Approach for an integrated interoperable system architecture for disaster management systems

Christian Paulus
Karlsruhe Institute of Technology
christian.paulus@kit.edu

Stefan Möllmann
Karlsruhe Institute of Technology
stefan.moellmann@kit.edu

Hagen Engelmann
Karlsruhe Institute of Technology
hagen.engelmann@kit.edu

ABSTRACT

In the field of information systems for disaster management there is a large variety of data formats, specifications and standards. Most of these standards only cover a specific part of this area, for example formats for geospatial data or message exchange. This diversity of isolated solutions, however, prevents those systems from interacting and exchanging data. To improve the interoperability in this sector there is a strong need for an integrated interoperable system architecture that is suitable for stand-alone systems as well as for the communication in a distributed heterogeneous system environment.

This paper shows an approach for such a system architecture. It presents the **D**isaster **M**anagement **M**arkup **L**anguage (DMML), which provides an architecture of data structures, services and service interfaces for crisis response systems. Furthermore, the **D**isaster **M**anagement **I**nteroperability **F**ramework (DMIF) is introduced, which supplies a software-engineering layout for DMML.

Finally, the implementation of the DMMapML module is presented, which handles data involved in the situation report. The basic structure of this implementation is described and its potential contribution to the interoperability of crisis response systems.

Keywords

Interoperability, disaster management systems, OGC, OASIS, GML, GeoTools, GeoAPI, JTS

INTRODUCTION

The management of natural disasters requires the ability to keep track of a huge amount of information such as the status of relief units, casualties, streets and buildings. To facilitate this task a considerable number of information systems is available. These diverse systems, which follow different approaches to support the manager of a disaster situation, can be classified into the following categories: management information systems, simulation systems, decision support systems and combinations of the three. A management information system focuses on providing a clearly arranged view on all relevant information. A simulation system is capable of simulating events and can for instance be applied in staff training. Finally, a decision support system assists the user in making decisions, e. g. by automatically detecting the closest relief unit to a given location.

As development in this field progresses the interoperability of these systems will gain importance, since it is reasonable to combine different systems in order to increase their benefit. For example a decision support system could be backed by a simulation system that simulates the consequences of a decision before it is executed. Likewise, both decision support and simulation systems need to be able to communicate with the management information system so as to display their contents to the users and to receive their inputs. Moreover, in large crisis situations the data exchange between various government agencies and relief organizations is of utmost importance to coordinate their work.

Unfortunately, many crisis response systems support different formats and standards, which complicates the data exchange. This paper presents an approach for an integrated, interoperable architecture to ease the consolidation of existing systems and to provide a software-development foundation for new systems.

DISASTER MANAGEMENT MARKUP LANGUAGE (DMML) AND DISASTER MANAGEMENT INTEROPERABILITY FRAMEWORK (DMIF)

First of all, there is interoperability on the “conceptual” level and on the “software/application related” level. At the beginning of the “conceptual” level a verification and analysis is made concerning data for such systems. The term “data” includes the identification, determination, description, structuring, handling, storage, exchange and presentation of data. Furthermore relevant standards which exist and are under research are verified and analysed. Standards which were taken into account, were for example the Geography Markup Language (GML) (Portele, 2007) of the Open Geospatial Consortium (OGC) for geospatial data and the Emergency Data Exchange Language (EDXL) (Aymond, Brooks, Grapes, Ham, Iannella, Robinson, Joerg and Triglia, 2008; Raymond, Webb and Aymond, 2006) as well as the Common Alerting Protocol (CAP) (Jones, 2007) of the Organization for the Advancement of Structured Information Standards (OASIS) for the exchange of messages and information. The development is based on results of two sub-projects of the preceding project "German Collaborative Research Center (CRC) 461". Their focus, however, was the development of simulation and decision support for earthquakes rather than interoperability. In these projects for example, the DMT-IXS format for the exchange of messages between instances of the management system (MIS) as well as for information exchange between management and simulation components was designed (Werder, 2007; Engelmann and Fiedrich, 2009). In the current research project “Model based resource management for flood water events and interoperability of involved components” at the Karlsruhe Institute of Technology (KIT), a similar system is under development for flood water with an interoperable system architecture.

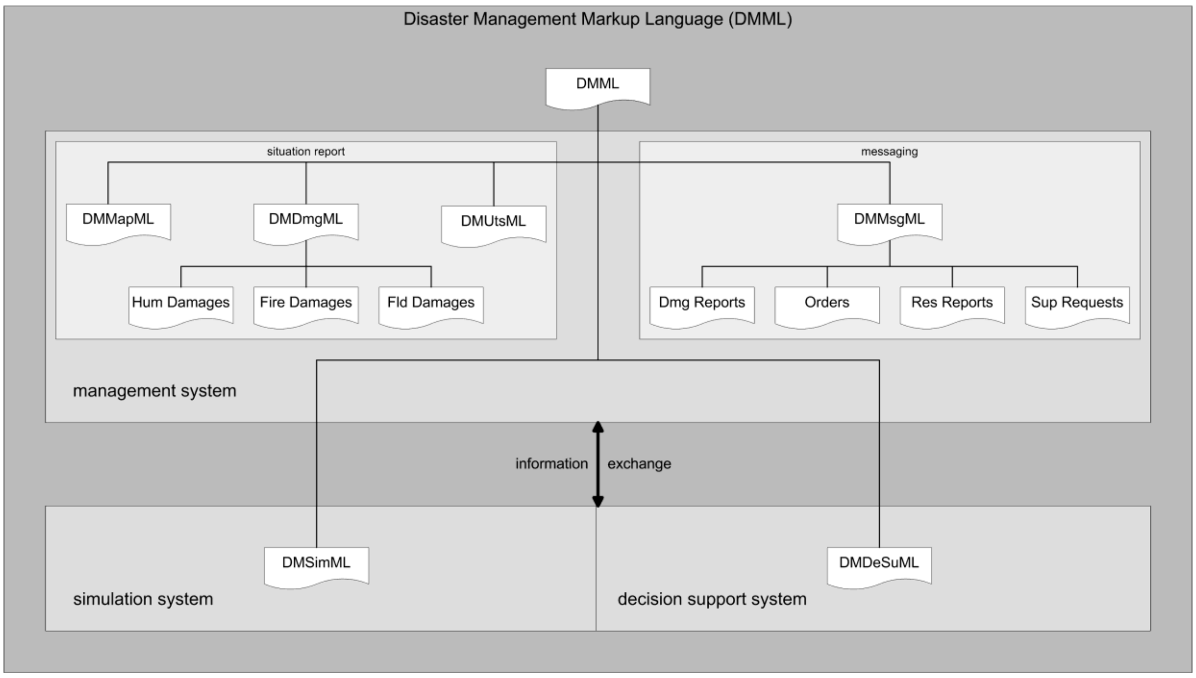

Figure 1. The Disaster Management Markup Language (DMML)

As shown in figure 1 the system world is divided into a management, a simulation and a decision support system. To define the interoperability architecture, the necessary information objects were identified and specified for each of those systems. The main components of the management system are situation report and messaging. A situation report includes three components: digital map, damages and units (see Figure 1). The digital map so far contains the disaster area, buildings and streets, whereas further objects can be added later. The damages in damage report messages are classified into human, fire and flood damages and others. Units, for instance, describe the resources in terms of relief forces. The messages are grouped into damage reports, orders, resource reports and support requests. The units exchange messages to communicate. There is a variety of relationships between the classified and defined objects of the system world. For instance, units and damaged objects have a relation to a digital map according to their positions. For example, if a building was damaged by a flood the corresponding damage object contains a reference to this building. All this information is included in the situation report. The objects classified and defined by the management systems can be connected to simulation and decision support systems. This provides various opportunities, e.g. training possibilities for relief forces, gathering of knowledge on the potential development of given disasters and scenarios as well as operation suggestions for emergencies based on the simulated development of a disaster.

```
<dmmapml:Building id="4711">
    <dmmapml:yearOfConstruction>1970</dmmapml:yearOfConstruction>
    <dmmapml:height unit="metres">24</dmmapml:height>
    <dmmapml:roofType>flat roof</dmmapml:roofType>
    <dmmapml:expectedNumberOfPersons timeSpecification="UTC" localTime="UTC+1">
        <dmmapml:expectedValue startTime="00:00" endTime="07:59">17</dmmapml:expectedValue>
        <dmmapml:expectedValue startTime="08:00" endTime="15:59">513</dmmapml:expectedValue>
        <dmmapml:expectedValue startTime="16:00" endTime="23:59">440</dmmapml:expectedValue>
    </dmmapml:expectedNumberOfPersons>
    <dmmapml:valueOfBuilding currency="Euro">4735000</dmmapml:valueOfBuilding>
    <dmmapml:damageState>
        <dmmapml:hazusDamageState hazusTypeCode="slight" hazusModelBuildingType="RM2">
        <!-- textual damage description --> </dmmapml:hazusDamageState>
    </dmmapml:damageState>
    <dmmapml:constructionClass>
        <dmmapml:hazusConstructionClass hazusTypeCode="C3M" hazusRange="Mid-Rise">
        <!-- textual construction description --> </dmmapml:hazusConstructionClass>
    </dmmapml:constructionClass>
    <dmmapml:occupancyClassification>
        <dmmapml:hazusOccupancyClassification>
            <dmmapml:mainClassification hazusTypeCode="COM">Commercial</dmmapml:mainClassification>
            <dmmapml:storeyClassification> <!-- ... --> </dmmapml:storeyClassification>
        </dmmapml:hazusOccupancyClassification>
    </dmmapml:occupancyClassification>
    <dmmapml:geometry>
        <gml:Polygon gml:id="Polygon_276930">
            <gml:exterior>
                <gml:LinearRing gml:id="LinearRing_276930.1">
                    <gml:posList srsDimension="2"> 429385.44 4920815.68 <!-- ... --> </gml:posList>
                </gml:LinearRing>
            </gml:exterior>
        </gml:Polygon>
    </dmmapml:geometry>
</dmmapml:Building>
```

Figure 2. XML instance of DMMapML

To model the system architecture the XML Schema Definition (XSD) (Fallside and Walmsley, 2004) for data structure and the Extensible Markup Language (XML) (Bray, Paoli, Sperberg-McQueen, Maler and Yergeau, 2008) for data instances were used as basic technologies. XSD and XML can be utilised to describe data in human readable text, define hierarchical data structures and describe relations and properties of objects and elements. They are widely spread, accepted and used standards of the World Wide Web Consortium (W3C).

The layout of the interoperable system architecture (for management, simulation and decision support systems) is shown in Figure 1. The **D**isaster **M**anagement **M**arkup **L**anguage (DMML) defines the main language and contains different sub languages for the specific systems. To describe the situation report, DMMapML for the digital map, DMDmgML for the damages (dmg) and DMUtsML for the Units (uts) are used for management systems. For example damages can be classified into human, fire or flood damages. A possible XML instance of DMMapML containing building data can be seen in Figure 2. The idea is to combine spatial information with information relevant for disaster management as for example the expected number of persons in a building at a specific time. The language DMMsgML is used to exchange messages (msg) (Figure 1). Damage reports, orders, resource reports and support requests are available as message types (Werder, 2007). The simulation and decision support system (if existing) communicates and exchanges information with the management system. In this context the language DMSimML is used for the simulation (sim) and DMDeSuML is used for the decision support (desu) to control and exchange information. Web services and web service interfaces are to be designed and implemented to allow the exchange of data and information with external systems. These could be used to load data of a specific disaster area. In this case the service interface interacts with a service which in turn loads the requested data from a (geo) database (e. g. PostGIS) and sends it to the interface using the appropriate format (DMMapML in this example).

For the "software/application related" level the **D**isaster **M**anagement **I**nteroperability **F**ramework (DMIF) is in development. DMIF is a reference implementation for DMML support, like for example reading and writing of DMML. DMIF is based on free, open source frameworks and programming libraries which in turn implement relevant standards and guidelines. The framework with its modules is shown in Figure 3.

For the practical usage of the research results, the DMIF framework will be integrated into the existing earthquake system and the flood water system, which is currently under development. This will serve as an example to show how systems can make use of the framework to support the development of their architecture and interoperability.

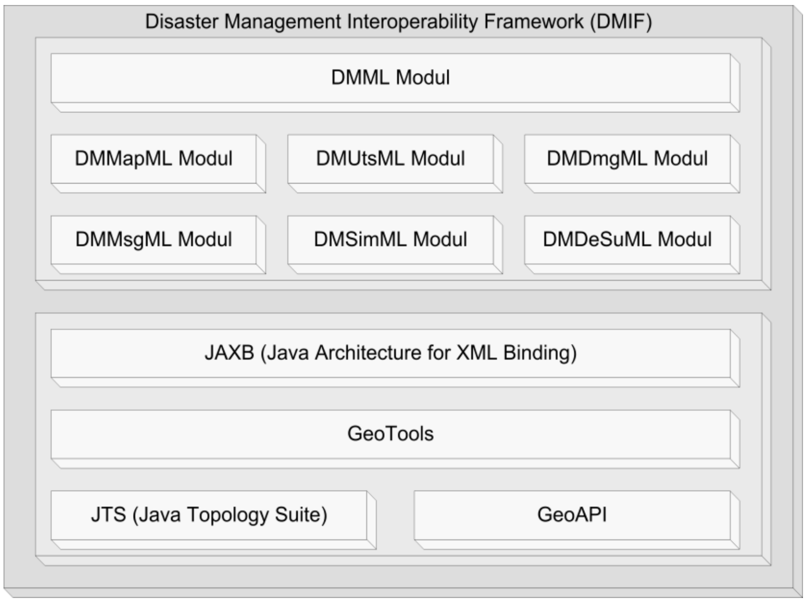
Disaster Management Interoperability Framework (DMIF)
DMML Modul
DMMapML Modul
DMUtsML Modul
DMDmgML Modul
DMMsgML Modul
DMSimML Modul
DMDeSuML Modul
JAXB (Java Architecture for XML Binding)
GeoTools
JTS (Java Topology Suite)
GeoAPI

Figure 3. The Disaster Management Interoperability Framework (DMIF)

DMMAPML FRAMEWORK

In the context of a diploma thesis a sub-framework was developed in Java to implement the DMMapML module which defines the structure and the possible elements of a digital map. The main goal of the framework is to provide functions to aggregate all necessary data for a digital map into one DMMapML file. As this data could be delivered in various formats, the framework offers methods to convert those formats into DMMapML as well as methods to manipulate this data. One crucial aspect which had to be considered in the design of the framework was the fact that many file formats exist which might need to be converted in the future. This calls for a design that can easily be extended by converters for further formats. Moreover, users of the framework should be able to add data manipulation functions which can be utilised for all data processed by the framework, regardless of the original format.

In order to increase the acceptance of the new format, great importance was given to complying with existing standards. The most important standard that was chosen as a basis for DMMapML is the OGC standard GML. GML specifies XML encodings of some of the conceptual classes defined by both the ISO 19100 series of International Standards and the OpenGIS Abstract Specification (Portele, 2007). To ensure the compatibility with those ISO and OpenGIS standards the GeoAPI (GeoAPI, 2010) was used as it interprets and adapts those specifications in the form of Java interfaces. The intention to provide data manipulation functions for geospatial or geometrical data led to the usage of the Java Topology Suite (JTS, 2010) which again conforms to an OpenGIS standard, in this case the Simple Features Specification (Herring, 2006). JTS offers fundamental 2D spatial algorithms such as the union or intersection of polygon surfaces. In addition to this, use was made of the GeoTools (GeoTools, 2010a) framework, which is listed as one of the known implementations of the GeoAPI and uses the geometry model of JTS. Furthermore, GeoTools supports the well established ESRI Shapefile format plus a parsing and encoding technology for GML that makes it the perfect foundation for the framework.

The philosophy of the DMMapML framework is to offer easy-to-use functions to read, write and convert data via one central class, the *DMMapMLManager*. That means when invoking a function – e. g. *getGeoDataModelFromFile* to read a file – the actual file format does not need to be taken into account as long as it is supported. As shown in Figure 4, the manager detects the file format by the file suffix (*determineFileType*) and then passes the invocation of the function to the subclass of *AbstractGeoFileAccess,* which is specialized for the determined file format. In order to extend the framework by further formats, classes need to be added that inherit *AbstractGeoFileAccess* (or *AbstractWFSAccess* in case of a web feature service). All implementations of these abstract classes need to override a read function that converts the content of a file into an instance of *GeoDataModel* which follows the DMMapML format. Depending on the intended usage, a write function that writes a *GeoDataModel* back into the original format might also be necessary. This way the framework internally only works with *GeoDataModels*. This makes sure that all data manipulation functions which are developed for *GeoDataModels* are automatically usable for all formats supported by the framework.

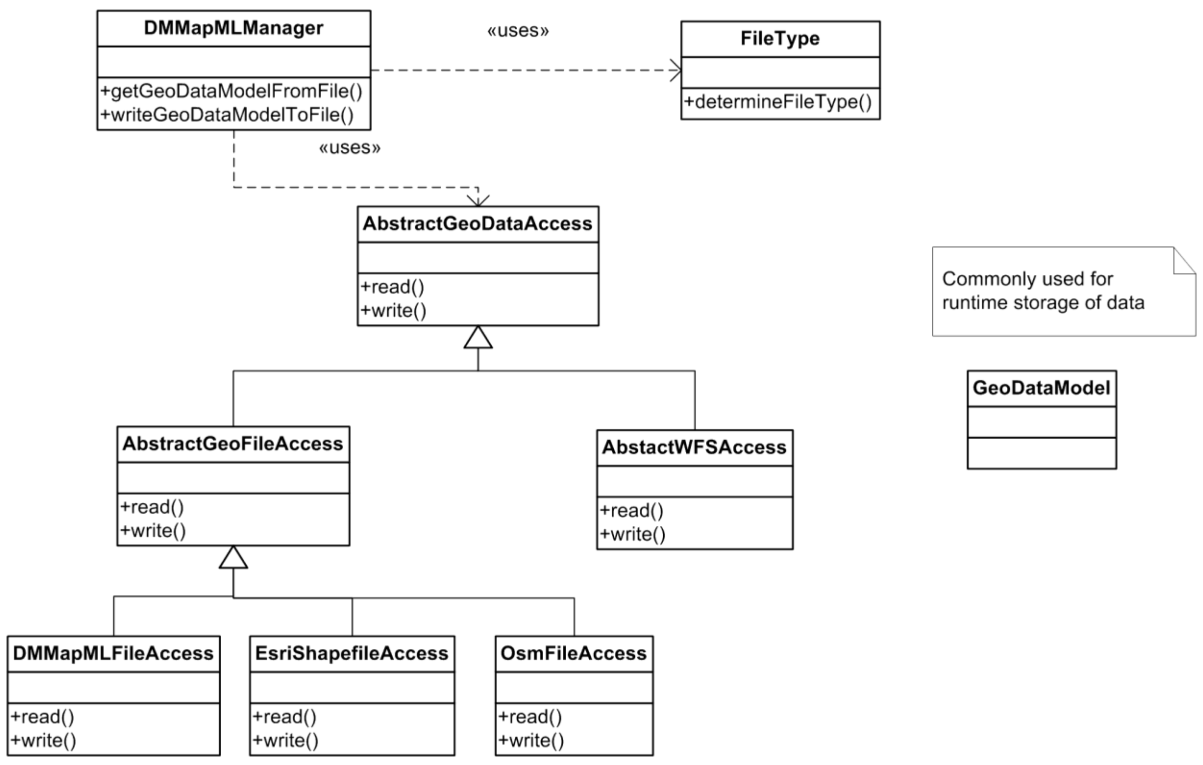

Figure 4: Simplified class structure of the DMMapML framework

In the development of the read and write functions for DMMapML a number of special properties of the GeoTools parser/encoder technology had to be dealt with. When the GeoTools parse function parses a certain element of an XML document it needs what is called a binding class to determine how to put the data of the element into a Java object. For example, if the following element was parsed: *<streetname>Main Street</streetname>* the parser would search its configuration for the appropriate binding class for the element 'streetname' and might find a class called '*streetnameBinding*'. The parse method of this class would then create an instance of a class called '*streetname*' and assign the value 'Main Street' to its attribute which would be called 'name'. This *streetname* object containing the value 'Main Street' extracted from the XML file would then be returned to the parser. The parser would process the entire input file accordingly and thereby create a complex structure of objects that matches the structure of the XML document. For an exact description of both the parsing and the encoding process please refer to (GeoTools, 2010b). The bottom line is that on the one hand this technology allows the developers to exactly define how the contents of an XML file are to be mapped onto Java objects. On the other hand the system needs one binding class for each XML element.

As it was clear that DMMapML is a highly complex XML schema that is likely to undergo frequent changes throughout the development process, the painstaking and error-prone procedure of writing binding classes for all elements of this schema was clearly not an option. This led to the idea of the previously mentioned class '*GeoDataModel*' which can contain any kind of data extracted from an XML file (as well as a whole tree of elements from such a file). In addition to this a '*SuperBinding*' was created that generically maps all given data to the *GeoDataModel* (Figure 5). As the GeoTools parser expects a different binding class for every element, a code generator was created to automatically write binding classes for all elements of an XML schema. All these bindings, however, just extend the *SuperBinding* without adding any functionality to it.

To be concise, it was refrained from describing the encoding process which writes a GeoDataModel into an XML or DMMapML file. That process follows the same principles, only encoding instead of parsing.

Apart from DMMapML, the framework so far supports ESRI shapefiles as well as the OpenStreetMap (OpenStreetMap, 2010) XML format for which it provides an option to automatically discard the building data and thus solely leaving the street data.

As an example of data manipulation functions a method called *mergeGeoDataModels* was implemented. Downloaded OpenStreetMap street data and government-provided building data of an arbitrary town, the latter in the form of an ESRI shapefile, can be read into *GeoDataModels* by the DMMapML framework. Using the *mergeGeoDataModels* function these different models can be merged into a single *GeoDataModel*. This model can then be written into one DMMapML file.

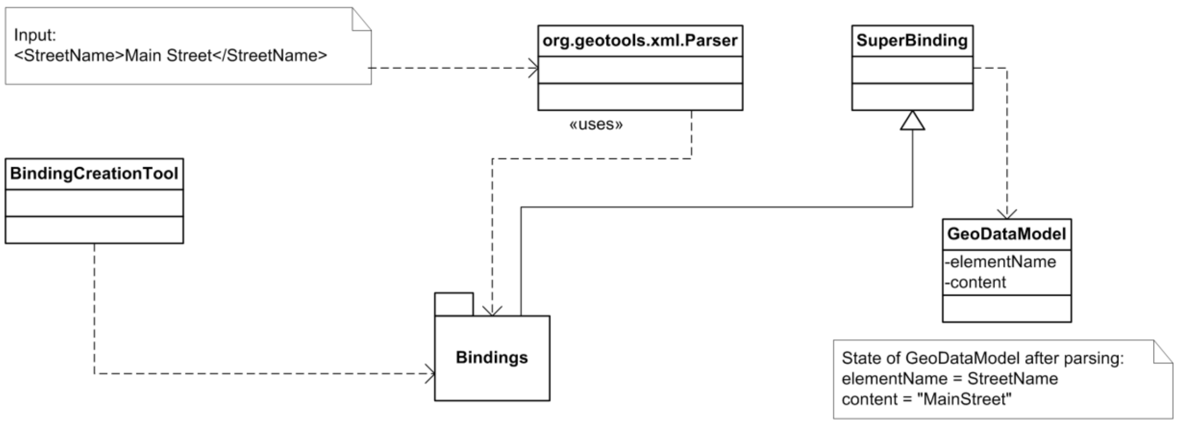

Figure 5. Usage of the GeoToolsParser in the DMMapML framework

CONCLUSION

The paper presents an approach for an integrated interoperable system architecture for management, simulation and decision support systems.

Different parts of the prototypical system architecture have to be completed and refined on the "conceptional" (DMML) and the "software/application related" (DMIF) level. To evaluate the research results the integration of DMIF into the existing earthquake system and the flood water system, which is currently under development, has to be pursued.

In doing so a visualisation package should be added in order to provide a graphical representation of the digital map e. g showing damages and unit positions. Furthermore, the automatic creation of the binding classes still requires the user to denote all relevant elements of the given XSD into a configuration file. This could be improved by automatically parsing the XSD and searching it for relevant elements.

REFERENCES

1. Aymond, P., Brooks, R., Grapes, T., Ham, G., Iannella, R., Robinson, K., Joerg, W. and Triglia, A. (2008) OASIS Emergency Data Exchange Language Resource Messaging (EDXL-RM) 1.0
2. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. and Yergeau, F. (2008) Extensible Markup Language (XML) 1.0 (Fifth Edition)
3. Engelmann, H., Fiedrich, F. (2009) DMT-EOC – A combined system for the Decision Support and Training of EOC Members, *Proceedings of the 6th International ISCRAM Conference*, Gothenburg, Sweden
4. Fallside, D. C. and Walmsley, P. (2004) XML Schema Part 0: Primer Second Edition
5. GeoAPI (2010) http://geoapi.sourceforge.net/, last visited February 25, 2010
6. GeoTools (2010a) http://geotools.org/, last visited February 25, 2010
7. GeoTools (2010b) http://docs.codehaus.org/display/GEOTDOC/XML+Developers+Guide/, last visited February 25, 2010
8. Herring, J (2006) OpenGIS Implementation Specification for Geographic information - Simple feature access, Version 1.2.0, Document # 06-103r3
9. Jones, E. (2007) OASIS Common Alerting Protocol, Version 1.1
10. JTS (2010) http://www.vividsolutions.com/jts/jtshome.htm/, last visited February 25, 2010
11. OpenStreetMap (2010) http://www.openstreetmap.org/, last visited February 25, 2010
12. Portele, C (2007) OpenGIS Geography Markup Language (GML) Encoding Standard, Version 3.2.1, Document # 07-036, Page 6
13. Raymond, M., Webb, S. and Aymond, P. (2006): OASIS Emergency Data Exchange Language (EDXL) Distribution Element, v. 1.0; OASIS Standard EDXL-DE
14. Werder, S. (2007) Knowledge Representation For Disaster Management, *International Symposium on Strong Vrancea Earthquakes and Risk Mitigation*, Bucharest, Romania