E-INSTRUMENTATION: A COLLABORATIVE AND ADAPTATIVE INTELLIGENT HCI FOR REMOTE CONTROL OF DEVICES

Christophe Gravier
DIOM laboratory
23 rue du docteur Paul Michelon, 42000 Saint-Etienne, France

Jacques Fayolle
DIOM laboratory
23 rue du docteur Paul Michelon, 42000 Saint-Etienne, France

Keywords: e-laboratories, HCI adaptation, CSCL (Computer Supported Collaborative Learning), SDTP (Same Time, Different Place).

Abstract: What is being discussed here is a Computer Supported Collaborative Learning System. The learning system aims at providing a distant, collaborative and adaptative access to high technological devices. Some strong points in the fields of HCI are to be stressed : the user is expected to access the device while being in a group of users, that is to say ”group awareness” must be supported by some mechanisms. In order to get a generic platform, being ”instrument-independant”, tools are expected to be provided for the graphic user interface to be easily (and without too much user skills) built. Moreover, the notion of sequence of utilization of a device could possibly be a tool enabling new models of evaluation of graphic user interface (consequently HCI) and/or user behaviors.

1 INTRODUCTION

Many Computer Supported Collaborative Learning systems had put the stress on the need of a correct modeling of devices. Mainly, issues already discussed are display models of an existing (complex) device and the modeling of interactions between human and computers. The ”eInstrumentation” platform offers the possibility to remote control devices such as microwave analyzers, hypermetric ellipsometers, antenna bench or even optic-fiber stretchers (for the time being). A distant user (formattor as example) can manipulate an instrument on the Internet, meanwhile users (learners as example) can watch and manipulate the the same distant device too. All users are being able to watch what another user is doing for the current manipulation.
Regarding the wide range of instruments being addressed by the platform, issues listed above were things we had to take into account. Challenges ahead were collaborative work, study of usability reflexes (cognitive factors), standard and non-standard use of instruments (Human-Computer Interaction (*henceforth HCI*) usage validators).
In order to response to all those problems, each issue will be exposed, with potential solutions and, eventually, a possible implementation suggested. At first, custom HCI tools used will be presented. Then, advanced HCI features are to be explained. This will help in introducing the notion of composite actions within devices' HCI. Next, group awareness support will be presented as a ”collaborative service” (just the same way ”security service” or ”data-access service” are widely accepted services that exist in applications). Finally, a use case will be presented to illustrate the past thoughts.

2 GRAPHIC USER INTERFACE FOR DEVICES

It is a matter of facts: the Graphical User Interface (*henceforth GUI*) representing such heterogeneous devices could not be generic.

2.1 Facts

A trivial example: thinking about a single GUI which would allow both the representation of electronic devices (mainly buttons) *and* mechanical devices (mainly things that move) is not something one can achieve without too much ”special case” within the ontology of the instruments. Moreover, all fields

Gravier C. and Fayolle J. (2006).
E-INSTRUMENTATION: A COLLABORATIVE AND ADAPTATIVE INTELLIGENT HCI FOR REMOTE CONTROL OF DEVICES.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - HCI*, pages 87-94
DOI: 10.5220/0002495100870094

of instrumentation could not be addressed in the ontology: some could be forgotten, others may not exist yet.
In the end, it can hardly be said that a single XML[1] file could possibly cover all the descriptions of all the amount of instrument that exists. That is the reason why tools have be developed in order to let one device's administrator *draw* the GUI representing the instrument he wants to share.

Instead of classifying with excess, help must be provided in order to generate a custom GUI. That is the idea here. Hence, severals ways of device's representation build have been studied. Finally, two principals methods have been point out:

- create a *GUI drawer*, which would integrate collaborative components and an "actions ordonancor" (see section 3)
- use image processing algorithm in order to create the virtual representation of the instrument from a picture performed by a camera.

2.2 GUI Drawer Helper

The drawer have the advantage to let the user build the entire representation of the device. He picks "collaborative aware" components up from a toolbox and drag them into the representation build in progress. Moreover, one can supply the list of functionalities of the instrument (i.e. name of the function to call instrument-side), the "builder" (the user that builds the representation) could possibly map widget to the corresponding function. All the values of those properties are stored within a XML file. A DTD[2] is provided to validate the XML. In those conditions, the representation of a instrument is fast, reliable, normalized and do not needs special technical needs (except one already require: the knowledge regarding the instrument's usage).

2.3 Image Processing to Reinforce Assistance

It has already been discussed that HCI "lies at the crossroads of many scientific areas" (Sebe1 et al., 2004). As for eInstrumentation's HCI, it is already at the crossroads of (just to quote few) : collaborative work, middleware or pedagogics. Nevertheless, another (unsuspected!) area can be added: this is image processing. Regarding the construction of the device's representation, a simple idea come up. One can take a picture of an instrument, then tag widgets that are "actionable" with an image manipulation tool (such a TheGimp (http://www.gimp.org/, 2005)). A widget that is colored in the picture is thus an active widget. Hence, image processing enables to analyze the picture taken, thus retrieving widgets composing the instrument (and their characteristics such as position or shape). As said earlier, the "drawer" produce a XML file which is validate by a DTD grammar. This image processing treatment also produce the same structured file. This allow to load within the drawer the result of the image processing algorithms and thus modify eventually the representation of the instrument (the image processing may not be perfect: sometimes it can hardly decide whether the widget is a state button (on/off as example) or simple circle push button). In other words, some adjustments can possibly be made in the drawer, as well as the mapping between widgets and functionalities.

The integration of the tools is illustrated in figure 1. It has to be noticed that Java technologies are being employed. Moreover, it is important to understand that there is only *one* DTD document and only *one* XML file. In fact, only the XML file is related to a specific instrument, all other files are generic.

3 ADVANCED HCI FEATURES FOR E-INSTRUMENTATION

It is obvious that the use of a computer *is* introducing a new intermediary. Nevertheless, those drawbacks are to be fought. That is to say "felt-life" (McCarthy and Wright, 2004) has *also* to be translate within the platform, for the computer link to be as transparent as possible. This is the purpose of this section.

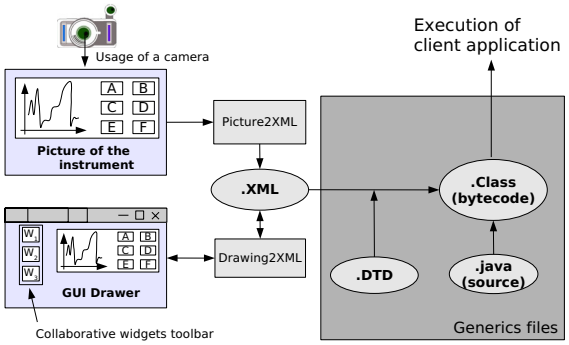

Figure 1: Integration of tools in a single GUI building architecture.

3.1 One Need to Feel

"Felt-life" refers to actions or reactions about what one had seen, heard or touched before. This is something very important in an project that relies on hand-

[1] eXtensible Markup Language
[2] Document Type Definition

on approaches. It sounds clear that feelings are part of the memory mechanism: one remembers the shape of an object, the smell of a device, the noise made by an instrument or even the touching of a device. These are all being stored in our memories in order to help us to remember previous the actions. In HCI, reproducing those feelings helps a lot in improving our capacities to develop memory reflexes and can even facilitate cognitive walkthrough processes (Wharton et al., 1994). In hand-on approaches, it is clear that reproducing the environment of manipulation of the devices could greatly improve the settle of memory mechanism. For eInstrumentation platform, this is done by playing sound (ex: an antenna that is moved on a bench makes a sound that can be reproduced). For an optic fiber stretcher or any device where mechanical constraints appear, widgets reproduce the mechanical resistance associated when actioning it. This example of mechanical resistance can be classified as interface metaphors and their need have already been discussed (Guss, 2003).

On the implementation part, this is quite widget-centered approach. It surely needs to become more generic. in addition and unfortunately, ”smelling factors” are not being discussed nowadays.

3.2 Collaborative Widgets: Widgets that Work for Collaborative Learning

It is obvious that collaborative learning means serverals users learning together, in the same time, using the device, inside the same group (”Same-Time-Different-Place” (Jr. et al., 2005)). This is generally spoken as ”group awareness” (M.Roseman and Greenberg, 1996). It is really important to present collaborative requirement as a *service*, just like security, transactions or persistence *are* (Dery-Pinna et al., 2003). The group awareness is excepted to provide learners to possibility to be aware that they are working within a group and that they are working for a common task (here and henceforth: basically to learn). Still, it is not easy to support group awareness (Lukosch and Roth, 2001). It is interesting to see how developers maintain it (Gutwin et al., 2004). Indeed, some basic approaches usually result in using one cursor for *all* users or one cursor for *each* user. This usually leads to whether a ”scrollwar” (Stefik, 1987) in case of a single cursor, or a loss of legibility. Regarding the case study eInstrumentation, it has been chosen that the group awareness will not be implemented using cursors but components notifications (change of behavior of widgets). Even if this is a little more important regarding implementation, it still offers more accurate and proper interactions inside a group.

3.3 Possible Implementation Using J2EE

For example, if several users are using a common device, when one performs a command, it is fired to all users in group and will make the widget blink for a short period of time. This way, all users will be ”friendly-notified” of an event concerning the group. In order to achieve this, basic Java widgets implementing this collaborative behavior had been produced. Those are widgets inheriting Java Swing components, implementing the use of an Asynchronous Middleware in its Publish/Subscribe model (see figure 2).

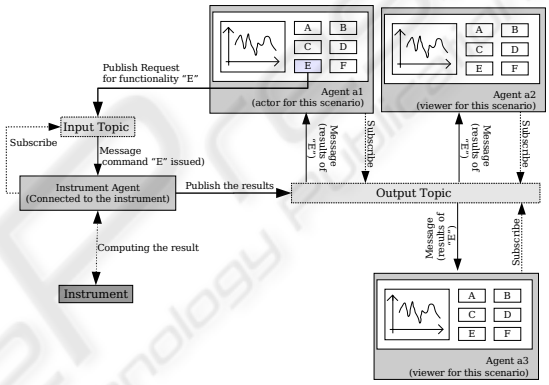

Figure 2: Using an Asynchronous Middleware in order to supply group awareness.

In fact, all client instances subscribe a common topic of interest. Then, when an action is performed by a single user, the actioned widget, which implement group awareness, relies the event on the asynchronous middleware. This way, all the users, who subscribed the topic, receive the message and the application automagically fires the corresponding event to the concerned widget(s). At the end, concerned widget(s) blinks and changes its color for a short period of time. The Asynchronous Middleware implementation is JORAM (http://joram.objectweb.org/, 2005) and the J2EE application server is JONAS (http://jonas.objectweb.org/, 2005). The widgets are a library that has been created for such a use in our laboratory. They are implementing basic collaborative widgets and are being enriched each time a new widget has to be created for a new device integration within the platform. One could think about what is achieved when several groups of users want to access the same instrument(s). An innovating mechanism is being developed in eInstrumentation : if the middleware is able to save/load the state of an instrument (value of all revelant inputs and outputs), at a given time, then there could be a state associated

to every group of users. This way, if two groups are using the same instrument, the middleware could switch the instrument from one state to another depending on the group being active. Of course, the more the groups tend to reach to same loads of utilization of the instrument, the less the platform will be reactive, since it will loose too much time in switching instruments' states. However, if there are severals instruments and severals groups, all the amount of devices could be used as a single device and being viewed by users as a generic device. This way, all commands entering the device will be reoriented to an instrument of the group of instruments, it will load the corresponding state, compute the command, and the send back the result using the asynchronous middleware. This leads to parallelism for the computing of the commands. It is just the same interpretation that the one with processor and "time slot" where instruments split their available computing times into parts for a (short) given period of time. Some dynamic load balancing can also be created in order to set strong priorities to devices geographically nearer to users using the "virtual" device. It would help too in transferring states from one instrument to another in case an instrument breaks down. This would result in *localization transparency* for the instruments within the platform: users will not be able to tell which instrument the command was sent to. Another idea would be to duplicate commands to severals instruments in order to insure HA (high availability) of the service.

3.4 Classifying Widgets for the Leverage of Pedagogical Materials

Away from the support of the memory mechanisms, one can be helped regarding its teaching by being introduced new notions step by step. That would imply for a eLaboratory to let the user discover advanced functionalities one by one after having been explained the basis. This approach implies technicals and process problems: *who* and *how* should decide of the level of usage to associate to a widget ? The point in this paper is to keep a user centric resolution. This means this would be users (in fact the users that manage the instruments) who will decide whether a widget is "level 1", "level 2" or more advanced. ELab must realize an easy widgets classification. In eInstrumentation, the choice has been made to use the "GUI drawer" (see 2.2). It has been said that the GUI drawer could eventually use an image processing algorithm in order to identify widgets (one has to color widgets first). The proposition (see 2.3) is to use colors in order to identify widgets but because the image processing algorithm cannot be perfect (in a two-dimensions representation), it is hard to differentiate a push button from a two state button. Consequently, a possibility is to associate one color to one kind of widget (the user could paste an orange polygon on each push buttons and a blue polygon on every two states button). This would result in the identification of all the widgets and all their nature (without any doubt). Then, it is easy to go on the building of the corresponding XML file validated by the DTD document presented above. Nevertheless, this may not be enough to ensure the differentiation between widgets' level of utilization. The proposition made by eInstrumentation is to use range of the same color to express the level of utilization. For example, stating that push button would be tagged in "pure red" (RGB would be 255,0,0). Then every push button requiring an upper level of utilization would still be tagged in red, but a slightly clearer one. This way, the lighter the red used is, the higher the level of usage associated is. Finally, the image processing algorithm is able to determine the number of level used (the number of color sued in the red color range). This method had to be applied for all the different kind of widget present in the picture of the device. The XML file is still easy to build since, basing on a picture tagged by an user, all the information required are present. Of course, there is still the possibility to change the level of utilization of a widget in the GUI drawer, by opening the XML file produced by the image processing algorithm.
Nevertheless, this solution, as it is, could not be implanted since it does not representing a real usage of the eLaboratory. Indeed, one should not think about a representation of a device by composing widgets *but* by composing valid sequences that aggregates widgets. Sequence of widgets is the purpose of the following section.

4 NORMALIZE WIDGETS, SEQUENCES AND ACTIONS DESCRIPTION

4.1 Aggregating Widgets and Setting up Sequence Validators/Invalidators

Still in order to perform a significant advance in the pedagogic field, using HCI capacities, one could except from the platform to make suggestions about the widgets to action, depending on the previous widgets actioned. This lead the user to a valid sequence of orders aiming at a specific action. This implies to think about the HCI (is this HDI, for Human-Device Interactions ?) in terms of sequences of actions. For deliv-

ering more powerful HCI features, sequences are to be extracted from a HDI study. One could possibility list all the valid sequences of utilization of a device in order to show the possible widgets depending on the previous widgets entered. The implementation result in checking, at each widget interaction, the possible sequence the user is up to realize. It is to be noticed that the opposite is still a pedagogical issue: if a user entered an invalid sequence (one that is not in the valid sequences), the platform has to react, explaining what the user has done wrong and eventually (if informed) why. Guidelines for such issues have already been covered previously (Fong et al., 2004). Next, the application provides shortcuts to potential sequences the user may have wanted to enter. This very user-centric approach helps in providing cognitive support (Soloway et al., 1994). On the programming part, the main point is to formalize those sequences for the *generic* (common for all devices) Java platform client that interprets the sequences associated to a device. In fact, since there is a XML document for widgets list, with its DTD associated, the list of the sequences that are expected with the use of an instrument are being wrapped in the same document. Of course this implies a cooperation between designers and developers (Borchers, 2001).

4.2 Using Boole's Algebra to Solve Sequence Normalization Problem

Nevertheless, the major issue in the ontology of sequences is to represent all the chronological possibilities. That is true that a sequence (and so an action !) could be validated by actioning different widgets, in a different order. For example : λ_1 is a sequence composed by widgets w_1, w_2, w_3 and w_4 and assuming that λ_1 can be written like:

$$\lambda_1 = w_1 \& (w_2 | w_3) \& \& w_4 \quad (1)$$

The sequence λ_1 must be read like follow : λ_1 is validate when widget w_1 is actioned, then w_2 *or* w_3 is used, and finally w_4 used. In this description language for sequences:

- & is coordination without order (A & B is widget A then widget B as well as B then A).
- && is causal ordering (A && B is widget A then widget B and *not* B then A, which would be B && A).
- | is OR (inclusive OR). (A | B is widget A or B, or both of them but not none of them).
- $\oplus$ is XOR (exclusive OR). (A $\oplus$ B is widget A or B, but not both of them, not none of them).

In this condition, it is obvious that sequence λ_1 strictly equals sequence λ_2 or even λ_3, as:

$$\lambda_2 = (w_2 | w_3) \& w_1 \& \& w_4 \quad (2)$$

$$\lambda_3 = (w_3 | w_2) \& w_1 \& \& w_4 \quad (3)$$

Of course, there is still the possibility to introduce some other operators (for theory use for now). For example the negation of order (based on previous declarations):

$$\overleftarrow{\lambda_1} = w_4 \& \& w_1 \& (w_2 | w_3) \quad (4)$$

$$= w_4 \& \& (w_2 | w_3) \& w_1 \quad (5)$$

Other possibility could be the negation of widget. If W is all the amount of widgets presents in the representation of the device, (sum of w_i) then

$$!w_i = W - \{w_i\} \quad (6)$$

Another issue is to say that λ_4 is included in λ_1 (or λ_2, λ_3 since it would inheriting the property to equal sequences), where

$$\lambda_4 = w_1 \& (w_2 | w_3) \Rightarrow \lambda_1 = \lambda_4 \& \& w_4 \quad (7)$$

Inclusion helps in stating about the difference between a *sequence* and an *action* in HCI. In fact, an action is the result of an operation (|, &, &&, $\oplus$ as example) on severals sequences. This is kind of a meta-sequence (it can even be fully qualified by a widget based expression).

This is not the point in this paper to list all the possibility of Boole's algebra applied to sequences composed by widgets in HCI. Nevertheless, those examples are to stress out the complexity that can result from a simple widgets activation sequence. This is due to the fact that a sequence can possibly be written as numerous different variations and that all those variations are to be understood by the generic client, when it will come to read the XML document containing all the valid sequences of use of an instrument. Moreover, it is expected that widgets sequence normalization, as it had been described here, is not limited to device interactions integration into HCI but also to common fields of HCI.

4.3 Why Attempting to Normalize Widgets Sequences

It is true that the settlement of sequences would help in improving the normalization of utilization of a GUI, and thus enables to get concrete informations on the real use of a GUI towards the use that have been designed originally. For example, about cognitive walkthrough, developers could list expected sequences using the normalized way within a XML document, still respecting the DTD grammar associated.

On the other hand, widgets could integrate a behavior that would allow to keep the history of entered sequences during a cognitive walkthrough (whether the sequences is valid or not). In the end, a basic statistic would be the ratio of right sequences entered to invalid sequences used. This ratio represents an *estimation* of the usability of the device. This way, usability measurement would no longer be subjective but also a value associated to it (that is still proper to each user indeed). The drawback is that users tend to find new utilizations of a tool that was not foreseeable (but that actually works). This is because users tends to adapt themselves to a new tool by diverting it from original functionalities. That would means that the list of the allowed sequences would hardly never be complete before a cognitive walkthrough. This may implies whether iterations (insert the sequences discovered during cognitive walkthrough and then make another cognitive walkthrough) until getting a stable value of the ratio, or finding an average value by experimentation on how far are usually developers from foreseeable usages to real usage of HCI.
Nevertheless, this observation is far outside from the scope of this paper. Regarding eLaboratories, being able to normalize expected sequences would mean that a graph could be built on those sequences. Then, graph theory could possibly offer a lot. As example, since it is possible to store sequences it allows to replay that sequence later. Replaying a HCI sequence in eLaboratories (and surely in most others learning systems) is really something enhancing every steps of the learning mechanism. At first, this could be used to play *demonstration* sequences to show what the device is able to do. That is to say a remote device can be controlled by a pre-defined sequence (automation of processes). This avoid needing too much qualified person for demonstration and avoid mistakes and lapse of memory. Moreover, this can be used in *tutorials*: students would be able to reproduce HCI, watch for them over times without the need of the teacher to make the manipulation every time (please understand this is complementary to widely accepted teaching methods, not replacing them). Next, being able to identify invalid sequence would result in two benefits: prevent damage to devices (for example mechanical devices that move objects) and helps the user in understanding what he has done wrong. The system could possibly notice the user that the sequence he entered is a non-sense, explain why and may propose other sequences that he may had wanted to operate. In the end, for evaluation purpose, the teacher would be able to compare sequences entered to sequence(s) expected for the questions he ask. This should be a very significant value helping evaluation in hand-on approaches using HCI, where the final value/solution is not always the only criteria of notation (this may be too much all or nothing).

In the end, we can say that mapping of levels is no more relevant if it stays to the level of widgets. It is true that some widgets could not have a meaning without the presence of others. This compose group of widgets. However, this notion of group of widgets is not to be confused with the notion of sequence, that is why it will not be discussed here. Anyway, the reader should only be aware that widgets are actually listed by group in the XML file representing a device.

It is clear that this new kind of platform enables services that the laboratory is able to propose to all of these users. Moreover, this help in the acquisition of the device: it had been bought by several public laboratory in order to divide the (high) cost of such instruments.

4.4 Illustration with a Very Simple Device

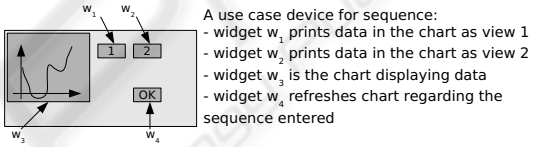

Figure 3: A simple instrument for the illustration of widgets and sequences.

Let's assume an instrument as described in figure 3. The legend associated presents basic functionalities: a chart widget displays data using view one or view two regarding the widgets being activated. A widget is held responsible for the action of validating the choice of a view. In those condition, the list of valid sequences is, assuming Γ_A is the sum of sequences for the instrument, each noted γ_i:

$$\begin{cases} \Gamma_A = \Sigma(\gamma_i) \\ \gamma_1 = w_1 \&\& w_3 \quad \text{(display using view 1)} \\ \gamma_2 = w_2 \&\& w_3 \quad \text{(display using view 2)} \\ \gamma_3 = (w_1 | w_2) \&\& w_3 \quad \text{(two curves displayed)} \\ \gamma_4 = w_4 \quad \text{(zoom the chart)} \end{cases} \tag{8}$$

Of course, the system (8) is equal to the system:

$$\begin{cases} \Gamma_B = \Gamma_A = \Sigma(\gamma_i) \\ \gamma_5 = (w_1 \oplus w_2) \&\& w_3 \quad \text{(1st view or/and 2nd view)} \\ \gamma_4 = w_4 \quad \text{(zoom the chart)} \end{cases} \tag{9}$$

4.5 Possible Implementation of a Sevice's Ontology

From those sequences, it is easy to build the corresponding XML files[3], which includes widgets and sequences (based on 9) (and is a preliminary version of the ontology of devices for the platform):

```
<?xml version="1.0" encoding="UTF-8"?>
<device>
  <name>Sample</name>
  <GUI>
    <composite-widget>
      <utilization-level>1</utilization-level>
      <widget>
        <ident>w1</ident>
        <w-type>IPushButton</w-type>
        <geometry>
          <x>150</x>
          <y>10</y>
          <width>15</width>
          <height>15</height>
        </geometry>
        <action-mapping>
          <func>setFirstView</func>
        </action-mapping>
      </widget>

      <widget>
    <!– widget w2 cut, same as w1 –>
      </widget>
      <widget>
    <!– widget w3 cut, same as w1,w2 –>
      </widget>

      <widget>
    <!– widget w4 cut, (nearly) same as w1,w2,3
–>
      </widget>

    </composite-widget>
  </GUI>

  <sequences>
    <widgets-used>
      <widget-ident>w1</widget-ident>
      <widget-ident>w2</widget-ident>
      <widget-ident>w3</widget-ident>
      <widget-ident>w4</widget-ident>
    </widgets-used>

    <sequence>
      <seq-ident>lambda-1</seq-ident>
      <content>
        <sequence>
          <ident>subseq-lambda-1</ident>
          <content>
            <widget>w1</widget>
            <operator>XOR</operator>
            <widget>w2</widget>
          </content>
        </sequence>
        <sequence-operator>THEN</sequence-
operator>
        <sequence>
          <ident>subseq-lambda-2</ident>
          <content>
            <widget>w3</widget>
          </content>
        </sequence>
      </content>
    </sequence>

    <sequence>
      <ident>lambda-2</ident>
      <content>
        <widget>w4</widget>
      </content>
    </sequence>
  </sequences>

</device>
```

[3]Present work aims at using Ontology Web Language (OWL) to get a standard compliant ontology, thus not present in this paper.

4.6 A Device Case Study: A Microwave Analyzer

- *Engineers* in industry (distant industrial processes),
- *Students* in schools (hand-on approaches on instruments)
- *Researchers* in laboratories (research purpose),
- *Researchers* in industrial laboratories (private supported research purpose)

Indeed, a Hyperfrequency Analyser helps in characterization in the field of high frequency for telecommunication meterials. The deployment of the generic client that load XML files is deployed via Java Web Start, which allows the automatic deployment of applications and manages versions automagically. This way, researchers are able to launch a distant application that represent their instrument, and are able to interact with the distant device. Of course, as described earlier, each command is relied on the asynchronous middleware (to implement "group awareness"). This way, all researchers are evolving in a group of manipulations, inside which they enter/leave when they launch/close their client application. For the time being, the sequence language have been used for the menu navigation. When a user enter a menu, it load

the corresponding submenu using sequences. Other sequences descriptions are in progress in order to supply a full sequenced HCI of the microwave analyzer.

5 CONCLUSION

What had been presented here is a generic approach regarding the studying and the construction of distant, collaborative and ”sequenced” HCI. The construction of the HCI in the presented eLaboratory management platform called eInstrumentation and is being supplied by several tools, specially a drawer and an image processing algorithm. Those two tools produced the same XML file (validated by a common DTD). Whereas each XML file are specific to each instrument, the DTD and the client interpreting the XML is a single generic application written in Java and using J2EE architectures (JonAs Application Server and JORAM Asynchronous Middleware). The application server and the asynchronous middleware connectors are supplied by the plateform widgets, implementing group awareness behavior. In the end, all the actions entered by users are compared in real time to valid sequences of usage of the instrument. That is to say users are guided to a correct usage of the device, mostly for learning and device protection purpose. It is highly possible that future works will consist in building an entire case study with its sequences and thus make experimentation on real hands-on courses. This would help in testing the robustness of the sequence abstraction and measuring the profit made by users when using such advanced HCI implementation. In addition, some design questions can be put and would need methods and experimentation interpretations (Reed, 2005). Others reflexions could also raise from GUI plasticity (which does not appear to be a real problem for asynchronous middleware in Java (Yoneki, 2003), and platform accessibility. There is also the study of the possibility to couple ”felt-life” (3.1) with sequences (for now these are two different things implemented). Its sound possible that a mapping should exist between ”felt-life” and sequences regarding the timing on which felt-life should be based on (it shall not play a multimedia content widget-centric, but when users reach a certain advance in a (sub)sequence). This way the multimedia content would not be played each time the widget is being actioned but only in a certain ”sequence context” of utilization.

REFERENCES

Borchers, J. (2001). A pattern approach to interaction design. *Symposium on Designing Interactive Systems*, 15(4):369 – 378.

Dery-Pinna, A., Fierstone, J., and Picard, E. (2003). Component model and programming: A first step to manage human computer interaction adaptation. *Lecture Notes in Computer Science*, 2795:456 – 460.

Fong, J., Ng, M., Kwan, I., and Tam, M. (2004). Effective e-learning by use of hci and web-based workflow approach. *Lecture Notes in Computer Science*, 3026:271 – 286.

Guss, S. (2003). Interface metaphors and web-based learning. *Lecture Notes in Computer Science*, 2783:168–179.

Gutwin, C., Penner, R., and Schneider, K. (2004). Group awareness in distributed software development. In *2004 ACM conference on Computer supported cooperative work*, pages 72–81, Chicago, Illinois.

http://jonas.objectweb.org/ (2005). Jonas: Java open aplication server.

http://joram.objectweb.org/ (2005). Joram: Java open reliable asynchronous middleware.

http://www.gimp.org/ (1995-2005). Gnu image manipulation program.

Jr., N. C. R., Sharda, R., and Lucca, J. (2005). Computer-supported collaborative learning requiring immersive presence (csclip): An introduction. *Information Systems Frontiers*, 7(1):5–12.

Lukosch, S. and Roth, J. (2001). Reusing single-user applications to create multi-user internet applications. *Innovative Internet Computing Systems*, LNCS 2060:79–90.

McCarthy, J. and Wright, P. (2004). Putting felt-life at the centre of human-computer interaction. In *Reflective HCI Workshop*.

M.Roseman and Greenberg, S. (1996). Building real-time groupware with groupkit, a groupware toolkit.. *Transactions on Computer-Human Interaction*, 3:66–106.

Reed, D. (2005). Learning from loseables, an exercice in critical reflection. *IFIP International Federation for Information Processing*, 178:121.

Sebe1, N., Lew, M. S., and Huang, T. S. (2004). The state-of-the-art in human-computer interaction. *Lecture Notes in Computer Science*, 3058:1–6.

Soloway, E., Guzdial, M., and Hay, K. (1994). Learner-centered design: the challenge for hci in the 21st century. *Interactions*, 1:36–48.

Stefik, M. (1987). Wysiwis revised: early experiences with multiuser interfaces. *Transactions on Office Information Systems*, 5:147–167.

Wharton, C., Rieman, J., Lewis, C., and Polson, P. (1994). The cognitive walkthrough method: a practitioner’s guide. *Usability inspection methods*, New York: John Wiley & Sons:105 – 140.

Yoneki, E. (2003). Mobile applications with a middleware system in publish subscribe paradigm. In *3rd Workshop on Applications and Services in Wireless Networks*.