A. A Development Framework for Data Sequencers

In this chapter a development framework for data sequencers will be presented. This development framework supports application development for the fixed data sequencer as described in appendix D.2 at page 263. Furthermore it provides an IP-generator for application specific data sequencer implementations as explained in section 6.5 at page 121.

The main goal during development framework implementation has been to reduce the design time for high performance memory communication applications. The problems of developers to understand and get into the novel data sequencing concept has been identified as a key problem. In [BHH98] this problem has been named the Education Gap. The development framework tries to solve this problem through the integration of multimedia features by an Internet[1]-based implementation.

A.1 The Internet-based Implementation

The increasing interest in the Internet of the recent years results in a growing number of users as well as improved network infrastructure, quality and performance. This is the basis for a new kind of software, which is running via the Internet [Den97] [Gep98]. That means the software is installed on a server connected to the Internet and accessed and executed by a remote client. Such Internet-based software provides several benefits. Once distributed by a server, Internet-based software is available world-wide and depending on its implementation it can be executed without previous installation on the client machine. Further software updates and patches are not distributed to the users any more. Only the server has to be updated and the new version is available to all users immediately. For commercial use it is not necessary that customers buy the software any more, they download and pay only the modules they need. This pay by use method makes very expensive design software also available to small companies. It is also possible to place a high-performance or application specific computer at the server side and provide computation time for special tasks to the users.

In the area of reconfigurable computing the described techniques allow to provide public access to accelerator prototype hardware and related development software. Since experimental prototypes mostly consist of especially designed components, which are also very expensive, Internet-based access increases the number of users efficiently. One of the first approaches, where remote testing was implemented, is

1. For information on the Internet refer to [Tho95].

WebScope [Guc98] [LG98]. A different approach, which is more going in the direction of operating systems, has been published in [Bre96] [Bre98]. Here reconfigurable hardware is regarded more abstract and its computational resources can be used in a client/server [And91] based approach.

In this section the concepts of a remote prototyping framework will be presented. For remote testing of applications a prototype hardware may be accessed. The software part of the framework forms the Xputer Multimedia Development System (XMDS, [HHN99b] [HHN99a] [HHH99b] [Sch98]). It incorporates the advantages of Internet-based software mentioned above. Further the XMDS enables access to the Map-oriented Machine with Parallel Data Access (MoM-PDA, see appendix D or [HHH98c]) for remote testing.

The Principles of the Xputer Multimedia Development System

The main part of the XMDS [XMDS] system is the XMDS server which holds the complete XMDS system including an user data library, a runtime system for the Xputer hardware and the MoM-PDA prototype hardware (figure A-1). Parts of the XMDS system are executed on the server and other parts are downloaded on the fly to the users computer and executed there. For communication between these computers a special protocol (XMDS messaging) has been developed. Further an Unix [Gul88] host may be connected to the XMDS server holding conventional design software written in C programming language [KR88].

Several concepts had been focussed during the XMDS implementation. In fact, the XMDS is a project that combines the features of a powerful Internet-based CAD system with the specific requirements to access the experimental Xputer hardware and the traditional (C-programmed) software. This led to the following specification items. The XMDS should provide:

- a user front-end that is accessible via WWW for several platforms,
- a central user administration,

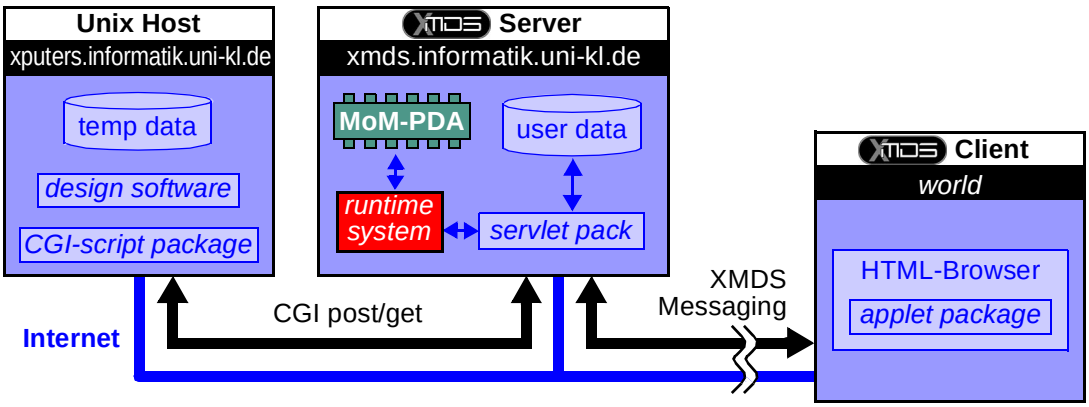

Figure A-1: Overview of the XMDS.

- a modularity of its components,
- a built-in support for further extensions,
- powerful network capabilities, and
- an embedding of multimedia features.

In the following, the realization of these concepts is described in detail.

The WWW front-end and the Client / Server Model

Because the WWW[1] is commonly explored with a WWW browser, the user front-end of the XMDS must have a WWW address, like normal WWW documents. By pointing his browser to that particular URL[2], an user of the XMDS may download the front-end to his client machine and display it on the screen. Therefore, the part of the XMDS that should display on the users machine must be executable by the most common WWW browsers. Therefore a Java (see [Fla98], [LP97], or [Rit95]) implementation has been chosen. The fact that a WWW browser forms the environment of the front-end ensures a certain independence against the underlaying operation system (e.g. Windows, MacOS, Unix [Gul88]). Portations of a particular WWW browser to different platforms ensure therefore the availability of the XMDS.

The different components that form the XMDS are realized following the client/server paradigm. Components are distributed in a logical manner on the client and on the server part. Every user front-end which is downloaded by an user forms a client of the XMDS. The user may reach it by requesting a specific URL targeting the computer where the XMDS files are installed. To allow the publishing of URLs, this computer must run a WWW server. The HTML[3] document loaded in the users browser window contains a reference to the XMDS client program.

The complete XMDS system is programmed in Java. While the Java applet technology realizes a dynamic download of the XMDS client to an users machine via Internet, the XMDS server makes use of the Java servlet technology to accept the connections from the client and perform the server-side actions.

The advantages of such a dynamic client/server model compared to a static installation of client software are obvious. With this concept, the XMDS

- is available world-wide,
- may be used without previous installation,
- is always available in its latest version,
- downloads only the components that are actually needed by the user, and

1. WWW = world wide web, see [Lem96], or [Ram95]
2. URL = uniform resource location
3. HTML = hypertext markup language, see [Lem96], or [Ram95]

- enables the users to evaluate the MoM-PDA without acquiring the hardware.

This concept realizes an easy way for distribution and provides easy access for interested persons.

The Central User Administration

The XMDS server also contains the central user administration: the user database and the user management system.

The users of the XMDS need to store the data, the state and the configuration of their XMDS sessions in order to continue their work in a next session. Because the XMDS-Client has no permission to store data on the clients machine due to the applet limitations, all the user-specific data must be stored on the XMDS server. Therefore the users of the XMDS must be registered. To secure intellectual property (IP) all users receive a personal password to protect their data.

The Network Capabilities

A very close binding between the client and the server is essential for a powerful distributed system like the XMDS. This is realized by the XMDS Messaging System which will not be discussed here (for details see [Sch98]).

An important task involving a network communication is to invoke the design software contained on an Unix host. Because this host runs a Web server, a medium-level communication is possible involving the Common Gateway Interface (CGI, [Ram95] [SB96a]) of the Unix host. Because Java applets (like the XMDS client) are not allowed to communicate with an Internet host different from their own home host, it must be the XMDS server that establishes this communication. Therefore the XMDS server provides a module for a comfortable use of the CGI-request-response interaction and extends the communication to the XMDS client using the XMDS Messaging System.

The Multimedia User Interface

The XMDS is prepared to cooperate with different media types to support the users in their application development process. It integrates complex animations within the user interface of the XMDS to allow visualization of non-trivial processes which efficiently supports application development.

On a more basic level, a support is integrated for playing audio files. This is useful for standard events like system sounds, or for requesting the users attention in special situations. By the ability to add explaining speeches to particular key situations, the user is supported actively on his way through the different parts of the development process.

Further the XMDS client window is able to load HTML [Lem96] [Ram95] documents in the browser window. This allows the construction of a context-sensual help system integrating text, image and hyperlink elements. The on-line-help is in fact expanded to a complete on-line-tutorial providing the most actual help documents available at the XMDS Server.

A.2 The Xputer Multimedia Development System Tools

In this subsection the XMDS toolset will be presented. The XMDS concepts have been developed to be a general development system for all hardware parts. Up to now only data sequencer related tools have been implemented. These data sequencer related tools support application development according to the data sequencing principles presented in this thesis.

The XMDS Design Flow

After login the XMDS starts an application chooser (see figure A-2 at page 230) which visualizes the XMDS design flow. This design flow for data sequencer application development is partitioned in three steps:

- the design input phase,
- the design validation phase, and
- the design output phase.

Each step in the chooser window represents a dedicated tool which may be invoked by selecting it. In the rest of this chapter these tools will be introduced, but first the three design phases are described.

During the design input phase the user enters all information needed by the data sequencer to execute an application. For this task three different tools are available, which differ in performance results and degree of difficulty, i.e. they are dedicated for differently experienced users. The three tools are:

- the hardware level task editor,
- the graphical task designer, and
- the high level language interface.

The hardware level task editor and the task designer allow only the development of single tasks.

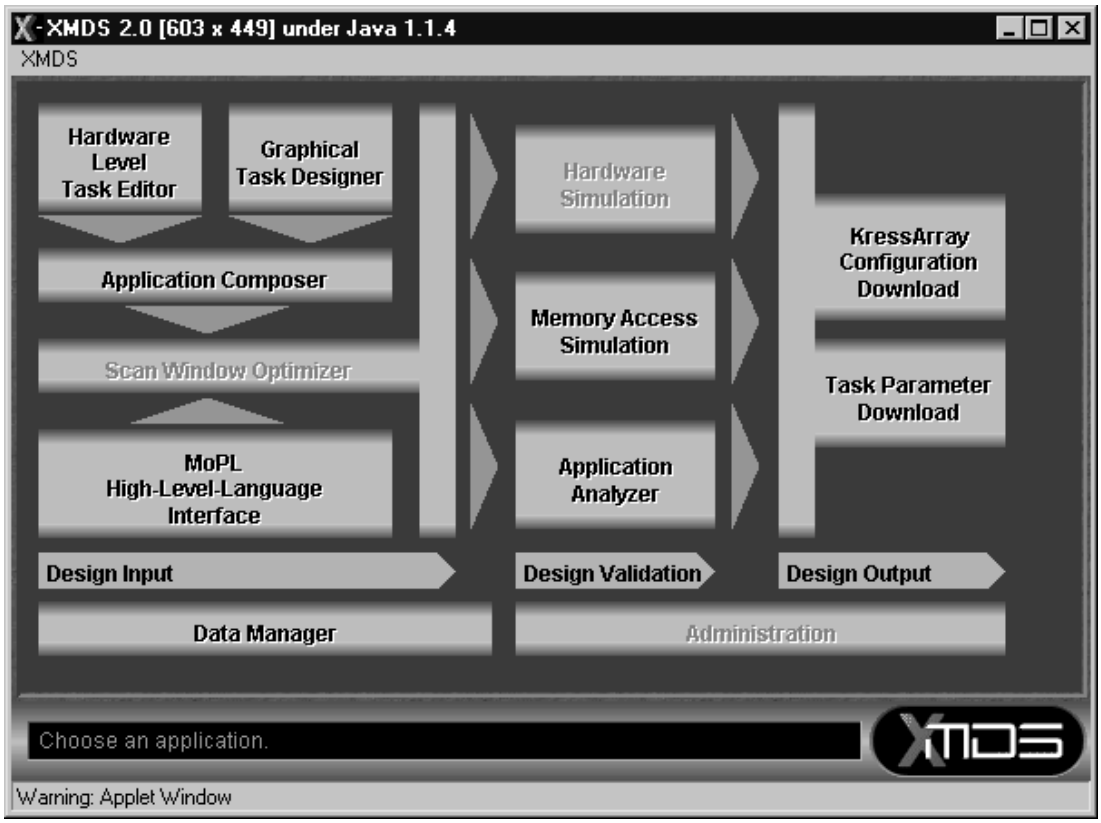

Figure A-2: The XMDS Chooser window.

Definition A-1: Task

A task is a definition of the data set, i.e. required amount of data memory, a complex scan, and a scan window.

Due to the internal organization of the XMDS a task consists of a maximum of 16 video scans which may be arbitrarily connected to a complex scan. The fixed data sequencer implementation presented in appendi xD.2 at page263 is limited to a maximum of 64 video scans (see table D-3 at page 286).

To implement an application several tasks may be needed. For this the application composer (see sectio nA.2 at page229) is utilized.

Definition A-2: Application

An application is a set of one or multiple tasks needed to compute a specific problem.

Unexperienced users may prefer automatic compilation of applications. For this the CoDe-X (Co-Design for Xputers, [HBH96b] [Bec97]) framework has been implemented. Because CoDe-X is programmed in C language [KR88], it may be installed on an Unix[1] host and will be made accessible via the XMDS. Currently it is not integrated into the XMDS.

The design validation is performed with two tools:

- the memory access simulator, and
- the application analyzer.

These tools visualize the programmed memory accesses and calculate the exact runtime of the programed application. If the design requirements are not met, the user may return to the design input phase. Otherwise the user will proceed to phase three.

The design output phase offers two alternatives:

- the KressArray configuration download, and
- the task parameter download.

Both tools save design data on the client computer. The KressArray configuration download first generates an application specific data sequencer as explained in section 6.5 at page 121. Then the input file for the MA-DPSS (see [HHH99a] or [HHH00]) is sent to the users computer. The task parameter download sends the programming data for the fixed data sequencer presented in appen d ixD.2 at pa ge263 to the users computer. These parameters will also be needed for the KressArray implementation. Since the fixed and the KressArray implementation are based on the same concepts, they also need the same programming data.

The Task Editor

The task editor allows the manipulation of the XMDS internal task data (for information on this topic refer to [Sch98]). This tool should only be used by well experienced designers. The task designer has no comfort and does not support the developer. It simply displays the XMDS internal task data.

The Task Designer

The task designer implementation is described detailed in [Bra99]. It is a graphical tool which allows the developer to enter a task in three steps. For each step the task designer provides a graphical window.

1. For details on UNIX refer to [Gul88].

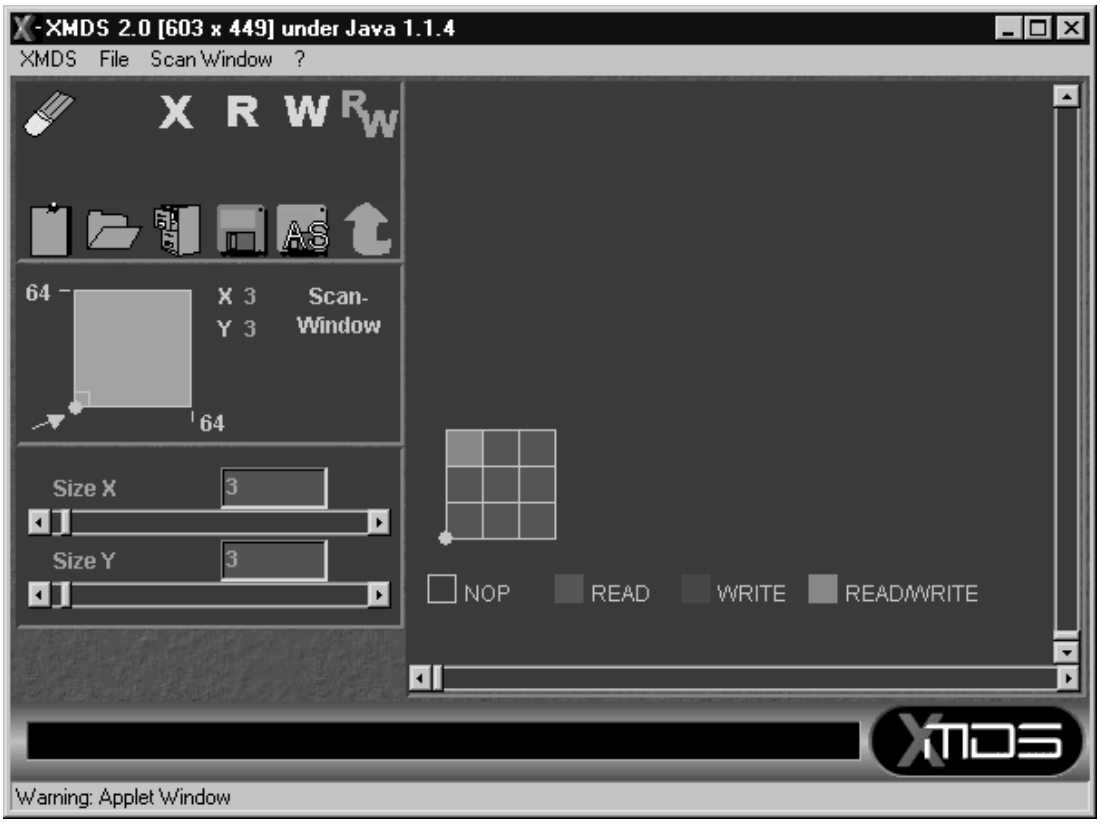

Figure A-3: The XMDS Task Designer window for scan window design.

Figure A-3 shows the task designer user interface for graphical scan window input. With the scrollbars on the left side the developer adjusts the overall size of the scan window. The XMDS internally limits the scan window size with 64 positions for each dimension.

On the right side of the graphical scan window input window the actual scan window is shown. The developer determines the performed data access for each position by simply selecting of the possible actions (no operation[1], read, write, and read and write) and clicking on the required scan window position.

1. NOP = no operation, see figure A-3

Figure A-4: The XMDS Task Designer window for video scan design.

Figure A-4 illustrates the task designer user interface for graphical video scan input. On the left side an overview on the current video scan is shown. Further the video scans of the current task are managed. The area on the right side allows to enter the data describing the video scan.

Figure A-5 at page 234 pictures the task designer user interface to graphically compose complex scans. In the center of the window a tree illustrates the complex scan. Previously entered video scans are shown as leaves of the tree. These video scans are combined by compound-, nested-, and meshed operators.

The Application Composer

The application composer is used to define applications from multiple tasks. The application composer user interface is depicted in figure A-6 at page 235. It is shown how a matrix multiplication application is composed from three tasks, which access the two source and the result matrix. Since each task is already saved, an application is saved only as pointers to the tasks. A developer should keep in mind, that if a task which is contained in an application is changed, also the application is changed.

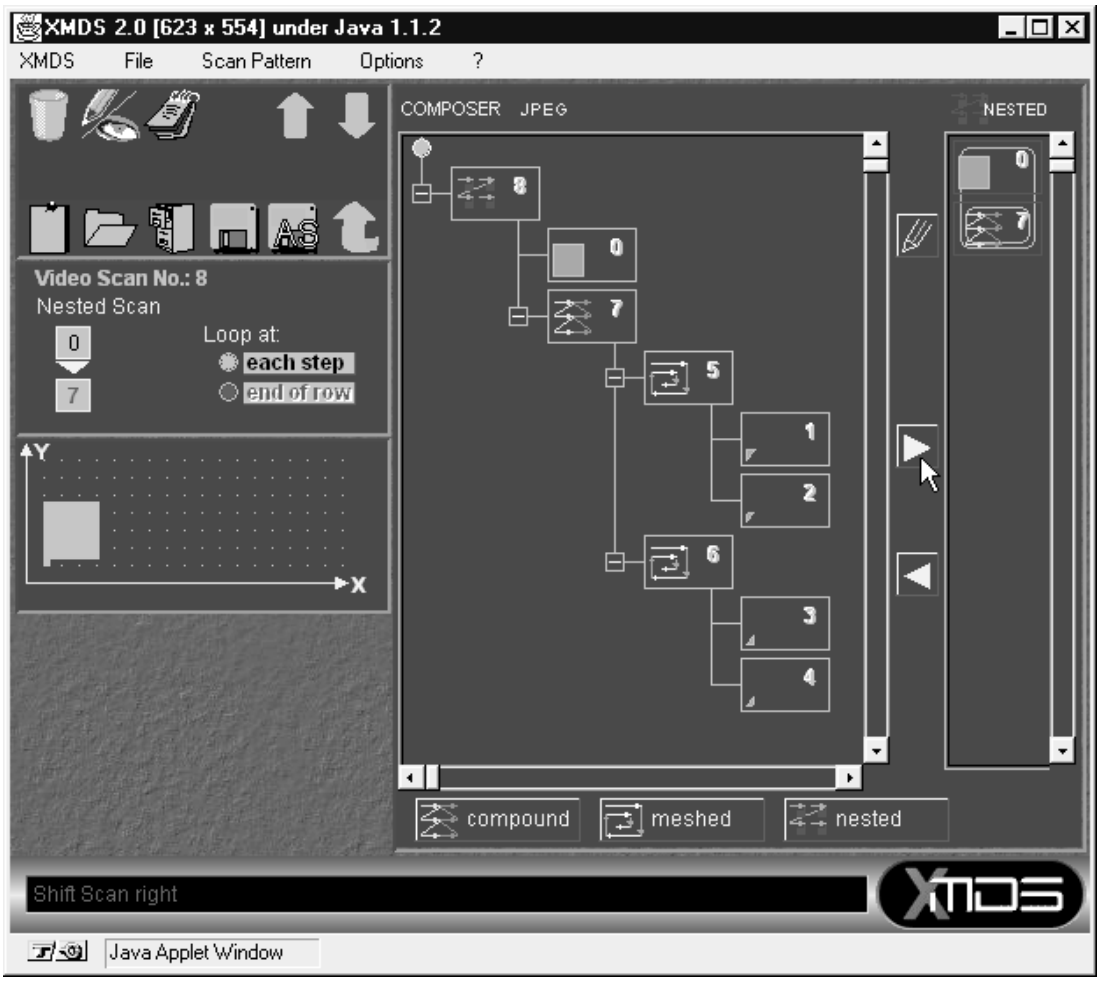

Figure A-5: The XMDS Task Designer window for complex scan design.

More details on the application composer may be found in [Wer99]. The complete matrix multiplication implementation is presented in [Buc99b].

The High Level Language Interface

The high level language interface window is pictured in figure A-7 at page 236. It mainly consists of an simple text editor for program input. Further the high level language input supports file up- and download between the XMDS server and the client. This enables the utilization of stand-alone text editors.

The high level language interface implementation is extensively described in [Bau00]. For data sequencer application development a MoPL compiler is used on the XMDS server side. For information on MoPL refer to [ABH93]. The MoPL compiler implementation is described in [Hen97].

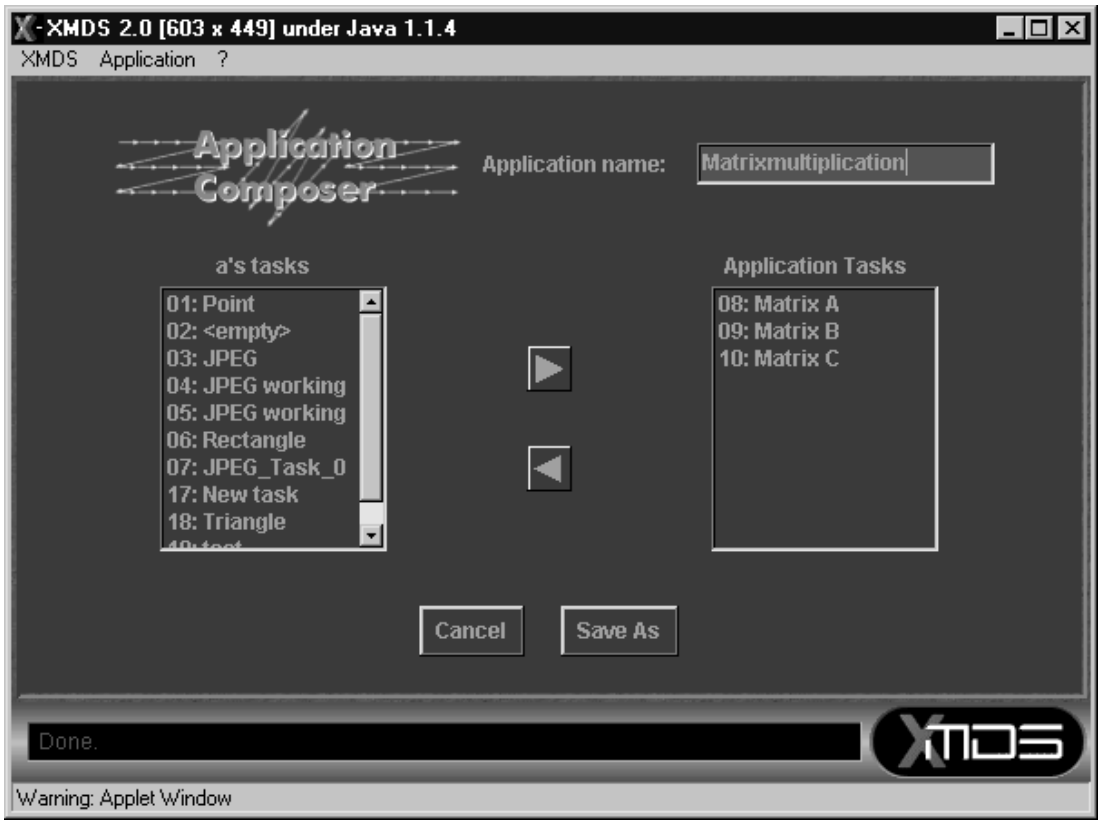

Figure A-6: The XMDS Application Composer window.

The Application Analyzer

The application analyzer window is shown in figure A-8 at page 237. Its implementation is described in [Wer99].

The application analyzer enables the developer to determine the exact runtime of an application. Therefore the application analyzer uses an exact hardware model to count the clock cycles the data sequencer hardware needs for the given application.

To calculate the execution time for different data sequencer implementations, hardware parameters may be changed. Also the hardware parameters for several data sequencers can be loaded from a library.

The application analyzer generates multiple reports, which are displayed in the Web browser window[1]. First a summary of the analysis is viewed. From this summary the developer may browse to all specific information on the execution of the application.

1. Informations on Web browsers may be found in [Lem96], or [Ram95].

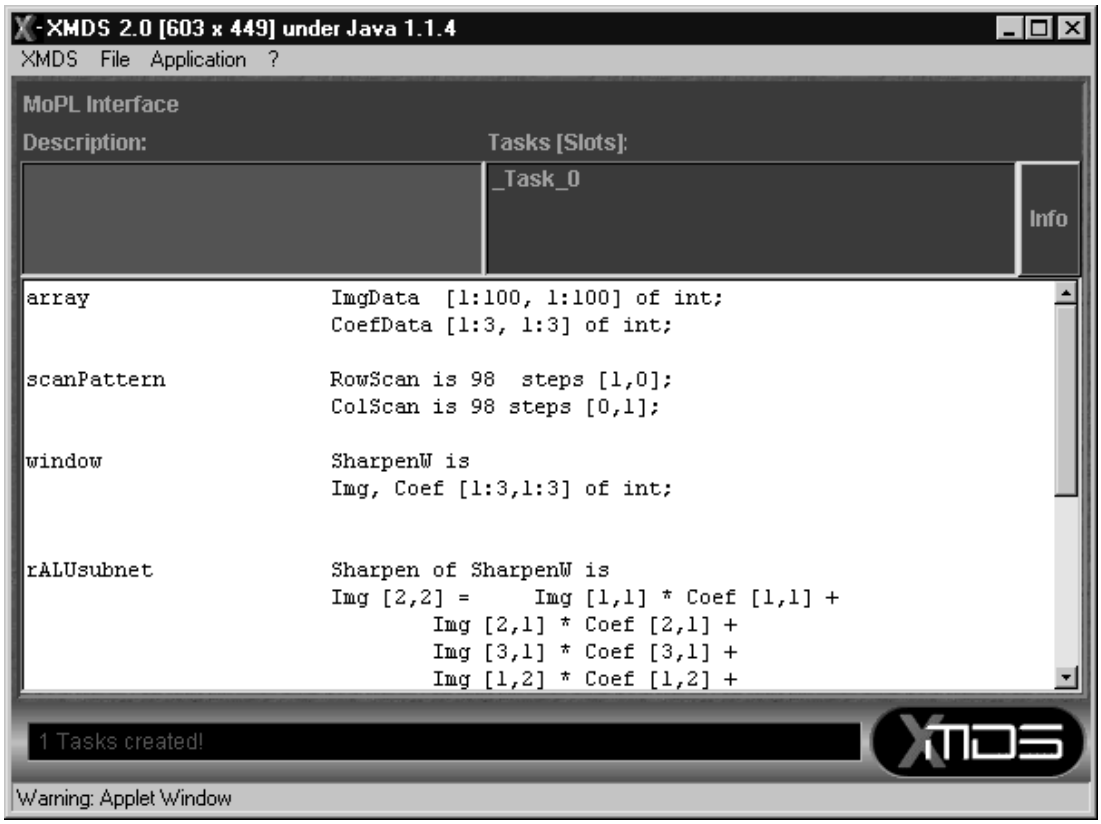

Figure A-7: The XMDS High Level Language Interface window.

The Memory Access Simulator

The memory access simulator [Kla00] visualizes the sequence of performed memory accesses (see figure A-9 at page 238). The sequence may be displayed as a static picture of all accesses performed by an application or may be viewed as an animation starting by the first access.

The KressArray Configuration Generator

The KressArray configuration generator produces application specific data sequencers based on the principles introduced in section 6.5 at page 121. It takes as input an application task and determines all contained video scans. The optimization may be performed for each single video scan or for the complete task. The KressArray configuration generator offers the following alternatives for optimization output:

- an explanation of the performed optimizations,
- the layout of the optimized data sequencer,

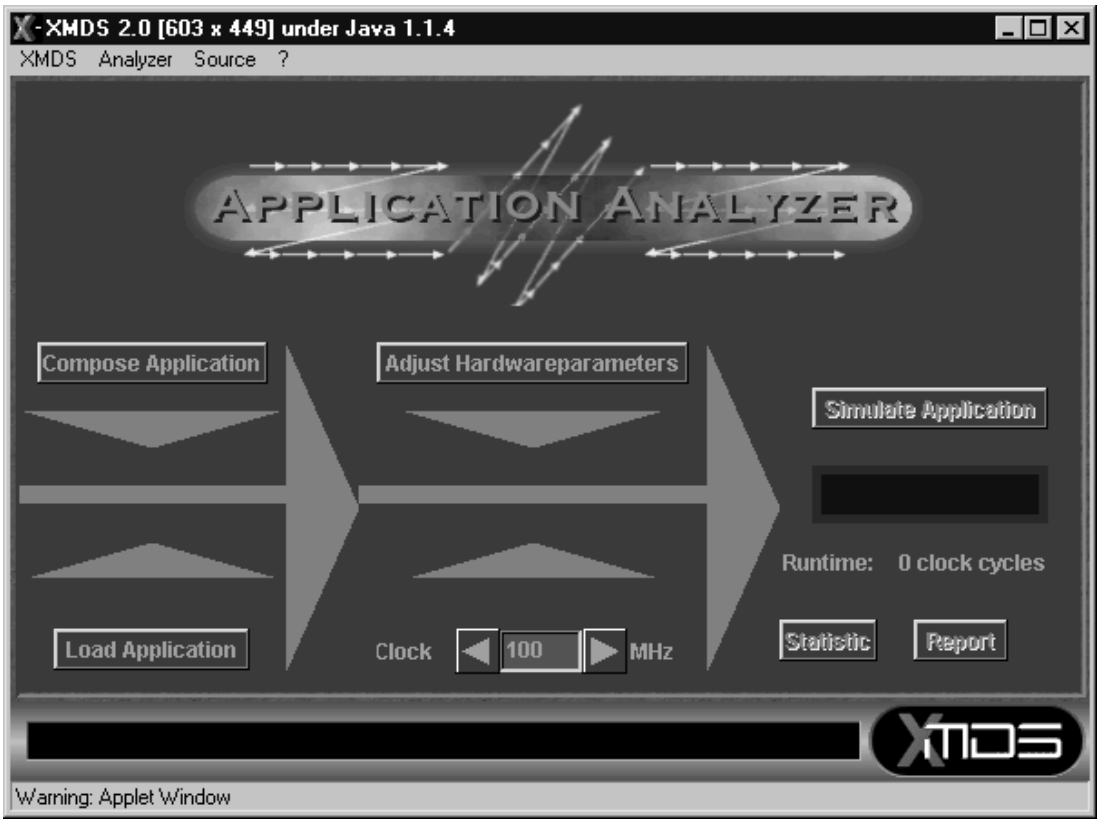

Figure A-8: The XMDS Application Analyzer window.

- the KressArray mapping file, which is used as input by the MA-DPSS[1] for the mapping to the KressArray,
- a picture of an example mapping with the MA-DPSS, and
- a movie of the mapping procedure, performed by the MA-DPSS for the example mapping.

The user interface of the KressArray configuration generator is pictured in figure A-10 at page 239. More information about the generator may be found in [Ero99].

The Data Manager

The XMDS uses four different data types:

- task parameters,
- applications,

1. For information on the MA-DPSS refer to [HHH99a] or [HHH00].

Figure A-9: The XMDS Memory Access Simulator window.

- high level language descriptions, and
- hardware parameters used by the application analyzer.

All users have an own password protected area to store data. Furthermore a public library is available, which is maintained by the administrator. This library contains exemplary designs, which might be interesting for new users.

The data manager allows the user to copy from public to private objects and to copy, delete, or rename private objects. Further the data manager supports a quick preview for all data types, or to invoke the related application. More information on the data manager can be found in [Koe00].

A.3 Chapter Summary

This chapter has introduced a development framework for data sequencers. It efficiently supports application development for data sequencers based on the concepts presented in this thesis. The development framework provides several tools for entering applications. The programming parameters may be used to program the fixed

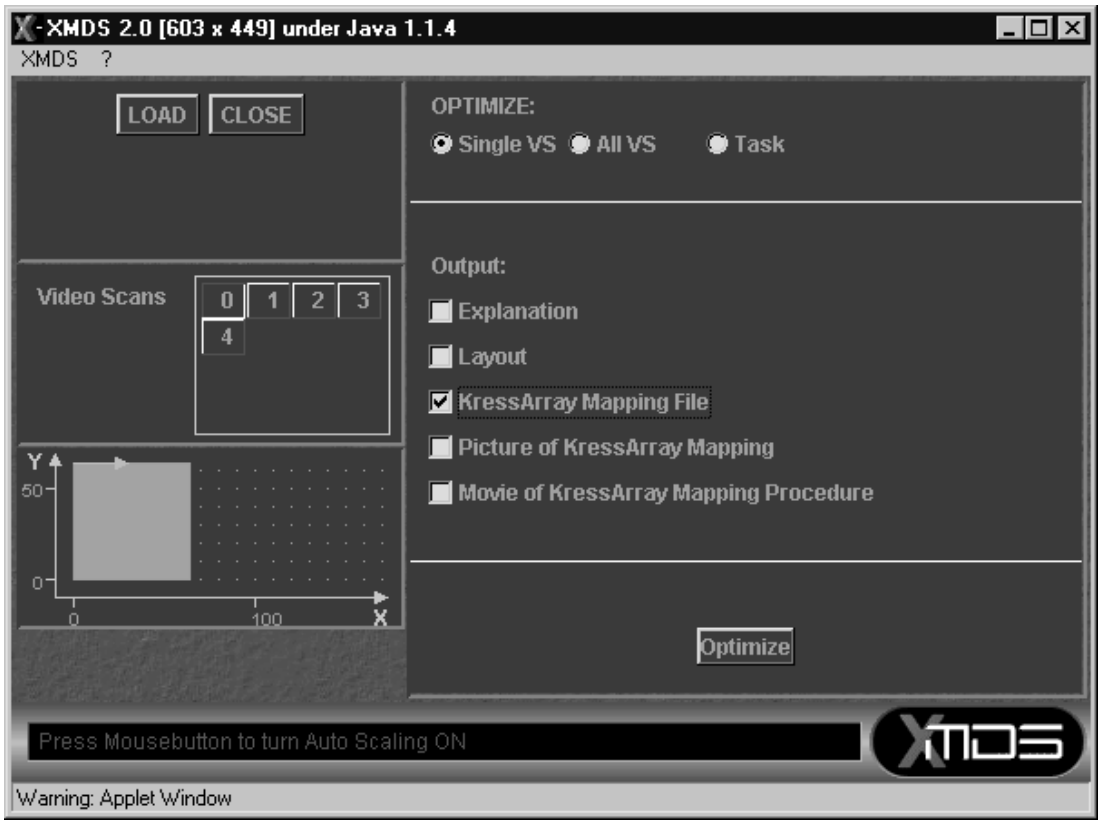

Figure A-10: The XMDS KressArray Configuration Generator window.

data sequencer presented in appendix D.2 at page 263 as well as application specific data sequencer (see section 6.5 at page 121). The development framework also includes a generator for such application specific data sequencers.

The Internet-based implementation required to work with new programming paradigms (e.g. client-/server-based implementation) and to develop new communication [Sch98] and synchronization [Bau00] methods. But developer benefit form this novel type of software:

- no installation of the software is required,
- inclusion of multimedia features eases application development, and
- the utilization of HTML [Lem96] [Ram95] documents and the WWW for data output and on-line help also supports the direct integration of knowledge databases to support developers.

Java (see [Fla98], [LP97], or [Rit95]) has proven as a very good selection for implementation. During an intense occupation with the user interface design of the XMDSClient, several inconveniences have occurred. They are mainly due to the defectiveness of the Java Abstract Window Toolkit (AWT, see [Fla98]). The platform independence - a major advantage of the Java language - is not yet guaranteed if

graphical elements get involved. Maybe the adoption of an unique "look and feel" for the Graphical User Interface (GUI) components on every platform will avoid these problems in future releases of the Java API.

In contrast, the "non-visible" parts of the Java API show an impressive reliability. The Java network package makes it easy to develop a comprehensive distributed application like the XMDS. In addition, the perfect complement of applets and servlets reveals Java as a serious alternative to "conventional" languages as C/C++ [KR88] for network based low and medium scale applications. Due to the URL class and the possibilities to communicate with the Web browser[1], a Java applet can make use of the hypertext paradigm of the World Wide Web to integrate information and multimedia items, as realized for the XMDS help system.

Unfortunately, the main Web browser vendors are always in some delay to the actual Java API standard. As a consequence, the developers of a project like the XMDS have to consider that new API features aren’t immediately embedded in the newest browsers and that not all the users take advantage of the newest browser. Further the programmers of the Web browsers have sometimes different interpretations of the Java standard.

All this leads to the conclusion that although Java is specified as platform independent a lot of software validation an different platforms has to be performed before Java programs really work. But the efforts are worth of because of the benefits of the Java implementation. Real Internet-based software may be programmed much easier compared to other programming languages because of the sophisticated network support.

1. Informations on Web browsers may be found in [Lem96], or [Ram95].