Sahana: Overview of a Disaster Management System

Ravindra Silva, Mifan Careem

2006 International Conference on Information and Automation

Sahana: Overview of a Disaster Management System

Mifan Careem*, Chamindra De Silva*, Ravindra De Silva*, and Louiqa Raschid†, Sanjiva Weerawarana*
* Lanka Software Foundation,Sri Lanka
mifan | chamindra | ravindra | sanjiva @opensource.lk
† University of Maryland
louiqa@umiacs.umd.edu

***Abstract*— Large scale disasters bring together a diversity of organizations and produce massive amounts of heterogeneous data that must be managed by these organizations. The lack of effective ICT solutions can lead to a lack of coordination and chaos among these organizations, as they track victims' needs and respond to the disaster. The result can be delayed or ineffective response, the potential wastage of pledged support, imbalances in aid distribution, and a lack of transparency. ICT solutions to manage disasters can potentially improve efficiency and effectiveness. Sahana is a Free and Open Source Software (FOSS) application that aims to be a comprehensive solution for information management in relief operations, recovery and rehabilitation. This paper addresses the alignment between FOSS development and humanitarian applications. it then describes the anatomy of the Sahana system. We follow up with a case study of Sahana deployment and lessons learned.**

***Keywords*—FOSS, humanitarian applications, disaster management systems, Tsunami**

I. Introduction

Recent disasters such as the 2003 SARS outbreak, the 2004 Asian tsunami, the 2005 Kashmir/Pakistan earthquake and 2005 hurricanes Katrina and Rita clearly identified the shortcomings of ICT solutions for disaster rescue and recovery. Large-scale disasters are typically accompanied by the need to effectively manage massive amounts of data. This includes data about victims and about relief personnel; data about damages to buildings, infrastructure and belongings; weather data; geographical data about roads and other landmarks; logistics data; communication and message data; financial data needed to manage the collection and distribution of donations; data in blogs; etc. Major disasters also involve multiple autonomous organizations (governmental, NGOs, INGOs, individuals, communities, and industry). This leads to a diversity of client needs that must be coordinated.

Despite the tremendous value of disaster management systems, there are only very few systems that exist today and none are widely deployed. The most widely used system appears to be non-Web based and uses proprietary non standard database technology. While there are various specialized components that exist, there does not exist a single cohesive system that organizations such as the United Nations Disaster Assistance and Coordination (UNDAC) can routinely deploy.

There are disaster information systems that focus on specialized application or data requirements including imagery and GIS data[1] [2] [3], early warning models using sensor data [4], mobile ad hoc networks and messaging, etc [5]. However, there is no current project that provides information system or data management support for the basic functionality of disaster management such as registering organizations, locating missing persons and requesting assistance.

Effective ICT solutions can support the coordination of AID groups and help them keep track of victims' needs while responding to the disaster. As a result, there is potentially less wastage of pledged support or imbalances in aid distribution. In particular, it is hoped that disaster information management systems can support efficiencies and effectiveness in coping with the information management and coordination needs during a disaster. Sahana is a Free and Open Source Software (FOSS) application that aims to provide a comprehensive solution for information management in relief operations, recovery and rehabilitation.

In this paper, we first address FOSS development strategies and humanitarian applications and make the case that there is a natural fit between the two. We then describe the anatomy, architecture and platform for the Sahana FOSS project. We then provide a case study of a Sahana deployment and use this case study to identify lessons learned for the future.

II. FOSS and Humanitarian Applications

Free and Open Source software (FOSS) constitutes a significant portion of the software market today. The application of FOSS has enabled various communities to exploit information technology in diverse operations and to reach a wider population. Innovation driven FOSS is free of charge. More important, it often addresses areas that may not be sought after by a profit seeking company. This is the case with disaster management; despite the tremendous value software can bring, there are very few systems that exist today and none of them are widely deployed. The Sahana project was born as a FOSS application to address disaster management, collaboration and coordination in the aftermath of the 2004 tsunami. The Sahana project almost immediately gained widespread attention from developers and humanitarian consultants worldwide. A major reason for Sahana's success is that the FOSS ethos and humanitarian requirements bond well together.

A. *FOSS alignment for humanitarian-ICT applications*

In general, there are numerous reasons why FOSS software finds a natural fit for information and communication technology applications in the humanitarian domain. We expand on a few reasons as follows:

- Very few countries and organizations can afford to invest a lot of resources in disaster management when there is no ongoing disaster; there are always higher priority items competing for limited funding. A FOSS approach provides a low budget, volunteer supplemented and global solution to build such systems.
- Disaster management is not a lucrative domain for software development, compared to business applications. During humanitarian disasters, software licenses are given freely and it almost seems unethical to restrict software usage during a disaster. Any source of revenue would involve pre-deploying software to governments and NGO's. FOSS can be used to develop such a solution without a focus on profits.
- Disaster management software should be shared, developed and owned globally as the problems they address are all too common for any country dealing with a disaster. Effectively it should become a global public good. The FOSS development and community mechanisms have a proven track record to build such goods.
- Medical practitioners provide a valuable service in relief operations with their domain knowledge. Volunteering to develop software to manage relief operations is an ideal match for the global community of IT volunteers who wish to contribute their expertise.
- As in conflict situations, during disasters segregation arises between governments, NGOs, INGOs and people. An open and transparent and globally owned system is more likely to be trusted to mediate between the groups
- Finally no two disasters are alike. There are often localizations and customizations needed before the software can be applied effectively to the next disaster. With FOSS, the code is available for anyone to provide the needed customizations rapidly and without licensing restrictions.

B. *History of Sahana*

The tsunami that hit Sri Lanka on December 26 2004 resulted in a massive outpouring of support for the relief of the nearly one million affected people. It became clear that without information technology it would be impossible to coordinate relief efforts to maximize the impact on the affected people. The Sahana project was thus born. The Sahana FOSS project was built over a 2-3 week period by a group of volunteers from the Sri Lankan IT industry, spearheaded by the Lanka Software Foundation, a FOSS R&D NPO. An implementation of Sahana was authorized and deployed by CNO (the main government body in Sri Lanka coordinating the early relief effort). Sahana is distributed from SourceForge `www.sourceforge.net` under the terms of the GNU Lesser General Public License (LGPL) Version 2.1 [6].

Despite the early stage of this project, Sahana has garnered much support and recognition [7] including the following awards: Sourceforge Project of the Month, June 2006; 2006 Google Summer of Code; 2006 Sand Hill Good Samaritan Award; Finalist, 2006 Stockholm Challenge.

III. The Anatomy of a Disaster Management System

The FOSS Sahana Disaster Management System is developed using LAMP (Linux-Apache-Mysql-PHP/Postgres) under the Opensource concept [8]. LAMP and Opensource has proven itself time and again in mission critical community applications. Sahana was developed *from scratch*; the code is written in PHP and it sits on top of the Sahana Application Framework, also written in PHP. This is a generic framework for ease of development and deployment of custom modules.

Figure 1 shows the layered architecture of Sahana, consisting of the Sahana Application Framework at the core, and surrounded by libraries and APIs. This in turn is wrapped by a set of modules, both core and optional. The architecture is depicted as a layered diagram, where outer layers have the ability to use the functionality of the inner layers.

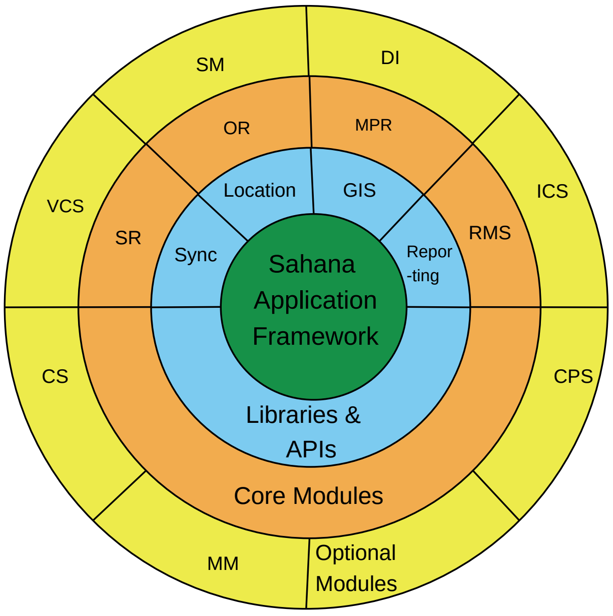

Fig. 1. Overwiew Architecture

The Sahana Application Framework provides the following features:

- A fexible and modular architecture, including a front-end controller that makes it easier to manage and synchronize system-wide tasks and events.
- Support for security at the modular and framework level, and support for Internationalization and Localization of content. [9].
- Web-based and console based commands to ease setup and configuration of Sahana and its database.
- Support for automatic detection of new modules, and dynamic plug and play installation and activation and configuration of 3rd party modules.
- A theme and layout engine and HTML template libraries for simple GUI development and customization.
- Support for accessing the schema and metadata and the data of a common database.

- Administration functionality for module administration.

The Sahana Application Framework is wrapped by a set of libraries and APIs that provide a set of core and optional capabilities to the framework and modules. This includes the following:

- Location API: Location hierarchy selection and storage.
- GIS API: GIS mapping interface and spatial capability.
- Reporting API: Automated and custom reporting capability.
- Synchronization API: Provides Database Synchronization between Sahana instances.

The core modules of Sahana are built on top of the Sahana Application Framework and libraries. The optional modules take up the outermost layer and are installed upon request. The core modules are as follows:

- Organization Registry (OR): Keeps track of all the relief organizations working in the disaster region. Stores contact information, personnel, areas where each is active, and the range of services that each can provide in a specific area.
- Request Management System (RMS): A repository where relief organizations, relief workers, government agents and camp managers can match requests of aid with pledges of support. It can function as an online trading system and can also track the fulfillment of requests.
- Shelter Registry (SR): A registry of shelters and camps. Keeps track of the location of all the camps and shelters, and has data on the facilities that they provide, e.g., medical, the number of people assigned to them, etc. It can also provide a GIS viewer to plot the location of the camps and shelters.
- Missing Persons Registry (MPR) : A bulletin board like capability of both missing and found people. It captures information about missing people, the people who report them to be missing, along with information of the person making a search. All such information can be linked to assist people in their search.

The optional modules of Sahana currently consist of the following:

- Volunteer Coordination System (VCS).
- Child Protection System (CPS).
- Inventory Control and Catalog System (ICS & CS).
- Situation Mapping (SM): Collaborative mapping applications for disaster monitoring.
- Data Import (DI): Data import utilities to support interoperability with other applications and datasets.
- Mobile Messaging (MM): Multi-format messaging capabilities for Sahana.

Figure 2 shows the component diagram of Sahana with relation to the LAMP platform and programming environment. The base consists of the Operating System, which hosts the Database servers and the Apache Web Server. PHP, the language used to develop Sahana, is included as an Apache module. Sahana also makes use of several 3rd party libraries for its operation. Some of these libraries are bundled with Sahana, including XAJAX [10] and NuSoap, shown at the top of the Libraries section. Some other libraries must be provided by the deployment environment, e.g., UMN/Mapserver [11]. 3rd party libraries shown at the bottom of the Libraries section, such as PHP GD and PHP GACL, are Apache modules; they need to be included within the Apache Web Server by the deployment environment.

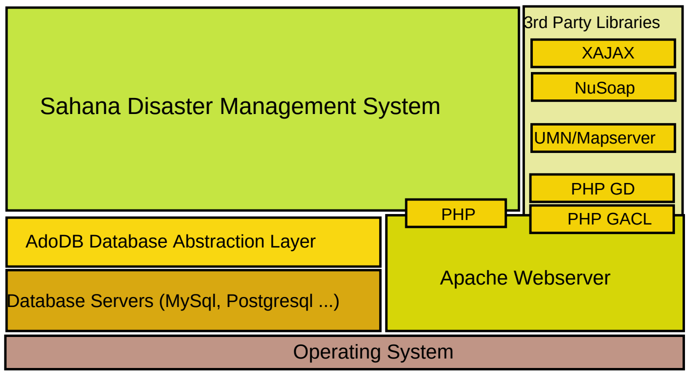
Sahana Disaster Management System
3rd Party Libraries
XAJAX
NuSoap
UMN/Mapserver
PHP GD
PHP GACL
PHP
AdoDB Database Abstraction Layer
Database Servers (MySql, Postgresql ...)
Apache Webserver
Operating System

Fig. 2. Component Architecture

The database layer is accessed via the AdoDB abstraction layer; this provides database independence to Sahana. Module developers have the leisure of using the abstraction layer instead of worrying about the underlying database server, and server specific API calls and queries. Currently Sahana supports MySQL [12] and PostgreSQL.[13]

IV. Case Study of Sahana Deployment

Sahana is not yet a prototype disaster management system with full capabilities; instead, it is a work in progress. The Sahana team therefore decided to give priority to deployments in order to obtain a complete set of requirements and to understand operational challenges during a disaster. As a result, Sahana capability and robustness has improved with each deployment. We provide a summary of Sahana deployments as of September 2006. We then provide a detailed description of one deployment followed by lessons learned.

A. *Summary of Sahana Deployments up to September 2006*

Sahana phase 1 was officially used by CNO (Center of National Operations) of Sri Lanka as a part of their official portal in 2005. It was also deployed by the Pakistan government during the 2005 Kashmir/Pakistan earthquake. Sahana phase 2 was deployed at several major disasters and there are several pre-deployments as well. This includes the Guinsaugon landslides disaster in Philippines 2006, and the Jogjakarta Earthquake in Indonesia 2006 for which Sahana was deployed for information management in relief operations. In September 2006, Sahana is being tested for deployment in Lebanon, Ecuador, Philippines and Indonesia. Among the pre-deployments, Sarvodaya, the largest NGO in Sri Lanka, customized Sahana for their hazard information hub and trained their staff in the use of Sahana. The Sarvodaya customization is populated with location data (province, district, and city/village), and Sarvodaya volunteer and inventory data.

B. Sahana deployment for the Philippines

In this section, we use the Sahana phase 2 deployment for the 2006 Guinsaugon landslides disaster in Philippines to develop a detailed case study. We provide a chronological history to provide a realistic view of how Sahana was deployed.

On February 17, a devastating mudslide killed over 1,800 Filipinos in Guinsaugon in the southern part of Leyte Island in the Philippines. IBM contacted the Sahana *core team* at the Lanka Software Foundation on February 22 to request the deployment of Sahana. The core team had to wait until the IBM Philippines team forwarded a list of customizations from the National Disaster Coordinating Council (NDCC) of the Philippines. The team in Philippines was a joint effort of NDCC and IBM Philippines and will be referred to as the *Philippines team*. The core team received the initial list of required customizations on February 28 via email. As an example, the GIS customization requirement was as follows:

```
5. GIS Mapping
       5.1. Philippine AOR
       5.2. Southern Leyte
       5.3. Google Earth interface
       5.4. UNOSAT images
```

The customizations were analyzed and divided into two groups. The first group of *simple* modifications consisted of those changes that could be directly implemented by Sahana clients using the *administrative interface*. It included features such as adding new locations, setting the initial GIS map to Southern Leyte, modifying the users and roles and organizational services and types. In addition, there were some features that the core team failed to initially identify as simple changes because the Philippines team used a different controlled vocabulary to refer to current Sahana capabilities.

An example is the `satellite based location feature` that can be used to collect contact information; this could be mapped to an operational feature of the organization registry. All the above customizations were completed within 24 hours.

The second group of customizations required code modifications. Example modifications are as follows:

- Changes were made to the database schema, to provide additional fields, e.g., to the missing person tables.
- The *mysql-config.sql* configuration file that was used to populate the database was modified. For example, SQL statements to generate the location levels were changed from ‘Country’, ‘Province’, ‘District’, ‘City/Village’to ‘Country’, ‘Region’, ‘Province’, ‘City/Municipality’.
- The User Interface (UI) was also modified. For example the Request Management System was labeled as *Relief Supply Management System*; such changes were made via changes to a configuration file so that the UI is easily customized.

Two additional requirements were provided later by the Philippines team. The first was to develop a data import capability to populate the Sahana database with missing people data captured in a spreadsheet. This was implemented by a PERL script. The second was to enable self registration of Sahana users as volunteers. For this the user registration interface was linked to the volunteer registration process; data was shared between the two processes using *Session* variables.

Next, we describe how the source code modifications from Sri Lanka were propagated to the disaster. A deployment server was located in Philippines running FreeBSD with Apache,PHP and MySql and configured by the Philippines team; it is labeled *deploy*. The Sahana code base is maintained at SourceForge.net under Concurrent Versions System (CVS). Two servers of CVS data are maintained; one for developer CVS access labeled *developer server* and the other for anonymous access labeled *pserver*. The developers such as the Sahana core team download and upload source code to the *developer server*. The *pserver* synchronizes with the *developer server* to obtain the latest source code. Thus, there may be a synchronization delay before the latest source code from *developer server* is propagated to *pserver*.

The core team initially decided to ask the Philippines team to download the source code from the *pserver* using anonymous access.

It was soon observed that the synchronization delay between the *developer server* and the *pserver* was significant. For example, a bug fix for the camp registry module was uploaded at 12.25 p.m. on March 6, but the code change propagated to *pserver* only after seven hours at 7.43 p.m. The solution to this latency / consistency problem was to download the Sahana code base code using one of the developer accounts and not using anonymous access. Therefore the core team was given SECURE SHELL access to *deploy*. The core team downloaded the source code to *deploy* using tis account, when ever the code base changed. Since developer commits and downloads are always from the *developer server*, the core team could ensure that the *deploy* server had the latest code base.

To further streamline the customization process, a demonstration server was setup at `www.sahana.lk/philippines`. The core team continually uploaded modifications to this server and and the Philippines team continually reviewed the customizations. In addition to the demonstration server, email and Internet chat was used for feedback.

The simple requests that could be completed through the Sahana administration interface were completed in 4 days. The major modifications were completed within 7 days. Validation lasted for 2 weeks and the entire deployment task was completed by March 15.

Over the course of the disaster, approximately 50 organizations and about 1000 disaster victims were managed. 5 camps were also established and supported. The Request Management System handled 4 requests and 71 pledges of support. It did not perform automated matching of requests to pledges since this capability was not requested.

The following is a quote from Avelino J. Cruz, Jr., Secretary of National Defense, Philippines, to exemplify the impact Sahana had as a disaster management system:

> The NDCC and OCD value SAHANA’s contribution to the relief and rehabilitation phases of the Southern

Leyte landslides and recognize the tremendous boost to our preparedness for future disasters.

While SAHANA cannot solve all the problems in a disaster, it is an excellent tool to create registries that can provide timely and reliable information on missing persons, donated goods and services, camp locations, and the like. It is technology that can help many people in a disaster. In fact, there is no greater innovation that matters more than that which saves lives.

C. Lessons learned

Out of this experience we realized that the following guidelines are useful when deploying a disaster management system for several types of disasters:

- Obtain system authorization preferably from some local government. Since the government of the Philippines actively supported the Sahana deployment, it was possible to collect missing people data, organization data and other related data without serious problems. A FOSS project such as Sahana is fortunate in that it can draw together a group of leading humanitarian consultants who can advocate Sahana to their respective governments if the need arises.
- Allow for the evolving granularity of the data. Sophisticated systems some times fail if they cannot accommodate the lack of precise data collection in a disaster. For example, while a logistics application for a manufacturing facility is able to account for every piece of inventory in a detailed manner, during a disaster the only available information may be that a crate of food is arriving, without details of the type of food, the type of container, the weight, etc.
- A disaster management solution should work even when there is a lack of a communication infrastructure. Adequate preparation must be made to support disconnected operation and the lack of immediate data synchronization should also be handled. Further, all solutions should work on commodity hardware Where ever possible.
- Localization is a required feature for most deployments. The main customization is the translation of sentences in the Sahana interface to some native language.
- Different countries have different hierarchies and ontologies in representing geographic locations. Any disaster management system should support modifying these hierarchies and other underlying data structures and databases.
- GIS capability adds value to Sahana and attracts a lot of attention in a disaster. Customizations might be required to change the initial GIS map to the location of the disaster and to upload map files of the affected areas.
- In many cases, there will be a need for data import of data that has been collected by individuals or organizations who did not use Sahana. Some support must be provided to facilitate the quality of the imported data.
- Organizations and governments apply different measures to guarantee the protection of data. In some cases, it is required to apply security controls at the level of specific fields of the database. In most cases, the requirements are more generic. For example the typical requirement is to protect pages containing sensitive data from anonymous users, rather than to protect the data of person X from user Y. How ever, the system should have some flexibility to customize for the specific security policy of the involved organization.

To summarize, the Philippines team plans to deploy Sahana for a second time for the expected Mayon volcanic eruption. The Organization registry, Camp registry, Inventory and Catalog systems are the modules that will be used. The IBM Philippines team and the Philippines government have indicated a willingness to consider Sahana to manage information in future disasters, as their expectations were met in the 2006 Guinsaugon deployment.

V. Conclusions and Future Work

The Sahana project has proved without a doubt the viability of FOSS solutions for humanitarian applications. More significantly, a disaster typically requires the coordination of multiple clients; hence Sahana has also provided a platform for interorganizational data sharing during a disaster.

There is much left to be in the technical development of FOSS disaster management softwre. Or primary importance is the need to deal with a heterogeneity of data types (text, semistructured, Web HTML and XML, GIS, tabular and DBMS), and to develop standards and protocols for data sharing [14]. Policies for data security and privacy during the different phases of a disaster must be developed. Sahana needs to be deployed in a variety of environments where the basic communications infrastructure may be destroyed, but it must provide robust performance and be able to provide additional capabilities as the communications infrastructure is restored. Real time response may be needed for some modules such as mobile messaging and situation control.

Acknowledgment

Sahana has been supported by the Lanka Software Foundation, the Swedish Development agency Sida, The University of Colombo School of Computing, Google and the US National Science Foundation under grant IIS0533986.

References

[1] W. .H.M., "The use of earth observing satellites for hazard support," in *Geoscience and Remote Sensing Symposium, 2001. IGARSS 01. IEEE 2001 International*, Aug. 2002, pp. 135 – 137.

[2] Hussain.M., Arsalan.M.H., Siddiqi.K., Naseem.B., and Rabab.U, "Emerging geo-information technologies (git) for natural disaster management in pakistan: an overview," in *Recent Advances in Space Technologies, 2005. RAST 2005. Proceedings of 2nd International Conference on*, Oct. 2005, pp. 487 – 493.

[3] Wegmuller.U., Wiesmann.A., Strozzi.T., and Werner.C., "Envisat asar in disaster management and humanitarian relief," in *Geoscience and Remote Sensing Symposium, 2002. IGARSS 02. 2002 IEEE International*, Nov. 2002, pp. 2282– 2284.

[4] R. .G. and Kumar.A., “A natural disasters management system based on location aware distributed sensor networks,” in *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, Dec. 2005.

[5] O. .S., Maly.K., F. .E.C., and Y. .S.M., “Wireless support for telemedicine in disaster management,” in *Parallel and Distributed Systems, 2004. ICPADS 2004. Proceedings. Tenth International Conference on*, July 2004, pp. 649 – 656.

[6] Lgpl license. [Online]. Available: http://www.gnu.org/copyleft/lesser.html

[7] (2005) Tsunami-inspired fsf award focuses on humanity. [Online]. Available: http://www.tectonic.co.za/view.php?id=686

[8] Lamp wikipedia entry. [Online]. Available: http://en.wikipedia.org/wiki/LAMP_(software_bundle)

[9] Web internationalization wikipedia entry. [Online]. Available: http://en.wikipedia.org/wiki/Internationalization_and_localization

[10] Xajax web site. [Online]. Available: http://www.xajaxproject.org/

[11] Umn/mapserver website. [Online]. Available: http://mapserver.gis.umn.edu/

[12] Mysql website. [Online]. Available: http://www.mysql.com/

[13] Postgresql website. [Online]. Available: http://www.postgresql.org/

[14] F. Naumann and L. Raschid, “Information integration and disaster data management (disdm),” University of Maryland, Tech. Rep., 2006.