Benchmarking Fusion Engines of Multimodal Interactive Systems

Bruno Dumas
University of Fribourg
Bd de Pérolles 90
1700 Fribourg, Switzerland
0041/26.300.92.96
bruno.dumas@unifr.ch

Rolf Ingold
University of Fribourg
Bd de Pérolles 90
1700 Fribourg, Switzerland
0041/26.300.84.66
rolf.ingold@unifr.ch

Denis Lalanne
University of Fribourg
Bd de Pérolles 90
1700 Fribourg, Switzerland
0041/26.300.84.72
denis.lalanne@unifr.ch

ABSTRACT
This article proposes an evaluation framework to benchmark the performance of multimodal fusion engines. The paper first introduces different concepts and techniques associated with multimodal fusion engines and further surveys recent implementations. It then discusses the importance of evaluation as a mean to assess fusion engines, not only from the user perspective, but also at a performance level. The article further proposes a benchmark and a formalism to build testbeds for assessing multimodal fusion engines. In its last section, our current fusion engine and the associated system HephaisTK are evaluated thanks to the evaluation framework proposed in this article. The article concludes with a discussion on the proposed quantitative evaluation, suggestions to build useful testbeds, and proposes some future improvements.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation]: User Interfaces – Input devices and strategies, Interaction styles, Prototyping.

General Terms
Design, Human Factors.

Keywords
Multimodal toolkit, multimodal interfaces, multimodal fusion, fusion engines evaluation.

1. INTRODUCTION
Since the first works of Richard Bolt [2], research in multimodal interaction has examined how to combine data coming from different modalities. This process, called fusion of input modalities, is key to the success of multimodal interaction. As users interact with a multimodal system, they have expectations about the way the system will react to their orders, and their experience will degrade if the system replies too slowly or too unreliably.

ICMI-MLMI'09, November 2–4, 2009, Cambridge, MA, USA.

Evaluation of multimodal systems has mainly focused on user interaction and user experience evaluation. These evaluations offer extremely important insights about a given user interface, but, in front of a complex interaction system like a multimodal interface, analysis of what to correct and how to correct it can become problematic. In this paper, we propose a complementary approach to user evaluation techniques, specifically fusion engine efficiency assessing techniques. These have the benefit to pinpoint specific shortcomings of a given fusion engine, helping correct them and thus improving the user experience. Furthermore, as frameworks for creation of multimodal interfaces (such as OpenInterface [17] or Callas [1]) become increasingly available, with the frequent goal of serving as a platform for fusion algorithms testing, common fusion engines evaluation procedures and metrics should develop accordingly. Thus, we tested our benchmark proposal in a framework for creation of multimodal interfaces, named HephaisTK. The results of this benchmark test are then discussed.

The paper briefly introduces the different concepts and techniques associated with multimodal fusion engines, and presents various recent implementations. It further discusses the importance of evaluation as a mean to assess fusion engines, not only from the user perspective, but also at a performance level. Section 4 proposes to benchmark multimodal fusion engines and introduces a toy testbed. Finally, section 5 briefly presents HephaisTK and the evaluation of its fusion engine following the testbed presented in the previous section.

2. MULTIMODAL FUSION ENGINES

2.1 Multimodal Systems Architecture
We describe in this section multimodal interaction from the machine side, and the major software components that a multimodal system should contain. The generic components for handling of multimodal integration are: a fusion engine, a fission module, a dialog manager and a context manager, which all together form what is called the "integration committee". Figure 1 illustrates the processing flow between these components, the input and output modalities, as well as the potential client applications (details on this architecture can be found in [9]). As illustrated in the figure, input modalities are first perceived though various recognizers, which output their results to the fusion engine, in charge of giving a common interpretation of the inputs. The various levels at which recognizers' results can be fused are described in the next section, together with the various fusion mechanisms. When the fusion engine comes to an interpretation, it communicates it to the dialog manager, in charge of identifying the

dialog state, the transition to perform, the action to communicate to a given application, and/or the message to return through the fission component. The fission engine is finally in charge of returning a message to the user through the most adequate modality or combination of modalities, depending on the user profile and context of use. For this reason, the context manager, in charge of tracking the location, context and user profile, closely communicates any changes in the environment to the three other components, so that they can adapt their interpretations.

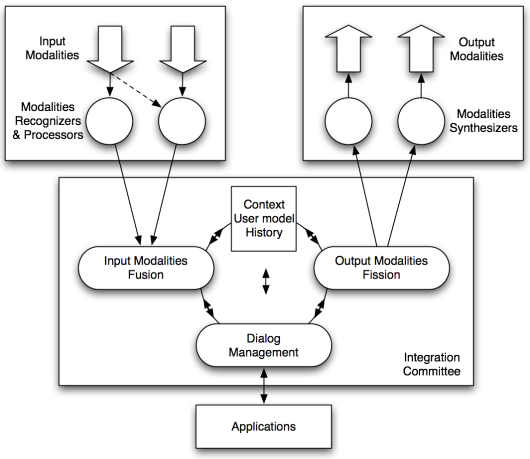


Figure 1. The architecture of a multimodal system, with its major software components.

2.2 Fusion of Input Modalities

Fusion of input modalities is one of the features that distinguish multimodal interfaces from unimodal interfaces. The goal of fusion is to extract meaning from a set of input modalities and pass it to a human-machine dialog manager. Fusion of different modalities is a delicate task, which can be executed at three levels: at data level, at feature level and at decision level. Three different types of architectures can in turn manage decision-level fusion: frames-based architectures, unification-based architectures or hybrid symbolic/statistical fusion architectures.

Sharma et al. [22] consider three levels for fusion of incoming data: data-level, feature-level and decision-level fusion. Each fusion scheme functions at a different level of analysis of the same modality channel. As a classic illustration, consider the speech channel: data from this channel can be processed at the audio signal (data) level, at the phoneme (feature) level, or at the semantic (decision) level. We will only detail the third level, which is generally favored when dealing with fusion of multimodal data in interactive systems.

Typical architectures for decision-level fusion are frame-based fusion, unification-based fusion and hybrid symbolic/statistical fusion (see [9] for more details).

- Frame-based fusion uses data structures called frames or features for meaning representation of data coming from various sources or modalities. These structures represent objects as attribute-value pairs.
- Unification-based fusion is based on recursively merging attribute-value structures to obtain a logical whole meaning representation.
- Symbolic/statistical fusion is an evolution of standard symbolic unification-based approaches, which adds statistical processing techniques to the fusion techniques described above. These kinds of "hybrid" fusion techniques have been demonstrated to achieve robust and reliable results.

The dialog management system and synchronization mechanism should consider multiple potential causes of lag (recognizers, system architecture) and for this reason, multi-agent architectures (or similar architectures such as components-based systems) are advantageous for distributing processing and for coordinating many system components (e.g., speech recognition, pen recognition, natural language processing, graphic display, TTS output, application database).

Bui [5] considers four different approaches to dialog management: finite-state and frame-based approaches, information state-based and probabilistic approaches, plan-based approaches, and collaborative agents-based approaches.

2.3 Current implementations

Table 1 summarizes the major architecture traits of recent implementations of multimodal systems, as well as their fusion mechanisms. Krahnstoever et al. [16] proposed a multimodal framework with a fusion engine using a unification-based method. Cohen et al. [6] worked on Quickset, a speech/pen multimodal interface, based on Open Agent Architecture, which served as a test bed for unification-based and hybrid fusion methods. Bourguet [4] endeavored in the creation of a multimodal toolkit in which multimodal scenarios could be modelled using finite state machines. In their multimodal system, Flippo et al. [12] used parallel application-independent fusion technique, based on an agent architecture, and fusion using frames. Bouchet et al. [3] proposed a component-based approach to fusion called ICARE thoroughly based on the CARE [7] design space. These components cover elementary tasks, modality-dependent tasks or generic tasks like fusion. Finally, communication between components is based on events. It is finally worth noting two comprehensive open-source frameworks called OpenInterface [17] and CALLAS [1]. These two frameworks share a similar conceptual architecture, but with different goals: OpenInterface targets pure or combined modalities, where CALLAS has more interest in situation awareness and video processing components.

3. THE EVALUATION BLACK HOLE

In a multimodal system, errors can originate from a number of different sources. Most prominent sources of errors are the modality recognizers, the fusion engine and, in some way, the user itself. In order to correct errors, one has to detect them, for example by means of user evaluations. In multimodal systems, problems arise when trying to detect the real cause of errors such as "system does not answer to a user's multimodal query". For example, in a speech/gesture system, multiple different reasons can lead to this single error: it can originate from a malformed query, a speech recognition problem, delay in the system leading to the multimodal command being not fused, the fusion engine itself not recognizing the multimodal command, problem in the feedback… or even a mix of different causes (see for example Holzapfel et al. [14] for different causes of multimodal integration errors). Likewise, the interaction possibilities being richer (and more complex) in a multimodal system, a standard user evaluation is not guaranteed to detect most

interaction problems: studies [19] have showed that the use of multimodality can heavily differ from one user to another.

	ICARE [3]	OpenInterface [17]	IMBuilder/MEngine [4]	Flippo et al. [12]	Krahnstoever [16]	Quickset [6]	Callas [1]	HephaisTK [10]
Finite state machine			x					
Components	x	x					x	
Software agents				x		x		x
Fusion by frames					x			x
Symbolic-statistical fusion						x		
CARE properties	x	x						x

Table 1. Architecture traits of current multimodal systems.

A good starting point to multimodal systems evaluation is some sort of "divide-and-conquer" method: the idea is to achieve the evaluation of a multimodal interface in a step-by-step manner, and base later evaluations on the results of the former ones. For example, first evaluate recognizers individually, then evaluate the fusion of example recognizers data, then evaluate the fission of output, still with example fusion engine data; and when the different modules composing the multimodal system have been tested against a sample of the type of data they have to manage, test the whole system in "real-life" conditions, with actual users injecting actual data in the system. In the view of such a step-by-step evaluation, the need for standard evaluation procedures dedicated at testing individual multimodal systems components appears. NIST Multimodal Information Group leader John Garofolo recently reckoned that, while evaluation of individual modalities is progressing at a good pace, the evaluation of hybrid technologies presents new challenges, in particular at the fusion engine level [13].

On a more technical note, now that toolkits for creation of multimodal interfaces are in full development in the scope of various projects, the interest of being able to compare those tools arises. Furthermore, some of these multimodal interfaces creation toolkits, such as OpenInterface or HephaisTK, have been created with the ability to plug and test different fusion algorithms. Thus, as these toolkits are made available to the multimodal interaction community, common metrics and tests are desirable in order to be able to compare different algorithms.

On the subject of algorithms for fusion of multimodal data, it is to be noted that research on advanced algorithms has remained sparse since the works of Oviatt et al., even if the need for such algorithms was expressed numerous times in the literature [13] [20]. We feel that, by providing a common measurement of the efficiency and effectiveness of fusion algorithms, and by testing these algorithms against a set of problematic – although common – cases of multimodal inputs fusion, strengths and weaknesses of these different algorithms, as well as precise use cases on which current fusion engines lack effectiveness, should appear.

Finally, a testbed on multimodal fusion processes should pay attention to a specificity of multimodal interfaces: user and context consideration. In detail, fusion of input data is frequently achieved according to context, be it the current state of the application (in particular for strongly modal applications), contextual data enriching the input modalities interpretation (e.g. drawing a circle would not have the same meaning when surrounding a building on a map, pulling a shape on a drawing application or dialing a number on an old-style phone dial), and context of use (e.g. using a mobile application at home, at work, in the street or in a car). On the subject of user consideration, it has been shown that, if integration patterns differ largely from one user to another, a given user tends to keep the same integration patterns and remain persistent throughout a same session. Thus, a testbed should take into account this specificity of multimodal interfaces, give information about the context of the application, the context of use, and supply a set of different sequences of consistent uses for a given use case, so that fusion engines would be exhorted to adapt to users' integration patterns, for example by using machine learning techniques.

4. BENCHMARKING FUSION ENGINES

This section proposes a benchmark to measure the performance of multimodal fusion engines in a replicable and controlled way. For this purpose, we discuss in the following sections about the possibility to set up a testbed, a software infrastructure, and a metric of performance, in order to compare precisely the quality and efficiency of multimodal fusion engines.

4.1 A testbed for fusion engines

To allow replicable testing in the case of multimodal fusion engines, the challenge is to create a set of problematic use-cases supporting their quantitative evaluations. In order to bypass the problems related to recognition errors introduced by each modality recognizers (speech, gesture, emotions, etc.), which intervene before the multimodal fusion itself, we propose to simulate the recognizers outputs and feed these outputs directly to the fusion engines. The goal of this testbed is to focus on fusion algorithms and rules; furthermore, by simulating the recognizers output, we are also able to simulate incorrect outputs, and assess how fusion engines react to recognizers failures.

As illustrated on figure 2, for a given testbed, i.e. a temporal and multimodal events' stream resulting from simulated multimodal recognizers, a fusion engine will generate a series of interpretation in time.

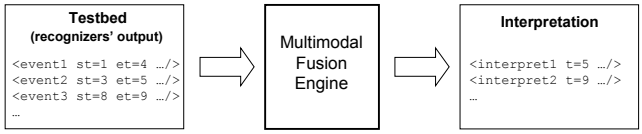


Figure 2. A multimodal fusion engine processes a series of multimodal events to generate a series of interpretations.

The resulting interpretation by the multimodal fusion engine can be then compared with a ground-truth associated with the testbed, in order to measure its performance, as illustrated on figure 3. As discussed in the following section, various factors will help computing a performance metric (response time, confidence and efficiency).

Figure 3. A ground-truth allows to rate performances of the fusion engine, according to the interpretations it gave.

As mentioned above, the testbed we propose will simulate the output of various recognizers. For the sake of standardization, the representation of this stream of events will use the Extensible Multimodal Annotation markup language (EMMA) [11]. EMMA is an XML markup language used for containing and annotating the interpretation of user input by multimodal recognizers. Examples of interpretation of user input are a transcription into words of a raw signal, for instance derived from speech, pen or keystroke input, a set of attribute/value pairs describing their meaning, or a set of attribute/value pairs describing a gesture. The interpretation of the user's input is expected to be generated by signal interpretation processes, such as speech, gesture and ink recognition. By using EMMA for the representation of the testbed and ground-truth data, we slightly divert the language from its original role envisioned by the W3C Multimodal Interaction Working Group. However, we feel that the EMMA language perfectly fits this new role of simulation and ground-truth data representation.

In order to be relevant, the testbed should allow testing most of the major difficulties related to multimodal fusion. This is opening an important research question: what are the issues related with multimodal fusion engines in interactive systems? What difficult use cases or combination of events generate interpretation errors?

In search for answers, two formal representations trying to model multimodal human machine interaction could provide us a relevant framework for the testbed:

- The CASE model [17], focusing on modality combination possibilities at the fusion engine level;
- The CARE model [7], giving attention to modality combination possibilities at the user level.

The CASE model introduces four properties: Concurrent – Alternate – Synergistic – Exclusive. Each of those four properties describes a different way to combine modalities at the integration engine level, depending on two factors: combined or independent fusion of modalities, and sequential or synergistic use of modalities on the other hand.

The CARE model is more focused on the user-machine interaction level. This model also introduces four properties, which are Complementarity – Assignment – Redundancy – Equivalence. Complementarity is used by the user when multiple complementary modalities are necessary to grasp the desired meaning (e.g. "put that there" [1] would need both pointing gestures and voice in order to be resolved). Assignment indicates that only one modality can lead to the desired meaning (e.g. the steering wheel of a car is the only way to direct the car). Redundancy implies multiple modalities which, even if used simultaneously, can be used individually to lead to the desired meaning (e.g. user utters a "play" speech command and pushes a button labeled "play", but only one "play" command would be taken into account). Finally, Equivalence entails multiple modalities that can all lead to the desired meaning, but only one would be used at a time (e.g. speech or keyboard can be used to write a text).

Since the goal of the testbed can be reworded as the task of measuring how well fusion engines are able to interpret the intention of the user and its usage of multimodality, the CARE model seems the most suited to structure the testbed; The CARE properties model the various usage of multimodality that a user can intentionally achieve to control an interactive system.

The time dimension is highly important when dealing with multimodal fusion. For a given multimodal command, the way modalities are synchronized will strongly impact the interpretation. For example, consider a multimodal music player, which would allow users to control the different commands with a number of modalities, in a redundant or complementary way, depending from the command. Our main example for this paper will be a vocal command ("play next track") combined with one pointing gesture, to play a musical track. The interpretation of this command can greatly vary depending on the time synchronicity and on the sequence in which commands have been produced. This variability is the most interesting aspect of this example, as it cannot be found in more classical examples such as Bolt's "put that there", which, although complementary, is resolved in a univocal manner.

For instance, in the following application, in which voice and gestures are used simultaneously to control this music player, depending on the order in which modalities are presented, the interpretation varies:

- <pointing> "Play next track": will result in playing the track following the one selected with a gesture, mixing assignment with complementarity;
- "Play" <pointing> "next track": will result in first playing the manually selected track and then passing to the following at the time "next" is pronounced, thus mixing complementarity with assignment;
- In parallel <pointing> && "Play next track": in this case the user the system should interpret that the two modalities are used in a redundant way;

While the cases above seem non ambiguous, other cases can be imagined, in which it becomes unclear what the user wanted to say and what should a perfect multimodal fusion engine interpret.

- "Play next track" <pointing>: In this case, the system can either interpret the commands as being redundant or as being complementary and, depending on its choice, will play a different track.

Potential interpretation problems in multimodal fusion engine may also occur for technical reasons impacting on the precision of time synchronicity. For this reason, the fusion engine (and the related testbed) should consider multiple potential causes of lag:

- Delay due to technology (ex.: speech recognition);
- Delay due to multimodal system architecture;
- User differences in their habitual multimodal integration pattern [19][21].

To illustrate our point, we propose to encode with EMMA the simplistic example presented above as an illustrative testbed. The following EMMA is composed of two sequences, the first corresponding to the testbed description, and the second sequence

corresponding to the ground-truth related to the testbed. The EMMA code for the testbed description would read like this:

```
<emma:emma version="1.0">
  <emma:one-of id="main">
    <emma:sequence id="testbed">
      <!--   <pointing>+"Play next track" case testbed   -->
      <emma:interpretation id="gesture1" emma:medium="tactile"
         emma:mode="video">
          <content>track_pointed</content>
      </emma:interpretation>
      <emma:interpretation id="speech1" emma:medium="acoustic"
        emma:mode="voice" emma:time-ref-uri="#gesture1"
        emma:offset-to-start="300">
          <content>play</content>
      </emma:interpretation>
      <emma:interpretation id="speech2" emma:medium="acoustic"
        emma:mode="voice" emma:time-ref-uri="#gesture1"
        emma:offset-to-start="400">
          <content>next track</content>
      </emma:interpretation>
      <!--   other testbed elements would come here      -->
    </emma:sequence>
```

Below is the corresponding ground-truth interpretation (as presented in figure 3). "emma:derived-from" elements allow to connect the ground-truth interpretation to the different testbed input elements that should lead to it. The message corresponds to the value that the fusion engine is supposed to produce. In the whole EMMA testbed file, we believe this is the only value dependant from the fusion engine, i.e. which would have to be adapted from one engine to another. Even when taking into account this "<message>" (which is considered as application-specific instance data), the whole EMMA testbed file is fully compliant to the W3C EMMA 1.0 recommendation [11].

```
   <emma:sequence id="groundtruth">
     <!--   <pointing>+"Play next track" case groundtruth   -->
     <emma:interpretation id="message1">
       <emma:derived-from resource="#gesture1" composite="true"/>
       <emma:derived-from resource="#speech1" composite="true"/>
       <emma:derived-from resource="#speech2" composite="true"/>
       <message>play_next_of_pointed_track</message>
     </emma:interpretation>
     <!--   other groundtruth elements would come here      -->
   </emma:sequence>
 </emma:one-of>
</emma:emma>
```

Of course, the testbed above is a toy example and a more complete and serious testbed should be discussed and agreed on by the overall community. In particular, if the CARE properties model brings to light a number of problematic fusion cases as shown above, it certainly does not deliver an exhaustive catalog of every difficult or tricky cases a fusion engine might encounter; such a catalog can only be achieved by collecting the real-world experience of many multimodal interaction practitioners.

4.2 Metrics and Software Requirements

We propose the following quantitative and qualitative metrics to measure the quality of a given multimodal engine according to a series of multimodal recognized events. For each multimodal event we plan to measure in a quantitative way:

- *Response time*: time that the fusion engine takes to return an interpretation after receiving multimodal inputs.
- *Confidence*: Level of confidence of the machine response, based for example on confidence scores indicated in the EMMA testbed interpretation elements.
- *Efficiency*: success or failure of the fusion engine to interpret correctly the testbed entries. Efficiency is measured by confronting the machine interpretation against the ground-truth data.

An efficient fusion engine should answer reliably and quickly to the requests of the users; furthermore, an efficient fusion engine should be extensible and easy to use. Although such characteristics are much harder to measure dynamically, they are nonetheless important. Thus, other features of fusion engines to be measured in a more qualitative way would be the following ones:

- *Adaptability*: whether a fusion engine is able to adapt itself to context and user.
- *Extensibility*: how much a fusion engine can be extended to new or different input sources.

On a related dimension, the testbed itself should be characterized, thus helping developers outline shortcomings of their fusion engines. Such characteristics would be:

- *Expressive power, or ability to be used by non-programmers*: the developer-, or user-friendliness of the mechanisms used to configure the fusion engine.
- *Level of complexity*: level of complexity (low/medium/high) of the test, following a number of criteria. For example, a test requiring complementary fusion based on incomplete or noisy data would be more complex than straightforward equivalence of two input modalities.
- *Problem type*: each test could be characterized by a number of keywords, describing particular goals or features to be tested, such as temporal constraints, presence of noisy data, etc.

Logging is a common mechanism for most development teams in order to debug software, to trace its usage, or to record and analyze users' behaviors. In multimodal systems, the time constraint is highly important and all the modalities should be properly time-stamped and synchronized. Time-sensitive architectures need to establish temporal thresholds for time-stamping start and end of each input signal piece, so that two commands sequences can be identified. Indeed, when two commands are performed in parallel, it is important to know in which order the commands have been entered because the interpretation will vary accordingly as seen in the previous section. Therefore, logging mechanisms are required for multimodal system benchmarking. In particular, as too important delays between the user input and the resulting output can ruin the user experience, ways to log data and timestamps passing through the fusion engine are recommended.

Any multimodal system able to log input events and fused multimodal events, as well as their timestamps, should be able to implement the proposed testbed. Input events can be generated either directly from the EMMA file, if the multimodal system already uses EMMA as data transfer format, or by means of a tailored component taking the EMMA file as input. As for the results analysis, it can be achieved either on the fly, as presented in the next section about our tests in HephaisTK, or after running tests, by analyzing log files.

5. ASSESSING FUSION IN HEPHAISTK

The testbed examples described in section 4 have been applied to HephaisTK, a toolkit for rapid creation of multimodal interfaces. Our goal was to assess

- The feasibility of the described testbed, in particular the use of the EMMA markup language to describe test inputs and ground-truth outputs;
- How a real-world tool using a fusion mechanism derived from meaning frames would react to the simple "play next track" example described in section 4.

This section first describes the HephaisTK platform as well as the markup language used to script it, named SMUIML. The section then goes on by describing how the results of a performance evaluation, which followed the testbed protocol described before.

5.1 HephaisTK Architecture

The main goal of HephaisTK toolkit is to allow developers to quickly develop and test multimodal interfaces. Its modular architecture allows developers to easily configure it according to their needs, and to plug new human-computer communications means recognizers. HephaisTK is designed to control various input recognizers, and more importantly user-machine dialog and fusion of modalities.

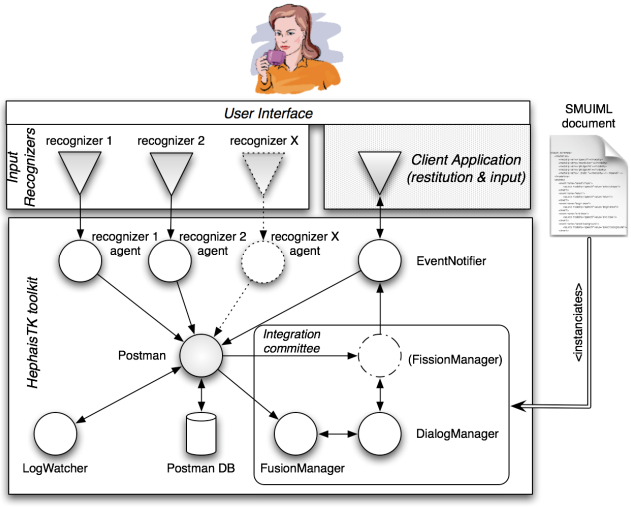


Figure 4. Architecture of HephaisTK.

A developer wishing to use HephaisTK to develop a multimodal application will have to provide two components: his application and a SMUIML script (Synchronized Multimodal User Interaction Markup Language). The developer's application needs to import one class of HephaisTK. This class allows communication with the toolkit via Java listeners. The toolkit does not manage the actual content restitution, but sends messages or callbacks to the application describing the content to be restituted to the user. The SMUIML document is used by the toolkit for a number of tasks: first, the definition of the messages that will transit from the toolkit to the developer's application; second, the events coming from the input recognizers that will have to be taken into account by the toolkit; last, description of the overall dialog management.

In order to account for the objective of modularity, the toolkit is built on a software agents framework, namely JADE. The architecture is shown in Figure 4. For each input recognizer, an agent is responsible of reception, annotation and propagation of data transmitted by the recognizer. For instance, the agent responsible of a speech recognizer would propagate not only the speech meaning extracted, but also metadata such as a confidence score. Messages are then sent to the postman agent. This postman agent is in fact a central blackboard collecting data from the recognizers and storing them in a local database. Hence, all data coming from the different sources are standardized in a central place, where other interested agents can dig them at will. Another advantage of central blackboard architecture is to have one central authority that manages timestamps. The problem of synchronizing different timestamp sources is hence avoided, at the cost of a potential greater shift between the timestamp of the actual event and the recorded one. It is to be noted that this central agent does not act like a facilitator: only agents dealing with recognizers-wise data communicate with him. The agents within the integration committee communicate directly. Moreover, the postman agent also offers a mechanism of subscription to other agents: any agent can subscribe to events it is interested in.

Communication with the client application is achieved through a set of messages. Those messages are predefined in the SMUIML script provided by the client application developer. The SMUIML document contains information about the dialog states (DialogManager), the events leading from one state to another (FusionManager) and the information communicated to the client application, given the current dialog state and context (FissionManager). The SMUIML markup language expresses in an easy-to-read and expressive way the modalities used, the recognizers attached to a given modality, the user-machine dialog, and the various triggers and actions associated to this dialog. More details about the SMUIML language and its structure can be found in [8].

5.2 Fusion in HephaisTK

As previously stated, the modular software agents-based architecture allows the toolkit to potentially offer a number of different fusion schemes, from rule-based to statistical to hybrid-based fusion schemes. At present, HephaisTK offers a rule-based approach, conceptually derived from artificial intelligence meaning frames.

The multimodal integration within the toolkit operates in an event-driven way: every time a new event is signaled to the integration committee (e.g. incoming input), it is matched against the possible frames of knowledge of the current context. Following the SMUIML script provided by the client application developer, the dialog manager indicates to the fusion manager in which state the application finds itself, and the fusion manager knows against which set of frames it will have to confront the incoming data. A typical frame of knowledge specifies a number of triggers needed to activate itself, as well as one or more actions to be taken when it activates. Moreover, frames activate following rules modeled from CARE properties [7], allowing temporal constraints to be specified.

SMUIML enables to specify the synchronicity of events in the fusion engine, i.e. how the incoming multimodal triggers should appear in time. Parallel and sequential triggers are distinguished, as well as coupled (and) and exclusive (or) triggers. Based on these four properties, four elements to describe the different behaviors have been designed: <par_and> is to be used when multiple triggers are to be fused together, as they all are necessary for the meaning extraction process; the order in which they appear does not have any importance, as long as they all appear in a defined time window. <seq_and> functions in a similar way than <par_and>, but with one major distinction: all triggers have to appear in the defined time window and in a defined order for the related actions to be fired. <par_or> describes redundant multimodal triggers having similar meanings. Each one is sufficient for the correct meaning to be extracted, but they all can be expressed at the same time by the user, increasing as such the robustness and recognition rate (for example, a user issuing a "play" vocal command and simultaneously pushing a play button). Finally, the <seq_or> element is to be used when multiple triggers can lead to the same result, but only one of them is to be provided. Those four integration describer elements can also be combined in order to express all kinds of multimodal interactions. In fact, three of those four elements correspond to three of the four

CARE properties of multimodal interactive systems as presented in the previous section, the only exception being <seq_and>.

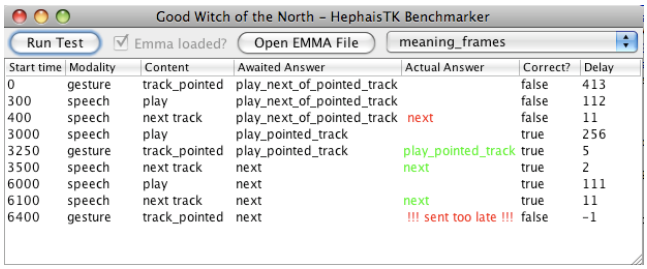
5.3 Performance evaluation

We used the testbed presented in section 4 to test two strategies of fusion with the fusion engine provided with HephaisTK, i.e. with and without temporal ordering constraints. HephaisTK was configured for the testbed using SMUIML. We described in SMUIML the different examples explained in section 4, as well as their corresponding ground truth. Below is the SMUIML related to the first case of the testbed.

```
<?xml version="1.0" encoding="UTF-8"?>
<smuiml>
  <integration_description client="musicplayer">
    <recognizers>
      <recognizer name="fakespeech" modality="speech">
        <parameter name="emma_file" value="PlayTrackEmma.xml"/>
      </recognizer>
      <recognizer name="fakegesture" modality="gesture">
        <parameter name="emma_file" value="PlayTrackEmma.xml"/>
      </recognizer>
    </recognizers>

    <triggers>
      <trigger name="play_trigger">
        <source modality="speech" value="play | play track"/>
      </trigger>
      <trigger name="nexttrack_trigger">
        <source modality="speech" value="next | next track"/>
      </trigger>
      <trigger name="track_pointed_event">
        <source modality="gesture" value="track_pointed"/>
      </trigger>
    </triggers>

    <actions>
      <action name="play_next_of_point_action">
        <target name="musicplayer"
          message="play_next_of_pointed_track"/>
      </action>
    </actions>

    <dialog>
      <context name="start">
        <transition leadtime="1000">
          <seq_and>
            <trigger name="track_pointed_event"/>
            <trigger name="play_trigger"/>
            <trigger name="nexttrack_trigger"/>
          </seq_and>
          <result action="play_next_of_point_action"/>
        </transition>
      </context>
    </dialog>
  </integration_description>
</smuiml>
```

The three example cases "play next track" with the pointing gesture event happening respectively before, in the middle of and after the speech event were modelled. Modelling in SMUIML was achieved by decomposing the speech acts in two, although a speech act for each word would have been possible as well.

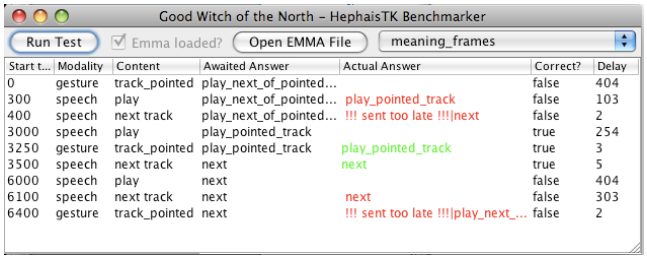
Results of feeding these three examples to the HephaisTK toolkit are shown in Figure 5. Start and delay times are expressed in milliseconds. The trigger events are listed in the order they were fed to the toolkit, in three groups of three events, with only the order in which they were sent changing. For each group of three events, the ground truth ("awaited answer") is indicated. The actual answer received from the HephaisTK fusion engine is then reported, and coloured in green if it corresponded to the ground truth answer, in red otherwise. As one can see, the first example case of pointing a track, and asking to play the next track of it, led to a false answer from the fusion engine, which assumed that the user was simply asking to play the next track of the current one. The second case was correctly understood, as was the third case, although in this last case the fusion algorithm dropped the gesture event (hence the -1 ms delay time). Lead times are computed as the time between the input event dispatch and the fused message reception by the client application. Consequently, in the first case, as the three input events were sent in a 400 ms time window, the delay between sending the first input event ("play") and the reception of the fused event is slightly above 400 ms. The actual fusion process took between 3 and 8 ms in all cases, between emission of the last needed input event and fused message reception by the client application.



Start time	Modality	Content	Awaited Answer	Actual Answer	Correct?	Delay
0	gesture	track_pointed	play_next_of_pointed_track		false	413
300	speech	play	play_next_of_pointed_track		false	112
400	speech	next track	play_next_of_pointed_track	next	false	11
3000	speech	play	play_pointed_track		true	256
3250	gesture	track_pointed	play_pointed_track	play_pointed_track	true	5
3500	speech	next track	next	next	true	2
6000	speech	play	next		true	111
6100	speech	next track	next	next	true	11
6400	gesture	track_pointed	next	!!! sent too late !!!	false	-1

Figure 5. "Play next track" test examples results in HephaisTK toolkit.

We ran the exact same three tests, but removed the sequential constraint from the fusion engine, thus firing a fusion event as soon as meaning frames were complete. The results of this test are shown in Figure 6. As could be expected, the results are worse in respect to the ground truth. Interestingly however, in the first case, "play pointed track" followed by a "next" command correspond to the awaited answer, although decomposed in two different steps. The delay times of the actual fusion stayed between 3 and 8 ms.



Start t...	Modality	Content	Awaited Answer	Actual Answer	Correct?	Delay
0	gesture	track_pointed	play_next_of_pointed...		false	404
300	speech	play	play_next_of_pointed...	play_pointed_track	false	103
400	speech	next track	play_next_of_pointed...	!!! sent too late !!!next	false	2
3000	speech	play	play_pointed_track		true	254
3250	gesture	track_pointed	play_pointed_track	play_pointed_track	true	3
3500	speech	next track	next	next	true	5
6000	speech	play	next		false	404
6100	speech	next track	next	next	false	303
6400	gesture	track_pointed	next	!!! sent too late !!!play_next_...	false	2

Figure 6. "Play next track" test examples results in HephaisTK toolkit, without sequential constraint.

6. CONCLUSION

This article proposes an evaluation framework to benchmark fusion engines of multimodal interactive systems. It first introduces the major concepts associated with multimodal fusion, the current implementations and systems, and further discusses the evaluation aspects. The testbed structure proposed in the article uses the standard EMMA to simulate interesting sequences of multimodal recognizers' output. The proposed testbed has been assessed through the evaluation of temporal ordering aspects of the fusion engine of HephaisTK. As such, the proposed testbed structure seems reliable to compare various implementations of fusion engines. However, the content of the testbed proposed is preliminary and deeper works should be performed to build a general testbed covering most of the challenging issues related to fusion engines. In particular, practitioners should reflect on the most critical issues related with multimodal fusion engines in interactive systems that should be solved, and on difficult cases or combination of events that generate interpretation errors. Issues related to fusion engines' adaptation to context (environment and also applications), as well as users' favorite usage patterns or repetitive errors, should be also considered. This paper does not intend to solve the problem of quantitative evaluation

of multimodal interfaces, nor to replace useful user evaluations, but rather to open a research that we believe crucial for the future developments of fusion engines.

7. REFERENCES

[1] Bertoncini, M. Cavazza, M. Emotional Multimodal Interfaces for Digital Media: The CALLAS Challenge. In Proceedings of HCI International 2007, Beijing (2007).

[2] Bolt, R.A. Put-that-there: voice and gesture at the graphics interface. *Computer Graphics, 14(3)*, pp. 262--270 (1980).

[3] Bouchet, J., Nigay, L., Ganille, T.: ICARE Software Components for Rapidly Developing Multimodal Interfaces. In: Conference Proceedings of ICMI'2004, State College, Pennsylvania, USA, Oct. 2004, ACM Press, pp. 251--258.

[4] Bourguet, M. L.: A Toolkit for Creating and Testing Multimodal Interface Designs. In: companion proceedings of UIST'02, Paris, Oct. 2002, pp. 29--30 (2002).

[5] Bui T.H.: Multimodal Dialogue Management - State of the Art. CTIT Technical Report series No. 06-01, University of Twente (UT), Enschede, The Netherlands (2006).

[6] Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., Clow, J.: QuickSet: multimodal interaction for distributed applications. In: Proceedings of the Fifth ACM international Conference on Multimedia, Seattle, USA, pp. 31--40, (1997).

[7] Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J. Young, R.: Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE properties. In: Proceedings of INTERACT'95, Lillehammer, Norway, June 1995, pp. 115--120, Chapman & Hall Publ. (1995).

[8] Dumas, B., Lalanne, D., Ingold, R.: Prototyping Multimodal Interfaces with SMUIML Modeling Language. In: CHI 2008 Workshop on User Interface Description Languages for Next Generation User Interfaces, CHI 2008, Firenze, Italy, pp. 63--66 (2008).

[9] Dumas, B., Lalanne, D., Oviatt, S. "Multimodal Interfaces: A Survey of Principles, Models and Frameworks". In Denis Lalanne, Jürg Kohlas eds. Human Machine Interaction, LNCS 5440, Springer-Verlag, pp. 3--26 (2009)

[10] Dumas, B., Lalanne, D., Guinard, D., Ingold, R., Koenig, R.: Strengths and Weaknesses of Software Architectures for the Rapid Creation of Tangible and Multimodal Interfaces. In: Proceedings of 2nd international conference on Tangible and Embedded Interaction (TEI 2008), Bonn (Germany), February 19 - 21 2008 , pp. 47--54 (2008).

[11] EMMA: Extensible MultiModal Annotation markup language: W3C recommandation. http://www.w3.org/TR/emma/

[12] Flippo, F., Krebs, A. Marsic I.: A Framework for Rapid Development of Multimodal Interfaces. In: Proceedings of ICMI'03, Vancouver, BC, Nov. 5-7, pp. 109--116 (2003).

[13] Garofolo, J. Overcoming Barriers to Progress in Multimodal Fusion Research. In AAAI Fall 2008 Symposium Proceedings (2008).

[14] Holzapfel, H., Nickel, K., and Stiefelhagen, R. Implementation and evaluation of a constraint-based multimodal fusion system for speech and 3D pointing gestures. In Proceedings of ICMI '04 (State College, PA, USA, October 13-15, 2004). ACM, New York, NY, pp. 175--182 (2004).

[15] Johnston, M., Cohen, P. R., McGee, D., Oviatt, S. L., Pittman, J. A., Smith, I.: Unification-based multimodal integration. In: Proceedings of the Eighth Conference on European Chapter of the Association For Computational Linguistics (Madrid, Spain, July 07-12, 1997), pp. 281--288.

[16] Krahnstoever, N., Kettebekov, S., Yeasin, M. Sharma, R.: A real-time framework for natural multimodal interaction with large screen displays. In: ICMI'02, Pittsburgh, USA, Oct. 2002 (2002).

[17] Lawson, J-Y., Al-Akkad, A., Vanderdonckt, J. and Macq, B. An Open Source Workbench for Prototyping Multimodal Interactions Based on Off-The-Shelf Heterogeneous Components. Proceedings of the First ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM Press, USA, July 14–17, 2009.

[18] Nigay, L., Coutaz, J.A.: Design space for multimodal systems: concurrent processing and data fusion. In Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (Amsterdam, The Netherlands, April 24 - 29, 1993). ACM, New York, NY, pp. 172--178 (1993).

[19] Oviatt, S.L.: Ten myths of multimodal interaction. In: Communications of the ACM, 42(11), New York: ACM Press, pp. 74--81 (1999).

[20] Oviatt, S. L., Cohen, P. R., Wu, L., Vergo, J., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J., Ferro, D.: Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions. In: Human Computer Interaction, 2000, vol. 15, no. 4, pp. 263--322 [Reprinted in Human-Computer Interaction in the New Millennium (ed. J. Carroll), Addison-Wesley Press, Reading, MA, 2001; chapter 19, pp. 421--456] (2000).

[21] Oviatt, S.L., Coulston, R., Tomko, S., Xiao, B., Lunsford, R., Wesson, M., Carmichael, L.: Toward a theory of organized multimodal integration patterns during human-computer interaction. In: Proceedings of ICMI 2003, ACM Press, pp. 44--51 (2003).

[22] Sharma, R., Pavlovic, V. I., Huang, T.S.: Toward multimodal human-computer interface. In: Proceedings IEEE, 86(5) [Special issue on Multimedia Signal Processing], pp. 853--860 (1998).