ICARE Software Components for Rapidly Developing Multimodal Interfaces

Jullien Bouchet
CLIPS-IMAG
38000, Grenoble, France
Jullien.Bouchet@imag.fr

Laurence Nigay
CLIPS-IMAG
38000, Grenoble, France
Laurence.Nigay@imag.fr

Thierry Ganille
THALES-Avionics
33187 Le Haillan, France
Thierry.Ganille@thales-avionics.com

ABSTRACT
Although several real multimodal systems have been built, their development still remains a difficult task. In this paper we address this problem of development of multimodal interfaces by describing a component-based approach, called ICARE, for rapidly developing multimodal interfaces. ICARE stands for Interaction-CARE (Complementarity Assignment Redundancy Equivalence). Our component-based approach relies on two types of software components. Firstly ICARE elementary components include Device components and Interaction Language components that enable us to develop pure modalities. The second type of components, called Composition components, define combined usages of modalities. Reusing and assembling ICARE components enable rapid development of multimodal interfaces. We have developed several multimodal systems using ICARE and we illustrate the discussion using one of them: the FACET simulator of the Rafale French military plane cockpit.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation]**: User Interfaces-** ***Input devices and strategies, Interaction styles, Prototyping*****;** **D.2.2** [Software Engineering]**: Design Tools and Techniques –** ***User Interfaces.***

General Terms
Algorithms; Human Factors.

Keywords
Multimodal Interactive Systems; Software Components.

1. INTRODUCTION
The area of multimodal interaction has expanded rapidly and since the seminal "Put that there" demonstrator [4] that combines speech and gesture: significant achievements have been made in terms of both modalities and real multimodal systems. Indeed, in addition to more and more robust and innovative modalities such as the phicons [11], conceptual and empirical work on the usage of multiple modalities is now available for guiding the design of efficient and usable multimodal interfaces. As a result, real multimodal systems are now being built in various application domains including medicine [16] and education.

ICMI'04, October 13–15, 2004, State College, Pennsylvania, USA.

The flexibility and robustness of multimodal systems give results of an increased complexity that current software development tools do not address appropriately. Although several multimodal systems have been built, their development still remains a difficult task. The existing frameworks dedicated to multimodal interaction are currently few and limited in scope. Either they address a specific technical problem including the fusion mechanism [9] [14], the composition of several devices [6] and mutual disambiguation [17][9], or they are dedicated to specific modalities such as gesture recognition [19], speech recognition [10] or the combined usage of speech and gesture [13].

In this paper we address this problem of software development of multimodal interfaces by describing a component-based approach, called ICARE (Interaction CARE -Complementarity Assignment, Redundancy and Equivalence-), which allows the easy and rapid development of multimodal interfaces. ICARE framework is based on a conceptual component model that describes the manipulated software components and we are currently developing a tool, a graphical platform for specifying a multimodal interface by direct manipulation. From this specification (ICARE schema), the code of the multimodal interaction is automatically generated. While the ICARE framework provides the services of a multimodal toolkit, the ICARE platform must be compared to a UIMS built on top of the toolkit.

Our discussion will concentrate on input (i.e., from the user to the system) although our model holds for output as well. Nevertheless we did not test our approach for output so far. For input we have developed several multimodal interfaces using our ICARE framework [5]: a Multimodal IDentification system (MID) and a mobile system MEMO. MEMO allows users to annotate physical locations with digital notes which have a physical location and are then read/removed by other mobile users: two versions are currently running, one using a PDA and another one using a Head-Mounted Display. In this paper we present as an example a larger system, FACET, a simulator of Rafale (a French military plane).

The structure of the paper is as follows: first, we present our conceptual model that includes elementary and modality dependent components as well as generic components (reusable components) for combining modalities (fusion mechanism). Going one step further in the implementation process, we describe how the ICARE components are implemented in our framework. We conclude with an example that illustrates how the ICARE components function together. As mentioned above, this example

is based on FACET, a simulator of a French military plane whose main features are presented in the next section.

2. ILLUSTRATIVE EXAMPLE: A MILITARY PLANE SIMULATOR

The FACET system, presented in Figure 1, is a real-time complete flight simulator of the Rafale, a French military plane, for studying future interaction techniques that will be embodied in the Rafale cockpit. Examples of tasks include:

- Pilot the plane. The pilot must follow a predefined trajectory while adapting it according to various parameters including meteorological/geographical conditions as well as plane characteristics.
- Navigate. The pilot must create or modify the flight plan for example by calculating new check points.
- Manage the plane. The pilot must constantly check hydraulic and electric systems.
- Manage the armory system. The pilot must be able to change armory depending on the mission.
- Manage the mission. For example the pilot must be able to protect herself/himself against attacks.
- Communicate with air traffic controllers and with other pilots of the patrol.

Figure 1: The multimodal cockpit of Rafale (a French military plane): FACET a flight simulator.

For performing such tasks, several input modalities are included in FACET, making FACET a good candidate for studying rich multimodal interaction in various contexts. The interface includes the following input modalities:

- The HOTAS (Hands On Throttle And Stick) are made of two command joysticks (one for each hand) to pilot the plane and to issue commands such as marking a point on the ground.
- The helmet visor which allows the pilot to select a target in the real world that depends on the orientation and location of both the pilot and the plane.
- Speech inputs for issuing commands such as marking a point on the ground.
- A tactile surface in between the legs of the pilot for specifying commands by direct manipulation.

As a unique feature and in addition to the multiple input modalities managed by FACET, several interaction contexts are clearly identified. Thus, the pilot can select modalities and forms of multimodality according to the phases of the military mission, the level of stress, the physical parameters (for example when the pilot is exposed to strong accelerations) and so on. As a conclusion FACET defines a very rich case study for multimodality. In this paper we will illustrate our ICARE framework by considering three tasks in FACET.

3. ICARE CONCEPTUAL MODEL

The ICARE conceptual model includes:

1. Elementary components: Such components are building blocks useful for defining a modality. Two types of ICARE elementary components are defined: Device components and Interaction Language components. We reuse our definition of a modality [14] as the coupling of a physical device d with an interaction language L: <d, L>. In [15], we demonstrate the adequacy of the notions of physical device and interaction language for classifying and deriving usability properties for multimodal interaction while in [14][15] we adopt a complementary perspective and examine the relevance of these notions for software design.
2. Composition components: Such components describe combined usages of modalities and therefore enable us to define new composed modalities. The ICARE composition components are defined based on the four CARE properties [15]: the Complementarity, Assignment, Redundancy, and Equivalence that may occur between the modalities available in a multimodal user interface. As opposed to ICARE elementary components, Composition components are generic in the sense that they are not dependent on a particular modality.

In the following sections, we first describe the two types of ICARE elementary components (i.e., Device and Interaction Language components). We then focus on relationships between modalities by describing the ICARE composition components.

3.1 Device Components

An ICARE Device component represents a supplementary layer of the physical device driver. For example, the mouse Device component abstracts the data provided by the mouse driver such as button pressed/released and movement. Likewise a microphone Device component abstracts the captured signal into a recognized utterance while another microphone Device component abstracts the captured signal into a level of noise. In FACET, as shown in Figure 10, examples include an HOTAS Device component and a Pilot's Orientation&Location tracker Device component. Indeed for capturing the pilot's orientation and location, we define only one Device component as we would do for a mouse with two buttons and a movement captor.

All ICARE Device components also enrich the raw data from the device driver by adding information that includes the working state of the device, a time-stamp as well as a confidence factor of

the produced data, and a description of the device in terms of human manipulation (passive/active modalities [5], human actions involved and physical location of these actions).

An ICARE Device component is then linked to a listener component, an ICARE Interaction Language component in order to form a pure modality.

3.2 Interaction Language Components

An ICARE Interaction Language component corresponds to the logical level of an interaction modality. For example an Interaction Language component abstracts the data from a mouse Device component into commands such as the selection of a menu option. In FACET, each Device component is linked with an Interaction Language component. Examples include the HOTAS Commands component which abstracts the data from the HOTAS Device component into a command and the Pilot's Orientation&Location Language component which transforms the raw data from the Pilot's Orientation&Location tracker Device component into a pilot's position.

These three examples of Interaction Language components underline the fact that such components need to rely on an external description of the well formed expressions to be obtained. Indeed in order to abstract data from the mouse into commands, a description of the graphical interface is required. Likewise the HOTAS Interaction Language component maps commands to specific HOTAS buttons. Finally the Pilot's Orientation&Location Language component requires a description of the two coordinates systems (one for the orientation and one for the location).

While an ICARE Device component relies on the underlying physical device that produces its inputs, an ICARE Interaction Language component relies on the Device components that can produce its required inputs.

As for Device components, ICARE Interaction Language components also enrich the data by adding generic information that include the Bernsen's language characterization [3] (linguistic/ analogue/ arbitrary/ dynamic characteristics), a time-stamp as well as a confidence factor of the produced data.

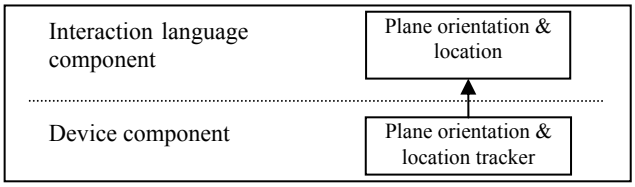

Figure 2: An example of a pure modality in FACET.

Device and Interaction Language components constitute the building blocks for defining modalities. As shown in **Figure 2**, the Device component "Plane Orientation&Location tracker" is linked to the Interaction Language component "Plane Orientation&Location" to form a pure modality. Composition of devices, languages and modality are also possible thanks to ICARE Composition components.

3.3 Composition Components

Although each modality can be used independently within a multimodal system, the availability of several modalities in a system naturally leads to the issue of their combined usage. The combined usage of multiple modalities opens a vastly augmented world of possibilities in multimodal user interface design that we study in light of the four CARE properties [15]. While Equivalence and Assignment express the availability and respective absence of choice between multiple modalities for performing a given task, Complementarity and Redundancy describe relationships between devices, languages or more generally between modalities for performing a given task.

Because the CARE properties have been shown to be useful concepts for the design and evaluation of multimodal interaction [15], we decided to reuse those concepts to make them explicit during the software development. We therefore define three Composition components in our ICARE conceptual model: the Complementarity one, the Redundancy one, and the Redundancy/Equivalence one. Assignment and Equivalence are not modeled as components in our ICARE model. Indeed, as shown in Figure 3, an assignment is represented by a single link between two components. An ICARE component A linked to a single component B implies that A is assigned to B. As for Assignment, Equivalence is not modeled as a component. As shown in Figure 4, when several components (2 to n components) are linked to the same component, they are equivalent. In our previous multimodal systems developed using ICARE components [5], we explicitly used an Equivalence component that had no functional role (no treatment except defining a new time-stamp to the data) but constituted an aid while manually assembling components. Using our platform under construction that will allow the user to graphically assemble ICARE components, such Equivalence component has no more utility. The ICARE conceptual model therefore contains only three Composition components that we describe in the following sections.

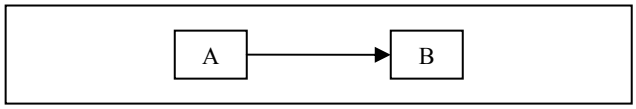

Figure 3: An ICARE component A assigned to another component B.

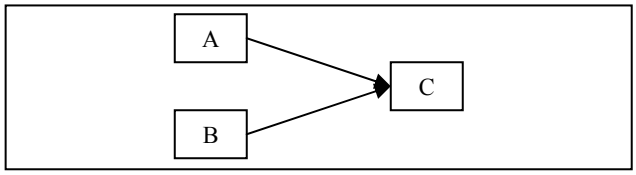

Figure 4: Two ICARE equivalent components (A and B) for a given component C.

These Composition components describe the fusion mechanism of data provided by 2 to n ICARE components. As in our generic fusion mechanism described in [14] as well as the Polymodal Input Module [8], the criteria for triggering fusion are twofold: the complementarity/redundancy of data, and time. As future work, the context and history of interaction needs to be taken into account. For the management of time, a temporal window Temp-win is associated with each piece of data handled by a Composition component. Temp-win defines the temporal proximity of two pieces of data (+/- Δt) and is used to trigger fusion. Two pieces of data d1 and d2 are combined if their time-stamps, t1 respectively t2, are temporally close:

Close (t1, t2) is satisfied if: $t2 \in [t1-\Delta t, t1+\Delta t]$

Δt has been tuned experimentally. One can improve the setting of this parameter by letting the system compute the appropriate values depending on performance of the platform as well as on the behavior of the user.

As opposed to our previous fusion mechanism [14], we assume that the Composition components receive the data in the order of the user's actions. We eliminate the problem of undoing erroneous fusions. Incorrect fusion may occur due to the different time scales required to process data specified through distinct languages and devices. As a result, the sequence of received data by a Composition component is not necessarily identical to that of the user's actions sequence. In the case of the Rafale cockpit, our hypothesis is not restrictive since all the modalities including speech have been developed specifically for the Rafale and guarantee equivalent time for processing the data. For a more general approach, a second version of our Composition components can be developed based on our previous fusion solution that is able to undo incorrect fusions.

3.3.1 Complementarity Component

The Complementarity component combines complementary data that are close in time. For example, in FACET, the orientation and the location of the pilot are combined (in complementary way) with the orientation and the location of the plane to compute which target the pilot specifies using with the visor helmet. As shown in Figure 5, data produced by two modalities are combined by a Complementary component. The Composition component relies on a common structure of the manipulated objects that enables the fusion of the received pieces of data. The structure is described in a declarative way outside the component. By so doing, the Composition component is domain independent: structures that rely on the domain are not "code-wired". They are used as parameters for the component. For the case of the example of Figure 5, the common object is made of two parts, one for containing the plane Orientation&Location and one for containing the pilot's Orientation&Location. We define a mapping between the structural parts of the object and the input ports of the component. For each object maintained by the component, we compute its temporal window Temp-win: Temp-win=[Tmin-Δt, Tmax+Δt], Tmin being the time-stamp of the oldest piece of data contained in the object while Tmax the time-stamp of the newest piece of data. Temporal windows are used to trigger fusion of objects.

The mechanism of the Complementarity component is driven by a set of rules:

R1. For a newly received piece of data, the Complementarity component first creates a new object that contains the new piece of data at the right place in the object structure by checking the input port through which the piece of data was received.

R2. If other incomplete objects exist and can be combined with the newly created object according to their contained pieces of data and temporal windows, fusion is performed while the initial new object is maintained. The newly received piece of data is therefore duplicated as many times as necessary.

R3. When only one object is complete, the Complementarity component propagates it to the following connected ICARE component. When several objects are complete, the component propagates the object with the shortest temporal window (i.e., the object that contains pieces of data that are the closest in time).

R4. When a complete object is sent to the following ICARE component, all the contained pieces of data of the object that are duplicated in the remaining objects are suppressed. Empty objects are then removed, and Rule 2 can be applied again on the remaining objects.

R5. An incomplete object can be removed from the set of candidates for fusion when its life span expectancy expires. The life span of an object equals Tmax+Δt, Tmax being the time-stamp of the newest piece of data contained in the object. Removing an incomplete object does not mean that the corresponding user's actions are ignored: The actions are processed in another ICARE component chain.

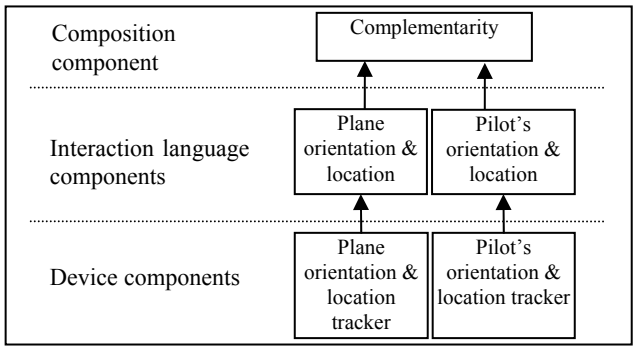

Figure 5: An example of complementarity in FACET: the task consists of identifying the point selected by the pilot using the helmet visor.

Another parameter of the Complementarity component enables us to specify a temporal order for filling an object. For example let us consider the "Put that there" example of R. Bolt [4]. We can specify that first the command must be received and then a first click for specifying "that" and finally a second click for specifying "there". Such temporal order reduces the flexibility offered to the user; nevertheless it can correspond to an empirical result from observed behaviours [18].

Finally our Complementarity component does not explicitly manage reference resolution. To do so, further work must be done. We plan to enrich the mechanism of the Complementarity component with the notion of confidence factor. The notion of confidence factor provides a simple mechanism for modeling uncertainty and can be usefully exploited for solving ambiguities in deictic expressions. Considering the previous example "Put that there", a click can either correspond to "that" or "there". In this case, the corresponding piece of data will be duplicated in the object O1 structure and each piece of data will be associated with a very low confidence factor. If a newly received piece of data stored in a new object O2 has a higher confidence factor, then the two objects O1 and O2 can nevertheless be combined although their contained pieces of data are not complementary since one piece of data has a very low confidence factor.

3.3.2 Redundancy Component

Redundancy [15] corresponds to the case where two modalities convey redundant pieces of information that are close in time. In such a case, one of the two user's actions must be ignored. For

example in FACET, we could use two redundant modalities, voice commands and commands specified by pressing HOTAS buttons for changing the phase (from a navigation phase to a fighting phase). In such a case shown in Figure 6, the current phase will be changed only if the pilot issues the speech command while pressing the right HOTAS button.

The mechanism of the Redundancy component is close to the one of a Complementarity component except that the values of the pieces of the data at the same place in the structure of two objects must be compared to determine a case of redundancy and therefore triggers a fusion of the two objects. The resulting object will then contain redundant pieces of data. The five rules of the Complementary component govern the mechanism of the Redundancy component, except that a complete object is not propagated as it is. Indeed the object contains redundant pieces of data (n redundant pieces of data if n modalities are connected to the Redundancy component) and only one piece of data is sent to the following connected ICARE component. The selection is based on the data confident factors: the piece of data with the highest confidence factor is sent to the following ICARE component.

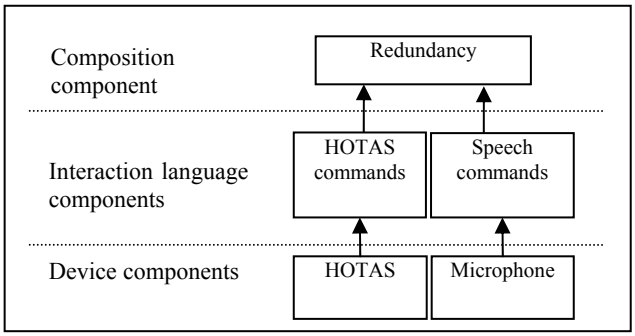

Figure 6: An example of redundancy in FACET: the task consists of changing the current phase (from a navigation phase to a fighting phase).

Moreover and as for the Complementarity component, a parameter of the Redundancy component makes it possible to specify a temporal order in the usage of the modalities that convey redundant information.

3.3.3 Redundancy/Equivalence Component

The Redundancy/Equivalence component is a mix of the two CARE properties: Redundancy and Equivalence. It corresponds to the Redundancy component where redundancy could be optional. For example as shown in Figure 7, the pilot can mark a point on ground, that will then be displayed in the Head-Mounted display, either by speech or by pressing an HOTAS button or by pressing the HOTAS button while issuing the voice command in parallel (redundant case). For all three possible cases, only one point will be marked.

As explained in Figure 8, such a component is not mandatory in the ICARE conceptual model. It is introduced for simplifying the ICARE specification since such flexible usages are quite common in multimodal user interfaces.

The mechanism of the Redundancy/Equivalence component is more complex than the Redundancy component since the corresponding supported interaction is richer. The Redundancy/Equivalence component includes two different strategies: the "eager" and the "lazy" strategies. We provide an efficient mechanism with the "eager" strategy and a safety one with the "lazy" strategy.

Adopting an "eager" strategy, the component does not wait for further pieces of data and therefore continuously propagates data to the following connected ICARE component. Each time a piece of data is sent to another ICARE component, the component nevertheless keeps track of the data and starts a timer (Δt) in order to be able to detect redundant pieces of data that may be received later. This approach has the advantage of providing the user with immediate feedback. The drawback is that the piece of data that is propagated is the first one received by the component and may not have the highest confident factor. As opposed to the "eager" strategy, the "lazy" strategy is waiting Δt before propagating the piece of data. The advantage of this strategy is to guarantee to always propagate the piece of data with the highest confidence factor as done by the Redundancy component.

As for the Complementarity and Redundancy components, a parameter of the Redundancy/Equivalence component makes it possible to specify a temporal order in the usage of the modalities that convey redundant information.

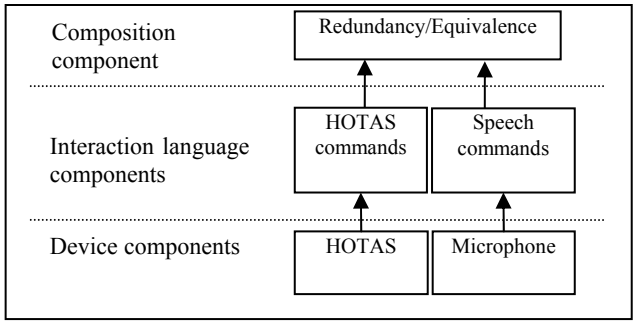

Figure 7: An example of Redundancy/Equivalence in FACET: the task consists of triggering the action "mark a point on ground".

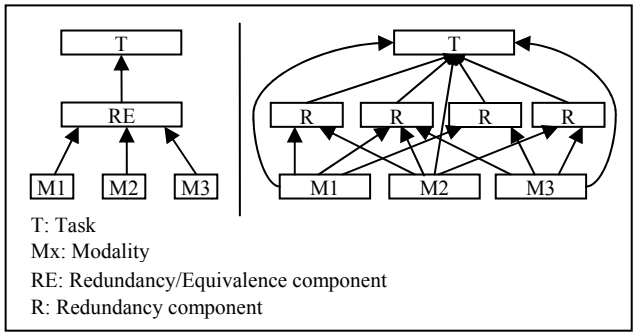

Figure 8: Redundancy/Equivalence of three modalities M1, M2 and M3.

To conclude, the three types of Composition components described above are generic as they are modality as well as domain independent. In addition as many ICARE components as necessary can be connected to a Composition component. Such reusable generic codes enable a rapid development of multimodal interfaces. In addition it is very easy to modify the supported form of multimodality by changing a Composition component by another one. For example if after experimental tests with pilots, we decide that the Redundancy for the task of changing phases is

too constraining for the pilot (Figure 6), we can easily change it by a Redundancy/Equivalence component (Figure 7) that offers more flexibility for the pilot.

Having presented the ICARE conceptual model made of Device and Interaction Language components as well as generic Composition components, we now focus on more technical details by presenting our ICARE implementation model.

4. An ICARE Implementation Model

4.1 Software Component Technology

Such as the other multimodal systems developed with ICARE [5], we have chosen the JavaBeans technology [12] to develop ICARE software components because it is multi-platform, thanks to Java, and it provides many mechanisms for graphical manipulation that we are reusing for our ICARE platform. Properties of ICARE components, such as the interaction location of a Device component, are class attributes which can be accessed/modified (get/set).

4.2 Communication between ICARE Components

A link between two ICARE software components denotes communication between them. For communication our implementation of the conceptual ICARE model is based on the Java event model. Events are messages that one ICARE software component sends to another one. A software component which generates and sends an event is called the "source component". Generated events transport the data to be processed, the identification of the source component, the time-stamp of its creation and its confidence factor. Each ICARE software component implements a FIFO (First In First Out) queue of events. A thread is created for processing data of the first event in the queue and to generate a new event. Events are managed in an asynchronous way and thus the source software component is not blocked by the processing of other software components. To assemble two ICARE components, it is necessary that one component subscribes to events generated by the other component.

4.3 Communication between ICARE Components and the Other Parts of the System

To fully understand the scope of our ICARE framework we show in Figure 9 where ICARE software components are situated along the ARCH software architectural model [2].

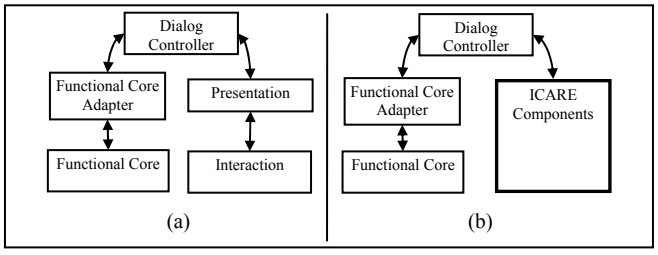

Figure 9: (a) The ARCH software architectural model. (b) ICARE components within an ARCH software architecture.

As shown in Figure 9, an ICARE component chain defines a pipeline from user's actions to commands or elementary tasks understandable by the Dialog Controller. So the last ICARE component of a chain must communicate the elementary task to the Dialog Controller. Such link can be performed by direct call of methods or by using TCP/IP, UDP, JavaRMI and so on. In the following section, we explain how we developed the multimodal interaction using ICARE for three interactive tasks of FACET, highlighting how we implemented the link between an ICARE component and the FACET Dialog Controller.

5. FACET ICARE COMPONENTS

We present the ICARE implemented software component assembly for three tasks of FACET. The first task (T1 in Figures 10 and 11) is to determine the orientation and the location of the plane, information useful for the FACET Functional Core. The second task (T2) consists of marking a point on ground while the third one (T3) corresponds to unmarking a point.

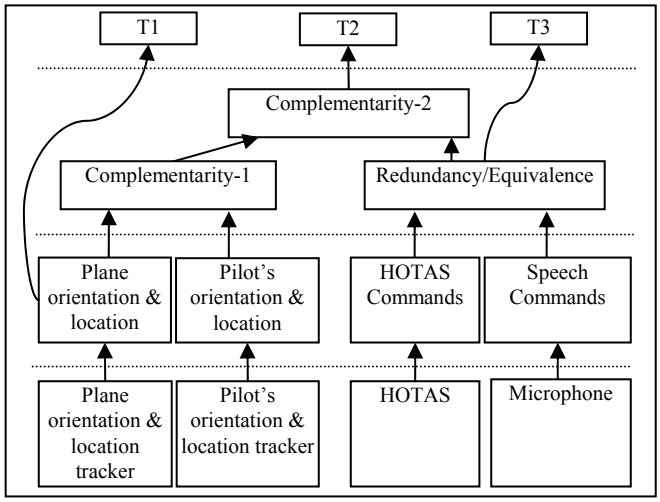

Figure 10: ICARE implemented component assembly for three tasks of FACET: T1=Orientation&Location of the plane, T2=Marking a point on ground and T3=Unmarking a point.

Figure 10 shows the assembly of components for the three tasks. Four modalities are implemented:

- M1= <Plane Orientation&Location tracker, Plane Orientation&Location>
- M2 = <Pilot's Orientation&Location tracker, Pilot's Orientation&Location>
- M3= < HOTAS, HOTAS Commands>
- M4 = <Microphone, Speech Commands>

M1 and M2 are used in a complementary way for detecting the target selected by the pilot using the helmet visor: Complementarity-1 component. In addition M1 is assigned to the task T1.

For performing a marking command, the pilot has the choice between two modalities, M3 and M4 that are functionally equivalent but can also be used in a redundant way thanks to the Redundancy/Equivalence component. As a consequence if the pilot is pressing the HOTAS button while speaking, one point will be marked and if the pilot presses the HOTAS button and later

issues the voice command <Mark>, two commands will be transferred to the Dialog Controller. Nevertheless two points will not be marked because only one point can be marked at any time. So the first marked point will be suppressed before displaying the new marked point that corresponds to where the pilot was looking at, while issuing the voice command. Due to the real-time constraint on the system, we used an eager strategy for the Redundancy/Equivalence component. Finally in order to obtain a complete marking command, the command <Mark> must be combined (Complementarity-2 component) with the target point defined by the pilot using the helmet visor.

For unmarking a point, the pilot needs to specify the command <Unmark> using M3 or M4 or (M3 and M4) used in a redundant way.

As 2 to n ICARE components can be linked to the Complementarity component, another possible assembly for the same FACET tasks is presented in Figure 11. Within this new assembly, we combine the two Complementarity components and we obtain one single Complementarity component with three connected ICARE components.

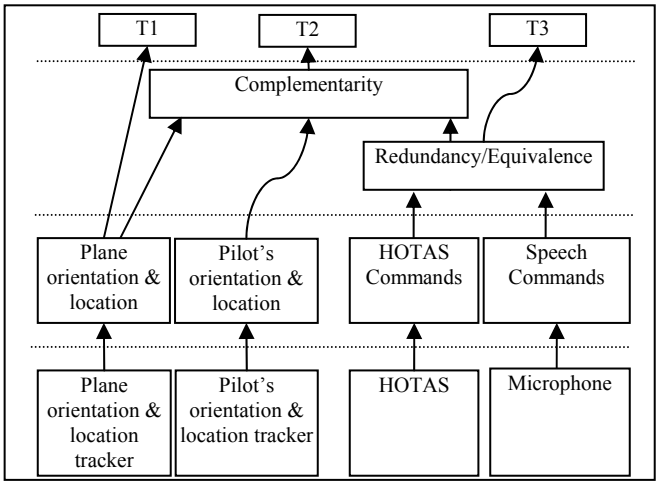

Figure 11: Another possible ICARE component assembly for three tasks of FACET: T1=Orientation&Location of the plane, T2=Marking a point on ground and T3=Unmarking a point.

For developing FACET, our ICARE approach enables us to reuse existing modalities such as speech (developed by THALES-Avionics) and will facilitate code modifications and evolution. For example allowing a new modality for marking a point will easily be managed by adding a new component connected to the Redundancy/Equivalence component. In addition forcing the redundant usage of voice commands and commands specified by pressing HOTAS buttons for marking a point can be achieved by solely changing the Redundancy/Equivalence component by the Redundancy component.

The whole architecture of FACET is along the ARCH model. Communication from the two ICARE components Complementarity-2 and Plane Orientation&Location has been implemented using Java RMI. We envision using TCP/IP for the tasks T2 and T3 and UDP for the task T1. UDP will provide a more efficient communication adapted for frequently sending the orientation and location of the plane, while TCP/IP a more safety communication for the commands issued by the pilot.

This first version of the FACET simulator allows us to validate the ICARE approach for a real-time system. We measured the processing time of each implemented ICARE component in FACET with the following frequencies:

- 100 Hz for the plane Orientation&Location tracker
- 60 Hz for the pilot's Orientation&Location tracker
- 0.66 Hz for the commands with HOTAS buttons (2 times over 3 were the marking command and 1 time over 3 was the unmarking command)
- 0.125 Hz for the commands using speech input modality (2 times over 3 were the marking command and 1 time over 3 was the unmarking command).

For 10000 performed tests we obtained the following results for each modality:

- M1, Plane Orientation&Location modality: 0.18 ms
- M2, Pilot's Orientation&Location modality: 0.17 ms
- M3, HOTAS commands: 0.16ms
- M4, Speech commands: 0.26 ms

For 300 performed tests, our evaluation of performance of the implemented Complementarity and Redundancy/Equivalence components was satisfactory with regards to the real-time constraints on FACET. For the Redundancy/Equivalence component we fixed Δt equal to 100 ms while for the Complementarity component of Figures 10 and 11, we fixed Δt equal to 10 ms.

6. CONCLUSION & FUTURE WORK

On the one hand, our ICARE conceptual model has been validated by manually assembling the components for the development of several multimodal systems. ICARE is based on a component-based approach and therefore offers the well-known advantages of component-based development, that are to reduce the production costs, and to verify the software engineering properties of reusability, maintainability and evolution [1]. With the real-time FACET system presented in this paper, we also validate ICARE for a real-time system.

On the other hand, we are aware that input and output interaction modalities are strongly linked. For example the usage of a particular input modality may have an impact onto the choice of the output modalities. In a near future, we will apply our ICARE approach for output multimodal interaction and study the links between the input and output ICARE components. Moreover the context may have an impact onto the modalities. For now on in FACET, the context is an external module that can for example stop a modality. For instance speech input cannot be used in a very noisy environment. A more complex approach than the basic on/off one would be to modify the confidence factors attached to the data.

As on-going work, we are finishing the development of our graphical ICARE platform that enables the designer to graphically manipulate and assemble ICARE software components in order to specify the multimodal interaction dedicated to a given task of the interactive system under development. From this specification (ICARE schema), the code of the multimodal interaction is automatically generated. The user of the ICARE platform selects the modalities (Device and Language components) and specifies

the combination of modalities by selecting a Composition component, all by graphically assembling software components without knowing the details of the code of the components. Figure 12 presents a sketch of the user interface of the ICARE platform: it contains a palette of components (area A on Figure 12), an editing zone for assembling the selected components (area B on Figure 12) and a customization panel (area C on Figure 12) for setting the parameters of the components. We plan to automatically check ergonomic properties while the designer is specifying the multimodal interaction. For example action continuity [7] can be automatically checked based on ICARE Device component properties and the global consistency in the usage of modalities can also be checked. A panel (area D on Figure 12) shows the alert messages related to ergonomic properties while the designer is specifying the multimodal interaction.

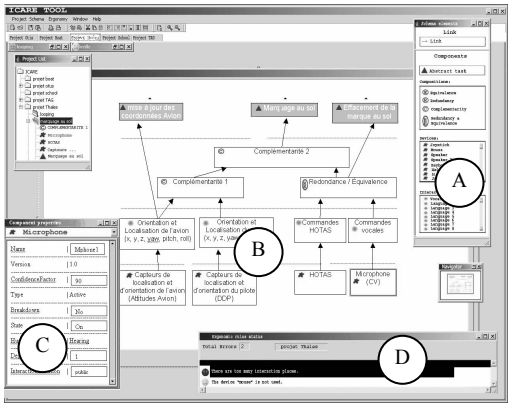

Figure 12: Sketch of the graphical ICARE platform.

For the ICARE platform, we identify two types of target users.

- First the developer is responsible for enriching the set of components by adding new Device and Language components. The developer could also decide to provide encapsulated components, for example a complete modality component (that includes a Device and a Language components). For adding an already developed modality to the platform, the developer simply needs to encapsulate the corresponding code into an ICARE component. It is our approach in the FACET system since all the modalities have been developed by THALES specifically for military plane cockpits.

- Second the designer is defining multimodal interfaces by directly manipulating components using the mouse. It does not need to understand the details of the component code. As opposed to existing JavaBeans editors including BeanBuilder, Jbuilder and ViusalAge, our ICARE platform makes explicit the notions of device as well as language and the CARE properties whose adequacy for studying the usability has been demonstrated [15]. At the completion of the ICARE platform, an experimental evaluation involving designers must be carried out.

7. ACKNOWLEDGMENTS

Many thanks to G. Serghiou for reviewing the paper. This work is partly funded by DGA (French Army Research Dept.) under contract #00.70.624.00.470.75.96 and by the SIMILAR European FP6 network of excellence dedicated to multimodality (http://www.similar.cc).

8. REFERENCES

[1] Bass, L. et al. market Assessment of Component-Based Software Engineering. *SEI Technical Report* (2000).

[2] Bass, L. et al. A Metamodel for Runtime Architecture of an Interactive System. *The UIMS Workshop Tool Developers, SIGCHI Bulletin, 24 (1)* (1992), 32-37.

[3] Bernsen, N. Modality Theory in support of multimodal interface design. *Proceedings of Intelligent Multi-Media Multi-Modal Systems* (1994), 37-44.

[4] Bolt, R. A. "Put-that-there": Voice and gesture at the graphics interface. *Proceedings of SIGGRAPH'80, 14, 3* (1980), 262–270.

[5] Bouchet, J., Nigay, L. ICARE: A Component-Based Approach for the Design and Development of Multimodal Interfaces, *Extended Abstracts CHI'04* (2004), 1325-1328

[6] Dragevic, P., Fekete, J.-D. ICON: Input Device Selection and Interaction Configuration, *Demonstration, UIST 2002 Companion* (2002), 47-48.

[7] Dubois, E., Nigay, L., Troccaz, J. Assessing Continuity and Compatibility in Augmented Reality Systems. *UAIS Journal, 4* (2002), 263-273.

[8] Elting, C. et al. Architecture and implementation of multimodal plug and play. *Proceedings of ICMI'03* (2003), 93-100.

[9] Flippo, F., Krebs, A., Marsic, I. A Framework for Rapid Development of Multimodal Interfaces. *Proceedings of ICMI'03* (2003), 109-116.

[10] Glass et al. A Framework for Developing Conversational User Interfaces, *Proceedings of CADUI'2004* (2004), 354-365.

[11] Ishii, H., Ullmer, B. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. *Proceedings of CHI'97* (1997), 234-241.

[12] JavaBeans 1.01 specification, Sun Microsystems (1997), http://java.sun.com/products/javabeans/docs/

[13] Krahnstoever et al. A real-time framework for natural multimodal interaction with large screen displays. *Proceedings of ICMI'02* (2002).

[14] Nigay, L., Coutaz, J. A Generic Platform for Addressing the Multimodal Challenge. *Proceedings of CHI'95* (1995), 98-105.

[15] Nigay, L., Coutaz, J. The CARE Properties and Their Impact on Software Design. *Intelligence and Multimodality in Multimedia Interfaces*, (1997).

[16] Oviatt, S. et al. Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions. *HCI*, 15, 4 (2000), 263-322.

[17] Oviatt, S. Taming recognition errors with a multimodal interface. *Communications of the ACM, 43, 9* (2000), 45-51.

[18] Oviatt, S. Ten Myths of Multimodal Interaction. *Communications of ACM, 42, 11* (1999), 74-81.

[19] Westeyn, T. et al. Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition. *Proceedings of ICMI'03* (2003), 85–92.