

X Window System

Reviewed by Group C

Matthijs Neppelenbroek

0331716

M.G.Neppelenbroek@students.uu.nl

Matthias Lossek

F100132

Matthias.Lossek@student.uni-
augsburg.de

Rik Janssen

3549402

R.B.M.Janssen1@students.uu.nl

Tim de Boer

0237884

T.deBoer@students.uu.nl

Software Architecture
Faculty of Science
University of Utrecht

January 30th, 2011

1. INTRODUCTION

This paper presents a review of the X-Window system described in [3]. We will review the paper by using a change impact analysis and a risk analysis. Change impact analysis attempts to find out the impact of a change on some artefact X to another artefact Y, especially important for (software) maintenance. In our review the focus is on requirements, that is, what parts of an architecture description are influenced when requirements change.

We introduce two changes to the systems architecture, which have different impact. The first is a small requirements change which has a low impact, this change should be implemented without much additional re-architecting. The second is an a requirements change which has a big impact and can only be implemented by re-architecting the system.

We finalize our review by presenting a risk analysis where we will describe the risk involving the two suggested requirement changes.

2. CHANGE IMPACT ANALYSIS

In order to describe the impact of requirements changes, we picked two different new features for this paper. The first one is integrated audio support, which is relatively easy to add to the original architecture of the X-Window system. The second change is an optional hardware accessing, which should bridge the gap of performance lacks of the X-Window system. This can only be integrated in the current architecture with a redesign of the original system.

2.1 *Change 1: Integrated Audio Support*

Inspired by the question of Ruben van Vliet (group E), we describe how audio support can be integrated into the original architecture of the X-Window system.

As illustrated in Figure 1, audio support can simply be added to the architecture in the same way as other features work in the X-Window system. New added parts are depicted in red.

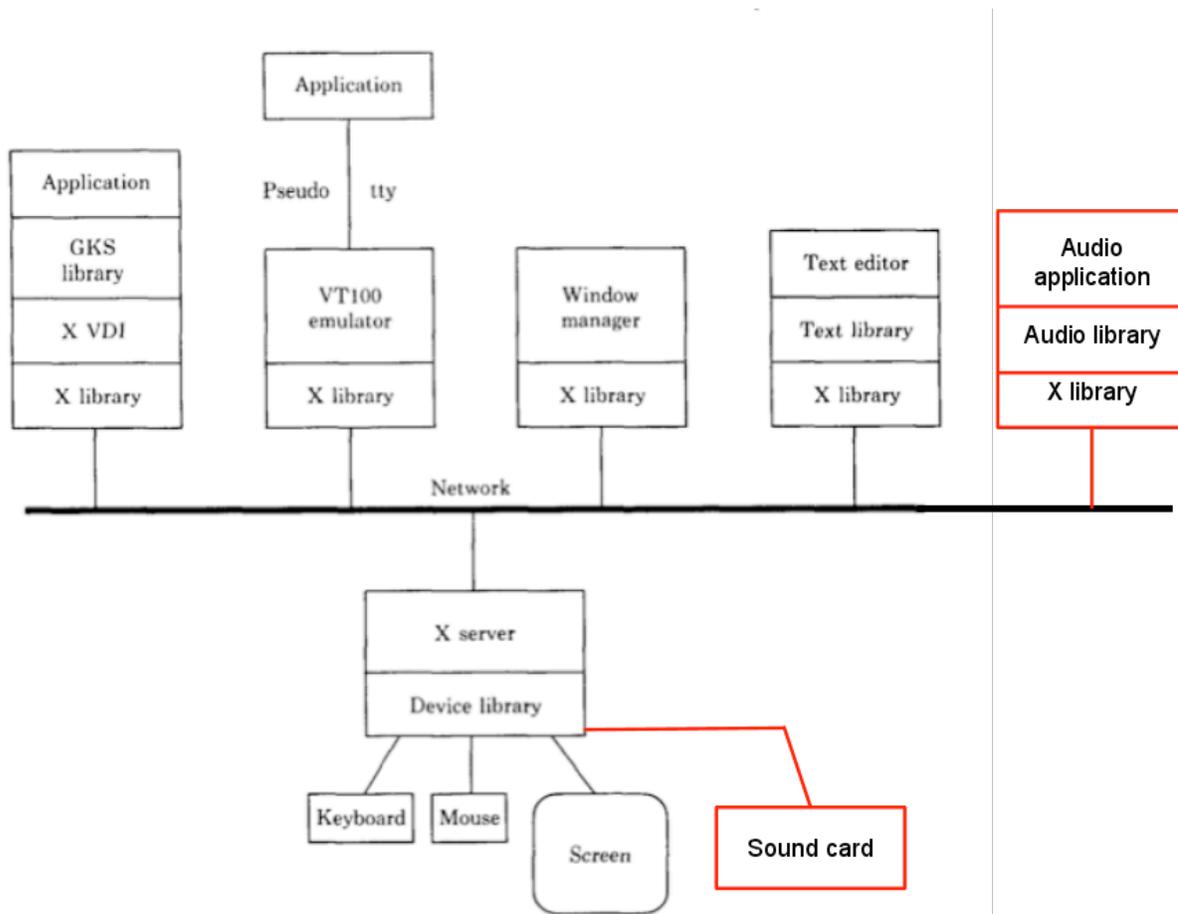


Figure 1: Integrating audio support (red lines) in the X-Window architecture.

For integration of the audio feature, the X library has to be extended with an audio library, which can be accessed by the applications. This represents the client side.

We don't want to give a full solution for how to build that audio library, but depending on the purpose, this can be provided by a simple streaming API for a compressed audio format like mp3, or maybe even with a raw data format for audio files. The second solution would result in a much higher traffic rate, but of course with a higher sound quality.

At the server side of the X-Window system, the audio support has to be added to the device library of the X server in the same way as the support for a mouse, a keyboard or a screen. Of course, this hardly depends on the available hardware at the server's machine, but this is exactly the same kind of problem that X already solves with the other devices.

These changes are enough for adding audio support to the X-Window system. It is obvious that these small differences in the architecture have more or less no influence on the mentioned quality aspects and

trade-offs which were explained in the original X-Window paper. Of course, there is a small impact on the quality aspects usability (higher attractiveness with audio) and efficiency (worse time behaviour due to higher traffic), but as X-Window was designed with the main goal of portability, that can be ignored. Furthermore the audio support is an additional feature and without using it, it is still the same system as before.

2.2 *Change 2: Optional Hardware Accessing*

As discussed in the X-Window paper, the biggest negative aspect of X is its worse performance compared to a direct access of hardware devices. Due to the abstraction of hardware components for providing a high portability like network transparency or device independence, a lot of overhead is required.

On the client side with the X library and on the server side with the device library the X system has to do a lot of steps for packing and unpacking the information. If the client and the server are on the same machine, all the overhead is useless and the whole process is a waste of performance. As explained in the X-Window paper, this is nowadays one of the main use cases of X and there are systems like Wayland available which try to solve that overhead problem.

But what architectural changes are essential to provide both functionalities in X? A highly portable protocol for the network and inter-device communication and a fast low-level communication for a local one-machine-environment.

Figure 2 shows the standard use case diagram of X as described in the X-Window paper with the abstracted communication between the X clients and the actual Kernel.

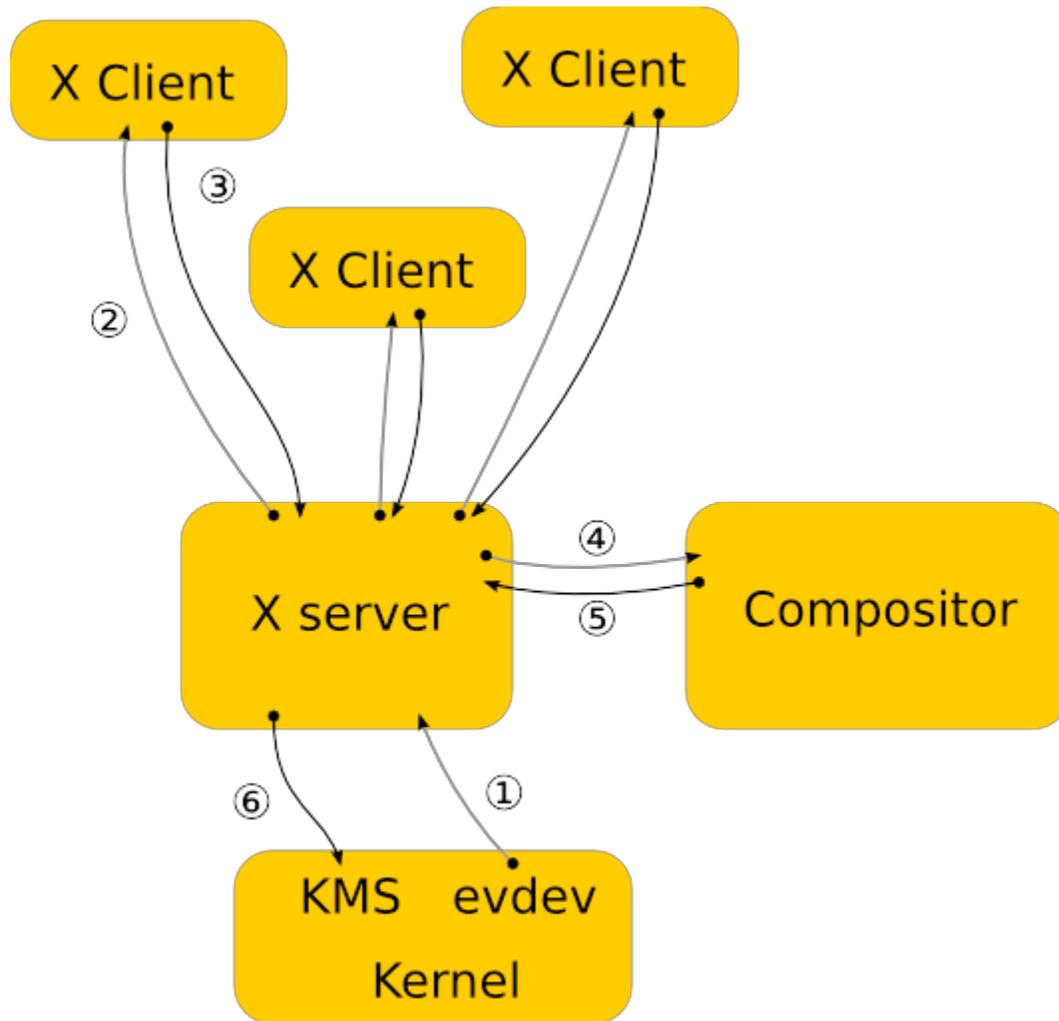


Figure 2: The default X-Window use case from [3].

We now want to integrate the new requirement, that an X client can also choose the direct way to the local kernel instead of doing all the unnecessary work for the hardware abstraction.

For fulfilling the new requirement of an optional direct communication without the abstraction overhead, some deeper changes in the X-Window architecture are required.

First of all, a component is needed which offers the same functionality as the X server for the remote clients, but for the local clients. In this case this component is called the local controller.

Figure 3 shows an overview of the briefly illustrated new architecture.

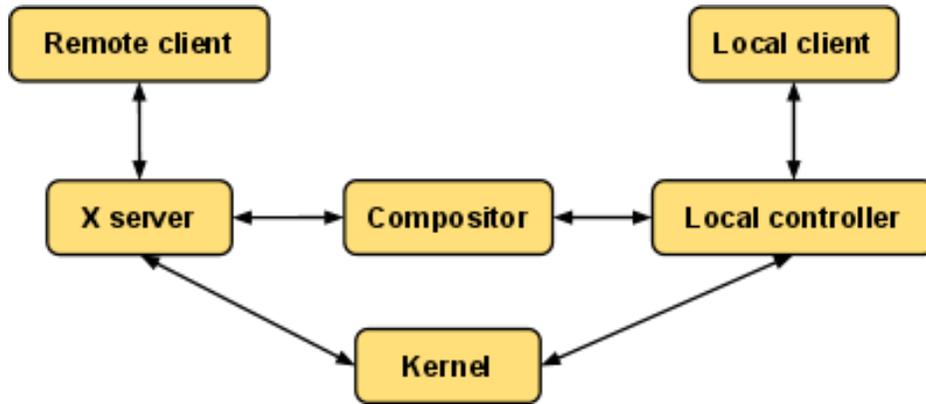


Figure 3: The new architecture with a local component.

Of course in this case the functionality with the remote client and the X server stays the same as before. The difference is only the new local side of the architecture. A local client has to know, that it is a local client, so all his communication skips the abstraction of the remote X protocol and directly talks to the local controller on a more low-level base. The local controller then does more or less the same as the X server does, instead of all the transformation procedure. It forwards the client's message to the compositor and interacts with the kernel. This steps could integrate the local support to X, but only with a lot of effort. The whole local part has to be developed and integrated into X.

While looking at Wayland as described in the X-Window paper, Wayland is the same as only the local part of the new X-Window architecture. Only the compositor and the local controller are combined in the Wayland compositor. So this new X architecture is simply a combination of X and Wayland.

As this combination of the advantages of fast local clients and flexible remote clients makes sense, there are projects available of combining the X-Window system with Wayland. For example Wayland itself can interact with a X server for supporting X's flexibility, but it is not yet a real replacement of X, as there is a lack of driver support [4]. Other approaches for using both systems are integrated into the GTK+ and the Qt libraries [2, 1].

X was designed for its flexibility and therefore adding this aspect is not the actual idea behind X-Window. Even if it is interesting to see, how to add this functionality, modern systems like Wayland already try to solve the lack of performance while providing flexibility by supporting X server interaction out of the box.

3. RISK ANALYSIS

In the change impact analysis we have seen two different changes in the architecture of the X-window system. The changes described above are changes of the requirements of the system. The risk for an

architecture is that the requirements change over time and that the artifact build by the architecture can not met his primary goal anymore. To say something about the quality of the architecture we want to check how good it can handle changes in requirements. First we are going to look at the likelihood that changes in the requirements will occur. For this we will use the proposed changes in the chapter above to examine how well the architecture can be adapted to those changes to meet these new requirements.

The first change in requirements for the X-window described in the change impact analysis is that of supporting sound cards by the X-window system. This change of requirement can be generalized to a broader need of users of the X-window system, namely the support of more in- and output devices. When we look at the development of in- and output devices the last years we can say that these devices are rapidly changing. When looking at Wii-motes, X-box Kinects and 3D screens already widely available on the market, we can state that the need of supporting more in- and output devices will only grow in the coming years. The likelihood that the requirement on the support of these in- and output devices will change is therefore also big.

As we have seen in the example of the sound card, the impact of a change for the requirement on in- and output devices is not very big on the architecture of the system. When the X-window system need to support new devices it can easily be extended by writing two libraries on the client and server side of the system. The risk of this change of requirement is therefore low. The quality aspect of usability would be exposed to this risk but will not suffer from it, because of the good extendability of the X Window architecture. The likelihood of the change is high, as already mentioned, but the impact it has on the architecture is relatively low. Therefore the risk of this threat, the requirement of supporting more in- and output devices, is relatively low.

The architecture document [3] gives a good insight why the architecture can deal with this risk so well. This has all to do with the principles behind the architecture of the X-window system and these principles are stated clearly in the architecture document.

The second change in the X-window system has to do with performance of the system, but eventually this also leads to better usability of the system. This second change in requirements has a much higher impact (this was also a goal for this assignment) and this already increases the risk of this change in requirement. Boosting up the performance as such that the usability of the systems will increase is a bridge to far for the X-window architecture. The changes that have to be made in the architecture are to big to fulfill the need of this requirement. Other systems (e.g. Wyland) that tried to meet this requirement, therefore broke with the original principles behind the X-window architecture. They threw out the X server and this will lead to such a change in the architecture that they can not make it network transparent anymore. The likelihood of this change is very low, this can be stated out of the fact that the performance of the system was good enough for the users of the system to use it. Unfortunately after thirty years, the biggest user of the X-window system (Ubuntu) declares that the architecture can not comply with the requirements they have, to "deliver a user experience with super smooth graphics and effects" as is stated in the architecture document [3]. So, as the risk of this change is relative low because the likelihood is low (the need of it did not appear within thirty years), the impact of it is so large that the architecture needs to be redesigned rigorously.

Also making the second analysis of this risk, could be done well by the information given in the architecture document about the X-window system [3]. The comparison with the other systems gave a good insight into the biggest problem the X-window system has: performance issues caused by not accessing the hardware directly. The other systems showed that the solution for this problem could not be made in the current architecture and that big changes within the architecture are needed to satisfy this requirement.

4. REFERENCES

1. Gitorious.org (2010). *The architecture of Lighthouse*. <http://qt.gitorious.org/qt/pages/LighthouseArchitecture>.
2. Larabel, M. (2010). *GTK+3 Now Uses X Input 2 By Default, New Back-End Caps*. Phoronix, December 22th, 2010. http://www.phoronix.com/scan.php?page=news_item&px=ODk0Ng.
3. Peersman, H., vanderVelden, J., Mitilinos, N., Hijlgaard, R. (2011). *X Window system*. <http://www.cs.uu.nl/wiki/pub/Swa/CourseLiterature/arch-D.pdf>
4. Wayland.freedesktop.org (2011). *Wayland architecture*. <http://wayland.freedesktop.org/architecture.html>.