Collaboration in the independent architectural office

Werner Lonsing

ecaade03@lonsing.com

Abstract: Collaborative work in architecture is commonly concentrated on the design process. Design teams and their members are working together on multiple virtual model from different, mostly remote locations. Internet- based collaboration software offers a project management platform for comprehensive networking. Complex projects can be better coordinated and documented,and executed even faster.

So in the building process at least two different kinds of collaboration can be noticed, collaboration in the design process where the architects are maintaining their modeling informations, and the construction process where the data is maintained be external software companies. Here another model is suggested. Hosting project data in their own building and so maintaining the physical representations of all project informations is the only way retaining control.

keywords: Collaborative Design, Communication, Database Systems

Introduction

Collaborative work, especially theoretical work in architecture is commonly concentrated on the design process. Design teams and their members are working together on shared virtual model from different, mostly remote locations. These models are very attractive. In such a distributed design situation some members in the team are directly responsible for the model-data, because they have access, while others are only contributing to the model. There are no further distinctions, especially in kind of access. All contributors are part of one design-team and contributions are applying to the shared model. Therefor decisions thought of to be collective. This is a common situation in design environment.

In later stages collaboration becomes a decisive part of the project management. Even often regarded as similar, this important part of the building process has one significant difference: different users are members of different organiza tions demanding different treatments.

As a result software becomes more complex. In order to establish a central access point for all project data a single website has to be created. Therefor all project information including all user informations must be stored on one logical volume. Some automatically information or notification system for members and partic-pants must be installed and some sort of granulated user access and a history tracking system established. Then projects can be better coordinated, documented and even faster executed.

So at least two different kinds of collaboration can be noticed in the building process. Collaboration in the design process where the architects are maintaining there graphical and modeling informations themselves for a good reason, and the construction process where administration becomes the most vital task. The software introduced here is focussed on the second stage.

Situation

Collaboration tools for project-management are already provided by some software companies. If the data is external maintained, their internet-based collaboration software offer project management platforms for comprehensive networking. By using those services architects are going to establish dependencies they not necessarily need.

These service-providers are hosting the servers themselves because the software has to be installed on special machines in a remote location. Project data is stored on redundant storage systems. Huge monster-like machines, guarded and under video surveillance, are working uninterruptible in bunker-like facilities. The main purpose is to control access and protect the data. For security, access is limited to authorized users. This comes in handy, when user manuals are more about password protection than using the software.

Offering their services enables the companies not only to maintain but to obtain informations architects normally never would give away voluntary. While the possibility of physical damage is technically out of sight, the remaining risks are still evident. What happens to the project data, if the provider goes bankrupt, cuts off all lines and leaves all informations on hard disks and tapes to his creditors? Isn't there a system administrator around selling business data to somebody else? What's about privacy, what's about unintended knowledge transfer?

And there is a greater risk. Gathering business informations from different offices in one central places at least offers the opportunity to compare productivity and prices. In every market this is a promising way to become a main player. Even if not planned this is an invitation a lot of people can't resist.

Suggestion

The only way retaining control over some data is maintaining all physical representations of the project informations. In other words: project data must be hosted by their owner themselves what means that an independent architectural office has to run its own web-server and its own database.

To do this architects must not become internet-provider. Web-server software is already in place. Unix system such as Linux or Mac OSX have their 'apache' build in, for most of the other operating systems downloading and installing is not a big deal. Then as part of the concept every computer connected to the internet can act as a server by its IP-address.

With the increase in telecommunication also the infrastructure usually demanded by a web-server becomes a real option. The speed of a so-called 'high-speed internet access' is usually fast enough, costs are at least competitive, not to mention if such a connection is already established but not used in this way.

Requirements

The requirements for this project are defined as follows. Besides the hardware, a computer acting as server and a reliable internet connection, the demands for the software are definitely authoritative. By means there has to be a web-server integrated in the software, a ftp-server may be optional, but nothing more. The -automatic-mail service must work with a normal email-account. The software must be executable and manageable on more than one operating system, a web-based interface as only existing universal graphic interface is mandatory and copies of the software must be legal so the license has to provide this.

More complicated server-side software is unnecessary, and because an architectural office has not to be transformed into an internet service provider business, has to be avoided.

Another important issue is the quality of the web-page that potentially would be generated generated. Only pages are acceptable, that every browser can display without any complication. The use of client-side scripting, embedded executables and multimedia tools whatsoever has to be abdicated. The only exception is the use of cookies to store logging-information on the client-side.

This seems not to be the matter of course, when the internet is still a playground where all bells and whistles are not satisfying. But when an architectural office will force its partner to use the internet it has to be as open as possible, or some support would be inescapable.

Development

With the requirements in mind the developing process has become somehow complicated. Initially the decision was made to use WebObjects in order to use a java-based object-oriented design with advanced database tools. It was a good choice to establish a theoretical concept, but it didn't really work because this software demands unpredictable license fees and in most cases special server software at additional costs (OS X-Server).

Making another choice was all but easy. Most web-application software packages are aiming at large companies, where they are used to develop one unique application for a special purpose. Therefor license fees are usually high. Also those software is also often depending on some special software or proprietary hardware, or they make use of scripting or other more or less advanced technologies.

Open-Source

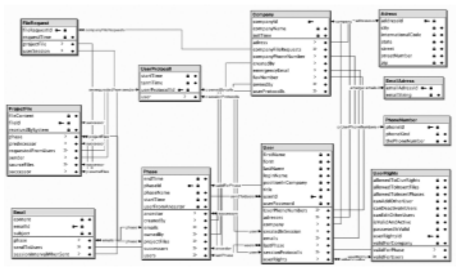
For this reason, prediction of the costs and an insight in the used technology, the general decision was made to change the software base to an open-source product, and then very quickly Zope (www.zope.org) was chosen. This python-based software is strictly Open-Source and therefor free, provides its own database and web-server (the old veteran medusa) and supports several platforms (Linux, Unix, Windows, and Mac OS X). If desired generated web-pages can be reduced to html, although scripting is possible.

The disadvantages are common in Open-Source, some lack of documentation, complex versioning and community driven support. But in the case of Zope this is somehow corrected by the fact, that there is one supporting company accepting the responsibility for releasing and basic documentation.

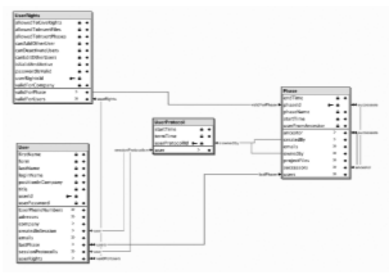
Theoretical concept

The theoretical concept determines the structure of the database necessary to develop the software. The basic elements are phases, users and messages.

The concept was initially developed using the database developing tool provided by WebObjects and is still in place even though the software base has changed. Therefor the following presentations are referring to this package.



Phase



Phases are the key elements of the concept

A phase has, besides its name and ID, a starting and a termination date allowing the decisive validation of all kinds of inputs. Input is only possible if and only if the phase is open or valid. Otherwise access is restricted only to read.

Valid input can only conducted by authorized users. Therefor each phase has several information about its users, the creator as initial owner of the phase, the owner as user responsible for the content, a maintainer to administer the content and a group of non-privileged users. If necessary, it is also possible to establish a single-user phase.

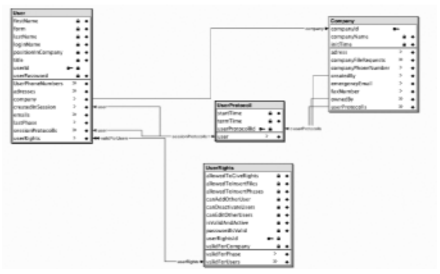
But the most important informations are how each phase is embedded in the overall structure of the project. To determine the hierarchical position a phase provides knowledge about its superior or parent-phase and its child-phases. Additionally a logically connection on the same level can be established. A common case is that a phase only is valid if another phase is closed.

Last but not least a phase can provide some valuable data, summarized as project files and messages, usually emails.

User

In a computerized environment a user is an acting human being.

Therefor a system dealing with users has to provide some personal information, first and last name, title, position, password and some contacting informations like address, phone-numbers



email-addresses. This is an ordinary computational task with minor differences in execution.

Unlike this common user-domain or user-machine relation typical users in a building situation are not on their own as individuals. They are acting as member of some company or organization and their relationships are based on their positions in those companies. Additionally the relations among companies are based on contracts. Therefor companies are vital elements here, although they are usually not appearing in the interface.

Another difference results from the concept based on phases. Commonly users are associated with their user-rights on a one-to-one relationship and some fine tuning related to the data, but not too much. This model cannot be used here. Instead every user is associated with user-rights specially defined for every phase the user has access to.

This has significant effects on the user-rights. If every phase provides a single record for every user the relationship duplication becomes inescapable. Therefor user-rights are established ass a many-to-many relationship with all its benefits and complications.

Inheritance

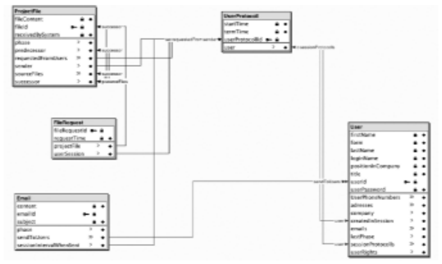
Both elements, phase and user, are inheritable meaning that some of their characteristics can be transferred to others instances. For example, a subordinate phase can never exceed the time frame of its parent phase, the chief of a company can create new user as member of his company and transfer some or all his rights but no more.

Messages

The main issue in project-management is handling time dependent data.

For this purpose the developing is concentrated on three tools to transfer data or messages: file sharing, message-board and email notifications.

This part of the program is straightforward. Contained in the phase most attention is paid to protocoling. Every activity is associated with a user-protocol, a record providing some logging



informations and thus remembering the exact state of the user when he is active.

Layout

The use of web-pages always implies mind-boggling layout discussion. To prevent this every phase can obtain its own style sheet (*.css - file). Style informations not found in the phase are inherited from the parent-phase. In most cases this simple approach, based on the phase-design, offers sufficient space for some creativity, far more than every centralized software-solution can allow.

Working Solution

After the switch to Zope developing was focussed on a working solution. Using a desktop computer, a cable-connection and an online domain registration a working prototype was established (www.arkitekters.com). Behind a firewall the prototype was running stable for more than a month.

The software now provides some basic functionality, up- and download, a 'black-board' and some automatic emailing. There is still a lot of work to realize the whole concept.

Summary

Even if the project isn't finished yet, it is a success because it demonstrates that an architectural office is capable of maintaining and distributing his own data electronically without using external service provider. Internet-based collaboration not necessarily needs a remote server, and, from the viewpoint of an architect, is not only restricted to the design process.

References

Diederichs ,Claus Jürgen: Führungswissen für Bau- und Immobilienfachleute: Bauwirtschaft, Unternehmensführung, Immobilienmanagement, privates Baurecht. Berlin u.a. (Springer) 1999.

Latteier, Amos and Pelletier, Michel: The Zope Book. Boston a.o. (New Riders) 2001.