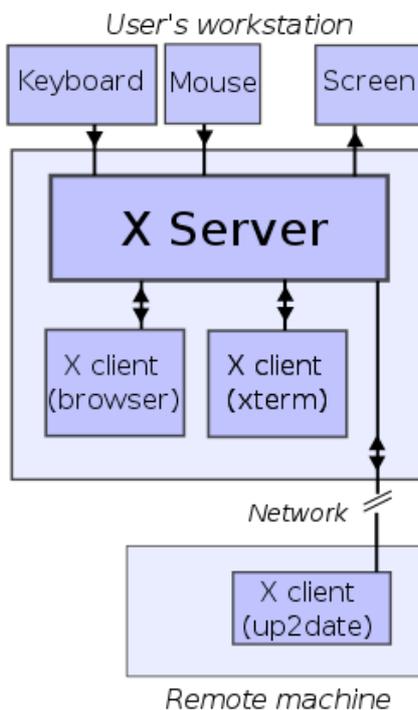
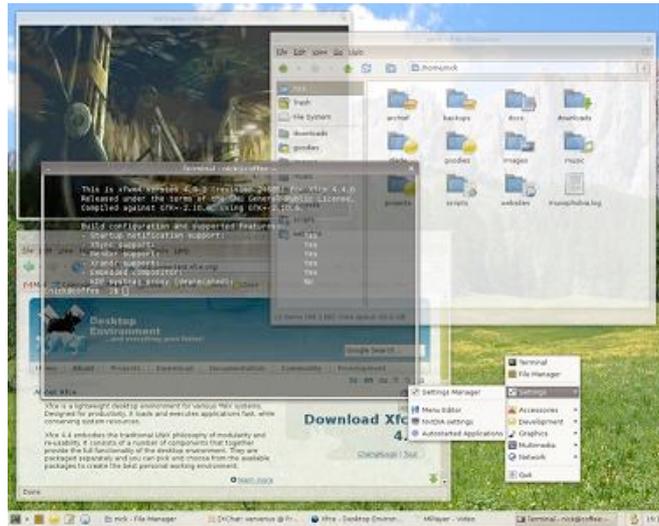


# X Window System



**Course:**  
Software Architecture

**Authors:**  
Hans Peersman  
Jeroen van der Velden  
Nikos Mitilinos  
Renato Hijlgaard

**Group:**  
D

## Contents

|   |    |
|---|----|
| X Window System .....   | 4  |
| History .....   | 4  |
| Design Principles .....   | 4  |
| Development .....   | 5  |
| Ownership .....   | 5  |
| Components of X Window System.....                                      | 5  |
| Client (application).....   | 5  |
| X Server .....  | 6  |
| X Window Core Protocol .....  | 1  |
| XLib (X Library) .....  | 7  |
| Window manager .....  | 8  |
| Modern usage.....   | 8  |
| Additional extensions.....  | 8  |
| Some alternatives .....   | 8  |
| Quality Aspects .....   | 9  |
| ISO9126-1 Model .....   | 9  |
| Functionality .....   | 9  |
| Reliability .....   | 10 |
| Usability.....  | 10 |
| Efficiency .....  | 10 |
| Maintainability.....  | 10 |
| Portability .....   | 11 |
| Quality Aspect 1&2: Functionality - Security & Interoperability .....   | 11 |
| Security.....   | 11 |
| Host & Token authentication .....                                       | 11 |
| Secure Shell .....  | 12 |
| Security Extension Specification .....                                  | 12 |
| Interoperability.....   | 12 |
| Quality Aspect 3: Reliability – Maturity .....                          | 13 |
| Quality Aspect 4&5: Usability – Attractiveness & Understandability..... | 14 |
| Quality Aspect 6: Efficiency – Resource Behavior .....                  | 15 |
| Quality Aspect 7: Maintainability – Analyzability .....                 | 16 |

|  |    |
|--|----|
| Quality Aspect 8: Portability – Adaptability .....                       | 16 |
| Trade-off between aspects .....  | 17 |
| Security vs Operability and Interoperability .....                       | 17 |
| Attractivity vs Understandability .....                                  | 18 |
| Usability vs Portability .....   | 18 |
| Realization and technical aspects of the implementation of aspects ..... | 19 |
| Security.....  | 19 |
| Host authentication .....  | 20 |
| Token Authentication.....  | 20 |
| SSH.....   | 20 |
| Toolkits (Attractivity vs Understandability) .....                       | 21 |
| Usability vs Portability .....   | 22 |
| Comparison with 2 other systems.....                                     | 24 |
| X Window vs Wayland.....   | 24 |
| Wayland.....   | 24 |
| Comparison .....   | 24 |
| Conclusion.....  | 27 |
| X Window vs MicroXwin .....  | 28 |
| Conclusion.....  | 28 |
| Questions from other Group members .....                                 | 29 |
| Questions Group A.....   | 29 |
| Questions Group B.....   | 31 |
| Questions Group C .....  | 33 |
| Questions Group E .....  | 34 |
| Questions Group F .....  | 35 |
| Questions Group G.....   | 37 |
| References .....   | 39 |

## X Window System

X Window System is a piece of software that provides a graphical user interface (GUI) for applications and systems in a network. It allows users to display a program on their own machine, while it was running on a server. It also accepts user input. The machine that runs the program is called the client and the machine that displays the output is called the server. This is counterintuitive with the basic thought behind the standard client-server principle, where the server delivers the data and the client is displaying the results. A nice advantage of X Window System is the fact that the client and the server don't have to be running on the same machine.

### History

Jim Gettys and Bob Scheifler founded X Window System in 1984. As a part of project Athena, at the Massachusetts Institute of Technology in Boston, they created the first version of X. Athena needed a platform independent graphical system in order to link all its multiple vendor systems. There was one solution but it didn't had a license and any other alternatives don't exist. In May 1984, Scheifler replaced the synchronous protocol of W, which can be seen as a predecessor of X and was a windowing system that ran under the V operating system, with a asynchronous protocol and X was born.

### Design Principles

The design principle of X Window Systems according to Scheifler and Gettys (1986):

1. The system should be implementable on a variety of displays; the system should work on any bitmap and input device.
2. Applications must be device independent; it must not be necessary to rewrite, recompile, or even relink an application for each new hardware that is added.
3. The system must be network transparent; an application that runs on one machine must be able to use a machine that is connected to another machine on the network. These machines may have different architectures, but that doesn't matter.
4. The system must support multiple applications displaying concurrently; multiple actions may occur at the same time in different windows.
5. The system should be capable of supporting many different application and management interfaces; there isn't such thing as a single user interface, so the application should provide the basic elements of this interface and let the graphical machine decide on what the output should look like.
6. The system must support overlapping windows, including output to partially obscured windows.
7. The system should support a hierarchy of resizable windows, and an application should be able to use many windows at once; by providing a true window hierarchy, application windows can be implemented as true windows within the system, freeing the application from duplicating machinery such as clipping and input control.
8. The system should provide high-performance, high-quality support for text, 2-D synthetic graphics, and imaging; the application describes the image and the system understands it immediately. They mention 3-D graphics in their article and say explicitly that it is not a requirement at that moment in time due to lack of expertise and time.

9. The system should be extensible; communities should be able to extend the system by their own, but they should be merged with the core seamlessly.

These 9 requirements date back to 1986, but they are still the basis of X Window System. The real 3D imagery arrived to X in version 11 (the current protocol) and isn't limited to only display 3D images on 2D devices, but also on 3D devices, such as 3D-tvs.

## *Development*

As stated above, the first version of X Window system was founded in May 1984. After this release, the team made progress very rapidly and the 6<sup>th</sup> version was released in January 1985. This version also got licensed to some outside groups for the first time. The next big improvement was in version 9 in 1985; color was now supported. At the end of 1985, a team at Brown University ported version 9 to the IBM RT/PC platform. Due to an incompatibility with the RT platform, changes in the protocol were necessary, which lead to version 10. The 3<sup>rd</sup> release of X10 (X10R3) was published freely under the MIT License. All future versions should use this license as well. The final release of X10, release 4, was developed in 1986. X10 was offering powerful functionality, but despite all that, a more hardware-neutral redesign was necessary before X more common. This improvement was made in the current version of X, X11, in 1987. This protocol is still in use and in December 2010 it has reached release 7.6.

## *Ownership*

- MIT; MIT decided in 1987, after the release of X11, to give up the ownership of X and to sell it to some vendors.
- MIT X Consortium; however, in 1988, the MIT X Consortium was founded as a non-profit group with Scheifler as the director. This consortium should develop X in a neutral atmosphere.
- The Open Group; in 1997, the Open Group took over the ownership of the consortium.
- X.Org; in 1999, the Open Group formed X.org
- X.Org Foundation;

## *Components of X Window System.*

The following section describes all the individual components which together form the X Window System.

### **Client (application)**

The X Client is the application that is running. Its main task is to provide data to the clients like:

- Draw a line from this point to this point
- Draw this string with this font at this position

How these commands are handled is not the concern of the client. The Server handles these commands.

## X Server

The X Server is the local machine that is connected to the X Client to display the program it is running. The Server contains the libraries that are specific for the types of hardware on that system. The global data that is generated by the client is passed to the server, which makes it machine specific so that it can handle it on its own machine.

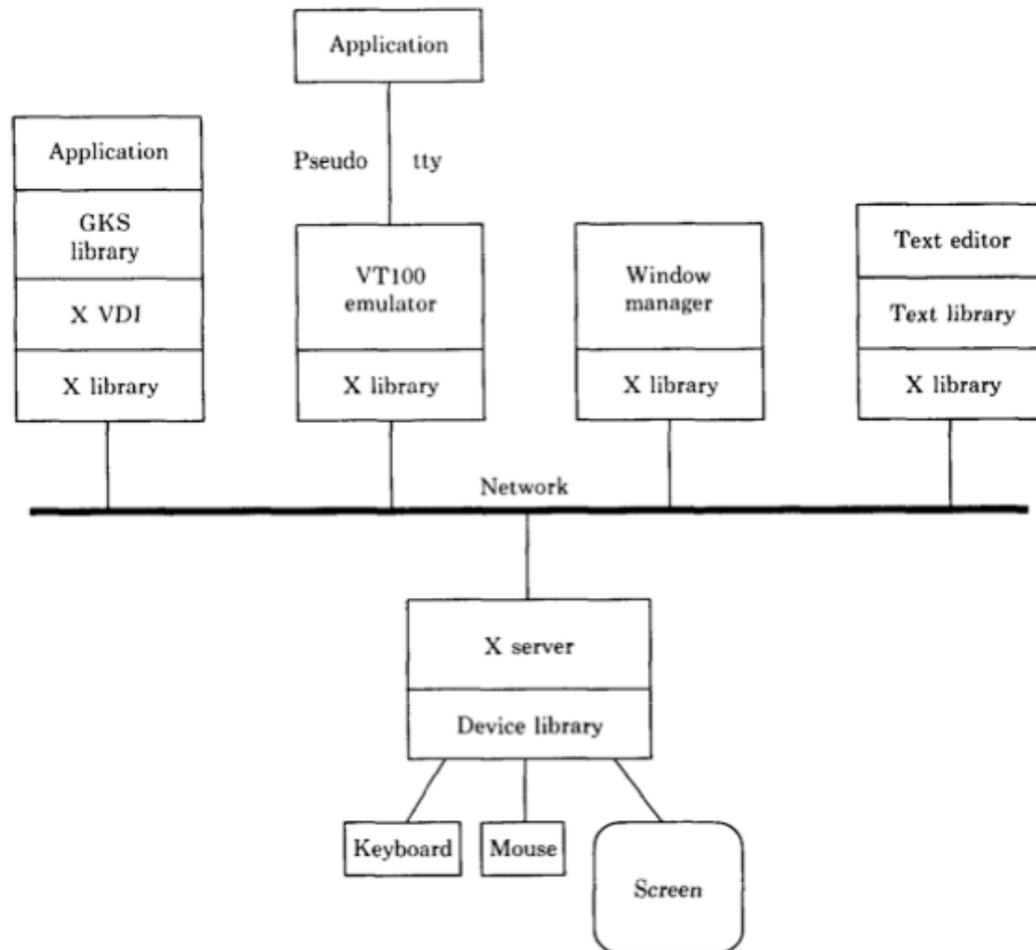


Figure 1 - X Window System Structure Scheifler and Gettys (1986)

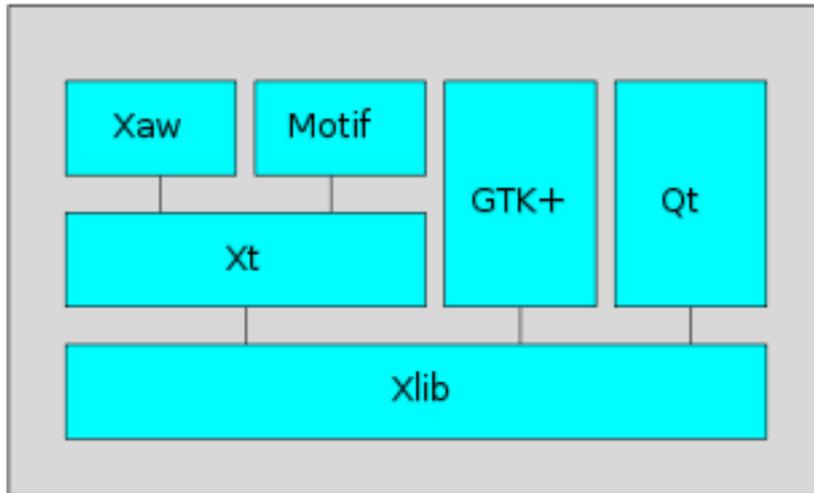
## X Window Core Protocol

In the core protocol, only four types of messages are sent over the network:

- Request: The client is requesting information from the server or it is requesting to perform some action, like creating a new window.
- Replies: The response from the server on a request by the client.
- Events: Information about any user activity (hardware movement, window moved/resized/exposed etc.) is sent from the server to the client.
- Errors: If a request is invalid, the server sends an error to the client.

A request can generate all other three types of messages and there is no order in how these messages should be transferred.

## XLib (X Library)



**Figure 2 - The X Library schema**

X Lib is a software package that is available on the client-side of the system. X Lib contains all the functions that are needed for the interaction of a client with an X Server. Because of the existence of these functions, programmers don't need to know the X protocol by hand, but only use those functions to interact with devices. Furthermore, the library itself isn't often called directly but mostly by the use of widget toolkits, like X Intrinsics (Xt). These toolkits are used to communicate with the X lib functions.

Within X Lib, three types of functions occur:

1. Connection functions. These functions are used to get connected with an X server. The function "XOpenDisplay" is used to set up this connection.
2. Request functions. All requests from the client to the server fall under this type of function. For example: "XCreateSimpleWindow" will request a new window and "XMapWindow" will display the window on the screen.
3. Local functions. Functions that are requests for local data or the event queue are placed in this category. For example: "XNextEvent" and "XSaveContext".

All the functions within X Lib are very basic. That means, if you need buttons or anything else than just window, you will end up using additional libraries. In the end this will mean that you use libraries that communicate with Xt or with X Lib directly. Xt provide an API to use the X Lib. Libraries like Xaw and Motif communicate with Xt in order to use the basic functions of the X Lib. Libraries like GTK+ and Qt will communicate with the X Lib directly.

## Window manager

The GUI exists of a top-level-window and it covers the whole screen. Within this top-level-window, other windows are allowed, sub-windows. A window can only be generated as a sub-window of a parent-window. Therefore, a hierarchy of windows exists, which can be stored in a tree.

## Modern usage

Modern applications require more and more bandwidth. For example, a full screen resolution movie of 1920x1200 pixels will require 1.65Gbit/s on data per client. In these situations, it is ideal to have both the server and the client on a single machine.

X is also being used as a desktop environment in mostly all UNIX like operating systems, like GNOME or KDE. Only Mac OS X, iOS and Google Android are the few UNIX systems that do not use X for graphics. These desktop environments have the client and the server running on the same machine. X offers both the ability to either have the server and client on remote machines offering communication trough TCP/IP or the possibility to have the server and client both on the same machine, whereas the traffic between the both would go via IPC. The environments are not only responsible for the handling of the graphics, but also provide some basic applications that all have the same interface.

## Additional extensions

Scheifler and Gettys designed the X server to be simple but extensible. Due to this, much functionality nowadays resides in extensions to the protocol. At the protocol level, every extension can be assigned new request/event/error packet types. Extension features are accessed by client applications through extension libraries. Adding extensions to current X server implementations has been made easier since the improved modularity change within X11.7.\*.

## Some alternatives

Although its popularity among UNIX systems, some people are still trying to create alternatives or replacements of X, like the Y Window System and Fresco. These alternatives are hardly used and criticisms doubt the success of these systems if they don't support backwards compatibility with X.

The mostly criticized part of X is its overhead. Time is wasted, because of all those libraries that are needed to communicate with the hardware. Wayland is such an alternative that can communicate directly with the hardware and therefore will bypass that shortcoming of X. Ubuntu will eventually use Wayland instead of X as the main display manager.

## Quality Aspects

This chapter will start explaining the ISO9126 model where the quality aspects are derived from. Next an overview of the most important quality aspects related to the X Window System will be given along with an explanation on why these aspects are most relevant in the continuous development of X Window Systems.

### ISO9126-1 Model

The ISO 9126-1 model is an international standard for the evaluation of software. The ISO 9126-1 model is divided into four parts addressing the following aspects: quality model; external metrics; internal metrics and quality in use metrics. The model its main goal is to define a set of software quality characteristics.

The ISO 9126-1 software quality model identifies 6 main quality characteristics, namely:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

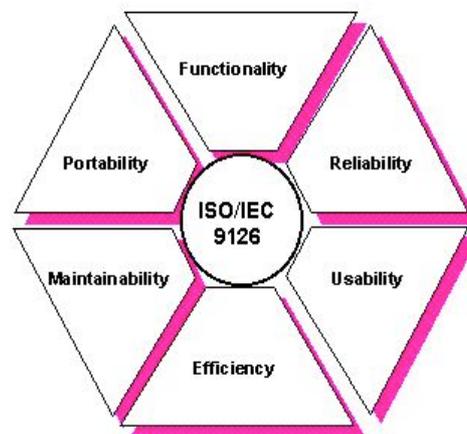


Figure 3 - ISO 9126-1 Model

These characteristics are broken down into sub characteristics, which are being used for quality measurements. The main characteristics of the ISO9126-1 quality model, are defined as next by Wallace and Reeker (2001) and Loasvio et al. (2004).

#### Functionality

The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

Sub characteristics:

*Suitability:* The presence and appropriateness of a set of functions for specified tasks.

*Accuracy:* The provision of right or agreed results or effects

*Interoperability:* Software's ability to interact with specified systems

*Security:* Ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.

*Compliance:* Adherence to application-related standards, conventions, regulations in laws and protocols.

## Reliability

The capability of the software product to maintain its level of performance under stated conditions for a stated period of time.

Sub characteristics:

*Maturity:* Attributes of software that bear on the frequency of failure by faults in the software.

*Fault tolerance:* Ability to maintain a specified level of performance in cases of software faults or unexpected inputs.

*Recoverability:* Capability and effort needed to reestablish level of performance and recover affected data after possible failure.

*Compliance:* Adherence to application-related standards, conventions, regulations in laws and protocols.

## Usability

The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.

Sub characteristics:

*Understandability:* The effort required for a user to recognize the logical concept and its applicability.

*Learnability:* The effort required for a user to learn its application, operation, input and output.

*Operability:* The ease of operation and control by users.

*Attractiveness:* The capability of the software to be attractive to the user.

*Compliance:* Adherence to application-related standards, conventions, regulations in laws and protocols.

## Efficiency

The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions.

Sub characteristics:

*Time Behavior:* The speed of response and processing times and throughput rates in performing its function.

*Resource behavior:* The amount of resources used and the duration of such use in performing its function.

*Compliance:* Adherence to application-related standards, conventions, regulations in laws and protocols.

## Maintainability

The capability of the software product to be modified. Modifications may include corrections, improvements or adaptations of the software to changes in the environment and in the requirements and functional specifications.

Sub characteristics:

*Analyzability:* The effort needed for diagnosis of deficiencies or causes of failures, or for identification parts to be modified.

*Changeability:* The effort needed for modification fault removal or for environmental change.

*Stability:* The risk of unexpected effect of modifications.

*Testability:* The effort needed for validating the modified software.

*Compliance:* Adherence to application-related standards, conventions, regulations in laws and protocols.

## Portability

The capability of the software product to be transferred from one environment to another. The environment may include organizational, hardware or software environment.

Sub characteristics:

*Adaptability:* The opportunity for its adaptation to different specified environments.

*Installability:* The effort needed to install the software in a specified environment.

*Co-existence:* The capability of a software product to co-exist with other independent software in common environment.

*Replacability:* The opportunity and effort of using it in the place of other software in a particular environment.

*Compliance:* Adherence to application-related standards, conventions, regulations in laws and protocols.

## Quality Aspect 1&2: Functionality - Security & Interoperability

The most important sub characteristics from the main characteristic Functionality are Security and interoperability. These quality sub characteristics will be explained on what they are and why they are important for the X Window System.

### Security

By default the X Window Server does not provide the user with many standard security options. This has the advantage that a person installing and working with X System has his system up and running in no time without having to put much effort in securing his X System. However X System does offer a variety of possibilities for the operating user on how to improve his security which will be discussed below.

### Host & Token authentication

The host and Token authentications are the most used approaches to secure an X Server. Host authentication is the potential acceptance of a connection based on its origin. Typically, this would be determined by the IP address of the connection's host. The Administrator of the X Server can define in a host file which IP addresses are allowed to connect to the X Server and which not. A program called xhost is available to control on a host-by-host level which hosts can display clients on the X Server. Xhost has made it very easy to add or remove hosts on a X Server without much knowledge needed (Fisher, 2002).

The second most used approach is the Token authentication. In this form of authentication the client is verified based on the token they offer. Using a program called xauth, each client is given a "magic cookie", a random value which it must offer the X Server to be allowed access (Fisher, 2002). Xauth works by creating a file called .Xauthority in a clients home directory. The X server reads this file and then requires any X program that is being executed to provide the same code listed within this file.

The .Xauthority file permission are standard chmodded to 600, preventing anyone else on the system from reading it. Furthermore MD5 encryption can be used for generating a random non-reproducible number (Gross & Buehler, 2002).

### Secure Shell

X Windows System does not provide any form of encrypted communication between both the client(s) and server. Due to this it is possible for potential attackers to use packet sniffers or other logging software to either listen to the requests (for example typing in a password) and responses communicated between the server and the client or in the worst case scenario send fake requests to either the client or server (Sachs, 2001). However when setting up Secure Shell (SSH) tunneling any user can make the communication between both the server and client encrypted. When using SSH's tunnelling, a user sets up his X server with Xhost security, but it says the only host it should trust is the localhost. Then you use SSH in place of telnet to login to an account on the server. As part of this login process, the SSH client negotiates with the server on the SSH server and together, they automatically set up a secure X-Windows connection with encrypted communication.

### Security Extension Specification

In 1996, an Security Extension was developed to help improve the security on X Windows Systems (Wiggins, 1996). It contains a new protocol needed to provide enhanced X server security. It contains the possibility to check whether an integrated extension is considered secure or insecure. This is based upon the following aspects; Checking if a client is trusted or not, Keyboard, Image and Property security related aspects. In addition a New authorization method was introduced named 'XC-QUERY-SECURITY-1' allowing an external agent such as the X firewall proxy to probe an X server to determine whether that server meets certain security criteria without requiring the agent to have its own authorization for that server. The agent may use the returned information to make a decision. For example, the X firewall proxy may choose not to forward client connections to servers that do not meet the criteria.

### Interoperability

Interoperability is the sub characteristic referring to the ability of diverse systems and organizations to work together. X Window System has been created for many different operating versions such as Microsoft Windows, Mac OS, UNIX, VMS and MVS. Although the initial implementation work was made for UNIX, it was clear that the network protocol should not depend on the aspects of the operating system (Scheifler, 1986). Nowadays, Microsoft Windows is no longer shipped with support for X, but many third-party implementations exist, mostly free and open source such as Cygwin/X and Xming.

X Windows operates over TCP, which is used as a reliable stream connection. TCP is one of the main protocols of the transport layer of the TCP/IP model used by all common systems. In case both the X server and client are running on the same machine, IPC (Inner Process Communication) is used. IPC has the advantage over TCP that the overhead of packet building and deciphering does not occur. In addition to this, X Windows also supports a large variety of X Window Managers such as Blackbox, Fluxbox and WindowMaker. Furthermore complete user-friendly desktops have been created who run in conjunction with a window manager, such as GNOME, KDE Software Compilation and Xfce.

### Quality Aspect 3: Reliability – Maturity

The X Window system implementation was driven by the need of applications that could be used in campus area network. The goal was to achieve synchronous resource creation in all the systems of the network, despite the heterogeneity of their hardware configuration. Thus, the implementers of X Window regarded the high level of a good performance as a vital quality aspect for their system. One of the performance requirements they specified for the final system is that it should provide high-quality support for text, 2-D graphics and imaging (Scheifler & Gettys, 1986). The visualization of graphical components is taking place “immediately” and “transparently” in workstations and mainframes connected through local-area networks.

While the widespread use of X Windows initiated back in the late 1980’s and the evolution of it has never stopped since then, it is plausible to consider it as a system that have reached an important level of maturity. Through all these years there were 29 releases that improved and updated in many ways the system, whilst there are plans for development of another two forthcoming releases in the following two years. Apart from the introduction of new features, the aim of each new release is to correct bugs and malfunctions of the older versions. Thus, the frequency of failure tends to decrease over the years.

| Version   | Year | Consists of  |
|-----------|------|--|
| X1        | 1984 | First use of the name X, fundamental changes from product W  |
| X6        | 1985 | First version licensed to handle of outside companies  |
| X9        | 1985 | Color version. First release under MIT license.  |
| X10       | 1985 | IBM RT/PC, AT (running DOS) and others   |
| X10R2     | 1986 | Bug fixes  |
| X10R3     | 1986 | First freely redistributable X release. UWM standard window manager.                                   |
| X10R4     | 1986 | Last version of X10  |
| X11       | 1987 | First release of current protocol  |
| X11R2     | 1988 | First X Consortium release   |
| X11R3     | 1988 | XDM display manager added  |
| X11R4     | 1989 | XDMCP added, TWM as window manager, new fonts  |
| X11R5     | 1991 | Introduction of PEX, Xcms (color management), X386 and X video extension                               |
| X11R6     | 1994 | ICCM v2.0 introduced, Inter-Client Exchange, X Session Management, X Image extensions, XTEST extension |
| X11R6.1   | 1996 | X Double Buffer Extension, X keyboard extension  |
| X116.2/3  | 1996 | Web functionality, LBX, Xprint and Xlib implementation   |
| X116.4    | 1998 | Xinerama   |
| X11R6.6   | 2001 | Bug fixes, XFree86 changes   |
| X11R6.7.0 | 2004 | First X.Org foundation release, removal of XIE, PEX and libxml2  |
| X11R6.8.0 | 2004 | Window translucency, Xfixes, XDamage   |
| X11R6.8.1 | 2004 | Security fix in libxpm   |
| X11R6.8.2 | 2005 | Bug fixes, driver updates  |
| X11R6.9/7 | 2005 | EXA, source code refactoring, modular autotooled version   |
| X11R7.1   | 2006 | EXA enhancements, KDrive integrated, AIGLX   |
| X11R7.2   | 2007 | Removal of LBX, XCB added, autoconfig improvements   |
| X11R7.3   | 2007 | XServer 1.4, DTrace probes, PCI domain support   |
| X11R7.4   | 2008 | XServer 1.5.1, XACE, PCI-rework, GLX 1.4   |
| X11R7.5   | 2009 | XServer 1.7, Xi2, XGE, MPX, DRI2, SELinux module   |
| X11R7.6   | 2010 | XServer 1.9, XCB requirement   |

**Table 1 - Release history**

The fault tolerance and recoverability quality sub-aspects in X Windows are realized through network redundancy which eliminates communications errors. While X Windows operates over TCP/IP, there is a two-layer scheme to support IP addressing. The extra layer helps to detect faults in the communication between terminals and to re-establish the connection with dynamic relocation.

### ***Quality Aspect 4&5: Usability - Attractiveness & Understandability***

One of the most important X Window architecture's aims is to enable applications to be run in different environments, with various operating systems that are running in diverse hardware configurations. Therefore, the focus is mainly on the capabilities of software that is able to run no matter how primitive or advanced the hardware is. Nevertheless, latest developments in hardware can provide new and more attractive ways of presenting the graphical environment of software applications. In order to be exploitable, all these technological achievements require hand-in-hand collaboration with the innovative and quite specific hardware (e.g., a 3-D graphic card). That is why hardware manufacturers are publishing their own software, for instance drivers, in order to support their systems.

Such a perspective is against the main principle of X Window, which demands applications that are able to be run in heterogeneous environments. For this reason, compliance is a principle quality sub-aspect that drives the development of X Window applications. The X Window architecture was designed in adherence to standards and regulations of existing hardware. This perspective renders the system aptitude to operate in – ideally – any kind of environment.

Understandability and learnability are also some quality aspects that characterize Window X applications in a high level; it is effortless for every user to be familiar with simple graphical representations, and to learn using the software quickly and adequately. However, while attractiveness can be affected positively by advanced technology, it can help by the same token the operability, understandability and the learnability of the applications.

The representation of the available functions of an application can be highly enhanced by advanced, easy-to-use and memorable graphical items, and thus leading to understandability, operability and learnability to even more high levels. Consequently, although the graphical environment of X Windows could be enhanced by advanced technology in hardware, the above-mentioned quality sub-aspects reach high levels in X Window applications.

## Quality Aspect 6: Efficiency – Resource Behavior

The most important sub characteristics from the main characteristic Efficiency Functionality is Resource Behavior. This quality sub characteristics will be explained on what it is and why this is an important sub characteristic for the X Window System.

Resource behavior characterizes the resources used, i.e. memory, CPU and disk and network usage. Both memory and disk usage are one of the pro's when using X Windows. The X Server is about 5MB disk size while it will take a maximum of 20MB in the users memory.

| <u>Name</u>   | <u>Last modified</u> | <u>Size</u> | <u>Description</u> |
|---|----------------------|-------------|--------------------|
|  <a href="#">Parent Directory</a>          |                      | -           |                    |
|  <a href="#">xorg-server-1.7.1.tar.bz2</a> | 22-Oct-2009 22:40    | 4.7M        |                    |
|  <a href="#">xorg-server-1.7.1.tar.gz</a>  | 22-Oct-2009 22:39    | 6.3M        |                    |

Figure 4: X Server 1.7.1 download size

Also the X Client does not take much room, this is all together approximately both 10mb on disk and memory usage.

The amount of CPU usage on a clients pc mainly depends on the kind of X Window Manager that is being used to access the X Server and Client. X Window Managers come in many different formats and sized. Since a user can pick whatever X Client Manager he prefers, the CPU usage will also depend on this.



Figure 5 – Most popular X Window Managers

In addition, the X Window System has a support for a wide variety of Graphic Cards supporting both OpenGL and Direct3D. While configuring X Window xorgconfig will ask for the type of Video Card from 3Dlabs till ATI.

## *Quality Aspect 7: Maintainability – Analyzability*

The most important sub characteristics from the main characteristic Maintainability Functionality is Analyzability. This quality sub characteristics will be explained on what it is and why this is an important sub characteristic for the X Window System.

Analyzability is the sub characteristic describing the ability to identify the root cause of a failure within the software. X Windows has different logging and debugging options available.

First of all X Windows has the possibility to debug X protocol errors containing of five different pieces of information; error types, major opcode, invalid value/resource, serial number of the failed request and serial number of the current request (Lee, 1993). These protocol errors can be debugged with using special tools, such as the UNIX dbx debugger or the xscope protocol analyzer (Lee, 1995).

Secondly, X Windows has the possibility to debug the X Server itself. This can be done by starting X in debug mode (`startx -debug`) or by using the command `'debuginfo'` usually available on modern UNIX based distributions. The `startx` command is a front end script to `xinit` used to start an X server. Using `'startx -debug'` gives the user the possibility to see each step from the X Server startup until the point of error both on screen as in the log file `/var/log/Xorg.0.log`.

Thirdly the client side is being analyzed through the X Window Manager. The X Window Manager is the graphical output for a user and errors and exceptions will be shown as error reports on the user its screen or in application exception logs.

## *Quality Aspect 8: Portability – Adaptability*

Portability is another primary quality requirement that affected X Window architecture from the early implementation process (Scheifler & Gettys, 1986). It was stated from the beginning that the network protocol and the operation of X Window applications should not depend on aspects of the operating system. The outcome was indeed a system that allows the transfer of the software from one configuration to the other, without implications to its flawless performance. Different systems based installability on various and not-compatible processors are able to host Window X applications.

The diversity of the operating systems is not hindering the portability of the applications, as long as the TCP communication standard is supported fully. Applications are also devoid of compatibility issues by the variety of visualization sub-systems in each hardware configuration. They can be installed smoothly by transferring the distribution files from one computer to the other, while the source files can be compiled in any of the supported hardware configurations. Thus, installability is ubiquitous in every X Window application and reaches very high levels.

The adaptability of the system to support different environments is another crucial sub-aspect for X Window applications. Disparate hardware configurations have been taken into account and every application is capable of exploiting the available resources during operation.

## Trade-off between aspects

Within this chapter a trade-off between different quality aspects is being discussed. Which of the discussed quality aspects discussed in the previous chapter have a negative influence for other quality aspects, in other words what is the downside by focusing on the earlier mentioned quality aspects for the X Window System?

## *Security vs Operability and Interoperability*

As explained in the previous chapters the X Windows System does not provide the user with many standard security options. Although this lack of standard security options is great for the operability and interoperability for a system it does cause problems when a user decides to modify the X Windows System by adjusting its security level.

Whenever a user sets host authentication on the server all different clients need to be allowed by the server. This is not a problem when the server and client run on the same machine but when running it in combination with an X Terminal hundreds of clients could be connecting towards the X Server. An X terminal is a computer especially designed to provide a low-cost user interface for applications that run in a network. Typically, X terminals are connected to a server running a UNIX-based operating system on a mainframe, minicomputer or workstation. This affects the interoperability, where all individual clients need to be allowed to connect on the X Terminal in order to be able to use the interface supplied by the X Server.

Furthermore the more different extensions people install in order to improve security such as the SELinux module and the XC-QUERY-SECURITY-1 (Wiggins, 1996) the more difficult it becomes to use the software in combination with other types of software and operating the X Window System with other software. When installing XC-QUERY-SECURITY-1, a person can give enhanced security over the keyboard and mouse commands, interfering with other applications who also need to use the X Window Manager but are not being seen as trustworthy applications. This can often result in access and exception errors. This additional level of security means the user not only has to define all the clients who are trustworthy, but also on which level they have access and which extensions they may use.

The user operating the system needs to spend much more time in analyzing the different kind of access and exception logs while also taking more time in account to configure the X Window System.

## *Attractivity vs Understandability*

As explained in the previous chapters X Window Systems consists of a Server, Client, Protocol and a Window Manager. One of the positive quality aspects hereby is the attractiveness of the Window Manager. Each user has the possibility to choose which Window Manager suits him the best to his needs.

There are tons of different Window Manager all available handing the requests and responses by the X Window Systems. However since one of the fundamental design goals of X was to separate the window manager from the window server a particular policy for human-computer interaction was not created. While this might have seemed a good idea from the point of research and development perspective it can create a veritable user interface. (Garfinkel, Weise & Strassman, 1994).

The lack of a policy for setting design guidelines can create problems that each X terminal can be created in a different way. A Mouse click or keyboard short cut can have different meanings on different machines and different terminals. Due to this end-users have to figure out different commands and interaction ways with similar X Terminals depending on each individual machine.

As a result, the ICCCM (Inter Client Communication Conventions Manual) was created. It describes protocols that X clients use to communicate with each other via the X server, including different topics like window management, selections, keyboard and colormap focus, and session management. (Rosenthal & Marks 1993) However by the time the ICCCM was published people were already writing window managers and toolkits forcing each new version of the ICCCM to be made backward compatible with problems created in the past. Due to this and the difficulty implementing the ICCCM compliance with X toolkits, window managers and even simple applications it still doesn't work. Anybody who would want to write an interoperable ICCCM compliant application would have to crossbar test it with every other application, and with all possible window manager. Since people already started writing X Window Managers and X Toolkits before the ICCCM was even released users still suffer a great deal of lack of design guidelines these days. A user switching from one Window Manager onto the next is forced to re-learn his way around the application and find out how this particular manager works. This all does not work in favor of the understandability of the X Window Systems.

## *Usability vs Portability*

As we have mentioned in the previous sections, X Window system is based on a communication protocol to support the client-server model: an X server is connected to various client programs through the TCP/IP standard. The use of the network, bandwidth and latency can both be significant concerns that affect the usability of the system. Bandwidth restrictions are permitting the transfer of limited size of data, while latency becomes an insurmountable obstacle in the interactivity of applications in responses between the client and the server. In these cases, we can just install the client applications and the server to the same machine. The communication is then absolutely direct without the need of a network for the operation of the applications. Developers can enhance their applications by allowing them to exchange unrestricted amounts of data and consequently lead to very high levels for the usability quality aspect of the system.

Nevertheless, X Window architecture was implemented in such a way that all device dependencies should be encapsulated by the server (Scheifler & Gettys, 1986). To cite an example, the addition of a new display type requires the addition of a new server implementation; there is no need for the applications to be changed since all device dependencies exist on one end of a network connection. Applications are then easily transferred among diverse hardware environments, and thus helping the system to reach high levels of portability. The decision though to combine the installation of applications with the installation of the server, and to develop applications without the bandwidth and latency restrictions, it simply reduces the portability of the applications to a great extent.

According to the fundamentals of X Window system, the client-server model through network communication is the feasible way to applications are truly independent from any hardware configuration. If the focus is laying on the usability, it will jeopardize the portability of applications to the diverse environments. In this case, applications are becoming depended on the local resources. Communication over the network is simply not applicable to support remote connection with the server and the portability aspect decreases dramatically.

## Realization and technical aspects of the implementation of aspects

Within this chapter the realization and technical aspects of the implementation of the earlier described quality aspects and their tradeoffs will be discussed.

### Security

As stated before X Window has several methods for ensuring security. In this section we will demonstrate how security is implemented in X Window. As explained in the previous chapter the more security measures taken by the user, the worse the operability and interoperability of the system becomes. A user has to choose which kinds of methods will be used in order to establish a confident level of security without having too much of a negative effect on the operability and interoperability of the system. This chapter will start off with a list of the threads when X Window is not secured, showing the risks a user faces when applying different kinds of security levels:

- Read the user's keyboard, including any passwords that the user types.
- Read everything that is sent to the screen.
- Write arbitrary information to the screen.
- Start or terminate arbitrary applications.
- Take control of the user's session.

Here is an example to show how vulnerable an unprotected X Window is, we give an example.

- With this line anyone can reverse the mouse button functions, making left click a right click and vice versa:  
local.host> xmodmap -display remote.host:0.0 -e "pointer = 3 2 1 4 5"
- With just one simple command, anyone can capture key logs at the unprotected server by performing the following command:  
local.host> xkey remote.host:0.0

## Host authentication

The following commands show how to enable Host authentication in X Window:

- To grant access to every remote machine (not recommended)  
local.host> xhost +
- To disable any incoming request, try:  
local.host> xhost -
- To grant a specific remote.host machine access to the display. This has to be done for each client that is allowed access:  
local.host> xhost +remote.host
- To remove permissions from a specific remote.host machine to access the display:  
local.host> xhost -remote.host

This is the most used and simplest form of authentication in X Window. Note that the server is still vulnerable if the remote machine is infected or unauthorized persons have access to an allowed host. Every allowed host has to be added individually and the host list must be maintained regularly. This is a vulnerability if the host list is not maintained. Another thing to be aware of is when the remote host is not on a secured intranet, but on the internet. Access will have to be granted based on an IP address. You cannot always be sure there are no other machines on the network of the remote IP address. Host authentication affects interoperability because a client cannot be moved to another location without a change in the Host list of the server.

To tackle this problem X Window has Token authentication or User based authentication.

## Token Authentication

Earlier in this document "MIT-MAGIC-COOKIE" was mentioned. When an X session is started the X Server generates a random string of bytes. This is the authorization record which is stored in the .Xauthority file in the user's home directory. If an X client is started on the local machine it automatically attempts to read this file and supplies it to the X Server as its cookie to get access. To grant access to accounts on other machines this file has to be ported to each client. To do this the `xauth` command can be used to extract the file. Now the file can be distributed to your clients. To merge the file with your clients .Xauthority file, the `xauth` command can be used again on the client machine. With this setup clients with the cookie can connect from any location. The token authentication isn't fool proof either. Someone with the ability to run a sniffer on a computer connected to your network can capture the network packets and decode the cookie. He could then use this to connect to your server.

## SSH

In order to make sure an attacker cannot capture the network packets sent between X Server towards X Client a user can choose to install SSH.

When SSH is installed the user has two ways in defining how to have the communication of the X Windows;

- either by adding particular hosts towards the ssh config file on  
/home/<os>/.ssh/config  
Host <hostname> Forward X11 yes
- either by adding all hosts towards the ssh config file on  
/home/<os>/.ssh/config  
Host \* Forward X11 yes

By using X Windows in combination with SSH the user avoids the risk of packets being captured by third parties.

### Toolkits (Attractivity vs Understandability)

The first design of X Window did not include graphical user interfaces. Xlib was created to handle communication between the client and the server. For user interfaces Xt (X Toolkit Intrinsic) was created. Xt is a library that uses the low level Xlib library and offers a more friendly object oriented API for developing X11 software with widgets.

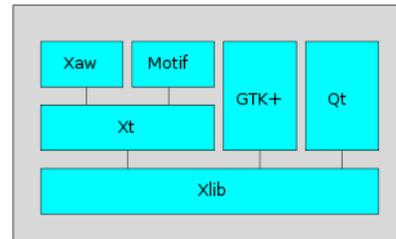


Figure 7 – Xt and related libraries

Widgets are all the items seen on a screen such as buttons, labels, checkboxes, menus, etc.. A window itself is actually also considered a widget, as it is also a graphical user interface.

Using the Xt library (See figure 8 for a X window application structure including the Xt library) a programmer can easily create new widgets using C or C++ language. Since most applications will use a standard set of widgets, standard libraries (toolkits) like Xaw and Motif were created. Xaw and Motif rely on Xt to create their widget. Other modern toolkits like FLK, GTK+ and Qt do not rely on the Xt library, but use the Xlib directly. That being said, a programmer is always free to create his own widgets by talking directly to the Xt or Xlib library. Usually a toolkit is used, because these two libraries are very hard to understand. Given the choice of toolkits out there, it's rare for a programmer to create his own widget.

### X Window application Structure

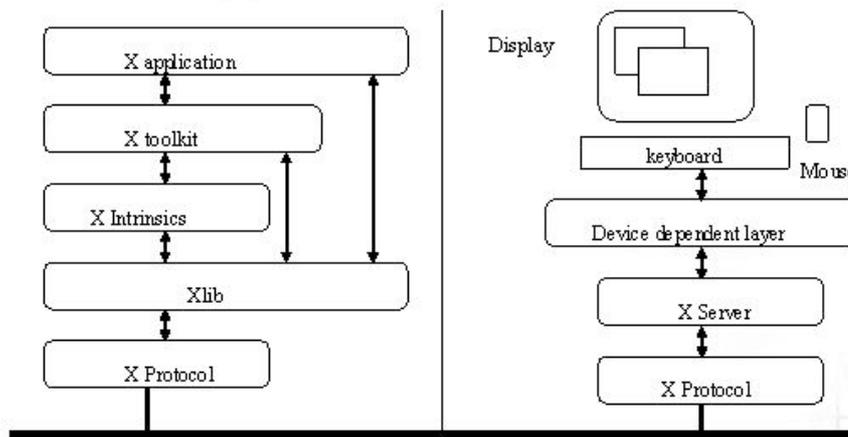


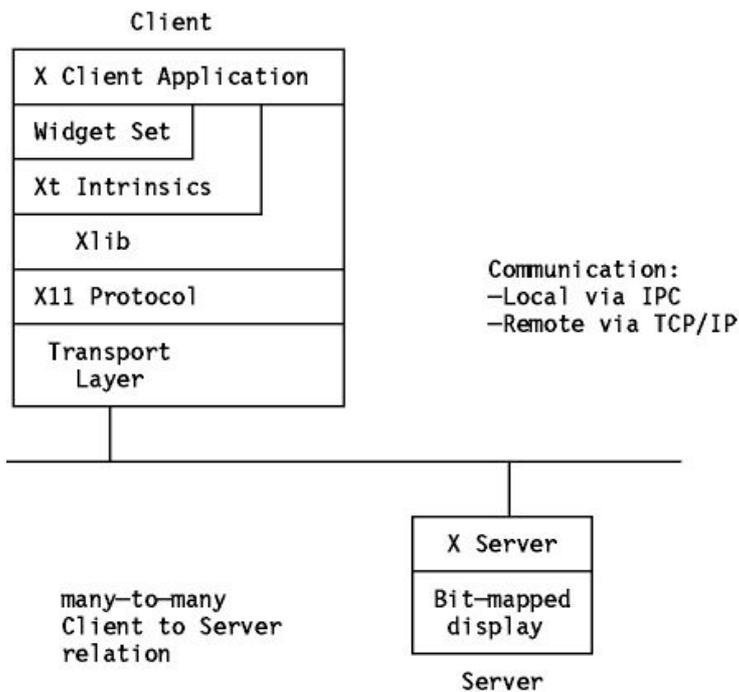
Figure 8 – X Window application Structure with Xt Library

Normally speaking all widgets in a toolkit have the same look and feel. This gives the user the feeling that all components of a screen belong together, making the application look coherent. Because different applications the user is working could have been made with different toolkits, it might become confusing for the user. That's why Linux graphical interfaces are made with the same toolkit, to make everything look de same. GNOME for example was made with the GTK+ toolkit.

## Usability vs Portability

One of the great main quality characteristic of the X Window System is the capability to learn understand and learn X while it also contains GUI interfaces that make it look attractive.

As previously explained in the first chapter, overview of X Windows, the X Window by default only consists of a small set of components (Client, Server, Protocol and Xlib) to support the X System.



**Figure 9 – Components of X System**

These components (See Figure 9) together allow the communication between both the client and server. Furthermore by default the user only needs to know the following important commands in order to successfully work with the X Windows;

- Startx ; enter startx in a console window to start a new X session (server side) with an user interface
- xinit; runs client side scripts (client side, normally automatically called by startx)
- xterm; start a client side X Window session for one process

This allows high usability for any new user to start working with the system. In addition, as mentioned in the previous chapter Quality Aspect Efficiency the user can quickly install any X Window Manager he desires fitting his needs. In case any user runs both the Server and Client on his own machine, X Systems will automatically switch towards IPC instead of TCP. All together, this forms great usability for any user, having an quickly to understand and learn the X Systems who are easy to operate and look as attractive as the user desires.

At this moment the portability of the X windows System is still at acceptable levels. The default installation of the X Windows can still easily be adapted to different environments and hardware and software can easily be adapted to the system.

However when a user decides to make the system more complex and expands it with different kind of extensions the X windows system becomes less portable.

The more extensions a user installs, the harder it becomes to adapt the system to different environments and add new software.

For example a user can expand the system with the following extensions to improve his usability while affecting the portability of the system;

- Installations of extensions to improve usability of the security

A user can choose to install the extension 'XC-QUERY-SECURITY-1' giving him the possibility to configure his X System at any given level and define exactly which client (application) has which kind of access (policies) and what is allowed to run at the X system.

For example performing the request SecurityGenerateAuthorization;

```
SecurityGenerateAuthorization <authorization-protocol-name>  
<authorization-protocol-data><value-mask><value-list><authorization-  
id><authorization-data-return>
```

The SecurityGenerateAuthorization request causes the server to create and return a new authorization with specific characteristics, whereas the user can define the level of access within the value-mask and value-list. Clients can subsequently connect using the new authorization and will inherit some of the characteristics of the authorization.

However while the user has more control over his security and has more ways of controlling and using the system the portability of the system goes down. It becomes harder and more difficult to install new software, since the new software has to be able to deal with the newly installed extensions and be able to co-operate with them.

Another example is as next;

- Installation of extensions to improve the usage of video output

A user can also choose to install Xvideo (Xv) improving the usage of video output. This extensions main use is to rescale video playback in the video controller hardware, in order to enlarge a given video or to watch it in full mode. However this does mean that the user must define which graphical card is available in the system. This again affects the portability, a graphical card can no longer be easily replaced without previously defined configurations needing to be changed.

## Comparison with 2 other systems

Within this chapter a comparison will be made with 2 other systems compared to the investigated X Windows System. The systems hereby compared mentioned are both Wayland and microXwin. The next chapter will explain what Wayland and microXwin are, and how they differ from X Windows System.

### *X Window vs Wayland*

#### Wayland

Wayland is a display server protocol for the Linux. Wayland provides a method for compositing window managers to communicate directly with applications, eliminating their dependence on X, allowing them to communicate directly with video and input hardware, becoming the display server. Applications render graphics to their own buffers, and a Wayland display server composites those buffers into the screen image. This is a simpler and more efficient approach than using a compositing window manager with the X Window System. It's supposed to be a complete replacement for X.

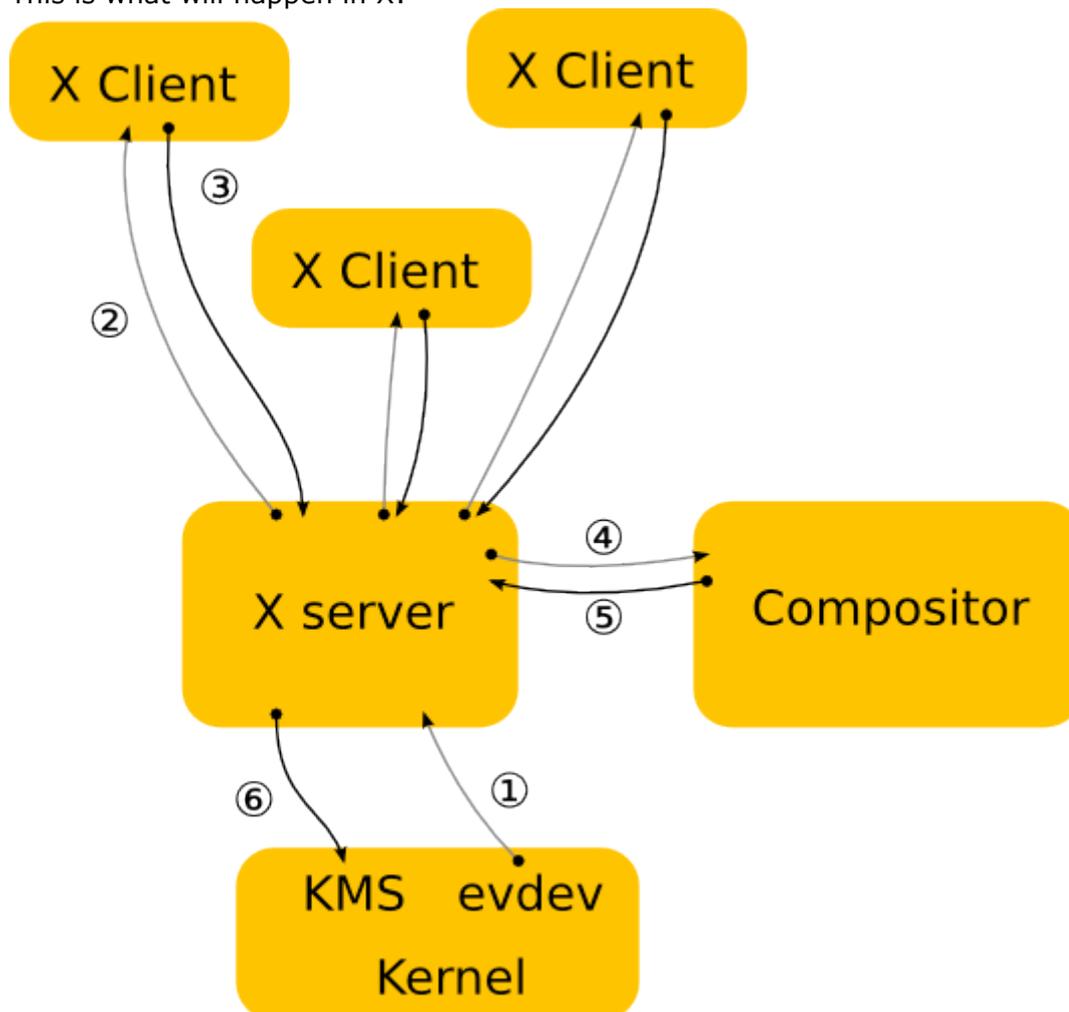
Ubuntu recently announced it will switch in the future to Wayland instead of X Window. So it should be interesting to compare the architecture of the two. Mark Shuttleworth, the founder of the Ubuntu project stated:

"We don't believe X is set up to deliver the user experience we want, with super-smooth graphics and effects," Shuttleworth explained. "I understand that it's \*possible\* to get amazing results with X, but it's extremely hard and isn't going to get easier."

#### Comparison

A good way to compare the Wayland architecture with the X architecture is by illustrating the differences with a simple use case (see figure 10) on a Linux machine. Let's assume the user clicks on a checkbox.

This is what will happen in X:



**Figure 10 – Example of pressing a checkbox on X**

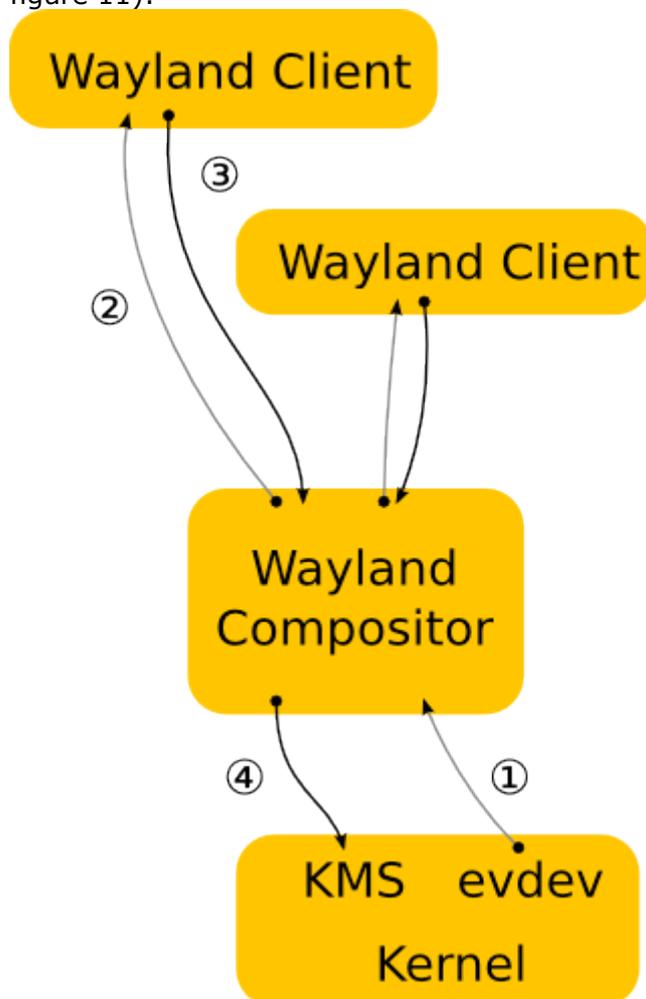
1. The kernel registers the mouse click from the user and sends it to X through the evdev input driver. The kernel does all the hard work here by driving the device and translating the different device specific event protocols to the linux evdev input event standard.
2. The X server determines which window the mouse click belongs to and sends it to the client controlling that window. The X server doesn't know anything about the window. The window location, size etc. is controlled by the compositor (window manager). The server just forwards the click event to the client application.
3. The client looks at the event and decides what to do. Often the UI will have to change in response to the event. In this case a check box was clicked. The client now sends a rendering request to the X server to draw a check in the box.
4. When the X server receives the rendering request, it sends it to the driver to let it program the hardware to do the rendering. The X server also calculates the bounding region of the rendering, and sends that to the compositor.
5. The compositor now knows something has changed and it has to redraw a part of the screen where the window is located. The compositor is responsible for rendering the entire screen contents based on its

scenegraph and the contents of the X windows. Yet, it has to go through the X server to render this.

6. The X server receives the rendering requests from the compositor and either copies the compositor back buffer to the front buffer or does a pageflip. In the general case, the X server has to do this step so it can account for overlapping windows, which may require clipping and determine whether or not it can page flip. However, for a compositor, which is always full screen, this is another unnecessary context switch.

As you can see above the X server is like a middleware that handles the communication between the client, window manager and kernel. Critics say that this causes a lot of unnecessary overhead.

The Wayland protocol lets the compositor send the input events directly to the clients and lets the client send the damage event directly to the compositor (see figure 11):



**Figure 11 – Example of pressing a checkbox on Wayland**

1. The kernel gets the click event and sends it to the compositor. This is similar to the X case, except that the X Server is replaced by the Wayland compositor.
2. The compositor looks through its scenegraph to determine which window should receive the event. The scenegraph corresponds to what's on screen and the compositor understands the transformations that it may have applied to the elements in the scenegraph. Thus, the compositor can pick

the right window and transform the screen coordinates to window local coordinates, by applying the inverse transformations.

3. As in the X case, when the client receives the event, it updates the UI in response. But in the Wayland case, the rendering happens in the client, and the client just sends a request to the compositor to indicate the region that was updated. Wayland uses direct rendering. With direct rendering, the client and the server share a video memory buffer. The client links to a rendering library such as OpenGL that knows how to program the hardware and renders directly into the buffer. The compositor in turn can take the buffer and use it as a texture when it composites the desktop. After the initial setup, the client only needs to tell the compositor which buffer to use and when and where it has rendered new content into it.
4. The compositor collects requests from its clients (which buffers changed) and then recomposites the screen.

Source: <http://wayland.freedesktop.org/architecture.html>

### Conclusion

As demonstrated above Wayland uses an entirely different way of rendering windows on the screen. Using direct rendering the X Server implementation becomes obsolete. This is of course not without any drawbacks. By using direct rendering network transparency is lost. There is no longer a complete independence between the client and the host the server is on. In the case of X, the X server does all communication with the hardware. In Wayland the client had direct access to the hardware. This makes it more difficult to put the client on a separate machine. This separation was the original thought behind X window. In operating systems like Ubuntu this network architecture is more of a drawback than an advantage. What use is it if all clients and the server or on the same machine. For implementations of such a system Wayland will give a better use experience than X Window, especially with 3D graphics and elegant user interfaces. If taken into account that nowadays in most X Window implementations the client and server are on the same machine, why have an X Server between the client and the hardware. As Mark Shuttleworth stated in his explanation why Ubuntu chooses will choose Wayland in the future:

“Some of the core goals of X make it harder to achieve user experiences on X than on native GL, we’re choosing to prioritize the quality of experience over those original values, like network transparency.”

## *X Window vs MicroXwin*

MicroXwin is promoted as a lightweight replacement for X. Although it is not a full replacement for X Window it does offer some performance advantages over X Window. MicroXwin abandons the client-server architecture of X Window. The X Server is in this case replaced by a kernel module which does the graphic processing. MicroXwin is still compatible with X clients and toolkits because it provides an interface to XLib.

MicroXwin provides the following advantages:

1. An improvement of 62% for asynchronous display or 384% for synchronous display of images of a 100x100 size.
2. There are only about 300 Kbytes of kernel memory in use by the kernel module. [X.Org server](#), however, has a run-time memory usage of 12MB.
3. The smallest MicroXwin distribution can fit within 1 megabyte of disk space in contrast to the X.Org Server, which has a disk footprint of 1.8MB.

Source: <http://www.microxwin.com/performance.html>

### **Conclusion**

MicroXwin like Wayland provides better window drawing performance than a standard X Window setup. What neither of them can provide is the network capabilities X Window has to offer. To separate the client from the window manager is not a part of the architecture in MicroXwin or Wayland.

As you can see there are some 'alternatives' to X Window. 'Alternative' is between brackets, because these systems discard the most important aspect of X Window, the network oriented aspect. They both promise improvement by leaving out the networking part. Clients and the server cannot be on different machines anymore. At least not as easy as X made it. As far as the network oriented graphics systems go, there is no alternative to X with the same networking capabilities. Once an attempt was made, with Y Window system, but development on that stalled in 2005.

## Questions from other Group members

Within this chapter all questions submitted by the other groups are explained. In case the answer has been described in the document a reference is made towards the chapter.

### Questions Group A

**Persons name: Mark Rouhof**

*Question:*

"All graphics and text requests include a logic function and a plane-select mask (an integer with the same number of bits as a pixel value) to modify the operation. All 16 logic functions are provided, although in practice only a few are ever used. "

Paragraph 6.2 Graphics

Which are those 16 logic functions and which are in practice used?

*Answer:*

The function in a GC defines how the new destination bits are to be computed from the source bits and the old destination bits.

**GXcopy** is typically the most useful because it will work on a color display, but special applications may use other functions, particularly in concert with particular planes of a color display.

The 16 GC functions, defined in `x11/x.h`, are:

---

| Function Name         | Value | Operation               |
|-----------------------|-------|-------------------------|
| <b>GXclear</b>        | 0x0   | 0                       |
| <b>GXand</b>          | 0x1   | src AND dst             |
| <b>GXandReverse</b>   | 0x2   | src AND NOT dst         |
| <b>GXcopy</b>         | 0x3   | src                     |
| <b>GXandInverted</b>  | 0x4   | (NOT src) AND dst       |
| <b>GXnoop</b>         | 0x5   | dst                     |
| <b>GXxor</b>          | 0x6   | src XOR dst             |
| <b>GXor</b>           | 0x7   | src OR dst              |
| <b>GXnor</b>          | 0x8   | (NOT src) AND (NOT dst) |
| <b>GXequiv</b>        | 0x9   | (NOT src) XOR dst       |
| <b>GXinvert</b>       | 0xa   | NOT dst                 |
| <b>GXorReverse</b>    | 0xb   | src OR (NOT dst)        |
| <b>GXcopyInverted</b> | 0xc   | NOT src                 |
| <b>GXorInverted</b>   | 0xd   | (NOT src) OR dst        |
| <b>GXnand</b>         | 0xe   | (NOT src) OR (NOT dst)  |
| <b>GXset</b>          | 0xf   | 1                       |

For further details of the use of these functions, please refer to: <http://www.tigr.net/afterstep/X/xlib/GC/manipulating.html>

**Persons name: Kevin van Ingen**

*Question:*

The requirements mentioned in the paper by Scheifler, R. W., & Gettys, J. (1986) describe that the X window system should be able to support a variety of displays and input peripherals. Multi-touch is being increasingly popular on multiple platforms. Is the X window system capable of supporting this method of input?

*Answer:*

Indeed, there is already in existence an application that makes it feasible to use multi-touch screens. Multi-pointer X (MPX) is a modification to the existing implementation of the X Window System that uses multiple independent mouse cursors to simulate multiple touch points. These cursors are all connected to the same computer and the multiple input commands occur concurrently, allowing the use of a multi-touch screen. Existing X Window applications are ready to exploit this technology without modification, as MPX is installed in the server's computer that has to run an operating system such as Linux which supports multiple pointing devices.

**Persons name: Robert Vroon**

*Question:*

X-window is a server client architecture, so there is the need to transmit information to the clients. How is security for this information provided?

*Answer:*

As discussed in the quality aspect Security X Window Systems has the possibility to transmit information between clients using Secure Shell. This way information been send between server and clients is encrypted and can not be intercepted and read by third parties.

**Persons name: Sjors Otten**

*Question:*

Question: X-Window-System has a server-client pattern. Both client and server can run locally on the same system and communicate via IPC or they can be physically separated and communication is handled via TCP/IP, DECnet and Chaos. I assume that communication via TCP/IP runs over IPv4. As you all know IPv4 is about to run out of ip-adresses.

Hence, my question: Is the X-Window-System able to cope with new network protocols like TCP/IPv6?

*Answer:*

According to the official specifications of the X Window system, there is native support for TCP/IPv6. The communication through the new protocol was incorporated to the system for the first time in a release which was published back in 2004, and nowadays the operation in the most recent versions has been stabilized and optimized.

## Questions Group B

**Persons name: Alessandro Vermeulen**

*Question:*

How easy is it to use modern developments in X-Windows? For instance, a lot of applications now make use hardware acceleration. How does this scale up when you have to run multiple applications on 1 server? I imagine that would slow things considerably.

In my experience with X-Windows the programs that ran over X-Windows are very slow, do not integrate well into the host (where host is the machine of the viewer) environment, and in every way act like they are running on the remote machine. This is certainly a design decision, and I can imagine that there are many reasons why this has become this way. However isn't there a way to let the programs have a look and feel that is more suitable for the environment where they will ultimately end up in?

Furthermore, the standard is quite old, and solid, and proven. Isn't there some new technology that doesn't have all these drawbacks and isn't based on techniques, and demands, and ideas of 30 years ago?

*Answer:*

As you can see in the chapter "Comparison to alternative systems" there are some 'alternatives' to X Window. 'Alternative' is between brackets, because these systems discard the most important aspect of X Window, the network oriented aspect. They both promise improvement by leaving out the networking part. Clients and the server cannot be on different machines anymore. At least not as easy as X made it. As far as the network oriented graphics systems go, there is no alternative to X the same networking capabilities. Once an attempt was made, with Y Window system, but development on that stalled in 2005.

**Persons name: Lambert Veerman**

*Question:*

X calls the program running on the local machine (the one accepting user input) the X server. Intuitively you would think this would be the clients, while the remote machine running the heavy weight stuff should be called the server.

I quote Wikipedia: "The client-server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services."

Apparently they seem to have switched the terms, is there an architectural decision behind this naming?

E.g. because the most services run on the local machine, or because the local machine is running his services all the time (even when you have no connections setup).

*Answer:*

In answering this question, it is useful to recapitalize what was stated in the introduction section about X Window client-server model: the X server communicates with client programs which send requests for graphical output and accept pack user input from the server. The terms are indeed appeared reverse, as the user's terminal is regarded as the server, whilst the applications are playing the role of the clients. The end-user who actually is using the services through the terminal is simple a perspective that is not adopted by X Window's

architecture. Instead, application is the dominant perspective in the system, with a server which provides display and input/output services to applications, and the clients to be the applications which use these services. The client and server may run on the same machine through a local connection, or on different machines through a network connection.

**Persons name: Kevin van Blokland**

*Question:*

The X window system is a library that allows a screen to be sent over network/ipc channels. In some sense this creates an overhead in the way several resource identifiers are created and sometimes a lot of graphical data has to be sent over the network. Nowadays there are several other protocols that have the similar functionality e.g. Windows Remote Desktop Connections, TeamViewer and the VNC (virtual network computing) protocol.

In the new systems there are several techniques to overcome the burden of transmitting a lot of graphical data. For instance the VNC protocol has the ability to only send (small) portions of the screen, consisting only of the things that changed. This protocol also uses hardware acceleration to render screens and determine changed areas.

Does the X window system also have the ability to perform similar techniques, or is the architecture limited in such a way that this functionality can never be incorporated.

*Answer:*

The functionality of a partial refresh of the screen exists in X Window system from the very beginning of its implementation. There is a provision that inhibits the redisplay of the full screen, but again the primary perspective is the applications and not the end user. Every client, i.e. every application, is defined with a specific background for every window. Clients are mapping their windows' areas without affecting other contents of the screen or generating exposure events. The updated contents are copied back onto the screen and the windows' areas are unmaped. During this process, concurrent output from other clients to same regions is prevented by freezing their activities until the unmap request of the client who initiated the process.

**Persons name: Thijs Alkemade**

*Question:*

Even for a local window server, X uses a client/server architecture. Does this have enough bandwidth to handle applications that require a lot of full screen updates? (E.g. playing full HD video.)

*Answer:*

Despite the fact that such a functionality is out of the scope of the original implementation of X Windows, it is possible use applications that require extensive full screen updates, such as video playback in full screen, through an extension for this purpose. The *X video* extension is a video output mechanism for the X Window System and its main use is to enlarge a given video or to watch it in full screen mode by using the video controller hardware in each terminal. Without XVideo, X would have to perform the scaling on the main CPU, which would consequently cause a considerable delay in video stream. The speedup is very noticeable even on a fast CPU and most of the video playback programs that run under the X Window system have an option to enable XVideo output.

## Questions Group C

### Persons name: Tim de Boer

#### Question:

The suggested article is an old one (1986) and did not spend much effort in explaining the security of the system. But for systems like X Window, with remote connections, this should be of high interest and importance for them. I'm very curious how X Window have upgraded their security since then. Is the usage of for instance SSH further evolved?

#### Answer:

By default the usage of SSH is not part of X. However SSH does fully cooperate with the X Protocol and all communication between both the server and client can be encrypted. More information regarding this answer can be found in the chapter quality aspect 1 Security, subpart SSH.

### Persons name: Matthijs Neppelenbroek

#### Question:

Version X11R7.7 is planned for 2011. Do you think they can compete with architectures such as Google Apps/Docs. How does this affect their strategic advantage?

#### Answer:

The architecture of X has changed ever since the introduction of the first X11R7.\* release. Instead of releasing from one monolithic source tree, X11R7.\* releases are now build from many individual modules. These modules are distributed as individual source code releases, and each one is released when its ready, instead of only when the overall window system is ready for release. By doing this, X can also respond quicker to a changing market.

### Persons name: Rik Janssen

#### Question:

In the article there is a description of the major protocol changes for the X-system(10.Future). Which of the six changes will have the most impact on the architecture of the X-system? Can you pick two out of these six and describe what will change in the architecture but also which software quality aspects are influenced the most by these changes.

#### Answer:

For this question we will look at number 10.3 Color and number 10.4 Graphics. Color is of course very normal today and contributes a lot to the attractiveness of X applications. It also affects usability, because a user can easily distinguish different parts of the screen. The architecture probably didn't change that much to support color, but the X protocol did change to allow transmitting of more color.

Graphics are also very important to the usability of a X application. Supporting more fonts made it possible to port X applications to other systems. Before X relied on kernel fonts, which were not machine independent. This also contributed to the portability of X.

**Persons name: Matthias Lossek**

*Question:*

As you could see from the discussions after your presentations, one of the biggest questions is the difference between server/client on the same machine and on different machines.

In section 3 the papers says:

*"If the client and server are on the same machine, the stream is typically based on a local interprocess communication (IPC) mechanism; otherwise a network connection is established between the pair."*

But how exactly does that work? Is there a more detailed description for the difference? After your presentation I thought for client/server on the same machine it is just a TCP connection to localhost.

*Answer:*

As can be read from the chapter quality aspects, quality aspect 1, sub paragraph Interoperability the communication between the X Server and Client goes by using IPC when both reside on the localhost. This has major advantages over using TCP such as faster response times between the server and client.

## **Questions Group E**

**Persons name: Geert Wirken**

*Question:*

How extensible is the X Window System? In the presentations touch screens were already mentioned, but I think X also doesn't support 3D graphics. Does X support this? If so, how does this perform? And if not, can X be extended to do so or could one better use another system for that?

*Answer:*

Touch Screens are supported by the use of an external library called XStroke. 3D is supported in the latest version. The performance of this 3D isn't very good, but in the future this might be improved by external libraries.

**Persons name: Sander van der Rijst**

*Question:*

On the aspect of client interoperability, what do you think are the main causes why the X-window system has problems communicating between different X Window System clients and the X server?

*Answer:*

This answer is explained in the chapter chapter "Trade-off between Aspects" subchapter "Security vs Replacability and Interoperability".

**Persons name: Theodoros Polychniatis**

*Question:*

X Windows is designed mainly for network usage but the largest percentage of its actual usage is on desktop operating systems, meaning that both the client and the server are on the same machine. In desktop systems X is not very efficient and cannot take full advantage of the underlying hardware. On that perspective, do you know if there are plans from the developers' community to make it more

desktop friendly, and if yes, what are they? If there are no plans, are there any alternatives?

*Answer:*

As far as we know, there aren't any plans for a redesign of this problem. The alternative described in this paper in the chapter comparison with 2 other machines, wayland, uses a direct call to the hardware and can only be executed on one machine. This makes it more efficient than X if you want to run it on a desktop for example.

**Persons name: Ruben van Vliet**

*Question:*

I've got the following three questions for the reading second assignment. I'm not sure of more questions are allowed, if not choose the one that is most interesting.. (or easy to answer)

- Is there any support for more advanced multimedia, like e.g. audio, video, 3d graphics?
- Is it possible that both server and client are waiting for input? If so, how does X windows prevents a deadlock?
- The document mentioned that performance can be optimized by minimizing the number of operating calls. Can you explain why this isn't done in the first place and what the trade-off will be when minimizing them?

*Answer:*

Audio isn't supported in X, video and 3D are supported in the latest versions. For more information see the question of Geert Wirken, Group E or this paper.

## **Questions Group F**

**Persons name: Leupolz, J.S.**

*Question:*

The Paper describes a situation in 1986. Are in modern X-based Desktop-Systems like Ubuntu still many of the primitive operations of the X-protocol, like an explicit operation for drawing a line, in use or are they just implemented in the X-Server and unused by a majority of X-Client-Applications and thus obsolete?

Today memory is cheap. Does the X-Server today still forward the "redraw"-messages? What happens when 3D-Window Managers like Beryl are in use?

How is Multi-Touch(see iPhone or Microsoft Surface) supported?

*Answer:*

Regarding the first question: yes the primitive operation for drawing a line is still available.

Regarding the second question: redraw messages are still used to notify the client of an user event, for example: The client must than redraw the screen. Most modern X applications only redraw the affected screen area and not the whole screen.

Regarding the third question: this question is asked by "Kevin van Ingen". So please refer to the answer at his question below.

**Persons name: Maria Hernandez**

*Question:*

The paper says-section 10, future- that they were planning to improve those aspects which were not as good as the want. Could you, briefly, say how is the current situation? which of them have they done them, and which are still under development? Especially, what about the 3-D graphics?

*Answer:*

Resource allocation: In X11 most of the resource allocation was moved from the X Server to the X Client.

Transparent Windows: Also transparent windows are supported by external libraries such as Xseethru.

Color: Color was added to X in the 9<sup>th</sup> version in 1985.

Graphics: 3D is currently fully supported.

Management: The latest window manager can handle all the problems of the version in X1.

Extensibility: With the latest version and all the available extensions, this part of development did succeed!

**Persons name: Stijn van Drongelen**

*Question:*

In what way does the X Window System support "new" types of devices (e.g. touch screens) within the existing kinds (e.g. input devices)?

Does an application need to know about these types to use them? To give an example for the latter question: who needs to determine that clicking with the mouse and tapping on a touch screen is pretty much the same thing?

*Answer:*

There's a special library called Xstroke that handles gestures on touch screen devices with X Window Systems. I can cope with all the usual touch screen activities like dragging, tapping, tap-and-hold etc.

For more information, visit:

[http://www.usenix.org/events/usenix03/tech/freenix03/full\\_papers/worth/worth.html/xstroke.html](http://www.usenix.org/events/usenix03/tech/freenix03/full_papers/worth/worth.html/xstroke.html)

**Persons name: Alejandro Serrano**

*Question:*

- How are 3D graphics drawn in the X Window? When you implement a program, it seems that you talk directly to OpenGL or the graphic card. Does X also take care of that or it's sort of a trick to compose things?

- It would be nice to have a diagram showing where things like GTK+ and Pango fit in the X tack. For example, how is TrueType fonts handled in X?

*Answer:*

The latest version of X has built in support for TrueType fonts. For a complete manual on how to use those fonts, visit:

<http://www.freebsd.org/doc/handbook/x-fonts.html#TRUETYPE/>

Shortly, the latest version can't render TrueType fonts out of the box. It needs preinstalled fonts, that can be shown on any device after installation.

## Questions Group G

**Persons name: Jacek Marek**

*Question:*

In the paper is mentioned that much of the design and implementation was handled solely by the first author. Therefore this author made a lot of first decisions. Do you know about some of his decisions that affected the future development negatively? E.g. something that happened to be a bad idea.

*Answer:*

There aren't any real development failures in the history of X. The only problems that occur during development were the problems related to compatibility issues with newly added platforms. For example, the release for version 10 was due to the fact that version 9 was incompatible with the IBM RT/PC platform.

**Persons name: Vladimir Smatanik**

*Question:*

Are there in the newest versions of X any real solutions for race conditions, not only work-arounds mentioned in the paper ?

*Answer:*

It is hard to find an answer if in the newest version there really is a suitable solution solving the work-arounds mentioned in the paper. However X does allow the usage of extensions of which a Network extensible Window System (NeWS) was created by Sun to make the user interface toolkit become an extensible server library of classes that clients download directly into the server. With NeWS, the window manager itself was implemented inside the server, eliminating network overhead for window manipulation operations and along with it race conditions.

**Persons name: Richard Derer**

*Question:*

The X is designed to allow extensions libraries. Were there some really significant extensions that were accepted by the X community and adopted as a part of the core ?

*Answer:*

Yes, there were several extensions accepted by the X community however they were not by default adapted in the core. As X was designed to be made as simple as possible while still being extensible the accepted extensions were not made part of the core. Very popular widely accepted extensions are for example; XFixes (provides several protocol changes), GLX (support for rendering OpenGL within Windows) and X video extension (support for hardware video overlays and hardware-based video scaling on playback).

**Persons name: Alexandru Dimitriu**

*Question:*

In the Resources part is written :

"Any client that knows (or guesses) the identifier for a resource can use and manipulate the resource freely, even if it was created by another client."

A few lines under is introduced the idea when the client forgets to clean up their resources these are destroyed automatically when client terminates. What happens to the other client that may use these resources ?

*Answer:*

No, if the resources are destroyed, they can no longer be used by other clients

## References

- Fisher, J. (2002). *Securing X Windows*. Network Security Library, 1-8
- Gross, B., Buehler, B. (2002). *Improving X-Windows security*. Network Security Library, 1-7
- Garfinkel, S., Weise, D., Strassman, S., (1994) *Unix-Haters handbook* IDG Books Worldwide
- Guilen, A., (2001), *How To Protect Your Display*. X Windows Security, November 16th
- Lee, K., (1993). *Debugging X Window System Protocol Errors*. The X Journal, March
- Lee, K., (1995). *X Application Software Engineering: Debugging X Input Events*. The X Journal, January
- Losavio, F., Chirinos, L., Matteo, A., Levy, N., Ramdane-Cherif, A. (2004). *ISO quality standards for measuring architectures*. The Journal of Systems and Software, 72, (2), 203-223
- Rosenthal, D., Marks, S., (1993) *Inter-Client Communication Conventions Manual: Version 2.0*. MIT X Consortium 1-61
- Sachs, J., G., (2001) *Secure X Windows with SSH*. News on the Net; The A3C Connection
- Scheifler, R., W., Gettys, J., (1986). *The X Window System*. ACM Transactions on Graphics, 5, (2), 79-109
- Wiggins, D., P., (1996). *Security Extension Specification*. X Consortium, 7 (1), 1-18

*Other not officially used sources found on the internet:*

<http://earthsci.stanford.edu/computing/unix/xterminal/xauthentication.php>  
<http://www.markshuttleworth.com/archives/551>  
<http://www.faqs.org/docs/Linux-HOWTO/XWindow-Overview-HOWTO.html>  
[http://en.wikipedia.org/wiki/X\\_Window\\_System\\_core\\_protocol](http://en.wikipedia.org/wiki/X_Window_System_core_protocol)  
[http://en.wikipedia.org/wiki/X\\_Window\\_System](http://en.wikipedia.org/wiki/X_Window_System)  
[http://en.wikipedia.org/wiki/X\\_Window\\_System\\_protocols\\_and\\_architecture](http://en.wikipedia.org/wiki/X_Window_System_protocols_and_architecture)  
[http://www.museumstuff.com/learn/topics/X\\_Window\\_System\\_core\\_protocol](http://www.museumstuff.com/learn/topics/X_Window_System_core_protocol)  
[http://www.sbin.org/doc/Xlib/chapt\\_01.html](http://www.sbin.org/doc/Xlib/chapt_01.html)  
[http://www.wordiq.com/definition/X\\_Window\\_System\\_protocols\\_and\\_architecture](http://www.wordiq.com/definition/X_Window_System_protocols_and_architecture)  
<http://www.linuxdocs.org/HOWTOs/XWindow-User-HOWTO-2.html>  
<http://wayland.freedesktop.org/architecture.html>  
<http://www.microxwin.com/performance.html>  
[http://www.usenix.org/events/usenix03/tech/freenix03/full\\_papers/worth/worth.html/xstroke.html](http://www.usenix.org/events/usenix03/tech/freenix03/full_papers/worth/worth.html/xstroke.html)  
<http://www.freebsd.org/doc/handbook/x-fonts.html#TRUETYPE/>  
<http://www.tigr.net/afterstep/X/xlib/GC/manipulating.html>  
[http://www.comptechdoc.org/os/linux/howlinuxworks/linux\\_hlxwindows.html](http://www.comptechdoc.org/os/linux/howlinuxworks/linux_hlxwindows.html)  
[http://stein.cshl.org/genome\\_informatics/unix1/xwindows.html](http://stein.cshl.org/genome_informatics/unix1/xwindows.html)