GIL: a Software Tool for the Construction of Information Dissemination Systems

João P. Campos
Ana Paula Afonso
Mário J. Silva

DI-FCUL TR–99–4

December 1999

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1700 Lisboa
Portugal

Technical reports are available at http://www.di.fc.ul.pt/biblioteca/tech-reports. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

GIL: a Software Tool for the Construction of Information Dissemination Systems

João P. Campos, Ana Paula Afonso, Mário J. Silva

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1700 Lisboa – Portugal
Phone: +351.1.7500153, Fax: +351.1.7500084
{ jcampos, apa ,mjs}@di.fc.ul.pt

Abstract

Push-based delivery has motivated a growing interest, allowing dissemination of information to a large set of users, both in wired and wireless environments. In this paper, we present GIL, a software tool to automate the development process of information dissemination applications, which run on top of Ubidata, our software framework for mobile computing. GIL makes the channel definition and creation an easy task, even for non-technicians.

Keywords: *Data dissemination, Mobile computing applications, Software engineering environments.*

1. Introduction

Most research in mobile computing has addressed adaptation of network protocols to wireless communications and providing operating system support for poor connectivity. We believe that it is essential to also consider the software engineering of applications for mobile users. The constraints of the small portable devices that mobile users will handle to access information and challenging usability requirements anticipate the need for new interaction metaphors and new development environments and processes.

To face the mobility constraints, applications must anticipate user's information needs as they roam with minimum user interaction. Given these intrinsic characteristics, we believe that the information-push paradigm is especially useful for the development of some mobile applications [14].

To address these needs, we have developed Ubidata, a framework to mobile information dissemination [1]. Ubidata looks into mobile computing problems mainly from the perspective of the software engineering of

information processing applications. This framework has been specifically designed for providing a solution to information distribution to a large set of mobile users, based on extensions to channels [6], and a hybrid scheme of dissemination of information [7]. These new channels are dynamic, in the sense that information pushed and presented to individual receivers depends on their location and changes as they roam. Ideally, those who define what information is to be disseminated should not be aware of the complexities of the infrastructure used to disseminate the information, so that those who create and manage the channels are experts in their application domain, and not necessarily computer experts. To provide automatic support for the analysis phase of the development process of Ubidata's applications, we have developed GIL, an open and portable tool for editing and generating dynamic channel specifications. GIL is a multi-user, collaborative, editing tool.

The paper is organized as follows. We discuss related work in Section 2. Section 3 and Section 4 present an overview of UbiData's information model, its architecture, and the development process for UbiData's applications. In Section 5, we describe GIL and present our conclusions in Section 6.

2. Related Work

Our research involves concepts from current work in information dissemination technologies and in object-oriented software engineering. To gain a common understanding of basic terms, we give a short overview about the work that is most related to the theme of this paper.

Information Dissemination Models

Recent advances in information technology have motivated a growing interest for solutions based on the *push* concept, allowing content providers to deliver information directly onto users' desktops. However, almost all existing so-called push applications are still based on the traditional client-server model [14].

Clients are configured to regularly ask for updates to data of interest, without the need for human intervention. These applications are called *smart pull*.

True push is the real broadcast of information, where the provider of the information "feeds" the receivers whenever information is produced. In this model, receivers have a passive role and providers can distribute/disseminate information to a restricted subset of the receivers. The concept of broadcast data delivery is not new, and a survey can be found in [13].

Our work follows a hybrid scheme of data delivery according to the classification suggested by Franklin and Zdonik [7]. In their work, they propose a general architecture for Dissemination-Based Information Systems (DBIS) that incorporates multiple modes of data delivery such as request/response, polling, publish/subscribe and periodic broadcast. In the polling approach (periodic pull), a system may periodically send requests to other sides to obtain data. In a publisher/subscriber system (aperiodic push), users specify the types of information they wish to receive.

Another body of relevant work is view maintenance in mobile environments [19]. When consumers access a continuously changing database from a mobile computer, two costs will usually be incurred: access cost and communication cost. To reduce these costs, the mobile user can materialize a view, called a *dynamic view*, which can be divergent from the one at the database publisher. However, our approach is different. Instead of seeing the information dissemination system as a set of materialized views replicated in mobile clients, we see information dissemination as a filtering process that conveys only that part of the data that is relevant to the mobile client.

Markup Languages for Information Dissemination

Push systems operating in heterogeneous environments need a common format to describe what information is disseminated to receivers. XML (Extensible Markup Language) is a meta-language that defines standardized conventions to represent any kind of information [20]. It was designed to provide support for meta-data on the WWW. There is an XML application proposal, named *CDF* (Channel Definition Format) [6], designed to structure information distribution through channels. Channels defined using this format are supported by Internet Explorer 4 applications. In our work, we extended CDF to accommodate new concepts added to support mobile computing specific requirements.

Software Tools

Some tools have been proposed to author CDF files without having any knowledge about the CDF format specification, such as Microsoft CDF Generator [11] and CDF Channel Maker [4]. However, we find that these tools lack a mechanism to introduce modifications interactively in the channel information model. GIL not only supports the dynamic characteristics of Ubidata's channel model but also provides an easy to use Web-based interface for Web publishers or information authors.

There are many tools on the Internet for handling XML files. In GIL we used XML Parser for Java, a package that contains classes and methods for parsing, generating, manipulating, and validating XML documents [8].

3. Dynamic Channel Model

The dissemination model of our software environment, called *Dynamic Channel Model*, is based on the notions introduced by Cheriton [5]. An information *channel*, shortly channel, is defined as an abstraction of communication and system resources necessary to distribute information to a set of users. Our application implements the publisher/subscriber model [7], in which a user receiving information is called the

subscriber and a user sending information is called the *publisher*. Publishers use channels to disseminate/distribute information to a potentially large group of subscribers. A channel may have its set of subscribers changing dynamically over time.

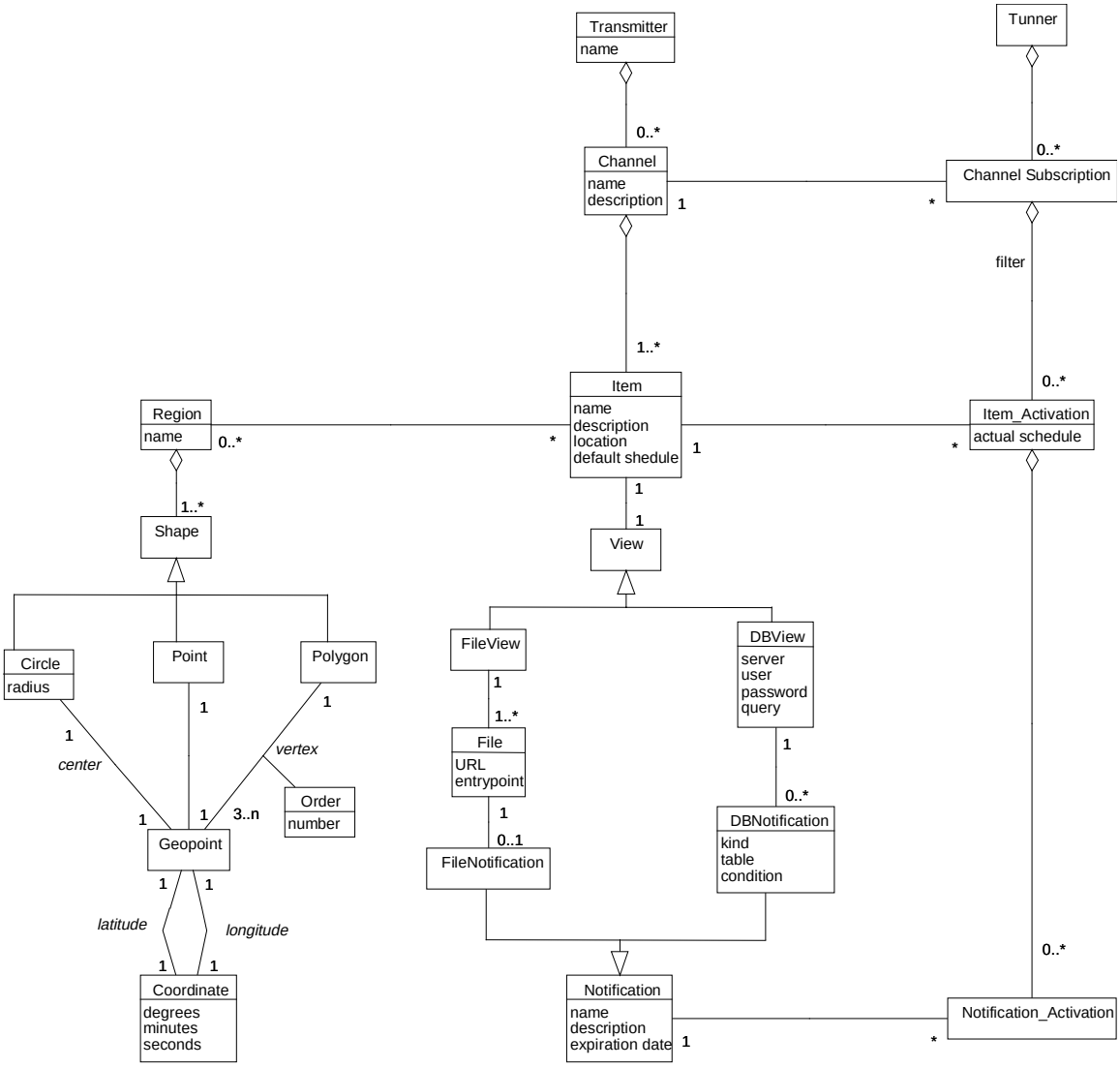


Figure 1: Class diagram of the Dynamic Channel Model.

The contents carried by a channel are grouped into logical and atomic units called *items.* An item may be a source file or a set of database objects specified as an SQL statement, in the case of a database source of

information. When a subscriber adds a channel to his reception list, and before starting receiving data on that channel, he must select and configure the items he is interested on.

Figure 1 represents the class diagram for the dynamic channel (in UML). The main classes of a channel are *transmitter* and *tuner*. The former encloses what channels and items are to be disseminated, the coverage regions and the conditions to inform the subscribers about new data available (called *notifications*). The later encloses which channels are subscribed, which items are activated, and which notifications the subscriber uses. A detailed description of the dynamic channel model is presented in [1].

4. Ubidata

In this section, we list the design objectives of our framework and present the essential stages and steps of the development of information dissemination applications based on dynamic channels. The details of the system design and the implemented architecture are described in [1]. We intended to build a flexible and general framework for mobile computing information dissemination systems. Ubidata can serve as the basis for the development of information systems that have in common some characteristics, such as the distribution of information sources to a potentially large set of users. The environment implements the main entities of the dynamic channel model presented above (publisher, network and subscriber), and has the following main functions:

- information dissemination to reachable subscribers;
- notification of changes to subscribed data;
- definition of individual schedules for each of the subscribed items;
- replication of channel items, according to the settings of the corresponding subscriptions;
- maintenance of coherency among the items data.

Development Process

In our perspective, the elaboration and construction of information dissemination applications consists in the personalization of our dissemination model and customizing the software components of Ubidata.

We organized the essential steps of the development process in a methodology that uses UML [3]. We start with the identification and characterization of the objects of the dissemination model. From this description, we create a channel configuration file for Ubidata. These configuration files, called dynamic channel description files (D-CDF) have a format that extends CDF [6]. Our extensions capture additional mobility-related information, such as location, notifications and a more complex scheme for describing replicated information, including support for passing database views. To represent these extensions, we introduce new XML elements into CDF, reflecting the semantics of the Dynamic Channel Model.

Both publisher and subscriber sides take the CDF as input. Publishers load the CDF files to know what information to disseminate, how to make it available, and when to notify subscribers about changes to the information. Subscribers load the CDF to get information about the channel items available, their relevance to the current position, and customize the notification and publisher polling schedules. The development process of Ubidata's applications is shown in Figure 2.

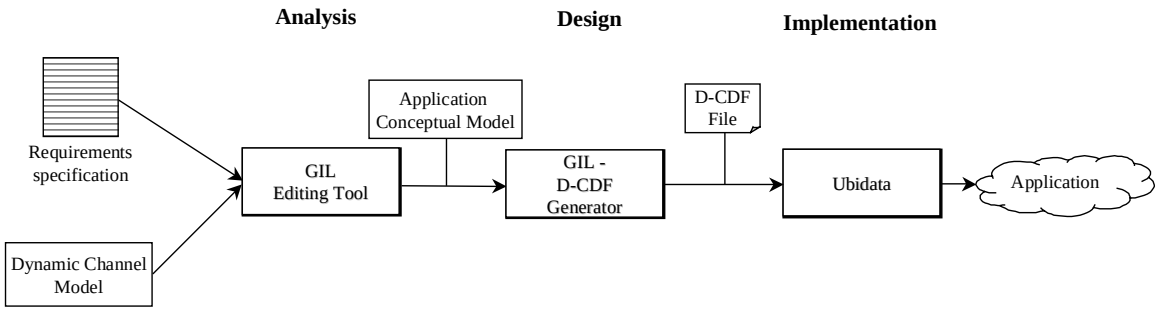


Figure 2: Ubidata application development process

5. GIL

A limitation of existing mobile computing environments is that every organization willing to use this kind of technology to disseminate information should have experts who would have to learn how to use the replication facilities. Ideally, those who define what information is to be disseminated should not be aware of the complexities of the infrastructure used to disseminate the information, so that those who create and manage the channels are experts in their application domain, and not necessarily computer experts.

GIL is a tool that guides users through the information dissemination model, allowing them to create and characterize channel representations of their Web applications, automatically configuring the Ubidata's framework to disseminate the specified contents.

GIL is part of the development process adopted for the Ubidata information dissemination process, shown in figure 3. It helps the user to identify the specifications of the applications to develop, to map those specifications on the dynamic channel model, and to automatically configure Ubidata accordingly.

GIL allows any user, even one that doesn't know how the infrastructure publishes the information, to specify what data he wants to publish, and how to publish it.

Architecture

GIL was designed to be platform independent, to maximize portability among different platforms. Whenever possible we tried to use open technologies, because these are likely to become available sooner in a greater number of platforms. GIL architecture is shown in figure 3. The data repository was implemented on a DBMS because it is easy to put data saved this way on the Internet; furthermore it is easy to write Java applets (or applications) to access the database. As we wanted to have channels information online,

accessible to multiple users, the DBMS could offer concurrent access and atomic operation on persistent data (transactions).

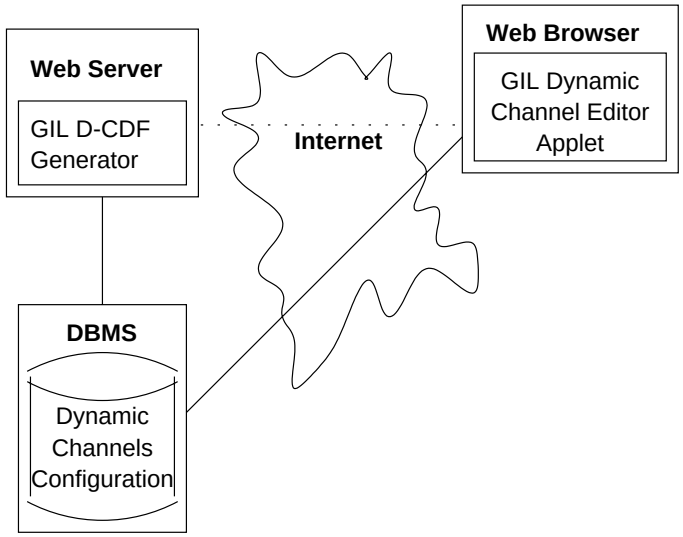


Figure 3: GIL Architecture

Design and Construction

GIL is composed of two main modules: a channel model editor and a configuration generator. To specify a channel, the user starts with the editor, going through a series of screens to enter all the required information. When all desired parameters are set, the user invokes the generation module, to create a D-CDF file with all the Ubidata needed configurations. There is no interaction with the user during the configuration generation process.

GIL's editor is a Java applet. This technology allows the creation of good interactive applications that may run across the Internet. The JDBC API has been used to access the database [9]. The user interface is based on the Java Swing API, which provide a comprehensive library of interaction objects.

The generation module is implemented as a Java servlet [2]. As this module doesn't interact directly with the user, it can run entirely on the server side and may be invoked from the Internet. We considered the

option of running it as an applet, but then it could not output results into the server file system. This module uses: the servlet API, to couple itself to the Web server; the JDBC API, to read the database; IBM's XML Parser for Java API, to build the Document Object Model (DOM) tree equivalent to the D-CDF file to generate. As the D-CDF is an application of XML, one can build a DOM representation of the D-CDF. Once the DOM tree is built, the D-CDF file is automatically created, as illustrated in the figure 4 [17].

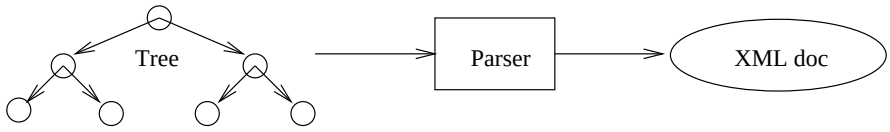


Figure 4: D-CDF Generation

The format of generated D-CDF files is defined by the DTD (Document Type Definition) shown in figure 5. In XML, the DTD defines a class of documents using a language that is a context-free grammar with several constraints on the document structure [10]. We defined our DTD as an extension of the specification of XML DTD of Microsoft's CDF [6].

```
<!ELEMENT Channel (Title, Abstract, Item+)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Abstract (#PCDATA)>
<!ELEMENT Item (Title, Abstract, View, Region, Schedule)>
<!ATTLIST Item HRef #PCDATA #REQUIRED>
<!ELEMENT View (Query?, File*)>
<!ELEMENT File (FileNotification?)>
<!ATTLIST File
   Href            #PCDATA      #REQUIRED
   Entrypoint      (YES|NO)     "NO"
   Priority        NUMBER       #REQUIRED>
<!ELEMENT FileNotification (Title, Abstract)>
<!ATTLIST FileNotification
   ExpirationDate  NMTOKEN      #IMPLIED>
<!ELEMENT Query (Select, ResultSet, DBNotification*)>
<!ELEMENT Select (#PCDATA)>
<!ATTLIST Select
   server      #PCDATA          #REQUIRED
   user        #PCDATA          #REQUIRED
   password    #PCDATA          #REQUIRED>
<!ELEMENT ResultSet (EMPTY)>
<!ATTLIST ResultSet
   HRef        #PCDATA          #REQUIRED>
<!ELEMENT DBNotification (Title, Abstract, Condition, ExpirationDate)>
<!ATTLIST DBNotification
   Kind                (insert, delete, update)      #REQUIRED
   Table               #PCDATA                       #REQUIRED
   ExpirationDate      NMTOKEN                     #IMPLIED>
<!ELEMENT Condition (#PCDATA)>
<!ELEMENT Region (Point*, Circle*, Polygon*)>
<!ELEMENT Point (EMPTY)>
<!ATTLIST Point
   x      NUMBER      #REQUIRED
   y      NUMBER      #REQUIRED>
<!ELEMENT Circle (EMPTY)>
<!ATTLIST Circle
   x          NUMBER      #REQUIRED
   y          NUMBER      #REQUIRED>
   radius     NUMBER     #REQUIRED>
<!ELEMENT Polygon (Point, Point, Point+)>
<!ELEMENT Schedule EMPTY>
<!ATTLIST Schedule
   StartDate          NMTOKEN      #IMPLIED
   StopDate           NMTOKEN      #IMPLIED
   IntervalTime       NMTOKEN      #IMPLIED>
```

Figure 5: DTD of D-CDF

A Scenario of usage of GIL

The information publisher starts with a Web page that initializes the channel editor applet. Then, the applet shows up a screen vertically divided into two distinct areas. The left side is for user navigation in the channel model being edited. The right side is for the edition of the properties of item selected on the left side. As GIL started on an empty model, a screen is presented for the creation of a new channel (see figure 6 (a)).

The user then creates a new channel to disseminate information about movies (available rooms, new movies, etc.) (figure 6 (b)). Within this channel, he can then create the items corresponding to the information to disseminate. In this example he will create one only item to disseminate the movies showing in Lisbon (figure 6 (c)).

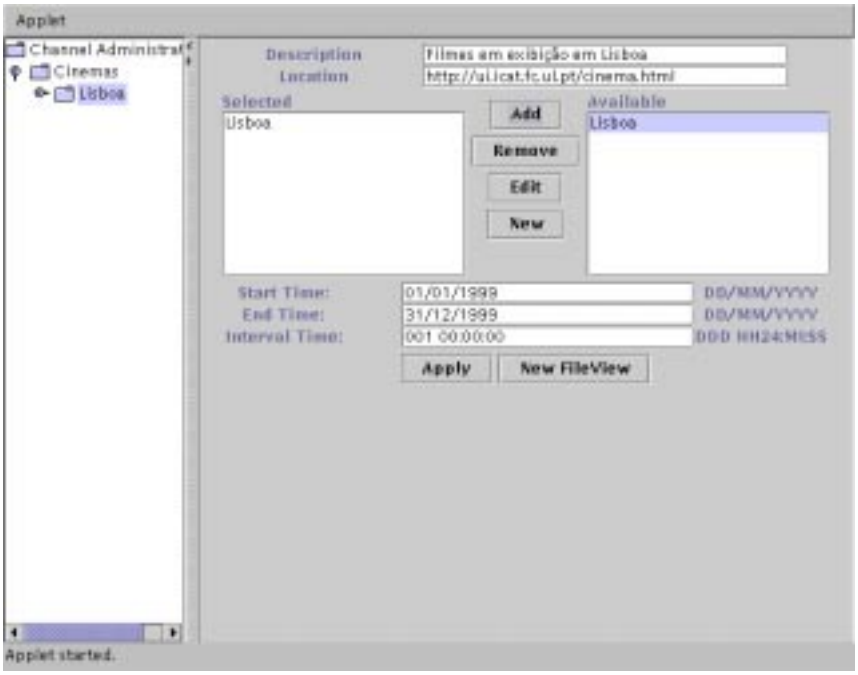
For each item, the publisher has to specify what geographic regions will receive the information. This is entered using the screen shown in figure 6 (d).

Next, the publisher defines a view on the information. In this scenario, he only defines one file view (figure 6 (e)).

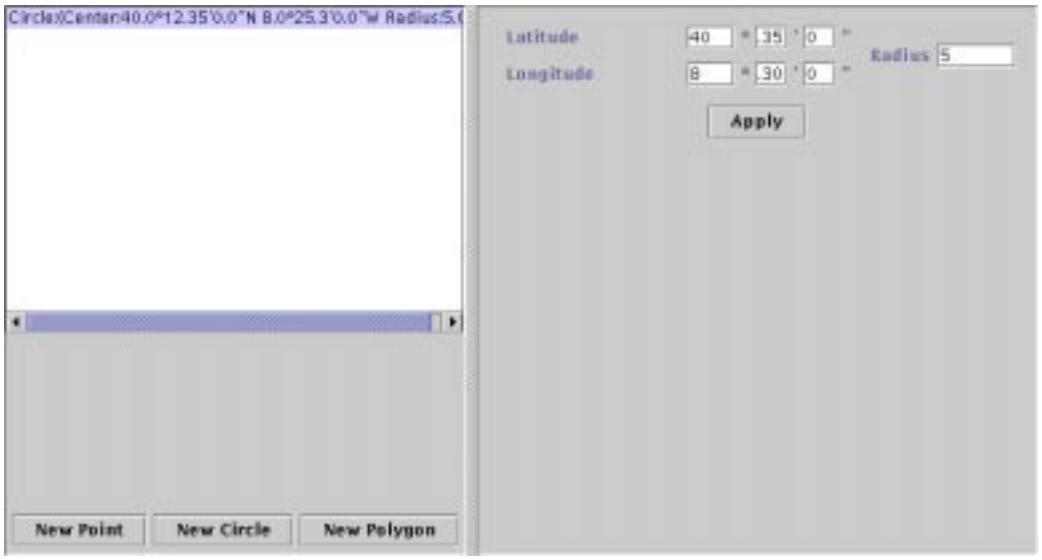
When the channel is specified, the user redirects its Web browser to the page for launching the generation module. He then creates the corresponding D-CDF file. The file is not returned to the user, but is saved in the GIL application server, to be delivered to Ubidata for reconfiguration of the infra-structure. The figure 6(f) shows the D-CDF given by the execution of this example.

(a) Channel creation

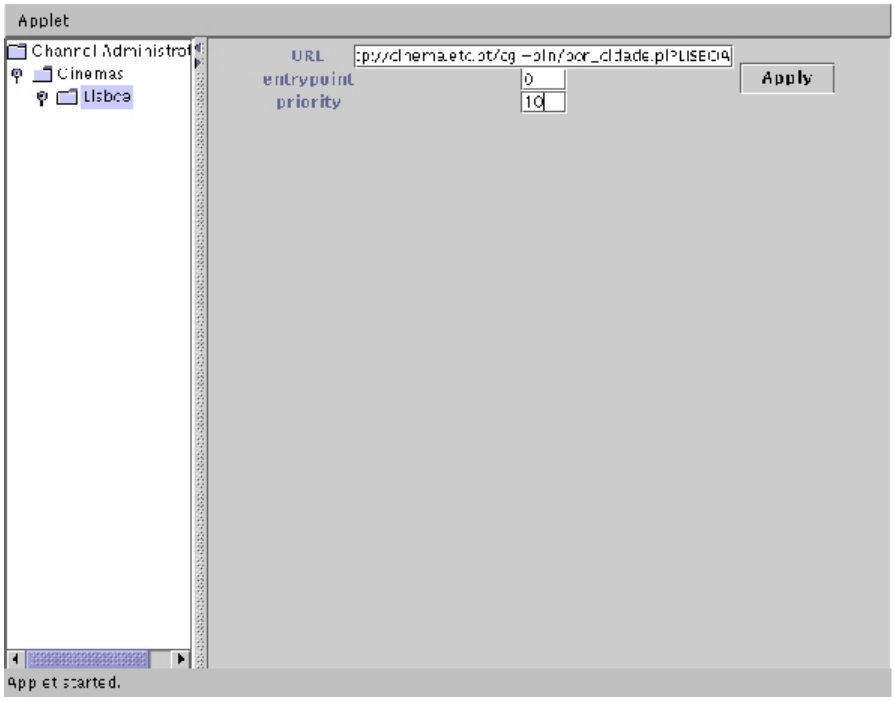
(b) Channel definition



(c) Item creation



(d) Region specification



(e) Item file view creation

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<channel>
   <title>Cinemas</title>
   <abstract>Sessões e filmes disponíveis</abstract>
   <item href="http://ui.icat.fc.ul.pt/cinema.html">
        <title>Lisboa</title>
        <abstract>Filmes em exibição em Lisboa</abstract>
        <view>
              <file href="http://cinema.etc.pt/cgi-bin/por_cidade.pl?LISBOA"
               entrypoint="1" priority="1"/>
         </view>
         <region>
             <circle   x="40.20583333333333333333333333333333333333"
                      y="8.42166666666666666666666666666666666667"
                       radius="5"/>
          </region>
          <schedule   startdate="01011999" stopdate="31121999"
                       intervaltime="001 00:00:00"/>
    </item>
</channel>
```

(f) D-CDF file

Figure 6: Channel definition and D-CDF generation in GIL

6. Conclusion and Future Work

We have finished the first prototype of GIL, which has approximately 3000 lines of code. The design and construction of GIL required a 4 persons×month effort. We intend to evaluate GIL and Ubidata usability by monitoring its use in the development of application prototypes and comparing the software metrics of these prototypes against those of identical applications built using generic software development environments. This information will be used to improve a second version of GIL.

As future work, we intend to make our architecture for the software infrastructure scale-up to a much larger number information subscribers, enabling large-scale distribution of "information appliances" that can adapt to changes in information sources and use much smaller and more usable devices, capable of connection-less communication [16]. With the new announced devices and technologies, information sources could be discovered dynamically, using RDF (W3C´s Resource Description Framework) [12, 15] and communication could be based on WAP (Wireless Application Protocol) services [18].

References

[1] A.P. Afonso, F.S. Regateiro, and M.J. Silva, Dynamic Channels: A New Methodology for Mobile Computing Applications, Technical Report DI/FCUL TR-98-4, Department of Computer Science, University of Lisbon, April 1998.

[2] C. Bloch, The Java Tutorial: Trail: Servlets. http://java.sun.com/docs/books/tutorial/servlets/index.html.

[3] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 1998.

[4] Channel Maker Homepage, http://www.anyware.co.uk/anyware/cm/.

[5] D. Cheriton, Dissemination-Oriented Communication Systems, Technical Report, Stanford University, 1992.

[6] C. Ellerman; Microsoft Corporation Channel Definition Format, http://www.w3.org/TR/NOTE-CDFsubmit.html, October, 1997

[7] M. Franklin and S. Zdonik, A Framework to Scalable Dissemination-Based Systems, *ACM OOPSLA Conference* (Invited Paper), October 1997.

[8] IBM XML Tools, http://www.alphaWorks.ibm.com/tech/.

[9] JDBC Technology, http://java.sun.com/products/jdbc/index.html.

[10] M. Johnson, XML for the absolute beginner, Javaworld, April 1999. http://www.javaworld.com/javaworld/jw-04-1999/jw-04-xml.html.

[11] Microsoft Channel Generator, http://msdn.microsoft.com/workshop/delivery/cdf/cdfgen.asp.

[12] E. Miller, An Introduction to the Resource Description Framework, *D-Lib Magazine*, May 1998. http://www.dlib.org/.

[13] E. Pitoura and G. Samaras, *Data Management for Mobile Computing*, Kluwer Academic Publishers, 1998.

[14] Push Publishing Technologies, http://www.strom.com/imc/t4a.html.

[15] RDF documentation, http://www.w3.org/RDF.

[16] M.J. Silva and A.P. Afonso, Designing Information Appliances using a Resource Replication Model, International Symposium on Handheld and Ubiquitous Computing, HUC'99, September 1999.

[17] N. Sundaresan, XML Technologies and Object-Oriented Paradigms: Tutorial Notes of ECCOP'99, 13th European Conference on Object-Oriented Programming, June 1999.

[18] Wireless Application Protocol Forum, http://www.wapforum.org/what/whitepapers.htm.

[19] O. Wolfson, P. Sistla, S. Dao, K. Narayanan, and R. Ray, View Maintenance in Mobile Computing, *ACM Special Interest Group on Management of Data*, 24(4), December 1995.

[20] XML W3C Homepage, http://www.w3.org/XML.