THE DEVELOPMENT OF A VIRTUAL STUDIO ENVIRONMENT TO SUPPORT COLLABORATIVE BUILDING DESIGN

Y.Z. Chen, I. Frame,
Building Performance Research Unit,
Anglia Polytechnic University,
Victoria Road South,
Chelmsford CM1 1LL, UK

T.W. Maver,
ABACUS,
University of Strathclyde,
131 Rottenrow,
Glasgow G4 0NG, UK

Abstract

This paper describes the development of a *virtual studio environment* to support collaborative working in the domain of building design. By applying and extending the real-world design studio model within the *Internet*-based distributed computing environments, the *virtual studio* concept has been refined as computerised settings, which integrate both the dispersed human designers and the distributed CAD applications. The hope is to achieve the similar effect as physical co-presence while providing extra advantages such as the support for automatic communication archiving and being less obtrusive than sharing a physical office.

Like its real-world counterpart (which usually consists of the office, desks, file cabinets, instruments etc), such a virtual studio consists of the several major components, including a multi-user graphical user interface displaying the shared virtual workspace on each designer's workstation, distributed multi-media databases and CAD tools for processing the domain tasks, and rich human-human interaction facilities supporting a variety of communication modes. Advance distributed object computing technologies (OMG CORBA in particular) have been adopted for modelling and implementing the distributed systems, W3 (world-wide-web) technologies have also been exploited for constructing the distributed multi-media databases and an image communication kit.

In contrast to the traditional CAD integration which is usually focused solely on the well-structured technical part of the product and process, the described research advocates a human-centred systems development strategy in which design is first of all taken as a process of social construction.

Key words

building design, collaboration, awareness, human-human interaction, virtual design studio, virtual studio environment, distributed computing

1. Introduction

Recognising that in the building industry the collaborators constitute a *virtual organisation* which exists throughout the life of the project then disbanding, virtual design studio has been coined in MIT to refer to "networked facilities that provide participants with access to the virtual organisation's databases and computational resources, messaging and data exchange, and vedio conferencing, in a highly integrated fashion" (PLAN, 1993).

The VDS concept actually conveys an ideal for design studio of the future. Yet the question is: *how to actually realise such an ideal?* To this end, there have been active VDS experiments in the last a couple of years (Cheng et al, 1994; Maher and Saad, 1995). In these experiments, which were participated by architectural students and staff plus some external practitioners, existing CMC (Computer-Mediated Communication) facilities, including e-mail, WWW and vedio conferencing systems, have been employed to support design collaboration across universities and countries. The experiments have been reported as quite successful with real design work carried out. Overall, as the experimenters also recognise, the present VDS activities have not touched high degree of collaboration yet, as the enabling technologies used are not sufficient or dedicated enough. While presentation level sharing via CMC has

been involved, semantic information sharing at CAD applications level has not been exploited. Furthermore, the VDS practice could have been better supported if dedicated virtual studio frameworks were available to integrate the separated CMC facilities involved.

In this paper the authors report a dedicated effort on VDS development, which attempts to seamlessly integrate within a collaboratve working context both the dispersed human designers and the distributed CAD applications. We'll first discuss from four dimensions several issues on design collaboration and computer support. A refined virtual studio model will then be presented, which absorbs the described issues into a computational model. This model is structured, by applying the spatial metaphor of the real-world design studio, as computerised settings for design collaboration. This will be followed by the description of the system implementation techniques, and the working virtual studio prototype system.

2. Issues on design collaboration and computer support

The nature of design collaboration and support may be discussed around issues such as the needs, the opportunities, and the methods. We are going to examine these issues from four dimensions, namely *social*, *information*, *processual* and *operational*.

Design As a Process of Social Construction Design always has a social dimension. The major issue in multi-disciplinary design is how to achieve high level of collaboration among designers, how each participant's design task can be managed so that it integrates well with the results of others. Modern design philosophy has began to address design ethics by urging designers to be more responsible to the society. The ultimate solution to this issue may be to enlarge the definition of "designer" by including all the relevant parties into the design practice. This calls for *participatory design*. Certainly this would push design collaboration issue to the limit. Participatory design naturally implies dynamic virtual organisations. Such virtual organisations are mostly characterised by the time and space disparities of the resources distribution and personnel participation. Contemporary technologies could support design in the social dimension via support for participation, sharing, awareness, understanding and coordination.

Design with common information space Information sharing is the fundamental means for achiveing mutual understanding. From this view, collaborative design can be perceived as the construction of a common information space, which allows team members to perceive, access, and manipulate the same set of information. Information sharing in distributed settings has much more implications than the traditional database systems do. Providing shared data access in mechanical sense is merely a minimum requirement. More importantly, mechanisms must be designed to allow collaborators to interpret and to resolve the meaning of the shared objects. Several techniques are regarded as particularly useful for design information to be effectively shared in collaborative design settings, including: recording information originator and intent, support for design object versioning and notification of changes.

Design with distributed tools Processually design may be perceived as interaction among design tools. A tool view of collaborative design is important: tools can be perceived as mediation artifacts for human-human interaction; more importantly, tools interaction reflects the purpose of collaboration, i.e., carrying out the design task. A tool perspective of design would allow the inter-dependencies in design production to be formulated in terms of the interaction amongst the distributed, autonomous applications. This could result in a medium-grained design process representation. Tools may interact with each in several ways. In a centralised design model, tools indirectly interact to one another via centralised databases. In a distributed design model, tools interact directly to each other through tool-tool communication interfaces such as messaging. These two approachs exemplifies a trade-off between consistency maintenance simplicity and system flexibility. A combined or hybrid model would centralise, as the centralised model does, the heavily shared data items into a core module, while leaving the more localised data items to the individual aspect modules as private data, as in distributed models.

Design As Computer-Supported Collaborative Working One major implication from the social dimension is that, as design becomes a natural activity and information technologies become so

been involved, semantic information sharing at CAD applications level has not been exploited. Furthermore, the VDS practice could have been better supported if dedicated virtual studio frameworks were available to integrate the separated CMC facilities involved.

In this paper the authors report a dedicated effort on VDS development, which attempts to seamlessly integrate within a collaboratve working context both the dispersed human designers and the distributed CAD applications. We'll first discuss from four dimensions several issues on design collaboration and computer support. A refined virtual studio model will then be presented, which absorbs the described issues into a computational model. This model is structured, by applying the spatial metaphor of the real-world design studio, as computerised settings for design collaboration. This will be followed by the description of the system implementation techniques, and the working virtual studio prototype system.

2. Issues on design collaboration and computer support

The nature of design collaboration and support may be discussed around issues such as the needs, the opportunities, and the methods. We are going to examine these issues from four dimensions, namely *social*, *information*, *processual* and *operational*.

Design As a Process of Social Construction Design always has a social dimension. The major issue in multi-disciplinary design is how to achieve high level of collaboration among designers, how each participant's design task can be managed so that it integrates well with the results of others. Modern design philosophy has began to address design ethics by urging designers to be more responsible to the society. The ultimate solution to this issue may be to enlarge the definition of "designer" by including all the relevant parties into the design practice. This calls for *participatory design*. Certainly this would push design collaboration issue to the limit. Participatory design naturally implies dynamic virtual organisations. Such virtual organisations are mostly characterised by the time and space disparities of the resources distribution and personnel participation. Contemporary technologies could support design in the social dimension via support for participation, sharing, awareness, understanding and coordination.

Design with common information space Information sharing is the fundamental means for achiveing mutual understanding. From this view, collaborative design can be perceived as the construction of a common information space, which allows team members to perceive, access, and manipulate the same set of information. Information sharing in distributed settings has much more implications than the traditional database systems do. Providing shared data access in mechanical sense is merely a minimum requirement. More importantly, mechanisms must be designed to allow collaborators to interpret and to resolve the meaning of the shared objects. Several techniques are regarded as particularly useful for design information to be effectively shared in collaborative design settings, including: recording information originator and intent, support for design object versioning and notification of changes.

Design with distributed tools Processually design may be perceived as interaction among design tools. A tool view of collaborative design is important: tools can be perceived as mediation artifacts for human-human interaction; more importantly, tools interaction reflects the purpose of collaboration, i.e., carrying out the design task. A tool perspective of design would allow the inter-dependencies in design production to be formulated in terms of the interaction amongst the distributed, autonomous applications. This could result in a medium-grained design process representation. Tools may interact with each in several ways. In a centralised design model, tools indirectly interact to one another via centralised databases. In a distributed design model, tools interact directly to each other through tool-tool communication interfaces such as messaging. These two approachs exemplifies a trade-off between consistency maintenance simplicity and system flexibility. A combined or hybrid model would centralise, as the centralised model does, the heavily shared data items into a core module, while leaving the more localised data items to the individual aspect modules as private data, as in distributed models.

Design As Computer-Supported Collaborative Working One major implication from the social dimension is that, as design becomes a natural activity and information technologies become so

powerful, the computer aids should not be limited anymore on the discrete, technical aspects of design problem solving. Instead, design systems should support seamless design work in the daily life. This takes design as computer-supported collaborative working (CSCW). A CSCW view of design would suggest that the support for direct human-human interaction (HHI) in terms of CMC mechanisms becomes an important part of the design system. Firstly, HHI support can provide "secondary" channel(s) to ensure the first channel (i.e., the domain application systems) to successfully operate. No matter how technically integrated the systems would be, there is always a good chance for "breakdowns" to turn up during use. Unlike in single user systems, the breakdowns here often involve multiple users, and therefore require some cooperative efforts to "recover". This further suggest that technical system integration should better be designed with the real collaborative use taken into account in the first place. Secondly, HHI support can encourage design process variety. Collaborative design does not have to be dependent on full integration of the technical systems. The intelligence and the flexibility of human beings can always compensate at least to some degree for the lack of application interfaces, when sufficient direct communication facilities are readily available at human users' finger-tip. Finally, HHI support can contribute for non-production functions (such as group well-being and member support) of work groups.

3. Virtual studio model

The discussed issues enforce the need to support the seamless design work: tool interoperation, human interaction, and the integration of the both. But how to computationally organise such computer supports and how will the computer supports be actually presented to and used by the designers? Since real-world design studio is the natural model for design collaboration, a computational model may be developed by applying and extending this real world model.

The use of metaphors in the design of computer artifacts is very common. Metaphors function as suggestions to the user that work with computer artifacts is similar to the work the user is familiar with in ordinary work situation. Properly applied metaphors make the systems look natural and easy to use. By purposefully applying sound metaphors the developers can more easily design and implement the systems functions and structures. Applying metaphor would normally involve extracting the relevant properties from the original image and mapping them into the computerised mirrors.

3.1 Real world design studio

Real world design studio's properties may be analysed in terms of its identity, its structural form and its behavioural functions. A design studio has its identity. A studio is used by specialists to engage in collaborative design work. This property of *use* makes studio differentiated from other types of offices, and further determines its structurs and functions.

A design studio defines a specific region through physical boundaries, which differentiate itself from other spaces. Such a bounded region serves to include some elements while excluding others. The normally included elements are the inhabitants and the facilities, such as drawing boards, desks, file cabinets, instruments, telecommunication facilities, notice boards.

Design studio accommodates both individualistic and collective activities. More personal workspace may be established around the desks allocated to individuals. More importantly, studio provides a collaborative environment for collective activities, which are constituted by the coordinated individual activities. This coordination is facilitated by a variety of resources and mechanisms provided by the studio, which may be broadly classified as two categories: the physical co-presence of colleagues makes it possible for each other to become instantly aware of *who is doing what* through the natural vocal and visual communication channels and other interactional cues. Other sharable resources such as public file cabinets and notice boards support semi-synchronous and asynchronous, formal and informal coordination. Furthermore, apart from "job-related" activities, design studio also supports socialising. In fact, in real-world workplaces formal and informal activities are always co-existent.

3.2 Virtual studio models

A virtual studio model emerges when we transfer the properties of the real world design studio into computer systems.

Virtual studio identity A virtual studio provides bounded virtual region in the form of integrated computer environment for a design project. It is "bounded" only in the sense that the memberships and resource inclusion are highly controllable. There could be no physical boundary; this mostly makes *virtual* studio different from *real* one. Instead, the physical boundary of a virtual studio extends to the limit of the effective communications, which the underlying computer networks can afford. Therefore, a virtual studio is a distributed, open, but highly integrated multi-user system.

Virtual studio's structures Virtual studio is structured by imitating the real design studio. The interfacing between human and computer should be designed by applying the spatial metaphor of the physical studio. Both human participants and studio resources are visually represented as icons in a graphical user interface, which is organised to resemble the looks and feel of real studio. Such a virtual studio can be visually presented as a 2D or 3D space, which is further divided into areas. Each area presents a particular studio resource or function. For example, a virtual note board can be designed as a studio area for users to post and read notes; a virtual co-present area may be designed to display the current inhabitants of the studio such that the participants can instantly "see" each other; and so on. Virtual studio resources include both CMC utilities and CAD applications.

Virtual studio's functions Virtual studio supports both private and collective activities by acting as both private and shared virtual workspaces. Ownership and accessibility are implemented with virtual studios. Interaction in virtual studio takes place through CMC and/or CAD applications. Mutual awareness is both facilitated and inhibited by virtual studios due to studio's spatial properties. For example, users would be more aware of each other when they participate in the same virtual studio than in different studios. Like its real world counterpart, virtual studio encourages informal socialising. This implies that virtual studio activities can be loosely defined. Such informality is also important to support design process variety.

With computer's automatic power, it is expected that many virtual studios can be instantiated and configured with ease to form different settings for different design tasks. Since one big project could use several virtual studios, some of the virtual studios would be interdependent on one another. This calls for mechanisms for the users to construct and navigate across different virtual studios. To meet such kind of demands, *virtual studio environment* (VSE) has been coined by the authors[Chen, 1996] to refer to such *a software environment which supports the creation, operation and management of the virtual studios.* VSE is thus a virtual studio management system, which can can provide persistency and manage the life-cycles for the virtual studios.

Extending from the concept of singleton virtual studio into VSE results in a big leap from the perspective of technology exploitation. Instead of serving as an one-off, special-purposed design system, a VSE possesses the kind of abstraction and generality, which allow different kinds of virtual studios (each of which serves as an integrated design environment) to be spawned with ease. Naturally, different virtual studios within a VSE can easily share resources such as CAD applications.

VSE is a very powerful integration vehicle. As *containers* virtual studios are units of focus on related tasks, and units of resource management. This could be very useful to tackle the technical complexity of the design work. *As shared work spaces* virtual studios support group formation and interaction on not only technical problem solving but socialising as well. Most importantly, virtual studio supports a level of user embodiment in the sense that human participants are somehow immersed within the computer system through mutual awareness mechanisms. The users thus form an integral part of the environment and the interaction takes place *within* the systems. It is in this sense that virtual studio is not just an integrated or multi-user design system; it is the fusion of *social* and *technical.*

In comparison to the original VDS notion and present VDS activities, the VSE concepts proposed represent a significant expansion and advance. Although the term "virtual design studio" itself implies a metaphor in the first place, but the present VDS efforts touches this metaphor only at a very superficial level. In comparison, the virtual studio model refined in this paper stands for truly integrated environment with explicit studio identity. The studio metaphor has been applied to levels from presentation through functions to structures. Furthermore, VSE is a virtual studio management system, which helps create, configure and manage potentially large amount of studios for intra-organisation or inter-organisation collaboration.

4. System design

This section describes the system design for a VSE prototype. This includes designing system architectures, finalising virtual studio models, and selecting an application scenario for prototyping.

4.1 VSE System Architecture

A distributed system architecture has been designed for VSE implementation. As illustrated in **Figure 1**, it consists of three types of modules: *VSE server*, *VSE client* and *domain resource*.

The VSE server and VSE clients together form a *VSE base system*. Such a base system is a multiuser interaction environment. The VSE server provides persistency for VSE objects and maintains an operational context. The VSE clients act as interfaces for human users to access VSE resources. VSE clients are presented through WYSIWIS (What-You-See-Is-What-I-See) graphical user interfaces (GUIs). Rich CMC facilities together with the default studio production databases will be designed as built-in resources bundled to the client-server environment. Although inter-client communication is primarily mediated through the VSE server, direct inter-client connections are provided in certain circumstances to reduce the overall communication load.

The domain resources correspond to CAD applications, which are *loosely coupled* with VSE base system through *tool agents*. Like VSE clients the domain resources are distributed in the *Internet*. Before being actually accessible within the VSE system, they have to be registered with the VSE server, which results in entries being added into the global resource directories. Once registered, the domain resources can be invoked or queried by different studios and/or by different users within a studio.

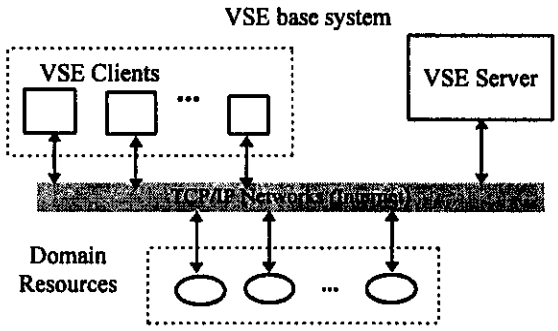

Figure 1: VSE system architecture

The separation of the domain resources from the VSE base system encourages autonomy and participation. For example, the individual participants may contribute their favourite tools to the studio, which can be either solely used by themselves or shared by other people as well.

A state diagram is used (**Figure 2**) to illustrate major VSE operations. It has been decided that the VSE system will be operated as other multi-user *Internet* applications (e.g. MUD and IRC) via users IDs (and passwords), based on which higher level of user embodiment may be implemented.

Three system states have been defined: *Outside Telecentre*, *Inside Telecentre*, and *Inside Studio*. Here *Telecentre* is used to denote the state immediately after a user has successfully logged in on VSE server. Therefore, Telecentre actually stands for the hall of the virtual building. The users need to register with the VSE server to obtain a user ID and password. After checked in, the users may create/destroy studios, register/withdraw resources, change the access state of the studios, or enter an existing studio. When

Inside (a) Studio, the user may add resources from the global resource directories into the studio's domain resource list, remove or access the resources which are already added in.

Most of the functions (including both CMC and domain resource invocation) are *studio sensitive* in the spirit of the spatial metaphor. For example, when someone "say[1]" something, only inhabitants in the same studio can hear; a design tool can only communicate with other tools which are added in the same studio; and so on.

4.2 Studio Model

As unit of resource management and medium for group formation, a virtual studio is proposed to be characterised by three major components.

The first component is the **Particulars**, providing the general information about the purpose and the activity for which the studio is created and used. This component includes elements such as *studio name*, *studio ownership*, *studio accessibility*, and *studio description*.

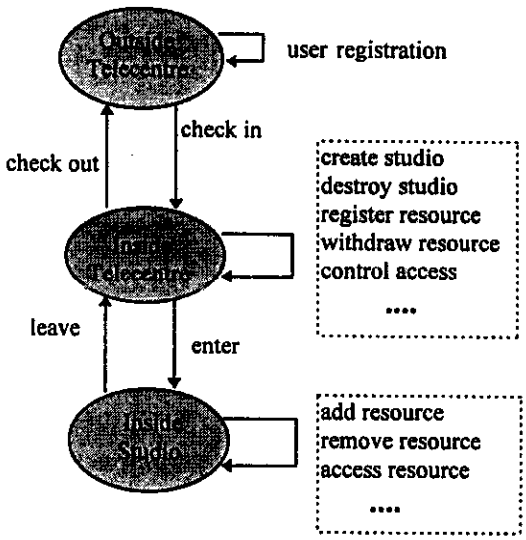

Figure 2: VSE state diagram

The second component is the **Resources** that populate the workspace of the virtual studio. The resources are classified into three categories: *CMC utilities*, *CAD tools* and *studio databases*. *CMC utilities* include (i) text-based commands for VSE information access, conversation, and message; (ii) an image communication kit for users to exchange on a real-time basis part or whole of the screen displays of each other's workstation; (iii) a note board for users and VSE server to announce events or for similar purposes. The *CAD tools* in studio model are actually pointers, pointing to the tools agents which are geographically dispersed along with the actual tools. Each CAD tool supports interactive data manipulation, local (multi-media) databases, and communication interfaces with other tools and VSE servers. Two default *studio databases* are "NoteArch" and "Production". The formal is for archiving old notes which have been removed from note board, while the later for recording domain-specific design production. Since each aspect tool maintains its local database, studio design database is actually formed as collections of pointers, which point to the distributed multi-media databases. Design database is maintained in terms of design production sessions, which function as versioning. Design intent, information originator and notification are also implemented.

The third component is **Persons**, who are participating in and interacting with the studio. Each VSE user is associated with a *Person* record/object, which is attributed with elements including *user ID* and *password*, *email address*, *visual image*, *info*, *working locale*, *state* and *activity*.

[1] If someone wants the people in other studios to hear, she can "shout" or "page".

4.3 A Collaborative Design Domain

Conceptual building design has been chosen as an application scenario for VSE prototyping. A hybrid design model has been adopted (Figure 3). The FGC (Function, Geometry & Construction) is a core module which centralises the heavily shared data items. The three building performance aspects are each treated as an individual aspect module. Inter-module interfaces are provided between FGC and each of the aspect tools. Amongst the aspect modules, thermal and costing require direct communication links to implement the sequential dependency on energy consumption data.

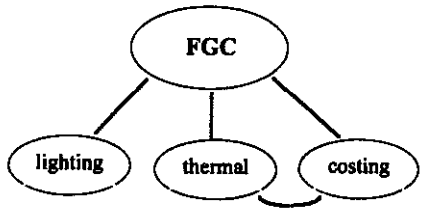

Figure 3. Conceptual building design

5. Prototype implementation

Two key issues on the realisation of a collaborative system like VSE are (i) how the distributed components are actually interconnected, and (ii) how the distributed system is presented to the collaborative users through multi-user interfaces. A OMG CORBA compliant toolkit called ORBeline (PostModern, 1994) is used to model and program the interfaces for *VSE server--VSE client, tool--tool, tool--VSE server, tool agent--VSE client* communications. WWW technologies are utilised for implementing the distributed multi-media studio design databases. A objeect-oriented database management system called ODE (Dar et al, 1992) is used for designing the administration databases for VSE server. The client GUI has been designed with a quasi-WYSIWIS presentation method and a hybrid implementation architecture, and programmed in X11 (OSF Motif).

5.1 Domain CAD Tools

The core module "FGC" has been implemented by customising and further developing AutoCAD (R12). A thermal tool called "AngliaDom" is an X11 implemetation of a BREDEM-8 model. A costing tool called "PSCost" and a lighting tool called "NATLIT" are based on the algorithms and sources codes of two existing programs from ABACUS of Strathclyde University.

Before a CAD tool can be plugged into VSE system it needs to be "agentised" through a tool agent(TA). TA serves (including registration and remote invocation) the tool to VSE community. TA hides the details of the tools from the VSE customers. Such a TA method is crucial for the sharing and management of tools in complicated settings which VSE is intended to accommodate. The abstraction provided through TA makes it possible to change the details of the domain tools without modifying the VSE system (both VSE server and VSE client). TA also takes care of the house keeping for tool invocation. By setting up different invocation contexts for different requests, TA facilitates the sharing of the same tool by different users or by different studios. Each invocation instance is shielded by its own operating context, and is thus not subject to interference with other invocation instances.

5.2 VSE Base System

The VSE base system is an Internet-based operating environment, which facilitates the creation, access and management of virtual studios.

(1). VSE server

VSE server provides persistency for VSE objects and manages all operations in regards to the use of VSE. The server program is organised through three function agents -- *Services Agent (SA), Heartbeat Agent (HbA)* and *Notification Agent (NA)* -- which run concurrently but cooperatively.

SA is responsible for actually providing services for the clients. It takes requests from clients, invokes services functions to access the working context and databases, returns results to the clients, and places update orders to Notification Agent when so needed.

HbA is responsible for monitoring clients connections by periodically sending "heartbeat" signals to connected clients. Once disconnection detected, HA notifies the SA to remove the clients from the client-list maintained in the working context. This is particularly designed for the situations in which the clients may be disconnected with abnormal methods (such as being accidentally terminated by the users or disconnected by networking failures). Such situations need to be identified for SA to get rid of the "ghost" clients from the working context.

In VSE the action from one client often needs to be propogated to other clients. For example, when a note is created into a note board by one user, the rest of clients in the same studio need to have this note displayed on their note boards as well. NA takes update orders from SA, and sends out update messages to clients which are affected.

(2). VSE client

VSE client program is organised through two cooperating agents: Interface Agent (IA) and External Event Aganet(EEA).

IA manages the window resources of the GUI, and handles user operations. IA collects user's actions to form VSE services requests, sends the requests to the SA agent in the server. The results from the requests are then displayed in the GUI. This could result in either printing some texts in the Monitor area, changing the layouts of GUI items, or popping up new windows. By taking the update messages from EEA, IA also updates the GUI to reflect other users operations in the same studio. Apart from acting as clients to VSE server, IA also makes requests for tool services to the tool agents, when the user invokes CAD tools.

EEA takes the responsibility to monitor the external requests coming from VSE server. Such requests usually ask the client to confirm alive connection or to update the GUI layout to reflect some operations which are initiated by other users in VSE. For example, when someone is trying to talk to this client user, the message should be printed into this client GUI's monitor area. The other examples include when other studio inhabitants change the studio resources (e.g. adding/removing a tool) or when new/existing users enter/leave this studio.

The client GUI is designed by applying the spatial metaphor. As depicted in **Figure 4**, the main window of client GUI displays a virtual studio in 2D space.

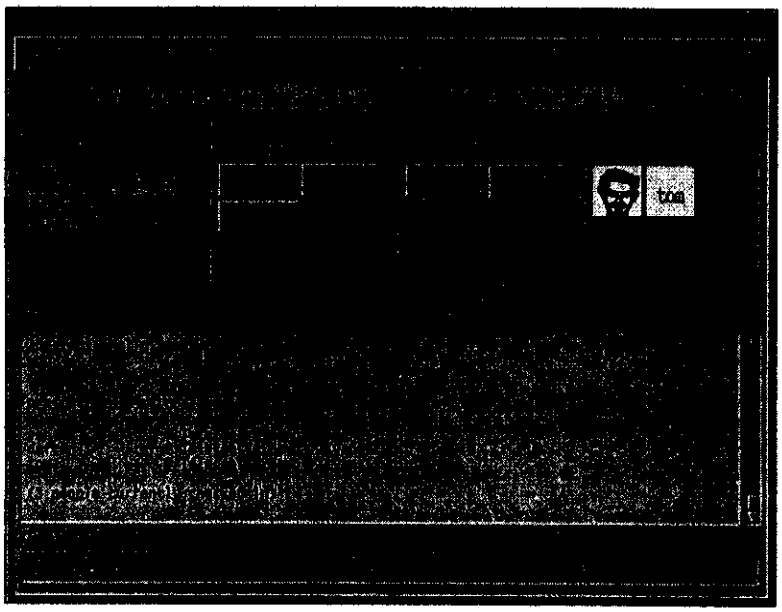

Figure 4: The main window of VSE client GUI

- **Locale** The Locale panel displays the user name and position in VSE for the client user.
- **Studio areas** Studio is visually presented as four areas: *Note Board*, *CAD Tools*, *Databases*, and *Co-present*. These areas display the resources/personnel which are currently available/present in this studio. Each area can display up to six items in the form of icons. These icons are active or invokable. By clicking a icon with the activation mouse button (the left button), the intended function associated with the icon will be invoked. This could be displaying a note in a pop-up note browser, activating a possiblly remotely hosted CAD tool but displaying it locally, browsing a studio database in a database browser, or showing a person's status in the monitor window. By clicking an icon with the selection mouse button (the right button), a small pup-up menu will be popped up to allow the user to remove the note from NoteBoard, to view the information about a tool, and so on. Note that while the size of each areas is designed as holding six items, the actual number of items for each category of the resource (and personnel) can be indefinite (up to computer's capacity). This is because internally the resources of the studio are represented as lists. While the latest six items are displayed in GUI areas visually as icons, the whole lists can be browsed and invoked from the pull-down menus under the top menu bar. In fact, all resource manipulation operations, most of which are not accessible through the area icons, are organised through the pull-down menus. Therefore, the capacity-constrained areas function as buffers or front end for the resource lists to achieve "studio-alike" viual impact, and the operation convenience through direct manipulation.

  Note that Co-present area does not display the icon of the client user himself/herself. This is because that in the real studio one can only see others, but not herself/himdelf. The GUI is supposed to be an electronic equivalent to the real studio, so the user cannot see self either.
- **Monitor** The Monitor displays texts for user operation feedback (including user's typing) and external events. This is implemented as a scrolled window with large buffering capacity. Therefore, the user can examine the history of previous VSE operations simply by scrolling this text window back.
- **Command** The Command text field in the bottom of the GUI allows text-based commands to be entered. These commands may relate to conversation, messaging or VSE information interrogation.
- **Menubar** The menubar on the top of the GUI provides through pull-down menus rich facilities for users to undertake all VSE activities. This menubar is organised as Seven categories: "Telecentre", "Studio", "NoteBoard", "Tool", "Databases", "Utilities" and "Help". "Utilities" is for organising those facilities which are not visually presented in studio areas. At the moment, the image-based communication kit is organised in there.

6. Conclusion

A virtual studio system has been developed, which supports with the same studio environment both the dispersed human interaction and distributed CAD resources integration. The system has been initially tested in distributed settings involving Anglia and Strathclyde sites (which are about 500 miles away). As far as speed is concerned, the implemented prototype system can deliver sufficient performance. The initial trial use of the prototype system through several informal demonstrations has been very positive. The demonstration attendees appeared to be mostly impressed by the rich communication facilities embedded, the integration and sharing of the distributed CAD resources, and the powerful visual impacts of the multi-user GUIs.

The implemented prototype system only supports conceptual building design at present. To cover more design phases/aspects would be straightforward by adding more design tools. The system architectures, particularly the loose coupling between the VSE baser system and the domain resources, and the tool agent integration method, has well prepared for such extensions.