InfoPipes: A Flexible Framework for M-Commerce Applications

Marcus Herzog[1,2] and Georg Gottlob[1]

[1] Vienna University of Technology, Database and AI Group
Favoritenstr. 9, A-1040 Wien, Austria
[herzog, gottlob]@dbai.tuwien.ac.at
[2] Electronic Commerce Competence Center - EC3
Siebensterngasse 21, A-1070 Wien, Austria

Abstract. M-Commerce applications are E-Commerce applications having at least at one end a mobile terminal. Therefore M-Commerce applications share a number of properties with E-Commerce applications while adding additional burdens on the application developer. In this paper we present a conceptual model of an application framework that provides services at the core of M-Commerce applications. We will also present an implementation of this framework and discuss the properties of the information processing involved.

1 Introduction

The World-Wide Web technology has triggered a new business era. While the web has been invented to share scientific documents the marketing departments soon discovered this technology to communicate the latest product information to potential customers world-wide. The next generation of E-Business solutions included online transaction processing facilities for actually selling those products to customers. With the advent of mobile devices capable to tap into the information universe the installed base of terminals for E-Commerce processing quickly increased.

Although the rising number of access and transaction points is a positive development it has also some hidden drawbacks. The environment gets more complex and heterogeneous. Different operating systems, smaller memory footprints, slower communications and data synchronization, and different data input methods all come into play. The incompatibilities of first releases of assorted standards such as early versions of the WAP — Wireless Application Protocol [9] did not help matters.

We now see a massive trend towards personalized information systems, and mobile devices enable users to access portals that can be configured according to their needs. But since M-Commerce applications are even more platform-specific than conventional E-Commerce applications, systems need additional adaptors in order to integrate with legacy systems or to retrieve specific Web content. Data must be enabled to be sent to different devices without being authored for each device individually. A scalable and flexible approach uses transformation to generate the content from a single source to fulfil the formatting needs of all clients.

2 Application Scenarios

In the following we will give some application examples that will underpin the need for general architectures of M-Commerce applications that can be configured to fit the specific application needs. In our application scenarios we will concentrate on information mediation services and will spare transaction processing systems for a later presentation.

2.1 Flight Information

Flight information is only one example of all kinds of travel information services that are vital to travelers around the globe. The information domain is quite similar among those including a departure location, an arrival location, and some time and date information at its core.

Although this kind of information is usually available on the web, it is often not available on a central point for all service providers. In the case of flight information, timetables of individual flights are either scattered about different airport information systems or about the portals of individual airlines.

Moreover, as a traveller is out of home by definition, this kind of information is best communicated over mobile devices. The user has the ability to subscribe to specific flights either by providing the flight number or the departure and destination location. The system will send the actual flight status to the user, but only if the status changed between consecutive requests.

2.2 Quoting Service

Another class of typical applications for M-Commerce are all sorts of quoting services that report the price for a certain product or service. The range of tradable goods spawn from stocks to second-hand articles to computer devices. All those share the property of being for sale at a given price, although the price might be determined by different methods and the rate of price building will vary.

If we take for instance the case of an auction site the notification of new quotes for bidding items can be very well communicated over wireless devices. Moreover a bid can be acknowledged instantaneously and at any location. This service can also be easily integrated with existing web services and be offered as an add-on. The integration of a number of source services can be very interesting, especially in comparison shopping applications, where a quote for a specific product is collected from different vendors.

2.3 Tourist Information

On of the disadvantages of wireless applications is the reduced bandwidth and complexity of interaction available. On the other hand the mobility of the device can be exploited for additional request properties such as the location from which the request is made. The location property can be used to provide personalized information such as the closest facility of a certain kind.

Location-based services are best suited for tourist information systems where travelers are searching for services in a given distance to their current location. For instance locating a hotel room in a given price range would be a typical task. Integrating such a service with the actual transaction processing of booking a free resource further enhances the value for the customer.

3 Information Processing

The previous section presented some application scenarios that expose typical properties of the information processing involved. In this section we will present our information processing model for M-Commerce applications derived from these scenarios. We show how to decompose the overall task of information processing into stages that can be used as building blocks for assembling an information processing pipeline which we call InfoPipes . We will also describe the characteristics that are associated with these sub tasks. The stages are as follows:

– *acquiring* the required content from the source locations;
– *integrating* the content;
– *transforming* the content;
– and *delivering* the content.

3.1 Acquiring the Content

Content can be retrieved from a number of sources with different characteristics. The main characteristics of content feeds are:

– relationship to the originator of the content;
– transport protocol used to transfer the content;
– format of the content feeds;
– access modality;
– periodicity of update.

The relationship to content owners can be somewhere between the two extremes of cooperative and non-cooperative (we are speaking of technical cooperation, not of legal implications, which have to be dealt with separately). In the cooperative case, the content feed is fully transparent to the M-Commerce applications. This is e.g. the case in direct access to databases or to files in the file system, where the structure is known and the access rights are granted. This is often the case when the system and the content feeds are under the same control such as in EAI systems. If the M-Commerce application has to integrate content from different content providers, the interface will often be restricted to access content only from web services. The issue of the relationship actually determines most of the following characteristics.

The transport protocol of choice will be in most cases HTTP, as it is well suited for all kinds of file access, wether remote or local. The access is also nicely abstracted in the URN respectively URL concept, which defines a transport protocol along with a resource identifier. HTTP furthermore allows to tunnel through firewalls, which relieves the problem of connecting to proprietary ports, albeit dodging the concept of assigning ports to specific services to a certain extend.

The format of content ranges from structured (e.g., table rows in a database) to unstructured (e.g., plain text). Semi-structured formats tend to increase in importance with the advent of XML [5]. XML brings together the advantages of the structured and unstructured format, allowing to define the structure to a certain degree while still providing a certain flexibility [1]. XML can be both used for document-based as well as message-based communication. Although XML seems to be the future, we still have to take legacy data into account, especially the myriad of HTML formatted resources. In this case transformation of HTML formatted content into an XML representation is an important subtask.

The access modalities differs according to the transfer protocol and to the formatting of the content feed. In the best case retrieving the content is a single command (e.g., executing a SQL statement or reading a file). Interacting with a web service is more tricky including the complexity of navigating the document structure. In this case the system has to emulate the browsing behaviour of the user to acquire the content.

Finally, M-Commerce applications have to deal with the high volatility of the source content. We can distinguish between pull and push content. While the update rate of push content is under control of the supplier-side, push content is actively fetched by the receiver. The receiver has to decide weather to fetch data on demand or in advance. On demand would fetch data only it data are currently needed, while in advance would fetch data at a given rate independent of an actual request for that data. For most applications both methods have to be supported to be able to trade accuracy of the data against response time.

3.2 Integrating the Content from Various Sources

In our model acquiring data from a number of heterogeneous sources is supported in the integration stage. We assume that the input to this stage is XML based, either by native XML data sources or by translation during the acquiring stage of the processing pipeline. The purpose of the integration stage is two-fold:

– *integrate and normalize the XML element structure*; this is achieved by mapping rules between incoming and outgoing XML document types. The mapping process can be necessary when conceptually similar sources do not share a common document type, although the structure of the domain is related;

– *normalize the content of XML elements*; it is sometimes necessary to map between textual fragments and concepts, e.g., the flight status could be denoted as *delayed* in one web source and as *verspätet* in another — both terms will be mapped to a single concept that can be processed by other stages.

The output of the integration stage is a canonical data structure that can be used as a data source for the subsequently following information processing steps. The integration needs some human intervention for the specification of the various mappings being applied. It is our intention that this mapping is carried out using interactive tools. Furthermore, it is an interesting research issue to apply techniques from various fields such as statistics and information retrieval to support the user in creating those mappings for the individual source structures.

3.3 Transforming the Content According to the User Needs

The purpose of the transformation stage is to enable the application designer to define XML transformations on the content. This is needed to customize the content according to the requirements of the user who is requesting the content. This stage can be compared to defining a query in traditional databases. In fact we are currently evaluating which XML query language can be applied to fulfil this task. In contrast to the classical relational approach there is no definitive standard for query formulation in the XML field. A recent survey of XML query languages is given in [4].

While the previous two stages can be performed independent of the application user, the transformation and also the subsequent delivery of the result will be tailored according to the user configuration. The transformation is composed by the application designer and parameters are submitted by the user. This strategy is well known from so-called publish-subscribe systems [2, 6, 11]. Applying the transformation on XML data allows to generate results independent of platform specific formatting constraints. The formatting of the result is part of the delivery stage.

3.4 Delivering the Content to the User

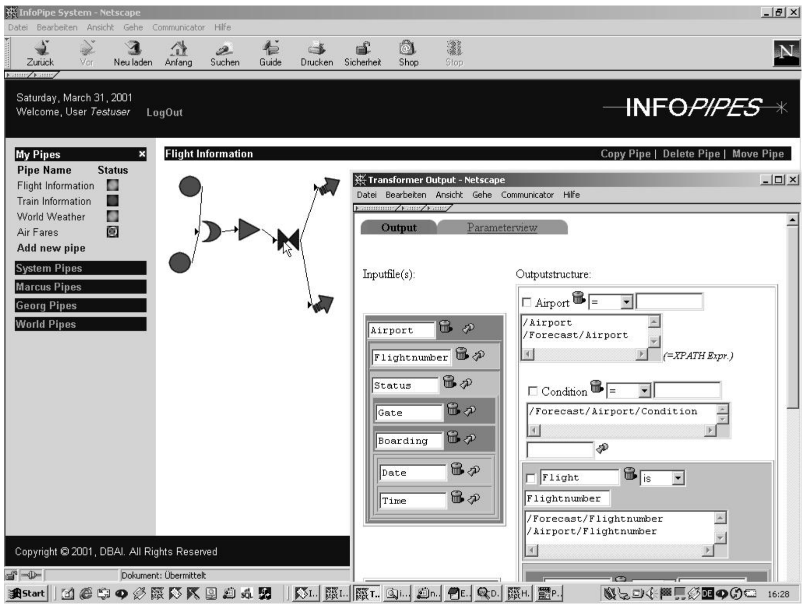
The final stage of the information processing chain is responsible for delivering the result to the target platform. In M-Commerce applications this can be a number of different clients ranging from mobile phones (usually with different capabilities) to PDAs and other mobile devices. While the basic infrastructure for E-Commerce and M-Commerce applications is pretty equivalent, M-Commerce applications have to face the multi-platform publishing challenge to a greater extend.

In principle the delivery stage can be regarded as the opposite side to the acquisition stage. During acquisition the application has to communicate with a number of different data sources by means of correct requests. In the delivery phase requests from heterogeneous clients have to be answered correctly. It is the challenge not to use only the least denominator of all features exposed by the various clients but to adapt the response according to the features of the client.

The task of formatting the information processing result has to be supported by graphical and interactive tools that are targeted at content managers in contrast to application programmers.

4 System Architecture

In the remainder of the paper we will present our approach towards an system architecture that enables application designers to rapidly build M-Commerce applications. In our system architecture the whole information processing task is split into a number of stages according to the task decomposition as presented in subsection 3. The stages are implemented as software components that can easily be combined to build full-featured information processing pipelines. The construction of these infopipes is supported by a visual tool (see figure 1). The infopipe architect creates and configures the elements of the infopipe by a number of mouse clicks and parameter entries.


InfoPipe System - Netscape
Datei Bearbeiten Ansicht Gehe Communicator Hilfe
Zurück Vor Neu laden Anfang Suchen Guide Drucken Sicherheit Shop Stop
INFOPIPES
My Pipes
System Pipes
Marcus Pipes
Georg Pipes
World Pipes
Flight Information
Copy Pipe | Delete Pipe | Move Pipe
Transformer Output - Netscape
Datei Bearbeiten Ansicht Gehe Communicator Hilfe
Output
Parameterview
Start
16:28

Saturday, March 31, 2001
Welcome, User *Testuser* LogOut

Pipe Name	Status
Flight Information	▨
Train Information	▨
World Weather	▨
Air Fares	▨
Add new pipe	

Inputfile(s):

Airport
Flightnumber
Status
Gate
Boarding
Date
Time

Outputstructure:

☐ Airport =
/Airport
/Forecast/Airport
(=XPATH Expr.)

☐ Condition =
/Forecast/Airport/Condition

☐ Flight is
Flightnumber
/Forecast/Flightnumber
/Airport/Flightnumber

Copyright © 2001, DBAI. All Rights Reserved

Dokument: Übermittelt

Fig. 1. Main infopipe screen and transformer configuration screen

The actual data flow within the infopipes is realized by handing over XML documents. Each stage within the infopipe accepts XML documents (except for the source, which accepts HTML documents), performs its specific task, and produces an XML document as result. This result is fed to the successor components which in turn will perform the next information processing stages. Components which are not on the boundaries of the network are only activated by their neighbouring components. Boundary components (i.e., source and deliverer components) have the ability to activate themselves according to a user specified strategy and trigger the information processing on behalf of the user. Figure 2 shows an infopipe topology representing the information processing according to the flight information scenario.

Two source components to the left wrap the airport web sites in Frankfurt and Vienna. The output of these source components are data on flight departures in XML format. These structurally differing outputs are fed into an integrator component which normalizes the data. The normalized data are transformed to select flight data based on the flight ID or the departure and destination attributes respectively, which are supplied by users of this information pipe. The deliverer components transform the XML output of the transformer components into formats appropriate for the target devices.

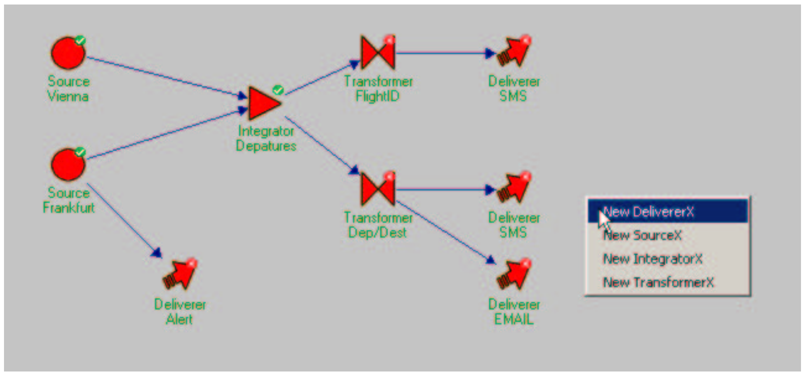


Fig. 2. An example of an infopipe topology

The topology of an infopipe is determined by the application designer. Once activated the information flow within the infopipe is request driven. In our architecture the activation is spreading from the sinks towards the sources. All stages use local caches to cache intermediate results that can be reused by other connected components. A checksum computed over the input XML document and the configuration of the component is applied to detect whether the cache is still valid.

In the following we will discuss the tasks that are carried out at the different stages of the infopipe information processing model.

4.1 Source

The source component is responsible for retrieving the data from the data sources. Data sources are any kind of machine readable data store that features an access protocol to fetch data over the network. Source components abstract the individual data stores and provide a unified interface for other infopipe components. In a first step the main focus of our InfoPipes system is on web sources. The task is to emulate the browsing behaviour of a user to retrieve the data from the web by utilizing a crawler engine [10]. The source component implements the HTTP, respectively HTTPS protocol to communicate with web sources. Moreover, it supports Cookies to handle client-side state management.

The source component is configured by the user through observing the interaction of the human user with the web source. The result of the source configuration is a list of HTTP method calls and associated arguments. The aim of the navigation is to locate a web page that holds the information the user is interested in. Once the user arrives at this page, the HTML code is handed over to the extraction component. In the InfoPipes system we use a novel approach for generating HTML extractor programs. Details on the extraction process are given in [3].

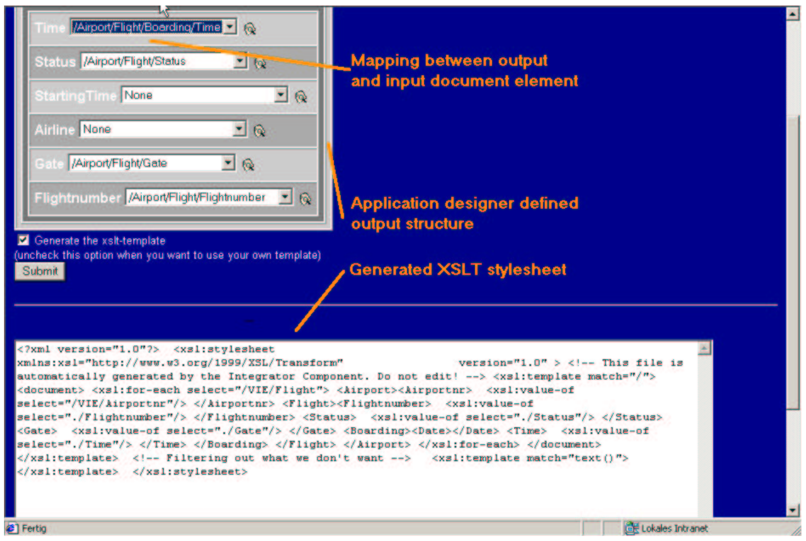
4.2 Integration

The source and extraction components are responsible for retrieving web data and translating it into an initial XML representation. The integration component is responsible for integrating the various fragments that are passed along from source components into a unified XML representation. The integrator will integrate data from information sources in the same domain that share a similar data structure, e.g., data that are retrieved from various airport flight information systems. The incoming data will be mapped to a unified output structure that incorporates all necessary XML elements.

The integration process is also configured via a graphical user interface (see figure 3). The user sees the document type structure of all incoming XML documents. After defining the document type of the output document, the individual incoming document types are mapped to the output structure. Furthermore regular expressions can be defined to perform the concept matching as described in the previous paragraph.

The result of the configuration are XSLT [8] programs that perform the necessary translations between the input documents and the output document. Again the user is not required to know the mechanism of XSLT but can take advantage of the full-fledged capabilities if the automatically generated XSLT does not fully satisfy the users' needs.

The result of the integration component is a normalized view on all integrated web sources that will serve as a data source for the following components in the pipeline. Due to the implicit cache architecture in the InfoPipes network only those documents have to be reintegrated that changed in between successive request.


Mapping between output
and input document element
Application designer defined
output structure
Generated XSLT stylesheet
Submit
Fertig
Lokales Intranet

Time	/Airport/Flight/Boarding/Time
Status	/Airport/Flight/Status
StartingTime	None
Airline	None
Gate	/Airport/Flight/Gate
Flightnumber	/Airport/Flight/Flightnumber

[✓] Generate the xslt-template
(uncheck this option when you want to use your own template)

```
<?xml version="1.0"?>  <xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"                version="1.0" > <!-- This file is
automatically generated by the Integrator Component. Do not edit! --> <xsl:template match="/">
<document> <xsl:for-each select="/VIE/Flight"> <Airport><Airportnr>  <xsl:value-of
select="/VIE/Airportnr"/> </Airportnr> <Flight><Flightnumber>  <xsl:value-of
select="./Flightnumber"/> </Flightnumber> <Status>  <xsl:value-of select="./Status"/> </Status>
<Gate>  <xsl:value-of select="./Gate"/> </Gate> <Boarding><Date></Date> <Time>  <xsl:value-of
select="./Time"/> </Time> </Boarding> </Flight> </Airport> </xsl:for-each> </document>
</xsl:template>  <!-- Filtering out what we don't want -->   <xsl:template match="text()">
</xsl:template>  </xsl:stylesheet>
```

Fig. 3. Integrator configuration dialog

4.3 Transformation

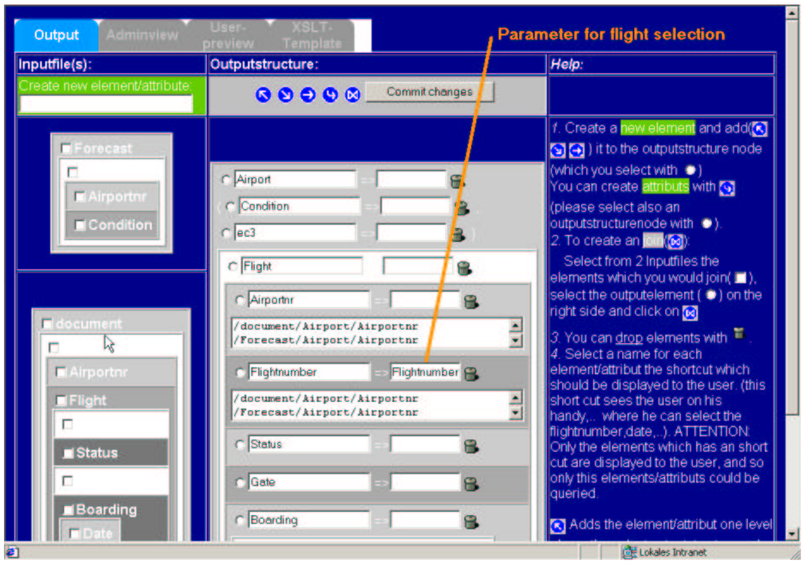
In addition to the expressive power of the query language we have to evaluate how queries can be formulated by graphical means to fit into the overall strategy of a fully visual interface. An example of a visual query language for XML documents is given in [7]. The first impression is that visual representations of queries tend to get rather confusing when complexity is increased. This leads to the assumption that we have to trade complexity for usability. An alternative is to allow the experienced user to formulate queries in the native query language while lay users use the visual interface with reduced expressive power.

Besides query formulation the transformation component will also generate query masks according to the formulated queries where users can submit actual values to arguments in the query. It is important to note that these masks are available for different clients on different platforms, e.g., for HTML and WML clients. These masks are used during the information processing when users can only state their information needs according to predefined transformations.

Figure 4 shows a screenshot of the transformer configuration dialog.

4.4 Delivery

The final stage in the processing pipeline is to deliver the information to the user. An important design principle of the InfoPipes architecture has been to support clients on various platforms. This implies a multi-platform publishing supporting both *push* and *pull* technology. The responsibility of the delivery component is to:



```
/document/Airport/Airportnr
/Forecast/Airport/Airportnr
```

1. Create a new element and add( ) it to the outputstructure node (which you select with ●)
You can create attributs with (please select also an outputstructurenode with ●).
2. To create an join( ):
Select from 2 Inputfiles the elements which you would join( ), select the outputelement (●) on the right side and click on
3. You can drop elements with .
4. Select a name for each element/attribut the shortcut which should be displayed to the user. (this short cut sees the user on his handy,.. where he can select the flightnumber,date,..). ATTENTION: Only the elements which has an short cut are displayed to the user, and so only this elements/attributs could be queried.

Adds the element/attribut one level

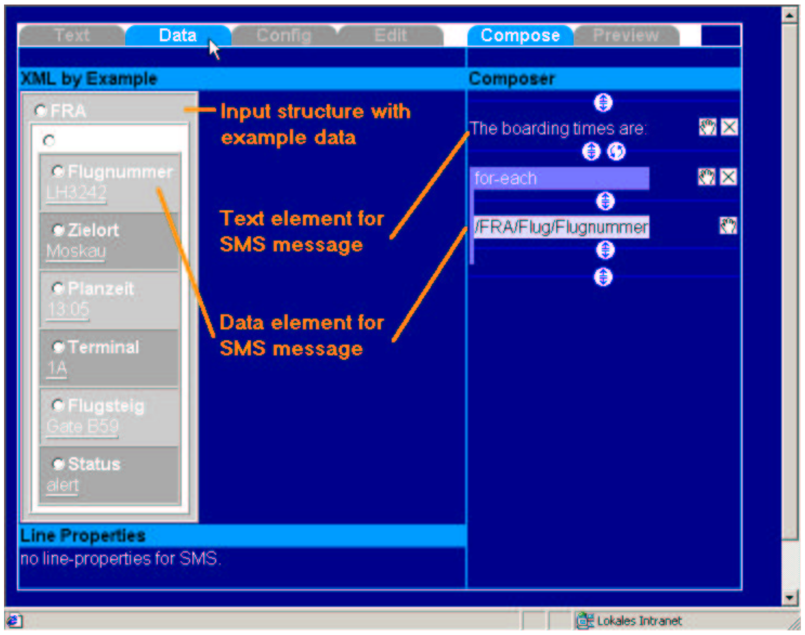
Fig. 4. Transformer configuration dialog

– check on delivery constraints such as time, date, or if the information content has changed since the last delivery;
– transform the information from XML into a format suitable for the requesting client, ranging from simple text formats (e.g., e-mail, SMS) to markup languages (e.g., HTML, WML, or VoxML);
– deliver the content to the client utilizing the appropriate communication channel and protocol.

The delivery component is on the boundary of the network structure and has the ability to activate itself and trigger the processing of the pipeline. The processing takes place through backward chaining to all immediate successor components which in turn activate their successor components. The user can configure at what time or in what intervals the component should activate the pipeline processing. On the other hand the delivery component gets activated if one of the source components in the same pipeline changes its state.

In the case of activation by source components checking delivery constraints is important because otherwise the infopipe would permanently push information to users even when they are not interested in it. Once the user has subscribed to the pipeline and the processing is started the infopipe continuously produces information output. On the other hand activation by delivery component is interesting for infopipes with "lazy" source components which change only infrequent but the user wants to be informed on a specific point in time.

The transformation process is visually configured by selecting elements from the DTD of the input document and assembling an output DTD in case of markup documents or assembling a text document in case of text based output formats. The configuration is then translated into an XSLT program that is executed at run time. Similar to the transformer component the infopipe architect can replace this program with a hand-coded one in case of more complex transformation needs. See figure 5 for a screenshot of the deliverer configuration dialog.



The boarding times are:

for-each

/FRA/Flug/Flugnummer

no line-properties for SMS.

Fig. 5. Deliverer configuration dialog

5 Conclusions

We have presented our novel approach towards an architecture for personalized information channels that feed content extracted from semi-structured information sources to clients on various platforms. The main focus of this paper has been on the presentation of the whole system architecture from a logical point of view and to clearly structure the problem domain. Some of the technical details had to be omitted due to the restricted scope of the presentation.

The InfoPipes system is currently under construction and is designed as a web-based application. The implementation language is Java and we utilize the J2EE infrastructure to translate the logical components into software components. The project is carried out in cooperation with the Electronic Commerce

Competence Center in Vienna, where we realize the airport scenario using the infopipe infrastructure. The prototype has received high attention from the mobile telecom companies that are partners of this center.

6 Acknowledgement

This work was partly supported by the Austrian Science Fund Project NZ29-INF and by the Electronic Commerce Competence Center - EC3, Austria.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann, 1999.

[2] Guruduth Banavar, Tushar Deepak Chandra, Bodhi Mukherjee, Jay Nagarajarao, Robert E. Strom, and Daniel C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Proc. of International Conference on Distributed Computing Systems*, pages 262–272, 1999.

[3] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with Lixto. In *Procs. of 27th International Conference on Very Large Data Bases (VLDB)*, Roma, Italy, 2001. to appear.

[4] Angela Bonifati and Stefano Ceri. Comparative analysis of five XML query languages. *SIGMOD Record*, 29(1):68–79, 2000.

[5] T. Bray, J. Paoli, C. Sperberg-MacQueen, and E. Maler. Extensible Markup Language (xml) 1.0 (Second Edition), 2000.

[6] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proc. of Symposium on Principles of Distributed Computing*, pages 219–227, 2000.

[7] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: A graphical language for querying and restructuring xml documents. In *Proc. of 8th Int. World Wide Web Conference*, pages 1171–1187, 1999.

[8] J. Clark (ed.). XSL Transformations (XSLT) Version 1.0, November 1999.

[9] WAP Forum. Wireless application protocol.

[10] Allan Heydon and Marc Najork. Mercator: A Scalable, Extensible Web Crawler. *World Wide Web*, 2(4):219–229, December 1999.

[11] Radu Preotiuc-Pietro, Joao Pereira, Francois LLirbat, Francoise Fabret, Kenneth Ross, and Dennis Shasha. Publish/subscribe on the web at extreme speed. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Cairo, Egypt, 2000.