Int. J. Communications, Network and System Sciences, **2013, 6, 221-235**
http://dx.doi.org/10.4236/ijcns.2013.65025 Published Online May 2013 (http://www.scirp.org/journal/ijcns)

Online Data Processing Methods in Xlet Applications for Seismic Early Warning and Emergency Services on Interactive Digital Television

Mustafa Asim Kazancigil
Doctoral School of Informatics, Università degli Studi di Milano, Milan, Italy
Email: mustafa.kazancigil@unimi.it

Received February 22, 2013; revised March 14, 2013; accepted April 14, 2013

ABSTRACT

In this paper, I described the methods that I used for the creation of Xlets, which are Java applets that are developed for the IDTV environment; and the methods for online data retrieval and processing that I utilized in these Xlets. The themes that I chose for the Xlets of the IDTV applications are *Earthquake and Tsunami Early Warning*; *Recent Seismic Activity Report*; and *Emergency Services*. The online data regarding the *Recent Seismic Activity Report* application are provided by the Kandilli Observatory and Earthquake Research Institute (KOERI) of Boğaziçi University in Istanbul; while the online data for the *Earthquake and Tsunami Early Warning* and the *Emergency Services* applications are provided by the Godoro website which I used for storing (and retrieving by the Xlets) the earthquake and tsunami early warning simulation data, and the DVB network subscriber data (such as name and address information) for utilizing in the *Emergency Services* (Police, Ambulance and Fire Department) application. I have focused on the methodologies to use digital television as an efficient medium to convey timely and useful information regarding seismic warning data to the public, which forms the main research topic of this paper.

Keywords: Xlet; Java TV; Interactive Digital Television; DVB; Data Processing; Seismic Early Warning

1. Introduction

In this paper, I have focused on the methodologies for using digital television as an efficient medium to convey timely and useful information regarding seismic warning data to the public through Xlet-based applications. The secondary goals include providing easy interactivity for television viewers in the IDTV environment through a simple user interface based on remote control buttons, while ensuring quick, flexible and simultaneous production, transmission and reception of the video broadcasts, emergency datacasts, and application services. The main theme that I chose for my Xlet applications is earthquake and tsunami early warning on digital television.

In the aftermath of the devastating Izmit earthquake of 17 August 1999, Kandilli Observatory and Earthquake Research Institute (KOERI) of Boğaziçi University in Istanbul was assigned with the task of establishing the Istanbul Earthquake Early Warning and Emergency Rapid Response System [1]. The construction of the system is realized by the GeoSIG and EWE consortium [1]. As of early 2013, communications (related only to the Rapid Response System) are provided by the Turkish GSM service provider Turkcell [1-4]. UDIM (National Earthquake Monitoring Center) bound to KOERI sends SMS text messages, e-mails, electronical fax, radio text signals and Twitter messages to the staff of KOERI; to the Turkish government (Prime Ministry, AFAD (Disaster and Emergency Administration Presidency), Governorate of Istanbul, etc); and to the AKUT search and rescue team, once initial estimation results regarding the epicenter and magnitude of an earthquake arrive [1-4]. As of early 2013, SMS messages regarding seismic warning data aren't sent to the common citizens in Turkey due to the lack of Cell Broadcast services in the country, which provide an efficient and economical way for simultaneously sending seismic early warning information to millions of people [2-4].

In January 2007, the Japanese national TV channel NHK began broadcasting earthquake and tsunami warning information through datacasts, using DVB-S (Broadcasting Satellite) digital broadcasts, terrestrial digital broadcasts (ISDB-T), and terrestrial digital broadcasts for

mobile receivers (1 seg) [5]. As of early 2013, Japan is the only country in the world with a fully active earthquake and tsunami early warning system, which is integrated with all available types of mass communication systems and networks in the country [5].

Previous research in the field includes the project of Rascioni *et al*. which addresses topics such as emergency and alert dissemination in the DVB-T environment, with special emphasis on the Civil Protection Operational Centre (CPOC) network in the Marche region of Italy [6]. The research project of Pau *et al*. focuses on Cell Broadcast messaging services [7]. Hester *et al*. have based their research on the integration of information systems for post-earthquake research response [8]; while Fortier *et al*. have developed an Early Warning System using IDEF0 and information modeling [9].

As of early 2013, KOERI in Istanbul is working on the available methods and alternative technologies to timely and efficiently disseminate earthquake and tsunami early warning alerts and quick seismic information reports to governmental institutions, schools, transportation networks and energy networks through the use of cell phone text messages and the Internet [2-4]. My goal is to develop Xlet-based IDTV applications that will promptly warn TV viewers through automated pop-up application layers instantly appearing (superimposed) over the video layers on their TV screens; thus enabling them to take caution for incoming earthquakes and/or tsunamis, and receive near-real-time online information. In the future, the live EEW data feeds can be received from the KOERI servers (http://www.koeri.boun.edu.tr); which, however, are still not available to the public as of early 2013, so I have to use an EEW server-simulator website (http://earthquake.godoro.com) that sends EEW data text files to my Xlet applications through the Internet.

I created my Xlet applications in two stages: First, I studied the general characteristics of Java applets, from which the Xlets were derived. During this process, I tested a number of useful Java applet commands to see how they worked with the Xlet codes that I edited using NetBeans IDE 7.0 and 7.2, and how they would complement each other to add new functions or improve existing ones. I tested the Xlet applications with an emulator-added version of the XleTView software, in order to see how they would appear on the screen of a TV receiver that's connected to a DVB-MHP set-top box. The second stage largely involved the retrieval and processing of online data by these Xlet applications, and the run-tests and re-edits for finalizing the Xlet codes in NetBeans IDE 7.0 and 7.2.

In order to improve the efficiency and speed of the Xlets, I designed a basic system architecture and content management; allowing the Xlets to easily access the resident MHP resources, as well as contacting external websites or links for online data retrieval and the submission of feedback information through the return channels for user interactivity. The simple user interface (which consists of the dials on the remote control device) helps to overcome an important hurdle in the digital divide, by ensuring that these Xlet applications are easy to access and use for all age groups.

The topics that are covered in this paper will be useful for developers of IDTV applications; IDTV broadcasters and service providers; and set-top box manufacturers.

2. Xlets of Interactive Applications

2.1. Recent Seismic Activity Report Application

I selected the channel (remote control button) number "7" for the "Recent Seismic Activity Report" application, which provides information on the latest seismic activities within the last 24 hours. The earthquakes are classified as *Low risk* (for up to 3.5 on the Richter scale, highlighted with yellow text); *Medium risk* (between 3.5 and 5.0 on the Richter scale, highlighted with orange text); and *High risk* (for 5.0 and above on the Richter scale, highlighted with red text) by this application.

In the simulated version (based on sample data from the .txt file at http://earthquake.godoro.com/list.txt) the earthquakes are listed in the application window according to the size of their magnitude on the Richter scale (from the largest to the smallest) as seen in **Figures 1** and **2**. The sample .txt file (which is received by the Xlet from the URL http://earthquake.godoro.com/list.txt) provides the following data:

Sea of Marmara;40.4938(N) 28.0917(E);> 7.6 Richter Scale;High;30;30
Central Anatolia;30.5434(N) 26.1214(E);> 5.2 Richter Scale;High;150;100
East Anatolia;40.4938(N) 33.1234(E);> 4.4 Richter Scale;Medium;325;110
Mediterranean Region;25.5434(N) 30.1214(E);> 3.9 Richter Scale;Medium;150;200
Black Sea Region;42.4938(N) 33.1234(E);> 3.2 Richter Scale;Low;225;50
Aegean Region;25.5434(N) 20.1214(E);> 2.1 Richter Scale;Low;25;100

In order to get detailed information regarding the listed recent earthquakes, the user must select them with the direction buttons "Up Arrow" and "Down Arrow", and then press on the "OK" button.

2.2. Recent Seismic Activity Data from KOERI

The URL http://www.koeri.boun.edu.tr/sismo/eng3.txt contains a constantly updated .txt file with data regarding

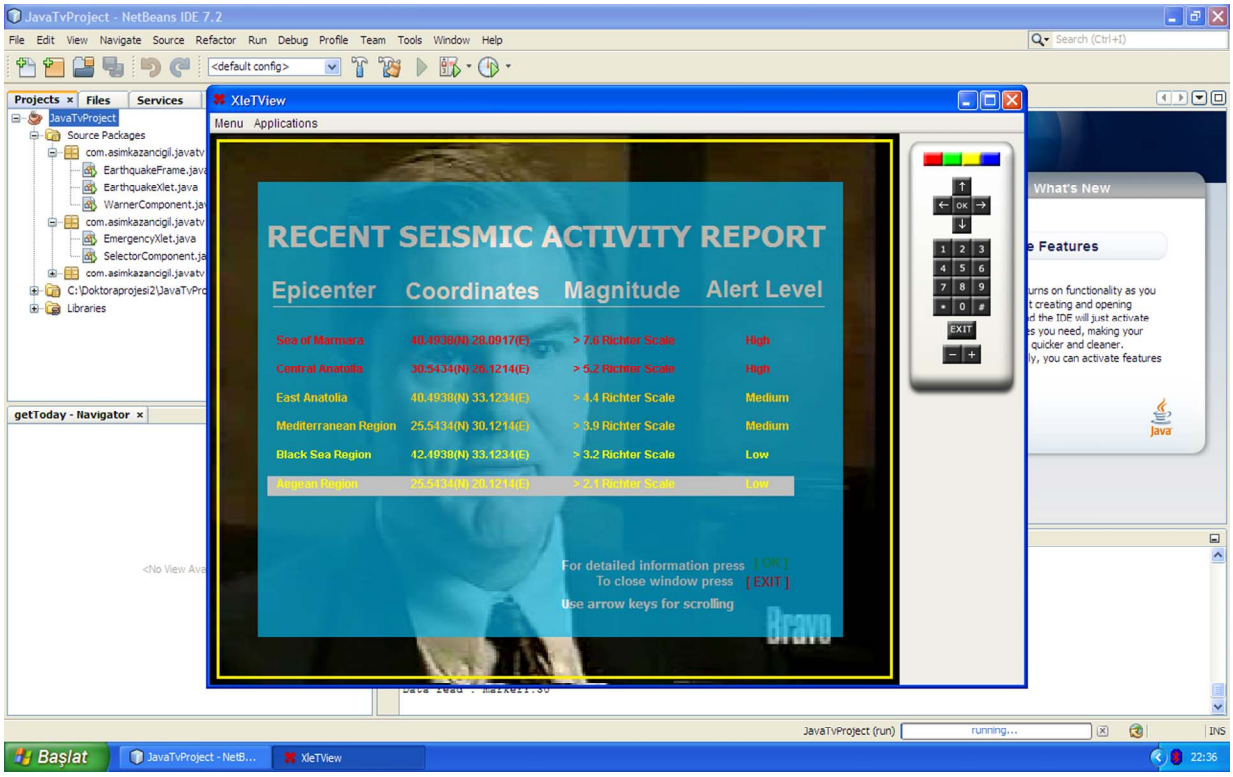

Figure 1. The "Up Arrow" and "Down Arrow" buttons on the remote control device are used for selecting a recent earthquake from the list, and the "OK" button is dialed to get detailed information.

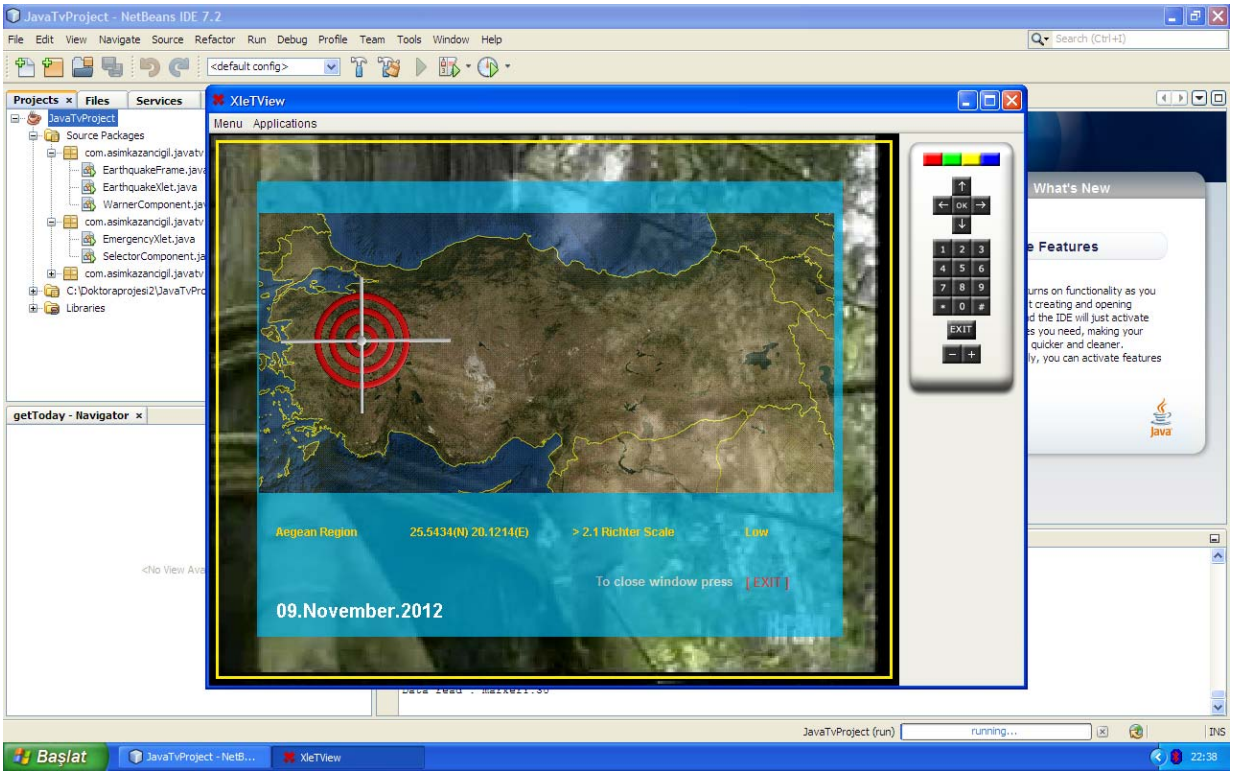

Figure 2. Once the "OK" button is pressed, detailed information regarding the selected earthquake's epicenter, coordinates, magnitude and risk level are given. In this example, the earthquake has a magnitude of 2.1 on the Richter Scale and is therefore a *Low risk* level earthquake, highlighted in yellow text.

the recent earthquakes in and near Turkey, as measured by KOERI in Istanbul. Based on similar text files which KOERI uses for SMS messages, it is possible to use the live GeoSIG and GeoDAS data through the internet for activating the *Earthquake Early Warning* and *Recent Seismic Activity Report* application Xlets.

I inserted this URL to EarthquakeXlet.java as seen below:

private String LIST_URL =
"http://www.koeri.boun.edu.tr/sismo/eng3.txt";

I used the parsing method for utilizing the KOERI data in my application, and to create a dynamic map showing the exact location of the earthquake epicenters on the map of Turkey based on latitude and longitude information provided by the dynamic .txt file in the KOERI website, with the parsing codes seen in **Appendix A**.

Through these parsing codes, the Xlet retrieves the necessary seismic data (date, time, latitude, longitude, magnitude, region (location of the epicenter)) from the constantly updated dynamic text file at the KOERI website (http://www.koeri.boun.edu.tr/sismo/eng3.txt) seen in **Figure 3** and utilizes them in the *Recent Seismic Activity Report* application's windows, seen in **Figures 4** and **5**.

2.3. Earthquake and Tsunami Early Warning

The "Earthquake and Tsunami Early Warning" application, which is assigned the channel number "8" on the remote control device, is an innovative one that enables observatories to send live early warning information regarding seismic activities (such as earthquakes, volcanic eruptions and tsunamis) to the digital video broadcasting networks. This application aims to provide a live information feed between the Kandilli Observatory and Earthquake Research Institute (KOERI) at the Boğaziçi University in Istanbul, Turkey, with the IDTV networks in the country such as Digiturk and D-SMART.

The early warning system will potentially be able to save many lives in situations of large-scale seismic natural disasters, by informing the citizens (in the case of my application, TV viewers) several seconds before an earthquake, or several minutes before a tsunami, upon receiving unusually large *P-wave* signals (which, in most cases, precede the *S-waves* of an incoming earthquake).

KOERI is currently working on methods to provide live seismic early warning alerts and quick seismic information reports to governmental institutions, schools, transportation networks and energy networks through the use of cell phone text messages and the internet, in cases of increased seismic activity detection [2-4].

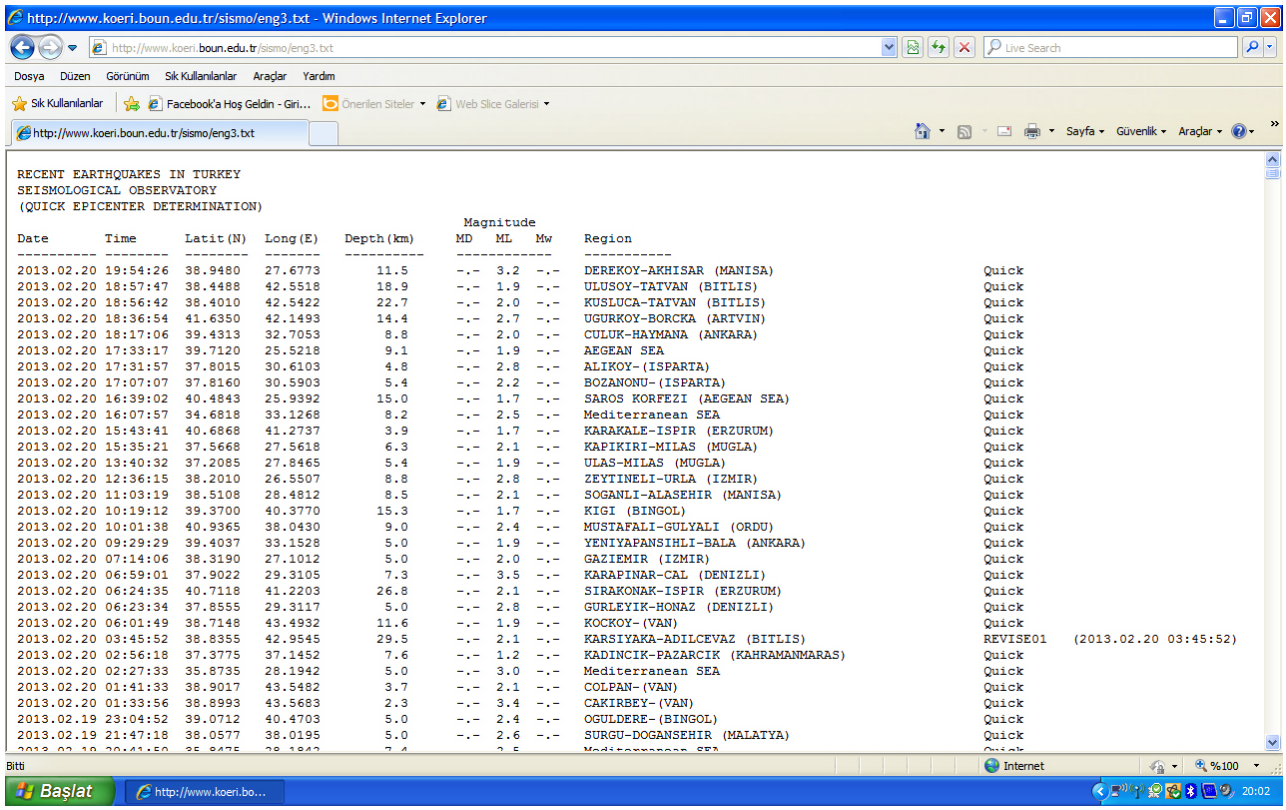

Figure 3. Screenshot of the KOERI link with the constantly updated text file which provides data regarding the recent earthquakes in and near Turkey.

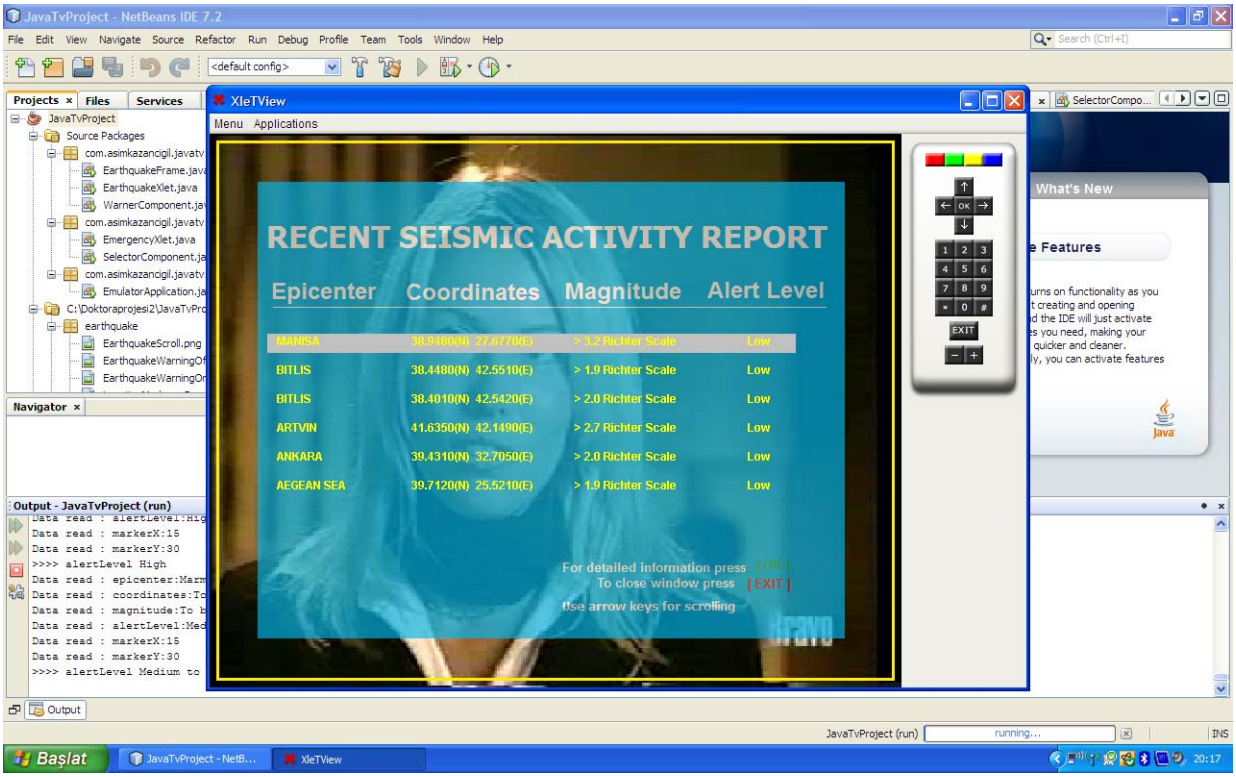

Figure 4. List of the recent earthquakes based on data retrieved online from the KOERI website. The user can select an earthquake from the list by utilizing the "Up Arrow" and "Down Arrow" buttons on the remote control device, then press on the "OK" button to see detailed information.

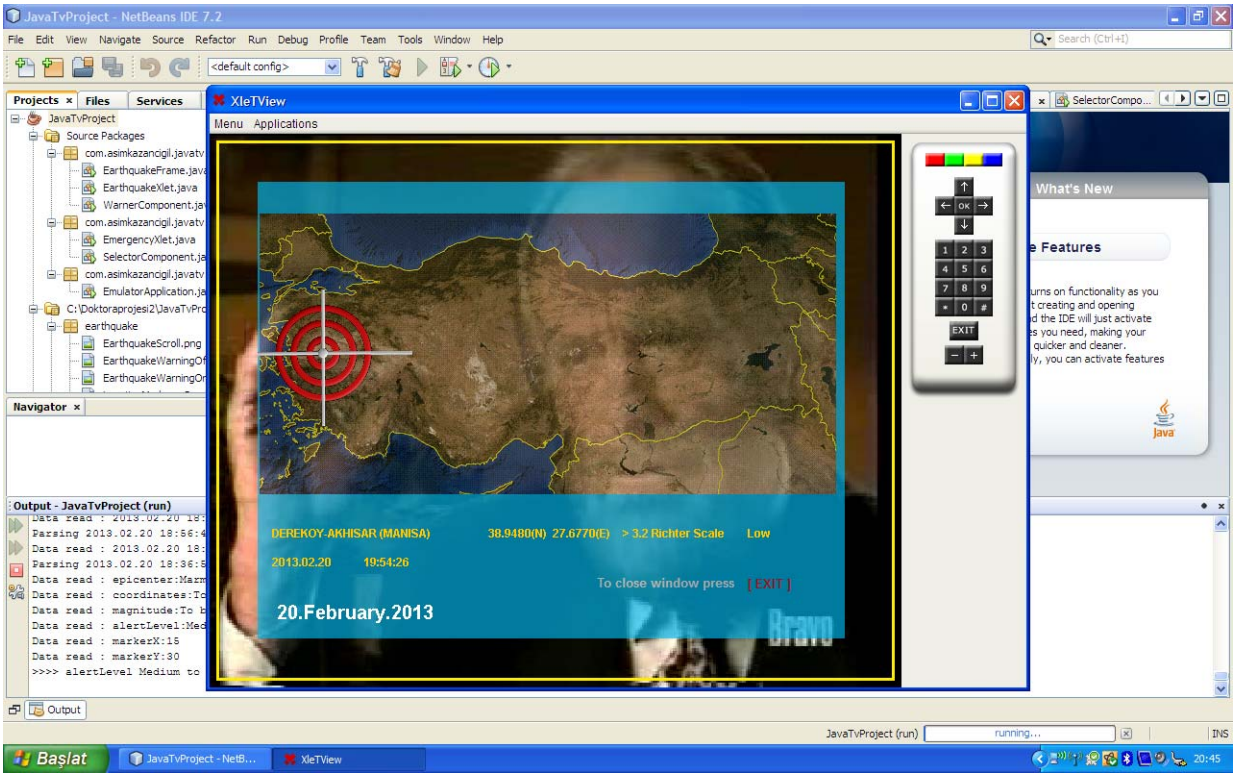

Figure 5. Detailed information (location, coordinates, magnitude, risk level, date and time) regarding the selected earthquake, based on constantly updated data retrieved online from the KOERI website.

My aim is to develop an Xlet-based IDTV application that will automatically detect any signal of abnormally high *P-wave* activity increase and promptly warn the television viewers through automated pop-up applications that will instantly appear on their television screens; thus enabling them to take caution for incoming earthquakes and/or tsunamis, and receive near-real-time online information.

The live seismic activity information feed can be received from the KOERI servers of Boğaziçi University (http://www.koeri.boun.edu.tr) in the future. However, as of February 2013, the live internet feed of Boğaziçi University's Earthquake Early Warning System regarding the signals gathered by the early warning stations is still not available to the public, so I had to create a sample "simulated .txt file" that provides earthquake warning information to my Xlet application through the internet, which can be seen at http://earthquake.godoro.com where I stored the code.

The URL http://earthquake.godoro.com/early.txt contains a sample information text code (.txt file) for the *Earthquake and Tsunami Early Warning* application demo, which indicates an increased *P-wave* activity alert and is received by the Xlet through the internet. Upon arriving, it opens a pop-up window above the video layer on the TV screen. The application can also be accessed by dialing "8" on the remote control device, but the pop-up window automatically opens only in cases of *Medium to High risk* earthquake early warning alerts, based on strong *P-wave* activity, as seen in **Figure 6**.

The URL http://earthquake.godoro.com/emergency.txt contains the sample information text code (seen below) for a *High risk* level earthquake warning example (7.6 on the Richter Scale, with its epicenter at coordinates 40.4938(N) 28.0917(E) in the Sea of Marmara, seen in **Figures 7** and **8**) which is received by the Xlet through the internet after the epicenter and magnitude of the earthquake are calculated with seismic data arriving from the measurement stations:

epicenter:Sea of Marmara
coordinates:40.4938(N) 28.0917(E)
magnitude:> 7.6 Richter Scale
alertLevel:High
markerX:15
markerY:30

It takes circa 1 minute (through computer-automated calculation) [4] or between 3 to 5 minutes (calculation through manually-added data arriving from at least 3 different measurement stations in the area (thus forming a triangle around the epicenter) into the software, which provide more accurate results) to determine the exact coordinates and magnitude of an earthquake [3,4].

Figure 6. "Medium to High risk" (possibly 3.5 or more on the Richter Scale) *P-wave* based live earthquake early warning information, which appears as a pop-up window on the TV screen. At this point, the exact magnitude and epicenter of the earthquake are not known.

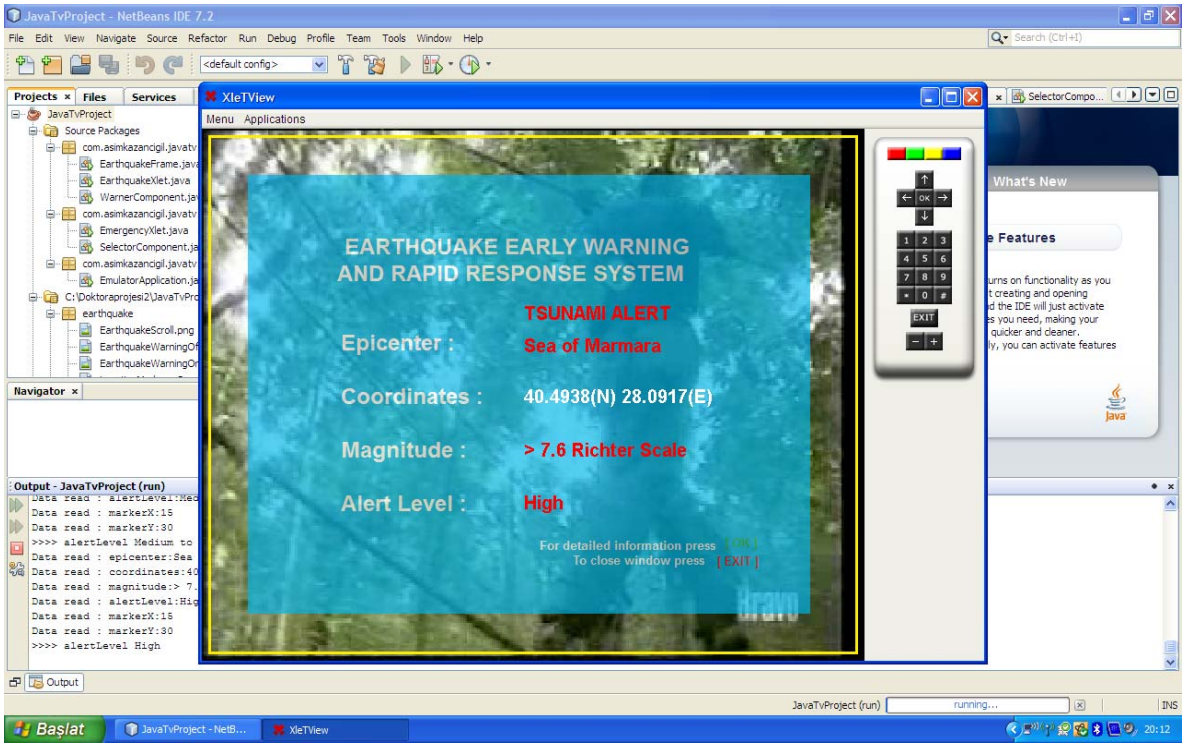

Figure 7. After calculations lasting circa 1 to 5 minutes following the earthquake, based on data from at least 3 different seismic measurement stations, the earthquake's epicenter (40.4938(N) 28.0917(E)) and magnitude (7.6 on the Richter scale) are determined and begin to appear on the pop-up application window, together with a "Tsunami Alert" for the coastal towns on the Sea of Marmara. It is estimated that the tsunami waves will reach the Sea of Marmara coastline of Istanbul in about 5 to 10 minutes.

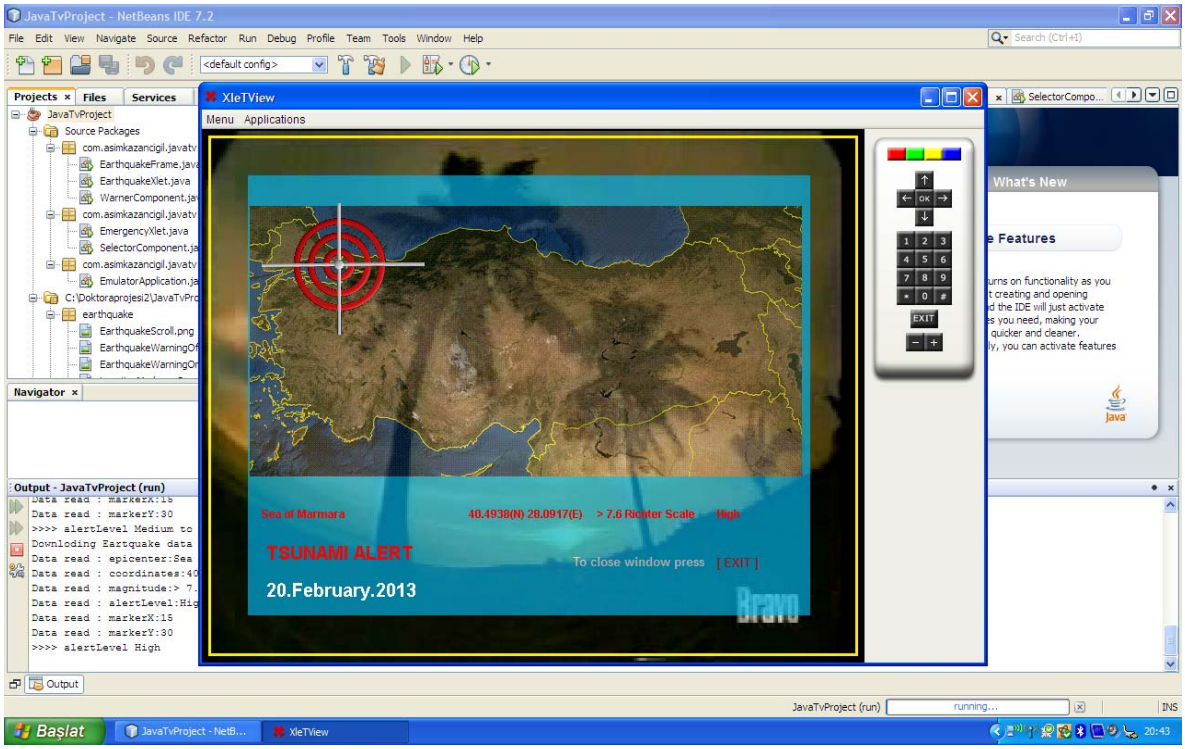

Figure 8. Map showing the location of the earthquake's epicenter coordinates, magnitude, and Tsunami Alert, all of which are highlighted in red text due to the *High risk* level.

The more data from different stations are added into the calculation, the more accurate the epicenter coordinates and magnitude figures will become [3,4]. As of October 2012, there were 205 seismic activity measuring stations across Turkey [3]. Earthquake data comes from these sensor stations to KOERI as 1-second packages of 24-bit information, with the arrival time of 500 milliseconds [4].

There are three *alert levels* in the Earthquake and Tsunami Early Warning application: *Low risk* (for up to 3.5 on the Richter scale, highlighted with yellow text); *Medium risk* (between 3.5 and 5.0 on the Richter scale, highlighted with orange text); and *High risk* (for 5.0 and above on the Richter scale, highlighted with red text). The early warning application that I developed is in the form of an automated pop-up window, which appears on the top of the video layer. Once the earthquake data (magnitude, epicenter location, coordinates) are calculated by KOERI (in between 1 to 5 minutes) the interactive functions of the application allow the user to get updated information regarding the *magnitude*, *epicenter location*, *coordinates*, *alert level* and, in some cases, *tsunami alert*, of a recent seismic activity.

In the example seen in **Figures 7** and **8**, the magnitude of the earthquake is 7.6 on the Richter scale; as such, it falls into the category of *High risk* and thus, its data and Tsunami Alert texts are highlighted in red. Earthquakes up to 3.5 on the Richter scale are categorized as *Low risk* (highlighted in yellow text); while those between 3.5 and 5.0 on the Richter scale are categorized as *Medium risk* (highlighted in orange text) by the application.

When the user clicks on the "OK" button in the middle of the remote control device in order to get detailed information about the earthquake, another window (seen in **Figure 8**) displaying the exact location of the seismic activity's epicenter over a map of Turkey opens. The window also includes information regarding the epicenter, its coordinates, the magnitude of the earthquake, and Tsunami Alert; all of which are highlighted in red due to the *High risk* alert level. The application is closed by clicking on the "EXIT" button on the remote control.

The system architecture consists of three layers: hardware and software resources; middleware; and applications.

Typical hardware resources are MPEG processing (such as video, audio, and data decoders); CPU; memory; a graphics processor (*i.e.*, OSD); modem and network interface; tuner and demodulator; demultiplexer and decryptor; smart card reader; remote control and storage devices, etc. [10-12].

The software resources include all device drivers, such as the real-time operating system [10,11].

Middleware includes a Java virtual machine; APIs; an application manager (and application specific libraries) and/or resident television-specific applications [10,11].

The real-time operating system (RTOS) and related device-specific libraries control the hardware via a collection of device drivers [10]. The operating system provides the system-level support needed to implement the Java virtual machine and class libraries that comprise the DVB-J platform [13].

The flow chart in **Figure 9** displays the current status of the earthquake early warning (EEW) communication systems in Japan. The flow chart in **Figure 10** displays the current status (as of early 2013) of the available EEW communication systems in Turkey, while the flow chart in **Figure 11** displays the possible future EEW services that are currently under consideration in Turkey.

Figure 9. EEW communication services in Japan.

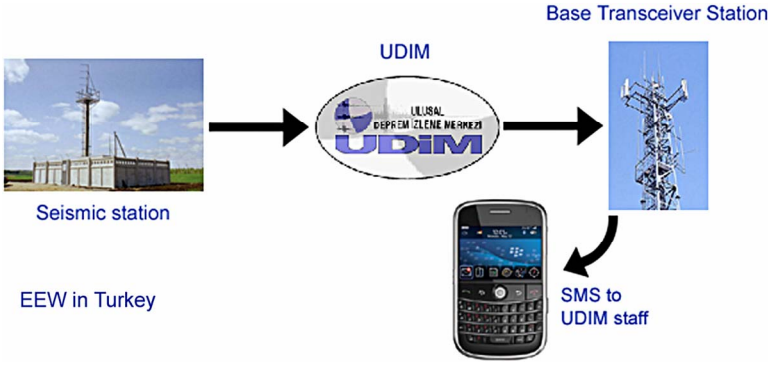

Figure 10. The currently available EEW communication services in Turkey. Information (location, epicenter, depth, magnitude, etc) regarding earthquakes larger than 3.0 on the Richter scale are sent to the UDIM staff; those larger than 4.0 are sent to the Turkish government institutions like AFAD; while those larger than 5.0 are sent to search and rescue organizations like AKUT. The SMS messaging service is provided by the cell phone operator Turkcell.

Figure 11. Possible future EEW communication services that are currently under consideration in Turkey.

The Java virtual machine (JVM) is used to interpret an application's Java bytecodes and is responsible for a system's hardware and operating system independence. It must have a relatively small size and the ability to execute code securely. A Java virtual machine knows nothing of the Java programming language, only a particular binary format known as the class file format [10,14].

A class file contains the Java virtual machine instructions (or bytecodes) and a symbol table as well as other information. Some important mechanisms provided by the Java virtual machine are vital to IDTV applications. For example, bytecode verification provides guarantees about the validity of instructions being executed; class loading mechanisms enforce how code is loaded into the machine and can provide guarantees about the code's source while strong name-space management decreases the chance of an intruding unauthorized code [10].

An API operates within the hardware context of the set-top box and encapsulates the functionality exposed by the system libraries that control the television-specific hardware on the device [13].

The APIs used in the system architecture include basic Java APIs; JMF APIs; a Java API for XML parsing and a Java API for XML Messaging [10,13].

The application libraries used by the resident applications are also stored on the set-top box. One of the basic requirements for middleware is a small footprint and, in addition, the middleware can be stored in Flash ROM to improve performance speed in a set-top box [10].

Many constraints exist for user interface development in this type of platform, such as TV remote control type buttons with limited functionality [10]. It is essential to use a simple remote control event model and lightweight drawing components, by using the Java Abstract Win-

dow Toolkit (AWT) and Java Media Framework (JMF) together with NetBeans IDE 7.0 and/or 7.2.

It is also necessary to enable interactivity methods using Java APIs via a return channel through the internet or a modem. The possible communication models include URL, Socket, Datagram, and SOAP models which the Xlets can choose in order to establish a connection with the service or broadcaster for data transfer [10].

2.4. Emergency Services Application

The channel number "9" on the remote control device initializes the main menu window of the "Emergency Services" application, seen in **Figures 12** and **13**. The main window of the application consists of three available options which can be chosen by using the "cursor buttons" (← (left) and → (right)) and the "OK" button. Once the selection is made among the available services (in the example on **Figure 14**, the Ambulance service) by dialing the left or right cursor buttons, the user must then press on the "OK" button to activate the application.

The detailed information windows of the services have similar content, which consist of a brief description of the service and a request for the user (in this case, a digital television network subscriber) to verify and confirm his/her address. If the address is correct, the user can press on the "OK" button for calling Policemen, Firemen or an Ambulance to his/her address. Pressing on the "EXIT" button allows the user to return to the main menu, while pressing on "EXIT" another time closes the application window.

People often panic during cases of emergency and the phone number of the Police, the Fire Department or the Hospital may not be immediately available. Using the internet to find the phone numbers might be an option; but it's time consuming, and especially senior citizens find such methods difficult. My application allows TV viewers to easily and quickly reach these emergency services through the use of the remote control device. Digital television network subscribers who agree to share their identity and home address will be able to take advantage of such a rapid service.

In the future, specifically designed remote control buttons for the Police, Ambulance or Fire Department services may also improve the speed and efficiency of such applications, especially for the senior citizens and people who have health problems.

2.5. Multithreading

Since Java applications have built-in support for multithreading, it is relatively easy for a Java developer to transform an application into a threaded one. Multithreaded programming in Java is provided by two classes:

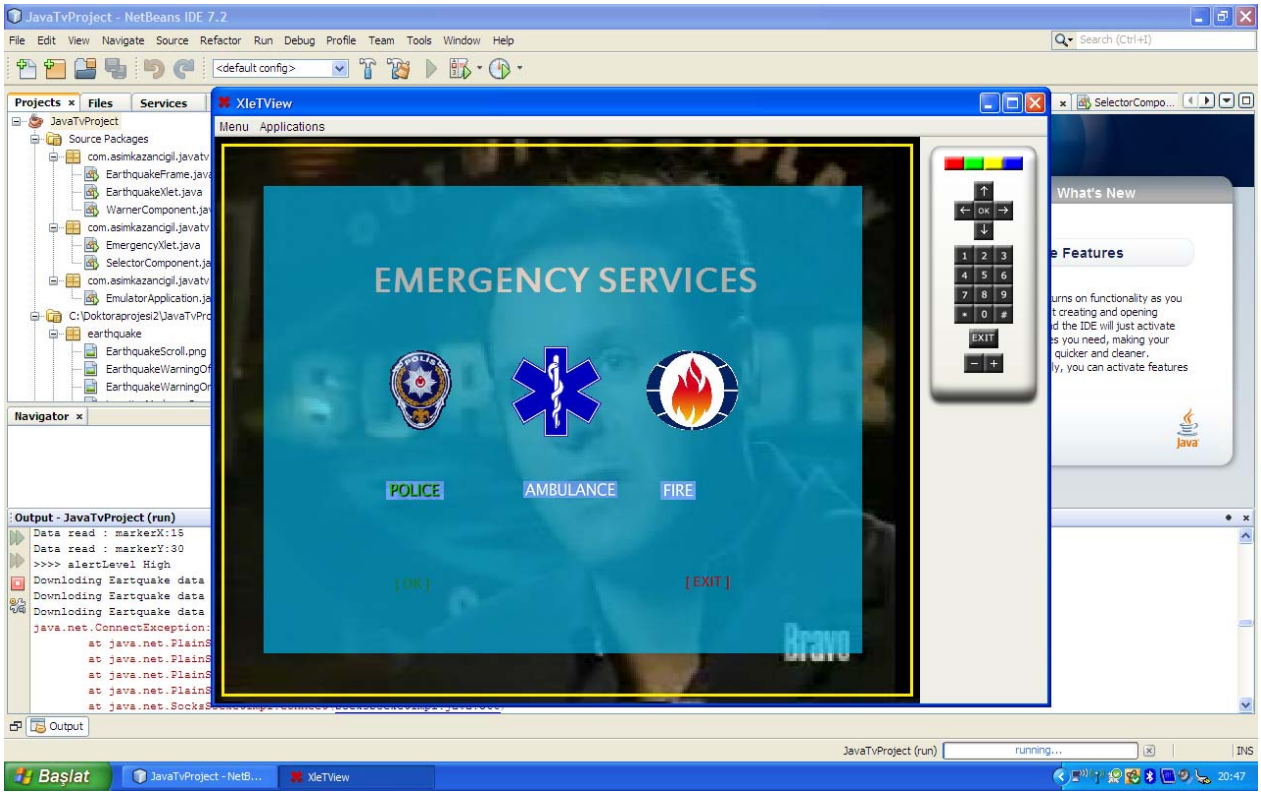

Figure 12. Dialing the channel number "9" on the remote control device activates the main menu window of the Emergency Services application.

Figure 13. Once the main menu window of the application is activated, the user can select among the available services (Police, Ambulance and Fire Station) by dialing the "cursor buttons" (← (left) and → (right)) and the "OK" button.

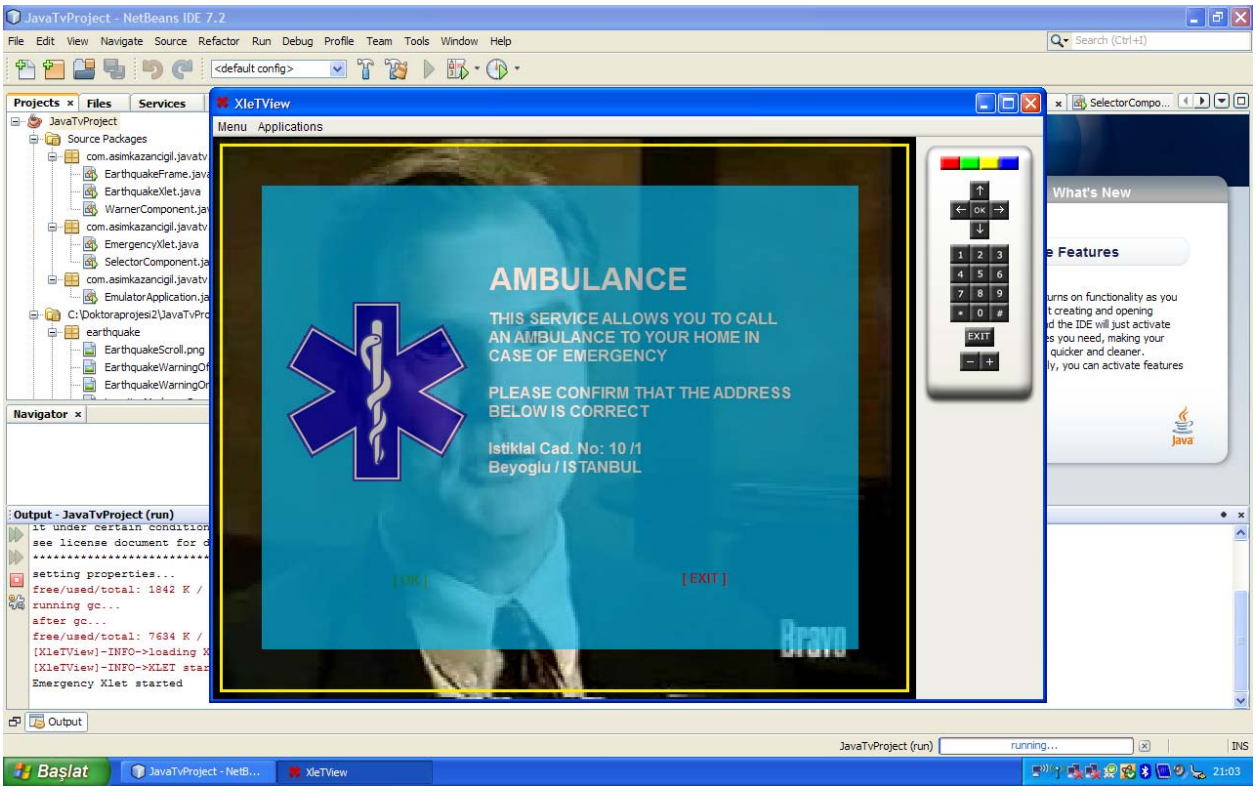

Figure 14. Description of the selected Emergency Service and the DVB network subscriber's address information. The user must dial the "OK" button for confirmation.

Thread and Runnable. The Runnable class is an interface that's defined as a virtual/abstract class; while the Thread class is capable of running tasks in the background. The Thread class can also be used for defining a task so that it can run asynchronously. However, it is easier and more common to use the Runnable interface for the task to be executed. The Runnable interface may be implemented in another class by the user of threading; however, in most cases, an inner class defines the run() method, so that no other class will be needed.

Multithreading in Java is implemented by defining a run() method and invoking a start() method. When Thread.start() is called, Java adds the Runnable.run() method to the list of tasks to be executed afterwards. After the start() method is called, the normal thread will continue to run. When the normal thread (with a high priority) is complete (or it abruptly ceases to work, or is waiting for another task to be done) Java may proceed from the normal thread to an alternative one defined by the thread. This process is administered automatically by a system called Monitor, which is transparent to the developer.

The code in **Appendix B** is a typical example for thread usage in Java. As it can be seen in the code in **Appendix B**, the run() method is defined by the start() method, which is called in the Java threading system. In my Xlets, the code is similar to the one in **Appendix C**. The method downloadData() retrieves data from the server via HTTP.

Although a simple thread is adequate for multithreading, repeating tasks are not easy to develop using the Thread class. For tasks which are executed in definite intervals, another class named Timer is generally used. The Timer class itself uses the Thread class in the background and waits (or sleeps) for intervals. Java TV supports another timer for repeating tasks. This is called the TVTimer. Alongside the TVTimerWentOffListener interface and TVTimerSpec class, Java TV supports a timer utility which is better for TV applications. The use of TVTimer is similar to the use of Thread.

The code in **Appendix D** calls repeatingTask() (or any method) in intervals of 30,000 milliseconds. Since TVTimer is defined specifically for TV sets, it is better to use it in TV applications.

In my Xlet applications, Thread and TVTimer are used together. TVTimer is used to trigger a connection to the server in definite intervals, while Thread is used to connect and retrieve data asynchronously. TVTimer also supports single execution, which means it may be used instead of Thread when the setRepeat() method is called by a "false" argument.

Some Xlets run perfectly on XleTView and JavaTV, but can't be decoded and visualized by certain set-top boxes or TV cards due to issues of incompatibility. A software programmer should be careful to use basic and universal Xlet codes to ensure a high degree of compatibility, without compromising the new functions and services that will be added to the IDTV environment, and try to find new methods to by-pass the shortcomings or correct the errors with a new approach.

There were numerous problems with the software of XleTView, which I had to resolve by compiling a Java TV emulator code that enables XleTView to run properly. Also, the software is compatible with only the earliest codecs of QuickTime video (such as Cinepak) which was a problem. In order to obtain a versatile and easy to use content management system, I stored the media files in the folder JavaTvResources, while the folder Libraries contains the class files. In recent years, JavaTV has emphasized more on Internet TV and broadcasts to handheld devices, therefore it is essential to keep the content management and the size of the files as basic and light as possible; thus optimizing the versatility of the Xlets and improving their usability by different devices with a high level of performance.

Set-top boxes will become the center of networked homes in the future and interactive applications will be an important part of this increased convergence between various device types. The evolution of the set-top box and convergence of computer, communications, and consumer products will define the future of similar IDTV applications. Mobile digital television enables the same information to be broadcast to the mobile (e.g. WAP, GPRS) and portable (e.g. PDA, laptop, iPhone) devices while roaming. However, the use of the same interactive application will not be possible for all types of devices, whose manufacturers should agree on common standards. Creating systems and devices entirely based on Java technology may improve the development speed and interoperability of diverse devices.

Since the DVB-T network must use the ACAP system for the implementation of ACAP-J API which codify the digital and interactive content that are generated by Java (while ACAP-X does the same thing for content generated by Markup), a software programmer must develop methods to optimize the use of both ACAP-X and ACAP-J API, particularly the latter.

One of the disadvantages of mobile systems such as DVB-H is the fact that they can't utilize the architecture that's developed for DVB-MHP set-top boxes which are used at home (due to the difference in the characteristics of various terminals); therefore a middleware architecture should be developed for mobile device (DVB-H) specific applications of IDTV, in order to handle the mobile teletext and DVB-H compatible EPG services more efficiently.

3. Conclusions

The main goal of the research that is covered in this pa-

per is to find a way to communicate information about natural disasters such as earthquakes and tsunamis in a timely and efficient way. My approach for achieving this goal is structured on the prototyping of Xlet-based interactive applications.

In order to realize this goal, I decided to emphasize on the Early Warning Systems for seismic disasters in the IDTV environment and collaborated with KOERI in Istanbul. To validate the approach, I developed Xlet-based interactive applications such as *Earthquake and Tsunami Early Warning*; *Recent Seismic Activity Report*; and *Emergency Services*. On January 31, 2013, I presented these applications to the members of UDIM and KOERI in Istanbul as part of a proposed communication strategy for reaching the disaster-stricken communities in seismically active zones.

For receiving and processing live online data and sending interactive feedback through the return channels, I elaborated a basic system architecture and content management system for Xlet-based applications. The design allows the Xlets to efficiently access the resident MHP resources, as well as contacting external websites for online data retrieval. Thanks to the simple user interface based on the commands of the remote control device, these Xlet applications are quick to access and easy to learn and use for all age groups, thus overcoming an important hurdle in the digital divide.

The research subjects that are covered in this paper will be useful for Java application developers; IDTV service providers; broadcasters; and set-top box manufacturers.

4. Acknowledgements

I would like to thank everyone who helped me during the realization of this project; in particular Prof. Ernesto Damiani, Prof. Giorgio Valle and Dr. Raffaella Folgieri from the University of Milan for their extensive guidance and supervision during my research on Xlets and IDTV applications. I would also like to thank Prof. Mustafa Erdik, Dr. Doğan Kalafat and Dr. Mehmet Yılmazer from the Kandilli Observatory and Earthquake Research Institute (KOERI) of Boğaziçi University in Istanbul for their valuable help in my research and for providing me with the necessary information regarding seismic matters, as well as the links to the online data about the most recent seismic activities in and around Turkey. Another person I would like to thank is Mr. Önder Teker for his guidance on Java and Xlet coding, which played an important role during the creation of my applications.

REFERENCES

[1] H. Alcik, O. Ozel, Y.-M. Wu, N. M. Ozel and M. Erdik, "An Alternative Approach for the Istanbul Earthquake Early Warning System," *Soil Dynamics and Earthquake Engineering*, Vol. 31, No. 2, 2010, pp. 181-187. doi:10.1016/j.soildyn.2010.03.007

[2] M. A. Kazancigil, "Interviews with Prof. Mustafa Erdik on the Istanbul Earthquake Early Warning System," Kandilli Observatory and Earthquake Research Institute (KOERI), Boğaziçi University, Istanbul, 2012.

[3] M. A. Kazancigil, "Interviews with Dr. Doğan Kalafat on the Istanbul Earthquake Early Warning System," Kandilli Observatory and Earthquake Research Institute (KOERI), Boğaziçi University, Istanbul, 2012.

[4] M. A. Kazancigil, "Interviews with Dr. Mehmet Yılmazer on the Istanbul Earthquake Early Warning System," Kandilli Observatory and Earthquake Research Institute (KOERI), Boğaziçi University, Istanbul, 2012.

[5] Y. Nomoto, "Earthquake and Tsunami Information Services via Data Broadcasting," 2007 *Annual Meeting*, Tehran, 30 October-1 November 2007.

[6] G. Rascioni, E. Gambi, S. Spinsante and D. Falcone, "DTT Technology for Rural Communities Alerting," *Proceedings of the 5th International ISCRAM Conference*, Washington DC, May 2008.

[7] L. F. Pau and P. Simonsen, "Emergency Messaging to General Public via Public Wireless Networks," *Proceedings of the 5th International ISCRAM Conference*, Washington DC, May 2008.

[8] N. C. Hester, S. P. Horton, J. Wilkinson and T. I. Jefferson, "Integration of Information Systems for Post Earthquake Research Response," *Proceedings of the 5th International ISCRAM Conference*, Washington DC, May 2008.

[9] S. C. Fortier and I. M. Dokas, "Setting the Specification Framework of an Early Warning System Using IDEF0 and Information Modeling," *Proceedings of the 5th International ISCRAM Conference*, Washington DC, May 2008.

[10] C. Peng, "Digital Television Applications," Ph.D. Dissertation, Helsinki University of Technology, Helsinki, 2002.

[11] EN 300 421, "Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for 11/12 GHz Satellite Services," European Telecommunications Standards Institute, Sophia Antipolis, 1997.

[12] G. O'Driscoll, "The Essential Guide to Digital Set-Top Boxes and Interactive TV," Prentice Hall PTR, Upper Saddle River, 1999.

[13] B. Calder, J. Courtney, B. Foote, L. Kyrnitszke, D. Rivas, C. Saito, J. Van Loo and T. Ye, "Java TV API Technical Overview," *The Java TV API White Paper*, Sun Microsystems, Palo Alto, 14 November 2000.

[14] T. Lindholm and F. Yellin, "The Java Virtual Machine Specification," Addison-Wesley Longman Publishing Co., Inc., Boston, 1999.

Appendix

Appendix A

```
private void downloadFormData(String address) {
        try {
            System.out.println("Downloading
Earthquake data from URL " + address);
            BufferedReader br = open-
Reader(address);
            Vector values = new Vector();
            String line = null;
            while ((line = br.readLine()) != null) {
                System.out.println("Data read : " +
line);
                int index = line.indexOf(":");
                String value = line.substring(index
+ 1).trim();
                values.add(value);
            }
            br.close();
            setFormData(values);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private void downloadListData(String address) {
        try {
            System.out.println("Downloading
Earthquake data from URL " + address);
            BufferedReader br = open-
Reader(address);
            Vector values = new Vector();
            String line = null;
            boolean parse=false;
            int row=0;
            while ((line = br.readLine()) != null) {
                System.out.println("Data read : " +
line);
                if(line.startsWith("----")){
                  parse=true;
                  continue;
                }
                if(parse){
                  System.out.println("Parsing
"+line);
                  String date=line.substring(0,10);
                  String
time=line.substring(11,19);
                  String lati-
tude=line.substring(21,27);
                  String longi-
tude=line.substring(31,37);
                  String
md=line.substring(55,58);
                  String ml=line.substring(60,63);
                  String
ms=line.substring(70,73);
                  String re-
gion=line.substring(71,116);

                  String magnitude=ml;
                  if(ml.equals("-.-")){
                    if(!md.equals(address)){
                      magnitude=md;
                    }else if(!ms.equals(address)){
                      magnitude=ms;
                    }else{
                      magnitude="0.0";
                    }
                  }
                  double latitudeDou-
ble=Double.parseDouble(latitude);
                  double longitudeDou-
ble=Double.parseDouble(longitude);
                  double magnitudeDou-
ble=Double.parseDouble(magnitude);
                  String level="Low";
                  if(magnitudeDouble>5){
                    level="High";
                  }else if(magnitudeDouble>3){
                    level="Low";
                  }
                  String[] fields=new String[8];
                  fields[0]=region;

fields[1]=String.format(Locale.US,"%6.4f(N)
%6.4f(E)",latitudeDouble,longitudeDouble);

fields[2]=String.format(Locale.US,"> %2.1f Richter
Scale ",magnitudeDouble);
                  fields[3]=level;

fields[4]=Integer.toString(calculateX(longitudeDouble));

fields[5]=Integer.toString(calculateY(latitudeDouble));
                  fields[6]=date;
                  fields[7]=time;
                  values.add(fields);
                  row++;
                  if(row==ROW_COUNT){
                    break;
                  }
                }
            }
            br.close();
            setListData(values);
        } catch (Exception e) {
            e.printStackTrace();
```



```
            }
        }
        private static int calculateX(double longitude){
            return
-MARKER_CENTER_X+MAP_X+(int)(MAP_WIDTH
*(longitude-LONGITUDE_MIN)/(LONGITUDE_MAX
-LONGITUDE_MIN));
        }
        private static int calculateY(double latitude){
            return
-MARKER_CENTER_Y+MAP_Y+(int)(MAP_HEIGH
T*(LATITUDE_MAX-latitude)/(LATITUDE_MAX-LA
TITUDE_MIN));
        }
        private void setFormData(Vector values) {
              String epicenter = (String) values.get(0);
              String coordinates = (String) values.get(1);
              String magnitude = (String) values.get(2);
              String alertLevel = (String) values.get(3);
              int markerX = Integer.parseInt((String) val-
ues.get(4));
              int markerY = Integer.parseInt((String) val-
ues.get(5));
              component.setEpicenter(epicenter);
              component.setCoordinates(coordinates);
              component.setMagnitude(magnitude);
              component.setAlertLevel(alertLevel);
              component.setMarkerX(markerX);
              component.setMarkerY(markerY);
              System.out.println(">>>>              alertLevel
"+alertLevel);
              if (alertLevel.contains("High")) {
                    if (!emergencyShown) {
                          earlyShowing = true;
                          emergencyShowing = true;
                          component.appear(true);
                    }else{
                          component.appear(false);

                    }
              }
        }
        private void setListData(Vector values) {
              component.setTable(values);
        }
```

Appendix B

```
Runnable runnable = new Runnable() {
        @Override
        public void run() {
            // Task to be done asynchronously
        }
};
Thread thread = new Thread(runnable);
thread.start();
```

Appendix C

```
Runnable runnable = new Runnable() {
        @Override
        public void run() {
            downloadData();
        }
};
Thread thread = new Thread(runnable);
thread.start();
```

Appendix D

```
TVTimerWentOffListener listener = new TVTimer-
WentOffListener() {
                @Override
                public          void          timerWen-
tOff(TVTimerWentOffEvent tvtwoe) {
                      repeatingTask();
                }
            };
          TVTimerSpec spec = new TVTimerSpec();
            spec.addTVTimerWentOffListener(listener);
            spec.setTime(30000);
            spec.setAbsolute(false);
            spec.setRepeat(true);

TVTimer.getTimer().scheduleTimerSpec(spec);
```

