Chapter 5: Groupware Architectures
Winter Semester 2009/2010

Lecturer:
Dr.-Ing. Stefan Werner
University Duisburg-Essen
Faculty of Engineering
Institute of Computer Engineering

Content

1. Introduction to CSCW and Software Engineering
2. Groups and Group Processes
3. Basic Concepts of CSCW and Groupware
4. Graphical User Interfaces and Awareness
5. Groupware architectures
   5.1 Design Principles for Groupware
   5.2 Reference Models
   5.3 Architectural Styles for Groupware
   5.4 Distribution Architectures
7. Virtual Teams
8. Virtual Teams and Communication
9. Cooperative Software Engineering

CSCW and Software Engineering
Winter Semester 2009/2010
Lecturer: Dr.-Ing. Stefan Werner
University Duisburg-Essen
Faculty of Engineering
Institute of Computer Engineering

University of Duisburg-Essen
Faculty of Engineering
Institute of Computer Engineering
Prof. Dr.-Ing. Axel Hunger

CSCW and Software Engineering
Winter Semester 2009/10
Lecturer: Dr.-Ing. Stefan Werner
University of Duisburg-Essen
Faculty of Engineering
Institute of Computer Engineering
Prof. Dr.-Ing. Axel Hunger

University of Duisburg-Essen
Faculty of Engineering
Institute of Computer Engineering
Prof. Dr.-Ing. Axel Hunger

1 of 47
5. Groupware architectures
5.1 Design Principles for Groupware

Modelling Groupware

- Groupware-applications are in general distributed applications, composed of different components distributed amongst several computers.
- The fragmentation of Groupware into components is described by the so called **Architecture model/style**
- The distribution of the components amongst several computers is described upon the basis of **Distribution Architectures**.

- According to Roth [Rot00] the design of a Groupware-application follows in accordance with an **Architecture Model**
- **Distribution Architectures** are used for implementation
5.1 Design Principles for Groupware
- Architectural Views -

**distribution architectures**
- represent the distribution of system functionality across connected computing platforms.

**reference models**
- divide complete systems into named functional elements
- specify data flow between those elements

**architectural styles**
- Prescription of components, and connector types and their allowed patterns of interaction

---

5. Groupware architectures
5.2 Reference Models

- are intended to describe the structure of complete groupware systems at a relatively abstract level.
- specify the complete structure of some class of system at a relatively large granularity.
- allow a wide range of existing systems to be easily mapped onto it
- provide useful insights into the properties of the systems.

**Key Idea:**
- an appropriate decomposition of an interactive system includes a clean separation between the application's underlying logic or functional core and its user interface.
- separation allows development of core and interface to be largely decoupled
Chapter 5: Groupware Architectures

5. Groupware Architectures

5.2 Reference Models

The Seeheim Model

- was developed in 1985 at a working group in Seeheim, Germany
- the functional core is allocated to the application component
- the user interface is divided into a
  - presentation component handles lexical aspects of the interaction,
  - a dialogue controller handles syntactic aspects and
  - an application interface.

is an adaptor, mapping between application-level concepts and dialogue concepts to reduce the coupling between functional core and user interface.

Patterson’s Taxonomy

- motivated by the observation that the primary challenge for synchronous groupware applications is to maintain the shared state.
- divides application state into four levels:
  - file state, the persistent representation of the model.
  - model state, the underlying data itself;
  - view state, a logical visual representation of the underlying data;
  - display state, which is implemented in the video hardware that drives the user's physical display;
- leaves all computational aspects of the application unspecified.
Chapter 5: Groupware Architectures

5.2 Reference Models

Patterson’s taxonomy differentiates the following model variations:

- shared states
- synchronised states
- a combination of shared and synchronised states.

**Shared state architecture**

- all levels above the first shared state are assumed to be shared as well.
- views and displays are separate, providing users with independently constructed views of common application data.
5.2 Reference Models
Patterson’s Taxonomy

Synchronized state architecture

- shared states are implemented using a synchronization architecture
- synchronized elements
  - are intended to contain exactly the same information
  - are replicated for performance or other pragmatic reasons.
- Patterson suggests that where views are synchronized it is also necessary to separately synchronize models.

Hybrid architecture

- Mixture:
  - the model is shared
  - the views are synchronized.
- This architecture provides
  - a simple mechanism for ensuring consistency of the model state
  - allows view sharing to be switched on and off as desired by the users.
5.2 Reference Models
Patterson's Taxonomy

Weaknesses of Patterson's Taxonomy

• No possibilities for modelling continuous data-streams
• No shared display states (e.g. Telepointer), which are not explicitly modelled on the model level
• A generalisation of the architecture model according to Patterson was introduced in 1996 by Dewan [Dew96].

=> In his model neither the number of model levels nor their meaning is fixed.

5. Groupware architectures
5.3 Architectural Styles for Groupware

• present a solution to a particular problem in terms of a vocabulary of components, connectors, and the allowed relationships among them
• a system may be completely implemented in an architectural style, or
• a variety of styles can appear in the design of a single system
5.3 Architectural Styles for Groupware
The MVC-Model

Functionality (1/2):

**Model:** all programme parts, composed of data structures and algorithms, that are not connected with the display.

**Controller:** user inputs, e.g. via mouse click or the keyboard.

**View:** functions for display of data (model).

- The Controller-View-Pair is normally part of the user interface and accesses exactly one Model.
- Several, different Controller-View-Pairs can have access to the same Model ⇒ implementation of different views.

It is also possible that the user input has a direct effect on the display, for example in order to scroll the screen. In this case the Model will not be changed.

Functionality (2/2):

The Controller reads the user input and forwards it to the Model (1).

The Model sends information to the Views (and Controller) that it has been changed (2).

The Views on the other hand ask for the individual changes (3) and updates its display.

It is also possible that the user input has a direct effect on the display (4), for example in order to scroll the screen. In this case the Model will not be changed.
5.3 Architectural Styles for Groupware

The Net-MVC-Model

- **Model** and **Controller-View-Pairs** are distributed in the Network
- The **Model** is implemented on the Server-computer
- **Controller-View-Pairs**
  - are summarised in the **User interfaces**
  - are located on Client-computers.
- The **Model** and the **User interfaces** communicate via their own communication channels

**Functionality:**
- The distribution of **Model** and **User interface** in the Network leads, in comparison to assignment onto one computer, to a reduction in speed with regard to communication.
  => Model transmits, not only the message, that it has been changed, but also directly in the first steps, the data required for a change in the display.
- Data are collected by the Proxy-Objects.
5.3 Architectural Styles for Groupware
The Net-MVC-Model

Functionality:
The Proxy-Objects‘ Duties:
- Intermediate memory
- Management of the model replications

Functionality of the Proxy Objects:
- New/changed model data are first used on the local data model thereafter forwarded to the user interface.
- Entries from the user interfaces are first of all collected in the Proxy-Objects, thereafter forwarded to the Model on the server.
- The Proxy-Object can collect many different requests for changes.

Proxy-Objects
- allow private views
- Increase the robustness of the system against network failures.

5. Groupware architectures
5.4 Distribution Architectures

• describe the run time distribution of system state and computation across computing platforms connected by a network.

Example,
• a groupware application may be implemented entirely on a server with client/server communication at the level of window system events, or
• an application may be replicated at each user's location and the replicas kept in a consistent state using sophisticated consistency maintenance mechanisms.
5.4 Distribution Architectures

Interlace

- a descriptive framework for distributed groupware systems
- represents a groupware system as a collection of
  - users,
  - devices,
  - concurrent processes, and
  - state elements
distributed across
  - interconnected computing platforms or
  - sites.

- Some platforms will have one or more local users; others will operate in a server role.
5.4 Distribution Architectures

Interlace Elements

Interlace diagramms consists of

- physical input devices connected to an input process, which transforms input into logical interface events;
- a chain of one or more update processes,
- which transform interface events into updates on state;
- a chain of one or more view processes,
- which collectively compute an interactive view from the state elements

- a rendering process, which presents the view to the user on physical output devices.
- a consistency maintenance process to ensure that state remains consistent in the face of possibly conflicting updates from multiple users.

In Interlace diagramms each user is supported by one or more input output loops

- through private state

University of Duisburg-Essen
Faculty of Engineering
Institute of Computer Engineering
Prof. Dr.-Ing. Axel Hunger

CSCW and Software Engineering
WS 2009/10
Lecturer: Dr.-Ing. Stefan Werner
23 of 47
5.4 Distribution Architectures

Interlace Elements

In Interlace diagram each user is supported by one or more input output loops

- through private state
- through shared state
- Any element in a diagram can be either shared or private

State sharing may be implemented

- through true sharing (as in the example) or
- by replication with synchronization

Data streams may also be synchronized by means of

- identically ordered streams of data
- in approximate temporal synchrony.
5. Groupware architectures
5.4 Distribution Architectures

The distribution of Groupware components on different computers can be divided into the following classes:

- **Centralized distribution architecture**
  - All elements of the application run/reside on a central computer.

- **Replicated distribution architectures**
  - Separate instance of the application is provided locally for each user.

- **Hybrid distribution architectures (semi-replicated)**
  - Some elements are centralized while others are replicated.

Respectively two variations possible:

- **Collaboration transparent**
  - Single user systems are used in a multi-user environment
  - Synchronisation of user inputs

- **Collaboration aware**
  - Application follows basic concepts of multi-user systems
  - Synchronisation of the states

---

**Centralised Distribution Architectures**

- All elements of the application run/reside on a central computer (server).
- Physical in- and output process on the clients.

---
5. Groupware architectures
5.4 Distribution Architectures

Centralised Distribution Architectures

Benefits
• simplicity
• internal efficiency of the application can be maximized
• state consistency can be guaranteed relatively easily
• good support for late-joiners

Drawbacks
• bandwidth intensive (both variants)
• sensitive to network latencies (both variants)
• poor scalability (collaboration aware variant)

Implementations
• Microsoft Net Meeting (centralized collaboration transparent)
• Rendevouz-System (centralized collaboration aware)

Replicated Distribution Architectures

• Complete copy of the application on each client
• all data and computation is replicated at all sites)

Collaboration transparent
internal state not externally accessible
⇒ state synchronization not generally possible.
⇒ synchronization of input streams

Collaboration aware
synchronized states, allow
• flexibility in selection of concurrency control protocols
• local states and relaxed WYSIWIS
5. Groupware architectures
5.4 Distribution Architectures

Summary: Replicated Distribution Architectures

Benefits
• may use less bandwith than equivalent centralized architectures
• enhanced interface responsiveness (collaboration aware)
• Better scalability (compared to centralized) as view and update processes are distributed to the users’ computers

Drawbacks
• require more aggregate resources
• realization of distributed cm is of high complexity
• problematic for late joiners

Hybrid Distribution Architecture
• some aspects (computation, state) are replicated while others are centralized
• The advantages of a replicated architecture can be used when the consistency maintenance components are centralised.
5. Groupware architectures
5.4 Distribution Architectures

**Summary: Hybrid Distribution Architecture**
provides a mix of the benefits and liabilities of the centralized and replicated architectures.

**Collaboration transparent hybrid applications**
- are more flexible than the centralized architecture
- latecomers are better supported than the with fully replicated architecture.

**Collaboration aware hybrid applications**
- generally scale better than centralized architecture
- are simpler to develop

**principal liability**
- responsiveness of the user interface may be impacted by network latencies

---

5. Groupware architectures
5.4 Distribution Architectures

**Centrally Coordinated Distribution Architectures**
is similar to the fully replicated architecture except that the consistency maintenance process is centralized

- **collaboration transparent variant** is directly comparable to its fully replicated counterpart
- **collaboration aware variant**, is different in principle from its fully replicated counterpart

**Collaboration transparent**

**Collaboration aware**
5. Groupware architectures
5.4 Distribution Architectures

Summary: Centrally Coordinated Distribution Architectures

- shares most of the benefits and liabilities of the fully replicated architecture.
- **primary benefit** of adding central coordination to a replicated architecture is simplicity of implementing consistency maintenance
- **additional liability**
  - system's reliance on a single consistency maintenance server
  - user interface response will generally be poorer than that of a purely replicated application with optimistic concurrency control.

3.8 Groupware architectures
3.8.4 Distribution Architectures

Summary: Replicated Distribution Architectures

**Benefits**
- may use less bandwidth than equivalent centralized architectures
- enhanced interface responsiveness (collaboration aware)
- Better scalability (compared to centralized) as view and update processes are distributed to the users’ computers

**Drawbacks**
- require more aggregate resources
- problematic for late joiners
Exercise

• Analyse the distribution architecture of the synchronous groupware PASSENGER

Solution

⇒ full application running at all clients
Exercise

• Analyse the distribution architecture of the synchronous groupware PASSENGER

Solution

⇒ full application running on all clients
Solution

⇒ private and shared states

⇒ how are these states implemented?

⇒ shared states at server-site

⇒ Replicated, collaboration aware architecture with central coordination as discussed during lectures?

⇒ private-states at clients-site

University of Duisburg-Essen
Faculty of Engineering
Institute of Computer Engineering
Prof. Dr.-Ing. Axel Hunger

CSCW and Software Engineering
WS 2009/10
Lecturer: Dr.-Ing. Stefan Werner
41 of 47
Solution

⇒ how are private and shared states implemented?

⇒ Replicated, collaboration aware architecture with central coordination as discussed during lectures?

How are data flows synchronized?
What is payload of dataflows?

Multimedia Data

Model data are exchanged always as full model
⇒ Primary copy of model/shared state at server site
⇒ Replications at clients site
Hybrid Distribution Architecture with primary copy

What about cm?

Passenger uses floor control with explicit and exclusive permissions to the full floor?

Floor requests are send to server => centralized coordination
Chapter 5: Groupware Architectures

The PASSENGER Distribution Architecture Solution

Hybrid Distribution Architecture with primary copy and centralized coordination

University of Duisburg-Essen
Faculty of Engineering
Institute of Computer Engineering
Prof. Dr.-Ing. Axel Hunger

CSCW and Software Engineering
WS 2009/10
Lecturer: Dr.-Ing. Stefan Werner
47 of 47