

A taxonomy for synchronous groupware architectures

Jörg Roth

University of Hagen, Department for Computer Science, 58084 Hagen, Germany

Abstract. In this paper we present a taxonomy for distribution architectures of synchronous groupware. As demonstrated by our own groupware platforms DreamTeam and QuickStep, a certain architecture may be a suitable for one scenario, but may have many disadvantages in another scenario. Former taxonomies divide distribution architectures into *centralized*, *replicated* and *hybrid* ones. Such a classification is not expressive enough to argue about pros and cons of a specific realisation, thus in the past discussions about distribution architectures were very emotional. Discussions about *best* architectures are going on. A taxonomy with stronger expressiveness would help to clear up this discussion.

Our taxonomy distinguishes five basic architectures and several subtypes. It allows discussing aspects of distribution architectures in detail. We used this taxonomy to classify some groupware platforms. A list of criteria at the end of this paper demonstrates the influence of specific architectures on groupware usage.

1 Introduction

Synchronous groupware brings together users which are geographically distributed and connected via a network. There exist a variety of platforms which relieve the groupware developer from struggling with standard problems like network details, synchronization algorithms, etc., and allow him or her to concentrate on application-specific details. A significant distinctive feature of groupware is the distribution architecture, which defines, which parts of a groupware application run on a central server, which parts run on decentral sites and how sites are logically linked which each other. Choosing a specific distribution architecture has an important influence on how groupware can be developed and used. The distribution architecture has affects on several topics, to mention some: application's response time and fault tolerance can be improved using decentralized architectures. In general, decentralised architectures scale much better than centralized. On the other hand, some runtime services such as storing documents can be developed much easier, if a distribution architecture with a central server is used.

Many designers of groupware focus on the distribution architecture when presenting their platforms. Unfortunately the terms defining a specific distribution architecture are extremely vague. The classical approach to describe distribution architectures distinguishes between the three types *centralized*, *replicated* and *hybrid*. Meanwhile several platforms exist which offer different distribution architectures. Unfortunately, the above classification into three types allows no detailed argumentation about advantages and disadvantages of specific architectures. Most of the newer platforms, which support collaboration aware applications, have to be summarized under the architecture *hybrid*, though there exist big differences between them.

The problem becomes even worse, when we take different services of groupware platform into account. There are different stages of collaboration, which have to be supported by a platform, e.g. preparing a session, joining a session, running the collaborative application, storing persistent artefacts. Each stage may have its own distribution architecture, e.g. an artefact can be stored centrally on a server, but the collaborative application runs replicated. Assigning one single distribution architecture (e.g. hybrid) to a specific platform means blurring the details.

2 Two groupware platforms

Before we introduce our taxonomy, we briefly present the two example groupware platforms DreamTeam and QuickStep.

The first example *DreamTeam* [Roth00] (figure 1a) is a platform for developing collaboration-aware groupware application for desktop scenarios. We developed DreamTeam for distance education scenarios, thus document editing and web browsing are typical DreamTeam applications. DreamTeam takes into account the slow bandwidths of usual students' network connections.

QuickStep [RU00b] (figure 1b) on the other hand is a platform specially designed for groupware applications running on handheld devices. The fundamental different nature of handheld devices compared to desktop computers has a great impact on the platform, e.g. resource limitations have to be considered. During collaboration, the handheld could be switched off because of the auto-power-off mechanism. Often, personal data are stored on handheld devices, thus mechanisms have to ensure privacy.

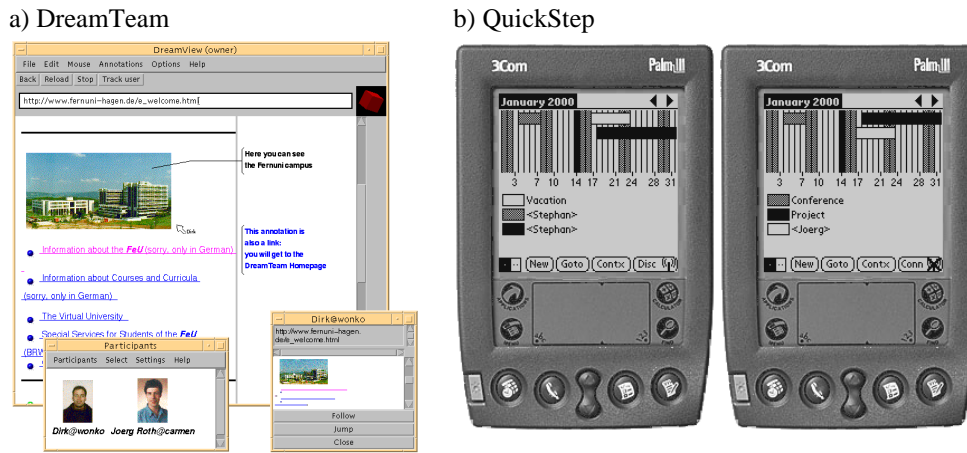


Figure 1: Screenshots of very different groupware applications

Both platforms are based on very different distribution architectures. In DreamTeam states are stored decentrally, thus an application can access its corresponding data locally without network delay. Only for updates, network transactions are necessary. In addition, avoiding a central server, several costs are saved, e.g. for administration and for setting up a fail-save hardware.

Handhelds running QuickStep applications are not capable to store huge amounts of data, thus a central server is unavoidable. In addition, some technical reasons lead to a central architecture, e.g., some handheld operating systems do not support threads or processes for background tasks. Building applications without threads leads to a completely different architecture.

These examples show that a certain architecture may be a suitable for one scenario, but may have many disadvantages in another scenario. To choose the *suitable* architecture, the specific development and usage scenarios have to be taken into account. To discuss pros and cons of architectures in more detail, a new taxonomy has to be introduced, which provides a richer expressiveness than the *centralized-replicated-hybrid* classification.

3 A taxonomy

In the following, we present a taxonomy solely designed for describing distribution architectures [RU00]. The taxonomy consists of two parts: the application scheme and the distribution scheme. The application scheme defines the components of groupware applications and the distribution scheme defines, how these components can be distributed among sites.

Figure 2 shows the application scheme. A groupware application according to this taxonomy consists of three components:

- The *application core* presents the application's function. It is divided into the components *functional core* and *presentation*.
- The *window system* displays the application's windows and receives user events from e.g. from keyboard and mouse.
- The *coordination* is responsible for running the application in a distributed environment. Coordination tasks are, e.g., synchronisation of user input, concurrency control, floor control, etc.

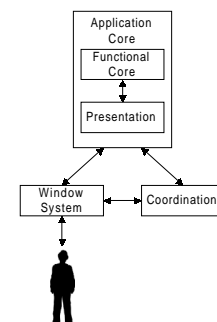


Figure 2: The application scheme

Links between components indicate data flow. In contrast to Patterson's taxonomy [Patt94], there is no *file* component. Persistence aspects, if any are subsumed under the functional core.

This application schema was influenced by the single user model Arch [UIMS92]. Arch components correspond to components of the application scheme (e.g. Arch's functional core and domain adapter correspond to the functional core). There is no counterpart in Arch to our coordination component, since the Arch model was designed for single-user applications only.

In the following, we use this taxonomy according to the following set of rules:

- Not all links between components need to exist.

- The coordination component is not always necessary. In such a case, coordination tasks are performed by the application core component.
- The window system is left out, if it has no special meaning inside a distribution architecture. In these cases, users and presentation are linked together.
- Depending on the architecture, the internal components of application cores may be hidden. If on the other hand internal components are displayed, the application core's frame can be left out to clarify the diagram.

The components of an application scheme can be distributed according to a distribution scheme. Figure 3 shows the basic distribution schemes:

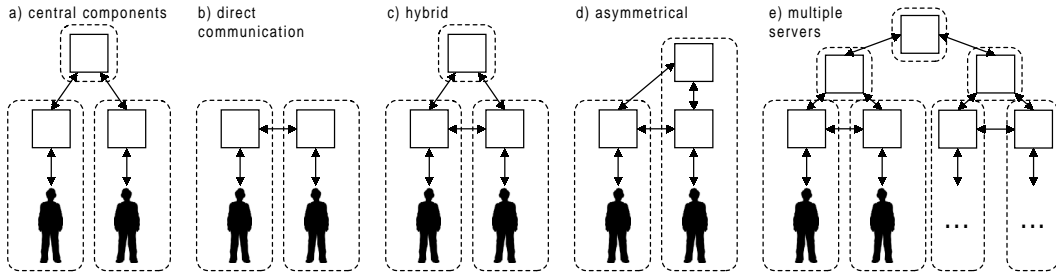


Figure 3: Basic distribution types

A rectangle presents one or more components described in the application schemes. Dotted frames indicate different sites inside a network. In the following, sites which are assigned to a user are called *peers*, other computers are called *servers*. Components hosted by peers are called *decentralized components*, components hosted by servers are called *centralized components*. Corresponding decentralized component are called *replica*.

The following basic distribution architectures exist in our taxonomy:

- Architectures with centralized components*: these architectures have at least one centralized component. Peers are not connected to each other.
- Architectures with direct communication*: these architectures have no central component at all. All peers are connected to each other.
- Hybrid architectures*: these architectures have at least one central component and allow direct communication between peers.
- Asymmetrical structures*: have no central component, but distribution of components among peers is not symmetrical, i.e. at least one peer component has no replica.
- Multiple servers*: these architectures use more than one server, i.e. central components are distributed to more than one site.

To reference a certain distribution architecture, we introduce the following abbreviations:

- An uppercase letter indicates the basic distribution types: **C**: central components, **D**: direct communication, **H**: hybrid, **A**: asymmetrical, **M**: multiple servers.
- A digit (**1**, **2** or **3**) indicates a subtype.
- Lowercase letters indicate different variations of a subtype.

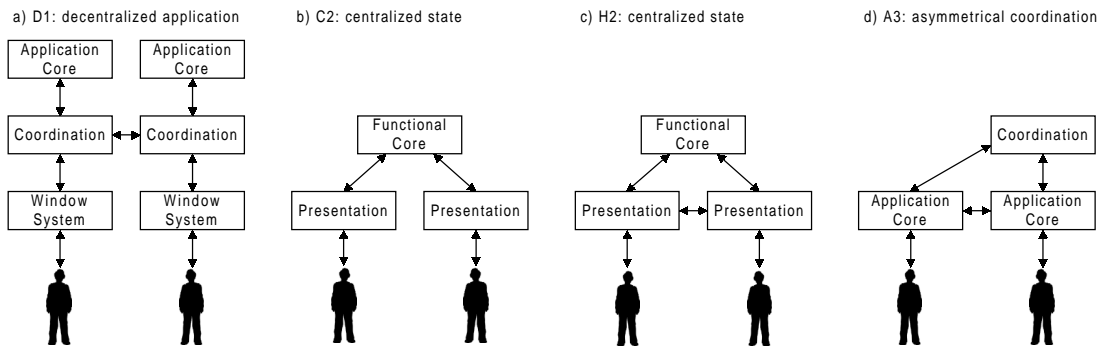


Figure 4: Some examples

A specific distribution architecture could be referenced, e.g. by the abbreviation **H2b**. Same subtype numbers across different basic type indicate resemblance. E.g. architectures **C2**, **D2** and **H2** have similar characteristics, although they were derived from different basic types.

Combining the application scheme with all conceivable distribution schemes leads to a total amount of 2624 variations. Fortunately, only a small amount of variations are sensible. Figure 4 shows four of them.

4 Discussion

The discussion about the *best* distribution architecture for synchronous groupware has a long tradition. Often distribution architectures are discussed more fiercely and in more detail than architectural styles of groupware applications themselves, such as MVC, ALV or PAC*. Sometimes, distribution architecture have made the crucial point of a groupware platform. Most distribution architectures have advantages as well as disadvantages. In our opinion, there is no *best* architecture, it's rather a question of how a specific groupware is to be used. In the following, we give a list of criteria to assess a specific architecture.

Technical criteria

Network related criteria have great influence on the several real-time aspects of a groupware environment. In scenarios with world-wide operating peers or slow network links (e.g. modems), D architectures are more suitable than C or H architectures. If, on the other hand, all sites are placed inside a LAN, network delays can be neglected; thus C architectures have advantages, because they lead to much simpler algorithms. Sometimes, a specific network infrastructure requires a distribution architecture. If, e.g., multicast services are based on native multicast, an architecture with direct connected sites should be used.

Organizational criteria

It is always difficult to introduce groupware to a team which has already formed its organizational structure, thus additional demands by the groupware should be avoided. E.g. C or H architectures require a central server, which has to be installed and administered. Some small departments or companies have no administrator, thus central servers would cause organizational problems. In addition, central servers are a cost factor. Normally servers are expensive because they have to be fail proof and perform a high throughput. From the view point of costs, D and A architectures are more suitable. Additional costs can arise from the network infrastructure. If, e.g., sites are connected to the Internet via a service provider, one could be charged according to the amount of network packages sent or received. Architectures based on a windows system transfer a huge amount of data between sites, thus would cause high communication costs.

Criteria related to groupware development

The easiest way to integrate an application into a group scenario is to use applications of-the-shelf and avoid expensive developments. For collaboration-aware applications, some specific development cannot be avoided. With the help of groupware platforms, the development cycle can be shortened. In addition, the choice for a specific distribution architecture can affect development costs. Architectures with centralized states allow to handle data management much easier than architectures with replicated states. Having data stored on a central server allows to keep track on all data manipulation, thus debugging becomes easy. A log can store all state changes for a certain time (e.g. some days), and thus makes it possible to analyse consistency problems. From the view of consistency, distribution architectures with central components lead to easier algorithms than architectures with direct communication.

This list of criteria shows that a choice for a distribution architecture is strongly influenced by the scenario to be realised. A certain architecture may be a suitable for one scenario, but may have many disadvantages in another scenario.

References

- [Patt94] Patterson J. F., A Taxonomy of Architectures for Synchronous Groupware Applications, in Proceedings of the CSCW'94 Workshop on Software Architectures for Cooperative Systems, Chapel Hill, North Carolina, 1994
- [Roth00] Roth J.: *'DreamTeam': A Platform for Synchronous Collaborative Applications*, AI & Society, Vol. 14, No. 1, Springer Verlag London, March 2000, 98-119
- [RU00] Roth J.; Unger C.: *An extensible classification model for distribution architectures of synchronous groupware*, in Dieng R. et al. (eds): Fourth International Conference on the Design of Cooperative Systems, Sophia Antipolis (France), 23.-26. Mai 2000, IOS Press, 113-127
- [RU00b] Roth J.; Unger C.: *Using handheld devices in synchronous collaborative scenarios*, Second International Symposium on Handheld and Ubiquitous Computing 2000 (HUC2K), Bristol (UK), 25.-27. Sept. 2000, in print
- [UIMS92] The UIMS Tool Developers Workshop, A metamodel for runtime architecture of an interactive system, SIGCHI Bulletin 24(1), 1992, 32-37