Design of Distributed Multimedia Systems

Jan Smets

Designing a distributed multimedia system for public places (e.g., hotels) in a non-traditional way is not a guarantee for success, but a traditional approach is almost a guarantee for failure. This article is about experiences with designing and managing multimedia systems.

In my short career in the multimedia world, I co-worked on an interactive teletext/virtual city project, and did the software design for a closed-network multimedia system based on central data processing and set-top boxes and/or network computers (NCs). The first system was based on self-made and self-assembled systems, with a lot of self-developed code. The second was based on as much standard components as possible, with very little software development. Both approaches had their (dis)advantages.

Traditional development

A traditional way of developing an IT system is the so-called waterfall model. Many traditional systems are implemented to replace time consuming tasks. The advantage of most waterfall-based project methods is that they are proven methods, and lots of people are familiar with them. Most traditional systems are implemented when the complete design and coding is finished.

Requirements collection

Most distributed multimedia systems are not a substitute for something already existing, but bring something new. The lack of a current situation makes the migration to the new situation difficult. Most customers ask a question like: “I want people to order pizzas from their hotel room, choose what movie they want to see, play games, send and receive messages, another wild idea, and . . . . Oh, and it must be done on our TVs, the costs must be low, and I want it in three months”. So you start writing down the requirements and, “Oh yes, it must be fancy”, start to realise that, “Oh yes, I want Internet”, the customer has no clue, “Is it interactive?”, what the consequences are when you implement all these ideas.

A developer and a customer have a different idea of the system, also the view of the end-user is a completely different one.

The developer’s view
The system must be robust, maintainable and technically well-documented, easy-to-use, and fully configurable.

The customer’s view
The system must be implemented tomorrow. The payback time on investment must be short and the initial investments must be low. The system must give the customer a competitive position: she/he wants to be the first that exploits it, but she/he wants references. The customer does not want to turn out to be a beta test site.

The end-user’s view
The system must be easy to use. It may cost little. Some parts should not be available when his children use it, such as ordering goods/foods or watching adult movies. The children want to play games with at least the quality they are used to (like Nintendo and Sega).

The above views contain a lot of contradictions. A designer of such a system has the task to understand all priorities, and to come with an alternative that is acceptable by all parties. Keep in mind that the customer pays the bill, and that the system is useless without end-users.

Project phases

Another non-traditional aspect of projects implementing distributed multimedia systems, is the lifetime, the develop time, and the implementation time. Because of the new technologies that are often used, the lifetime of a distributed multimedia system is about three years. If you are not the first one with such a concept, the lifetime of the product is a fixed date. In this respect, lifetime is the time that the system is used and accepted by the end-users. It is very important that the first version is released very soon, even if it is just a basic and not fully functional version. This implies that the system has to be divided into modules, and the core technology has to be implemented first (which is not new). Starting from a basic system with basic functionality, every week or month a new module can be implemented and added. This incremental implementation is very important, because feedback from the customer and end-user may partly direct the result. During the implementation period, little parts of the software and the user interface can still be changed.

Component selection
There are some traditional criteria for component selection like *hardware or software* and *make or buy*.

- *Hardware or software.* Important for a multimedia system is the way it can be configured, and of course the performance. Core technology must have a good performance, so this is mostly done in hardware, e.g., MPEG encoding and decoding. Configuration and flexibility requires software, e.g., administration, logging, content such as films and games.
- *Make or buy.* It is recommended to use world standards in a system if possible. Some standards are only available on paper. If this is the case it is better that the standard must not be used or implemented yourself. Because of the lifetime of the system, it is recommended to make use of as much standard components as possible (buy). The logic between all systems is a matter of software (make, assemble).

Always try to find out if components in the system are available during the whole system lifetime. When computers or workstations are used, they will probably not be always available. To circumvent any wrong assumptions, the supplier of these products can clear things; she/he can tell about future extensions, changes in interfaces, and changes in performance. Sometimes, a product has not enough performance when you start developing, but there will be one when you start implementing. A migration path should be created, and the system should be designed in a way that components can be switched during its lifetime. Another advantage of good contact with the suppliers is the availability of information. Nowadays, most suppliers of multimedia components want to cooperate when implementing their technology.

New techniques

Distributed multimedia systems make use of a lot of new technologies and traditional technologies in a new way.

- Databases, SQL for logging, trend analysis, and maintenance of the system. Also contents can be easily updated when using a database.
- Object oriented code for the software.
- Networks for distribution of information and connecting servers, workstations and video/teletext processors.
- Set-top boxes or Network Computers (NCs) for communication with the end-user.
- Teletext for static information distribution, or simple interactive applications.
- Telephone for communication between the end-user and the central server.

- Video for content and highly interactive applications.
- MPEG-1 or MPEG-2 coding and decoding for storage and distribution of video and audio content.
- Cable television (CATV).
- Fast ethernet for distribution of MPEG data.

Maintenance

A multimedia system has a lot of aftercare and maintenance, both technical and content maintenance.

- *Content maintenance* refers to the updating of content, e.g., news, horoscope, and movies.
- *Technical maintenance* refers to access control for new and existing users, trend analysis on usage, enabling and disabling parts of the system.

Risks

A risk analysis should both take care of the traditional risks and the risks typically associated with a multimedia system.

The 'developer risk'
A multimedia system makes use of different technologies: databases, embedded software, and user interfaces. Specialists are required for all of these. All specificalists, however, talk completely different languages. They all have a different view on the system. Traditionally, a database is the heart of the application, here it is just a way of data storage. Functionality must be in software, and the performance needs to be guaranteed (less stored procedures, redundancy is sometimes allowed). The interface designer is not always aware of the consequences of a specific implementation, but a nice interface can also hide performance problems (entertain the user when waiting). The system architect must act like a referee between these people.

The 'customer risk'
The customer does not think in computer terms, but expects a set of functionalities. Customer needs must be known. When a customer sees the first presentation of the user interface, expectations and requirements will increase.

The 'end-user risk'
It is important to find out what the end-user wants. It is recommended to make a test interface in a very early stadium and try to find out what the failures are.

The 'money risk'
The most important issue of all systems is money. What are the costs, and how is the payback time related to the lifetime.

Conclusion

Designing a distributed multimedia system is fun, but it requires much more than a technical background. Systems like this are developed for ordinary people, so they make the requirements. Listen carefully to your customer, and know the customer of your customer! □

Jan Smets finished MTS-TCK in 1990 and started a study Information Technology at the Hogeschool Eindhoven in his free time. He started his career at Simac Systems in Veldhoven in 1990 programming and designing products for mobile datacommunication. Currently, he works as a designer for Origin/Eindhoven IPS since 1995. Jan Smets is especially interested in system architecture and multifunctional systems.