Research Repository UCD

Title	An open source vision for HAZUS
Authors(s)	Moffatt, Stuart, Laefer, Debra F.
Publication date	2010-01
Publication information	Moffatt, Stuart, and Debra F. Laefer. "An Open Source Vision for HAZUS" 24, no. 1 (January, 2010).
Publisher	American Society of Civil Engineering (ASCE)
Item record/more information	http://hdl.handle.net/10197/2277
Publisher's version (DOI)	10.1061/(ASCE)CP.1943-5487.0000022

Downloaded 2023-04-07T10:52:29Z

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

An Open Source Vision for HAZUS

Stuart Moffatt¹ and Debra Laefer, M.ASCE²

Despite the Federal Emergency Management Agency (FEMA) having spent \$63M developing and administering the disaster management software *Hazards U.S.* (HAZUS) over the last decade, the system's 2008 rating by the Office of Management and Budget was only "moderately effective". Today's proprietary HAZUS software code precludes usage of two major advancements in the computing industry, namely the burgeoning open source community and the advent of sophisticated, Internet-based applications.

A 2007 National Research Council study entitled "*Improving Disaster Management: The Role of IT in Mitigation, Preparedness, Response, and Recovery*" proclaimed "federal, state and local agencies should embrace... open source software and open standards development" to complement traditional IT acquisition." In accordance with current FEMA strategic planning and performance-based program management, a new software development paradigm is needed to update HAZUS and propel it to the forefront of the next generation of disaster management software tools.

Although HAZUS is distributed at no cost by FEMA, the program's functionality is restricted by an underlying commercial off-the-shelf (COTS) and closed source geographic information system (GIS). Originally, two commercial GIS programs were supported: ArcGIS and MapInfo. As features were added, ESRI's ArcGIS became the platform-of-choice. A large portion of the development work today is spent keeping HAZUS synchronized with ArcGIS upgrades. Furthermore, from its infancy HAZUS has been developed on a single, closed source operating system (namely, Microsoft Windows). The linkage with ArcGIS and Windows has precluded capitalizing on a decade of explosive Internet growth and today's innovative web services infrastructure. Multiple users on a distributed HAZUS system are not part of today's reality.

Proprietary, stand alone, and single-user disaster management systems prevent efficient data gathering and sharing capabilities and result in circumscribed utility and productivity. Thus, the shortsighted HAZUS system architecture hampers the field of disaster mitigation, and as the software quietly (and unofficially) becomes a tool for response and recovery efforts, a redesign for expanded functionality and improved performance is even more imperative.

The real need is to replace the proprietary, standalone desktop version of HAZUS with a three-tier, distributed architecture based on open source design and development principle. The three tiers should include: (i) a data tier (on the server), where the database resides, (ii) a business logic tier (also on the server) and (iii) a client tier, which displays the user interface to control the program. The system should include a substantial application programming interface (API) and a rich, web-based user interface. Application logic, including complex risk analysis models, should be decoupled from the application code and be represented in an editable, expert system called a *rule engine*. Lastly, and as an essential driver of future growth and usage, a new open source

¹ Technical Manager, Disaster Resistant University, Department of Environmental Health and Safety, University of Utah, stuart.moffatt@utah.edu, phone: (801) 581-6590, fax: (801) 585-7240

² Tenured Lecturer and Lead Investigator, Urban Modeling Group, School of Architecture, Landscape, and Civil Engineering, Newstead Rm G25, Dublin, Ireland

project called OpenHAZARDS should be created to reach these goals. Each of these aspects is described below.

One critical advantage of such an envisioned system over the current embodiment is that a well-defined API allows many types of clients to interact with the server to run the core program. According to Joshua Bloch, former architect of the Java programming language and currently a principal engineer at Google: (i) good programs are modular, and intermodular boundaries define APIs, (ii) good modules get reused, and, (iii) good APIs create long-term customers. Thus, the usefulness and strength of an enterprise application's API determines, to a great extent, how successful the application will be. Robust APIs allow other developers the opportunity to use core functionality to develop new, interoperable applications. Well-developed APIs also allow judicious project managers to parcel long-term projects (like converting HAZUS to open-source) into manageable activities.

The first client application for any new system in today's Web 2.0 landscape should be a rich, web-based user interface capable of the same functionality present in the HAZUS desktop application. Incompatible browser technology has made it difficult, historically, to create a rich client application that works on all browsers and across all platforms. Google has solved this central problem of Web 2.0 interface development with the release of Google Web Toolkit (GWT). This toolkit allows a developer to design programs in the Java language. GWT does the work of converting Java to Javascript (for the final application without the need of a browser plugin) so the developer can concentrate on designing solutions to business problems without having a "PhD in browser quirks."

When converting desktop applications to web applications, questions may arise regarding recreating a complex user interface in a "dumb terminal" like a web browser. However, technical hurdles such as these are not insurmountable, and applications developed in GWT allow a Java programmer to leverage existing tools in their development environment. Concern over long analysis run times can also be overcome. From the user's perspective, "HAZUS runs" would be requested and then queued on the server via a *delegate-and-wait* methodology. This server-client implementation is characteristic of Web 2.0 applications that are mature replacements for the "glue applications" of the early web and at the same time provide unprecedented levels of ease and functionality for both users and developers.

Maintaining a large application can be a difficult and expensive process. One software algorithm that has ameliorated this problem is a *rule engine*. Rule engines are frameworks that allow a programmer to remove logic embedded in the application code and place it in a separate, editable text file. This permits application developers to collaborate with domain experts to create an application that is more extensible. If rules or policies describing the logic of an application change, rule engines enable the majority of these changes to occur in near real-time without recompiling or redeployment. Rule engines employ the concept of *declarative programming* where rules are single atomic definitions that can be declared by the thousands to determine the logic for a program. Rules become the domain for knowledge and decisions.

The open source rule engine JBoss Rules (also known as Drools) is known for its ease-of-use over competing products because of its object-oriented, Java implementation of the Rete rule

engine algorithm. One distinct advantage in JBoss Rules is the capability to implement a domain specific language (DSL). DSLs can bridge the lexical gap between hazard domain experts and disaster application developers by employing an English-like syntax of rules written by the domain expert that trigger Java-like rules written by the developer.

Presently, although HAZUS loss estimation methodologies are peer-reviewed, freely available and well documented, the logic that supports them in the software code is so deeply embedded that several independent groups have identified extensibility as a major difficulty. In 2000, a group of volunteer experts charged with evaluating HAZUS as a result of the Disaster Mitigation Act of 2000 noted “unique analytical methods (called ‘work arounds’) have been developed for use when HAZUS [could] not estimate the losses adequately”.

Loss estimation models are not the only elements affected by legacy software deployment. Researchers at the NASA Jet Propulsion Laboratory have had technical and organizational problems trying to integrate *interferometric synthetic aperture radar* (InSAR) data into the HAZUS earthquake model. In the absence of being able to programmatically connect their mapping algorithms to HAZUS through a flexible rule engine (or a modular API as noted previously), the InSAR geoscientists could not test their hypotheses.

By designing a rule engine framework to separate domain logic, hazard researchers could design and test hypotheses for risk analysis or other features directly from within an OpenHAZARDS by creating rule sets that capture knowledge from up-to date hazard or economic models and third-party programs.

Another advantage of redesigning HAZUS as an open source code base is the potential for improved quality. In 2005, Raghunathan and associates demonstrated in their article, “Open Source Versus Closed Source: Software Quality in Monopoly and Competitive Markets” that in a monopolized market the quality of software decreases. However, when closed source and open source projects compete in the same market, closed source quality decreases as open source quality increases.

It is anticipated that FEMA will spend about \$5.6M on HAZUS in 2009, a figure that matches program expenditures for the past two years. This represents roughly 0.001% of FEMA’s overall 2009 budget. Because FEMA’s budget is primarily used in supporting administrative programs to assist states and local jurisdictions in all phases of disaster management, good fiscal policy would suggest that OpenHAZARDS, though led chiefly by FEMA, should create a unique public-private partnership to further the entire industry. This finds precedence in the IBM-sponsored donation of over \$100M to the Linux development effort. Besides corporate sponsorship of OpenHAZARDS, FEMA may draw upon its extensive network of HAZUS User Groups (HUGs). By leveraging the inroads open source has made in the academic community, a critical NRC recommendation would be initiated: “the federal government should sustain (and develop as needed) a network of research centers where IT researchers, hazard and disaster researchers, and disaster management practitioners can collaborate to study and evaluate the use of IT for disaster management from both a technological and an organizational perspective, to transition knowledge and technology to those who practice disaster management, to build human capital at the intersection of IT and disaster management, and to develop future IT capabilities.”

Participation by computer science and natural hazards faculty or students in OpenHAZARDs may spawn the collaboration the NRC recommends.

A domain-specific example already exists in converting closed source disaster management tools to open source web applications in the form of the Inventory Collection Survey Tool (InCAST) Web Edition. The original InCAST program is a helper application for collecting building inventory data for use in HAZUS. InCAST Web Edition is being prototyped at the University of Utah as a replacement for the current desktop system. A FEMA Pre-Disaster Mitigation grant awarded to the university under the Disaster Resistant University program supports this project. The motivation for the work was the set of problems that arose in collecting building inventories by a distributed team of researchers. Synchronizing updates as new information emerged was impeded by numerous and differing versions of the data source. To address the deficiency of the single-user, stand-alone InCAST desktop application, development of InCAST Web was begun using the core design goals of a multi-user, web-based application supported by a robust API that includes an extensible rule engine.

A variety of generic open source tools serve as a platform for InCAST Web: a web browser displays the multi-lingual user interface developed with Google Web Toolkit; the Apache Tomcat application server connects the user interface to the core API; standard J2EE compliant Java classes comprise the InCAST API; the JBoss Rule engine contains the business logic for saving and editing buildings and, lastly; the JBoss Hibernate object-relational mapping system is used to abstract the data access layer to the PostgreSQL database (which is extended using the PostGIS plugin to make it GIS-aware). Each of these third party components in the system is open source. InCAST Web will be delivered to FEMA in early 2009 and it is anticipated it will be available for use by others shortly thereafter.

The current limitations of HAZUS are inherent in its 10-year old design of a closed-source, stand-alone, single-user desktop system. The impetus to begin the OpenHAZARDs project is driven by a need for better, more user-friendly, and easily accessible hazard risk assessment software tools. As these tools are made available through the ubiquity of the Internet, more local jurisdictions in the U.S. and internationally will be able to better mitigate natural and technological hazards. Broad acceptance of OpenHAZARDs necessitates public-private cooperation that is well demonstrated in the open source community.

The OpenHAZARDs project will also bridge the disparate worlds of academia and practice by providing an extensible open source platform to incorporate practitioner feedback and research innovation. As natural hazards intensify and the associated damage costs continue to increase, failure to fully exploit existing and emerging technologies is increasingly difficult to justify. By following the specific recommendations outlined here through an OpenHAZARDs project, the Federal Emergency Management agency would demonstrate innovative foresight in the hazard risk assessment field.