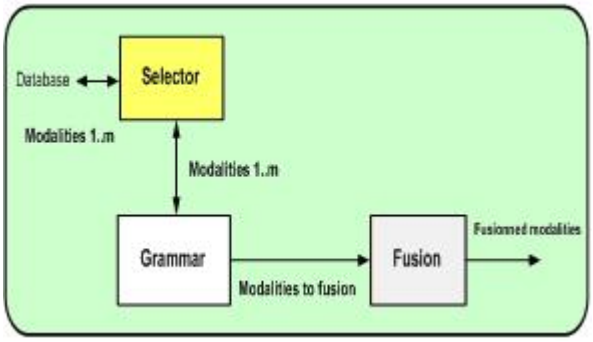

Figure 4. The fusion agent

The algorithm of the fusion process is shown in Figure 5. The essence of time is important in determining whether the actions involved concerning modalities are qualified for fusion not.

The process begins with accessing the database and taking the data associated with one modality (step 1). The next step is to check the grammatical implication of the associated modality (step 2). Here, the implication analysis could determine if the action to be taken involving such modality could be interpreted as a unimodal or multimodal action (needing complementary data from another modality). For example, the speech command "*Put that there*" implies that some data associated with "*that*" and "*there*" are expected. On the other hand, a speech command such as "*Draw a box*" implies that the command can stand on its own – its meaning is complete, needing no complementary data – and therefore should be taken as a

unimodal activity. If the desired action requires multimodal inputs, then fusion is performed whereas if it is a unimodal one, no fusion should be performed. In both of these cases, their results are stored in the database for future reference. As the diagram shows, the fusion process is a continuous process, always in effect while the user is logged on into the computing system.

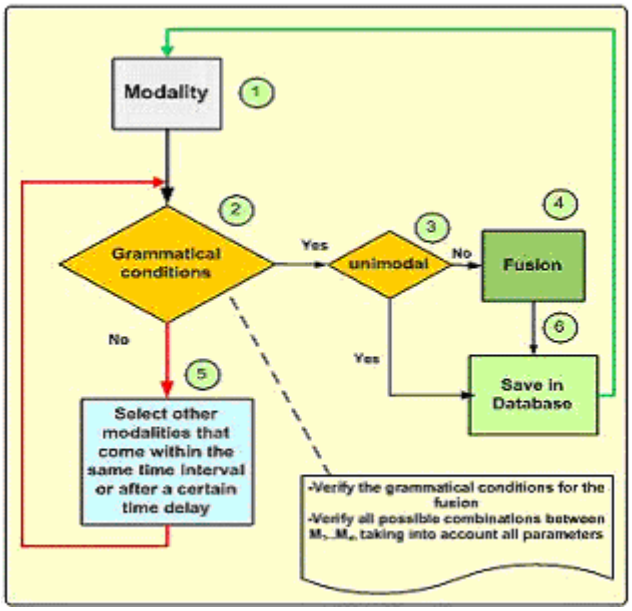

Figure 5. The fusion algorithm

Failure in grammatical conditions may also arise. For example, a vocal command "*Put there*" is a failure if there is no other complementary modality – such as touch, eye gaze, mouse click, etc. – is associated with it. If such case arises, the system looks at some other modalities that come within the same time interval as the previous one that was considered.

C. Time Element in the Fusion Process

Here, we shall illustrate various variations of time frame which will decide if fusion is possible or not. For all the cases involved, the following notations shall be used: M_i = modality *i*, t_1M_i = arrival time of modality *i*, t_2M_i = end time of modality *i*, t_{min} = minimum time delay allowed and is dependent of the modality involved. With reference to Figure 6, here are some of the possible cases:

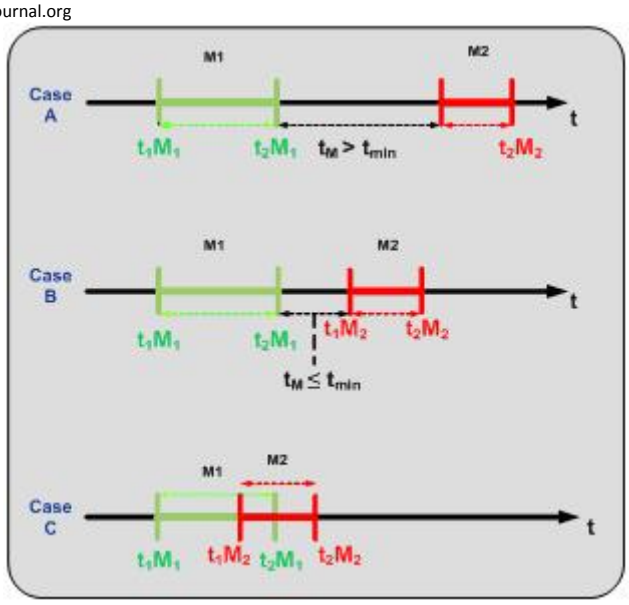

Figure 6. The time element involved in two modalities

- *Case A:* Modality 1 arrives at t_1M_1 and ends at t_2M_1. Another modality 2 arrives at t_1M_2 and ends at t_2M_2. They, however, occur on different time frame and are mutually exclusive of one another. As they are independent of one another, each modality is unimodal. Each modality is treated separately. Hence, f_1: $C_1 = M_1$ and f_2: $C_2 = M_2$
- *Case B:* Modality 1 and modality 2 arrive separately but the arrival time of these two modalities is lesser than t_{min}. In this case, a test is needed to determine if fusion is possible. In the affirmative case, the resulting function would be: f: $C = M_1 + M_2$
- *Case C:* Modalities 1 and 2 arrive within the same time frame. Here, fusion is obvious and the resulting function is f: $C = M_1 + M_2$.

4. COMPONENTS OF MULTIMODAL FUSION AND THEIR SPECIFICATIONS

In this section, we present the different components that are involved in the multimodal fusion process and provide each component's specification. The formal specification tool Petri Net as well as an actual program in Java are used to elaborate on the system components' specifications.

A. The User Interface

As true with every computing system, our system has a user interface (Oviatt and Cohen 2000) with which the user can communicate to the computing system. In the user interface, the user can select modalities that he wishes. An event concerning the modality is always detected (e.g. *was there a mouse click? was there a vocal input?*, etc.). The system keeps looping until it senses an

event involving modality. The system connects to the database and verifies if the event is valid. An invalid event, for example, is a user's selection of two events using two modalities at the same time when the system expects that only one event can be executed at a time. If the event involving modality is valid, an XML file is created for that modality and its associated parameters. The XML file is forwarded to the parsing module. The parser then extracts data from the XML tags and sends the result it obtained to the Multimodal and Fusion module. The diagram in Figure 7 demonstrates this process.

B. The Parser

This module receives as input XML files containing data on modalities, usually captured using sensors (e.g. webcam, micro, touch screen, etc.). From each XML file, this module extracts some tag data that it needs for fusion. Afterwards, it creates a resulting XML containing the selected modalities and each one's corresponding parameters.

In conformity with W3C standard on XML tags for multimodal applications, we use EMMA. EMMA (Desmet, Balthazor et al. 2005; Johnston 2009; W3C 2010) is a generic tagging language for multimodal annotation. It is an integral part of the W3C norm for multimodal interactions. Its general objective is to automatically represent the information extracted from user inputs through interpretation of system components. The EMMA tags represent the semantically recovered input data (e.g. gests, speech, etc.) that are meant to be integrated to a multimodal application. EMMA was developed to allow annotation of data generated by heterogeneous input media. When applied on target data, EMMA result yields a collection of multimedia, multimodal and multi-platform information as well as all other information from other heterogeneous systems.

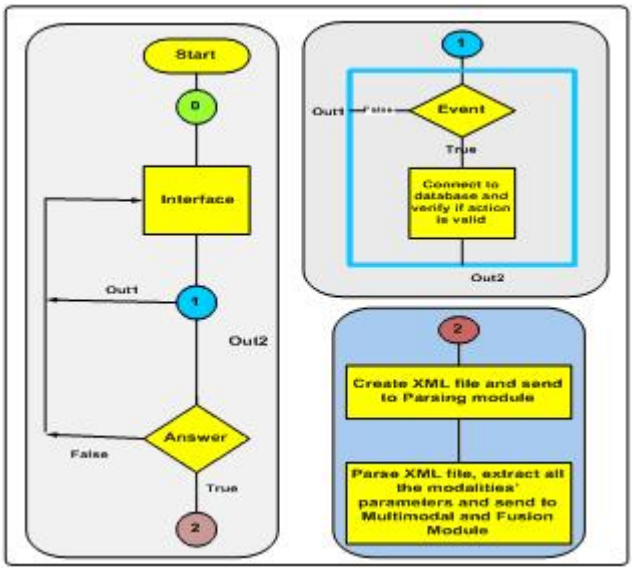

Figure 7. The system's user interface

For instance, assume that the module receives the following XML files in EMMA notation (see Figure 8). The corresponding EMMA tags and their meanings are as follows:

- <***emma:emma***> is the root node, holding EMMA version and namespace information, and providing a container for one or more of the following elements:
- <***emma:interpretation***> is used to define a given interpretation of input, and holds application specific markup;
  - ***id*** is a required xsd:ID value that uniquely identifies the interpretation within the EMMA document.
- <***emma:group***> is a general container for one or more interpretations. It can be associated with arbitrary grouping criteria; the emma:group element is used to indicate that the contained interpretations are related in some manner.
- The ***emma:model*** annotation may refer to any element or attribute in the application instance data, as well as any EMMA container element (emma:one-of, emma:group, or emma:sequence).
- The ***emma:lang*** annotation is used to indicate the human language for the input that it annotates. The values of the emma:lang attribute are language identifiers. For example, ***emma:lang="fr"*** denotes French while ***emma:lang="en-US"*** denotes US English
- The confidence score in EMMA is used to indicate the quality of the input, and it is the value assigned to ***emma:confidence*** in the EMMA namespace
- ***emma:start*** and ***emma:end*** are attributes indicating the absolute starting and ending times of an input in terms of the number of milliseconds

Using speech and touch screen specimen modalities shown in Figure 8, the resulting combined XML file is similar to the one shown in Figure 9 (Left). Then the fusion of these two modalities yields the result that is shown in Figure 9 (Right). The fusion result indicates that the object cube is moved to location (a,b).

```
<emma:emma>
  <emma:group
      emma:medium="acoustic,tactile"
      emma:mode="voice,touch" emma:function="dialog">
      <emma:interpretation id="speech1"
       emma:confidence="0.9" emma:verbal="true"
       emma:start=" 2010-03-26T0:00:00.1 " emma:end=" 20010-03-26T0:00:03.1 "
       emma:medium="acoustic" emma:mode="voice"
       emma:confidence="0.8" emma:lang="en-US"
       emma:process="smm:type=asr&version=asr_eng2.4"
       emma:media-type="audio/amr; rate=8000">
       <emma:literal>Put this cube here
       </emma:literal>
      </emma:interpretation>
      <emma:interpretation id="touch1"
            emma:confidence="0.8"
            emma:medium="tactile" emma:mode="touch"
            emma:start=" 2010-03-26T0:00:01.6 "
            emma:end=" 2010-03-26T0:00:02.1 ">
              <x>0.253</x>
              <y>0.124</y>
      </emma:interpretation>
      <emma:interpretation id="touch2"
            emma:confidence="0.9"
            emma:medium="tactile" emma:mode="touch"
            emma:start=" 2010-03-26T0:00:03.5 "
            emma:end=" 2010-03-26T0:00:04.1 ">
              <x>3.768</x>
              <y>2.324</y>
      </emma:interpretation>
    <emma:group-info>temporal</emma:group-info>
  </emma:group>
</emma:emma>
```

Figure 8. Specimen XML files containing modalities (speech and touch screen)

```
<combine>
  <speech startTime= '500' endTime= '1000'>
    <action>move</action>
    <object> cube</object>
    <aim>put it in position</aim>
  </speech>
  <touch startTimet= '1010' endTime= '1015'>
    <position>
      <X> a </X>
      <Y>b</Y>
    </position>
  </touch>
</combine>
```

Left: XML file showing combined modalities

```
<Fusion>
  <action>move</action>
  <objet>cube</objet>
  <old position>
    <X>i </X>
    <Y>j</Y>
  </old position>
  <new position>
    <X> a </X>
    <Y>b</Y>
  </new position>
</Fusion>
```

Right: XML file after the fusion

Figure 9. (Left) The resulting combined XML file and (Right) the resulting XML file after the fusion.

C. The Parameter Extractor

The manipulation of an XML file is a task usually performed within the development phase of an application. It is usually undertaken by a parser. A XML parser is a library of functions that can manipulate on an XML document. In selecting a parser, we usually look for two characteristics – that of parser being efficient and rapid.

The parser used in this system is called DOM (Document Object Model) (Wang, Li et al. 2007). It is a large, complex and stand-alone system that uses object model to support all types of XML documents. When parsing a document, it creates objects containing trees with different tags. These objects contain methods that allow a user to trace the tree or modify its contents. See Figure 10.

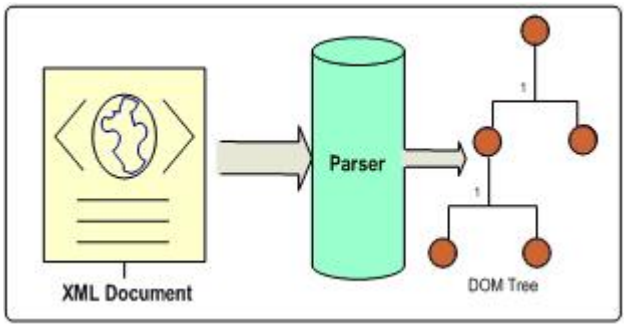

Figure 10. DOM parsing XML document

DOM works in two steps. The first step involves the loading of an XML document (this is also when DOM consumes memory space). The second step involves performing different operations on the document. The advantages of using DOM are: (1) the easy traversal of its tree, (2) easy way of modifying the contents of the tree, and (3) traversal of file in whatever direction the user desires. On the other hand, its disadvantages include: (1) large consumption of memory and (2) processing of the document before using it.

Using the same example cited in the previous sub-section, the resulting DOM tree after the parsing process is shown in Figure 11.

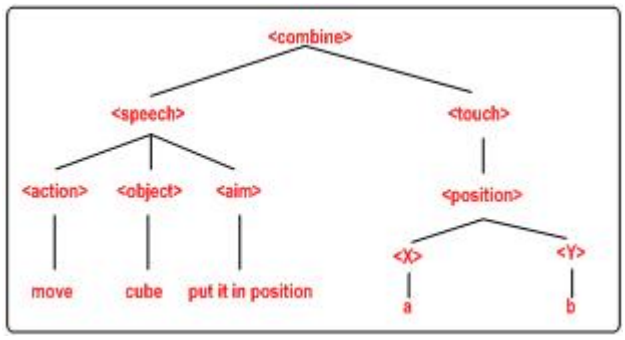

Figure 11. A sample DOM tree

D. The Database

The database stores all modalities identified by the system and the modalities' associated parameters. There are many available databases, such as PostgreSQL, MySQL, SQL Server, Oracle, Access, etc. In this work, the database used is PostgresSQL (PostgreSQL 2010). Using PostgresSQL, the parameters, values and entities of the database are defined dynamically as the module parses the XML file.

As shown in Figure 12, our database consists of seven tables, namely:

- *Modality* – this table contains the names of modalities, the time an action involving the modality begins and the time that it ended.
- *Modality_Added_Parameters* – this table contains all the attributes of every modality.
- *Modality_Main_Parameters* – contains the name of all parameters and their values
- *Union_Modality_Main_Parameters* – this table links the modality and their parameters
- *Fusion* – this table contains all the fusions that had been implemented. This table allows us to keep the previous historical data that can be used later for learning.
- *Fusion_Main_Parameters* – contains the names of parameters and their values that are associated with the multimodal fusion.
- *Union_Fusion_Main_Parameters* – this table serves as a link to the multimodal fusion that was just made, including its corresponding parameters

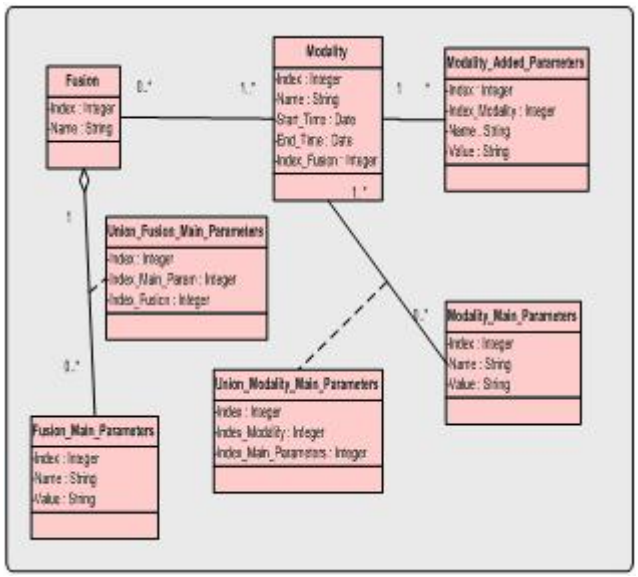

Figure 12. Tables that make up the database

To demonstrate what data gets stored into the database, let us cite an example using the following XML file (see Figure 13):

```
<combine>
 <speech startTime= '10:11:11' endTime = '10:12:13'>
      <action>move</action>
       <object>yellow box</object>
        <goal>put in POSITION</goal>
   </speech>
<click startTime= '10:12:00' endTime= '10:13:01'>
       < positionX> a </ positionX>
       < positionY>b</ positionY>
 </click>
</combine>
```

Figure 13. Specimen XML file

Hence, given the cited case (Figure 13), the contents of the *Modality* table are shown in Table 1.

Table 1. A specimen Modality table

Index	Name	Start_Time	End_Time	Index_ Fusion
1	Speech	10 :11 :11	10 :12 :13	1
2	Mouse click	10 :10 :11	10 :12 :13	1

The contents of this table and their meanings are:

- *Index* – this is a redundant element used by database management for optimizing requests. In our case, it represents the index for each modality.
- *Name* – this column tabulates all the names of all modalities.
- *Start Time* – indicates when a modality is started.
- *End Time* – when a modality activity ended.
- *Index_Fusion*: indicates the index of the fusion table. This column is filled with the necessary data during the fusion process to denote the modalities that have been fused.

The content of the Modality_Main_Parameters table, for the cited case is shown in Table 2. The contents and meanings of each element in this table are as follows:

- *Index* – shows one index for every parameter of a modality.
- *Name* - provides the name of each parameter.
- *Value* – identifies the value of each parameter.

Table 2. A specimen Modality_Main_Parameters table

Index	Name	Value
1	Action	move
2	object	Yellow
3	aim	Put in position
4	Position X	a
5	Position Y	b

Likewise, the Union_Modality_Main_Parameters table contents, for the cited case, are shown in Table 3. The items in this table are:

- *Index* – same as earlier definition.
- *Index_Modality* – shows the index to the table "Modality".
- *Index_Main_Parameters* – shows the index to the table "Modality Index Parameters".

Table 3. A specimen Union_Modality_Main_Parameters

Index	Index_Modality	Index_Main_Parameters
1	1	1
2	1	2
3	1	3
4	2	4
5	2	5

The contents of the Fusion table, of the Union_Fusion_Main_Param table, and that of the Fusion_Main_Param are all shown in Table 4, Table 5, and

Table 6, respectively.

The meanings of the columns in Table 4 are as follows:

- *Index* – same as before.
- *Name_Fusion* – identifies the name of the fusion.

Table 4. A specimen Fusion table

Index	Name_Fusion
1	fusion 1
2	fusion 1

The contents of Table 5 are as follows:

- *Index* – same meaning as before.
- *Index_Param* – shows the index of the parameters of the table "Fusion_Main_Param".
- *Index_Fusion* - tells us the index number of the fusions associated to each parameter.

Table 5. A specimen Union_Fusion_Main_Param table

Index	Index_Param	Index_Fusion
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1

In

Table 6, the meanings of its columns are:

- *Index* – same as before.
- *Name* – present the names of each parameter involved in the fusion.
- *Value* – present the values of each parameter involved in the fusion.

Table 6. A specimen Fusion_Main_Param table

Index	Name	Value
1	action	move
2	object	cube
3	old position X	i
4	old position Y	j
5	new position X	a
6	new position Y	b

In Table 7, the meanings of its columns are:

- *Index* – same as before.
- *Index_Modality* – the data here serves as index to the table "Modality".
- *Name* – lists down the name of the attribute of each modality.
- *Value* – lists down the value of value of the attribute of each modality.

Table 7 contains the attributes of each modality. In the cited case, however, we did not use attributes therefore the table column is empty.

Table 7. Modality_Added_Parameters

Index	Index_Modality	Name	Value
1			

E. Sample Case and Simulation using Petri Net

Here, we will demonstrate a concrete example and describe its specification using Petri Net. We will briefly discuss the basic concepts of Petri Nets in order that the next diagrams involving it would be clear and easily comprehensible.

Petri Net (ISO/IEC-15909-2 2010; Petri-Nets-Steering-Committee 2010) is an oriented graph. It is a formal, graphical, executable technique for the specification and analysis of a concurrent, discrete-event dynamic system. Petri nets are used in deterministic and in probabilistic variants; they are a good means to model concurrent or collaborating systems. They also allow for different qualitative or quantitative analysis that can be useful in safety validation. The symbols and notations used by Petri Net are shown in Figure 14. Places (represented by circles) are states in a simulated diagram whereas transitions (represented by rectangles) are processes that are undertaken by a certain element. A certain element goes from one state to another through a transition. Usually a certain element begins in an initial state (manifested in Petri Net via an initial token in a place). When an element goes from state "*a*" to state "*b*" through a transition, it is

manifested in the Petri Net via a movement of token from place *"a"* to place *"b"* via transition *"x"*.

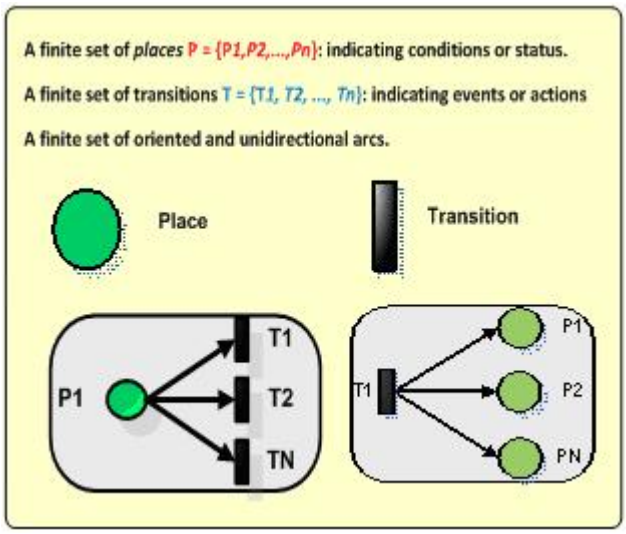

Figure 14. Petri Net symbols and notations

In the specifications that will follow in this paper, only a snapshot of one of many possible outcomes is presented; this is due to space constraints in which we do not have the space to show all possible variations of inputs and corresponding outputs. The application software PIPE2 is used in simulating Petri Net. PIPE2 (Bonet, Llado et al. 2010) is an open source, platform independent tool for creating and analysing Petri nets including Generalised Stochastic Petri nets.

As shown in Figure 15, the sample application is about car rental. As shown in Figure 15(A), the menu is composed of four selections – the rental option, the sign-up option, the manage reservation option and the usage option. For simplicity of the discussion, the usage option, as shown in Figure 15(B), allows the user to select and identify his preferred modalities. In this example, we listed 4 specimen modalities, namely: (1) voice, (2) touch screen, (3) keyboard and (4) eye gaze. When the user signs up for a rent, the rental period needs to be specified, hence, in Figure 15(C), the interface allows the user to specify the month, the day, and the time for both the start and end of the rental period. Finally, in Figure 15(D), we provide an interface which allows the user to input his coordinates as well as the credit card information.

To make use of the above-mentioned application, we will cite a sample case. Wishing too know the country better, four students have decided to take a trip in a weekend. They decided to rent a car for them to see some regions of the country. One night, François opened his computer equipped with touch screen and connected himself to a rent-a-car website. Using speech, touch screen, eye gaze and keyboard, he was able to do a car reservation for the following week.

During the reservation process, some XML files are created, with different modalities used. These files are sent to the server of the car rental enterprise within the Internet network using the http protocol. These files are to be sent first to the "Parser" module to extract all the modalities involved. Then this module will create another XML file that contains all the different modalities as well as their corresponding parameters. Then this file will be sent to the "Parameter Extractor" module which will extract all the parameters of the modalities involved and send them to the "Fusion" module.

E.1. Scenario

François runs the rent-a-car application software. The initial interface is displayed. Using voice, he selected "Rent". Then the second interface is presented; using touch screen, he chose "Start Day". Using keyboard, he types "25". Then using eye gaze, he selected "Start Month" and via speech, he said "January". Then using keyboard, he selected "Start Time" and he entered "1:30 pm" using keyboard. Then "End Day" is selected using speech and he uttered "27". Using keyboard, he selected "End Month" and types in "1". At the end, using eye gaze, he chose "End Time" and said "6:00 pm". At the end of the reservation process, he received a confirmation message through his laptop computer. This scenario is depicted in the diagram of Figure 16.

E.2. Grammar

The diagram in Figure 17 shows the grammar used for the interfaces A and B (see Figure 15) of the sample rent-a-car application. The choice, for instance, is defined as a selection of one of the menus (i.e. rent, sign up, manage reservation and usage) in the user interface. The choice of time is in American time format (example: 12:30 pm); choice of month can be numeric (e.g. 1) or alphabetic (e.g. January) and the interface2, among others, is defined as a selection of either an entry of month, day or time or a back/next/cancel selection.

E.3. Simulation 1

The diagram in Figure 18 shows the interactions involved in interface A in which the user would have to choose one option in a rent-a-car menu. The rest of the diagram demonstrates all activities when and after the user chooses "Rent" via speech. As shown, an XML file is created which is then sent to the network. The Parser module parses the XML data and extracts tags that contain modality information including its associated parameters. The parameter extractor module extracts the necessary parameters and is then forwarded to the Multimodal and Fusion Module. As it is a unique action, in this example, no fusion is implemented. It is a unimodal action. Nonetheless, it is saved onto the database and the interface B and all menus associated with the "Rent" option are to be instantiated.

E.4. Simulation 2

The diagram in Figure 19 demonstrates a net showing the system activities when the "Rent" option is selected (see the token in the place called "Rent Selected" in the middle of the net). At the same time that this option is selected, the three modalities are also selected (see the tokens in keyboard modality, speech modality and touch screen modalities). The Petri Net diagram shows us all possible variations that could arise. For example, when the keyboard is selected (upper right corner of the net), the next possible transition may be one of the following: (1) Select month by keyboard, or (2) Select Day by keyboard or (3) Time selected by keyboard. The same can be said of the two other modalities – speech and touch screen. Here, our desired output is a data entry for month, day or time which needs to be implemented using only one modality per parameter. For example, month selected by two or more modalities is invalid. In the diagram, a snapshot of one of the many possible outcomes is shown – here, the month is selected by touch screen, day is chosen using keyboard and time is chosen via speech. We colour the states for easy viewing – yellow is associated with speech, blue for keyboard modality and pink for touch screen modality; the red circle denotes that it is assumed that the "Rent" option is already selected.

E.5. Simulation 3

The diagram in Figure 20 shows the intricate Petri Net demonstrating various states and transactions as the user uses four modalities (e.g. keyboard, speech, touch screen and eye gaze) in selecting and entering specimen data for "Start Month", "Start Day" and "Start Time". This diagram partly shows the Petri net involved in implementing the activities as shown in Figure 16. Again, for the purpose of simplicity, we put colours on the places of the net: (1) blue – for the multimodal transactions in which touch screen is used to select "Start Day" and keyboard to enter "25", (2) green – to show the places and transitions involved when user uses eye gaze to select "Start Month" and speech to input "January", and (3) pink – in which "Start Time" is selected using eye gaze, "1:30" is uttered vocally, and "pm" is entered via keyboard.

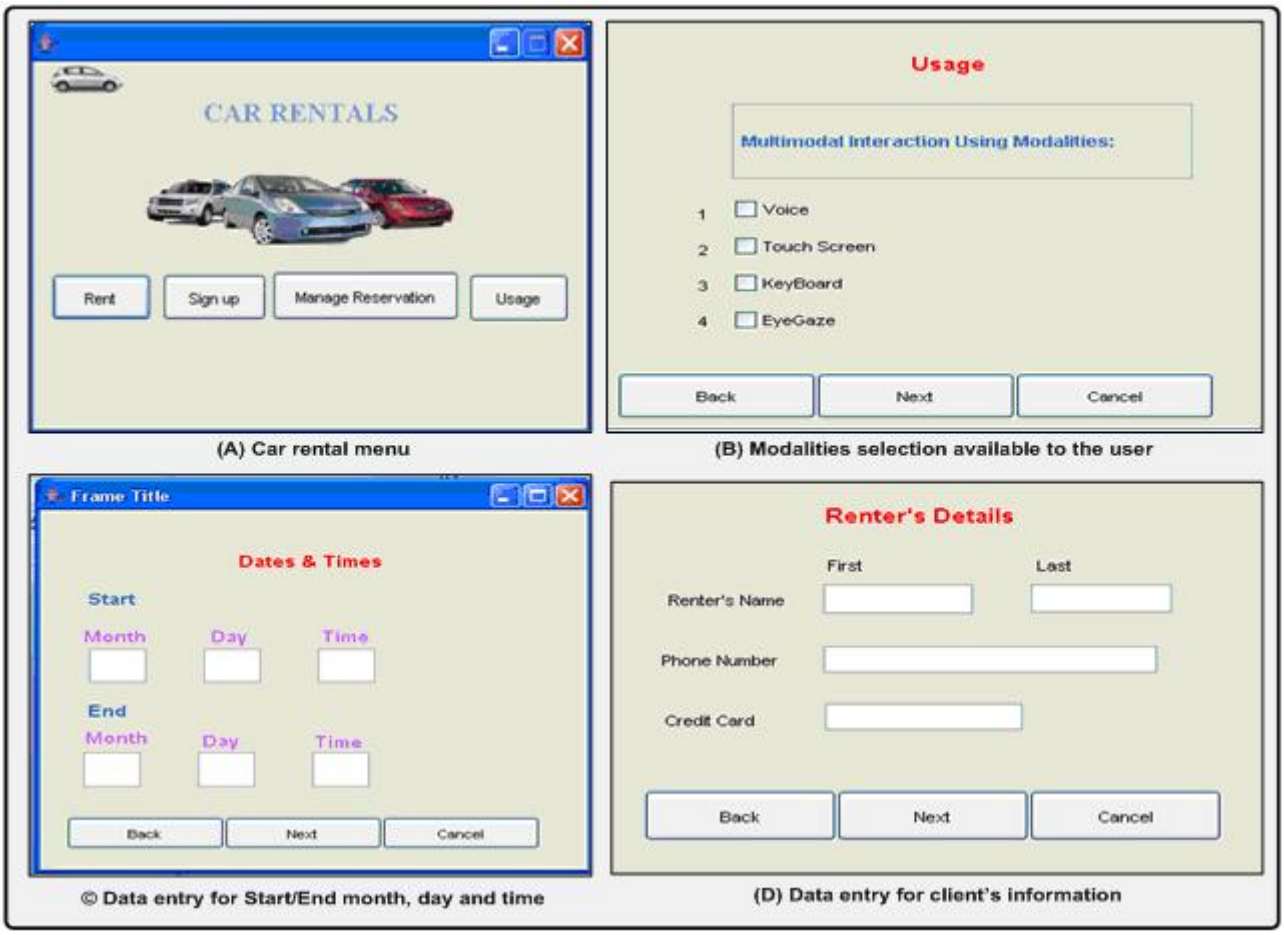

Figure 15. (A) Car rental menu, (B) Available modalities, (C) Data entry for month, day and time, and (D) Data entry for renter's information

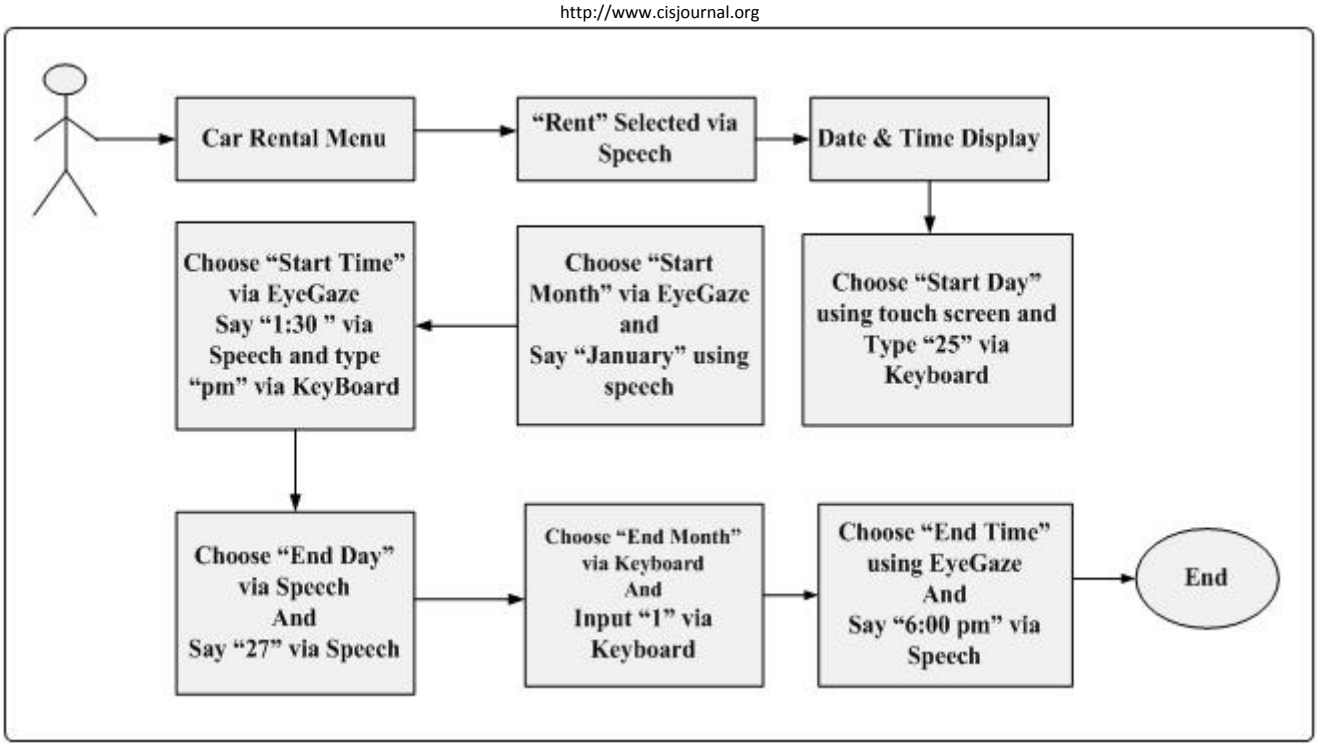

Figure 16. A sample scenario showing multimodal interactions between the user and the machine

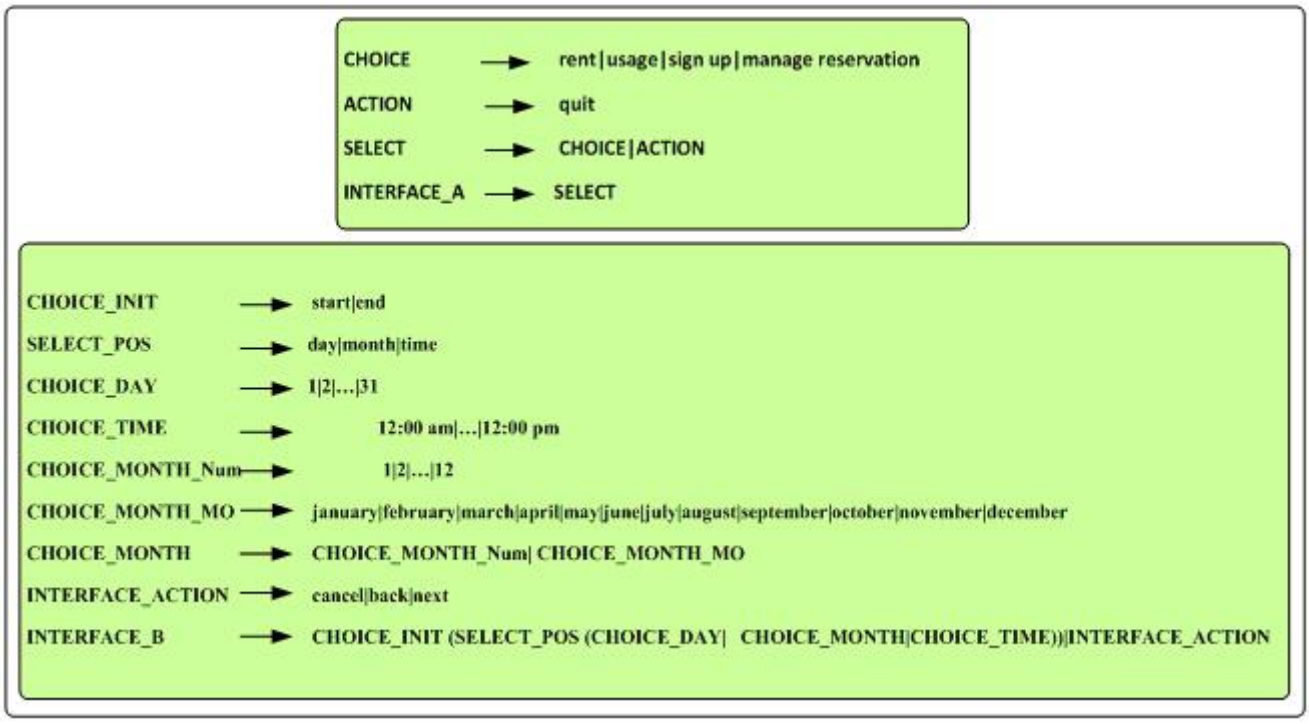

Figure 17. Grammar for the period that a user would rent a car

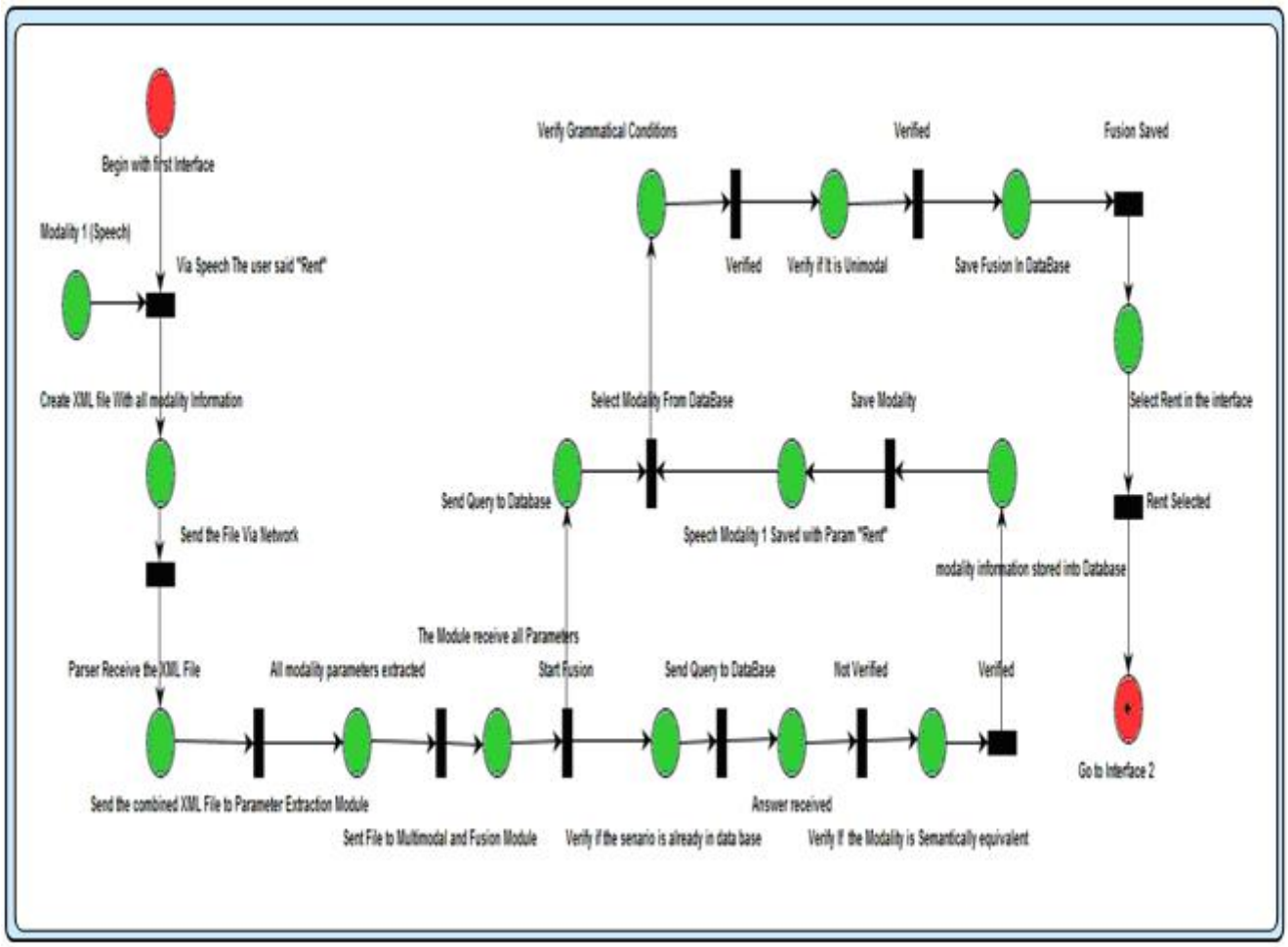

Figure 18. System activities as the user chooses “Rent” via speech

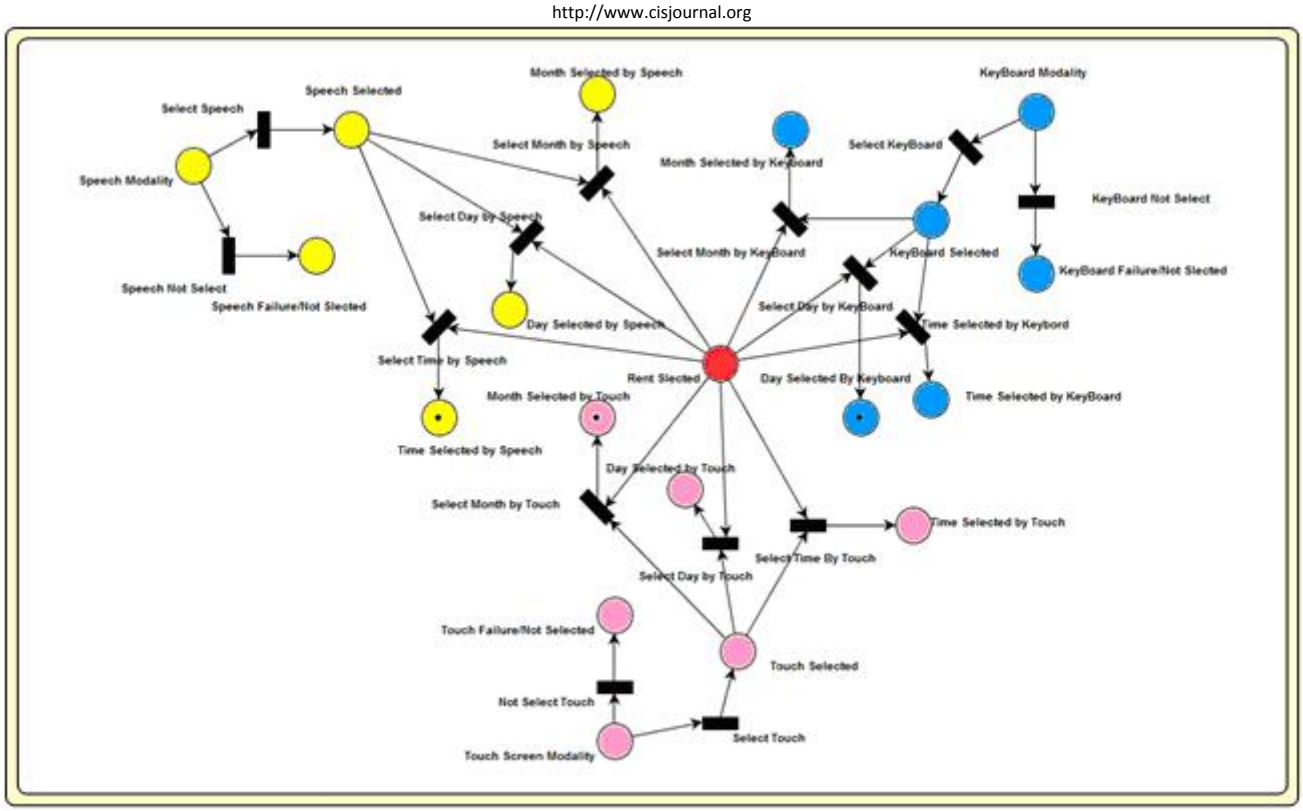

Figure 19. Petri Net showing all possible variations of data entry for month, day and time using speech, keyboard and touch screen when the “Rent” menu is selected.

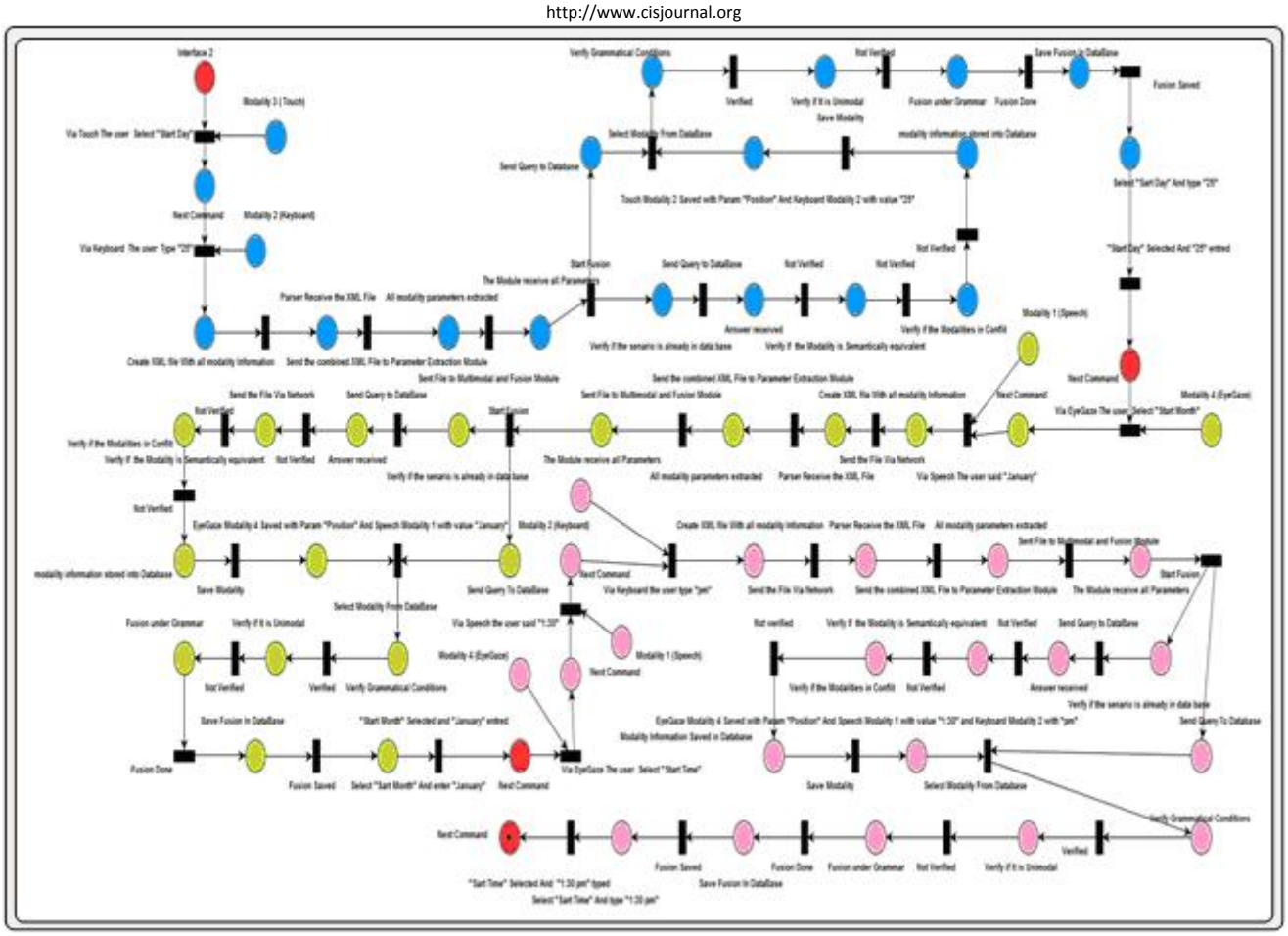

Figure 20. Petri Net showing all operations involving various modalities as the user chooses and enters specimen data for "Start Month", "Start Day" and "Start Time".

5. CONCLUSION AND FUTURE WORKS

Even as we look at the current state of the art involving multimodal fusion, we realize that the multimodalities involved are pre-defined from the very start. This set-up is correct only on the condition that the fusion is implemented in a controlled environment, one in which the environment parameters remain fixed. In a real-time and real-life set-up, however, this setting is incorrect since too many parameters may change while an action (web service) is being undertaken. In this paper, we present a more flexible approach in which the user chooses the modalities that he sees fit to his situation, therefore, the fusion process is not based on the modalities that are already pre-defined from the very beginning.

In this paper, we present our approach on multimodal fusion based on the modalities that the user himself selects. The intended application is to access web services. We showed that an event involving a multimodal action is captured in an XML file clearly identifying the involved modality and its associated parameters. We showed the parsing mechanism as well as the parsing extractor. Then, the fusion of two or more modalities is presented in concept. We then illustrate a concrete example on how our system works in practice via a car rental system. The sample application's specification is presented using the Petri Net tool.

In this work, the user chooses the modalities he sees suitable to his situation. To further advance this work, we intend to consider the user's interaction context (the overall context of the user, of his environment, and of his computing system) in determining which modalities are suitable to the user before the multimodal fusion process is undertaken.

Acknowledgement

We wish to acknowledge the funds provided by the *Natural Sciences and Engineering Council of Canada* (NSERC) which partially support the financial needs in undertaking this research work.

References

Aim, T., Alfredson, J., et al. (2009). *"Simulator-based human-machine interaction design."* International

Journal of Vehicle Systems Modelling and Testing 4(1/2): pp. 1-16.

Apple. (2010). *"Apple Mac OS."* from www.apple.com/macosx/.

Ballinger, K. (2003). *"NET Web Services: Architecture and Implementation"*. Boston, MA, USA, Addison-Wesley.

Bates, B. and Sierra, K. (2003). *" Head First Java: Your Brain on Java - A Learner's Guide"*, O'Reilly Media.

Bolt, R. (1980). *"Put that there: Voice and gesture at graphics interface."* Computer Graphics Journal of the association of computing and machinery 14(3): pp. 262-270.

Bonet, P., Llado, C. M., et al. (2010). *"PIPE2."* from http://pipe2.sourceforge.net/index.html.

Carnielli, W. and Pizzi, C. (2008). *"Modalities and Multimodalities"*. Campinas, Brazil, Springer.

Caschera, M. C., D'Andrea, A., et al. (2009). *"ME: Multimodal Environment Based on Web Services Architecture "*. On the Move to Meaningful Internet Systems: OTM 2009 Workshops, Berlin (Heidelberg), Springer-Verlag.

Debevc, M., Kosec, P., et al. (2009). *"Accessible multimodal Web pages with sign language translations for deaf and hard of hearing users"*. DEXA 2009, 20th International Workshop on Database and Expert Systems Application. Linz, Austria, IEEE**:** pp. 279-283.

Desmet, C., Balthazor, R., et al. (2005). *"<emma>: re-forming composition with XML."* Literary & Linguistic Computing 20(1): pp. 25-46.

Giuliani, M. and Knoll, A. (2008). *"MultiML: A general purpose representation language for multimodal human utterances"*. 10th International Conference on Multimodal Interfaces. Crete, Greece, ACM**:** pp. 165 - 172.

ISO/IEC-15909-2. (2010). *"Petri Nets."* from http://www.petrinets.info/.

Johnston, M. (2009). *"Building multimodal applications with EMMA"*. ICMI 05, International Conference on Multimodal Interfaces. Cambridge, MA, USA, ACM**:** pp.

King, K. N. (2008). *"C Programming: A Modern Approach"*, W.W. Norton & Company.

Kress, G. (2010). *"Multimodality: Exploring Contemporary Methods of Communication"*. London, UK, Taylor & Francis Ltd.

Lai, J., Mitchell, S., et al. (2007). *"Examining modality usage in a conversational multimodal application for mobile e-mail access."* International Journal of Speech Technology 10(1): pp. 17-30.

Lalanne, D., Nigay, L., et al. (2009). *" Fusion Engines for Multimodal Input: A Survey"*. ACM International Conference on Multimodal Interfaces. Beijing, China**:** pp. 153-160.

Li, Y., Liu, Y., et al. (2007). *"An exploratory study of Web services on the Internet"*. IEEE International Conference on Web Services. Salt Lake City, UT, USA**:** pp. 380-387.

Madani, L., Nigay, L., et al. (2005). *"Synchronous testing of multimodal systems: an operational profile-based approach"*. IEEE International Symposium on Software Reliability Engineering. Chicago, IL pp. 334-344.

Malik, D. S. (2010). *"C++ Programming: From Problem Analysis to Program Design"*, Course Technology.

Microsoft. (2010). *"Microsoft Windows "*, from www.microsoft.com/windows.

Mohan, C. K., Dhananjaya, N., et al. (2008). *"Video shot segmentation using late fusion technique"*. 7th International Conference on Machine Learning and Applications. San Diego, CA, USA, IEEE**:** pp. 267-270.

Oviatt, S. (2002). *"Multimodal Interfaces: Handbook of Human-Computer Interaction"*. New Jersey, USA, Lawrence Erbaum.

Oviatt, S., Cohen, P., et al. (2000). *"Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions."* Human-Computer Interaction 15(4): pp. 263-322.

Oviatt, S. L. and Cohen, P. R. (2000). *"Multimodal Interfaces that Process What Comes Naturally."* Communications of the ACM 43(3): pp. 45 - 53.

Pérez, G., Amores, G., et al. (2005). *"Two strategies for multimodal fusion"*. ICMI'05 Workshop on Multimodal Interaction for the Visualisation and Exploration of Scientific Data. Trento, Italy, ACM**:** pp.

Petri-Nets-Steering-Committee. (2010). *"Petri Nets World."* from http://www.informatik.uni-hamburg.de/TGI/PetriNets/.

Pfleger, N. (2004). *"Context Based Multimodal Fusion"*. ICMI 04. Pennsylvania, USA, ACM**:** pp. 265 - 272.

PostgreSQL. (2010). from http://www.postgresql.org/.

Raisamo, R., Hippula, A., et al. (2006). *"Testing usability of multimodal applications with visually impaired children."* IEE, Institute of Electrical and Electronics Engineers Computer Society 13(3): pp. 70-76.

Red_Hat_Enterprise. (2010). *"Linux."* from www.redhat.com.

Ringland, S. P. A. and Scahill, F. J. (2003). *"Multimodality - The future of the wireless user interface."* BT Technology Journal 21(3): pp. 181-191.

Schroeter, J., Ostermann, J., et al. (2000). *"Multimodal Speech Synthesis"*, New York, NY.

Sears, A. and Jacko, J. A. (2007). *"Handbook for Human Computer Interaction"*, CRC Press.

Shin, B.-S., Ahn, H., et al. (2006). *"Wearable multimodal interface for helping visually handicapped persons"*. 16th international conference on artificial reality and telexistence. Hangzhou, China, LNCS vol. 4282**:** pp. 989-988.

Snoek, C. G. M., Worring, M., et al. (2005). *"Early versus late fusion in semantic video analysis"*. 13th annual ACM international conference on Multimedia. Hilton, Singapore, ACM**:** pp.

Steele, R., Khankan, K., et al. (2005). *"Mobile Web Services Discovery and Invocation Through Auto-Generation of Abstract Multimodal Interface"*. ITCC 2005 International conference on Information Technology: Coding and Computing, Las Vegas, NV.

Ventola, E., Charles, C., et al. (2004). *"Perspectives on Multimodality"*. Amsterdam, the Netherlands, John Benjamins Publishing Co.

W3C. (2010). *"EMMA."* from http://www.w3.org/TR/emma.

Wang, D., Zhang, J., et al. (2006). *A Multimodal Fusion Framework for Children's Storytelling Systems*. Book *"LNCS "*. Berlin / Heidelberg, Springer-Verlag. 3942/2006**:** pp. 585-588.

Wang, F., Li, J., et al. (2007). *"A space efficient XML DOM parser."* Data & Knowledge Engineering 60(1): pp. 185-207.

Weaver, J. L. and Mukhar, K. (2004). *" Beginning J2EE 1.4: From Novice to Professional"*, Apress.

Wöllmer, M., Al-Hames, M., et al. (2009). *"A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams."* Neurocomputing 73(1-3): pp. 366-380.

Yuen, P. C., Tang, Y. Y., et al. (2002). *"Multimodal Interface for Human-Machine Communication"*. Singapore, World Scientific Publishing Co., Pte. Ltd.

Zhang, Q., Imamiya, A., et al. (2004). *"A gaze and speech multimodal interface"*, Hachioji, Japan, Institute of Electrical and Electronics Engineers Inc., Piscataway, USA.