OpenHazus Research and Analysis

Version 1.1

April 8th, 2019

Prepared by the IBM Risk MAP CDS Team

Table of Contents

Document Revision History

Version	Date	Author	Summary
1.0	3/12/2019	IBM Risk MAP CDS Team	Initial Submission
1.1	4/8/2019	IBM Risk MAP CDS Team	Revised to incorporate FEMA feedback

1 Executive Summary

The goal of this document is to describe the current and future states of Hazus, FEMA's nationally standardized methodology for estimating losses due to natural hazards. Hazus currently has an estimated 10,000 users globally and is widely respected for its peer-reviewed scientific methodology and multi-disciplinary approach to disaster preparedness. Since its inception in the 1990s, Hazus has continued to expand its advanced analysis capabilities for natural hazards, while making use of the latest desktop technology in viewing, analyzing, and presenting the results of modeled scenarios. If Hazus is to maintain its relevancy and credibility among disaster professionals, it must continue to take advantage of new technology, and new scientific methods for loss estimation. This paper will present an overview of existing Hazus technology and evaluate options for moving the current state to web-based, open source architecture.

It is important to note that Hazus, and subsequently, the proposed OpenHazus solution, is a highly complex set of tools which include academic hazard and engineering science, advanced computational methods, and a diverse collection of coding languages and technologies. This document outlines requirements and technology recommendations for OpenHazus. It is not intended as a technical design document, but it does encompass all of these components to present OpenHazus as a single, comprehensive system.

The following five categories of requirements have been proposed for OpenHazus through ongoing discussions with FEMA, Risk MAP CDS, and the user community: 1) Independence from desktop technology; 2) Independence from Esri ArcGIS products, with minimized dependence on GIS overall; 3) Modularization of existing Hazus capabilities, to streamline development and allow users to customize their experience; 4) Create data sharing opportunities for baseline FEMA datasets, third party authoritative data, and custom user data; and 5) Increase automation for common, baseline, or bundled analyses.

An Analysis of Alternatives (AoA) was performed on over 30 technology options to determine which would best meet the five requirements categorized above. Weighted evaluation factors were applied to over 100 sub-requirements across each of the technology options, with top recommendations identified for each category. An overall system architecture for open source modules within a web-based framework is also proposed, allowing for adaptability to structure the recommended technologies within the architecture as needed. A final recommendation is made for existing Hazus functionality and source code to be re-architected to open source modules according to the optimal technology for each function, then structured in a component MicroServices container environment. For some of the recommended technologies, proofs-of-concept (PoCs) are discussed to better inform future OpenHazus design sessions.

A number of next steps are proposed as a result of the AoA, architecture solution, and PoCs, including: 1) Conducting a working session with FEMA to determine the sensitivity of, and possible changes to, the weighting criteria used in the AoA; 2) Completing all proposed PoCs; 3) Phasing OpenHazus design to begin with inventory updates and flood methodology including a lightweight desktop utility for secure data users, the latter already in progress through the PoCs; 4) Beginning development and deployment of the loss modeling library to provide Hazus results to a broad online audience; and 5) Beginning to establish governance and policy for management of open source data and code.

2 Introduction

Hazus is the standard loss estimation methodology program for natural hazards, used and published by the Federal Emergency Management Agency (FEMA). Since its inception in the 1990s, Hazus has provided stakeholders with a nationally standardized, peer-reviewed, and scientifically robust methodology for estimating potential natural hazard risks. Primary stakeholders include decision makers, such as state and local elected and appointed officials, who provide the leadership and resources to apply Hazus for risk assessment as well as practitioners like state and local emergency managers, planners, and floodplain managers, who are directly responsible for mitigation planning and other emergency management tasks. However, Hazus has faced many challenges as an academic program developed decades ago, and now must undergo a revitalization to remain relevant both from a scientific and technological perspective. This document outlines recommendations for a new OpenHazus platform, which should primarily be a web-based management system for open source Hazus components. Not only will this change make Hazus more viable as an industry-standard disaster product, it will meet numerous long-term goals from the program as outlined in the 2016 Hazus Strategic Plan. It is important to note that Hazus, and subsequently, the proposed OpenHazus solution, is a highly complex set of tools which include academic hazard and engineering science, advanced computational methods, and a diverse collection of coding languages and technologies. This document outlines requirements and technology recommendations for OpenHazus. It is not intended as a technical design document, but it does encompass all of these components to present OpenHazus as a single, comprehensive system.

2.1 Purpose

The purpose of this document is to define the functionality, interfaces, performance, attributes, and design constraints of OpenHazus, the proposed future of FEMA's Hazus risk assessment software. The hypothetical end product of this effort has been coined "OpenHazus" to denote not only a programmatic shift to open source software and technology, but also a movement toward datasets, methodologies, and model functionalities that are more accessible and transparent for Hazus stakeholders. This document includes sections addressing current challenges, future requirements, architecture, technology evaluations, and prototyping.

Table 1: Programmatic Goals from the Hazus Strategic Plan (2016)

Goal	Objective	OpenHazus
Enable Hazus access and usability for a broader group of users	• Consolidate Hazus online resources into a single portal	☑
	• Standardize and simplify Hazus results	☑
	• Provide nationwide basic results for earthquake, flood, and hurricane online	☑
	• Create and maintain online repository of Hazus results and products driven by user submitted content	☑
Ensure Hazus software is reliable, scalable, and up-to-date	• Increase Hazus stability	☑
	• Integrate subject matter experts in the development process to maintain modeling accuracy	☑
Continue to update methodologies with the latest established science	• Perform routine assessment of methodologies to ensure they reflect the latest science	
	• Perform post-event studies comparing Hazus modeling against real world results to assess methodology accuracy	
	• Create national resource for developing standardized risk assessments	☑
Further engage the community	• Ensure training is robust, applicable, and timely, leveraging real-world scenarios	
	• Perform routine stakeholder analysis	
	• Increase transparency by communicating the development and release schedules, modeling changes, and other Hazus updates	☑
Establish a structure for updating, maintenance, and implementation of all Hazus elements	• Develop performance metrics	
	• Draft a data management plan	
	• Coordinate supporting documentation and training materials with software releases	

2.2 Scope

This paper accomplishes tasks outlined by the FEMA Hazus Program in 2018 in preparation for a potential open source transition for Hazus. These tasks include identifying key motivations for redesigning Hazus in an open source, web-based framework, outlining baseline requirements for OpenHazus, and exploring available open source and web technologies for feasibility in OpenHazus architecture.

2.3 Background

Hazus serves as one of the only freely available and technically accessible loss assessment tools for local, state, and regional planning officials. Hazus-based risk assessments serve as the backbone of local and state mitigation plans nationwide. However, for many users, the design and functionality of Hazus have become outdated as interest in risk information has widened and the technical landscape for data analysis has dramatically shifted. Planners and policymakers need risk information that is interpretable by a broad audience and delivered in efficient and universal data formats. Technical analysts and hazard specialists need risk modeling tools that are customizable and efficient with fully documented methodologies. While Hazus in its current form meets the minimum of these demands, an open source, web-based version of Hazus could meet these needs more fully and evolve more efficiently as new needs arise. This paper explores possible approaches and outcomes for an open source redesign of Hazus aimed at broadening its scope and better supporting its role as a leading publicly available risk modeling platform.

In an age of increasingly frequent and extreme weather events, applied researchers in a range of social and natural sciences are now dealing with the complexities of long-term planning, infrastructure protection, public health, and social vulnerability. This increased interest in quantitative hazard risk assessment has widened the scope of academic communities working with Hazus and, in turn, increased the amount of expert input available for the continued improvement of Hazus modeling capabilities. However, the development of Hazus as centralized desktop software requires an update cycle structured to deliver periodic, costly releases. This approach does not support the efficient integration of new hazard and loss modeling methodologies proposed by hazard and risk researchers on a rolling basis.

An open source collection of independent analytical tools would dramatically decrease the resources required to implement methodological upgrades to Hazus and make it easier to solicit contributions from subject matter experts because they can now comment and improve upon methodologies that are available at source code level. This will build the end user's confidence in Hazus results and increase the software's accuracy. One of the main advantages of OpenHazus will be the establishment of a community of "power users" with development capabilities that, like in many Free and Open Source Software (FOSS) projects, handle some software support duties by participating in code maintenance activities on repositories like GitHub and bug tracking on forums like StackExchange. This community of expert contributors can act as a conduit between users and developers and efficiently extend Hazus development resources. This type of user is already active in the Hazus community, but is prevented from participating directly in Hazus development activities by the program's closed source software release cycles. The increased transparency and accountability provided by a more significant web presence for Hazus development and deployment should improve stakeholder confidence significantly, and would align FEMA's Hazus Program with the broader initiative to shift DHS applications to cloud technologies, as announced by the DHS Chief Information Officer in May 2018.

Since the Disaster Mitigation Act of 2000 (DMA 2000), demand for quantitative risk information has increased alongside a widening user base, evident in new regulatory and non-regulatory risk products delivered by the Risk MAP program and incentives offered by FEMA programs for communities or projects that implement data-driven decision making towards the reduction of risk (https://www.fema.gov/risk-map-flood-risk-products). However, the Hazus Program has not effectively distinguished between non-technical users who benefit primarily from the risk information offered by Hazus *results* and technical users who benefit primarily from the ability to *run Hazus analytical models*. An open source redesign of Hazus should seek to better meet the separate needs of these different user communities. OpenHazus will have a single online resource where planners, project managers, and policymakers can access summarized risk data produced by past Hazus studies at varying geographic levels nationwide. This online portal will make Hazus results for probabilistic and deterministic scenarios as well as historic events discoverable and downloadable for all hazard models, with updates scheduled at a set frequency, thereby considerably widening the user base of Hazus. Standardized, user-tested visualizations for results will allow complicated technical model outputs to be easily digestible and better understood by the end user, bridging the gap between risk analysts and decision makers. Redirecting non-technical users toward a more immediately useful risk assessment resource will allow for the decentralization of Hazus analytical capabilities on a separate (but connected) platform geared toward more experienced data analysts.

As the audience and demand for quantitative risk information have grown, the technical infrastructure available to derive such information has drastically changed. Cloud computing capabilities have become standard, data visualization tools are numerous and designed for non-technical users, and traditional GIS tasks are increasingly integrated into generalized analytical workflows with fewer distinctions between spatial and non-spatial processes. Hazus dependency on Esri software was initially an advantage for Hazus because the majority of technically well-versed users had GIS experience and Esri's ArcGIS Desktop platform has been the most widely used GIS in the United States for many years. However, as the community of stakeholders working with spatial data spreads beyond traditional GIS users, the need arises for visualization and analysis tools that effectively integrate spatial and non-spatial data in systems that are agile, accessible, and transparent – regardless of their interface.

A 2018 inventory found that the majority of Hazus code (over 500,000 lines) does not employ geospatial analysis. In fact, the earthquake and hurricane models do not make calls to any Esri geoprocessing tools. While maps are a highly efficient tool for communicating natural hazard risk information, a large fraction of Hazus model results are more effectively communicated through traditional visualization techniques or their modern interactive counterparts (charts, graphs, infographics, etc.). The tight coupling of Hazus with ArcGIS Desktop places a significant burden on users who want to customize the software and extend its capabilities, and on the Hazus Program itself, whose resources are largely spent on cumbersome software releases that employ complex code changes to address otherwise straightforward functionality upgrades. Hazus is so tightly integrated with the dynamic link libraries of ArcGIS Desktop that a switch to ArcGIS Pro would require a complete rewrite of Hazus.

If GIS capabilities represent a small and burdensome fraction of Hazus functionality, it becomes practical to leverage available open source geospatial libraries to accomplish spatial data visualization and analysis, rather than designing all Hazus capabilities around a GIS interface. In addition to decreasing development costs and increasing customizability of Hazus methods, reducing the role of geospatial processing will substantially improve runtimes because Hazus will leverage the increased capabilities of large, tabular lookup tables and stored, pre-computed damage factors. Similarly, a relatively small percentage of Hazus users require a desktop environment for their risk modeling projects. Users with a legal obligation to maintain secure risk modeling data inputs in a desktop environment (actuaries, certain military planners, etc.) represent an estimated 5% of Hazus use cases. Although FEMA has yet to develop a cloud environment capable of hosting, processing and exchanging large amounts of public data in a variety of formats, DHS has announced an intended shift in computing technologies away from local environments and onto cloud resources.

Transitioning Hazus to a web-based architecture could fulfill 95% of user needs while increasing modeling speed and access and radically expanding the transparency of model methodologies, data, and results. Leveraging increased storage capabilities on the cloud and increased processing speeds of web servers will also eliminate current constraints on desktop inventory database sizes, including the requirement to use aggregated Census Tract or Block data. A key objective of the OpenHazus initiative is to provide nationwide site-specific risk analysis as a default capability, which will significantly increase the accuracy of FEMA loss estimates and more effectively communicate risk and vulnerability in order to drive cost-effective mitigation strategies. OpenHazus will have an easily searchable online repository for users to share local structure data, best practices, and risk assessment results. This requires the development of a quality control process to vet new data in collaboration with within FEMA and among FEMA partners.

OpenHazus has been conceptualized as a toolbox of multi-hazard methodologies that may, but do not have to, use GIS, to visualize the spatial distribution of risk to given assets. Removing the dependency not only on Esri GIS software but also on traditional GIS architecture, in general, can significantly broaden the range of potential OpenHazus users. Hazus users and developers would benefit from two different aspects of the proposed "toolbox" of open software architecture.

In a highly modularized system leveraging as-needed open source analytical libraries, each tool is well documented and can be accessed individually. Users can employ only those Hazus modules that make sense for their projects, and developers can more efficiently expand and update Hazus functionality by accessing and updating separate analytical modules rather than editing a large, rigid and interdependent workflow. For example, researchers developing tsunami casualty estimation methods that consider debris paths can incorporate this added functionality into an extended version of the existing Hazus tsunami casualty module.

OpenHazus will also be designed to seamlessly integrate hazard inputs from authoritative federal agencies, including latest state-of-the-art flood, earthquake, hurricane, and tsunami hazard models. This will increase the quality of risk analysis results and eliminate the need for hazard modeling capabilities to be embedded within Hazus. These embedded processes have proven costly to maintain and consistently lag behind the latest available hazard modeling techniques.

Another beneficial aspect of an open architecture is the reliance on FOSS, which would make it easier to customize Hazus according to user needs for specific risk assessment projects. As advanced users develop customized versions of Hazus analytical processes, the Hazus Program can devote more resources toward collaborating with these users to review new model versions and integrate them into the official Hazus architecture, thereby leveraging the largely untapped expertise of the nationwide risk modeling community. A transition to FOSS would also increase the much-requested transparency of data and methods relied upon by federal agencies, allowing OpenHazus users to access key aspects of risk model architecture, thereby increasing trust in model outcomes. For example, the Oregon Department of Geology and Mining Industries (DOGAMI) published a Python-based extension of the Hazus flood module in 2018 that allows users to more easily customize damage functions for all structures in their analysis and quickly derive economic loss and recovery time for multiple flood hazard inputs. An open source architecture for Hazus will facilitate greater direct collaboration between the Hazus Program and partners like DOGAMI to incorporate new methods based on vetted external research projects. A transition to FOSS would also increase the much-requested transparency of data and methods relied upon by federal agencies, allowing OpenHazus users to access key aspects of risk model architecture, thereby increasing trust in model outcomes.

The next steps for OpenHazus include a phased development built around a modular architecture of Hazus analytical capabilities, as described in the remainder of the document.

2.4 Document Conventions

See Appendix A for a Glossary of Terms used in this document. The term "Hazus" refers to the FEMA program and methodologies, and may include both legacy software versions and future proposed solutions. "Legacy Hazus" is used to describe the current desktop Hazus application. "OpenHazus" refers to the solution being proposed in this document. It should be noted that from Hazus' inception in the 1990s through 2013, the name of the software was written as an acronym (HAZUS-MH). This notation is no longer in use by FEMA and should only be used if directly quoting a document from before 2013.

2.5 Intended Audience and Reading Suggestions

This paper was written for review by members of FEMA's Risk Management Directorate, including the Actuarial and Catastrophic Modeling Branch, the National Flood Insurance Program, the Engineering Services Division, and others, and certain portions may be of interest to members of the Hazus, risk assessment, hazard science, and mitigation planning professional communities. The proposed integration and reliance on authoritative hazard data products will be of special interest to other federal agencies

3 Overall Description

This section describes the existing Hazus product, including the original context for development, ongoing challenges, and the current user community.

3.1 Product Perspective

Hazus is a software suite within the risk mapping, assessment and planning program (Risk MAP) of the Federal Insurance and Mitigation Administration (FIMA) of the Federal Emergency Management Agency (FEMA), which is one of over 20 component agencies of the Department of Homeland Security (DHS). Hazus provides a cost-effective method for quantifying potential social, economic, building and building losses resulting from given hazard information for earthquakes, floods, tsunamis and hurricanes. Hazus results allow users to develop policies that aim to decrease the risk of future loss due to natural hazards – a capability required by risk reduction policies developed SLTT. The overall objective of the existing Hazus program is to implement a nationally applicable set of standardized multi-hazard methodologies for estimating potential hurricane, flood, tsunami and earthquake losses on both structure-specific and regional scales.

When combined with detailed analyses and expert knowledge, the default datasets and model parameters included with Hazus provide an effective risk assessment tool. Depending on the availability of reliable hazard and inventory data for a given region or disaster, Hazus results can be used to assess damages immediately following a real event, drive planning activities for catastrophic scenarios, or inform mitigation initiatives at the community, state, regional or national level. Hazus is also used in support of DMA 2000 to develop the Risk Assessment portion of Hazard Mitigation Plans.

Subject matter experts from leading scientific, engineering, and academic communities collaborate with Hazus program staff to develop analytical capabilities for each hazard, improve the accuracy and software speed of existing Hazus tools, expand model functionalities, and foster community involvement from risk management professionals. Hazus provides the integrated multi-hazard loss estimation capabilities, varied according to user needs, hazard analyzed, and data availability. To provide flexibility, losses are estimated based on the accuracy of input data (Figure

1). Basic analyses (“Level 1”) are based on default inventory data aggregated at census geographies and parameter data provided within Hazus. Advanced analyses (“Level 2” and “Level 3”) are based on more accurate, user-supplied data for hazard, structure or damage parameter inputs. The appropriate level of analysis must be determined to meet the needs and resources of the user.

Hazus is distributed free of charge. However, it is designed to run as an extension to the commercial GIS software ArcGIS Desktop, developed and distributed by Esri, Inc. This software can be cost-prohibitive for many potential users. Esri is phasing out its development of ArcGIS Desktop in favor of a more modern architecture called ArcGIS Pro. Hazus is so tightly integrated with the dynamic link libraries of ArcGIS Desktop that a switch to ArcGIS Pro would require a complete rewrite of Hazus. While there are other reasons to pursue “open” solutions, the significant impending cost of rewriting Hazus to maintain compatibility with ArcGIS Pro has precipitated the need to redesign Hazus independent of external GIS software packages, which provides an opportunity to identify key obstacles preventing Hazus from effectively meeting the needs of users and to revisit Hazus requirements.

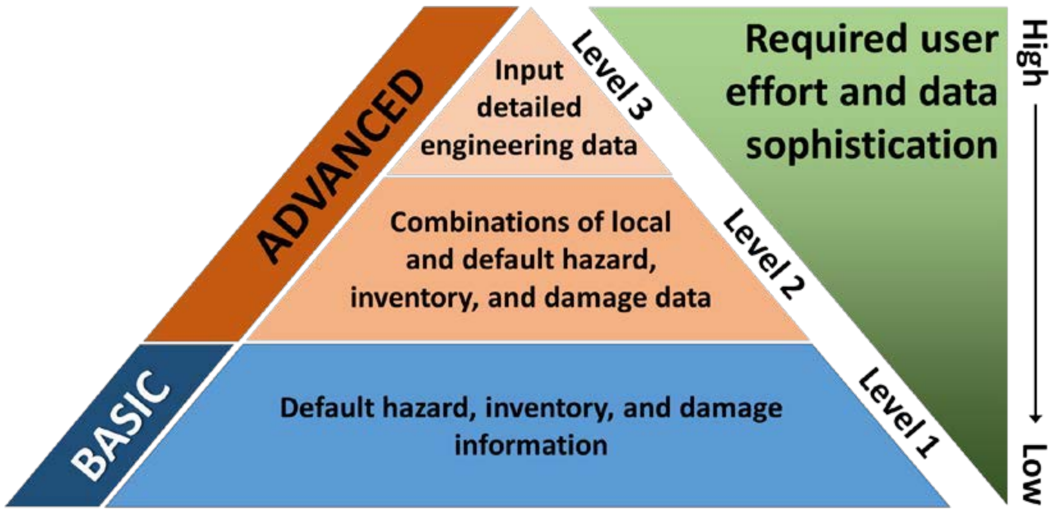

ADVANCED
BASIC
Input detailed engineering data
Combinations of local and default hazard, inventory, and damage data
Default hazard, inventory, and damage information
Level 3
Level 2
Level 1
Required user effort and data sophistication
High
Low

Figure 1: Legacy Hazus Usage Levels

3.2 Product Features

Table 2: Legacy Hazus Product Features

	Earthquake Ground Shaking Ground Failure	**Flood** Frequency | Depth Riverine | Coastal Surge	**Hurricane** Wind | Surge	**Tsunami** Depth | Momentum Flux | Runup | Velocity
Inputs				
Historic	✓		✓	
Deterministic	✓	✓	✓	✓
Probabilistic	✓	✓	✓	
User-supplied	✓	✓	✓	✓
Other supported inputs	Real-time & scenario USGS ShakeMaps	Risk MAP, User-supplied depth grids (ArcGRID, GeoTIFF, IMAGINE), HEC-RAS (.FLT)	Hurrevac, User-supplied wind files (.dat)	NOAA PMEL SIFT, State models
Direct Damage				
General Building Stock	✓	✓	✓	✓
Essential Facilities	✓	✓	✓	
Transportation Systems	✓	✓		
Utility Systems	✓	✓		
User-Defined Facilities	✓	✓	✓	✓

Induced Damage				
Fire Following	✓			
Debris Generation	✓	✓	✓	
Direct Losses				
Cost of Repair	✓	✓	✓	✓
Income Loss	✓	✓	✓	✓
Agricultural		✓		
Casualties	✓			✓
Shelter and/or Evacuation Needs	✓	✓	✓	✓
Average Annualized Loss (AAL)	✓	✓	✓	

3.3 Product Challenges

Because of its current architecture and software dependencies, Legacy Hazus faces a number of challenges. These are categorized below based on user observation, help desk reports, and tester feedback.

3.3.1 Desktop Architecture

Users must download and install Hazus in a desktop environment. Hazus software development and release cycles take place every year and accomplish a pre-established set of tasks requested by FEMA.

- Release cycles cannot accommodate ad-hoc methodology improvements discovered year-round through engagement with researchers and risk management professionals. Risk assessment technology and methods evolve at a much faster rate than Hazus software, leaving Hazus continually behind what is considered state-of-the-art in risk analysis.

- Release cycles cannot accommodate real-time bug fixes.
- Installation of Hazus releases is a cumbersome process for users that can take hours depending on security permissions and at worst days if issues persist.
- High version dependency between Hazus and other desktop products (ESRI ArcGIS, SQL Server, and Windows OS) makes installation difficult for users and development costly for the Hazus Program.
- Hazus model processing speeds are dictated by user desktop hardware capabilities.
- The storage of large input datasets and Hazus results on local machines discourages efficient sharing among users and between users and the Hazus Program, contributing to a lack of transparency and accessibility in the Hazus community.

Designing OpenHazus to be independent of desktop software would drastically reduce development costs; eliminate complications with user desktop installs, increase processing power and associated analytical capabilities, and increase transparency and accessibility of model data, methods, and results.

3.3.2 User Input

There is no structured system by which users can recommend model improvements, submit customizations, or share results from successful Hazus projects. User input is incorporated into Hazus Program activities through ad-hoc connections made between Hazus Program staff and the user community. Hazus result data are produced in formats that are either GIS-specific or stored in databases that are difficult for users to access.

- Customized or improved analytical processes or datasets are not effectively incorporated into Hazus software.
- Hazus project results are not effectively shared among the user community.
- Hazus user community efforts remain siloed instead of collective, risk assessment efforts are often duplicated, and decision-makers remain largely unaware of Hazus capabilities.

3.3.3 User Categorization

Hazus users have been categorized primarily according to the “level” of analysis they perform (basic or advanced), encouraging a broad base of risk management professionals to be trained to perform basic Hazus analyses, while more technically experienced users are trained for advanced analyses.

- Significant resources are spent providing Hazus model training for non-technical members of the risk management community who are more interested in applying and communicating Hazus results than deriving them.
- The subsets of Hazus users who primarily need to interpret, apply, and communicate Hazus loss estimates (rather than generate them) are not specifically targeted by Hazus Program activities.
- Hazus results are presented in a way that can lead to misinterpretation by users without proper training

3.3.4 Credibility Management

Hazus source code is developed based on transparent engagement with the academic community through technical advisory committees, long-running partnerships with federal agencies, and consistent engagement with expert members of the risk modeling community in order to ensure state-of-the-art methodology. However, Hazus source code is not accessible by users and the provenance of Hazus model results generated by the user community is not centrally tracked.

- Users cannot verify for themselves the accuracy of Hazus model results.
- Users cannot easily distinguish between authoritative and erroneous Hazus results.

3.3.5 Incorrect, misinterpreted or out-of-context Hazus results are often circulated among the risk management community, which erodes Hazus Program credibility GIS Dependence

The traditional GIS architecture of Hazus obscures the technically complex processes of hazard risk modeling behind a map interface. Users with GIS specializations are expected to undertake risk assessment projects that require professional or academic expertise in engineering, earthquake physics, structural dynamics, and advanced statistics, while experts in these areas without GIS skills are prohibited from leveraging Hazus for their risk modeling projects. As data analysis skill sets become more common throughout disaster-related disciplines, the risk modeling community has grown to include those who are interested in estimating hazard impacts but are unfamiliar with traditional GIS technology. The reliance of Hazus on desktop GIS software alienates these potential users. A more agnostic risk modeling platform – one that seamlessly incorporates both spatial and non-spatial components – would address the needs of a much broader user base.

A small fraction of current Hazus analytical processes are uniquely geospatial. These include traditional desktop GIS functions for clipping, mapping, projecting, and developing flood-related rasters. Geoprocessing steps dedicated to developing flood hazard data will be eliminated by the OpenHazus emphasis on external authoritative hazard datasets as model inputs. Aside from these geospatial analysis functions, the primary role of desktop GIS software in Hazus is the spatial visualization of data inputs (inventory, hazard) and model results (losses). As spatial and non-spatial data analytics become increasingly integrated, mapping functionality is more logically provided to users under a larger umbrella of data reporting tools. The relatively simple geospatial analysis and map functions required by Hazus can be easily replaced by open source geospatial libraries (Fiona, shapely, gdal, javascript, plotly, etc.) that meet these needs independent from a traditional desktop GIS architecture, and can more easily evolve along with geospatial technologies.

3.3.6 Flood Hazard Generation

The lack of sufficiently detailed flood hazard data nationwide has created a significant gap in probabilistic flood risk that no authoritative agency has yet filled. Hazus developed an internal flood depth grid generation tool in order to partially fill this gap. This capability leverages simplified hydrologic analysis to derive a rough estimate of flood depths in gridded form. Hazus loss estimation based on these internally generated depth grids have been shown to have low accuracy, which has eroded trust in Hazus as a credible flood risk analysis tool. Hazus internal flood hazard generation processes are also computationally intensive and exceedingly slow to complete, which, when combined with their inaccuracy, results in significant user dissatisfaction. Computational requirements for flood hazard generation pose a singular restraint on Hazus processing speed, since the remaining steps in a risk analysis – identifying hazard severity at inventory locations, damage, loss, and impact estimation, as well as the generation of report elements that are simple tabular calculations handled easily by lightweight desktop utilities or web services.

3.3.7 Outdated and Generalized Inventory Data

A significant obstacle to a more widespread use of Hazus is the amount of effort it takes to create inventory data. Developing the high-quality data required for an accurate and applicable risk assessment is without question the costliest part of performing a risk study, and often cost-prohibitive. FEMA has developed an incomparable resource of some 16 GB of generalized inventory data that are editable through the custom-designed Comprehensive Data Management System (CDMS). While these data could be a good starting point for authorities who do not have the personnel resources to develop their own inventories, they are provided only as backend SQL SERVER tables during routine Hazus installation and are not accessible to average users for custom analysis or visualization. Furthermore, demands for accurate risk information and targeted risk reduction have outgrown census tract-level analyses and with the availability of free building-level data, expectations have grown for FEMA to leverage a centralized national building data set for risk assessment.

Hazus contains baseline essential facility inventories for the U.S. provided in downloadable State databases in SQL SERVER format that will soon be sourced from the Homeland Infrastructure Foundation Level Data (HIFLD) program. These baseline inventories provide the user an out of the box capability to run Hazus analyses anywhere in the U.S. However, updating these static data and disseminating to users is a lengthy manual process, so many of the facility layers in Hazus have not been updated since they were developed in 2001.

Hazus demographic data updates are also completed manually, following decennial census releases. However, agencies like the Census Bureau provide dynamic data feeds through web Application Programming Interfaces (APIs) that would ensure constantly up-to-date demographic information, provided that the Hazus program maintains consistent collaboration with external agencies serving authoritative datasets necessary for accurate risk assessment..

3.4 User Classes and Characteristics

Hazus user classes were differentiated based on frequency of use, subset of Hazus functions used, technical expertise, and experience according to standard practices for user classification in IT system. It is important to capture these user classes and characteristics to ensure any future implementation of Hazus accurately captures the breadth of Hazus requirements across its broad and diverse user communities, and the description of ways in which users engage with the software is a standard practice for the initial phases of any software redesign. Please see Appendix B for full use cases based on user type.

The Hazus user community is large and diverse, with an estimated 10,000 users globally according to a internal program survey completed in 2013. As FEMA's standardized method for estimating damage and loss due to natural disasters, the intended audience initially for Hazus was emergency managers who work in mitigation planning. However, as indicated in previous sections, the wide range of functionality available within Hazus makes it a useful tool for many other sectors and industries both within and outside of emergency management. More recent surveys of the user community have found that nearly all attendees at FEMA-certified training events come from an emergency management job role, whereas a separate survey of Help Desk tickets found that Hazus users seeking assistance through the Help Desk are almost evenly distributed across job categories, sectors of emergency management, and even usage level within the software functionality. See Table3 for examples of job categories and sectors.

Table 3 shows possible user categorizations and sub-categories, which may be overlapping and cyclical according to where the user works, their specific job roles as related to Hazus, and training received in GIS or SQL SERVER platforms. See Appendix B for user profiles, developed according to industry standards for use cases. The user profiles in Appendix B provide notional examples of different Hazus users and demonstrate how one users can map to multiple user categories simultaneously.

OpenHazus provides an opportunity to rethink the classification of Hazus users in light of their overlapping and shifting roles. If baseline Hazus model results are provided in a user-friendly online platform, users seeking only these results will be able to view and download them without running their own analysis, as is currently required in Legacy Hazus. OpenHazus analytical modules can remain for users still requiring customizations, either in damage functions, inventory data, or hazard data specific to a scenario or community. OpenHazus will be specifically designed to accommodate this flexibility.

Table 3: Possible User Categorizations

Hazard	Job Category	Emergency Mgmt Phase	Hazus User Level
Flood	Local Government	Mitigation	1: No user-defined data
Hurricane	State Government	Preparedness	2: Imports user-defined inventory or hazard data
Earthquake	Federal Government	Response	3: Imports Level 2 data, plus reviews and modifies engineering/damage parameters
Tsunami	NGO/Research	Recovery	
	Government Contractor		
	Private Sector		
	Academic		
	International		

3.5 User Documentation

Comprehensive documentation, including User Manuals and methodology documentation (Technical Manuals) are available at: https://www.fema.gov/media-library/assets/documents/24609. Please note, online documentation varies in age and relevancy. For up-to-date documentation on known bugs or discrepancies between documented methodology and Hazus source code, please see the User Release Notes specific to each release, also available at the link. Training videos are currently in process for being posted publicly to the FEMA YouTube channel and fema.gov.

3.6 Assumptions and Dependencies

Challenges with Hazus in its current form have been discussed in this section; moving forward, this document will address OpenHazus requirements and recommendations for addressing these user needs and existing challenges. The following assumptions are made when discussing these options:

Hazus is a loss estimation tool. This is the core application of OpenHazus that takes precedence over other potential demands in supporting the mission of FEMA in general and of the Risk MAP program in particular. Any rewrite of Hazus will have to fulfill its current set of capabilities related to the estimation of social (fatalities, displacement and injuries), infrastructure, building, and economic losses due to natural hazards. OpenHazus implementation will emphasize the use of external, authoritative hazard datasets.

Hazus is free for users. OpenHazus as a methodology toolset shall remain free, as should access to core national datasets. OpenHazus server traffic can potentially be prioritized according to user types, such that users with secure data requirements have additional levels of security, or users in active response situations with time-sensitive analysis requirements can receive priority.

Hazus is publicly available. Hazus methodology and data shall continue to be publicly available in the form of accessible download files, user manuals, technical manuals, and analytical code.

4 System Features

The following section describes high-level requirements for OpenHazus. These were derived first from a brainstorming session held in November 2018 between Risk MAP CDS and FEMA. The features identified from this initial session were further refined into categories, and then itemized within each category to provide input to an Analysis of Alternatives (AoA). The Boulder features and detailed category breakdowns are provided in this section, with the itemized requirements listed in Appendix C. Each category listed below the Boulder features provides a brief description for how the categorized items were weighted in the AoA tables. The weighting is based on a scale of 0 to 100, with a weighting of 100 defined as a necessary feature.

4.1 Boulder Features

In November 2018, the Hazus Team met in Boulder, Colorado to outline the System Features envisioned for OpenHazus. A set of eight objectives for a hypothetical OpenHazus system were established at this meeting:

① Data sharing platform. In order to better meet the needs of non-technical users interested in risk model results, OpenHazus will include an online platform for viewing, sharing and downloading generalized but authoritative risk assessment results from across the U.S. This includes a standardized way of storing and accessing metadata associated with available risk assessment results, including the lineage of how the data were created – even if this happened outside of OpenHazus. The standards requirements for metadata, as defined by the Federal Geographic Data committee (FGDC) and Open Geospatial Consortium, must be adhered to. A first expansion of these data sharing platform capabilities includes a seamless link between OpenHazus and FEMA's Mapping Information Platform (MIP) and Map Service Center (MSC) for discovery and integration of flood hazard data to better integrate FEMA's risk assessment and mitigation programs and investments in flood hazard data.

② Support import, dynamic integration, and discovery of authoritative hazard data. In the spirit of openness, OpenHazus must be able to integrate authoritative data from external providers, including the USGS, NOAA, as well as FEMA's own flood hazard engineering data. Users must be able to explore existing hazard datasets from authoritative agencies in their study area and select their desired hazard source for direct integration in their analysis, without downloading or cleaning the data. Parallel to that, stakeholders should be able to upload user-defined hazard data with standardized QA/QC measures to a staging area or library, where it can be shared more broadly through a public domain repository with FEMA's stamp of approval after it has passed internal quality control.

③ Modularization. One of the advantages of a thorough overhaul of the Hazus software architecture is the possibility to create a library of modules that encapsulate individual processing steps. Each significant component of Hazus, such as the debris or shelter modules would be separated with the ability to run independently. Basic users can choose to interact with only those analytical steps that apply to their project goals, rather than adhering to a fixed workflow (though user-friendly workflow suggestions will be provided.) Experienced users and developers should then be able to create their own workflows, adapting them to agency and application needs.

④

User-friendly GUI. The current user interface is an extension of GIS desktop software. With its separation from GIS, the OpenHazus user experience can be improved significantly with expanded options for interactive data visualization. Whereas Hazus was always conceived as a desktop (workstation) software, OpenHazus should be flexible enough to serve stakeholders in a web-based, desktop-based, and even mobile environment. Not all functions will be available in all environments, but the transition should be seamless from a UI/UX perspective.

⑤ Secure data integration. An estimated 5%of Hazus stakeholders are working with PCII and PII data that require secure storage, access and communication mechanisms (estimate based professional experience of OpenHazus team, including lead developers and program manager).Conversely, the vast majority of OpenHazus stakeholders will be better served by web services. As OpenHazus is built around the latter, the needs for secure or private data integration on desktop and mobile computers must be addressed without impacting the UI/UX design of the majority.

⑥ Improve Hazus inventory data. Hazus has historically provided a baseline inventory dataset for the nation so that Hazus Level 1 capabilities are available out of the box. However, much of this data is aggregated at census block or tract levels introducing inaccuracy in loss estimation capabilities that can be addressed with a future dataset provided using a site-specific inventory. The ability to accurately intersect the hazard information at the location of each inventory item will enhance OpenHazus loss estimation quality over existing methods of using area-weighted aggregated block data or census tract centroids. The additional challenge is to ensure we can provide reasonable estimates for Hazus building required attributes. As specified in requirements ① and ②, in which users must have access to risk model results and authoritative hazard data inputs, establishing OpenHazus in a web service environment should facilitate users uploading their own improved inventory data into access-limited sub-clouds. As with the flood data in ②, a process should be established that allows merging higher quality inventories with the main repository.

⑦ Must be accessed through web. For most stakeholders, i.e., those who are not working with private or secure data, most of the data and all of the processing should occur on a server accessible through a web-browser. Hazus desktop installs are cumbersome and require significant user support and user resources. Providing broad access of Hazus capabilities through the web is essential for users. In addition, Operations and Maintenance, including defect fixes and enhancements are expected to be far more efficiently enables through a web server platform. The restricted desktop version should be a thin client, with few restrictions on compatibility or operating systems.

⑧ Ability to automate workflows. A number of model runs can take quite a while to execute, especially if users want to run an ensemble of scenarios. It is therefore desirable to have the ability to run complete workflows in batch mode without any user interaction.

4.2 Desktop Independence

Critical to OpenHazus is maintaining the existing functionality while expanding into a 21st century computing environment. The inventory data, as well as the Risk MPA flood hazard data holdings in the MIP and MSC, should now be directly accessible for any visualization, analysis and reporting function of OpenHazus, and have therefore been given the level of a necessity (Weight of 100, see Appendix E). Not absolutely necessary, but highly desirable, are password-protected repositories of user-defined inventory data and access to authoritative input data for hurricanes, earthquakes, and tsunamis (weights of 80-90, see Appendix E). Of similar importance is the ability to integrate user-defined input data.

The current version of Hazus contains hazard generation modules and allows for import from third party hazard generation modules for probabilistic hurricane hazards as well as user-defined earthquake and tsunami events. These may not be part of the first generation of OpenHazus but should be prioritized for later versions in the near future, which is why these priorities were assigned a weight of 50-60.

In addition, there is demand for OpenHazus also serving as a repository for source code modules, the results of scenario runs, the distribution of user-run results, and possibly even for continued storage of analysis results that are meant to be kept from public consumption but are accessible in user-specific parts of an OpenHazus cloud repository. These desirables were given a relatively low weight of 15-30.

For the relatively small audience of some 5% of Hazus users who need an offline solution, the agility of the client desktop application is essential. Having a relatively thin client for this limited audience is important enough to warrant a weight of 100.

One of the advantages of moving away from the Esri platform is that it opens the realm of other operating systems. MS Windows support remains essential (weight = 100) but in light of supporting as wide a range of open source solution, Linux assumes the same level of necessity. Mac OS and the Chrome operating system have been assigned lesser but still significant weights of 80 and 70 respectively. iOS and Android support, on the other hand is deemed to be for specialty applications only, which is why the need of support for these has been assigned relatively low weights of 20 and 30 only.

- OpenHazus shall operate primarily in a web environment.
- The OpenHazus web environment shall provide users access to interactive analytical modules and required input.

- The OpenHazus web environment will connect to related interactive resources like a source code repository, baseline inventory data, authoritative hazard scenario data and Hazus results.
-

4.3 GIS Independence

One of the driving factors for the development of OpenHazus is to gain independence from Esri, while also broadening the Hazus user base beyond traditional GIS analysts. It is important, however, to ascertain that this does not come at the price of diminished functionality. Maintaining continued support for legacy data formats are a must – both in the realm of raster and vector analysis. While not all future users may choose to employ a map interface, having the continued opportunity to do so is essential to the success of OpenHazus. All of these criteria have therefore been given a weight of 100.

The only aspects deemed slightly less important are that study regions can be created anywhere in the world (this is not a limitation for any of the software packages investigated), and that the user interface is adaptable for possible future mobile applications. Most of the software packages studied are not constrained here either; the lesser weight of 70 for these two criteria turned out not to be a serious restriction.

- OpenHazus will be designed independent of proprietary GIS technology. OpenHazus will accept input data in spatial data formats, so that traditional desktop GIS technology may still be used by stakeholders for preparing advanced inventory or hazard data, as well as subsequent analysis, visualization, and interpretation beyond the scope of Hazus software.
- Traditional GIS interface functions will be utilized only when required for loss estimation (clipping, intersecting, projecting, etc.).
- Analysis components requiring spatial operations will be rebuilt using open source geospatial libraries that can be run in a cloud environment or on a local machine through a lightweight, downloadable OpenHazus utility.

- Study regions shall be created using a spatial or tabular interface, allowing the user to select the aggregation level and geographic location from a selectable map or menu of jurisdiction names. Study region creation shall be global and only limited by geography where hazard dictates.
- Study regions and model results for all hazards shall be viewable through an interactive, selectable map or tabular menu interface.
- Results should be viewable through both maps and traditional data visualization techniques. Spatial visualization functions will be accomplished through open source libraries and seamlessly bundled with non-spatial data visualization libraries in one intuitive "toolkit" that provides users with diverse options for interpreting and representing their model results.
- The OpenHazus interface should be flexible enough to serve stakeholders in a web-based, desktop-based, and potentially mobile environment. Not all functions will be available in all environments, but the transition should be seamless from a UI/UX perspective.

4.4 Modularization

Hazus has become somewhat unwieldy as it has grown as a family of code developments. Any major revision of the software, even without the emphasis on "open" would be well-served by a redesign of the code base that emphasizes modularization. This becomes an absolutely necessity once Hazus is opened up to non-FEMA developers. This is envisioned to occur in phases, where an initial phase might handle core functionalities and continuation of current requirements, while later versions accommodate interaction with third party software and developers.

Based on these criteria, all code bases that deal with default inventories, spatial calculations, and the map-based user interface were assigned top weights. The same holds for the need to follow standard procedures for documentation and code maintenance; this is an industry best practice and must be enforced from the initial migration of code, even if it will be made available to the OpenHazus developer community only in years to come.

Weighting procedures often suffer from the view that "everything is important" and hence everything should receive top scores. In light of a phased approach to developing OpenHazus, it is recommended that calculations of social, economic, building, and infrastructure losses, as well as the requirement to address defects and enhancements on a rolling basis across individual modules are given a weight of 80 and all other aspects of modularization, including the support of third-party functionality receive a weight of 60.

- Hazus source code shall be divided into encapsulated analytical modules according to hazard, functionality, data inputs, usability, etc., with major components rewritten in a language commonly used in the data science, research and engineering communities.
- OpenHazus analytical modules will be provided in a library for users to run individually in whatever combination meets their risk assessment needs. Basic users should be able to interact with only those analytical steps that support their project goals, rather than adhering to a fixed workflow, though user-friendly workflow suggestions will be provided.
- OpenHazus users should have the ability to design their own workflows and customize or extend OpenHazus modules to meet their needs.
- The OpenHazus analytical module library should include standard methods for:
    - Inventory data collection and updates by attribute type
    - Classifying occupancy of buildings and facilities
    - Classifying building structure type
    - Using authoritative hazard datasets
    - Importing user hazard data
    - Using ancillary datasets like demographic and elevation information
    - Extracting hazard severity measurements at inventory locations
    - Calculating damages at inventory locations based on hazard inputs
    - Calculating social and economic impacts based on damages and hazard inputs
    - Developing building damage functions
    - Grouping, ranking and analyzing lifelines
    - Visualizing and interpreting model results
- The OpenHazus development team will provide a reference implementation, such as module prototypes, that stakeholders may use out-of-the-box, with the ability for interested parties to create their own tools, models, and workflows.

- OpenHazus development activities will address bugs, methodology updates, and functionality expansions for individual bugs on a rolling basis without impacting the rest of OpenHazus code. OpenHazus developers should work directly with users as they customize modules, incorporate new hazard or inventory inputs, or experiment with new methods for calculating impacts. If user-derived modifications are applicable to a broader set of the Hazus user community, the OpenHazus development team should incorporate them into the official reference implementation as resources become available, rather than within a rigid update cycle.
- OpenHazus development should be coupled with engagement with the research community so that methodologies can evolve as the state-of-the-art advances.
- The technical requirements of each OpenHazus analytical module should be fully exposed to users so that they can more effectively leverage those modules that fall within their professional purview.
- Training materials should be reorganized according OpenHazus modules, as users will no longer be required to understand all components of a large, rigid workflow, but can instead run only those modules that are familiar to their particular industry or applicable to their particular data inputs.

4.5 Data Sharing

The emphasis on "open" in OpenHazus is at least as much on sharing capabilities as it is on exposing any source code. These sharing aspects can be partitioned into four separate requirement groups: default inventories, access to authoritative data, default results inventory, and user inventories. The tables in Appendix E list over 40 different requirements for these groups. Following the logic of the discussion in previous sections, all data sharing requirements that are at the core of existing Hazus functionality are assigned top weights. The greater the dependence on legacy third party code, or the export to non-standard GIS formats, have been assigned relatively light weights to reflect their lower rank for data sharing.

4.5.1 Results Platform

- OpenHazus will include online platforms for non-technical users to view, share and download authoritative risk assessment information from across the U.S.
- OpenHazus will include a web interface for Hazus results generated by authoritative groups performing local or regional Hazus projects.
- OpenHazus will include a web interface for nationwide baseline Hazus results generated by FEMA for each hazard using authoritative probabilistic hazard data and site-specific default inventory. Users seeking to run Hazus models using these input hazard and inventory data will be redirected to this baseline results platform.
- Users should be able to submit Hazus project results for inclusion in the local/regional results platform after an internal FEMA review for compliance with to-be-determined quality standards.
- The OpenHazus results platforms will include a standard for storing and accessing metadata associated with results from any given Hazus analysis, including the lineage of how the model input data were created, even if data creation took place outside of OpenHazus.
- Results data will be provided for download in a limited set of tabular and spatial formats. Metadata for results datasets shared on these platforms should include succinct documentation of processing steps and input data used to derive model results in order to encourage transparent communication about project methodology.
- Datasets on OpenHazus results platforms need to be discoverable by location, hazard type, and model attributes, including creator.

4.5.2 Authoritative Hazard Data Integration

- OpenHazus should not provide capabilities for generating hazard data, but instead ingest hazard data from users or from dynamic connections to authoritative agency repositories. Without hazard generation capabilities, Hazus risk analyses will be limited by the hazard data that can be provided by users or authoritative sources.

- Users must be able to integrate external authoritative hazard data directly from the OpenHazus interface, including from the USGS for earthquakes, NOAA and HURREVAC for hurricanes, DOGAMI, PMEL, and NOAA for tsunamis, and FEMA's own engineering data and other existing sources for floods.
- Users must be able to explore existing hazard datasets from authoritative agencies in their study area and select their desired hazard source for direct integration in their analysis, without downloading or cleaning the data.
- Users should be able to upload their own hazard data for integration in their analyses. OpenHazus should check uploads for the attributes required to complete analyses before allowing data to be integrated into the repository.
- OpenHazus should seek to fill the nationwide gap in baseline flood hazard data. For example, by the time OpenHazus is fully functional, the 3DEP LIDAR program is likely to be complete, facilitating completion of the USGS nationwide 1-meter DEM effort currently underway (vertical accuracy of 6-8 cm). This nationwide DEM could then provide the basis for the development of a nationwide, algorithmically-derived probabilistic depth grid.

4.5.3 User-defined and Secure Data Integration

- OpenHazus will provide a repository for user-defined inventory, hazard, and results data in a web environment.
- OpenHazus will provide the capability to upload data to a secure cloud repository or provide lightweight desktop analytical capabilities designed specifically for users who need to model hazard impacts using Personally Identifiable, proprietary, or protected information.
- OpenHazus capabilities for users with secure data will be designed separately from the rest of OpenHazus so as to not impact the accessibility and processing improvements provided for the vast majority of Hazus users through a web environment.

4.5.4 Updated Site-Specific Default Inventory

- OpenHazus should provide a nationwide site-specific risk assessment inventory for use in studies where more accurate inventory data is not available.
- The OpenHazus nationwide inventory should be readily accessible for viewing, downloading and editing in a public web interface and be easy to update by the Hazus Program or by individual users.
- Default, site-specific inventory data shall remain categorized according to physical structure type, with categories aligning as closely as possible with legacy Hazus (eg. Essential Facilities, Bridges, Pipelines, etc.).
- OpenHazus shall support timely downloads of reasonably sized default datasets.
- Blank default databases will be provided for international Hazus model adaptation.
- OpenHazus should facilitate users uploading their own improved inventory data into access-limited sub-clouds in order to leverage the many jurisdictions without budget for multi-hazard loss estimation projects, but with high-quality inventory data available for use by other analysts in Hazus. A process should be established that allows users to merge higher quality inventories with the main repository in coordination with the Hazus Program.
- OpenHazus should leverage dynamic data available through the annual American Community Survey, Longitudinal Employer-Household Dynamic datasets, and 327 Census Bureau datasets offered through a robust API.
- OpenHazus should integrate dynamically with web-based essential facility layers provided by the newly redesigned HIFLD Open program. OpenHazus development should include the deployment of scripts and support data that facilitate on-demand updates from HIFLD Open layers with automatic attribute assignment for inclusion in Hazus analyses.

4.6 Automation

Similar to modularization, automation is an important modernization aspect of moving to OpenHazus. If OpenHazus is to expand its user base, it needs to provide processing templates that novice users can run as defaults, while allowing expert users to modify these to their own needs. With the requirement to provide results to a broad group of Hazus stakeholders, the vision of OpenHazus mandates frequent updates of scenarios, risk assessments, as well as automated procedures in response to disaster events on FEMA servers without increasing staffing requirements. All aspects of automation are therefore given high weights of or near 100, with a slight decrease for mitigation strategy evaluations since manual review of these is commonly needed (weights of 85 and 80 respectively).

4.6.1 Baseline Results Repository Updates

- Analysis results generated using the authoritative probabilistic hazard data source, such as that provided by the USGS AND baseline Hazus inventory data shall be automated by FEMA for the entire U.S. to provide updated annualized analyses.
- Analysis results generated using authoritative historic or other deterministic hazard data AND baseline Hazus inventory data shall be automated by FEMA for the entire U.S.

4.6.2 Probabilistic Flood Risk Assessment

- Automation should be established to support multi-frequency probabilistic or uncertainty analyses for flood, supporting multiple depth grids for each frequency.
- Automated analysis should be established to support multiple, separate depth grids associated with individual flood types or scenarios, such as breach, pluvial and alluvial flooding scenarios.

4.6.3 Mitigation Strategy Evaluations

- Automated analysis should be established to support multiple mitigation strategy scenarios, such as freeboard, NFIP regulations and building code adoption and enforcement.
- Automated analysis should be established to support analysis of multiple depth grids associated with floodplain mitigation scenarios, such as protection measures and upstream storage scenarios.

4.6.4 Analysis and Results Dissemination for Response Events

- Automated analysis should be established to support analyses for response events, including ShakeMap events associated with a PAGER Yellow or above, landfalling tropical storms and above, as well as when flood depth grids are provided for damaging flood events.
- Automated results export and dissemination should be provided and hosted using OpenHazus providing an authoritative source for the run of record in accordance with disaster SOPs.

4.7 Challenges Addressed by Requirements

Recalling the challenges listed previously, Table 4 maps these to the categories of requirements discussed in this section.

Table 4: Challenges Addressed by Requirements

OpenHazus Requirement Category	Challenges Addressed						
	Desktop Architecture	User Input	User Category	Cred Mgmt	GIS Dependence	Flood Hazard	Inv Data
Results Sharing Platform		X	X	X		X	
Secure Data Integration							
Authoritative Hazard Data Integration				X		X	
Modularization	X	X					
GIS Independence		X			X		
Updated Site-Specific Default Inventory		X		X			X
Desktop Independence	X						
Automation				X	X		X

4.8 Software Quality Attributes

In addition to the system features discussed in this section, any evaluation of potential open source software packages needs to consider aspects of adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. These software quality attributes are distinct enough from the system requirements to warrant a separate weighting scheme. With both end users and future co-developers of OpenHazus modules, the perceived ease of use of each software package analyzed has been assigned a weight of 85. The build-and-fix frequency is both a measure of maturity as well as ease of maintenance, warranting a weight of 65. Another aspect of maturity is the foreseeable longevity of open source software packages. There are many exciting developments that cease to exist after a few years – a dangerous situation for developers of authoritative systems like OpenHazus. As a result, rather than assigning this aspect a higher weight, a score of 50 for foreseeable longevity was given to each software package received in the evaluation.

The difficulty of migrating existing Hazus analyses to an open source environment and the degree to which existing Hazus features are recognizably matched in OpenHazus was discussed briefly under the Esri Independence category. To avoid double-counting, the porting characteristics mentioned above (such as portability) were assigned relatively low weights of 45 and 50. This is somewhat countered by a strict scoring of how much development effort it would take to port functionality.

4.9 Other Requirements

The discussion of system features and software quality attributes with FEMA will likely reveal additional requirements that are not included here, either due to omission or an inability to assign proper weighting without further discussion. These will become important design features and must be discussed prior to design requirements being established. Some examples include:

Primacy of Web Environment: It will be fairly straight forward to set up the default inventory database(s) on FEMA servers. Their replication on client computers for offline use is a separate matter. It is one thing to provide the inventory data in form of web services, it is another (mostly bandwidth-related) to facilitate repeated downloads. It will be worthwhile to discuss the notion of distributed repositories similar to the CRAN sites for the statistical software R, trying to avoid standard commercial providers such as Amazon, Google, or Microsoft. Of similar importance to be able to connect with third party providers to authoritative hazard data input sources such as Hurrevac, ShakeMap, or PMEL. This could be accomplished by writing a web service to these USGS and NOAA datasets. For this to be sustainable would require, however, a standing committee that assures that any changes to the API are properly planned and communicated and goes beyond the current scope of this document. Similar considerations apply to third party hazard generation modules. Their support is certainly desirable, but it requires coordination in form of, for example, a sub-committee of the FGDC.

Governance of Modules: The move to Open Hazus gives rise to a range of governance issues, which have little to do with the choice of open source software used. Relatively unproblematic is the choice of a distributed version control system such as GitHub, Bitbucket or Sourceforge. More delicate is the question which components of Open Hazus – if not all – should be published in such a repository. Open Hazus will be a significant development effort and based on the examples of successful open source projects of similar size, it will be necessary to have an organization with a charter, a project steering committee, board of directors, user groups, voting members, and financial auditors for the non-FEMA part of OpenHazus. The range of such efforts spans histories like the directed move of GRASS GIS from USACE to the OSGeo Foundation or the self-organized creation of the QGIS Association. FEMA's role may be envisioned as changing from the former to the latter example. In either case, it would be up to such an organization to set up procedures for vetting and releasing user customizations to modules and to develop a regular cycle for soliciting module updates from hazard committees and users. The Board of such an organization would also set the standards for how to document all code bases.

Data Sharing Policies: A number of requirements in the data sharing category are more policy-oriented and have only indirect effect on a subsequent software evaluation. If Open Hazus contains a repository of analysis results generated using the authoritative probabilistic / historic or other deterministic hazard data source and default hazard inventory data that are pre-run by FEMA for the entire U.S., then it does indeed make sense to prohibit client services from re-running such scenarios. The consequence of such a policy decision is that Open Hazus will need to provide significant server resources, both on the storage and on the computing side. Much of such functionality would replicate the functionality of Google Earth Engine and it would be yet another policy decision to investigate similar usage in OpenHazus.

Supported Data Formats: Another policy-oriented decision is the notion of data formats supported by Open Hazus. Modern GIS standards such as GeoPackage, GeoJSON, and Cloud-Optimized GeoTIFF are easy to accommodate with the GDAL library. On the technical side, the same holds true for NetCDF and NASA's Hierarchical Data Format, which come in so many different profiles that the Hazus-specific implementation would have to be determined by a steering committee as indicated in previous sections.

Such a committee, or committees, would also have to determine the QA/QC procedures for staging and sharing user-generated data before it can be released to the public with the authoritative stamp of FEMA. A phased transition to OpenHazus will allow for community input with respect which – if any – legacy modules for generating earthquake and tsunami scenarios will be migrated to Open Hazus. If the notion of "open" is taken seriously, then such coding efforts might better be left to the user communities rather than seeing it as the responsibility of the core development work.

Allotted Server Space: If end users are able to store data within OpenHazus, how much server space should be allocated to each user? Similarly, can a user have multiple accounts for different projects or different agencies?

High-resolution datasets must also be considered. At an example scale of one foot resolution for the entire United States, the boundaries between the conceptual models for raster versus vector data begin to blur. Storing and processing all data in raster format could be conceivable and opens possibilities of software packages like Rasdaman, which is widely used in the European-wide CORINE project. From a coding perspective, this would simplify the rewriting of existing Hazus functionality. However, it would require a shift in mindset among users and developers who are used to thinking of building inventory data as spatial features rather than pixels.

Similarly, it might be worthwhile to consider moving away from the safe solution of PostGIS to Hadoop-or Spark-based OLAP datastores. This idea will be picked up in the following chapter.

5 Analysis & Discussion

This section evaluates the requirements derived in the previous section against possible technology options. The raw evaluations tables are available in Appendix E to illustrate the process. It is recommended that prior to making design decisions, a working session is held with FEMA to adjust the criteria and review the sensitivity of the weighting factors in selecting different technology options for the various OpenHazus components. In this section, these components are similarly discussed from an overall solution architecture standpoint, to provide a final recommendation both for OpenHazus architecture, and optimal technology solutions for the individual components.

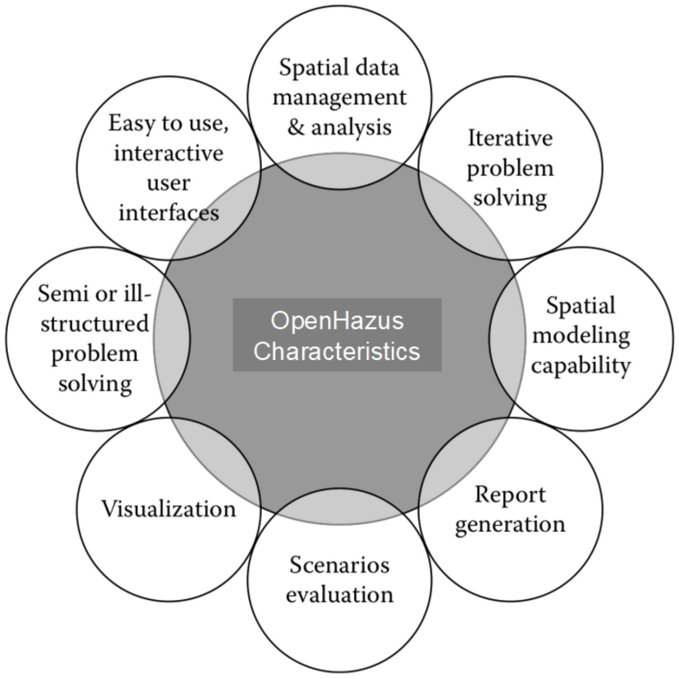
5.1 Recommended Solution Architecture

When one examines OpenHazus use cases and functional requirements outlined above, it becomes apparent that OpenHazus is essentially a spatial decision support system or SDSS. A spatial decision support system (SDSS) is an interactive, computer-based system designed to assist in decision making while solving a semi-structured spatial problem – in the case of Hazus, it is designed to assist planners and analysts in making decisions based on possible structural, economic, and social loss in a particular geography due to particular natural hazard. A SDSS is a system which models decisions to help identify the most effective decision path. An SDSS typically comprises a decision support system (DSS) and a geographic information system (GIS). This entails use of a database management system (DBMS), which holds and handles the geographical data; a library of potential models that can be used to forecast the possible outcomes of decisions; and an interface to aid the users' interaction with the computer system and to assist in analysis of outcomes. In their 2011 book, *Spatial Decision Support Systems – Principles and Practices*[1], the authors Ramanathan Sugumaran and John DeGroote provide a wealth of information on the components that encompass a SDSS.

SDSS are systems that combine analytical tools with functions available in GIS as well as models for evaluating various options. The future OpenHazus platform must be built to be flexible

[1] http://www.gisresources.com/wp-content/uploads/2014/06/spatial-decision-support-system.pdf

to accommodate various Hazus stakeholder preferences and restrictions and allow for effective user interaction in an iterative problem-solving environment. To meet these requirements, we recommend a micro-services based architecture be developed with easy-to-use graphical user interfaces and functionality for spatial database management and analysis, scenario evaluation, modeling, visualization through maps, graphs, tables, and report generation. From a Hazus-specific functional perspective, a SDSS should provide the characteristics shown in Figure 2.



Spatial data management & analysis
Iterative problem solving
Spatial modeling capability
Report generation
Scenarios evaluation
Visualization
Semi or ill-structured problem solving
Easy to use, interactive user interfaces
OpenHazus Characteristics

Figure 2: SDSS Perspective for OpenHazus

The current (legacy) Hazus application is more akin to a combined Decision Support System (DSS) and Geographic Information System (GIS). It is also a traditional desktop monolithic architecture.

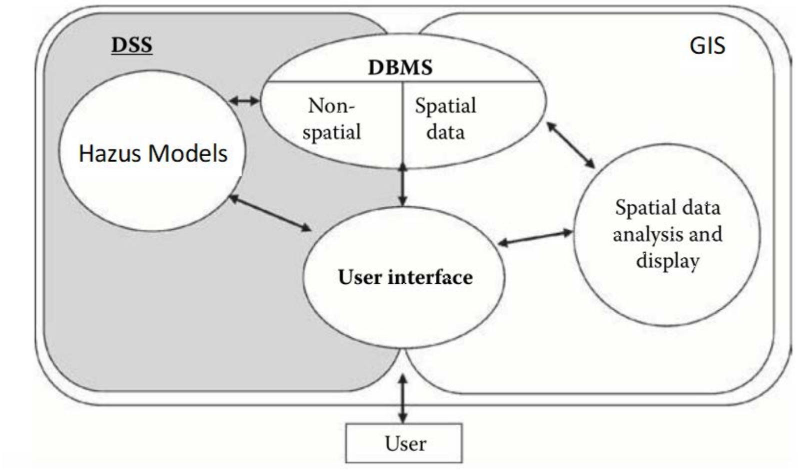


Figure 3: Legacy Hazus Architecture Components

The database component within a decision support system (DSS) mostly deals with nonspatial data collection, retrieval, management, and analysis. The Esri GIS provides spatial and nonspatial data collection, storage, management, and cartographic display functionalities. The database component in both systems feeds appropriate information to the other components as needed. These legacy services form the foundation for the envisioned MicroServices for a web-based OpenHazus SDSS platform. A generic conceptual diagram is presented in Figure 4.

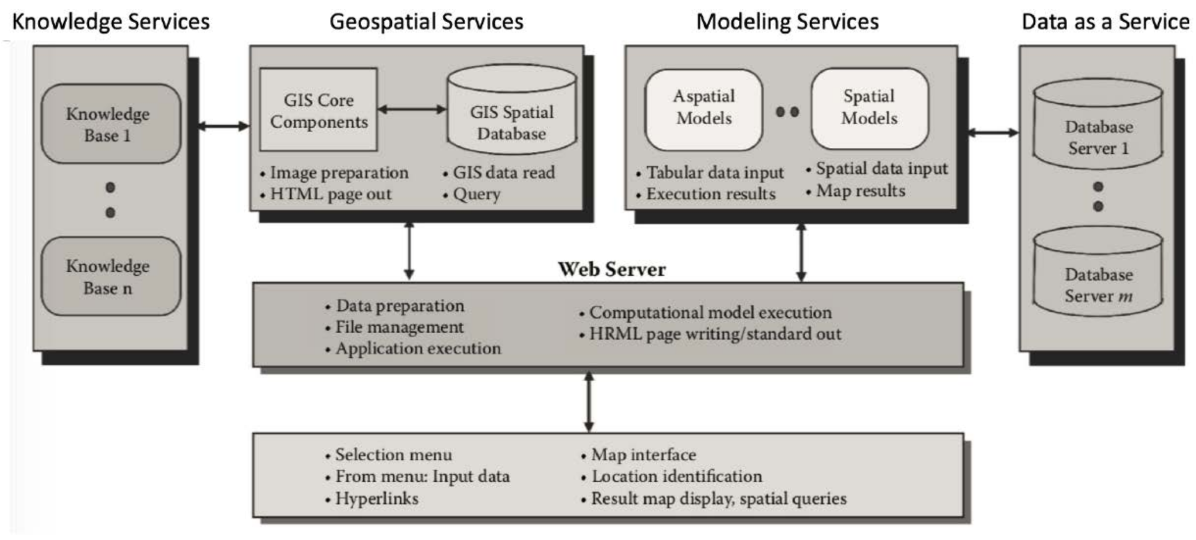


Figure 4: OpenHazus Envisioned Services

An expanded version of Figure 4 is shown in Figure 5, with Hazus-specific details added to the generic architecture.

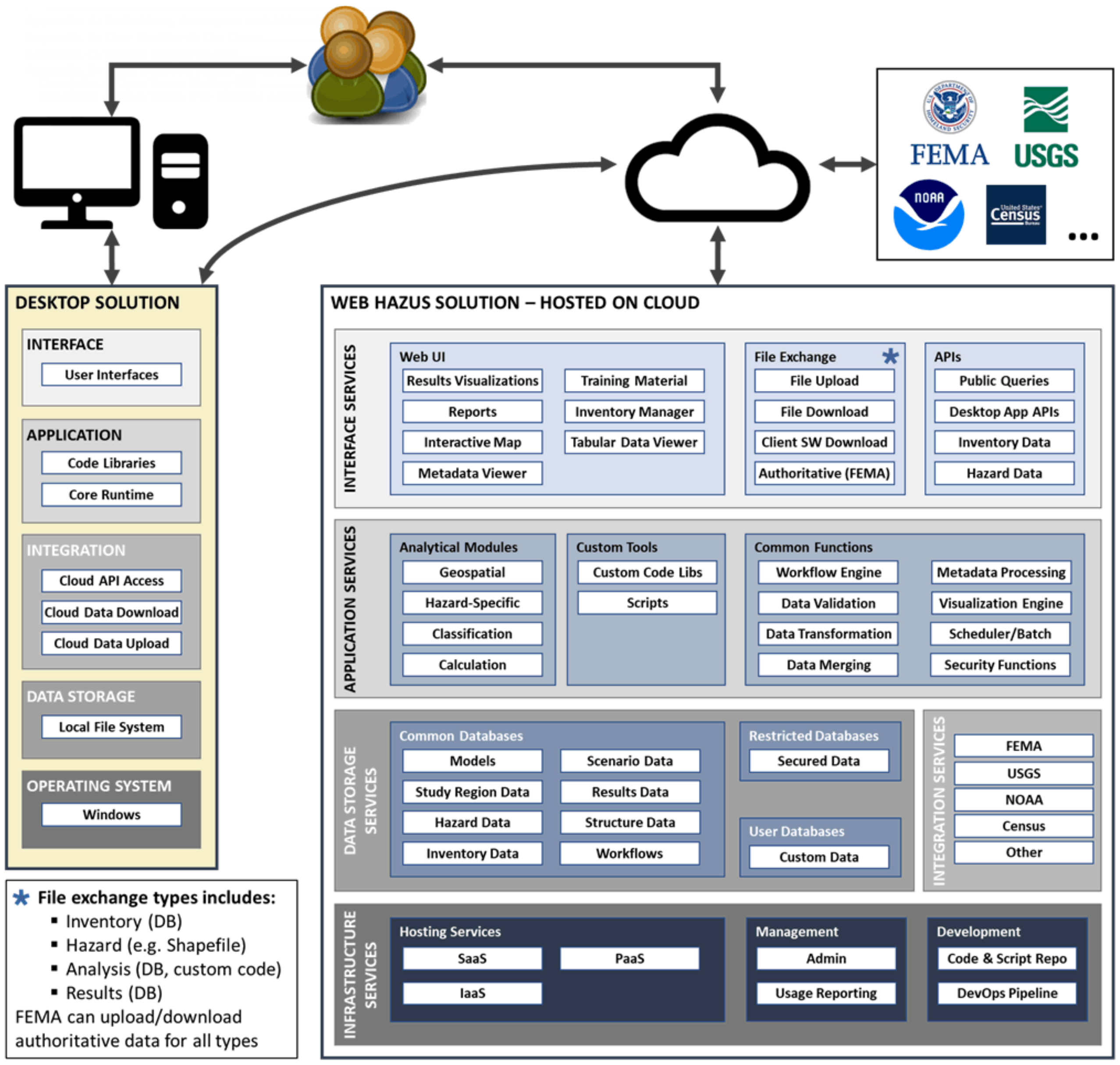


Figure 5: Proposed Web Architecture for OpenHazus

MicroServices is an application architectural style in which an application is composed of many discrete, network-connected components called MicroServices. The OpenHazus SDSS would include a collection of geospatial MicroServices to facilitate geographic data retrieval, display, and analysis. But a key element of any future Hazus SDSS platform will be an open approach to external (user defined) models (both spatial and non-spatial). A MicroServices architecture and containerized deployment approach would give FEMA the greatest flexibility (with respect to deployment options) and OpenHazus should conform to standards for published MicroServices reference architectures. The OpenHazus-specific MicroServices would be part of the 'container environment' MicroServices seen in Figure 6.

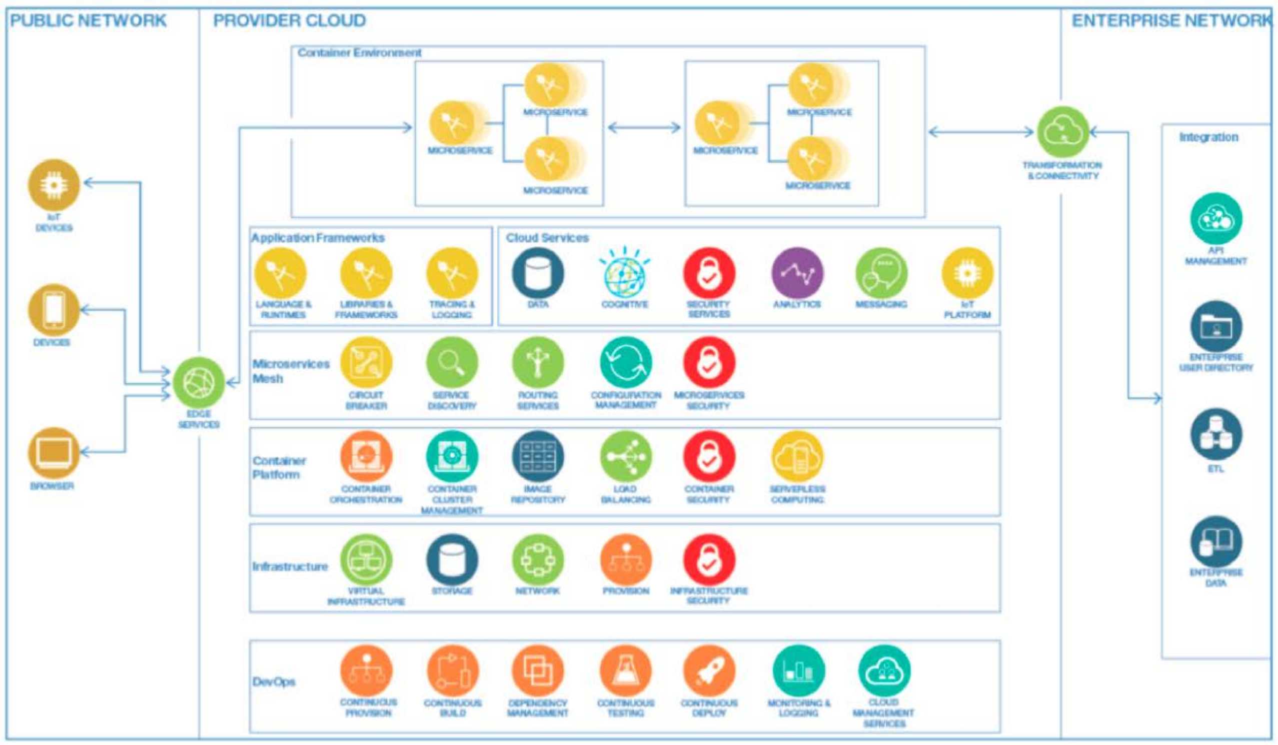


Figure 6: MicroServices Reference Architecture[2]

With respect to an open approach to creation, management, and consumption of modeling services, Colorado State University created an Object Modeling System (OMS) platform[3] which is

[2] https://www.ibm.com/cloud/garage/architectures/microservices/reference-architecture

based on open source technologies. OMS is distributed under the terms of the GNU Lesser General Public License, version 2.1. OMS is a framework for designing, building, validating, and deploying agro-environmental models, but it is fully applicable to Hazus models. The OMS framework consists of a development kit for designing, building, validating, and deploying models. It contains four platforms: (1) model development, (2) model deployment (cloud services), (3) data provisioning, and (4) knowledge base (component repository).

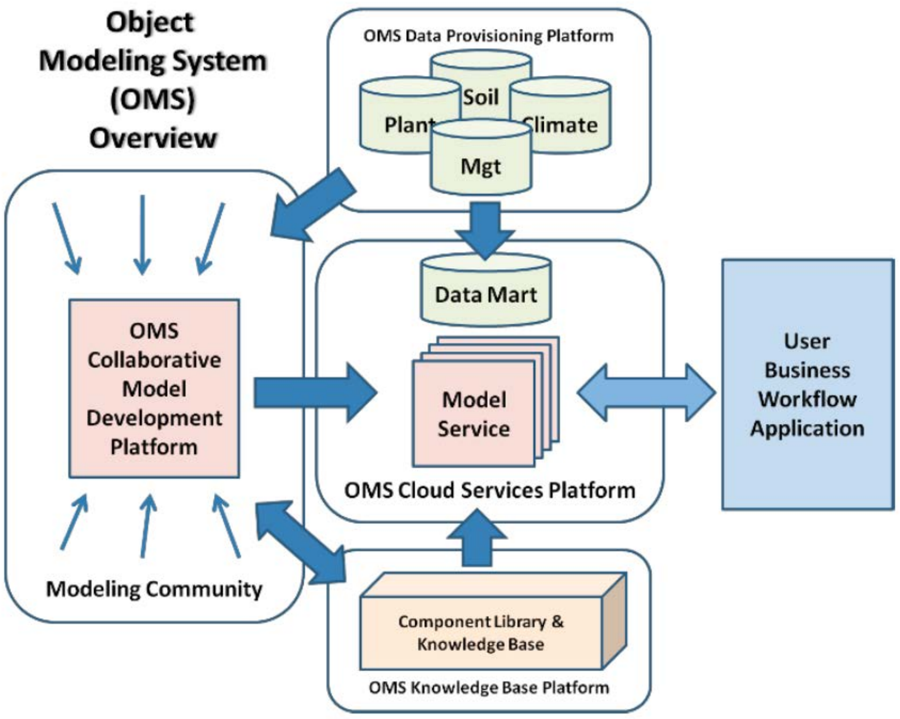

Object Modeling System (OMS) Overview
OMS Data Provisioning Platform
Soil
Plant
Climate
Mgt
Data Mart
OMS Collaborative Model Development Platform
Model Service
OMS Cloud Services Platform
User Business Workflow Application
Modeling Community
Component Library & Knowledge Base
OMS Knowledge Base Platform

Figure 7: Object Modeling System (OMS) Architecture Overview

The OMS framework supports the deployment of models as services in a cloud computing infrastructure. The platform also supports building of new models and decision support tools from reusable/standardized components from a library. One key advantage of such a modeling services

[3] https://alm.engr.colostate.edu/cb/wiki/16961

approach is that legacy Hazus models could be refactored, encapsulated, and leveraged as modeling services using OMS. Third party models could be used as well.

5.2 Discussion of Web and Open Source Alternatives

The authors of this document researched some 30 state-of-the-art software packages in eight different functional realms, namely:

Desktop GIS:	As thin clients and for Hazus users who need local, secure installations
Web GIS:	Map, feature, coverage and geoprocessing services, spatial data discovery
JavaScript libraries:	Web and mobile app development
Geospatial libraries:	Low-level, performance-optimized geometry, import/export and conversion
Spatially enable platforms:	Full-fledged not GIS-based solutions that can do all the GIS work on the side
Geospatial databases:	Special-purpose built databases for georeferenced data
OLAP databases:	Very fast, multi-dimensional databases for business analytics
NoSQL databases:	For unstructured and big data (high-resolution Open Hazus is big!)

The following items were investigated for each category. Additional details and descriptions can be found in Appendix D.

Desktop: PostGIS, SpatiaLite and Rasdaman.

For Web GIS: Four web mapping server platforms - MapServer, MapGuide, GeoMOOSE, and Mapbender, two true Web GIS - GeoServer and Carto, and one web geoprocessing platform - PyWPS).

JavaScript libraries: items specifically developed for geospatial applications (OpenLayers, Leaflet, turf), are widely used visualization packages with a strong subset of geospatial routines (D3), or are serving specialized geospatial audiences, such Cesium and kepler.gl.

Geospatial libraries can be divided into those that underlie most of the desktop and Web GIS described before (GeoTools, GDAL/OGR, GEOS), and those that have been developed for the storage and processing of big spatial data (GeoTrellis, GeoWave, and STAC).

The spatially enabled platforms are either well-developed (mature) software solutions for statistical and data mining (R) and for individual-based dynamic models (Repast), or very generic software development platforms that are so feature-rich that they include virtually everything an

Open Hazus developer could ask for (Eclipse and Jupyter) – with the added benefit that their interoperability has been field-tested.

In the open source world, it is taken as a given that PostGIS is the database management system of choice, although two others were evaluated for specialized use cases: SpatiaLite for field work and Rasdaman for storing massive multi-scale coverage (rasters and images). Based on the system requirements outlined, OpenHazus will have to store massive amounts of results data from a large number of scenario runs. These become intelligible only if stored in OLAP data stores such as Kylin, Pinot and OpenCube.

No-SQL databases are known in the GIS world mostly for streaming and unstructured social media data. In contrast to the domain of geospatial databases, no-SQL databases are too new to have easily identifiable winners for our software evaluation. With the exception of the popular CouchDB, most of them (OrientDB, redis, RavenDB and Cassandra) have at least one geospatial reference implementation. GeoWave is included here as well, revealing how difficult it is to drawn sharp boundaries between the categories.

Winners in each of the eight categories are described in the following sections. However, most of the software packages discussed here deserve a second look and the choices are not necessary mutually exclusive. An example is the absolute necessity to incorporate the GDAL/OGR library for import/export and conversion functions. But this is all that this library has to offer; it does not substitute for the computational geometry operations of GEOS or the parallel processing capabilities of GeoWave. Similarly, in the long run, chances are that Open Hazus will store hazard data in a PostGIS database, analysis results in an OLAP database and serve thin clients on mobile devices using SpatiaLite. Later sections will discuss the pros and cons of each recommendation and suggest priorities for different phases of the Open Hazus development.

5.3 Desktop GIS

From a system features perspective, QGIS, GRASS, and even gvSIG are very similarly ranked. All three are full-fledged GIS that fulfill a large set of the requirements. The reason that they did not receive perfect scores is that they are not (or in case of QGIS only to a very limited degree) available on mobile platforms, and do, as of now, not yet support native access to authoritative hazard input sources. In the end, QGIS emerged as the winner because GRASS lost in the software quality attributes category with its low score for the perceived ease of use (GRASS employs an arcane viewport-based user interface that is less than ideal even savvy GIS users, and most GRASS algorithms can be accessed through QGIS) and gvSIG scored just a little lower than QGIS on foreseeable longevity and the matching of existing Hazus features. The recommendation is to use QGIS as the reference desktop implementation of Open Hazus but several of the other desktop GIS packages might well be picked up by a future Open Hazus developer community.

5.4 Web GIS

This category of software packages contains three distinct groups of software solutions that cannot easily be compared to each other. MapServer, MapGuide, GeoMOOSE and Mapbinder are web mapping servers and will not be able to meet the analysis requirements of OpenHazus source code modules. Similarly, PyWPS is specialized software that Open Hazus developers might utilize for the sharing of geoprocessing libraries – PyWPS easily beat all other packages in the category of automation. In the end, however, the recommendation is a virtual tie between the very comprehensive solutions offered by GeoServer and Carto. Compared to GeoServer, Carto is a relatively new product; but it has garnered an impressive list of Fortune 500 and government clients. Carto is extremely user friendly, emphasizing visualization and business analytics over GeoServer's more traditional GIS perspective. GeoServer is the more conservative and expensive almost-all-in-one solution that would minimize development efforts especially in the startup phase of the Open Hazus project. The supporting company Boundless is committed to working with stable open source code bases and plays a similar role in the geospatial world as Red Hat does in the Linux world. GeoServer and Carto represent one end of the spectrum of Open Hazus approaches, with Jupyter and R representing the exact opposite – high development effort but maximum flexibility and "openness."

5.5 JavaScript Libraries

OpenLayers and Leaflet are the two libraries that every geospatial application developer would name as obvious choices for browser-based clients. Leaflet is the newer of the two and has a very slight edge in the system features group, while OpenLayers reverses its position in the software quality attributes category. The differences in either case are too small and dependent on weighting factors to overcome individual developers' preferences. It is worthwhile noting that in the systems features group, the relatively new and not as widely used turf library is beating its better-known competitors. The Open Hazus development team would be well advised to monitor this library to see how it fares in the long run in regard to foreseeable longevity. In the combined ranking, it is already beating OpenLayers.

5.6 Geospatial Libraries

As in the Web GIS category, the entrants in this group offer a diverse selection of technology types with typically highly specialized functionality. As elaborated previously, these libraries are not mutually exclusive. GDAL/OGR got a perfect score in its intended purpose and yet ranks only fourth for system features. It will nevertheless undoubtedly be part of any Open Hazus solution. The same is probably true for the GeoTools library, which every serious developer of GIS-able software should be familiar with. In the end, the clear winner in this category, however, is GeoWave. It may not be part of the first phase of Open Hazus but with its lead position in both geospatial libraries and no-SQL databases, it cannot escape the attention of anyone who endeavors to support the massive data requirements of OpenHazus.

5.7 Spatially Enabled Platforms

This may be the most challenging group of open source solutions for traditional Hazus users, especially if their expertise is rooted in the GIS realm. Of the four solutions researched here, the easiest to exclude is Repast. The user communities for Repast and Hazus overlap only minimally and the conceptual burden to perceive of Hazus through the eyes of individual-based modeling is probably too big to warrant further scrutiny in this direction. Eclipse and Jupyter represent the extreme opposite end of the development effort spectrum. Both offer an extremely well established and feature-rich environment and, of course, a huge developer community. In addition, the freedom to combine any and every conceivable software solution is a very promising prospect in the long run. Compared to an Open Hazus solution based on GeoServer or Carto, the first phase of

OpenHazus would most likely take as many years to develop as it would take months with the ready-made packages.

Less extreme, and with almost the same amount of flexibility, is an Open Hazus implementation based on R. Compared to the geospatial community, the developer base is huge and the range of solutions in regard to usability, visualization, and especially analytics is impressive. The development effort would be considerably less than it is for Eclipse and Jupyter, meaning that a prototype could be developed relatively easily. The main drawback is a not yet convincing support for massive data (although the STARS package shows an interesting pathway). From the perspective of software quality attributes, R is the clear winner in the category of spatially enabled platforms. It is Jupyter's perfect score on the automation side that beats it in the overall evaluation.

5.8 Geospatial Databases

PostGIS is the clear winner in this category and is highly recommended as the foundation for the first phase of OpenHazus. The other two software package scrutinized here are for specialized solutions. SpatiaLite is extremely easy and capable for small to medium-sized projects and hence ideal for field-based applications, for example in emergency response. Rasdaman is a data manager for huge multi-resolution raster data and as briefly mentioned in earlier sections, it could be a game changer if Hazus analyses are re-conceptualized in terms of high resolution rasters.

5.9 OLAP Databases

Online analytical processing (OLAP) and GIS have so far had very little overlap, and virtually none in the application of hazard management. OLAP databases have been developed in business analytics as a way to mine internal business data by treating it as n-dimensional data stores. Traditional inventory or hazard data does not fall into this category. But results databases do, and as one of the explicit goals of Open Hazus is to build a vast repository of scenario results, such a repository becomes only useful if it is organized as an OLAP data store. It enables the mitigation community to perform analyses that are based on latest developments in data science and while probably not part of the first phase of OpenHazus, will have to be kept in mind for future versions. Of the three open source OLAP databases investigated here, the result is mixed. On the systems features side, the European OpenCube project narrowly wins over Pinot. On the other hand, its functional advantage vanishes in light of a seeming lack of recent development of the software itself. There are a few sample implementations by regional governments and in the banking industry but OpenCube fails in the software quality attributes category. It would probably take a major grant-funded or similar research project to make OpenCube a feasible solution for OpenHazus. Pinot (developed by LinkedIn, now Microsoft) may be the safer bet for a results database. The authors of this document recommend reaching out to Microsoft to determine their level of interest in a joint development for this part of OpenHazus.

5.10 No-SQL Databases

Non-relational databases are garnering a lot of attention in the open source developer communities. They are popular both in the world of social media and, closer to the realm of geospatial applications, for data streams. One of the difficulties in the evaluation of non-relational databases is their novelty. Although most of the software packages researched here feature a geospatial reference implementation, each of these is a one-off case study – insufficient to check for the stability, not to mention maturity of the implementation. Among the true no-SQL databases, Cassandra wins on the system features side but ranks lowest on the software quality attributes side. In the end, none of these options are recommended over GeoWave, which is the winner in both categories and was already chosen in the group of geospatial libraries.

5.11 Recommended Software Architecture Conclusions

We propose a phased development of OpenHazus and anticipate the choice of different software packages for each of these phases based on a combination of the overall architecture and proposed Proofs of Concept (PoCs). The proofs of concept will be based on sample implementations of QGIS on the desktop and/or client side, Leaflet for a browser-based app, GDAL and GeoTools underlying any low-level code rewrites, and PostGIS as the default database.
The following phases are dependent on the business logic adopted by FEMA. An easy and early success solution could be achieved by adopting either Carto or GeoServer and handing over significant parts of the future development to either Carto (the company) or Boundless. The

opposite solution path would be a handful of small sample implementations in Jupyter or R and their release to the open source community to take these as examples for a family of code development efforts that are administered by something like an Open Hazus Foundation. The decision is less based on technical feasibility than on organizational flexibility and the allocation of resources.

6 Proofs of Concept – Completed and Proposed

Based on the above requirements and analysis, a number of Proof of Concept (PoCs) are recommended to support the future development of OpenHazus. Those underway include innovative and rapid ways to provide results and reporting, as well as rapid extraction of flood depths and support of an Open Source desktop utility for flood losses. Proposed PoC's include an assessment of the ability to incorporate nationwide building outlines and footprints to provide an inventory that meets the accuracy requirements of OpenHazus, as well as a PoC to determine if sufficient coverage of flood hazard data are available through online sources to support the requirement of OpenHazus to leverage authoritative datasets.

6.1 Open Source Utility for Hazus Data Export and Reporting

Hazus inputs and results are stored in backend SQL Server tables not intended for direct access by users. Exporting Hazus loss model results out of Hazus for further visualization or analysis using other technologies has historically been cumbersome. FEMA has distributed some Esri-based scripts that automate parts of the export process for a more streamlined user experience, but these scripts have comparatively slow runtimes, require advanced Esri licensing, and export results in traditional Esri-based geospatial formats. SQL Server databases can be accessed and manipulated through a wide range of Open Source analytical libraries and the vast majority of Hazus model results are effectively managed and visualized through non-spatial processes. GIS scripting in general and Esri tools in particular can therefore be bypassed in the development of a simple utility that extracts and summarizes Hazus model results in universally recognized text and image file formats.

A prototype utility for Hazus data extraction and summarizing has been developed for high-level earthquake model results. The utility employs Open Source Python libraries to extract several earthquake damage data points from Hazus SQL Server tables, summarize those data points across a variety of categories, and join them to a shapefile of study region tracts converted from SQL Server Spatial. The utility produces four CSV files and one shapefile summarizing earthquake loss results and publishes those results to a public FTP site in an average of less than 10 seconds.

This prototype is being expanded to include flood, hurricane, and tsunami loss results and to provide basic data visualizations (spatial and graphical) as standard image outputs.

If this utility can be effectively implemented as a user-friendly web service, it will serve as a proof of concept that provides interactive data visualization and results reporting in a web-based, Open Source version of Hazus.

6.2 Nationwide Essential Facility Inventory Updates using HIFLD Open

The Homeland Infrastructure Foundation-Level Data program provides regularly updated, centrally managed nationwide inventory datasets similar to those used by Hazus. Hazus inventory datasets, however, are costly and inefficient for the Hazus Program to maintain and are therefore significantly outdated. Dynamic integration of Hazus with HIFLD layers would eliminate the duplicative burden to continuously update nationwide facility databases. Scripts and supporting data are being developed in order to assign attributes required by Hazus loss models to HIFLD layers so that legacy Hazus facility data can be updated and OpenHazus facility data can be automatically and dynamically updated going forward. A working partnership between HIFLD and the Hazus Team has been established so that each program can make the changes required to implement this dynamic data link.

6.3 OpenHazus Site-Specific Flood Analysis

The accuracy and credibility of a flood loss estimation study is substantially enhanced when the specific location of a structure is known in relation to the flood hazard. Unlike other hazards, such as wind or earthquake, the severity of the flood hazard (usually water depth) may change dramatically over short distances. Floods may result in deep water within residential streets but no or limited flooding may occur at structures. Assumptions made when distributing aggregated building stock or area weighting flood depths across a Census Block introduce significant sources of error or uncertainty. An objective of OpenHazus is to provide an Open Source, enhanced, nationwide site-specific flood loss analysis capability. This requires significant optimization of the existing legacy Hazus UDF analysis in order to extract depth values and process large numbers of

facilities quickly, as well as the dynamic application of vulnerability attributes and Depth Damage Functions (DDFs) to user-supplied or nationwide building footprint-based inventory. While most users will be able to leverage a new web-based OpenHazus capability for flood loss estimation, users that are unable to share Personally Identifiable Information (PII) or sensitive datasets online would require a lightweight Open Source desktop utility to perform flood loss analysis locally.
An example of using the Hazus-based functions within an open source context is given by Gutenson and others (2017), whose Flood Damage Wizard built on Hazus DDFs using R (statistical computing language) package constructed by Gopi Goteti (Damage functions from FEMA's Hazus software for use in modeling financial losses from natural disasters, https://cran.r-project.org/package=hazus). This Open Source publication of the Hazus flood damage functions include not only the building and contents DDFs but the damage parameters for other infrastructure as well as crop losses. In addition, this package includes an Open Source visualization utility (ggplot2) for graphic display of selected damage functions.

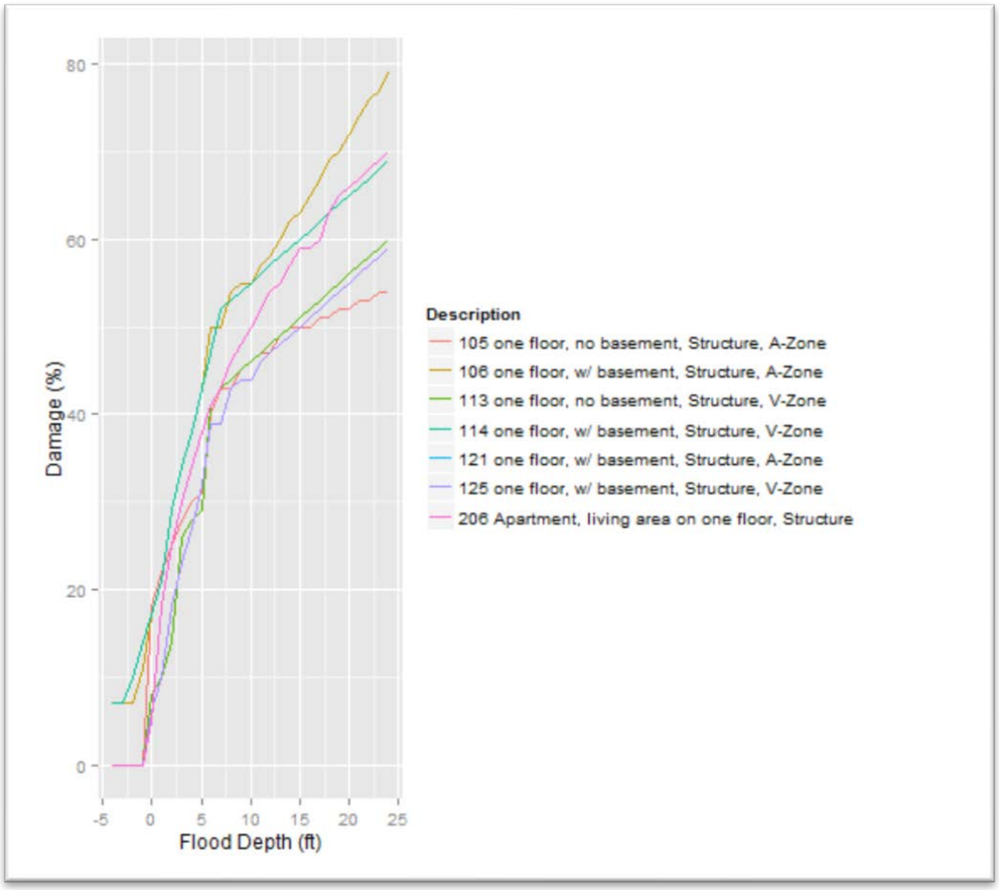


Figure 8: Sample open source publication of Hazus DDFs

A Proof of Concept (PoC) was developed to demonstrate the feasibility of the following flood loss estimation capabilities:

1. Leverage free and open source tools
2. Optimize processing time with the goal of extracting flood depths at structures and estimating losses for ~10K records per second
3. Provide both a light weight desktop utility and web-based tool

The OpenHazus Site-Specific Flood Analysis POC has evolved through four versions to arrive at its current state. These incremental versions are outlined and described below.

Table 5: Proposed Incremental File Formats (Flood)

Version	Software	Runtime	Desktop Utility Size*	Input Format	Output Format

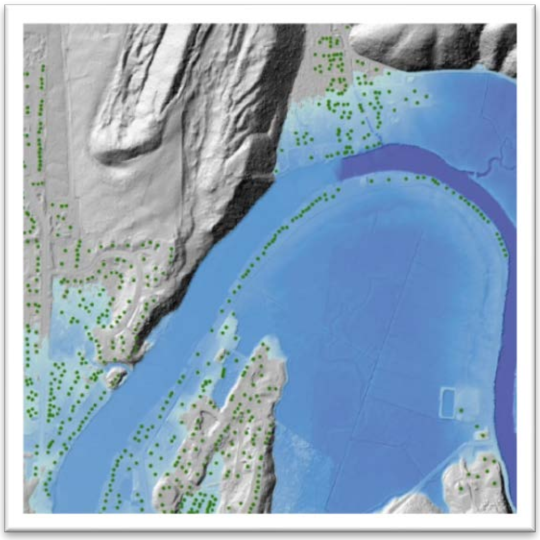
Hazus 4.2[1]	Hazus and ArcGIS with Spatial Analyst	UDF in 44 min 20 sec	12.4GB Hazus; 2.28GB ArcGIS with Spatial Analyst	.mdb or .xls (using CDMS)	SQL Server tables
Hazus 4.2[1]	Hazus and ArcGIS with Spatial Analyst	UDF AAL in 6 sec	12.4GB Hazus; 2.28GB ArcGIS with Spatial Analyst	SQL Server tables	SQL Server tables
OpenHazus0[2]	ArcGIS (DOGAMI)	84 sec	2.28GB	.gdb, .mdb, .shp	.gdb
OpenHazus1[2]	QGIS and .shp	300 sec	1.95GB	.mdb, .shp	.shp
OpenHazus2[2]	QGIS and .cavy	40 sec	1.95GB	.mdb, .shp	.shp, .cash
OpenHazus3[2]	GDAL and .shp	13 sec	56.5MB GDAL plugins; 206MB Python libraries	.shp	.csv
OpenHazus4[2]	GDAL and .csv	4 sec	56.5MB GDAL plugins; 206MB Python libraries	.csv	.csv
OpenHazus5[2]	Hazus Web	~1 min up and down	None; 6.6MB .csv upload and 11.3MB .csv results	.csv	.csv

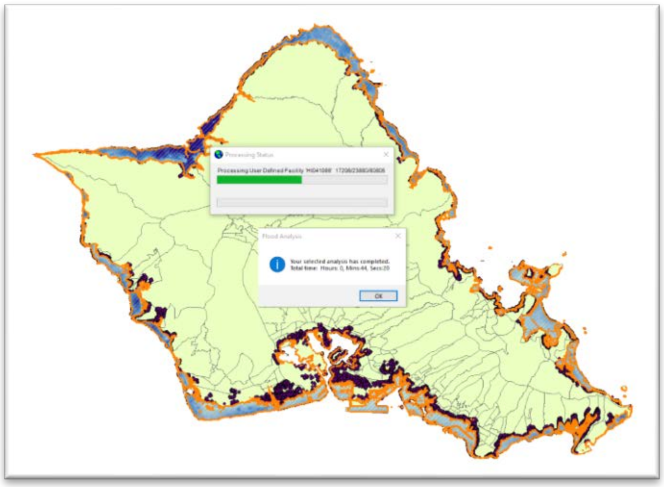
Note: Oahu site-specific building inventory (27K bldgs) was 35MB shapefile or 6.6MB csv. Oahu input hazard was 232MB GeoTIFF depth grid representing potential inundation from Great Aleutian Tsunami.
1- Accounting for RDBMS overhead capabilities such as Backup/Recovery, Relational Algebra, Spatial Algebra and all the remaining ones listed in 0

2- Flat files, such as .csv don't have the overhead of RDBMS, and also have none of those capabilities.

6.3.1 Current Hazus (Version 4.2) Capabilities

The current Hazus flood loss model supports UDF capabilities that have been enhanced over time with processing optimization, the ability to assign custom DDFs, streamlined reporting and the ability to estimate Average Annualized Losses (AAL) when 5 return period depth grids are available. Preparing UDF data for Hazus has been supported with the Comprehensive Data Management System (CDMS), which helps users develop default building areas, building and content valuations, foundation types and finished floor elevations for their user-supplied datasets. CDMS will prepare UDF data from either Personal Geodatabases (.mdb) or Excel spreadsheets (.xls) and load the results into a user's state dataset or study region. If users do not assign DDFs from the Hazus library, defaults based on occupancy type, number of stories and foundation type, as well as hazard type (riverine, coastal A or coastal V) will be used during analysis. The current Hazus UDF module performs a point by point extraction of flood depth at user-supplied locations and analyzes losses for buildings, contents and inventory in a linear non-optimized trend, using a record set, with the worst-case scenario time of O(n). An optimized Hazus AAL UDF module, utilizing sets, performs in O(log(n)) has been developed for the computation of annualized losses after the full suite of return period UDF analyses are performed. This distinction while insignificant for n in (100s, 1000s) becomes a factor when n > 10000 and is largely why the current Hazus UDF process is not yet optimized and comparably slower than the new utility approaches. In addition, income losses, debris estimates, and functionality losses are not provided in the current Hazus desktop UDF capability.




Processing Status
Processing User Defined Facility 'HI041088' 17206/23880/60806
Flood Analysis
OK

Your selected analysis has completed.
Total time: Hours: 0, Mins:44, Secs:20

Figure 9: A typical Hazus UDF flood analysis includes depth grids and a point-based structure inventory. An Oregon lidar-based study region is shown of the left and the island of Oahu on the right with Hazus 4.2.1 runtime.

6.3.2 ArcGIS DOGAMI Tool (POC Version 0)

The Oregon Department of Geology and Mineral Industries (DOGAMI) serves as FEMA's Cooperating Technical Partner (CTP) in implementation of the Risk MAP program across the State. In order to optimize UDF flood analysis times and incorporate additional capabilities, DOGAMI exported Hazus flood damage functions and developed an ArcGIS Python tool that streamlines analysis steps and provides additional site-specific results such as debris and days to restoration. The DOGAMI tool takes tabular input data, including .mdb or shapefiles. The tool looks for specifically named attributes/columns required for analysis including user provided building, content and inventory damage function IDs matching those in the Hazus DDF libraries. Both the field headings and data types need to be provided in a required format. Coordinate attributes are used to extract flood depth at UDF locations by referencing a supplied flood hazard raster file. The tool also references supplied csv files containing the various damage functions extracted from Hazus 4.0 SQL Server tables. The end result is another inventory table with additional results columns representing building, content, and inventory percent damage and economic losses, as well as debris estimates and loss of use in days. In the DOGAMI tool, only the extraction of flood depth at structure requires the use of GIS.

The DOGAMI tool provides .csv versions of Hazus loss model tables exported from SQL Server. These include lookup tables that provide building, content and inventory losses for all three hazard environments (riverine, coastal A and V), economic income loss table parameters as well as finish, structural and foundation debris parameters:

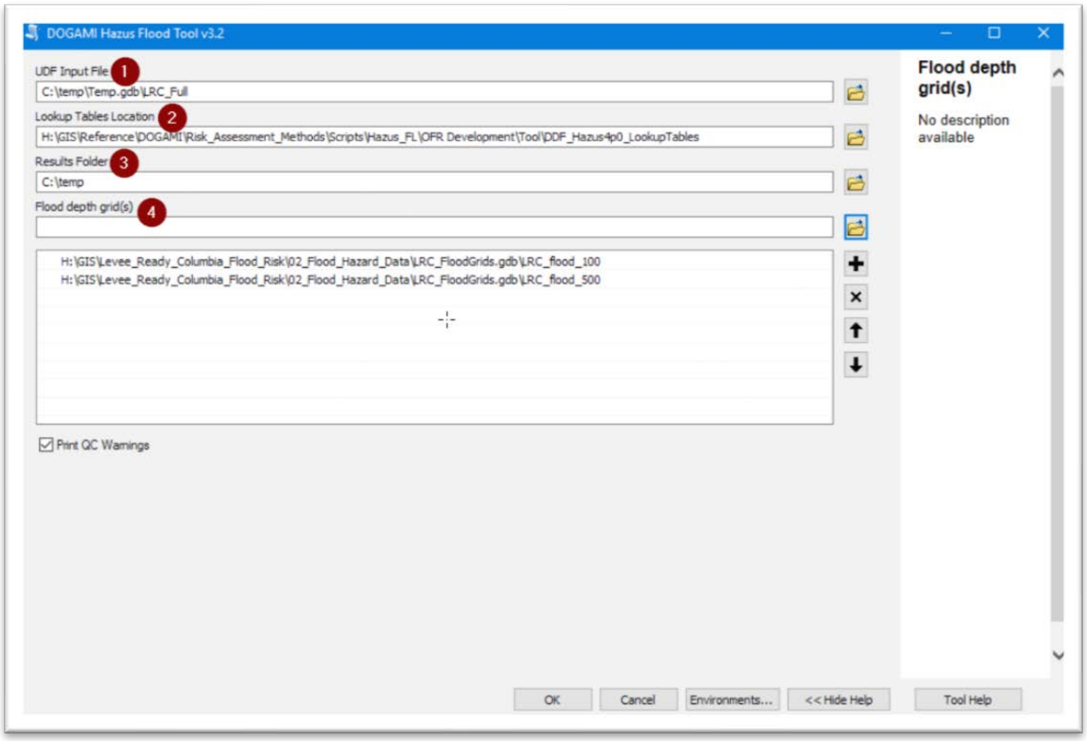
DDF_Hazus4p0_LookupTables *folder, containing the following CSV (comma-separated values) files:*

```
        Building_DDF_CoastalA_LUT_Hazus4.0.csv
        Building_DDF_CoastalV_LUT_Hazus4.0.csv
        Building_DDF_Riverine_LUT_Hazus4p0.csv
        Content_DDF_CoastalA_LUT_Hazus4p0.csv
        Content_DDF_CoastalV_LUT_Hazus4p0.csv
        Content_DDF_Riverine_LUT_Hazus4p0.csv
        flBldgContDmgFn.csv
        flBldgEconParamOwnerOccupied.csv
        flBldgEconParamRecaptureFactors.csv
        flBldgEconParamRental.csv
        flBldgEconParamSalesAndInv.csv
        flBldgEconParamWageCapitalIncome.csv
        flBldgInvDmgFn.csv
        flBldgStructDmgFn.csv
        flDebris_LUT.csv
        flRsFnGBS_LUT.csv
        Inventory_DDF_LUT_Hazus4p0.csv
```

The development of the DOGAMI tool is well documented in Open File Report O-18-04: https://www.oregongeology.org/pubs/ofr/p-O-18-04.htm. In addition, the development included extensive validation against the Hazus flood model itself. While the tool delivered the same DDF libraries available in the Hazus flood model, additional user-developed DDFs can be added as additional rows in the tool's input .csv files. The DOGAMI tool process about 10,000 records per minute, while the current Hazus UDF flood analysis process a similar number of records in ~15 minutes.

The DOGAMI flood tool requires 4 basic steps:

1. Define location of input file either file or personal geodatabase, or shape file
2. Define location of damage parameter tables (.csv)
3. Define folder location for results and a new file geodatabase will be created
4. Define location of depth grid(s) supporting various formats and any projection


DOGAMI Hazus Flood Tool v3.2

UDF Input File 1
C:\temp\Temp.gdb\LRC_Full

Lookup Tables Location 2
H:\GIS\Reference\DOGAMI\Risk_Assessment_Methods\Scripts\Hazus_FL\OFR Development\Tool\DDF_Hazus4p0_LookupTables

Results Folder 3
C:\temp

Flood depth grid(s) 4

H:\GIS\Levee_Ready_Columbia_Flood_Risk\02_Flood_Hazard_Data\LRC_FloodGrids.gdb\LRC_flood_100
H:\GIS\Levee_Ready_Columbia_Flood_Risk\02_Flood_Hazard_Data\LRC_FloodGrids.gdb\LRC_flood_500

☑ Print QC Warnings

Flood depth grid(s)

No description available

Figure 10: DOGAMI Input Sample

6.3.3 QGIS and .shp (POC Version 1.0)

Quantum GIS or QGIS is perhaps the most widely used free and Open Source GIS system. This first alternative version used QGIS and the Python package side of QGIS, called PyQGIS. PyQGIS was used to process the input files and provide results identical to those provided from the UDF flood analysis methods in ArcGIS. This version was indented to be equivalent to the original DOGAMI script. This program ran slower than the original due to inherited differences between ArcPy and PyQGIS. There are aspects of the PyQGIS package that appear underdeveloped compared to ArcPy, in particular its much smaller active user community available for debugging. Another limitation is the size of the overall QGIS package at 1.95GB. While smaller than Hazus, this version is almost as large as the original ArcGIS package leveraged in the DOGAMI script.

6.3.4 QGIS and .csv (POC Version 2.0)

This second alternative version is still based in QGIS and utilized QGIS tools to convert input tables into .csv format. However, it then uses a more efficient built-in Python tool for processing the .csv input and output tables. This processing tool results in an order of magnitude decrease in overall processing time compared with the original QGIS version. However, the approach requires input shapefiles to be converted to .csv before processing, and the conversion of outputs back to shapefile formats using PyQGIS would increase processing time. This version is far more efficient (x10) in terms of processing time compared to v1.0; however, it still requires the large QGIS package download.

6.3.5 GDAL and .shp (POC Version 3.0)

A third alternative version diverged more from the original, including a .shp input and creating .csv outputs. This more simplified program allowed for the use of Python's faster .csv processing tools. However, the significant benefit was in the use of GDAL for raster processing. GDAL is a powerful and mature library for reading, writing and manipulating raster datasets, written in C++ with bindings to other languages. This version had dramatically lower overhead than previous versions (the environment for running this version is much smaller than previous versions since QGIS was not needed) and the least reliance on external packages, resulting in a 256MB total package size down from almost 2GB. This version also achieved much faster processing times – about 13 seconds for ~27K records, with the majority of processing time spent on converting the .shp to .csv.

6.3.6 GDAL and .csv (POC Version 4.0)

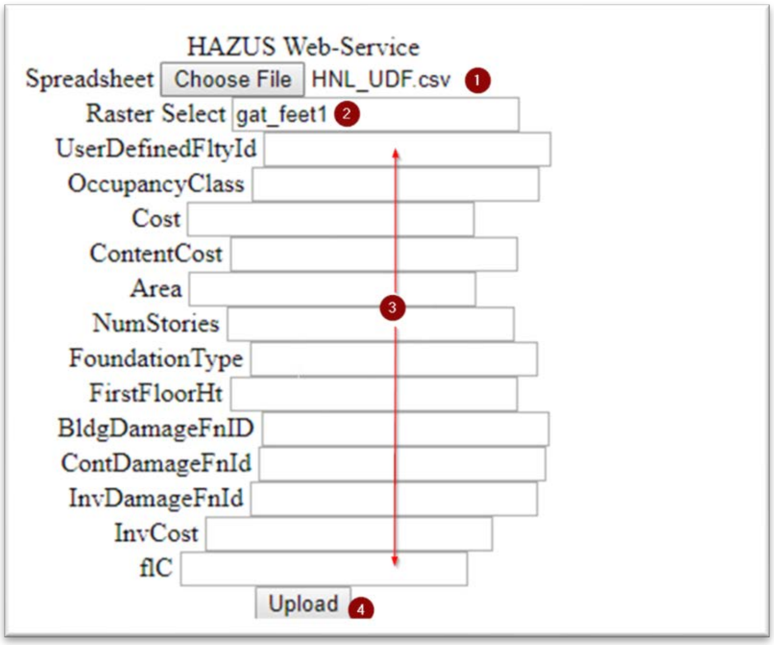
This version is identical to v3.0 above but allows the use of .csv input, thus not requiring any conversion of .shp to .csv. This version completes analysis and provides results in less than 4 seconds for ~27K records. This version very nearly achieves the POC objective of 10K records per second. This version is being enhanced for use as a local user-friendly version of the tool with a graphical user interface with customization similar to the web-version (v5.0) that supports attribute mapping and raster selection based on a library of available depth grids. This fast and lightweight utility will help meet the loss estimation requirements of OpenHazus users that will be unable to upload PII or secure data to a future OpenHazus web platform.

6.3.7 Hazus Web POC (POC Version 5.0)

A very simple browser interface was developed to prove web capability and test upload of inventory, download of results and runtimes based on a version deployed on a test web server running PoC Version 4.0. This web version is made using a Python micro-framework for web tools called Flask. When users run the tool from the web page, the hosting server computer performs computational work by running the associated Python script on the server. The computation time is based on available server power, but total processing time will include time for user upload of input data and user download of results. Use of the smaller .csv formatted files will improve upload and download times as well as benefit processing time. Based on the ~27K record Oahu dataset, total upload and download times were ~1 minute. As with other PoC versions, the .csv results file contained the original inventory attributes plus building, content and inventory damage percentage and economic losses, as well as estimates of debris tonnage by type (finish, structure and foundation), and days to restoration.

There are 4 steps required for the online version:

1. Browse to input file
2. Select depth grid
3. Use default fieldnames or enter fieldnames if different
4. Select upload



Spreadsheet	HNL_UDF.csv
Raster Select	gat_feet1
UserDefinedFltyId	
OccupancyClass	
Cost	
ContentCost	
Area	
NumStories	
FoundationType	
FirstFloorHt	
BldgDamageFnID	
ContDamageFnId	
InvDamageFnId	
InvCost	
flC	

Figure 11: Sample Hazus Web Graphic

6.4 OpenHazus Assignment of Default Depth Damage Functions (DDFs)

To provide a rapid site-specific flood loss estimation capability within the OpenHazus environment, techniques to select and assign appropriate DDFs need to be developed. User-provided DDFs offer the opportunity for additional customization and detail, such as the use of long-duration DDFs in areas behind levees. However, a significant number of users will not be familiar with the process of assigning DDFs. This proposed proof of concept will develop sophisticated methods for assigning DDFs to user provided, as well as national building datasets. When structure, content or inventory DDFs are not provided by the user, OpenHazus will assign best fit defaults based on user input for general building characteristics and proximity to riverine, coastal A and coastal V hazard areas:

Occupancy Type: based on Hazus 33 model building occupancy type categories

Number of Stories: used to classify buildings into low-, mid- and high-rise types (*note that Hazus also provides damage functions for split-level single family residential, for these 99 is used* *to delineate these types*). We also need to handle cases where the use may not know or enter 0 num stories that should default to low rise.

Basement: a flag indicates if the structure has a basement (a 4 is used to indicate a Hazus basement foundation type)

The combination of these three attributes is used to assign 1 of 196 specific occupancy type IDs (SOoccupID). OpenHazus (as well as current Hazus) then assigns default DDFs for structure, contents and inventory based on these 196 specific occupancy type categories and the type of hazard, including riverine, coastal A or coastal V zones:

Structural Damage DDFs: Default structural DDFs are contained in *flBldgStructDmgFinal* and assigned based on the SOoccupID and hazard type (*HazardR*, *HazardCA*, *HazardCV*) to determine the Building Damage DDF (*BldgDmgFnId*).

Content Damage DDFs: Default structural DDFs are contained in *flBldgContDmgFinal* and assigned based on the SOoccupID and hazard type (*HazardR*, *HazardCA*, *HazardCV*) to determine the Building Damage DDF (Cont*DmgFnId*).

Inventory Damage DDFs: Default structural DDFs are contained in *flBldgInvDmgFinal* and assigned based on the SOoccupID and hazard type (*HazardR*, *HazardCA*, *HazardCV*) to determine the Building Damage DDF (Inv*DmgFnId*). (*Note that only a few commercial and industrial occupancy types are susceptibility to inventory losses.*)

6.5 Site-Specific Data for the Nation PoC

This proposed PoC supports the OpenHazus goal of providing a new Level 1 analysis capability using a site-specific national inventory, such as the Microsoft Building Footprints (https://blogs.bing.com/maps/2018-06/microsoft-releases-125-million-building-footprints-in-the-us-as-open-data). The ability to accurately intersect the hazard information at the location of each inventory item will enhance OpenHazus loss estimation quality over existing methods of using area-weighted aggregated block data or census tract centroids. The additional challenge is to ensure reasonable estimates for Hazus building required attributes can be provided. This project will evaluate and prototype the use of Open Location Codes (OLC's) such as Google's Plus Codes that provide a unique location specific ID anywhere on earth.

A design for methods to assign Hazus required attributes to site-specific data, including occupancy and building types, design levels, building area and valuations, as well as foundation and finished flood elevations should be developed and evaluated.

Review of quality regarding other available datasets for General Building Stock (GBS) solutions should be completed, including:

•Hazus legacy national baseline data

•National Structure Inventory (NSI) 2.0

•Oak Ridge National Labs (ORNL) building outlines

•Lidar-based building footprints.

6.6 Core of Hydraulics Replacement Solutions

The FEMA Hazus program developed and released the Flood Model in 2003. At the time, the flood hazard data required to support loss modeling, specifically flood depth grids were not widely available. FEMA's Map Modernization (2003-2008) and Risk MAP programs had not yet begun to modernize and transform the nation's flood data. As a result, an innovative automated Core of Hydraulics (COH) coding effort was included with the Hazus flood model. This automated Hydrology and Hydrologic (H&H) capability was at the time state-of-the-art and to this day is still the only freely available automated H&H capability available to the public. However, COH now requires significant methodology updates and is limited in capability when compared to the proposed OpenHazus solution of integrating high quality user-supplied flood data depth grids. COH is also computationally intensive and as such would limit proposed OpenHazus solutions.

While there is considerably more high-quality flood hazard data available for the nation than in 2003, unfortunately there are still gaps in coverage across the nation and no uniform national flood hazard depth grids freely available for OpenHazus users. The COH replacement solution looks at a broad range of potential solutions and has determined several concurrent approaches are required to provide adequate flood hazard data coverage for OpenHazus. The potential solutions should be assessed based on the following criteria for OpenHazus:

1. ***Accessibility***: Includes ease of discovery from existing online resources. Offline or secure (non-public) resources will receive low grades.
2. ***Quality***: Based on resolution and FEMA regulatory flood mapping quality criteria. Lidar-based depth grids will receive higher scores based on agreement with high resolution terrain data.
3. ***Coverage***: Includes geographic coverage with higher weights to coverage of high risk areas.
4. ***Nationally Applicable***: Based on the ability of the solution to provide a standardized nationwide depth grid solution for all OpenHazus users.

6.7 Online Discovery and Access of FEMA Mapping and Information Platform (MIP) and Map Service Center (MSC) Data

This approach would consist of the ability to discover and integrate depth grid products published in FEMA's Mapping and Information Platform (MIP) https://hazards.fema.gov, as well as the Map Service Center (MSC) https://msc.fema.gov. The MSC typically represents the final current effective products, while the MIP includes studies that are in progress and requires login credentials. With the MIP re-architecture in 2017, the data is the same in both places and the MSC will be updated to display the data directly from the MIP rather than creating copies. The available data include depth grid products (commonly GeoTIFF) that were developed from hydraulic models associated with current effective products, as well as preliminary data and non-regulatory products. The Data Capture Technical Reference that applies to both the MSC and MIP indicate that GIS data should be submitted in 3 zip files, each with these words in their file names "shapefiles", "geodatabase", and "geotiffs". While it will be difficult to identify if a geodatabase contains a raster depth grid, searching based on geotiff should enable discovery of the zipped depth grid files. The Flood Risk Products (FRP) database published to the MSC has included a requirement for multi return period depth grids when a new flood hazard analysis is completed for riverine since the new standards were updated in February 2018. The depth grids are commonly GeoTIFF format. Plans to create a national FRP layer and enhancement of the architecture to establish a national grid layer or layers has been proposed, however, the work was put on hold since no FRP data that was submitted to date meets the new (Feb 2018) quality criteria. While neither platform currently supports geographic queries based on a user's Hazus Study Region, an intermediary product such as the

NFIP Community Information System https://data.femadata.com/FIMA/FIMA_Community_Boundary_Data/Community_Layer/ may be used to relate study region and search/discover depth grids available for the included communities. Both the MIP and MSC represent significant and growing archives of high-quality flood mapping products, the extent of depth grid coverage, especially multi-return period depth grid products are limited. Assessing the geographic coverage of available grids is complicated since the extent of the grid can be much less than the community. However, the coverage in the nation's highest risk areas, including coastal regions, is more complete.

Strengths:

- Leverages significant investment in high quality data
- Leverages existing Risk MAP CDS systems for data access and retrieval

Weakness:

- Incomplete geographic coverage
- Incomplete coverage of multi-return period products

7 Conclusion and Next Steps

The earlier sections of this paper demonstrated the need for Hazus to explore new technology options. OpenHazus must be less dependent on third party and/or outdated products, more open to customizations from users, and better able to adapt to new advances in data development, data management, hazard science, and results reporting. These changes are necessary if Hazus hopes to maintain its position as a globally recognized standard for estimating losses due to natural disasters, including the credibility of its results. Hazus overall must be more open to its users, thus the name OpenHazus.

Based on the ongoing challenges of Hazus in its current form, and feedback from users through a variety of channels, five categories of requirements were determined: 1) Independence from desktop technology; 2) Independence from Esri ArcGIS products, with minimized dependence on GIS overall; 3) Modularization of existing Hazus capabilities; 4) Data sharing opportunities for baseline FEMA datasets, third party authoritative data, and custom user data; and 5) Increased automation for common, baseline, or bundled analyses. These requirements categories yielded over 100 sub-requirements, which the authors evaluated against over 30 different open source and web-based technology options. The recommended solutions for each category of technology are listed in Section 5, along with proposed overall system architecture for a web-based OpenHazus. It is recommended that the open source technologies form the basis of a container environment within a web-based MicroServices architecture.

The following next steps are proposed:

- With FEMA, conduct a sensitivity analysis of the weighted criteria used in the AoA tables. The technologies recommended in these tables are highly sensitive to even slight adjustments in the weighting criteria, and if any priorities are adjusted, could results in different outcomes from the analysis tables. It is crucial to determine if the weighting criteria need any adjustments to better align with FEMA and/or OpenHazus needs, thus impacting the recommended technologies for OpenHazus modules and functionality.
- Complete the proposed PoCs from Section 6. Some work has already been completed by third party agencies and Hazus partners and upcoming PoC results will help to guide future design sessions and phased rollouts of OpenHazus.

- Begin a phased approach to design, integrating work already completed with the most recent PoCs. Inventory updates will be a simple and straightforward starting point, as indicated by the PoC discussion. Initial changes can be achieved both with the HIFLD Open datasets, as well as options for migration away from aggregated census data to site-specific as a new standard. Similarly, PoCs are already underway to determine alternatives for flood hazard analysis, thus removing the current Hazus spatial analysis process requirements for determining an approximate depth grid.
- Development of a solution to provide access to flood hazard data (depth grids) suitable for loss estimation is a critical next step for OpenHazus. A myriad of existing flood hazard data sets exists from Federal, State and local resources. A phased approach to develop a library of flood hazard data should be implemented, including developing partnerships and leveraging existing APIs.
- Begin deployment and build out of the OpenHazus loss modelling results library identified as the solution to provide Hazus results and capabilities to a broad range of Hazus users. By placing these comprehensive results within a readily accessible interactive viewing environment, many user needs will be met while the modular development of OpenHazus is completed.

Appendix A: Definitions, Acronyms and Abbreviations

AAL	Average annualized loss
AAR	After action report
AEBM	Advanced Engineering Building Model
BFE	Basic Flood Elevation
BIT	Building Import Model
ADCIRC	ADvanced CIRculation model for oceanic, coastal and estuarine waters, developed at UNC
CAS	Chemical Abstracts Service registry number
CAT	Crisis Action Team
CDMS	Comprehensive Data Management System
CDS	Customer and Data Services, section of FIMA's Risk Management Division
CERT	Community Emergency Response Team
CEMP	Comprehensive Emergency Management Plan
COG	Continuity of Government
COOP	Continuity of Operations Plan
CISM	Critical Incident Stress Management
CIKR	Critical Infrastructure and Key Resources
CNMS	Coordinated Needs Management Strategy
DEM	Digital Elevation Model
DHS	(United States) Department of Homeland Security
EF	Essential Facilities
FEMA	Federal Emergency Management Agency, a unit of DHS
FGDC	Federal Geographic Data Committee
FIMA	Federal Insurance and Mitigation Administration, a unit of FEMA
FIRM	Flood Insurance Rate Map
FIS	Flood Insurance Study
FIT	Flood Information Tool
GBS	General Building Stock
GBT	General Building Type

GIS	Geographic Information System
H*WIND	Hurricane Surface Wind Database
H&H	Hydrologic and Hydraulic (modeling studies)
HEC	Hydrologic Engineering Center (of the U.S. Army Corps of Engineers)
HEC-RAS	HEC's river Analysis System that models the hydraulics of water flow through natural streams
HFT	Hazard Factor Tables
HIFLD	Homeland Infrastructure Foundation-Level Data
HPLF	High Potential Loss Facilities
HVA	Hazard Vulnerability Assessment
HSEEP	Homeland Security Exercise and Evaluation Program
INCAST	Inventory Collection and Survey Tool
LFD	Letter of Final Determination
LOMR	Letter of Map Revision
MAP	Mapping, Assessment and Planning
MIP	Mapping Information Platform (part of CDS Risk MAP)
MMI	Modified Mercalli Intensity
MSC	Map Service Center; one of five units in CDS (see above)
NAVD	North American Vertical Datum
NHC	National Hurricane Center
NED	National Elevation Dataset
NEHRP	National Earthquake Hazards Reduction Program
NFIP	National Flood Insurance Program
NHD	National Hydrography Dataset (1:24,000)
NHRAP	National Hazard Risk Assessment Program
NTHMP	National Tsunami Hazard Mitigation Program
NWIRP	National Wind Hazards Reduction Program
NWIRP	National Windstorm Impact Reduction Program
NWM	National Water Model
OGC	Open Geospatial Consortium
ORNL	Oak Ridge National Laboratory

P4	Risk MAP Project Planning and Purchasing Portal
PCII	Protected Critical Infrastructure Information
PESH	Potential Earth Science Hazards
PGA	Peak Ground Acceleration
PGD	Permanent Ground Deformation
PGV	Peak Ground Velocity
PII	Personal Identifiable Information
PODs	Points of Distribution
Risk MAP	Risk Mapping, Assessment and Planning, a unit of FIMA
SFHA	Special Flood Hazard Area
SLOSH	Sea, Lake and Overland Surges from Hurricanes model developed by NOAA
SSI	Sensitive Security information
SWAN	Simulating WAves Nearshore wave model (developed at TU Delft, NL)
TIGER	Topologically Integrated Geographic Encoding and Referencing system
UDF	User-Defined Facility
UI/UX	User Interface / User Experience
USGS	United States Geological Survey
WKT	Well-known Text, an ISO and OGC markup language standard for vector geometries
WSEL	Water Surface Elevation

Appendix B: User Profiles & Use Cases

1. Local Certified Floodplain Manager

Robert works for the City of Davenport, IA as a Certified Floodplain Manager. His primary job functions include monitoring new development or ongoing changes to ensure they comply with floodplain regulations, monitoring flood control infrastructure, and identifying new opportunities for flood mitigation. He also works closely with Rock Island and Moline, IL, directly across the Mississippi from Davenport. Using Legacy Hazus, Robert is able to update the existing Hazus inventory data for Davenport with his own local data on demographics and structures from the city's offices so that anytime he wants to analyze a possible flood scenario, he can create a new study region with an updated data set.

For generating his flood hazard, Robert typically asks friends in the Iowa Flood Center to generate a raster GRID file for him. He has received training in generating a depth grid using Hazus, but finds the process can take days on his desktop machine, and the resulting grid lacks in accuracy. He has also experimented some with the USACE HEC-RAS tool, but the state center is able to provide a much more accurate and hi-res depth grid using their LiDar DEM data and Robert can import this directly into Hazus. Using his own inventory data and the depth grids from the state office, Robert is able to assess most situations as needed for his job functions; although he worries that real-time analysis for a major flooding event using Hazus is risky. Fully processing the data from the first step of creating a study region, to the last step of generating loss reports, can still take hours even with a pre-generated depth grid. The depth grid files themselves are also difficult to transfer to his desktop machine, usually with a folder size of 20+ GB containing a number of files that comprise the raster in spatial format. Robert also must trust that the desktop installations of Hazus will not crash or throw an error during this process, and that the incoming depth grids from the state office are valid.

In future versions of Hazus, Robert is hoping to be able to run analyses in a web environment in real time, and not worry about taking time to set up study regions and depth grids in his desktop environment. Robert would also like to reduce his dependence on the flood office, and have more options for obtaining a depth grid from alternate sources or generating one himself according to his preferred parameters. Similarly, if Robert finds he needs to work closely with partners across the river in Illinois, sharing depth grids and results in the web environment would be much less cumbersome.

Pain points:

- Data sharing: user inventory, hazard input and analysis results
- Maintenance and upkeep of default inventory
- Desktop environment: installation, performance, ease of use
- Processing time for data input and analysis results

2. State Mitigation Planner

Jane is a mitigation planner for the State of Texas. In addition to assessing local mitigation plans and prioritizing proposed mitigation projects, she is also responsible for the scientific hazard assessment portion required in all state-level mitigation plans. These plans must be updated and re-submitted to FEMA every five years in order to maintain an active status and allow the state to receive certain types of disaster funding. The next iteration of the mitigation plan hazard assessment is due soon, and Jane is concerned about the hurricane portion. The last version of the assessment was completed more than five years ago by a private contractor who no longer works for the state. Jane has only the results of a Hazus hurricane study that was included in the last mitigation plan. The contractor did not leave any documentation on their methodology, input files, or Hazus study regions. Since the mitigation plan was accepted five years ago by FEMA, Jane would like to reproduce the approved methodology as much as possible.

Jane reasons from clues in the report and Hazus hurricane manuals that the contractor used a historic storm combined with the current building inventory provided in Hazus to estimate damages for a coastal urban area. Using the historic file in Hazus for the Galveston Hurricane of 1900, and the current Hazus inventory data on the FEMA MSC website, Jane is able to get results close to, but not exactly the same as the old mitigation plan. Additional documentation on FEMA's website indicates that changes have occurred in Hazus software and data over the last five years, and her small differences in results might be accounted for in any number of updates to inventory or hazard data, ArcGIS platform, and her desktop operating system since then.

It occurs to Jane that if the historic storm file is sourced from NOAA, it would be unlikely to have changed much in five years. Similarly, if the aggregated Hazus inventory data was from Census Bureau and other government data around eight years old, it would not have changed either. The contractor was trying to gain a broad estimate of hurricane damages across several counties, so did not use site-specific data that might have conflicted with the Hazus inventory data. Jane wishes there was only one run of the historic storm using the Hazus default inventory data. This would produce the same results each time, and could be used concurrently by anyone involved hurricane mitigation planning for the region. Jane hopes these model runs of Hazus will be available in the future, where she can download consistent results and not have to guess at what was done prior, download Hazus data or software, or re-run Hazus analyses.

Pain points:

- Transparency of processes and methodology
- Reproducibility and credibility of results
- Maintenance and upkeep of default inventory
- Desktop environment: version dependency on 3rd party products
- Streamlined results reporting

3. Federal Response Coordinator

Amy is a Response Coordinator in FEMA Region X. While she is accustomed to managing response for flood and earthquake events, there has been a push lately from the states in her region to boost tsunami preparedness. Tsunami is the latest hazard within Hazus, and because Hazus is a FEMA product, Amy believes she can receive a lot of support in running some analyses for potential tsunami scenarios.

She begins by collecting the data types needed for tsunami analysis. She is able to obtain archived tsunami hazard data from NOAA for a few different historic scenarios, but wants to update the population and building data for most of the coastline areas of both Washington and Oregon. The Hazus inventory data for these areas is a combination of aggregate and notional site-specific, and Amy knows at least some areas in these states have recently done their own tsunami studies with updated inventory.

Amy contacts the State of Washington first, but they express similar issues with data. With the last major census nearly ten years prior, the state is still in the process of working with coastal communities to update their demographic and building inventory. Until they can collect more data, they will continue to use the Hazus inventory data and supplement it with localized findings. The state office tells Amy it is happy to share these data updates as they become available, but there is still a long way to go. Next, Amy contacts the state office in Oregon to make the same request. Oregon, it turns out, has just completed a massive project to generate demographic and inventory data along its coast, for the very purpose of improving tsunami preparedness. However, the office explains to Amy, this was a costly project. Local and state office used numerous contractors, at a high price and in some cases with proprietary methods, to guarantee high quality data. Similarly, their critical infrastructure inventory contains sensitive information that must be kept secure. The Oregon office is concerned about sharing the data with Amy, both due to its cost and sensitive nature, so they agree to share a small subset with critical infrastructure removed.

Amy begins her tsunami analyses in Hazus using the data from the two states, and quickly runs into a number of challenges. First, the data sets are formatted differently, so she must get everything into the same format and schema before importing to Hazus. Second, the Hazus module for importing user-defined inventory (CDMS) does not yet have tsunami functionality built in so she must manually add thousands of records to her individual tsunami study regions. Finally, once Amy completes the manual work of getting both sets of inventory data into Hazus in its proper format, she finds the difference in quality of the datasets yields results of equally varying quality. She is unsure if the results are trustworthy, and is concerned about planning a large-scale evacuation and response effort based on the model results.

In an OpenHazus environment, Amy would benefit most from being able to share datasets quickly and securely. Users who generate data at any level and are comfortable sharing can do so, while other users can download and rate the shared data. Users who have specific security requirements can also protect their data, or make it sharable only with certain other security permission groups. Similarly, maintaining a standard format and schema, including online QA/QC tools, for user-defined inventory data will reduce the burden on users who wish to analyze their own data using Hazus analysis tools, and eliminate the manual steps required currently in the desktop environment.

Pain points:

- Data sharing: user inventory
- Reproducibility and credibility of results
- Transparency of processes and methodology
- Processing time for data input and analysis results

4. Academic researcher

Michael is a graduate student at a major research university, studying civil engineering. He is interested in loading and structural failure for a new building material in manufactured housing. Using the Hazus technical manuals, Hazus software, and supporting SQL database tables for each of the hazards, he is looking for damage functions expressed as equations and ultimately wants to perform sensitivity analyses on the sets of damage functions to determine expected upper and lower limits of damage for the new material. Michael is able to extract some damage functions from flood and earthquake, but he must still add the x and y values to a spreadsheet and then fit a curve to determine the equation. For hurricane and tsunami, he is not able to find even the tables of x and y values. Using the existing tables and curves in the manuals and software, Michael attempts to create his own equations using Excel.

Michael uses his spreadsheets to estimate damages on a small set of test buildings available to him in his lab, without using Hazus. He gets a series of results and then runs the same analyses in Hazus for all four hazards. The results do not align well with his spreadsheet. Michael would like to see the Hazus source code to determine how some of the variables are calculated, and possibly reverse-engineer his damage functions from that, but Hazus source code is not publicly available. With OpenHazus, Michael would benefit not only from better documentation, but also from smaller, modularized sections of code that would allow him to see the equations being used for damage functions, and exactly how the calculations are being performed. If Michael has field or lab test results on the new construction material he is researching, he would also be able to alter the code or create his own code using his own damage functions to run additional analyses.

Pain Points:

- Transparency of processes and methodology
- Reproducibility and credibility of results
- Desktop environment: ease of use

5. Private sector resource and logistics manager

Gabriel works for a national chain of home improvement stores, and is responsible for warehouse, distribution, and retail facilities in the Los Angeles metro area. Gabriel's company also has a number of pre-arranged contracts to support government recovery efforts in the event of a major earthquake. He has been trained on response and recovery procedures, and attends regular meetings with government partners to plan for distribution of recovery materials after an event. Gabriel knows it will be critical in the days following an earthquake to quickly assess the magnitude and scope of damage not only to homes and businesses in the area, but also the facilities he manages.

He creates a database of the company's facilities for the counties in southern California, and loads these into Hazus' advanced module for earthquake engineering (AEBM). After this, he runs a variety of earthquake scenarios – including, historic, probabilistic, and one he made up based on fault line locations. The process involves a lot of manual work, especially preparing and entering the facilities data, and comparing the numerous results reports for the different scenarios. Gabriel is concerned that if an earthquake occurred, he would not be able to complete this process quickly, and would struggle to get reliable data to make decisions in real time about how to protect his facilities from secondary damage, move recovery materials, and meet his contractual obligations.

Gabriel would like to see new functionality in OpenHazus where he is able to pre-load data and scenarios, and store them privately for future use. He would also benefit from more flexible reporting capabilities, to quickly communicate dangers or urgent needs during an active recovery effort.

Pain Points:

- Data sharing: hazard input and user inventory
- Desktop environment: installation, performance, ease of use
- Reproducibility and credibility of results
- Streamlined results reporting

Appendix C: System Requirements

OpenHazus requirements were itemized and expanded in order to provide a basis for evaluating available Open Source technologies for incorporation in future OpenHazus development.

1. Requirements for Desktop Independence

1.1. OpenHazus shall operate primarily in a web environment to include the following:

1.1.1. Default inventory data (file format, file size range)

1.1.2. Repository for public, user-defined inventory data (file format, file size range)

1.1.3. Permission-protected storage for private user-defined inventory data (file format, file size range, permission structure)

1.1.4. Access to authoritative hazard input data sources libraries, to include the following:

1.1.4.1. Flood → Our own (MIP, MSC) repository (see Req X), OFA's

1.1.4.2. Hurricane → NOAA, Hurrevac, ASCE Windfields (probabilistic only)

1.1.4.3. Earthquake → USGS ShakeMap, National Hazard Maps (historic, probabilistic)

1.1.4.4. Tsunami → NOAA PMEL, State product libraries

1.1.5. Access to user-defined/generated hazard data where products are not available from authoritative libraries, such as:

1.1.5.1. Flood → User-defined

1.1.5.2. Hurricane → User-defined

1.1.5.3. Earthquake → User-defined (arbitrary will be handled by either deferral to USGS probabilistic or we will explore options for migrating the existing code to open source)

1.1.5.4. Tsunami → User-defined

1.1.6. Access to damage and loss analysis modules, according to hazard type

1.1.7. Ability to integrate input data (default inventory, user-defined inventory, hazard input, or any combination of these) to damage and loss analysis modules

1.1.8. Documentation, for each analysis module, of the required input data type(s)

1.1.9. Repository of vetted source code modules

1.1.10. Repository of results for pre-run, deterministic scenarios

1.1.11. Repository for public, user-defined results

1.1.12. Repository for permission-protected, private, user-defined results

1.2. A separate OpenHazus will be designed offline on desktop environments for use with secure data such as the Privacy Act protected NFIP policy data or similar applications

1.2.1. OpenHazus desktop utility shall be a fast running (e.g. greater than 10,000 records per second) thin client application based on the OpenHazus site-specific flood POC

1.2.2. OpenHazus desktop utility shall be compatible with the common Windows operating system

1.2.3. OpenHazus desktop utility customizable software script will be available on GitHub for advanced users to customize, enhance, or update

2. Requirements for GIS independence

2.1. Where GIS or spatial components are required to maintain existing functionality, no Esri ArcGIS products will be used

2.2. Traditional GIS interface functions not required for loss estimation (clipping, intersecting, projecting, etc.) will not be included in OpenHazus designs.

2.3. OpenHazus datasets and results will be made available by OpenHazus in a format that is compatible with Open Source GIS Esri for continued analysis outside of Hazus at the user's discretion

2.4. Analysis components requiring spatial operations will be rebuilt using Open Source geospatial technologies with flexibility to be executed in either a web/cloud or desktop environment

2.5. Analysis components that are re-written to meet 2.4 will include site-specific analysis only as a site-specific utility

2.6. Analysis components requiring spatial analysis (e.g. interpolation of a surface during raster generation) shall be migrated and re-architected to a spatial platform separate from Esri products

2.7. Study regions for all hazards shall be created using a spatial or tabular interface

2.7.1. Users shall select the aggregation level and geographic location from a selectable map, or through a tabular menu selection based on jurisdiction names

- **2.7.2.** Study region creation shall be global, and only limited by geography where hazard dictates

2.8. Study regions and model results for all hazards shall be viewable on an interactive, selectable map interface

2.9. Results shall be viewable in an interactive, spatial interface similar or concurrent to the study region interface

2.10. Results shall be viewable in tabular format in addition to interactive map format, with options to export both

2.11. Results shall be viewable in interactive charts and graphics where users can select results by jurisdictions and input data types

2.12. The spatial interface shall be flexible enough in architecture, platform design, and appearance to adapt to future migration to mobile environments, as deemed appropriate by FEMA

3. Requirements for Modularization

3.1. Existing Hazus architecture and source code shall be divided into subsets or modules of existing code, refactored where necessary

3.2. Recommended modules include but are not limited to:

- **3.2.1.** Baseline inventory datasets (maintained and updated by FEMA – add file format and size)
- **3.2.2.** User-defined inventory upload/import, classification, storage, and editing for a variety of common tabular and non-proprietary GIS formats
- **3.2.3.** Import of authoritative hazard data for all hazard modules
- **3.2.4.** User-defined hazard upload/import or creation, storage, and editing for a variety of common tabular and GIS formats
- **3.2.5.** Viewing, editing, and selecting for analysis from existing Hazus damage function libraries for a variety of common tabular formats
- **3.2.6.** Upload/import, storage, editing, and selecting for analysis from user-defined damage functions for a variety of common tabular formats

3.2.7. Rapid (e.g. greater than 10,000 records per second) calculation of hazard severity values at inventory locations

3.2.8. Rapid (e.g. greater than 1,000 records per second) calculation of damage values at inventory locations based on hazard severities and associated damage functions

3.2.9. Calculating social, economic, building and infrastructure losses based on hazard severity and building-specific damage functions

3.2.10. Visualizing and interpreting analysis results

3.3. Modules shall be catalogued in an online library

3.4. Modules shall be re-written as needed in open source languages to allow customization and enhancements by users

3.5. The OpenHazus development team will provide a reference implementation for user-defined customization

3.6. OpenHazus development activities should address defects and enhancements on a rolling basis across individual modules, without impact to other modules

3.7. A uniform process for governance, including vetting and releasing user customizations to modules shall be developed and managed by FEMA

3.8. A regular cycle for soliciting module updates from hazard committees and users shall be developed and managed by FEMA

3.9. Any and all source code that is also open shall be documented according to industry standard

4. Requirements for Data Sharing

4.1. Baseline Results Repository

4.1.1. Analysis results generated using the authoritative probabilistic hazard data source AND baseline Hazus inventory data shall be pre-run by FEMA for the entire U.S.

4.1.2. Analysis results generated using authoritative historic or other deterministic hazard data AND baseline Hazus inventory data shall be pre-run by FEMA for the entire U.S.

4.1.3. Users interested in results described in 4.1.1 and 4.1.2 should be redirected to the results repository

4.1.4. Results for all pre-run scenarios shall be viewable in an interactive spatial interface

4.1.5. Results for all pre-run scenarios shall be queryable and downloadable from the interactive spatial interface, in either spatial or tabular format

4.1.6. Results reporting shall be limited to the formats agreed upon in the interactive spatial interface supporting common user applications including Risk MAP databases, mitigation planning, response and recovery plans, etc.

4.1.7. User customized reporting for deterministic results data shall be supported through the use of downloadable results

4.2. Baseline Hazus Inventory

This inventory will include the Legacy Hazus state database inventory – a standardized dataset of buildings, critical infrastructure, facilities, and components (power lines, pipelines, etc.) for use as input to the hazard scenarios for damage and loss.

4.2.1. Baseline inventory data shall be provided by FEMA in a public web interface for viewing and download

4.2.2. Baseline inventory data shall not be editable within the web interface – user customizations will be performed via user download, then treated as user-defined when uploaded to OpenHazus for use

4.2.3. Baseline inventory data for OpenHazus should discontinue use of the Legacy Hazus aggregated GBS and substitute nationwide site-specific building stock

4.2.4. Baseline Essential Facility inventory for OpenHazus shall be updated with data available through HIFLD Open

4.2.5. Baseline, site-specific inventory data shall remain categorized according to physical structure type, with categories aligning as closely as possible with Legacy Hazus (e.g. Essential Facilities, Bridges, Pipelines, etc.).

4.2.6. Baseline inventory shall be organized by state and inventory type, as in Legacy Hazus, with sub-categories based on available study region types: county, community, census tract, census block, watershed

4.2.7. State data will use an open source database file format, and be subdivided for download by jurisdiction and inventory type

4.2.8. State data will not exceed 10 GB in uncompressed and 2 GB in compressed size to continue to support download and transportability

4.2.9. To accommodate 4.2.6 and 4.2.7, OpenHazus upload/download times for a single state database shall not exceed one hour based on an average internet download speeds for typical users

4.2.10. Modular download of inventory data should be supported where a user selects data by type and jurisdiction for download and update in local environment and re-upload for user-defined analysis (see section 4.4). These tables shall be small and efficient tabular formats (e.g. .csv) easily read by a variety of software.

4.2.11. Dynamic updates via direct FTP or API connection to default inventory sources (American Community Survey, Longitudinal Employer-Household Dynamic datasets, HSIP, HIFLD, etc.) should be evaluated for future OpenHazus development.

4.2.12. To accommodate OpenHazus Inventory data interoperability Google Plus Codes will be introduced for each and every point data facility data table.

4.3. Access to Authoritative Hazard Data

4.3.1. Users must be able to discover and explore existing hazard datasets from authoritative agencies in their study area and select their desired hazard source for direct online integration in their analysis.

4.3.2. Users should be able to upload their own hazard data for integration in their analyses

4.3.3. Standardized QA/QC measures should be established allowing users to transfer their hazard data to a sandbox, where it can be shared more broadly through a public domain repository.

4.3.4. FEMA should develop a governance process that includes approvals and internal quality control.

4.3.5. Flood

4.3.5.1. Flood hazard data shall be sourced from users and external data sources (FEMA libraries, NOAA, etc)

4.3.5.2. Legacy Hazus automated H&H capabilities for generating depth grids will not be migrated to OpenHazus

4.3.5.3. OpenHazus will provide online flood hazard data discovery and act as a repository for depth grids to include a standard QA/QC process for reviewing and accepting data

4.3.5.4. Depth grids in the repository shall be available in open source formats

4.3.5.5. Users in the desktop environment should be able to extract from the online repository individual flood depths based on lat-long coordinates or individual structures.

4.3.5.6. The flood hazard repository shall include a map interface for users to view, query, and download depth grids, or designate them for analysis using other Hazus modules

4.3.5.7. Options for generating a national-level depth grid in future versions of OpenHazus should be evaluated. For additional information, please see Section X.X. (POC write up)

4.3.6. Hurricane

4.3.6.1. Hurricane hazard data for both wind and surge – probabilistic, historic, and NHC Advisory – shall be sourced from ASCE, NOAA or Hurrevac and made available in the OpenHazus repository for online analysis or download

4.3.6.2. Legacy Hazus capability for generating user-defined hurricane scenario windfields shall be migrated to a module in OpenHazus, including the ability to incorporate topographic factors

4.3.6.3. Supplemental hurricane hazard data such as topographic speedup factors, terrain roughness, tree coverage, and debris recovery factors will be available for download and enhancement by the user.

4.3.6.4. Hurricane data in the repository shall be in open source formats

4.3.6.5. Typical file sizes for download and upload will continue to be less than 10 MB consistent with Legacy Hazus hurricane hazard data.

4.3.7. Earthquake

4.3.7.1. Earthquake ground motion hazard data – probabilistic, historic, scenario and actual – shall be sourced from the USGS Earthquake Hazard Mapping and ShakeMap programs

4.3.7.2. Legacy Hazus capability for generating arbitrary, fault-based, and historic epicenter-based earthquake scenarios will not be migrated to OpenHazus

4.3.7.3. Use of user-supplied ground motion hazard data will be supported through an upload function based on common open source files types such as .xml or .shp files.

4.3.7.4. Data in the repository shall be available for download and enhancement by the user in open source file formats (e.g. .shp, .xml).

4.3.7.5. Integration of supplemental hazard data including liquefaction and landslide susceptibility, NEHRP Vs30 soil types, as well as direct ground deformation hazard maps will be supported through an upload capability based on common open source file formats (e.g. .shp and .xml)

4.3.7.6. OpenHazus will provide for discovery and a repository for authoritative earthquake hazard data including liquefaction and landslide susceptibility, NEHRP Vs30 soil types, as well as direct ground deformation hazard maps

4.3.7.7. Typical file sizes for download and upload will continue to be less than 100 MB consistent with Legacy Hazus earthquake hazard data.

4.3.8. Tsunami

4.3.8.1. Tsunami hazard data – including runup, inundation depth, velocity grids, momentum flux grids, as well as deformed and undeformed DEM raster data – shall be sourced from NOAA PMEL, or other authoritative state and local sources

4.3.8.2. Legacy Hazus capability for generating median depth above ground level and median momentum flux grids based on 3 potential levels of input data shall be migrated to one or more modules in OpenHazus

4.3.8.3. OpenHazus shall support the capability for user-supplied upload of tsunami hazard data in common open source raster file formats

4.3.8.4. Integration of supplemental hazard data for evacuation and casualty modeling, including TIGER roadway networks (polyline), NED-based DEM data (raster), and USGS Pedestrian Evacuation Analysis products (polygon) will be supported through online linkages with these U.S. Census and U.S. Geological Survey authoritative datasets.

4.3.8.5. Typical file sizes for download and upload will continue to be less than 100 MB consistent with Legacy Hazus tsunami hazard data.

4.4. User-Defined Data (Inventory, Hazard, Results) Repository

4.4.1. OpenHazus will provide a repository for user-defined inventory, hazard, and results data, in a web environment

4.4.2. File format and transfer times will align with the data types for Hazus default or authoritative data as outlined in previous Data Sharing requirements

4.4.3. Security permissions and access will be regulated to allow levels of access according to user needs, e.g. if a user wishes to make their data public, they can do so.

4.4.4. Public datasets shall be made searchable by location, hazard type, and creator

4.4.5. A process will be established for users to submit user-defined data to become part of the Hazus baseline databases available for other users where applicable

4.4.6. OpenHazus will provide a recommended framework for documenting user-defined metadata, according to FGDC and OGC standards

4.4.7. OpenHazus will provide lightweight desktop analytical capabilities designed specifically for users who need to model hazard losses using Personally Identifiable, proprietary, or protected information.

4.4.8. OpenHazus capabilities for users with secure data such as NFIP policy data will be supported through the use of the downloadable desktop utility so as to not impact the accessibility and processing improvements provided for the majority of Hazus users through a web environment.

4.4.9. Authoritative Federal, State and local users and FEMA partners should be especially encouraged to share datasets publicly

4.4.10. Blank default databases will be provided as templates for international Hazus users

4.4.11. OpenHazus will adhere to the natural hazard catastrophe modelling assumptions and concepts by separating the concerns of inventory and hazard (to be stored separately)

5. Requirements for Automation

5.1. Automated Updates to Baseline Results Repository

5.1.1. Analysis results generated using the authoritative probabilistic hazard data source AND baseline Hazus inventory data shall be automated by FEMA for the entire U.S. to provide updated annualized analyses

5.1.2. Analysis results generated using authoritative historic or other deterministic hazard data AND baseline Hazus inventory data shall be automated by FEMA for the entire U.S.

5.2. Automated Probabilistic Flood Risk Assessment

5.2.1. Automation should be established to support multi-frequency probabilistic or uncertainty analyses for flood, supporting multiple depth grids for each frequency

5.2.2. Automated analysis should be established to support multiple, separate depth grids associated with individual flood types or scenarios, such as breach, pluvial and alluvial flooding scenarios.

5.3. Automated Mitigation Strategy Evaluations

5.3.1. Automated analysis should be established to support multiple mitigation strategy scenarios, including freeboard, NFIP regulations and building code adoption and enforcement

5.3.2. Automated analysis should be established to support analysis of multiple depth grids associated with floodplain mitigation scenarios, such as protection measures and upstream storage scenarios.

5.4. Automated Analysis and Results Dissemination for Response Events

5.4.1. Automated analysis should be established to support analyses for response events, including ShakeMap events associated with a PAGER Yellow or above, landfalling tropical storms and above, as well as when flood depth grids are provided for damaging flood events.

5.4.2. Automated results export and dissemination should be provided and hosted using OpenHazus providing an authoritative source for the run of record.

Appendix D: Open Source Product Review

The purpose of this document is to define the functionality, external interfaces, performance, attributes, and design constraints of OpenHazus. It is based on a Product Requirements Assessment (PRA) focused on the major capabilities and features needed by its stakeholders. This document will not define additional capabilities but focuses on the provision of the capabilities as defined in the PRA. Because Open Hazus will be distributed for free by the federal government, other topics, such as pricing, competition analysis, marketing issues, are not relevant and are not addressed.

Description of Open Source Desktop GIS

There are dozens of open source geospatial desktop applications and it is beyond the scope of this document to describe them all. The following six sub sections are intended to describe well-established, widely supported and implemented GIS packages that fit the scope of OpenHazus. Specialty applications, developed for niche application areas (the Whitebox geospatial analysis tool from the University of Guelph might serve as an example here) as well as applications whose core area is image processing are specifically excluded from the discussion here.

Geographic Resources Analysis Support System (GRASS)

The Geographic Resources Analysis Support System, commonly referred to as GRASS GIS, is used for data management, image processing, graphics production, spatial modeling, and visualization of many types of data. It is Open Source released under GNU General Public License (GPL) >= V2. GRASS GIS is an official project of the Open Source Geospatial Foundation.

Originally developed by the U.S. Army Construction Engineering Research Laboratories (USA-CERL, 1982-1995), a branch of the US Army Corp of Engineers, as a tool for land management and environmental planning by the military, GRASS GIS has evolved into a powerful utility with a wide range of applications in many different areas of applications and scientific research. GRASS is currently used in academic and commercial settings around the world, as well as many governmental agencies including NASA, NOAA, USDA, DLR, CSIRO, the National Park Service, the U.S. Census Bureau, USGS, and many environmental consulting companies. The GRASS Development Team has grown into a multi-national team consisting of developers at numerous locations.

GRASS GIS contains over 350 modules to render maps and images on monitor and paper; manipulate raster, and vector data including vector networks; process multispectral image data; and create, manage, and store spatial data. GRASS GIS offers both a graphical user interface as well as command line syntax for ease of operations.

Raster analysis: Automatic raster line and area to vector conversion, Buffering of line structures, Cell and profile data query, color table modifications, Conversion to vector and point data format, Correlation / covariance analysis, Expert system analysis , Map algebra (map calculator), Interpolation for missing values, Neighborhood matrix analysis, Raster overlay with or without weight, Reclassification of cell labels, Resampling (resolution), Rescaling of cell values, Statistical cell analysis, Surface generation from vector lines

3D-Raster (voxel) analysis: 3D data import and export, 3D masks, 3D map algebra, 3D interpolation (IDW, Regularized Splines with Tension), 3D Visualization (isosurfaces), Interface to Paraview and POVray visualization tools

Vector analysis: Contour generation from raster surfaces (IDW, Splines algorithm), Conversion to raster and point data format, Digitizing (scanned raster image) with mouse, Reclassification of vector labels, Superpositioning of vector layers

Point data analysis: Delaunay triangulation, Surface interpolation from spot heights, Thiessen polygons, Topographic analysis (curvature, slope, aspect), LiDAR

Image processing: Support for aerial and UAV images, satellite data (optical, radar, thermal), Canonical component analysis (CCA), Color composite generation, Edge detection, Frequency filtering (Fourier, convolution matrices), Fourier and inverse Fourier transformation, Histogram stretching, IHS transformation to RGB, Image rectification (affine and polynomial transformations on raster and vector targets), Ortho photo rectification, Principal component analysis (PCA), Radiometric corrections (Fourier), Resampling, Resolution enhancement (with RGB/IHS), RGB to IHS transformation, Texture oriented classification (sequential maximum a posteriori classification), Shape detection, Supervised classification (training areas, maximum likelihood classification), Unsupervised classification (minimum distance clustering, maximum likelihood classification)

DTM-Analysis: Contour generation, Cost / path analysis, Slope / aspect analysis, Surface generation from spot heights or contours

Geocoding: Geocoding of raster and vector maps including (LiDAR) point clouds

Visualization: 3D surfaces with 3D query (NVIZ), Color assignments, Histogram presentation, Map overlay, Point data maps, Raster maps, Vector maps, Zoom / unzoom -function

Map creation: Image maps, Postscript maps, HTML maps

SQL-support: Database interfaces (DBF, SQLite, PostgreSQL, mySQL, ODBC)

Geostatistics: Interface to "R" (see section 0), Matlab, and other software packages

Temporal framework: support for time series analysis to manage, process and analyze (big) spatio-temporal environmental data. It supports querying, map calculation, aggregation, statistics and gap filling for raster, vector and raster3D data. A temporal topology builder is available to build spatio-temporal topology connections between map objects for 1D, 3D and 4D extents.

Furthermore: Erosion modeling, Landscape structure analysis, Solution transport, Watershed analysis.

Quantum GIS

Quantum GIS, better known by its acronym QGIS, is the leading open source ***desktop*** GIS. It allows users to create, edit, visualize, analyze and publish geospatial information on Windows, Mac OS, Linux, BSD and Android (via the QField app). There is also an OGC Web Server application, a web browser client and developer libraries. QGIS is licensed under the GNU General Public License. The QGIS project is under very active development by an enthusiastic and engaged developer community with good mechanisms for help via stack exchange, mailing lists and through a global network of commercial support providers.

QGIS has grown from a simple ArcView-like shapefile manipulation program to a full-fledged GIS that mirrors to a large degree the concepts and functionalities of ArcGIS. With this came a now confusing plethora of UX items that lacks the streamlining that Esri has gone through in its development from ArcGIS Desktop to ArcGIS Pro. However, if one were to base a desktop implementation of Open Hazus on QGIS, then it would be possible for developers to customize the user interface to the specific needs of a hazards-oriented spatial decision support system.

QGIS connects the spatial versions of Postgres, SQLite, MS SQL, and Oracle and reads through the OGR and GDAL libraries every conceivable raster and vector format, as well as OGC web services. QGIS has morphed from being its own desktop GIS to a shell that integrates open source GIS functions from GRASS, SAGA, gvSIG, and others. Like ArcGIS, it is relatively poor with regard to manipulating attribute data, which in both cases is better done outside the GIS.

While most core functions are written in C, QGIS offers both a Python console as well as a plethora of Python plugins or extensions provided by developers world-wide.

User-friendly Desktop Internet-oriented GIS (uDig)

uDig is an open source spatial data viewer/editor, with special emphasis on the OGC standards for Internet GIS, the Web Map Server (WMS) and Web Feature Server (WFS) standards. The goal of uDig is to provide a complete Java solution for desktop GIS data access, editing, and viewing. There are standalone versions for Windows, Mac OS and Linux, however, for the purposes of Open Hazus, the application framework, built with Eclipse Rich Client (RCP) technology might be more promising. Their gallery of sample applications built with uDig illustrates implementations in hydrology, logistics, forest management, and the Distant Early Warning System for Tsunamis. uDig is also the underlying platform for DIVA GIS.

DIVA-GIS is a free GIS used for the analysis of geographic data, in particular point data on biodiversity, published under the GNU General Public license. The software was first designed for application to the study of wild potatoes in South America. DIVA-GIS was developed as a joint project by the International Potato Center in Peru, the International Plant Genetic Resources Institute, the Museum of Vertebrate Zoology at the University of California at Berkeley, the Secretariat of the Pacific Community, and the FAO. DIVA-GIS has a wide range of tools for evaluation of mapping and modeling of habitats. DIVA-GIS can process data in all standard GIS formats, including data from Esri's ArcGIS programs. The program runs on Windows and OS X. DIVA raster files generated may be imported and exported into R or the modeling program Maxent.

The previously discussed GRASS and QGIS programs were developed with end users in mind, i.e., a lot of effort has been expanded on provided a one-stop solution for the user's GIS needs. uDig's philosophy is different in that it acts more like a (Java-based) framework that allows (experienced Java) developers to write their own GIS (like, for example, JGRASS) based on the Eclipse platform.

System for Automated Geoscientific Analysis (SAGA)

SAGA started out as a set of programs the quantitative description of topographies (landform shapes). It has now become a serious competitor for the raster-based GRASS GIS. SAGA's overarching goal is to provide (geo-) scientists with an effective API for the implementation of geoscientific methods. Conceived of in the early 2000s and unencumbered by legacy code, SAGA's true strength lies in its architecture and very fast code.

SAGA is written in C++ and has an object oriented system design. It uses the cross-platform GUI library wxWidgets for user interface functionality. Because wxWidgets enables operating system independent software development, SAGA runs on MS-Windows as well as on Linux.

One of the attractions of SAGA is its compact modular design that allows running the software without installation even from flash drives. It has an unusually flexible command line interface that supports scripting in Python, Java, and R. There are more than 450 raster functions for geodata analysis (including a good amount of image processing and geostatistics) – but only minimal vector GIS support for clipping, buffering and raster to vector conversion.

Description of Open Source Web Mapping Applications

The OSGeo Foundation supports some ten web mapping applications. Detailed descriptions can be found at https://www.osgeo.org/projects. The following notes concentrate on what might be useful in the context of this document, including non-OSGeo projects like Carto.

MapServer

MapServer is one of the oldest web mapping engines around. Written in C, it runs on all major platforms, has some pretty advanced cartographic features and supports most OGC map-related standards. Although it can be customized to have a relatively light footprint, the full-fledged version with all bells and whistles is so feature-rich that it can be daunting to maintain. Best suited for developers who are not afraid of editing configuration files. MapServer can be used as a CGI or a plug-in to various languages.

MapGuide

Where MapServer is the well-established can-do-everything behemoth, MapGuide offers a very practical subset of functionalities that are easy to link with a server-side GIS. The cartography is not sophisticated but suitable for web-based applications. Unusual for a web *mapping* application, MapGuide supports a number of basic measurement and topology operations. It has a modern XML-based resource library that allows for a high degree of customization.

deegree

Java-based degree differs from the two introduced web mapping applications in one important way: it has been written for spatial data infrastructures and includes OGC-compliant packages for cataloguing, data access, discovery and security. It handles all queryable properties of the Dublin Core and supports KVP as well as XML and SOAP requests. It is fully transactional, even for rich data models and supports a flexible mapping of GML application schemas (like the CityGML mentioned as a basis for the Open Hazus building inventory) to relational models. This makes degree uniquely qualified to become a component of the Open Hazus server – even if it is more commonly used in Europe rather than the United States.

GeoMOOSE

GeoMOOSE is a browser-based mapping framework for displaying distributed cartographic data. It is particularly useful for managing spatial and non-spatial data within county, city and municipal offices (from which GeoMoose originated). It extends the functionality of MapServer and OpenLayers to provide built in services, like drill-down identify operations for viewing and organizing many layers, selection operations and dataset searches. GeoMOOSE is fast, performing well with hundreds of layers and/or services at a time. Data from multiple custodians can be maintained with different tools and on different schedules as each map layer has its own set of configuration files for publishing, symbols, templates as well as source data. The user interface is easily configurable, and additional services can be added through a modular architecture. GeoMOOSE allows for distributed maintenance amongst multiple owners – useful in Open Hazus' multi user environment. Similar to degree (although not as sophisticated), it supports discovery and filtering from data catalogs.

Mapbender

Mapbender is a content management system for geospatial data services and map applications. Mapbender is mostly written in JavaScript and based on the frameworks Symfony, JQuery and OpenLayers. Mapbender is platform independent and can be used on Windows, Mac and Linux. Mapbender has its own template for mobile applications, which has been optimized for use on smart phones and tablets. Administration of OGC Services in a Service repository. Individual configuration of Services in every application. Customization of applications to individual requirements via the web interface with a collection of customizable elements. The adaption of the designs can be done via customizable templates and CSS. Based on the integrated user management, individual applications, single functionality and services can be assigned to specific users or groups.

GeoServer

GeoServer is an open source software server written in Java that allows users to share and edit geospatial data. It is the reference implementation of the OGC's Web Feature Service (WFS) and Web Coverage Service (WCS) standards, as well as a high performance certified compliant Web Map Service (WMS).

Started in 2001, GeoServer has become the more modern competitor of MapServer. Those involved with GeoServer project also founded the GeoTools project (see 0), an open source GIS Java toolkit. Together with PostGIS (see 0) and OpenLayers, and with significant support by the company Boundless, GeoServer has become the industry standard in web mapping software. Similar to degree, it features cataloging services (although not as comprehensive as those of degree) and web processing services (see also 0). GeoServer is comprehensive enough to require either a dedicated staff person or the kind of commercial managed services that allows Boundless to place this product in the public domain and still stay in business. GeoServer is widely used in US federal, state, and larger local government agencies. GeoServer has just been ported to run on Java 11.

Carto

Carto (formerly CartoDB) is a complete web-based GIS that is entirely based on dozens of open source packages and hence itself available in source code on GitHub. Similar to GeoServer, the plethora of packages (as of December 2018, more than 300) that make up Carto is so big that a company, Vizzuality, makes its living by hosting customized Carto GIS server instances and developing specialized front and back ends. Carto does an excellent job of bridging the gap between complex data analysis under the hood and intuitive, dash board-based user interfaces that are serving a wide range of stakeholders. Carto has been shifting its emphasis from that of Web-GIS to a big geospatial data visualization platform.

PyWPS

PyWPS is not a classic web mapping application but a platform for web-based geoprocessing services – in this case based on Python. PyWPS implements the OGC web processing services specification and integrates with GRASS, GDAL, R, Fiona, RasterIO, MapServer, and QGIS. It is ideal for setting up, running and archiving all Open Hazus process models.

JavaScript Libraries

JavaScript libraries serve mainly as the front-end of web-based solutions facing the end user. The choice of programming framework (not covered here) can be confusing but the review of the following six packages covers the current state of the art for geospatial applications.

OpenLayers

OpenLayers is a JavaScript library that allows for putting a dynamic map in any web page. It can display map tiles and markers loaded from any source. In contrast to the previously discussed applications, OpenLayers is not a full-fledged web mapping server software but an easy to use library that allows for light-weight web applications, leveraging WebGL and HTML5. This makes it eminently suitable for supporting mobile applications.

Leaflet

Leaflet is a younger web mapping API that is designed to be lightweight, mobile-friendly, and easy to get started with. It places heavy emphasis on the use of tiled maps and client-side vector graphics drawn from sources such as GeoJSON. For basic maps that use these layer types, Leaflet is an excellent choice that has already endeared itself to many GIS developers. Impressive examples for spatio-temporal visualizations can be found at http://apps.socib.es/Leaflet.TimeDimension/examples/.

D3

D3.js, an acronym for Data-Driven Documents, is a JavaScript data visualization library that is frequently used for charting but also contains many map examples. It binds data elements to the page's document object model (DOM), allowing for interesting and flexible data animations and transitions. D3 is a nice option for composing a web app with interactive maps and charts. It also offers examples for using non-Mercator projections. Although D3 has originally been designed as a visualization tool, the library contains almost as many geospatial as visualization functions, especially once one considers shape and scale-related routines to fit neatly into either domain.

turf

Turf is a JavaScript library for spatial analysis. It includes traditional spatial operations, helper functions for creating GeoJSON data, and data classification and statistics tools. Although developed by Mapbox, it is not a plug-in but a standalone JavaScript library that works with any web mapping package.

Cesium

CesiumJS is a JavaScript library for world-class 3D globes and maps. It uses WebGL for hardware-accelerated graphics, and is cross-platform, cross-browser, and tuned for dynamic-data visualization. Cesium does not actually provide analytical functionality but makes for attractive fly-through visualizations of past and potential hazards. Melbourne's Intelligent Disaster Decision Support System (IDDSS) is a fine example for that.

kepler.gl

Kepler.gl is a data-agnostic, high-performance web-based application for visual exploration of large-scale geolocation data sets. In the context of Open Hazus, it could be used for client-based exploratory analysis of massive (multi-million events) incidence data. The developers at Uber, the company where kepler.gl was developed, have been using it for dashboarding apps, and it is in this context, that this library could provide some of the functionality that has made Carto such a successful (easy to use) Web GIS.

Geospatial Libraries

Geospatial libraries are low-level resources for software developers. They underlie many of the previously discussed web-based and desktop GIS applications. Here, we list only libraries that serve classic GIS tasks; others, like for example image processing and remote sensing-related libraries are beyond the scope of this document.

GeoTools

A classic example for an open source geospatial library is GeoTools. GeoTools is a Java code library that provides standards compliant methods for the manipulation of geospatial data. It implements OGC specifications as they are developed. The OGC holds the copyright on the library and has made it available under the LGPL license. Based on the open source Java Topology Suite (JTS), GeoTools provides interfaces for key spatial concepts and data structures. It features a stateless, low memory renderer that is particularly useful in server-side environments.

GDAL/OGR

There is no commercial or open source GIS product in active development anymore that does not rely on the geographic data abstraction library or GDAL. GDAL is a C++ translator library for 240+ raster and vector formats. Like GeoTools above, it is now managed and licensed by OGC. Its API serves C, C++, Python, Perl, C# and Java.

GEOS

The Geometry Engine – Open Source library or GEOS for short, is a C++ port of the Java Topology Suite. As such, it aims to contain the complete functionality of JTS in C++. This includes all the OGC Simple Features for SQL spatial predicate functions and spatial operators, as well as specific JTS enhanced topology functions.

GeoTrellis

GeoTrellis is a Scala library and framework that uses Apache Spark to work with raster data. It is released under the Apache 2 License. GeoTrellis reads, writes, and operates on raster data as fast as possible to transform user interaction with geospatial data by bringing the power of geospatial analysis to real time, interactive web applications.

GeoTrellis is a general framework for low-latency geospatial data processing developed using Scala and Akka. The goal of the project is to transform user interaction with geospatial data by bringing the power of geospatial analysis to real time, interactive web applications. It is complementary to other open source geospatial projects such as GeoServer, OpenLayers and PostGIS. GeoTrellis aims to:

- Create scalable, high performance geoprocessing web services;
- Create distributed geoprocessing services that can act on large data sets; and
- Parallelize geoprocessing operations to take full advantage of multi-core architectures

GeoTrellis is designed to help a developer create simple, standard REST services that return the results of geoprocessing models. In an application that relates to Open Hazus, the USACE Institute for Water Resources (IWR) is using the GeoTrellis framework for their Water Infrastructure Systems Data Manager to supports budget decisions based on multiple geographic criteria with real-time visual feedback as assumptions are interactively adjusted.

GeoWave

GeoWave is an open source library that leverages the scalability of a distributed key-value store for effective storage, retrieval, and analysis of massive geospatial datasets. Whereas Geo Trellis is more raster oriented, GeoWave adds multi-dimensional indexing capability to Apache Accumulo, HBase and Cassandra as well as Google's BigTable and Amazon DynamoDB, thereby supporting geographic objects and operators. The library comes with a plug-in for GeoServer that allows geospatial data in Accumulo to be shared and visualized via OGC standard services.

GeoWave provides Map-Reduce input and output formats for distributed processing and analysis of geospatial data, as well as a PDAL plug-in for interacting with point cloud data in Accumulo through the PDAL library. As such, GeoWave attempts to do for distributed key-value stores as PostGIS does for PostgreSQL (see also 0).

Spatio Temporal Asset Catalog

The SpatioTemporal Asset Catalog (STAC) specification aims to standardize the way geospatial assets are exposed online and queried. A spatiotemporal asset is any file that represents information about the earth captured in a certain space and time. While the initial focus is primarily remotely-sensed imagery, the core is designed to be extensible to SAR, full motion video, point clouds, hyperspectral, LiDAR and derived data like NDVI, Digital Elevation Models, mosaics, etc.

The goal is for all major providers of imagery and other earth observation data to expose their data as SpatioTemporal Asset Catalogs, so that new code doesn't need to be written whenever a new JSON-based REST API comes out that makes its data available in a slightly different way. This will enable standard library components in many languages. STAC can also be implemented in a completely 'static' manner, enabling data publishers to expose their data by simply publishing linked JSON files online.

Spatially Enabled Platforms

HAZUS has entirely been based on GIS. This chapter of the document has so far been based on geospatial solutions that are closely tied to traditional GIS, even as the horizon expanded to open source desktop and web-based applications, as well as some of the lower-level libraries that these applications are based on. Two other aspects deserve consideration though. One is that awareness of geospatial functionality has spread beyond traditional GIS audiences, resulting in an increasing number of software packages that provide more than enough of the geospatial functionality needed for Open Hazus. The other is the fact that (Open) Hazus can be seen as an instance of a more generic family of software called spatial decision support systems (SDSS). There are multiple ways to implement an SDSS but increasingly, this is not on top of a GIS but like Repast Simphony discussed beneath, as a set of interlinked libraries that are embedded in general purpose development environments.

R

R is a language and environment for statistical computing and graphics. Readers of this document might be surprised that R is considered a general enough platform to allow an Open Hazus to be built on top of it. But R has a huge developer community, including in all aspects of a spatial decision support system. There are very active communities for geo-spatial, operations research and machine learning tools. Packages like rattle() illustrate how convenient it is to develop a workflow-based user interface. The main drawback to conceiving of R as an SDSS platform is the underdeveloped link to databases. There are plenty of packages that aim to provide links to DBMS, but they are slow and a de facto bottleneck. Calling R routines from a DBMS would be far more convincing.

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term “environment” is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in R, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and FORTRAN code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

R can be extended (easily) via packages. There are about 13,640 packages available on the Comprehensive R Archive Network (CRAN) that extend the core functionality of R in, among others, the realms of Bayesian modeling, cluster analysis, environmetrics, machine learning, official statistics, spatio-temporal analysis, and web technologies.

Repast Simphony

Repast (Recursive Porous Agent Simulation Toolkit) is probably the most widely used free and open-source, cross-platform, agent-based modeling and simulation toolkit, developed at by Argonne National Laboratory and is now managed by the non-profit volunteer organization ROAD (Repast Organization for Architecture and Development). Repast provides a core collection of classes for the building and running of agent-based simulations and for the collection and display of data through tables, charts, and graphs. A particularly attractive feature of Repast is its ability to integrate GIS data directly into simulations.

Repast has been released in five versions supporting model development in various programming languages,

- Repast (Java based)
- RepastPy (based on the Python Scripting language)
- Repast.Net (implemented in C# but any .Net language can be used)
- RepastS (Repast Simphony, Java based, designed for use on workstations and small computing clusters)
- Repast High Performance Computing (an expert-focused C++-based modeling system designed for use on large computing clusters and supercomputers)

.. as well as built-in adaptive features, such as genetic algorithms and regression.

Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins for more than 25 languages. Plug-ins can be plugged-stopped dynamically.

From an Open Hazus perspective, it would be important to hide compiling, debugging, etc. components of the user interface in favor of the model definition, diagramming, testing, publishing, etc. tasks. The main reason to go through this effort is the universality of the tool. As Eclipse supports all the previously mentioned libraries, it is a convenient provisioning platform to manage all aspects of Open Hazus. The above mentioned Repast Simphony is a prime example for such an approach.

Jupyter

A Jupyter notebook is an excellent means of capturing code, text, widgets, graphics, and other rich media in a computational narrative that distills data into insights. Today, notebook authors can share their notebooks for others to view and run in the Jupyter Notebook web application. Jupyter Notebooks are revolutionizing the way engineers and data scientists work together. Jupyter is built for writing and sharing code and text, within the context of a web page to build a "computational narrative that distills data into insights." (IBM 2015[4])

While Jupyter's roots are in Python, it is now multi-lingual. Language support comes by way of modular kernels for more than 50 programming languages. The Toree kernel supports the Scala programming language and the Spark distributed computing platform (plus Python and R), simplifying the task of building large, data-intensive distributed applications.

Jupyter comes in three parts, a notebook frontend, a Jupyter server, and a kernel protocol. The frontend is a JavaScript web application that supports real-time dashboards. Jupyter can be easily extended. There are widgets for creating maps based on OpenStreetMap, and interactive data visualization with d3.js (see0). The Urban Data Science Toolkit is a fine example for an Open Hazus-like SDSS on Jupyter.

Geospatial Databases

Open Hazus will require extensive database support both on the server and client side. As a large percentage of the data is geospatial in nature, it is important to have that aspect being explicitly supported

[4] http://blog.ibmjstart.net/2015/08/22/dynamic-dashboards-from-jupyter-notebooks/

by the database(s) of choice. In addition, especially the server-side database should have OLAP support to facilitate the mining of results databases. Finally, as the requirements for the inventory databases, depth grids, and possibly spatio-temporal features of dynamic hazard models keep increasing the storage and accessibility demands, non-relational options that still support geospatial needs should be considered either for future generations of Open Hazus or as a separate profile of the open solution set.

PostGIS

PostGIS is a spatial database extension for the PostgreSQL DBMS. It provides new types to PostgreSQL geometry, geography, raster, and topo-geometry and SQL/MM OGC Simple Feature SQL compliant functions for doing GIS work such as cadastral management, back-end for Web mapping services. It can also be used for network and road-routing using the complimentary pgRouting PostgreSQL extension. Although there are spatial extensions to many other commercial and open source relational DBMS, PostGIS has become the de facto standard, definitely in the open source community but increasingly also in enterprise environments that have a well-established DBMS infrastructure in support of non-geospatial applications. PostGIS is unique in the DBMS domain for the range of geospatial support it provides, including:

- Processing and analytic functions for both vector and raster data for splicing, dicing, morphing, reclassifying, and collecting/unioning with the power of SQL
- raster map algebra for fine-grained raster processing
- Spatial reprojection SQL callable functions for both vector and raster data
- Support for importing / exporting Esri shapefile vector data via both command line and GUI packaged tools and support for more formats via other 3rd-party Open Source tools
- Packaged command-line for importing raster data from many standard formats: GeoTIFF, NetCDF, PNG, JPG to name a few
- Rendering and importing vector data support functions for standard textual formats such as KML, GML, GeoJSON, GeoHash and WKT using SQL
- Rendering raster data in various standard formats GeoTIFF, PNG, JPG, NetCDF, to name a few using SQL
- Seamless raster/vector SQL callable functions for extrusion of pixel values by geometric region, running stats by region, clipping rasters by a geometry, and vectorizing rasters
- 3D object support, spatial index, and functions
- Network Topology support

- Packaged Tiger Loader / Geocoder/ Reverse Geocoder / utilizing US Census Tiger data

SpatiaLite

Like PostGIS to Postgres, SpatiaLite is a SQLite database engine with spatial functions added. Each SQLite database is simply a file that can be copied, compressed and ported between MS Windows, Linux, Mac OS, etc. The SpatiaLite extension enables SQLite to support spatial data conformant to OGC specifications. SpatiaLite is *not* intended for server-side use and should not be used for databases beyond a few GBytes. However, it works very well for desktop and mobile applications (including on Chrome OS and Android devices). In particular, SpatiaLite offers:

- Standard WKT and WKB formats
- SQL spatial functions such as AsText(), GeomFromText(), Area(), PointN() …
- OpenGis spatial analysis functions provided via GEOS, such as Overlaps(), Touches(), Union(), Buffer() …
- Full Spatial metadata in line with OpenGis specifications
- Numerous Geometry notations - EWKT, GML, KML, and GeoJSON
- Importing and exporting to shapefiles
- Coordinate reprojection via PROJ.4 and EPSG geodetic parameters dataset
- Locale charsets via GNU libiconv
- Spatial Index based on the SQLite's RTree extension
- Access shapefiles as VIRTUAL TABLEs, enabling standard SQL queries on external shapefiles, without importing or converting them
- Access external CSV/TxtTab files or xls spreadsheets as VIRTUAL TABLEs
- Access XML documents, stored BLOB compressed binary objects, including syntactic "well formed" and XSF schema validation constrained checks. Specific support for ISO-Metadata, SLD/SE styles and SVG graphics.XML documents can be queried using standard XPath syntax.
- Query external WFS servers.
- Parse external DXF files (all versions) and store layers and geometries found.
- Generate and Export DXF files

Rasdaman

Rasdaman, short for raster data manager, allows storing and querying massive multi-dimensional arrays, such as sensor, image, simulation, and statistics data appearing in domains like earth, space, and life science. This array analytics engine distinguishes itself by its flexibility, performance, and scalability. Rasdaman can process arrays residing in file system directories as well as in databases. Rasdaman is the first fully implemented, operationally used system with an array query language and optimized processing engine with unprecedented scalability. Known Rasdaman databases exceed dozens of TB; EarthServer, is establishing intercontinental fusion of Petabyte datacubes. Rasdaman integrates smoothly with R, OpenLayers, Leaflet, NASA WorldWind, GDAL, MapServer, or Esri ArcGIS.

OLAP Databases

Relational databases are good for traditional ETL tasks but rather limiting in the context of mining the results of ensembles of model runs. OLAP databases address the rapid access to analysis results question, allowing the interactive analysis of multidimensional data from multiple perspectives. OLAP hypercubes are dissections of the multidimensional database, similar to how an octree tessellates a 3D space. Each cube contains aggregated data related to elements along each of its dimensions. A typical user interface to interact with the data cubes is a Pivot table. The combination of all possible aggregations plus the base data contains the answers to every query that can be answered from the data.

Kylin

Built on top of Hadoop/Spark Apache Kylin has a SQL interface, and can be used to carry out OLAP multidimensional analysis on Hadoop supporting extremely large datasets. The Kylin OLAP Engine is made up of a metadata engine, a query engine, a job engine and a storage engine. It also includes a REST Server to service client requests. The addition of support for RDBMS data sources means that data in formats such as Oracle, SQL Server and MySQL can be used within the analyses. Tools such as Apache Sqoop can also be used to export the data from RDBMS to HDFS to make it easier for Kylin to get the data and then build that into cubes. Kylin's ODBC and JDBC drivers support tools such as Tableau for immediate graphical inspection.

Pinot with SpatialHadoop

Another classic OLAP database is the column-oriented Pinot datastore developed by LinkedIn. The data is stored in an HDFS (Hadoop distributed file system), which allows it to be linked with SpatialHadoop (2018). Hadoop's Pig Latin in turn can be extended via user-defined functions, which users can write in Java, Python, JavaScript, Ruby or Groovy, and in the case of SpatialHadoop mimic PostGIS's functionality. As a Java application, Pinot runs on all virtually operating systems.

SpatialHadoop is an extension to Hadoop that provides efficient processing of spatial data using MapReduce. It provides spatial data types to be used in MapReduce jobs including point, rectangle and polygon. It also adds low level spatial indexes in HDFS such as Grid file, R-tree and R+-tree. SpatialHadoop extends Hadoop in a similar way as PostGIS does with Postgres, providing spatial operations that are implemented as MapReduce jobs that access spatial indexes using the new components. Developers can implement myriad spatial operations (e.g., range query, k-nearest neighbor and spatial join) that run efficiently using spatial indexes.

OpenCube Toolkit

The European OpenCube Toolkit (2018) is a fine example of how all this can be put together to process RDF data cubes. All user interactions are browser-based, including the Pivot table view and all OLAP operations. The OpenCube Toolkit executes R scripts, has interactive visualization widgets, and even a map interface in support of OLAP operations on the geo-spatial dimension.

Non-relational Databases

There is a little of an overlap with the previous section as some of the open source OLAP databases, when based on the HDFS, are also non-relational. Non-relational databases are typically built for specific data models (document, graph, key-value, etc.) and have very flexible schemas. They also scale very well. The tradeoff is that non-relational databases cannot guarantee the four hallmarks of Codd’s (1970) RDBMS, namely atomicity, consistency, isolation and durability (although RavenDB claims to be ACID). Excluded from this preview here are the two popular NoSQL databases from Amazon (DynamoDB) and Google (Firebase) because they do not qualify as free and open source (FOSS). The same holds for MongoDB, which is published under a so-called “server-side public license”, which does not fit the needs of the Open Hazus project.

CouchDB

Apache’s Cluster of Unreliable Commodity Hardware (Couch) database features a document-oriented NoSQL architecture to store JSON data that can be queried with JavaScript. The main design goal was to have software that scales from mobile phones (it runs on Android and iOS) to cloud-based Big Data stores. CouchDB has drivers for over 15 programming languages. One of CouchDB’s strengths is the ability to synchronize two copies of the same database. Its replication feature supports both transient and persistent states, which might prove useful for multiple instances of Open Hazus' inventory databases. The replication protocol should address concerns of those 5% of Hazus users who require their own offline databases. CouchDB does not know of spatial data types or functions (other than the above radius around a point query) but it has the ability to create geospatial indices and to run radius queries using a number of distance measures around point locations.

Orient DB

OrientDB is a schema-free, graph-based database, written in Java, with a SQL-like query language but no ability to perform classic relational joins. It has drivers for about a dozen popular programming languages and runs a choice of JavaScript or Java server-side scripts. As of 2016, OrientDB has a spatial module that supports a number of vector geometries and a subset of SQL-MM functions (the only coordinate system supported is WGS84). Geospatial data can be imported and exported as CSV files with geometries in WKT. The original developer spun off a company OrientDB, which has an interesting Open Hazus-related case study on their website.

redis

The Remote Dictionary Server (redis) is the most widely used NoSQL database implementing a distributed, *in-memory* key-value database with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, streams and spatial indexes with radius queries. Redis is written in ANSI C and hence works on most operating systems (Microsoft is maintaining the Windows version). The internal scripting language is Lua but there are drivers for some 35 programming languages. Redis does not know of spatial data types or functions (other than the above radius around a point query) but there is a Geodis library on GitHub that provides geohashing to index locations. Redis has been benchmarked as the world's fastest database but this comes at the price of being limited to in-memory operation, which precludes typical Open Hazus applications.

Raven DB

RavenDB, developed by a company that goes by the name of Hibernating Rhinos, is available on GitHub in source code and precompiled for a number of different operating systems for small to medium sized applications. But to use, for example, SQL ETL one would have to purchase commercial licenses at not insignificant costs. Unusual for open source software, RavenDB is.net-based (yet runs on many operating systems and has drivers for six popular programming languages). What really sets RavenDB apart is the fact that despite being a NoSQL database it supports all ACID requirements and most important for Open Hazus, RavenDB has built-in geo-spatial indexing, spatial data types (WKT) and four spatial relationships. It supports server-side scripts, concurrency, and Hadoop. As the database itself has relatively small demands on hardware resources, it might serve well as a particular Open Hazus application profile for users that have streaming data demands.

Cassandra

Originally developed in-house at Facebook, Cassandra is now an Apache NoSQL database that is geared to provide high availability across multiple datacenters. The obvious use case for Open Hazus is in case of a Katrina- or Sandy-like event that might cause regional web services to be unavailable. Cassandra's own query language has drivers for a dozen common programming languages. Brahmin *et al.* (2016) developed a spatial extension for Cassandra. An example for its application can be found in the tutorials for GeoMesa (see also 0).

GeoWave

GeoWave is a OSGeo library that connects the scalability of distributed computing frameworks and key-value stores with modern geospatial software to store, retrieve and analyze massive (i.e., terabyte) geospatial datasets. It utilizes distributed computing clusters and server-side fine grain filtering to execute interactive time and/or location specific queries on datasets containing billions of features.
GeoWave adds multi-dimensional indexing to Accumulo, HBase, BigTable and others. It indexes multidimensional data in a way that ensures values close together in multidimensional space are stored physically close together in the distributed datastore of a client's choice. GeoWave allows for easy feature extension and platform integration – bridging the gap between distributed technologies and minimizing the learning curve for developers. A GeoServer plug-in allows geospatial data in a GeoWave datastore to be shared and visualized via OGC standard services. In addition, GeoWave provides Map-Reduce input and output formats for distributed processing and analysis of geospatial data.

Appendix E: Analysis Models

Software Evaluation Table for OpenHazus

Lookup Table for the Numbering of System Features in Tables

1 OpenHazus Shall Operate Primarily in a Web Environment

1.1 Default inventory data

1.2 Repository for public, user-defined inventory data

1.3 Permission-protected storage for private user-defined inventory data

1.4 Access to authoritative hazard input data sources

1.4.1 Flood, our own repository

1.4.2 Hurricane -> NOAA, Hurrevac

1.4.3 Earthquake USGS ShakeMap

1.4.4 Tsunami NOAA PMEL

1.5 Access to hazard generation modules

1.5.1 Flood

1.5.2 Hurricane, probabilistic, user-defined

1.5.3 Earthquake, arbitrary, probabilistic, user-defined

1.5.4 Tsunami, user-defined

1.6 Access to damage and loss analysis modules

1.7 Ability to integrate input data

1.8 Documentation, for each analysis module, of the required input data type(s)

1.9 Repository of vetted source code modules

1.10 Repository of results for pre-run, deterministic scenarios

1.11 Repository for public, user-defined results

1.12 Repository for permission-protected, private, user-defined results

1.13 Thin client

1.14 Operating system compatibility

1.14.1 Windows

1.14.2 Mac OS

1.14.3 Linux

1.14.4 Chrome

1.14.5 iOS

1.14.6 Android

2 Esri ArcGIS Independence

2.1 Maintain existing GIS functionality

2.2 OpenHazus will not have any dependency on Esri ArcGIS products

2.3 Ability to maintain ArcGIS output format compatibility

2.4 Vector analysis components in open source

2.5 Raster analysis components in open source

2.6 Map interface for study regions

2.6.1 User can choose granularity/aggregation level

2.6.2 Study region creation anywhere on Earth

2.7 Model results for all hazards shall be viewable on an interactive, selectable map interface

2.8 Results shall be viewable in an interactive, spatial interface similar or concurrent to the study region interface

2.9 Results viewable as tables in addition to interactive map format, with options to export both

2.10 UI adaptability to possible future mobile apps

3 Modularization

3.1 Existing Hazus code shall be chopped up into smaller modules

3.1.1 Default inventory datasets

3.1.2 User-defined inventory upload/import, classification, storage, and editing (file format and size)

3.1.3 Import of authoritative hazard data

Desktop GIS

.

Feature		***GRASS***		***QGIS***		***uDig***		***gvSIG***		***SAGA***		***OpenJUMP***	
	Weight	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score
I		*OpenHazus Shall Operate Primarily in a Web Environment*											
I.1	100	100	100	100	100	100	100	100	100	100	100	100	100
I.2	100												
I.3	80	100	80	100	80	100	80	100	80	100	80	100	80
I.4	100						0						
I.4.1	100	100	100	100	100	100	100	100	100	100	100	100	100
I.4.2	90	67	60	67	60	67	60	67	60	67	60	67	60

I.4.3	90	67	60	67	60	67	60	67	60	67	60	67	60
I.4.4	80	67	54	67	54	67	54	67	54	67	54	67	54
I.5		*Access to hazard generation modules*											
I.5.1	100	100	100	100	100	33	33	100	100	67	67	33	33
I.5.2	70	100	70	100	70	33	23	100	70	67	47	33	23
I.5.3	60	100	60	100	60	33	20	100	60	67	40	33	20
I.5.4	55	100	55	100	55	33	18	100	55	67	37	33	18
I.6	60	100	60	100	60	100	60	100	60	100	60	100	60
I.7	90	100	90	100	90	100	90	100	90	100	90	100	90
I.8	70	67	47	67	47		0		0		0		0
I.9	20	100	20	100	20	100	20	100	20	100	20	100	20
I.10	25	100	25	100	25	100	25	100	25	100	25	100	25
I.11	30	N/A		N/A		N/A		N/A		N/A		N/A	
I.13	100	67	67	100	100	100	100	100	100	100	100	100	100
Feature		***GRASS***		***QGIS***		***uDig***		***gvSIG***		***SAGA***		***OpenJUMP***	
	Weight	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score
I.14	100	N/A		N/A		N/A		N/A		N/A		N/A	
I.14.1	100	100	100	100	100	100	100	100	100	100	100	100	100
I.14.2	80	100	80	100	80	100	80	100	80		0	100	80
I.14.3	100	100	100	100	100	100	100	100	100	100	100	100	100
I.14.4	70	N/A		N/A		N/A		N/A		N/A		N/A	0
I.14.5	20	N/A		N/A		N/A		N/A		N/A		N/A	0
I.14.6	30	N/A		N/A		N/A		N/A		N/A		N/A	0
II		*Esri ArcGIS Independence*											
II.1	100	100	100	100	100	67	67	100	100	33	33	100	100
II.2	100	100	100	100	100	100	100	100	100	100	100	100	100
II.3	95	100	95	100	95	100	95	100	95	100	95	100	95

II.4	95	100	95	100	95	67	64	100	95	33	31	100	95
II.5	100	100	100	100	100	100	100	100	100	100	100		0
II.6	100	100	100	100	100	100	100	100	100	100	100	100	100
II.6.1	100	100	100	100	100	100	100	100	100	100	100	100	100
II.6.2	70	100	70	100	70	100	70	100	70	100	70	100	70
II.7	100	100	100	100	100	100	100	100	100	100	100	100	100
II.8	100	N/A		N/A		N/A		N/A		N/A		N/A	
II.9	100	100	100	100	100	100	100	100	100	100	100	100	100
II.10	70	33	23	100	70	67	47		0		0	67	47
III		*Modularization*											
III.1		N/A		N/A		N/A		N/A		N/A		N/A	
III.1.1	90	100	90	100	90	100	90	100	90	100	90	100	90
III.1.2	75	100	75	100	75	100	75	100	75	100	75	100	75
III.1.3	95	67	64	67	64	67	64	67	64	67	64	67	64
III.1.4	60	100	60	100	60	67	40	100	60	100	60	67	40
III.1.5	100	100	100	100	100	33	33	100	100	67	67	67	67
III.1.6	60	100	60	100	60	33	20	67	40	67	40	67	40
Feature		***GRASS***		***QGIS***		***uDig***		***gvSIG***		***SAGA***		***OpenJUMP***	
	Weight	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score
III.1.7	100	100	100	100	100	100	100	100	100	100	100	100	100
III.1.8	100	100	100	100	100	100	100	100	100	100	100	100	100
III.1.9	80	100	80	100	80	33	26	67	54	67	54	67	54
III.1.10	100	100	100	100	100	100	100	100	100	100	100	100	100
III.2	70	100	70	100	70	100	70	100	70	100	70		0
III.3	100	100	100	100	100	100	100	100	100	100	100	100	100
III.4	100	67	67	100	100		0		0		0		0

III.5	80	100	80	100	80		0	67	54	67	54	33	26
IV		*Data Sharing*											
IV.1		N/A		N/A		N/A		N/A		N/A		N/A	
IV.1.1		N/A		N/A		N/A		N/A		N/A		N/A	
IV.1.2		N/A		N/A		N/A		N/A		N/A		N/A	
IV.1.3		N/A		N/A		N/A		N/A		N/A		N/A	
IV.1.4	100	100	100	100	100	100	100	100	100	100	100	100	100
IV.1.5	100	100	100	100	100	100	100	100	100	100	100	100	100
IV.1.6		N/A		N/A		N/A		N/A		N/A		N/A	
IV.1.7		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2.1		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2.2		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2.3		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2.4		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2.5		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2.6	100	100	100	100	100	100	100	100	100	100	100	100	100
IV.2.7	100	100	100	100	100	100	100	100	100	100	100	100	100
IV.2.8		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2.9		N/A		N/A		N/A		N/A		N/A		N/A	
IV.2.10		N/A		N/A		N/A		N/A		N/A		N/A	
Feature		***GRASS***		***QGIS***		***uDig***		***gvSIG***		***SAGA***		***OpenJUMP***	
	Weight	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score
IV.3		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.1		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.2		N/A		N/A		N/A		N/A		N/A		N/A	

IV.3.3		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.4		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.5		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.6		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.7	100	100	100	100	100	100	100	100	100	100	100	100	100
IV.3.8		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.9		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.10	70	100	70	100	70	100	70	100	70	100	70	100	70
IV.3.11		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.12	30	100	30	100	30	100	30	100	30	100	30	100	30
IV.3.13	70	100	70	100	70	100	70	100	70	100	70	100	70
IV.3.14		N/A		N/A		N/A		N/A		N/A		N/A	
IV.3.15	30	100	30	100	30	100	30	100	30	100	30	100	30
IV.3.16	70	100	70	100	70	100	70	100	70	100	70	100	70
IV.4		N/A		N/A		N/A		N/A		N/A		N/A	
IV.4.1		N/A		N/A		N/A		N/A		N/A		N/A	
IV.4.2		N/A		N/A		N/A		N/A		N/A		N/A	
IV.4.3		N/A		N/A		N/A		N/A		N/A		N/A	
IV.4.4		N/A		N/A		N/A		N/A		N/A		N/A	
IV.4.5		N/A		N/A		N/A		N/A		N/A		N/A	
IV.4.6		N/A		N/A		N/A		N/A		N/A		N/A	
IV.4.7	80	100	80	100	80	100	80	100	80	100	80	100	80
IV.4.8	100	100	100	100	100	100	100	100	100	100	100	100	100
V		*Automation*											
V.1		N/A		N/A		N/A		N/A		N/A		N/A	
Feature		***GRASS***		***QGIS***		***uDig***		***gvSIG***		***SAGA***		***OpenJUMP***	
	Weight	Score	Weighted	Score	Weighted	Score	Weighted	Score	Weighted	Score	Weighted	Score	Weighted

			score		score		score		score		score		score
V.1.1	100	100	100	100	100	100	100	100	100	100	100	100	100
V.1.2	100	100	100	100	100	100	100	100	100	100	100	100	100
V.2		N/A		N/A		N/A		N/A		N/A		N/A	
V.2.1	90	100	90	100	90	100	90	100	90	100	90	100	90
V.2.2	90	100	90	100	90	100	90	100	90	100	90	100	90
V.3		N/A		N/A		N/A		N/A		N/A		N/A	
V.3.1	80	100	80	100	80	100	80	100	80	100	80	100	80
V.3.2	85	100	85	100	85	100	85	100	85	100	85	100	85
V.4		N/A		N/A		N/A		N/A		N/A		N/A	
V.4.1	95	100	95	100	95	100	95	100	95	100	95	100	95
V.4.2	100	100	100	100	100	100	100	100	100	100	100	100	100
			75.9		**77.1**		65.6		73.3		67.4		66.5

Feature		***GRASS***		***QGIS***		***uDig***		***gvSIG***		***SAGA***		***OpenJUMP***	
	Weight	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score	Score	Weighted score
1	85	30	26	90	77	90	77	80	68	80	68	80	68
2	65	80	52	90	59	70	46	80	52	70	46	60	39
3	50	100	50	100	50	100	50	90	45	85	43	70	35
4	50	90	45	80	40	40	20	70	35	80	40	50	25
5	45	100	45	100	45	40	18	85	38	80	36	50	23
			43.5		**54.0**		42.0		47.7		46.4		37.9

Appendix F: Bibliography

ANSI/IEEE 1016-1998, "Recommended Practice for Software Design Descriptions" Institute of Electrical and Electronics Engineers, Inc., New York, 1998.

ANSI/IEEE 1016-1998, "Recommended Practice for Software Requirements Documents" Institute of Electrical and Electronics Engineers, Inc., New York, 1998.

ANSI/IEEE 829-1998, "Standards for Software Test Documentation", Institute for Electrical and Electronics Engineers, Inc., New York, 1998.

Arundel, S.T., Archuleta, C.M., Phillips, L.A., Roche, B.L., and Constance, E.W., 2015, *1-meter digital elevation model specification*: U.S. Geological Survey Techniques and Methods, Book 11, chap. B7, 25 p. with appendixes, http://dx.doi.org/10.3133/tm11B7.

Barrington-Leigh C, & A Millard-Ball 2017. The world's user-generated road map is more than 80% complete. *PloS one*, **12**(8), e0180698. doi:10.1371/journal.pone.0180698

Basiri A, Jackson M, Amirian P, Amir Pourabdollah, Sester M, Winstanley A, Moore T & L Zhang 2016. Quality assessment of OpenStreetMap data using trajectory mining, *Geo-spatial Information Science*, **19**:1, 56-68, DOI: 10.1080/10095020.2016.1151213.

Brahmin M, Drira W, Filali F and N Hamdi 2016. Spatial data extension for Cassandra NoSQL database. *Journal of Big Data*, **3**:11, DOI: 10.1186/s40537-016-0045-4.

Breakstone C and T Anderson 2017. Census Data API User Guide, version 1.5. Suitland, MD: The US Census Bureau.

Brzev S, Scawthorn C, Charleson AW, Allen L, Greene M, Jaiswal K and V Silva 2013. *GEM Building Taxonomy Version 2.0*, GEM Technical Report 2013-02 V1.0.0, 188 pp., GEM Foundation, Pavia, Italy, doi: 10.13117/GEM.EXP-MOD.TR2013.02.

Codd, E 1970. A relational model of data for large shared data banks. *Communications of the ACM*, **13**(6): -377-387.

Department of Homeland Security (DHS) Strategic Plan (2014 – 2018)

Federal Insurance and Mitigation Administration (FIMA) Strategic Plan (2012 – 2014)

FEMA Administrator's Intent (2015 – 2019)

FEMA National Advisory Council (NAC) Recommendations (2016)

FEMA Strategic Plan (2014 – 2018)

Lloyd C, Sorichetta A and A Tatem 2017. High resolution global gridded data for use in population studies. *Scientific Data*, **4**: 170001. https://www.nature.com/articles/sdata20171.ris.

Natural Hazard Risk Assessment Program (NHRAP) Strategic Plan

NWIRP Year. Title.

Presidential Policy Directive 8: National Preparedness (PPD-8)

Risk Mapping, Assessments, Planning (Risk MAP) Program, FIMA Multi-Year Plan: Fiscal Years (2010 – 2014).

Sehra S, Singh J and H Rai 2014. A Systematic Study of OpenStreetMap data Quality Assessment. *Proceedings of the Eleventh International Conference on Information Technology*: New Generations. Las Vegas, NV. Berlin (DE): FI-STAR Consortium. DOI 10.1109/ITNG.2014.115.

Technical Mapping Advisory Council (TMAC) Program Vision and Goals (December 2015)

TMAC Annual Report (December 2015)

TMAC Future Conditions Report (December 2015)

UNECE 2017. *UN Recommendations on the Transport of Dangerous Goods. Model Regulations* (Twentieth ed.), New York and Geneva: United Nations, 2017, ISBN 978-92-1-139120-6, ST/SG/AC.10/1/Rev.20.