computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific Journal of Computing

A COLLABORATIVE PLATFORM FOR HETEROGENEOUS CSCW SYSTEMS: CASE STUDY OF ACADEMIC APPLICATIONS

Rahat Iqbal, Anne James, Richard Gatward

School of Mathematical and Information Sciences
Coventry University
CV1 5FB, UK
{r.iqbal, a.james, richardg}@coventry.ac.uk

Abstract: *A variety of computer based information systems are used to support the activities in an academic environment. These systems are used for conducting lectures, designing and reviewing modules, designing and writing assignments, laboratory work, and computer based assessment. The systems are typically designed from scratch if the existing systems do not meet the requirements. This incurs significant costs, and inconvenience. This paper reports on work concerning the integration of existing computer based systems which is formally known as computer supported cooperative work (CSCW) in order to support every day activities. A framework for CSCW integration is presented.* A integrative methodology based on this framework is proposed. *An example application scenario involving integration of asynchronous application of our university is discussed.*

Keywords: - *Collaboration, Coordination, Integration.*

1. INTRODUCTION

Computer Supported Cooperative Work (CSCW) emerged as an important research area, which focuses on suitable forms of cooperation between users or a group of users to perform a common task. It is concerned with design, implementation and realisation of computer support for cooperation to achieve the common goals.

Generally, CSCW supports a range of applications such as shared editors, audio/video conferencing, computerized meeting rooms, group design tools, co-authoring systems, shared calendars, workflow system, voting tools, whiteboard and message based conferencing [1]. Unfortunately, all these application are closed and limited to registered users. These applications do not get advantage of each other as they are working in isolation.

An open CSCW system is required that supports a wide range of applications and a variety of cooperative users in order to get the advantages communally [3]. To make all the applications work together a platform is required that can contain a collection of heterogeneous applications, paradigms and models. This should provide interoperability among different applications running locally or remotely at different platforms supporting synchronous or asynchronous activity. Such a CSCW system can meet the requirements of all the users [1]. This should allow the users of these applications to register an activity or a group of activities to share with other users and applications.

The following are some reasons for the usefulness of interoperation:

- *Support activities and share resources:* Users need to communicate with each other in order to support their activities and to share resources. For example, in our university, different lecturers are working on various modules. They are cooperating by sharing teaching modules, revising and reviewing the contents of the module etc.
- *User preferences:* CSCW systems are heterogeneous and each offers a unique set of benefits. Users may be using their preferable system for long and they do not wish to give it up and adopt a new system.
- *User constraints and training:* Users are trained in constrained to use different CSCW systems. They may not have the time, desire, or ability to learn a new, common system in order to collaborate with each other [2].
- *Reduce cost and inconvenience:* CSCW systems are typically designed from scratch if the existing ones do not meet the needs. This incurs significant costs, and inconvenience.
- *Improve efficiency and enhance functionality:*

CSCW systems are heterogeneous and each system offers limited functionality to its users. The efficiency can be improved and functionality can be enhanced if these system can communicate with each other.The rest of the paper is organised as follow. Section 2 describes different levels of integration. Section 3 discusses related work and presents our framework of integration. This section also describes an integrative methodology. Section 4 discusses heterogeneous applications of our university. Section 5 discusses integration process. Finally, section 6 discusses the work and provides some conclusions.

2. LEVELS OF INTEGRATION

Different levels of integration are possible. At one level two autonomous systems may interoperate by passing data to each other either directly or through a common "blackboard" area. Thus, activities in one application may be affected by information received from other applications but integration here is at a loose level of coupling and may be termed surface integration. A deeper level of integration would involve merging or consolidating some activities or resources. This process may involve resolving conflicts between comparable activities or resources in different systems. A complete integration would involve making one single system from underlying systems where all conflicts among activities or application objects have been resolved. At levels two and three, the question of virtual or real integration arises. At these levels, both real or virtual integration is possible. In the case of the latter mappings would need to exist from a conceptual integrated model to underlying physical applications.

It is worth noting that surface integration can be achieved fairly easily using current technology at the level of service provision. A CSCW system that wishes to make its functionality and information available publicly can do so by participating in distributed system services such as CORBA or Web Services. However such systems provide integration or interoperability only at a syntactic level. Semantic detail concerning real-world understanding of what the CSCW system does and what information it has, is not supported. Thus the use of this type of integration alone is limited. However middleware such as CORBA and Web Services is likely to be used at a lower infrastructure level upon which more semantically driven integration can take place.

3. RELATED WORK

Although the problem for integration of CSCW systems was identified in the early 90's, it has not yet been solved. There has been no work since then on developing an integrative approach with the exception of following work.

(i) Dewan's work [2]: Dewan addresses some of the basic issues in interoperating heterogeneous collaborative systems. They include coupling, semantic and some implementation issues. This work mainly focuses on the integration of floor control mechanism with locking system. The results of this work show that it is possible to interoperate a synchronously coupled, fully replicated, floor control system with a flexibly coupled, partially centralised, lock system. Floor control is the simplest form of concurrency control which allows only one user to input to the system at any given time. The user who wishes to operate the system has to request for floor control. The floor will be granted if it is free otherwise request will be discarded or enqueued. Different techniques are in use such as turn-taking protocols. The problem with this technique is that it does not allow multiple users to perform actions in parallel even if their actions do not conflict. Lock based concurrency control has addressed these problems. It allows users to obtain locks and work concurrently as long as they do not wish to work on the same objects. This work adopts an approach that assumes the source code and internal knowledge of the groupware applications to be interoperated is known.

(ii) Li's work [4]: This work addresses the problem called intelligent collaboration transparency (ICT), in which the issues of interoperability between single-user heterogeneous applications are addressed.. This work adopts a 'blackbox' assumption, which assumes that the source code and internal knowledge of the groupware applications to be interoperated is unknown and no modification is allowed. Using this approach, the applications sharing infrastructure is interposed between the applications to be shared and their window environment at each site. Users can collaborate on the common task using their favourite single-user heterogeneous applications. The infrastructure captures and replays user input to the applications.

(iii) LaMarca's work [5]: This work provides support for content as well as for coordination in collaborative work. It considers coordination and collaborative functionality as an aspect of the collaborative artefact rather than a collaborative application. Basically, this work considers coordination and collaboration as separated and independent of applications. This approach provides a mechanism to monitor the application access to a shared data repository and trigger user supplied programs when interesting operations are performed. It enables heterogeneous single user applications to be interoperated and converted into groupware

without modification. This approach is limited and not referred to as application sharing systems.

None of the above approaches provides full integration. They provide only partial solutions. The proposed framework [6] focuses on an approach leading to full integration. According to this work, a CSCW system may be seen as consisting of an ontological model, a co-ordination model and a user interface model. The description of these models are given below:

- The ontological model specifies all objects in the application, their relationships and terminologies.
- The co-ordination model specifies how interactions occur within the system and describes workflow.
- The user interface model describes how the users see the system and how the system is presented at an interface level.

In a fully integrated system all three aspects would need to be integrated. Figure 1 shows the different levels of integration from an architectural viewpoint. The concept of the security model and the transaction model is omitted in the above description.

An integrative methodology is proposed based on this framework. This methodology involves decomposing the components of applications in order to fully understand the underlying concepts and analyzing them. It supports different levels of integration including ontology, security, coordination, transaction and user interface. Furthermore, it emphasises the structural and terminological transformation as well as encoding and decoding in order to achieve different levels of integration.

The methodology consists of five steps:

(i) selection of applications (which are based on either same model or different model);
(ii) analysis of applications (in terms of the ontological model, the security model, the coordination model, the transaction model and the user interface model);
(iii) finding common concepts (and resolve conflicts between the concepts);
(iv) explanation of context and implicit concepts;
(v) mapping specification (at three levels: structural transformation, terminological transformation and encoding & decoding).

The components of the methodology are shown in figure 2.

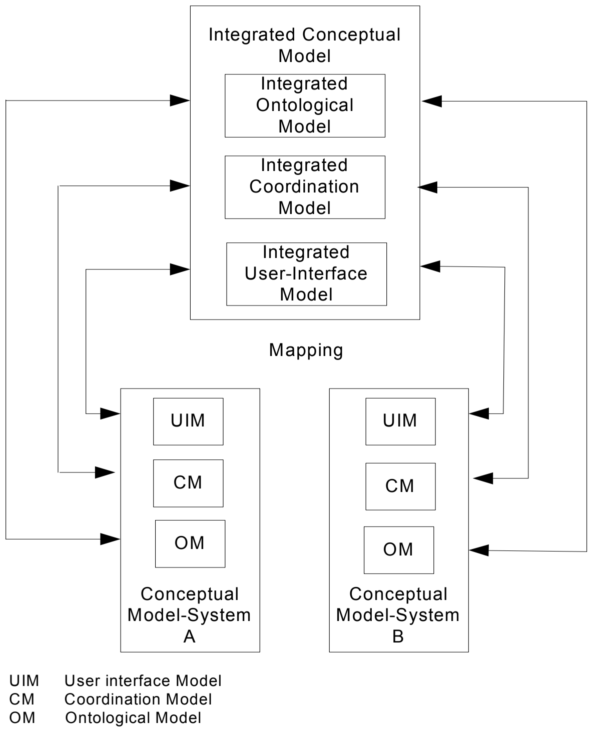

Integrated Conceptual Model
Integrated Ontological Model
Integrated Coordination Model
Integrated User-Interface Model
Mapping
UIM
CM
OM
Conceptual Model-System A
UIM
CM
OM
Conceptual Model-System B
UIM User interface Model
CM Coordination Model
OM Ontological Model

Fig 1: Framework for integration of CSCW

4. CASE STUDY

Education is a cooperative activity [7] where different synchronous and asynchronous applications work together in order to support on-line lectures, designing and reviewing modules, designing and writing assignments, laboratory work, and on-line assessment. Here, we discuss the following two heterogeneous applications of our university.

5. DOCUMENT MANAGEMENT SYSTEM

Document Management System (DMS), is an asynchronous collaborative application. This application helps us monitor modules of different disciplines in the university. This application involves the following two main activities. Some problems related to these activities are also discussed in brief.

1. *Revise Module:* This is similar to editing a document. More than one lecturer is involved in revising the contents of each module. Mostly, the lecturer who is teaching that module is responsible to make changes in the contents of that module if required. In case of brand new modules, the administrator assigns the job to one or more than one lecturers to write contents of the module. Some of the issues are involved in accomplishing this activity: (i) *version control*, as more than one lecturers is involved in revising the contents of a module and sending different copies to the administrator time to

time. Later on, it becomes difficult to decide on the version of the module and find out which one is the latest. In most cases, the lecturer concerned names a module according to his wish and the administrator names it based on the available version of the related modules; (ii) *role and responsibility,* as more than one lecturer is involved in revising or writing a module, so at one stage, it becomes difficult to know who is responsible for which part of the module or even for which module;

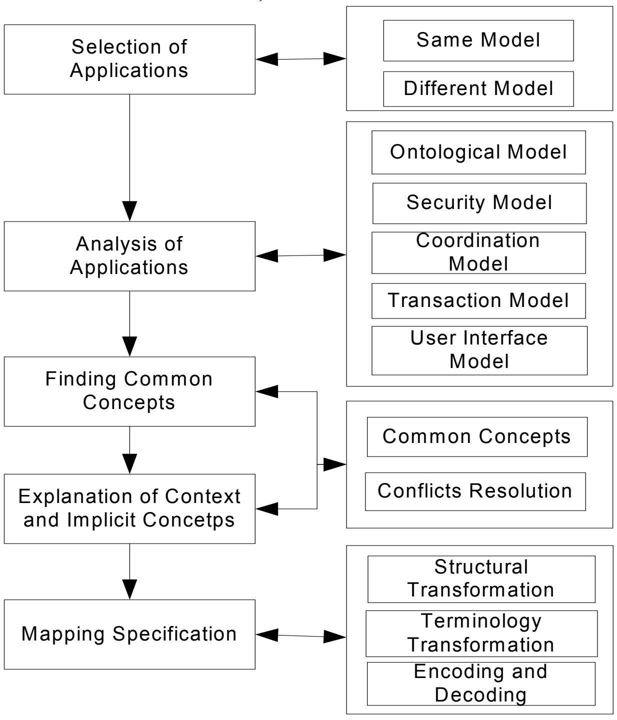


Figure 2: Methodology for the integration of CSCW

(iii) *meeting deadlines,* with a huge amount of work in an academic institute, it becomes hard for lecturers to revise the module on time and if more than one lecturers are involved, they can not get a time for face to face meeting. Normally, this is not always the case, but it happens sometimes;

2. *Review Module:* After the module has been revised or written as a new module, it needs to go to a member of Subject Quality Group (SQG) for quality assurance, which is a necessarily required in an academic environment. The following issues are involved in this process: (i) *version control,* what is the latest copy to send to SQG; (ii) no record to keep track of the modules sent to SQG. Some times SQG does not respond in time about the acceptance or rejection of the module, which delays the process and creates some other problems which effects the activities involved in Module Assignment System, another asynchronous groupware application.

The above activities requires different actors to perform some actions on them. These actions differ and depends on the role played by an actor. The actor has one or more of the following roles:

- The first role is of a writer who can edit a document (module). The writer can make necessary changes in the whole document or in the part of the document. In this example, a writer can be a module leader who is allowed to modify or change an existing module or write a new module.
- The second role is of a viewer who can view the document but cannot modify it. In our example, a viewer can be a lecturer who can view the modules but can not modify it.
- The third role is of an administrator who can assign different roles to other actors. The administrator who is also a head of department in our case provides different roles to the lecturers. One lecturer can be a viewer at one moment and at the second moment the same can be a editor of a specified module.

7. MODULE ASSIGNMENT SYSTEM

Module Assignment System (MAS) is another asynchronous collaborative application. This application is used to assign different modules to the lecturers. It involves the following activity. Some issues related to this activity are also discussed below:

1. *Module Assignment:* The administrator views lecturers' list and modules' list and then assign different modules to different lecturers but it is not as simple as it looks. The following are some of the issues related to this activity: (i) *updated module,* the list of updated module must be available when the administrator is assigning the modules; (ii) *updated list of lecturers,* it should also be made available at the time of assignment; (iii) *lecturers' preference,* the lecturers have their own preference about the module which they want to teach; (iv) *working load,* the administrator must know how much teaching load is to be allocated to each lecturer keeping in view the other activiaties the lecturer is involved in, which may include administration duties, admission duties, supervision of research students.

The above activities requires different actors to perform some actions on them. These actions differ and depends on the role played by an actor. The actor has one or more of the following roles:

- The first and the most important role in this application is of an administrator who is allowed to view the modules and assign these modules to lecturers.
- The second role is of a viewer who can view the

module but cannot modify it. In our example, a viewer can be a lecturer who can view the modules but can not modify it. The administrator assign this role to the lecturer only when he or she finishes his or her tasks which is assigning modules to lecturers.

6. IMPLEMENTATION

This section addresses implementation of heterogeneous DMS and MAS of our university by building an integrated ontological model as shown in figure 3. Mapper creates a map between the local ontological model of both applications and represent them using XML. We propose that an integration model should contain some primitive concepts such as'actor', 'activity' and 'object', as the building blocks for the definition of other concepts [8], [9]. These concepts are based on the strengths and commonalities of different models and theories i.e., coordination theory [10], activity theory [11], tasks manager model [12], action/interaction theory [13] and object oriented activity support model [14]. For more detail, the reader is referred to [15]. We use these concepts as the building blocks of our integration model. These concepts are common to all applications and the advantage is that they are independent of target application [16].

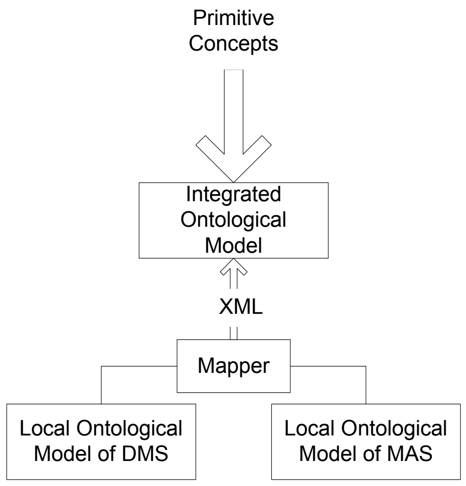

Primitive Concepts
Integrated Ontological Model
XML
Mapper
Local Ontological Model of DMS
Local Ontological Model of MAS

Figure 3: Integrated Ontological Model

The heterogeneous systems use integrated ontological model (or shared ontology) to communicate with each other in order to share information to support activities. The concept of shared ontology is not only limited to human actors but agents also use ontology for communication purposes.

Artificial intelligence and knowledge engineering use different ways to represent an ontology such as the logical, semantic, datalog, and frame-based. To address an integration issue in CSCW we have adopted the following ways.

An initial step towards building a shared ontology is to develop a glossary of terms as used in Unified Modeling Language (UML) or data dictionary as used in database applications. Full knowledge of the applications is required to develop a glossary of terms. We have developed the glossary of terms of both applications and described in table 1 and 2. The glossary of terms has been established based on the available contextual information. The contextual information is important because we aim to achieve integration at semantic level. The integration model should have a full knowledge of context and implicit concepts used in the application models. We have used the following terminology classification to resolve the conflicts between the two ontological models [8], [9] [17]:

- *Identical concept:* Same concept, same meaning and same structure/constraints.
- *Synonyms:* Same concept (meaning) but different name.
- *Homonyms:* Same name but different meaning.
- *Compatible:* Same concept, same meaning and different structure/constraints but not contradictory.
- *In-compatible:* Same concept, same meaning and different structure/constraints but contradictory.
- *Complex concept:* A group of one or more concepts in one application corresponds to one or more concepts in an integration model.
- *Partitioned concepts:* Two or more concepts in one application corresponds to a single general concept in an integration model.

Table 1: Glossary of terms in DMS

Name	Type	Description
Lecturer	Actor	Person who views and edits the module and log book
Administrator	Actor	Person who views, adds, deletes, and archives module and views log book
SQG	Actor	Person who views log book, and module and accepts or rejects module
Module	Object	The document on which different operations are carried out by different actors
Log Book (this concept is omitted in the description	Object	Book on which different operations are carried out by different actors to keep record

for simplicity)		
Monitor	Activity	Activity performed by different actors to monitor the module
Revise	Activity	Activity performed by lecturer to revise the module
Approval	Activity	Activity performed by SQG to make decision on the acceptance or rejection of modules

Table 2.: Glossary of terms in MAS

Name	**Type**	**Description**
Administrator	Actor	The person who views lecturer list and module list and then assign modules to lecturer
Module	Object	The document on which different operations are carried out by Administrator
Lecturer	Actor	The person who is assigned different modules
Assignment	Activity	Activity performed by Administrator in which modules are assigned to lecturers and decision is made on running modules

As a second step towards the design and development of ontology, we employ XML to represent ontologies because it provides a uniform platform for representing heterogeneous concepts.

XML is considered a potential for information exchange between different systems. For the representation and exchange of information between DMS and MAS, We have developed the following three DTD (Data Type Definition) models. These models are developed to represent the data only.

- DTD for representing domain ontologies based on the primitive concepts discussed in previous section.
- DTD for Document Management System
- DTD for Module Assignment System

```
<! - -PRIMITIVE CONCEPTS - ->
<! ELEMENT ACTIVITY (GOAL, STATE,
ACTION?, SUBACTIVITY*)>
    <! ELEMENT GOAL (# PCDATA)>
    <!ELEMENT STATE (#PCDATA)>
    <!ELEMENT ACTION (OBJECT, ACTOR)>
<! ELEMENT OBJECT (OBJECT NUMBER,
OBJECT NAME, SOURCE? DESTINATION?,
ACTORS?, ACCESSPATH?, PLACEIN,
MODIFIED, OBJECTCLASS, CONTENT)>
…..
…..
    <! ELEMENT SUBACTIVITY (GOAL, STATE,
ACTION)>
    ….
    ….
        <! ELEMENT ACTOR (ACTORNUMBER,
ACTORNAME)>

<! - -DTD MODEL FOR DMS - ->
<! ELEMENT ACTIVITY (NAME, STATUS,
ACTION)>
<! ELEMENT ACTIVITY NAME = MONITOR
MODULE>
<! ELEMENT  MONITOR MODULE (MODULE
NAME, MODULE CODE, MODULE STATUS,
LECTURER, ADMINISTRATOR )>
<! ELEMENT MODULE NAME ( #PCDATA)>
<! ELEMENT MODULE CODE (#PCDATA)>
<! ELEMENT MODULE STATUS (#PCDATA)>
<! ELEMENT LECTURER (LECTURER NAME,
LECTURER NUMBER,  ADDRESS)>
<! ELEMENT LECTURER NAME (FIRST NAME,
MIDDLE NAME,LAST NAME>
<! ELEMENT FIRST NAME (#PCDATA)>
<! ELEMENT MIDDLE NAME (#PCDATA)>
<! ELEMENT  LAST NAME (#PCDATA)>
<! ELEMENT LECTURER ADDRESS (STREET,
CODE, CITY)>
……
…….
<! ELEMENT ADMINISTRATOR
(ADMINISTRATOR NAME, NUMBER,
ADDRESS?)>
<! ELEMENT ADMINISTRATOR NAME (FIRST
NAME, MIDDLE NAME,LAST NAME>
<! ELEMENT FIRST NAME (#PCDATA)>
…..
…..
<! - -DTD MODEL FOR MAS - ->
<! ELEMENT ACTIVITY NAME = ASSIGN
MODULE>
<! ELEMENT ASSIGN MODULE (MODULE
NAME, MODULE CODE, LECTURER )>
<! ELEMENT MODULE NAME ( #PCDATA)>
<! ELEMENT MODULE CODE (#PCDATA)>
<! ELEMENT LECTURER (LECTURER NAME,
TEACHING MODULE,  ADDRESS?)>
<! ELEMENT LECTURER NAME (NAME, LAST
NAME?)>
<! ELEMENT FIRST NAME (#PCDATA)>…..
…..
<! ELEMENT ADMINISTRATOR
(ADMINISTRATOR NAME,  ADDRESS?)>
```

```
<! ELEMENT ADMINISTRATOR NAME (NAME, ADDRESS?)>

<! ELEMENT NAME (#PCDATA)>
…..
```

8. CONCLUSION AND FUTURE WORK

The need for open CSCW systems has been discussed. To this end we have looked at generic models for CSCW and developed our own model based on previous work. A framework for integration has been discussed. An integrative methodology based on our framework has been proposed and discussed. In this paper, we have described an integrated ontological model, and discussed its implementation using two applications; DMS and MAS of our university. The novelty of the proposed work is that no work in the integration of CSCW has been done to our best knowledge with the exception of those, which are quoted. Our further work will include detailed development and further evaluation of the framework.

9. REFERENCE

[1] Benford, S., J. Mariani, L.Navarro, W.Prinz and T.Rodden, (1993): 'MOCCA: An Environment for CSCW Applications', ACM Organizational Computing Systems, Milpitas – California, Press, PP.172-77.

[2] Dewan, P., and Sharma, A., (1999): ""An experiment in Interoperating Heterogeneous Collaborative Systems"": In Proc. of Sixth European Conference on Computer Supported Cooperative Work - ECSCW'99, Copenhagen, Denmark.

[3] Navarro, L., Prinz, W., and Rodden, T.,(1993): 'CSCW requires Open Systems', Computer Communications, Vol. 16, No. 5, pp. 288-297.

[4] Li D., Li R., (2002): 'Transparent sharing and interoperation of heterogeneous single-user applications', ACM Conference on Computer Supported Cooperative Work (CSCW'2002), pp. 246-255

[5] LaMarca A., Edwards K.W., Dourish P., Lamping J., Smith, I., Thornton, J., (1999), "Taking the Work out of Workflow: Mechanisms for Document-Centered Collaboration", In Proc. of the 1999 European Conference on Computer Supported Cooperative Work (ECSCW '99).

[6] Iqbal, R., James, A., Gatward, R., (2002): 'A Framework for Integration of CSCW', in Proc. CSCWD02, Computer supported cooperative work in design conference, IEEE, Brazil.

[7] Dewan, P., (1993): "A Survey of Applications of CSCW Including Some in Educational Settings". Proceedings of ED-MEDIA'93, pp. 147-152.

[8] ISO TC184/SC4/WG10N320, (2000): 'Industrial Automation Systems and Integration', Integration of Industrial data for exchange, access, and sharing.

[9] ISO TC184/SC4/WG10N342, (2002): 'Industrial Automation Systems and Integration', Integration of Industrial data for exchange, access, and sharing.

[10] Malone, T. W. and Crowston, K.(1990): 'What is Coordination Theory and how can it help design cooperative work systems' ACM Conference on Computer Supported Cooperative Work (CSCW'90).

[11] Kuutti, K.: (1991): 'The concept of activity as a basic unit of analysis for CSCW research', in Proc. of the Second European Conference on Compute Supported Co-operative Work (ECSCW'91), pp.249-264.

[12] Kreifel ts, T., Hinrichs, E. and Woetzel, G.: (1993): 'Sharing To-Do Lists with a Distributed Task Manager', in Proc. of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93), pp.31-46.

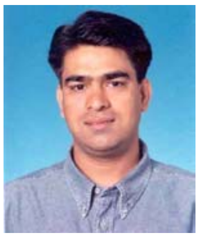
[13] Fitzpatrick, G., Tolone, W.J. and Kaplan, S.: M. (1995) 'Work, Locales and Distributed Social Worlds', in Proc. of the 1995 European Conference on Computer Supported Cooperative Work (ECSCW '95).

[14] Teege, G.: (1996.): 'Object-Oriented Activity Support: A Model for Integrated CSCW Systems. Computer Supported Cooperative Work (CSCW)', The Journal of Collaborative Computing, 5(1), pp. 93-124.

[15] Farias, C. R. G., Pires, L. F., and Sinderen, M. van: (2000a):'A Conceptual model for the development of CSCW systems'. Fifth International conference on the design of cooperative systems (COOP 2000), Sophia Antipolis, pp.189-204.

[16] Farias, C. R. G., Pires, L. F., and Sinderen, M. van: (2000): 'A component-based groupware development methodology'. In Proc. of the 4th Intl. Enterprise Distributed Object Computing Conf., Makuhari, Japan.

[17] Batini C., Lenzerini M., and Navathe S.B., C., (1986): 'A comparative analysis of methodologies for database schema integration', ACM Computer Survey, Vol 18, No.4, 323-364.



Rahat Iqbal *qualified with a Master of Science degree in IT/Computer Science from the University Science Malaysia (USM) in 1999. He started his career as a research assistant in the school of communication in USM but it was only a short span. During this time, he was also employed by IOMEGA; the Multi-National Computer Hardware Company. After three months, he joined INTI College Malaysia as a lecturer. Rahat is now pursuing his PhD in School of Mathematical and Information Sciences, Coventry University. He is also teaching a range of subjects from information systems to computer science in the University. His research mainly focuses on integration of CSCW including ethnography. His research interests also include HCI, Database, Software Engineering, System Development Methodologies & Management and Grid Computing.*

Anne James *obtained her BSc from Aston University in 1980 and PhD from CNNA, UK in 1986. Since then she has been working in academia specialising in databases, data modelling and information systems. She is currently Associate Head of Computer Science at Coventry University During her career Anne has published around 60 academic papers in journals or at conferences. She has also been a member of various national and international programme committees and has edited a number of conference proceedings. As well as this, Anne has maintained an interest in her profession through activities with the British Computer Society of which she is a member.*

Richard Gatward *became involved in academic research in 1985 by undertaking a full-time MSc course in Information Systems Engineering, the project component being in speech recognition at LUCHI, in the computer studies department at Loughborough University. This established a long-term interest in artificial intelligence in general, and computational linguistics in particular.*

Following this, a post as research assistant at Aston University led directly to an appointment at Coventry University (then polytechnic) where teaching and research have concentrated on Knowledge Based Systems in general.

Current research interests have expanded into educational technology. This has also led to an increased interest in teaching and learning issues and being made leader of the Education Research Group within the Computing subject group in the school.