Ingrid, Geraldine and Amidala – Competent Agents for Pen Gestures Interactivity

Khaireel A. Mohamed and Thomas Ottmann

Institut für Informatik, Albert-Ludwigs-Universität Freiburg
Georges-Köhler-Allee, 79110 Freiburg, Germany
{khaireel, ottmann}@informatik.uni-freiburg.de

Abstract. There are several factors that make delivering lessons on digital boards uptight. Among them, focus on the subject is lost to the time taken for tutors to acquaint with the technology infrastructure, and that students' concentration on the tutor's chain of thoughts are interrupted by watching them running up and down the length of the classroom hunting for the mouse (or keyboard) to perform commands to the UI. Based on intelligent agent interfaces, this paper presents an innovative way to ease the above drawbacks and assist tutors by proactively anticipating their commands, through ubiquitously analysing their ink-traces. The conceptual breakdown of ideas and our basic design principles of the generic interactivity framework are highlighted in detail with two application examples. We also show that it is advantageous to utilise goal-oriented agents in domains similar to eLearning, where handling digital ink is an important aspect.

Keywords: Digital pen gestures, InkML, interface agents, eLearning
CR Classification: G.1.10, H.3.3, I.7.5

1 Introduction

A paradigm that emphasises adaptivity, cooperation, proactiveness and autonomy – in both design and run times – for abstruse software development proposes the engagement of *agents*. For when programs work with humans on a task in an interactive situation, the interaction is often two-way. Each side, acting as *peers*, are occupied with their own tasks to achieve their goals, while at the same time *proactively* helping one another as the work towards solving the overall problem progresses.

Woolridge and Viano et al. [19, 18] term these programs as "interface agents" and describe them as taking initiatives, rather than waiting for explicit commands from users, which makes these semi-autonomous agents seemingly to cooperate with their users quite naturally. These agents perceive their user-input environment through predefined sensors, and then act upon that environment through effectors – by directing automatic assumptions to the application program or as suggestions for the user to make further decisions. Their means of

acclimatic reasoning and goal-oriented aims make them useful assistants when processing in the background. Involving their learning capabilities in complex algorithms will prove their invaluable presence in most interactive user domains. Through the agent communication language (ACL), the agents can directly interact with each other while hiding the details of their internal workings to result in a society of agents that tackle problems no individual agents could [8].

To date, object-oriented routines (OOR), used in popular programming languages, have no doubt conceived many successful and practical initiatives to make inputs from digital pens more robust in invoking extra capabilities for the interface between users and software programs. Most of these initiatives, which correspond directly to having extra command buttons mounted onto the current user interface (UI) and incorporating the use of gestures to implore additional functionalities [9, 6, 15], are hard-wired into the toolkits for use only in specified scenarios for the analysation of special problems. The OOR controls every aspect of the programming details and it needs to cater to all possible facets of the users' behaviours and react in accordance to the overall plan for the domain it is operating in.

In interactive domains, it is not very practical to program the OORs to anticipate every environmental changes that are brought about by the users from the applications they are serving. These applications range for a wide variety of purposes - spanning from the simple and personalised portable programs on tablet PCs, to the more complex codes intended for scenarios such as teaching in a lecture hall and conducting live corporate meetings across continents on huge digital boards. We seek to make this interaction between users and applications easy and tolerable enough for anyone to use. Since studies have shown that "gesturing" to the interface to invoke commands (during a presentation, for example) is much less of a hassle than having to walk over to a keyboard or mouse to do the same thing [3, 12], we can then prove that the 'agency paradigm' above not only supports the backbone processes practically, but also accord them in a general manner that is applicable to an array of proverbial domains.

For in order to efficiently interpret a series of digital ink traces as command gestures interacting with the generic environment (while proactively supporting users' input signals), we require a separate program module to seriously analyse those traces before concluding the validity of the command. Part of our solution recommends the adaptive and abstract methodologies that are vibrant enough for incorporation with new technologies and elements of the modern eLearning environment. These includes the composition of transparent agent-based systems communicating between the physical UIs and the background computers, rather than simply involving OORs.

In this paper, we combine the techniques in interpreting and reacting to digital pen gestures, with our inscription of three autonomous agents – so called Ingrid, Geraldine and Amidala – to handle these tasks competently. We begin in the next section by stating our case for our choice of utilising interface agents for the overall problem solving approach. The next three sections after that introduces the canvas environment, discusses the design of our three agents, and

their integration to a system, respectively. Section 6 illustrates our findings of the agents' behaviours through two controlled scenarios, and section 7 puts forward a proposal of increasing the number of modalities the agents can serve.

2 Agents for Interactive Domains

There are many literary discussions on topics that formalise the differences (as well as similarities) between objects, agents, and expert systems. But here, we will highlight only those that are related to our work in order to justify our choice of utilising interface agents in interactive eLearning domains.

2.1 Objects, Agents and Expert Systems

Woolridge [19] defines *objects* as computational entities that encapsulate some program state – and that objects are able to perform actions or methods on this state and communicate by message passing. Although, by this definition, objects can be thought of exhibiting autonomy over their states, they still do not have control over their own behaviours. Agents on the other hand, embody a stronger notion of autonomy than objects, in that they are able to decide for themselves whether or not to perform an action upon request from another agent. Due to this nature of their flexible behaviours, agents are more suitably made for the reactive, proactive, and social interactions within their environments. This is seen as an advantage for our domain specification, as it is a practical way of providing structural supports so that both users and computers can communicate proactively.

A comment from Booch on *active* objects comes close to the definition of agents, in which agents are not only autonomous, but are assumed to have their own threads of control [2]. This spells out an important difference between agents and concurrent OORs; agents are constructed for goals special only to each of them, and not as object entities that are simply tasked for parallel execution of a common routine. With agents, a range of simultaneously independent and diversified capabilities are at the disposal of any application program that chooses to engage in one.

Unlike expert systems, agents are not disembodied. They are clearly hooked up to environments where they can act upon. Expert systems, on the other hand, only behave as 'middlemen' through users' inputs without having to directly interact with the environment, and then giving their feedback or advise to a third party.

Hence, apart from the fact that multiagent systems are indeed interoperable – in the sense that they are easily pluggable into well-defined systems to take advantage of all their agential properties – we can also enjoy their societal-expandability. For instance, our work started off by attempting to classify digital ink traces as recognizable gesture-classes [11]. By breaking down the problem, we realised the need for a purely reactive agent to handle and track ink traces on the UI environment for efficient accessibility to the entire ink information (we

named her Ingrid). Another agent is required to percept the same environment, extract features from a current trace and match them to a library of trained gesture-classes (that's Geraldine). With just these two in the society, we are able to put together a good recogniser system, capable of interpreting gestures in any ink domain. But as the library of recognised gestures grow, Geraldine's output list of probable gesture-classes needed more attention in order to assist users more effectively. Consequently, we expanded the current society to include an additional agent (Amidala) whose job is to decide convincingly which of Geraldine's list is to become (or not to become) the next interpreted gesture.

2.2 Cooperative Distributed Problem Solving

The distributive aspect of this problem's decomposition was done after conceptualising our long-term plans to accomplish "communicative protocols" through gestures [10]. We believe that by solving smaller sub-problems in an ordered hierarchical manner, the overall global objective of our bigger project can be realised rather systematically. The corollaries of which are simple by-products that are surprisingly usable outside the scope of our experimental domain.

We described briefly in the previous sub-section of how we came about to having Ingrid, Geraldine and Amidala. This problem decomposition through agents is the first stage of the cooperative distributed problem solving (CDPS) technique highlighted by Durfee et al. [5]. Figure 1 illustrates the complete three stages of the CDPS technique for our work.

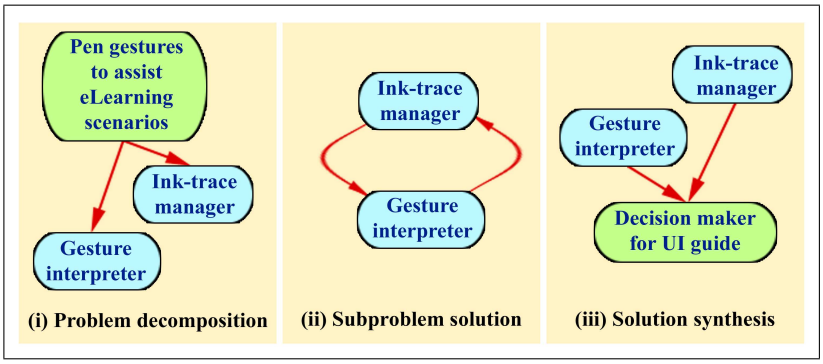

Fig. 1. The three stages of CDPS.

In the *subproblem solution* stage, the two agents depicted in Figure 1 may inter-share their information to help one another out – that is if each has information that may be useful to the other. For example, Geraldine may wish to refer to a trace data scribed previously at time $t - n$, or Ingrid may inquire about the density feature of the current trace for part of its data management criteria.

The *solution synthesis* stage combines all sub-solutions into an overall solution. As we mentioned before, Amidala either strengthens or weakens the stance of Geraldine's list of hypothetical gesture-classes before shortlisting it for her verdict. Should it not be a gesture, a reference to the complete trace information is conveniently provided by Ingrid.

3 Trace Canvas as Environment

Modelling this work for the agent-oriented programming (AOP) criterion, to ensure the compatibility of our three semi-autonomous but pervasive agents, we must include the trace canvas (where digital ink is captured and rendered) as the *environment* to be percepted. From the standards laid by Russell and Norvig on environment classifications [17], we consider the trace canvas to be all of the following:

- **Continuous** (rather than discrete), as there are uncountably many states this environment can take depending on the variety of non-uniform traces (or gestures) a user can input;
- **Accessible** (rather than inaccessible), as any agents coupled to this environment are able to obtain complete, accurate and up-to-date information from the rendered traces;
- **Deterministic** (rather than non-deterministic), as any actions by the agents are guaranteed effect on the environment, with no uncertainty about the resultant state of the environment; and
- **Dynamic** (rather than static), as apart from the agents, the foreground application and users are also acting on the environment that results in the change of state which is beyond the control of any of the agents.

4 Designing the Agents

The grounds for the AOP criterion suggests that we program our agents directly in the terms of *mentalistic* notions, through the belief-desire-intention (BDI) architecture, based on the agents' set of unique properties. BDI offers explicitly represented data structures that are somewhat loosely corresponding to the mentalistic notions [19]. This architecture is illustrated in Figure 2. Putting it simply, *belief* is what the agent senses, *desire* refers to the agent's objectives (or goals), and *intention* is the course of action the agent takes.

4.1 Ink Agent (Ingrid)

As a 'purely reactive' agent, Ingrid does not refer to her past actions when processing ink information from the environment.

Belief: This agent converts all sampled trace-points into standardised ink representations. W3C's version of InkML [16] is implemented here, stamping

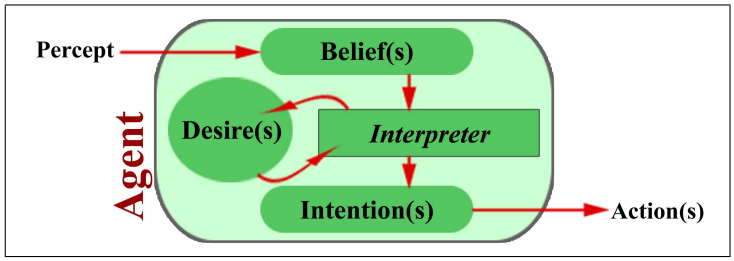

Fig. 2. Mentalistic notion through BDI architecture.

discrete time values to the sets of x-y coordinates perceived from the sketching environment in order of appearance

Desire(s): Ingrid's main task is to ensure that her entire ink-trace database is always up-to-date and correctly synchronised with the environment. On top of that, she is also to maintain as few *trace-groups* as possible by looking at their spatial locations and some temporal feature-constraints.

Intention: The resulting InkML from Ingrid's database, which includes parameters such as the channel and pen information, is made available for any other agents or programs that might be interested.

4.2 Gesture Agent (Geraldine)

The gesture agent is a 'standard' agent that perceives the environment and does automatic training or recognition of gesture-classes.

Belief: Geraldine reacts to her percepted information with a filter that removes unwarranted noise and then rationalises the data set to feed them as parameters to its training and recognition algorithms. Here, we implemented Rubine's linear discriminator [14], which utilises *feature* vectors extracted from rationalised traces, that produces optimal classifiers, on a per-gesture-class distribution. Rubine's method trains the classifiers from given sets of proper input gestures, and works the recognition mechanism by evaluating new feature vectors to the trained classifiers in accordance to the gesture classes. The output is a set of classified gesture classes ordered according to their attached *recognised*-probability values.

Desire(s): Apart from maintaining the best values for the optimal classifiers to effectively balance the linear-cognition scale, Geraldine also ensures that all calculations in the linear discriminator are error-free. As the bulk of the algorithm entails complex matrix manipulations, variable values that are 'out of range' have the tendency of generating 'NaN error' when the matrices are inverted [10]. Hence it is crucial for the agent to learn these erroneous values early, and refuse to calculate them if necessary.

Intention: The gesture agent proffers her findings of a list of *recognised*-probability values to the set of predetermined gesture classes (hypotheses) ar-

ranged in descending order. In the case of a calculation error, or a refusal by the agent to perform any calculations, that list is made empty.

4.3 Decision Agent (Amidala)

Our decision agent is also a 'standard' agent. Amidala's only difference from the previous two is that this agent percepts her information from Ingrid's and Geraldine's intentions instead of the pre-defined environment.

Belief: Amidala places a higher priority to her gesture-percepts than her ink-percepts. Within herself, the agent retains a threshold *recognised*-probability value which she uses as a comparator for accepting or rejecting the list of hypothetical gesture-percepts. If the highest value on that list exceeds this threshold, then a decision is made to upgrade the trace to a gesture, identified by its unique class-type. Otherwise, the trace remains a trace with all its properties intact within the ink-percept. This is also the case if the gesture-percept contains an empty list.

Desire(s): The agent makes use of her ink-percepts as a feedback for determining its threshold probability value, by informing itself of its previous decision. The goal here is to optimise the threshold value so that good and correctly intended gestures are not rejected, and that ambiguous gestures can be resolved. Rubine [14] recommended using the Mahalanobis distance to determine the standard deviation that a gesture is away from its chosen class-type. Based on this feedback loop, the agent's threshold value is either strengthened with correctly made decisions, or weakened by the wrong ones.

Intention: Amidala makes available a two-tuple information about the input trace – [*InkML representation, Gesture class-type*]. Agents or applications interpreting this information will know that the trace is not a gesture if the class-type is set to "Not A Gesture".

5 System Integration: Plug and Play

We foresee, that in the final stages of our work, the developed agency society are to be easily incorporated into the computer's main operating system for generally and ubiquitously assisting ink domains used for interactive eLearning scenarios. Due to the autonomous nature of the agents, this background integration can be done in two distinct ways that will not affect the main flows of other programs running in the foreground. One way is to create threads and adapt the agents to work from the interrupt timings generated by the system. The other is to use Java's *Observable-Observer* paradigm as depicted in Figure 3.

This paradigm is possible because the interaction process between users and the program interfaces are completely discrete in nature – there is time enough for the system to react between every pen-up and pen-down events. Our previous experiments show that the overall time taken by the agents to calculate and classify each incoming trace, to run routines for achieving each agent's desire(s),

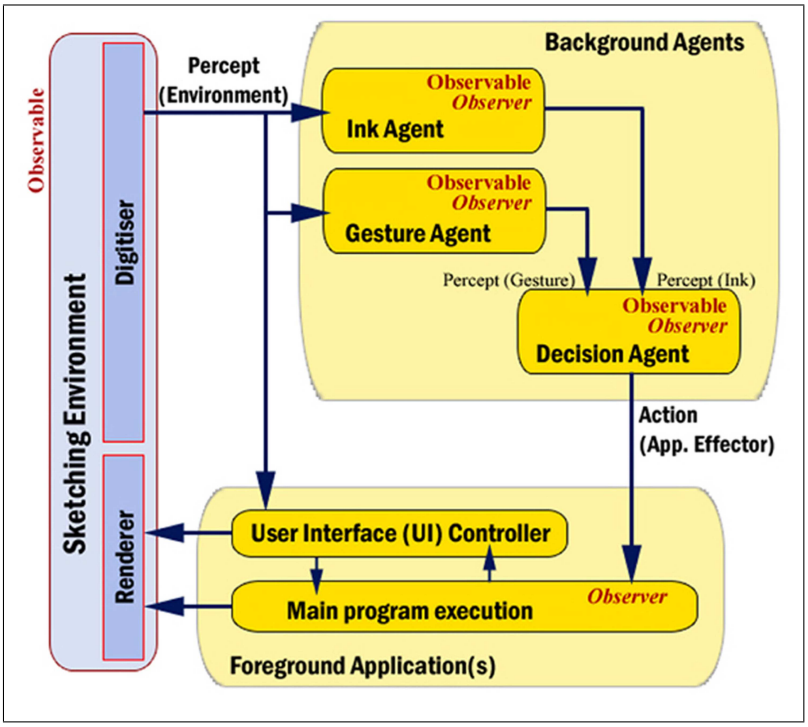

Fig. 3. Overall integrated system architecture.

and to communicate their findings to the foreground application is indeed negligible [11]. The paradigm also emphasises the simplicity of attaching the agents to any programs wanting their proactive services. The only catch is to declare the canvas (sketching) environment as Observable and the running foreground applications as Observers. We illustrate this in the next section in more details.

The agents react whenever there is a change occurring within the canvas environment – either affected by users with ink-traces, or by the foreground applications themselves. Those traces are then processed by Ingrid and Geraldine to formalise the data representation and to extract gestural features, respectively, before making the refined information available to any other observers (both agents and application programs) down the chain of the paradigm. While this is going on, the two agents continue to analyse their own sets of data to complete their respective proactive routines. Amidala then reacts to both information made available by the two agents, deciding if the newly scribed trace on the visual board is a gesture, and if so, what kind.

6 Behavioural Findings on Interactive Domains

In the course of testing out the versatility and compatibility of the "pluggable" structure of our agents, we created two test programs to evaluate the coherence and coordination of the agency. In addition, we also noted each agent's sphere of influence in the following sub-sections.

By *coherence*, we mean the measure of how well the overall multiagent system behaves as a unit in terms of solution quality (correctness of assistance), and the efficiency of resource usage. The degree of synchronisation between the agents and the foreground applications determines the level of *coordination*. The *sphere of influence* the agents have on the developed programs contributes to the factual hypothesis of this paper.

6.1 The Whiteboard Assistant

Interactive Scenario Description As naturally as possible, we simulated the trace canvas environment to act *exactly* like the digital version of whiteboards used in lectures and tutorial classes. Without adding too much details about the pen attributes or the canvas' rendering properties, we concentrated on the issues of our agents assisting tutors delivering "stress-free" lessons. Minimising the trips to pick up board erasers is the main objective of this evaluation, and that substituting preferred gestures to the button clicking of the digital pen is the other aspect we want to observe.

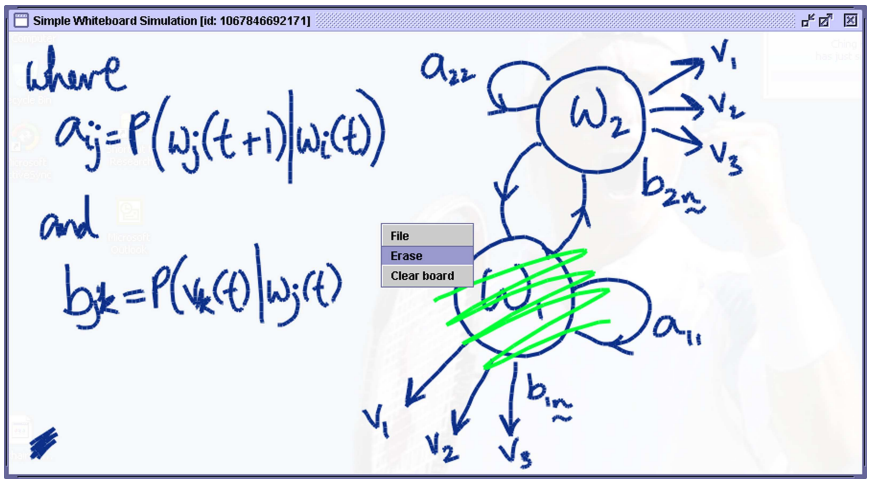

Fig. 4. Simple interactive whiteboard assistant simulation. A popup menu appears near the gesture to confirm the agents' proactive anticipation.

Coherence This scenario is simple enough so as not to require too many single-gesture-operated commands. In fact, we found that only three is enough for the primitive whiteboard simulation; the 'erase' gesture – for erasing partial traces, the 'clear' gesture – for clearing the entire board, and the 'file' gesture – for invoking file menu commands.

The agents are always attentive to the inputs on the whiteboard canvas. Amidala recognises whenever a trace is expected to be a gesture, or dismisses Geraldine's suggested hypotheses when the trace is to remain as ink. The foreground program reacts to Amidala's verdicts and churns out a popup menu with the gesture's generic name whenever appropriate, see Figure 4. Our society of agents are able to proactively anticipate about 97% of all true positives from the users' intentional traces (Figure 5), through the Pereira et al.'s *expectation list* technique of handling ambiguity and errors in traces [13]. Ingrid's resourceful management of the complete ink-trace data proved useful in reacting efficiently and correctly to the 'erase' gestures, and her InkML data representation keeps the file access procedures standardised.

Whiteboard assistant - Coherence test				
User's intention	**Ink traces: 798**		**Gestures: 51**	
Agency anticipation	**788**	10	**49**	2
Percentage %	**98.75%**	1.25%	**96.08%**	3.92%
	Correct	*Incorrect*	*Correct*	*Incorrect*

Fig. 5. Tabulated coherence of the Whiteboard assistant, showing the relationship between a user's intention and the agency's collaborative anticipation after 10 whiteboard sessions. In total, users scribed 849 traces on the whiteboard, of which 798 were intended as ink traces and 51 were supposedly gestures. The agency anticipated 788 of the intended ink traces and 49 of the intended gestures correctly. This averages to 97.42% 'true positive' assistance.

Coordination Responses from this whiteboard program are immediate to users. This means that both the foreground application and the background interface agents handle ink information in a good and coordinated manner. It does not give rise to any confusion for the user when interacting with the program. On average, it takes less than 20 msec for the agents to process a trace and then synchronise with the Observer program.

Sphere of Influence The coherent and coordinated performance of the three agents in this simple scenario shows that they are indeed "peers" to the users. They can anticipate users' intentions and assist them (on the spot) without having the users detour to pulling down system menus – which may be located

at the other end of the long screen. We also note the ease of using the digital pen without having to figure out its click-buttons or turning it around to use the in-built eraser function.

6.2 The Presentation Assistant

Interactive Scenario Description Imagine delivering a presentation in front of important clients, where you feel it more comfortable standing near the digital screen rather than staying anchored at a lectern. Underlining, highlighting and illustrating directly on the screen on core concepts during the presentation make you appear naturally composed, and, to a certain extent, professional. What's more, the digital screen is seen to respond proactively to your gestures when you progress forwards (or backwards) and enlarging certain portions of the screen to emphasise some of your facts. All these carried out without having to walk over to the computer console (where the mouse and keyboard are) to call out menus for performing exactly the same tasks.

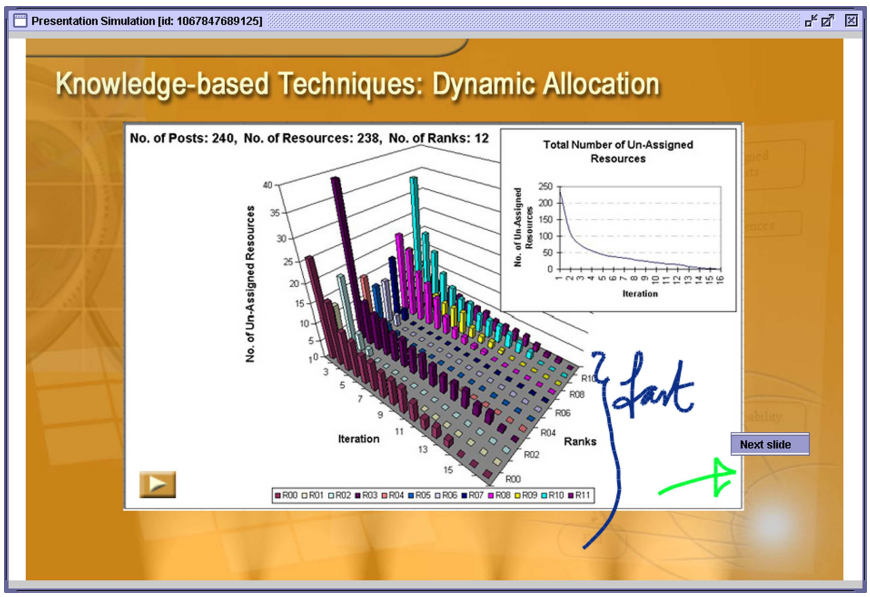

Fig. 6. Presentation assistant simulation. A popup menu inquires if the user would want to advance to the "next slide".

Coherence Similar to the previous whiteboard scenario, this presentation interface program also allows users to write and gesture directly on screen. The

only difference is that slides are used in place of the clean whiteboard slate, so that this program will need to additionally manage 'slides' data on top of the ink-trace information.

Again, the agents pay good attention to the users' inputs, with Ingrid, Geraldine and Amidala all proactively helping to assist in the delivery of the presentation, see Figure 6. Albeit a small percentage of false positives, which suggests the agents' anticipations as too eager, we recorded an average of 96% true positives from all intended gestures (Figure 7).

Presentation assistant - Coherence test				
User's intention	Ink traces: 85		Gestures: 102	
Agency anticipation	81	4	99	3
Percentage %	95.29%	4.71%	97.06%	2.94%
	Correct	*Incorrect*	*Correct*	*Incorrect*

Fig. 7. Tabulated coherence of the Presentation assistant, showing the relationship between a user's intention and the agency's collaborative anticipation after 10 presentation sessions. In total, users scribed 187 traces on the slides, of which 85 were intended as ink traces and 102 were supposedly gestures. The agency anticipated 81 of the intended ink traces and 99 of the intended gestures correctly. This averages to 96.18% 'true positive' assistance.

Coordination and Sphere of Influence The good and uninterrupted flow of presentation would be a measure of success users look for. The agents' ability to pervasively coordinate their desires through their intentions, offers an advantage to presenters who are comfortable with this approach. Users' own decisions are simplified whenever they gesture their intentions to the program's interface. This directly contributes to the societal learning within the agency.

6.3 Overall Performance

Our data show that users tend to write more on the whiteboard than on the presentation slides; 81.95% of total scribes on the whiteboard as compared to 18.05% on the presentation slides. This is an indication that different scenarios have different application types, and that they may differ on their dependency towards gesticulation services. The pie chart in Figure 8 shows the combined anticipation of the agency in both scenarios, and that the overall coherence is above 98%.

6.4 Threshold for Accepting Gestures

As the final decision maker down the chain of the *Observable-Observer* paradigm, it is imperative that Amidala is able to provide a convincing and dependable

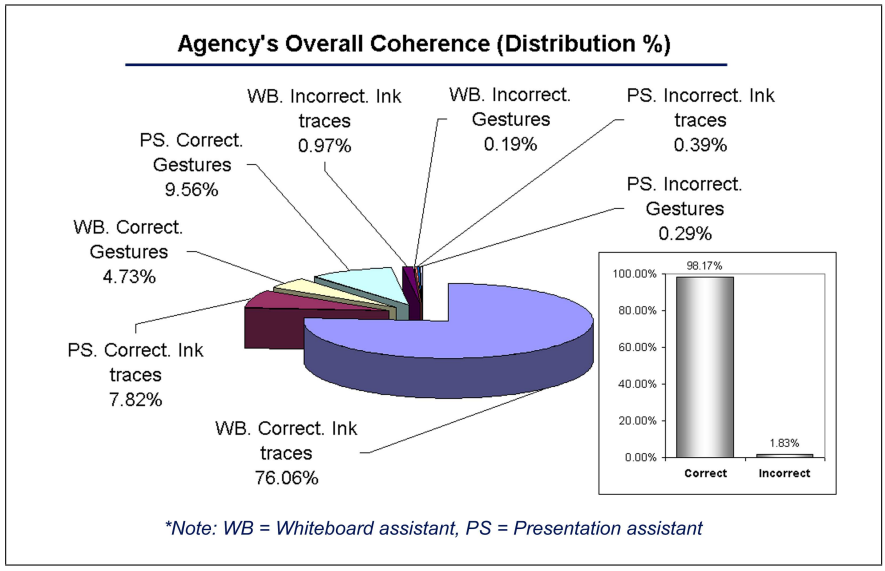

Fig. 8. Agency's overall coherence evaluated from the two test scenarios.

verdict. Her decisions are what makes the entire agency coherent, and is reflected previously in Figures 5 and 7. We mentioned in section 4.3, that Amidala maintains a threshold *recognised*-probability value that is used as a comparator for accepting or rejecting the list of hypothetical gesture-percepts from Geraldine. Figure 9 illustrates Amidala's learning curve of this probability value. After handling 1036 *operations*, that is the process of determining the intended entity of an ink-trace, the graph shows the strictness of the agent in allowing only probability values that are above 0.95478 to 'pass' the shortlisting procedure test.

6.5 HCI Issues

HCI experts would insist on the complete testing of any new software or system interfaces in order to conclude good standards for menu placements, timing effects, users' adaptability, etc. But since our agents are the main point of discussion in this paper, and that at this stage, their functions are only to assist users with regards to their gesticulation styles, we will not address the matter for now. However, it is sufficient to know that they have met the standards set by Woolridge and Jennings [20] as outlined by their coherence, coordination and sphere of influence in the previous sub-sections.

7 Expanding for Multimodality

Construing eLearning techniques, particularly in lecture deliveries, to just elemental pen and board interactivity may seem enough – for now. However, as

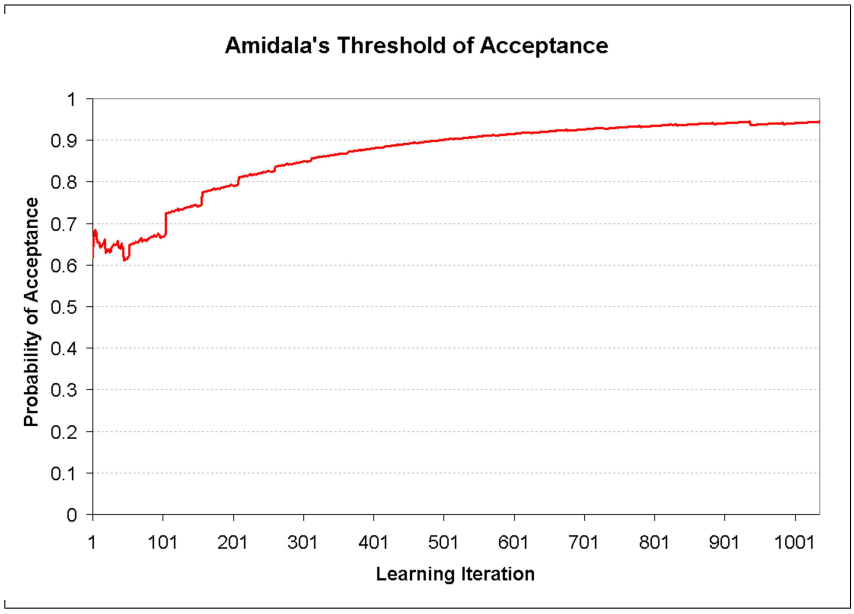

Fig. 9. The graph of Amidala's learning threshold, and after in handling 1036 operations, the final threshold *recognised*-probability value is set to 0.95478.

advances in very accurate speech recognition have attributed significantly to a change of mindset in engaging "audio-aids" within domains such as telecommunication, eCommerce, gaming, robotics, and vehicular navigation, the eLearning scene too has constructively been able to make its own progresses [4, 7].

As a matter of befitting this paper's aim, that is to provide a transparent and proactive assistance to users utilising softwares for teaching in classrooms equipped with digital boards, we plan to include the *sound* modal to integrate with the current gesticulation services. Aspects of synchronisability and data management of this new entity is seen to be adaptable enough to participate in our current system architecture. Expanding the society for an additional 'sound' agent to percept a second environment (the audio channel) can be incorporated to coordinate with Ingrid, Geraldine and Amidala on the same *Observable-Observer* paradigm.

Figure 10 illustrates this in further detail – where attached within the sound agent are modules that parse and interpret words spoken by the user, and modules that match them to a library of recognised words. The sound agent will forward its action in real time to Amidala (or any other interested Observers) every time there is hit to its *keywords* list. Since we have implemented InkML's specifications to store all our resultant ink-traces, this same standard can also accommodate data from other input modals, particularly the sound modal for our case, as defined by the W3C Multimodal Interaction Framework [1].

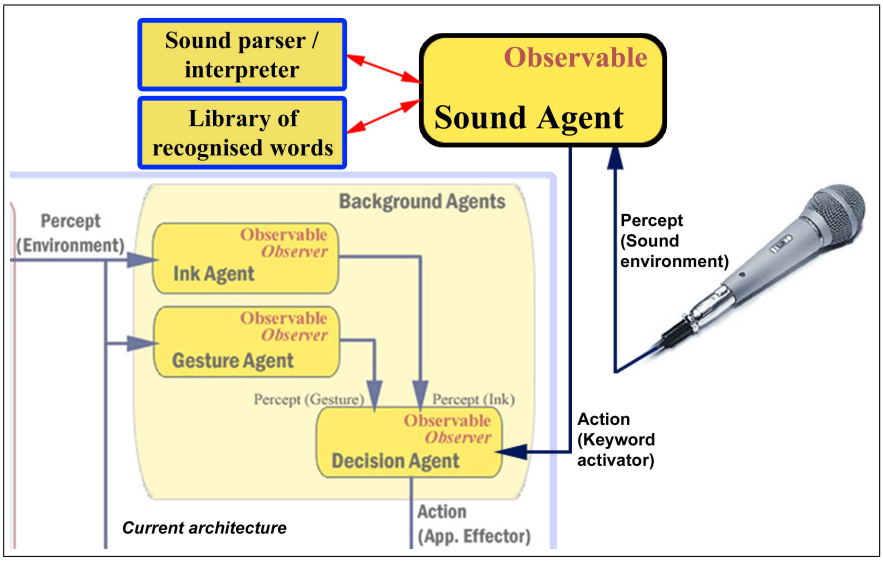

Fig. 10. Including a 'sound' agent to expand the current societal architecture.

Now, through their reliable pen gestures and accurate audio commands, users' writings, gesticulations and voices can be scrutinised deeper by the agency to pervasively anticipate their intentions, and to have the system interact with the users even more convincingly. In the longer run, the agency may even put a finger on identifying the tutors from their various writing styles (by learning from their lesson deliveries), and perhaps accord specialised interpreter libraries for them. All these without the need for new users to learn the entire procedural complexities to effectively use the system.

8 Conclusion

We have shown through two principle examples, that having interface agents ubiquitously monitoring input gestures to assist in interactive eLearning situations are helpful in making presenters feel at ease when delivering their materials. The conceptual "plug and play" design of our agents are indeed grounds for making other domains, apart from eLearning, benefit the gestural interactivity offered through this agency.

In contrast to the OORs and other expert systems, providing proactive assistance to users for a wide range of applications are both coherent and coordinated with interface agents. Their sphere of influence show that handling interactive situations is a task achievable by the 'agential paradigm' in practically supporting the backbone processes necessary to sustain the continuous flow of foreground applications.

Acknowledgement

This research is funded by the Deutschen Forschungsgemeinschaft (DFG) as part of the research initiative for the "Algorithmen und Datenstrukturen für ausgewählte diskrete Probleme (DFG-Projekt Ot64/8-3)".

References

1. Bodell, M., Johnston, M., Kumar, S., Potter, S., and Waters, K. W3C multimodal interaction framework [online], Available: http://www.w3.org/tr/mmi-framework/ [Accessed 29 October 2003]. *W3C NOTE 06 May 2003* (May 2003).
2. Booch, G. *Object-Oriented Analysis and Design*, 2nd ed. Addison-Wesley, 1994.
3. Cutler, R., and Turk, M. View-based interpretation of real-time optical flow for gesture recognition. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition* (1998), IEEE, pp. 416 –421.
4. Datta, A., and Ottmann, T. Towards a virtual university. *Journal of Universal Computer Science 7*, 10 (October 2001), 870–885.
5. Durfee, E., Lesser, V., and Corkill, D. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering KDE-1*, 1 (March 1989), 63–83.
6. Hong, J. I., and Landay, J. A. Satin: a toolkit for informal ink-based applications. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology* (2000), ACM Press, pp. 63–72.
7. Hürst, W., and Venkata, L. S. K. A. Interface issues for accessing and skimming speech documents in context with recorded lectures and presentations. In *Proceedings of HCI International 2003, 10th International Conference on Human-Computer Interaction* (June 2003), Crete University Press, pp. 656–660.
8. Labrou, Y., Finin, T., and Peng, Y. Agent communication languages: the current landscape. *Intelligent Systems, IEEE 14*, 2 (April 1999), 45–52.
9. Lin, J., Newman, M. W., Hong, J. I., and Landay, J. A. Denim: Finding a tighter fit between tools and practice for web site design. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems* (2000), ACM Press, pp. 510–517.
10. Mohamed, K. A. Recognising sketch objects as intended valid gestures via autonomous agents. *Algorithms and Data Structures, Albert-Ludwigs-Universität Freiburg – Seminarial Report* (June 2003).
11. Mohamed, K. A., and Ottmann, T. Fast interpretation of pen gestures with competent agents. In *Proceedings of the Second International Conference on Computational Intelligence, Robotics and Autonomous Systems* (December 2003), IEEE press, p. To appear.
12. Mynatt, E. D., Igarashi, T., Edwards, W. K., and LaMarca, A. Flatland: New dimensions in office whiteboards. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1999), ACM Press, pp. 346–353.
13. Pereira, J. P., Fonseca, M. J., and Jorge, J. A. Handling ambiguity and errors: Visual languages for calligraphic interaction. In *Proceedings of the 14th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAP'01)* (2001), IEEE Computer Society, pp. 312–319.
14. Rubine, D. Specifying gestures by example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques* (1991), ACM Press, pp. 329–337.

15. Rubine, D. Combining gestures and direct manipulation. In *Conference Proceedings on Human Factors in Computing Systems* (1992), ACM Press, pp. 659–660.
16. Russell, G., Chee, Y.-M., Seni, G., Yaeger, L., Tremblay, C., Franke, K., Madhvanath, S., and Froumentin, M. Ink markup language [online], Available: http://www.w3.org/tr/inkml/ [Accessed 14 October 2003]. *W3C Working Draft* (August 2003).
17. Russell, S. J., and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, USA, 1995.
18. Viano, G., Parodi, A., Alty, J., Khalil, C., Angulo, I., Biglino, D., Crampes, M., Vaudry, C., Daurensan, V., and Lachaud, P. Adaptive user interface for process control based on multi-agent approach. In *Proceedings of the working conference on Advanced visual interfaces* (2000), ACM Press, pp. 201–204.
19. Woolridge, M. J. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.
20. Woolridge, M. J., and Jennings, N. R. Intelligent agents: Theory and practice. *The Knowledge Engineering Review 10*, 2 (1995), 115–152.