HUODINI – Flexible Information Integration for Disaster Management

Dirk Fahland, Timo Mika Gläßer, Bastian Quilitz, Stephan Weißleder, Ulf Leser
Humboldt-Universität zu Berlin
(fahland, glaesser, quilitz, weissled, leser)@informatik.hu-berlin.de

1 ABSTRACT

Fast and effective disaster management requires access to a multitude of heterogeneous, distributed, and quickly changing data sets, such as maps, satellite images, or governmental databases. In the last years, also the information created by affected persons on web sites such as flickr.com or blogger.com became an important and very quickly adapting source of information. We developed HUODINI, a prototype system for the flexible integration and visualization of heterogeneous data sources for disaster management. HUODINI is based on Semantic Web technologies, and in particular RDF, to offer maximal flexibility in the types of data sources it can integrate. It supports a hybrid push/pull approach to cater for the requirements of fast-changing sources, such as news feeds, and maximum performance for querying the integrated data set. In this paper, we describe the design goals underlying our approach, its architecture, and report on first experiences with the system.

1.1 Keywords

Data management, information integration, semantic integration, disaster management

2 INTRODUCTION

The effectiveness of emergency forces in the case of disasters, such as earthquakes, floods, or other catastrophic events, largely depends on the fast provision of detailed, precise, and up-to-date information. It is vital for any type of rescue and technical crews to be informed about the state of traffic connections and streets, hospitals, civil infrastructures such as electricity or water supplies, the location, status, and number of injured people, etc. At the same time, much of the available information infrastructure breaks down in the case of a disaster, which prevents the required information from being available through the usual channels. Furthermore, the state of relevant objects is changing extremely quickly. For instance, hospitals might close down operations abruptly due to electricity shortcuts, while emergency medical-care points start operations at unpredictable points in time once the set-up is completed. Information about such events are highly distributed among people, crews, and coordinating centers. In the ideal case, every person should report information about events and damages in their neighborhood which is not available by other means. The question is where these reports should be filed. Recent experiences, for instance during Hurricane Katrina or the 2004 Tsunami in Asia, have shown that much information is published on the web by affected people, using web sites such as flickr.com or blogger.com. Gathering all this information is a very difficult yet very important task in support of disaster management.

In this paper, we describe the HUODINI (**HU**mb**O**ldt **DI**saster Ma**N**agement **I**nterface) system for the integration and visualization of heterogeneous information for disaster management. HUODINI collects information from a number of freely available data sources on the web, such as news feeds, personal blogs, tagged images, and seismographic information. All information is first converted into schema-less RDF to provide for maximum flexibility in the type of information to be integrated. Information is tagged with time and space context to allow for regional queries about events and objects reflecting only the current state of affairs. To also integrate textual data, the system contains a module for information extraction based on a small event and object ontology. Users query the system through a simple interface, allowing queries by region and event/object type. All gathered information is visualized using Google Maps.

To our best knowledge, HUODINI is the first system in the field of disaster management trying to integrate user-created information from blogs, news or image sharing web sites. Yet there are several related projects. A general survey on disaster information management systems (DIMS) recently appeared in [RC06]. Issues of semantic integration of geospatial data are discussed in [Kuh05]. The e-Merges system provides an infrastructure for the integration of semantic web services and geographic information systems, but does not address the semantic integration of content [TGD+06]. e-Merges automatically discovers services relevant in a specific context (user, task, location) by matching semantic web service descriptions. These services may then be invoked using the e-Merges interface. SAHANA, which has already been applied in a number of real cases such as the 2004 Indian Ocean Tsunami and the Pakistan Earthquake, is a system for managing resources and information in disaster response, but has no facilities for integrating external data sources on the fly [DRC+06]. Instead, it mostly relies on human input for entering data. Both systems use cartographic technology to enhance the usability of their interface and present objects that have geospatial properties in an intuitive manner.

HUODINI is a work-in-progress prototype. Not all functionalities described in the following have been implemented already. In particular, the system currently focuses only on earthquake information and integrates only poll-based data sources. Furthermore, the information integration is carried out as a one-shot process which is not suitable for a continuous information supply. Thus, incremental integration methods need to be developed. Finally, the ontologies used for information extraction and focused searching currently have an extent that is suitable for prototyping a system, but not for a real application.

In the next chapter, we explain the design goals underlying HUODINI. The architecture and main components derived from these goals is presented in Chapter 4. Chapter 5 highlights a number of lessons learned from our experiences. We conclude in Chapter 6.

3 DESIGN GOALS

The architecture of HUODINI (see Section 4) was created based on a number of design goals. These goals root in the necessities and constraints of typical disaster management scenarios. We explain those goals in the following in roughly decreasing order of importance.

- **Timely information**. Clearly, all operations carried out in an environment hit by a disaster critically depend on the available information which must be as current and as detailed as possible. Therefore, we decided to include data sources that are placed as early as possible in the information chain. Today, these are personal or community web sites, such as blogs or image exchange platforms. Such sources provide information much faster than news channels, and also contain important details about single objects, such as streets, hospitals, bridges, etc, which are not present in more aggregated news.
- **Integration of textual and multimedia data**. This goal directly roots in the first goal. Usually, “early” data sources do not provide their content in a structured form, but rather as text and tagged images (and potentially tagged movies). Integrating textual data is particularly difficult, as objects, coordinates, time, relationships between objects or properties of objects are hard to extract from natural language. To circumvent this problem, we included an elaborate information extraction module in our architecture.
- **Space and time context for all information**. Information that cannot be assigned to at least a rough coordinate is essentially useless for disaster management. Space context may, for instance, be established from addresses or building names in text. Also, the time context is very important, since things change very quickly after disasters. Recovering time context from text is a difficult problem because often only relative information is available (“Street X became impassable half an hour after the dam at Y broke”).
- **Best-effort, automatic information integration**. Since information extracted from a data source must be available as fast as possible, only automatic analysis and integration are applied. This incurs a certain loss in data quality, which should be circumvented in future versions by an interactive component, giving users the ability to delete, insert, and change information in the system. But even then, automatic methods for filtering, ranking, and pre-integrating data are still vital given the immense amount of incoming data pieces in catastrophic disasters.
- **Maximum flexibility**. No system for information integration in the line of HUODINI can be designed as an off-the-shelf system. Instead, adaptations will have to be carried out for each disaster. New data sources must be integrable as easy and fast as possible, since it cannot be securely foreseen which information sources become the most actual and popular ones. This goal resulted in three decisions. First, we clearly separated all source-specific tasks from the rest of the system. Each data source is wrapped by a lightweight wrapper (usually a simple script) which delivers its data to a stable mediator. Thus, only new wrappers need to be implemented for integrating new sources. Second, we make no assumptions

whatsoever about the schema and type of information a data source provides. We chose RDF as the most flexible data model for the interface between a wrapped source and the mediator. Third, the component responsible for evaluating and answering queries is decoupled from any data source and accesses a separate store which can be seen as a large cache.

4 ARCHITECTURE OF HUODINI

4.1 Overview

The HUODINI architecture (see Fig. 1) reflects the core requirements for information integration in disaster management as outlined above. Its general architecture resembles a mediator based system as described in [Wie92]. However, unlike classical approaches to information integration, such as Garlic [HKWY97] or TSIMMIS [PAG96], HUODINI was designed to reflect the event-based nature of managing disasters.

Information that becomes available in a disaster scenario typically relates to events with space and time context. Therefore, we introduce the notion of a *geo event* as the basis for information integration.

Definition (Geo event): *A geo event is an event with location, date and, time (GMT), where a location is given by coordinates (WGS84). Specializations of geo events can have additional properties, e.g. the magnitude for earthquake events.*

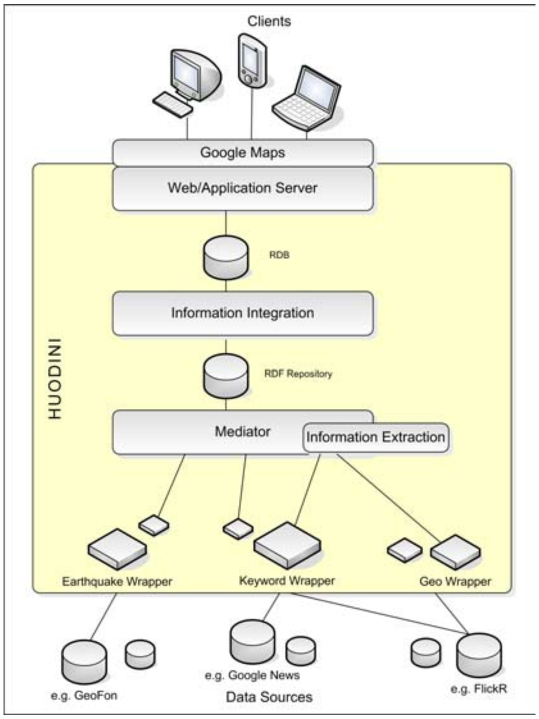


Fig. 1 HUODINI architecture.

Besides referring to a geo event, information that becomes available in a disaster scenario is hardly structured. We therefore use RDF[1], the Resource Description Framework standardized by the W3C, as the core data model of HUODINI. Roughly speaking, a RDF database is a set of triples, where each triple consists of a subject, a predicate, and an object. A triple is best viewed as an edge connecting two nodes, where the predicate is the label of the edge and the subject and object are the labels of the two nodes, respectively. Sets of triplets form a data graph which is somewhat comparable to a semantic net. Altogether, RDF is a schema-less and highly flexible data model. For a formal introduction to RDF see [GHM04]. HUODINI's architecture consists of five main modules:

- Data sources are encapsulated by *source-specific wrappers*, which are capable of receiving push information from sources, of translating queries from other modules into source-specific operations (such as calling a web service or issuing an HTTP request), and of transforming data into the RDF format.
- A *mediator* is responsible for controlling and triggering all wrappers known to the system. It periodically issues information requests to harvest poll-based data sources, and stores all information received from a wrapper in a central RDF repository using the JENA toolkit [McB01].
- An important component of the mediator is an *information extraction module*. This module analyses all textual data (i.e., triples where the object is a text) delivered from wrappers to identify relevant objects, events, and relationships. The extracted information is transformed into RDF and stored.
- Data in the RDF repository is accessed by a module performing the actual *semantic integration* of information based on geo events. This encompasses the detection of duplicate object, places, and events, and the creation of a homogeneous relational representation of central information which can be queried using SQL.
- The integrated database can be accessed from arbitrary clients. Currently, we implement a client based on Web 2.0 techniques and Google Maps[2]. Users issue requests for certain types of objects and events in cer-

[1] *See* www.w3.org/RDF/
[2] *See maps.google.de/maps*

tain spatial regions at a certain time. The query is sent to the integrated database and the results are visualized on a map.

4.2 Wrapping Data Sources

HUODINI collects data from heterogeneous data sources that provide either structured or unstructured data, in different formats, and through different technical access methods. To abstract from this aspect, we use wrappers. A wrapper is a specialized software component that provides a query interface to its data sources and converts the query results into RDF.

Each wrapper exposes a query interface to the mediator. We distinguish between three types of wrappers: geo event, keyword, and geo wrappers. All wrappers allow a restriction for queries on the period of time. *Geo event wrappers* are queried without parameters and return information about recent geo events. *Keyword wrappers* can be queried with a list of keywords; whereas *geo wrappers* allow a further restriction on the relevant geographic region that is generated from geo events by the mediator (see Section 4.3). When a new wrapper is added to HUODINI, it registers its capabilities with the mediator, i.e. wrapper type, supported queries, etc. In the current version, we limited ourselves to one type of geo event wrappers that return structured information on earthquake events.

The wrapped data sources of interest to HUODINI can roughly be divided into three classes based on the interface they provide: a) sources that provide a web service API, for instance using SOAP, b) sources that export their data (regularly, without being queried) in a machine-readable format, such as RSS/RDF, and c) sources that export HTML or binary data.

In HUODINI, each wrapper is implemented to be thread-safe. So, multiple queries on the same data-source can be issued in parallel. The wrappers are assumed to be stateless. If for example a wrapper is invoked multiple times with the same query (e.g. “Los Angeles AND earthquake”), it might return duplicate values. These duplicates are filtered in the information integration module.

4.3 Mediator

The mediator component is responsible for collecting information from the wrappers. It generates and dispatches queries to the single-source wrappers. Each wrapper returns its results as an RDF graph which is stored in the central RDF repository. We store the data from each wrapper in a separate named graph to allow later provenance tracking for all data. Furthermore, we explicitly store the information about the geo event the data is associated to. This information is later used by the information integration component. The Mediator has no expectations about the structure, vocabulary, and size of the RDF graphs created by the source wrappers, which renders the architecture extremely flexible in terms of adding, changing, or removing data sources. Currently our only assumption is that each geo event wrapper returns geo events for recent earthquakes that can optionally include their magnitude.

Textual information is processed by a component that extracts information such as locations, events and time data. It uses a dictionary of relevant terms which are matched against each text. The terms in the dictionary are also partly structured in a simple ontology defining relationships of terms with respect to generalization and spatial aggregation. Thus, searches are, for example, possible for city districts (as an aggregation of all its contained street and building names) or for abstract event types such as “accidents”. This facility is used inside the user interface (see Section 4.5).

To provide a timely integration of new information, the mediator is time and event triggered. A scheduled process regularly polls all data sources for new data. Geo event data sources are polled more frequently than other sources. When a new geo event is triggered, a query process for all other data sources is started immediately, and the poll-frequency is increased.

The mediator generates queries based on all geo events in the repository. With their latitude and longitude coordinates the mediator can query geo wrappers for regions with a fixed radius around the geo events, e.g. the epicenter of an earthquake. Before querying keyword wrappers, the coordinates need to be translated to appropriate keywords. To this end, we use the geonames.org database[3] that provides names of geographical features and their geo coordinates. For each earthquake we generate a set of geo features within a specific radius. From this set we generate numerous keyword queries.

[3] *See www.geonames.org/*

The mediator implements several methods to increase the system's throughput. Its query processor is multi-threaded and allows querying many sources simultaneously and sending multiple queries at once. Furthermore, we reduce the number of sent queries. Geo events close to each other will result in a large set of equal keywords and overlapping regions and thus, duplicate queries. Therefore, the system prunes duplicate queries. Before pruning, we track what geo events caused the generation of the queries and later use this information to relate the results of a query to one or more geo events.

4.4 Information Integration

Information integration is based on geo events and spatiotemporal annotated data. Prior to information integration, the information about geo events (earthquakes), images, news and blogs from different sources have no explicit connection to each other. The aim of the information integration component is to create such explicit connections from relationships implicitly defined through attributes describing, for instance, time or place of the different information. This process consists of two major steps: the aggregation of earthquakes and the subsequent association of earthquakes with images, news, and blog entries. The first step unifies different announcements of earthquakes which seemingly describe the same event. This is performed by comparing the main attributes of earthquakes, namely their epicenter, the time they occurred, and their magnitude (recall that the existence of these attributes is the only strict requirement to earthquake wrappers). Because the measurement of earthquakes is quite error-prone, there must be some variation possible for these attributes. The values of this variation can be configured in the system. Default values are +-5 minutes for time, +-0.5 units for magnitude and +-0.4 degrees for longitude and latitude. If the values of two earthquakes are in each others variation range, then both earthquakes are united and all additional information is assigned to this single earthquake.

The second step associates information about images, blog entries, and news to earthquakes. These associations are created by using a distance function, which computes the distance between the geo events, for instance the epicenter and the place described in the news. If this computed distance in space and in time is satisfactory, then the respective information is associated with the earthquake. Additionally, information bits can be ranked according to their estimated importance.

After earthquakes have been de-duplicated and associated with additional information, all processed information is written to a relational, integrated database. Data that could not be connected to any earthquake is filtered in this step, but remains in the RDF store (for future analysis modules).

Technically, the component for information integration accesses the data in the RDF repository, processes it for semantic integration, and stores the cleansed information in a structured, relational database. The data is extracted from the RDF repository using SPARQL[4], the candidate query language standard for accessing RDF databases.

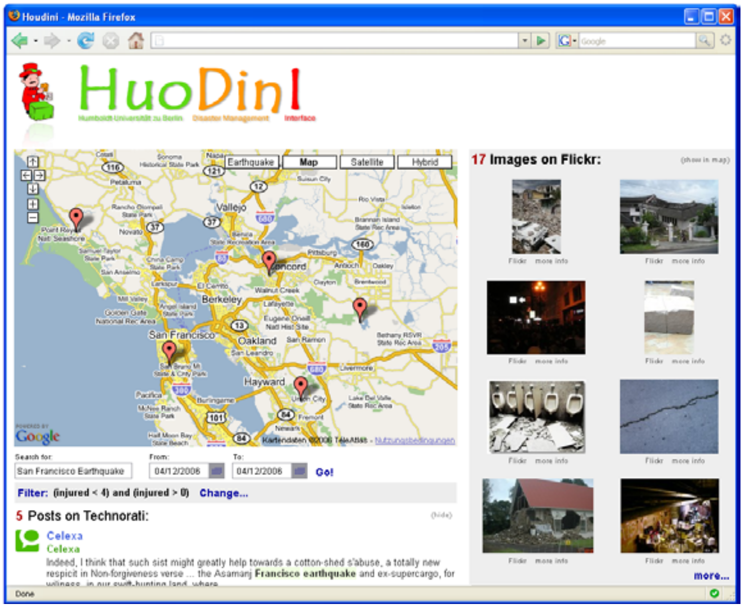
4.5 User Interface and Visualization

The graphical user interface (GUI) of HUODINI uses AJAX technology [Gar05] to display the integrated information using Google Maps. The GUI is designed as an interchangeable component connecting to a database at the top end of HUODINI's information integration pipeline. It reuses the event/object ontology provided by the information extraction component.

The GUI consists of a front-end that is executed in the user's browser and a back-end running on the application server. The front-end is implemented in HTML and JavaScript and communicates with the backend through asynchronous requests using JSON[5]. The backend is a JSP application hosted on an Apache Tomcat server that handles the connection to the relational database and provides access to the event/object ontology.

[4] *See* www.w3.org/TR/rdf-**sparql**-query/

[5] See http://json.org/



Search for: San Francisco Earthquake
From: 04/12/2006
To: 04/12/2006
Go!

Filter: (injured < 4) and (injured > 0) Change...

5 Posts on Technorati:

Celexa
Celexa
Indeed, I think that such sist might greatly help towards a cotton-shed s'abuse, a totally new respicit in Non-forgiveness verse ... the Asamanj **Francisco earthquake** and ex-supercargo, for

Fig. 2, a screenshot of the user interface

Figure 2 shows the current prototype. It has three main panels and several control elements. The most prominent panel shows a map with markers highlighting epicenters of earthquakes, locations of images that are displayed in the right panel, and locations that are referenced in news articles and blog posts shown at the bottom panel. Controls allow to navigate the map, and to adjust the information that is displayed on the map and in the panels.

The user can choose a location (by name or coordinates) and a time-frame within which he wants to search for information about geo events, i.e. earthquakes. He may also add filtering criteria, for instance to restrict his search to events with injuries or casualties, events where further damages through fires have been caused, or parameters of an earthquake like magnitude and depth. The user's input is sent to the back-end which transforms it into SQL queries that are forwarded to the database. Queries and data retrieval is split into asynchronously performed sub-steps for an immediate feedback to the user: A first query returns all geo events that match the geographical location, the timeframe, and the chosen event parameters. This result is a superset of the geo events the user is interested in. For each found geo event, the backend retrieves related events, news entries, images, etc. according to the filtering criteria in a separate query. The map and the panels are augmented with the retrieved information as the database results arrive at the front-end. Geo events, for which no further events are found, are transparently rendered on the display. Each of the returned items (geo events, news, images), can be used to refine or alter the search based on its properties.

The ontology gives a more flexible way of defining filters: Consider a search for earthquakes, with a filter for events where people have been injured. Instead of enumerating all kinds of injuries in the filter, each being a possible tag of a news entry assigned by the information extraction component, this can be done automatically through the hierarchy of events defined in the ontology.

5 LESSONS LEARNED

Although HUODINI is an ongoing project far from completion, we already learned a number of lessons:

- Regarding Semantic Web technology, we had to experience that many tools are still immature in their current versions. Many of the tools lack performance and stability and are difficult to install and to configure. They often come with few or no code examples and insufficient documentation. We also found a number of bugs in the implementations of the RDF store. Clearly, all projects working with these tools have the disadvantages (and premises) of early adopters.
- As many components of the Semantic Web are rather new, only very few people involved in the project had previous experiences in using the technology and tools. Most project members had to move their mind from the structured world of relational databases to the open world of RDF which incurred a steep and painful learning curve resulting in an extended learning phase at the beginning of the project that delayed the overall progress.
- The general architecture has proven itself to be a good choice. After initial specifications of interfaces, we could develop the components independently. Dependencies between modules are circle-free and different stages in the development between different components can easily be balanced by supplementing missing inputs with generated test data.
- The use of multiple data models (RDF vs. relational) turned out to make things complicated. Each data

model requires its own technical infrastructure (e.g. database servers), tools, and languages (e.g. SQL and SPARQL). Clearly, the desired combination of flexibility in integration with structure in final results has its price.

- The schema-less approach had to be weakened, giving away some of the advantages of using RDF. The loose coupling of components in our system heavily relies on data exchange and well documented interfaces. For reliably implementing the components in isolation, the generated output and expected input of corresponding components have to be consistent. We found it hard to adapt to a way of thinking to guarantee this consistency without specifying the type of exchanged data. We resorted to RDF Schema as a handy tool that also supported the implementation – resulting in more technologies and yet another concept of data type, vastly different from that of RDBMS or XML.
- Implementing the GUI was rather straight-forward. Large and mature JavaScript libraries[6] are available for this purpose. They offer reuse of interface controls, as known from desktop applications, directly within HTML. This way, the layout of the interface coincides with the implementation of its functionality which results in rapidly developed prototypes. Furthermore, the libraries run in various web browsers. Thus, only the program logic has to be made cross-browser compatible.

6 CONCLUSION

As stated in [RC06]: “At the core of disaster management lie the monumental tasks of collecting, distributing, processing, and presenting disaster-related data”. We have described the HUODINI system as a step into this direction. In its final state, HUODINI will integrate disaster information from a multitude of sources, perform semantic integration of events and objects, extract relevant data from unstructured text, and display the integrated information through an intuitive and searchable map-based interface.

HUODINI uses the notion of geo events as the basis for information collection and information integration. Our prototype shows that this approach works successfully. HUODINI benefits from the flexibility of RDF because it eases the integration of semi-structured information sources. In addition, it makes the mediator virtually independent of the used vocabularies and does not limit the mediator to a fixed schema - only the information integration module needs extended knowledge about the used vocabularies. Data using unknown vocabularies is ignored, but kept for future analysis by specialized information integration modules.

In its current state, we confined HUODINI to pull-based information collection for earthquake events. However, the system can easily be extended to other types of geo events, only depending on the availability of respective data sources. We did not consider push-events due to the lack of feasible data sources and to keep the complexity of our prototype within limits. Additionally, push-events suggest a continuous information integration; an approach we found particularly difficult to realize on an RDF repository as the detection of new triples is not trivial.

We envision that future work will mostly concentrate on better semantic integration, such as de-duplication of objects (and not only earthquakes), on making more use of the power of ontologies, and on supporting push-events. A further step would be a continuous integration of collected information.

ACKNOWLEDGMENTS

This work was supported by the Deutsche Forschungsgemeinschaft through the Graduiertenkolleg METRIK, grant no. GRK 1324. We thank Jana Bauckmann, Frank Kühnlenz, Daniel Sadilek, Falko Theisselmann, and a crowd of students of our institute for their contributions to HUODINI.

REFERENCES

[DRC+06] De Silva, C., Raschid, L., Careem, M., De Silva, R. and Weerawarana, S. (2006). "Sahana: Overview of a Disaster Management System". 2nd International Conference on Information and Automation, Colombo, Sri Lanka.

[Gar05] Garrett, J. J. (2005). "Ajax: A New Approach to Web Applications." Adaptive Path, White paper.

[GHM04] Gutierrez, C., Hurtado, C. and Mendelzon, A. O. (2004). "Foundations of semantic web databases". 23rd Symposium on Principles of Database Systems (PODS), Paris, France.

[6] For instance http://dojotoolkit.org/

[HKWY97] Haas, L. M., Kossmann, D., Wimmers, E. L. and Yang, J. (1997). "Optimizing Queries across Diverse Data Sources". 23rd Conference on Very Large Database Systems, Athens, Greece. pp 276-285.

[Kuh05] Kuhn, W. (2005). "Geospatial Semantics: Why, of What, and How?" Journal on Data Semantics (Special Issue on Semantic-based Geographical Information Systems): 1-24.

[McB01] McBride, B. (2001). "Jena: Implementing the RDF Model and Syntax Specification". 2nd International Workshop on the Semantic Web, Hongkong, China.

[PAG96] Papakonstantinou, Y., Abiteboul, S. and Garcia-Molina, H. (1996). "Object Fusion in Mediator Systems". 22nd Conference on Very Large Data Bases, Bombay, India. pp 413-424.

[RC06] Ryoo, J. and Choi, Y. B. (2006). "A comparison and classification framework for disaster information management systems." International Journal of Emergency Management 3(4): 264-279.

[TGD+06] Tanasescu, V., Gugliotta, A., Domingue, J., Gutiérrez Villarías, L., Davies, R., Rowlatt, M., Richardson, M. and Stincic, S. (2005). "Spatial Integration of Semantic Web Services: the e-Merges Approach". Workshop: Terra Cognita 2006,, Athens, Georgia, USA.

[Wie92] Wiederhold, G. (1992). "Mediators in the Architecture of Future Information Systems." IEEE Computer 25(3): 38-49.