What is groupware?

Software *specifically* designed

- to support group working
- with cooperative requirements in mind

NOT just tools for communication

Groupware can be classified by

- *when* and *where* the participants are working
- the *function* it performs for cooperative work

Specific and difficult problems with groupware implemention
The Time/Space Matrix

Classify groupware by:
- **when** the participants are working, at the same *time* or not
- **where** the participants are working, at the same *place* or not

<table>
<thead>
<tr>
<th></th>
<th>same place</th>
<th>different place</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>same time</strong></td>
<td>face-to-face conversation</td>
<td>telephone</td>
</tr>
<tr>
<td><strong>different time</strong></td>
<td>post-it note</td>
<td>letter</td>
</tr>
</tbody>
</table>

Common names for axes:
- **time**: synchronous/asynchronous
- **place**: co-located/remote
Classification by Function

Cooperative work involves:

**Participants** who are working

**Artefacts** upon which they work

What interactions does a tool support?

- **computer-mediated communication**
  - direct communication between participants

- **meeting and decision support systems**
  - common understanding

- **shared applications and artefacts**
  - control and feedback with shared work objects
Email and bulletin boards

asynchronous/remote
familiar and most successful groupware

Recipients of email:
   direct in To: field
   copies in Cc: field
delivery identical — difference is social purpose

differences between email and BBs
fan out
   one-to-one — email, direct communication
   one-to-many — email, distribution lists
                  BBs, broadcast distribution

control
   sender — email, private distribution list
   administrator — email, shared distribution list
   recipient — BBs, subscription to topics
Structured message systems

*asynchronous/remote*

- 'super' email — cross between email and a database
- sender fills in special fields
- recipient filters and sorts incoming mail based on field contents

---

Type: Seminar announcement  
To: all  
From: Alan Dix  
Subject: departmental seminar  

Time: 2:15 Wednesday  
Place: D014  
Speaker: W.T. Pooh  
Title: The Honey Pot  
Text: Recent research on socially constructed meaning has focused on the image of the Honey Pot and its dialectic interpretation within an encultured hermeneutic. This talk ...

but, work by the sender ... benefit for the recipient

**conflict**

*global structuring* by designer  
*vs.*  
*local structuring* by participants
Video conferences and communication

*synchronous/remote*

Technology emerging: ISDN + video compression

major uses:
- video conferences
- pervasive video for social contact
- integration with other applications

often cheaper than face-to-face meetings

(telecommunications costs vs. air flights)

but not a substitute:
- small field of view
- lack of reciprocity
- poor eye contact

One solution for lack of eye contact — the video-tunnel

![Diagram of video-tunnel setup](image-url)
Meeting and decision support systems

In design, management and research, we want to:
- generate ideas
- develop ideas
- record ideas

primary emphasis — common understanding

Three types of system:

- **argumentation tools**
  - *asynchronous co-located*
  - recording the arguments for design decisions

- **meeting rooms**
  - *synchronous co-located*
  - electronic support for face-to-face meetings

- **shared drawing surfaces**
  - *synchronous remote*
  - shared drawing board at a distance
argumentation tools

asynchronous co-located

hypertext like tools to record design rationale

Two purposes:
- reminding the designers of the reasons for decisions
- communicating rationale between design teams

Mode of collaboration:
- very long term
- sometimes synchronous use also

Example: gIBIS (issue based information system)

various node types including:
- issues e.g., ‘number of mouse buttons’
- positions e.g., ‘only one button’
- arguments e.g., ‘easy for novice’

linked by relationships such as:
- argument supports position
  e.g., ‘easy for novice’ supports ‘only one button’
Meeting rooms

synchronous co-located

electronic support for face-to-face meetings

- individual terminals (often recessed)
- large shared screen (electronic whiteboard)
- special software
- U or C shaped seating around screen

Various modes:
  brainstorming, private use, WYSIWIS

WYSIWIS — ‘what you see is what I see’
  all screens show same image
  any participant can write/draw to screen
Typical meeting room

shared screen
Issues for cooperation

Argumentation tools

concurrency control
  two people access the same node
  one solution is node locking

notification mechanisms
  knowing about others’ changes

Meeting rooms

floor holders  one or many?

floor control policies
  who can write and when?
  solution: locking + social protocol

group pointer
  for deictic reference (this and that)
Shared work surfaces

*synchronous remote*

At simplest, meeting rooms at a distance, but ...  

- additional audio/video essential for *social protocols* and discussion  
- network delays can be major problem

Additional special effects:

- participants write onto large video screen problems with *parallax*  
- shadow of other participant’s hands appears on screen  
- electronic image integrated with video and paper images

**Example:** TeamWorkStation  
remote teaching of Japanese calligraphy  
student’s strokes on paper overlaid with video of instructor’s strokes
Shared Applications and Artefacts

Compare purpose of cooperation:

**meeting rooms and decision support systems**
— develop shared understanding

**shared applications and artefacts**
— work on the same objects

technology similar but primary purpose different

many different modalities (time/space matrix)

- **shared windows** — *synchronous remote/co-located*
- **shared editors** — *synchronous remote/co-located*
- **co-authoring systems** — largely *asynchronous*
- **shared diaries** — largely *asynchronous remote*
- **shared information** — any, but largely *asynchronous*

synchronous remote applications usually require additional audio/video channel
Similar – but different

Shared PCs and shared window systems

• Multiplex keyboard and screen

• Individual applications *not collaboration aware*

• Floor control problems:
  - user A types: ‘interleave the’
  - user B types: ‘keystrokes’
  - result: ‘inkeytersltreaokeve tshe’

Shared editors

• An editor which is *collaboration aware*

• One document — several users

• Similar to shared screen in meeting room . . .
  . . . with similar floor control problems!

• Additional problem — multiple views
Shared editors — multiple views

We will look at some of the options and how they affect the style of cooperation.

Thinking about the shared view vs. different view options, it at first seems obvious that we should allow people to edit different parts of a document.

This is certainly true while they are working effectively independently.

More adaptable systems are needed to allow for the wide variation between groups, and within the same group over time.

We will look at some of the options and how they affect the style of cooperation.

Thinking about the shared view vs. different view options, it at first seems obvious that we should allow

Options:

same view or different view

single or separate insertion points

Single view $\Rightarrow$ scroll wars

Multiple views $\Rightarrow$ loss of context with *indexicals*

‘I don’t like the line at the top’

‘but I just wrote that!’
Co-authoring systems

Emphasis is on long term document production, not editing

Two levels of representation

- the document itself
- annotation and discussion

Often some form of hypertext structure used

Similar problems of *concurrency control* to argumentation systems

Sometimes include *rôles*:

- author, commentator, reader, …
- but who decides the rôles?
- and how flexible are they?
Shared diaries

Idea:

- make diaries and calendars more easily shared
- allow automatic meeting scheduling etc.

Issues for cooperation:

privacy who can see my diary entries?
control who can write in my diary?

Similar to file sharing issues, but need to be lightweight

Many systems have failed because they ignored these issues
Communication through the artefact

When you change a shared application:

- you can see the effect — feedback
- your colleagues can too — feedthrough

feedthrough enables

communication through the artefact

Not just with ‘real’ groupware

Shared data is pervasive:

- shared files and databases
- casework files (often non-electronic)
- passing electronic copies of documents
- passing copies of spreadsheets

Often need direct communication as well, but indirect communication through the artefact central

Few examples of explicit design for cooperation.

Liveware is an exception,

a database with ‘merging’ of copies
Time/space matrix revisited

- **co-located**
  - meeting rooms
  - shared work surfaces and editors
  - shared PCs and windows
  - argumentation tools

- **remote**
  - video conferences, video-wall, etc.
  - email and electronic conferences
  - co-authoring systems, shared calendars

# synchronous

# asynchronous
### Mobile workers and home workers have infrequent communication — they require *unsynchronised* groupware

Few ‘research’ systems address this area

NO current system allows fluid movement between synchronised/unsynchronised operation
Shared information

Granularity of sharing

chunk size
small — edit same word or sentence
large — section or whole document

update frequency
frequent — every character
infrequent — upon explicit ‘send’

level of sharing

output: shared object
shared view
shared presentation

input: single insertion point

• shared virtual keyboard

multiple insertion points

• other participants visible
• group pointer
• no visibility
Levels of shared output

### Levels of shared output

#### Presentation

<table>
<thead>
<tr>
<th>houses</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>23</td>
<td>79</td>
</tr>
<tr>
<td>51</td>
<td>123</td>
</tr>
</tbody>
</table>

#### View

```sql
select houses, population from VILLAGE_STATS
where population < 200
sort by houses ascending
```

#### Object

<table>
<thead>
<tr>
<th>village</th>
<th>houses</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burton</td>
<td>23</td>
<td>79</td>
</tr>
<tr>
<td>Marleigh</td>
<td>339</td>
<td>671</td>
</tr>
<tr>
<td>Westfield</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>Thornby</td>
<td>51</td>
<td>123</td>
</tr>
</tbody>
</table>

VILLAGE_STATS
Integrating communication and work

Added: *deixis* — reference to work objects
*feedthrough* — for communication through the artefact

Classified groupware by function it supported

Good groupware — open to all aspects of cooperation

e.g., annotations in co-authoring systems
embedding direct communication

bar codes — form of deixis
aids diffuse large scale cooperation
Architectures for groupware I

Client-server architecture

Feedback and network delays

At least 2 network messages + four context switches
With protocols 4 or more network messages
Architectures for groupware II

Different architectures:

**centralised** — single copy of application and data

*client-server* — simplest case

N.B. opposite of X windows client/server

*master-slave* special case of client-server

N.B. server merged with one client

**replicated** — copy on each workstation

also called *peer-peer*

+ local feedback

− race conditions

Often ‘half way’ architectures:

- local copy of application
- central database
- local cache of data for feedback
- some hidden locking
Shared window architecture

- Non-collaboration aware applications
  \[\Rightarrow\] client/server approach
  corresponding feedback problems

- no ‘functionality’ — in the application
  but must handle floor control

![Diagram of shared window architecture]

- User 1
- User 2
- User n
- User stub 1
- User stub 2
- User stub n
- Application stub
- Application
- Xevents
- Xlib calls
- Xevents
- Xlib calls
Feedthrough

Need to inform all other clients of changes

Few networks support *broadcast* messages, so . . .

\[ n \text{ participants } \Rightarrow n - 1 \text{ network messages!} \]

Solution: increase granularity

reduce frequency of feedback

but . . .

poor feedthrough \( \Rightarrow \) loss of shared context

Tradeoff: timeliness vs. network traffic

Graphical toolkits

Designed for *single* user interaction

Problems for groupware include

- pre-emptive widgets
  
  \( \text{(e.g., pop-up menus)} \)

- over-packaged text
  
  \( \text{(single cursor, poor view control)} \)

*notification* based toolkits with *callbacks* help (see Ch. 10)
Robustness and scaleability

crash in single-user interface — one sad user

but, groupware complex: networks, graphics etc.

• network or server fails — standard solutions

• client fails — three ‘R’s for server:
  - robust — server should survive client crash
  - reconfigure — detect and respond to failure
  - resynchronise — catch up when client restarts

• errors in programming
  - defensive programming
  - simple algorithms
  - formal methods

• unforeseen sequences of events
  - deadlock — never use blocking I/O
  - never assume particular orders
  - network packet ≠ logical message

Scaling up to large numbers of users?

Testing and debugging: hard!