White Paper

for

OpenHazus

Version x.x

Prepared by <author(s)>

<date created>

Table of Contents

Table of Contents ii

Revision History iii

1. Executive Summary 1

2. Introduction 1

2.1 Purpose 2

2.2 Document Conventions 7

2.3 Intended Audience and Reading Suggestions 7

2.4 Project Scope 7

2.5 References 7

3. Overall Description 8

3.1 Product Perspective 8

3.2 Product Features 11

3.3 User Classes and Characteristics 11

3.4 Operating Environment 13

3.5 Design and Implementation Constraints 13

3.6 User Documentation 13

3.7 Assumptions and Dependencies 14

4. System Features 16

4.1 System Feature 1 – Desktop Independence 16

4.1.1 Description and Priority 16

4.1.2 Stimulus/Response Sequences 17

4.1.3 Functional Requirements 17

4.2 System Feature 2 – Esri ArcGIS Independence 18

4.2.1 Description and Priority 18

4.2.2 Stimulus/Response Sequences 19

4.2.3 Functional Requirements 19

4.3 System Feature 3 – Modularization 21

4.3.1 Description and Priority 21

4.3.2 Stimulus/Response Sequences 22

4.3.3 Functional Requirements 22

4.4 System Feature 4 – Data Sharing 24

4.4.1 Description and Priority 24

4.4.2 Stimulus/Response Sequences 26

4.4.3 Functional Requirements 27

4.5 System Feature 5 – Automation 32

4.5.1 Description and Priority 32

4.5.2 Stimulus/Response Sequences 33

4.5.3 Functional Requirements 33

5. External Interface Requirements 35

5.1 User Interfaces 35

5.2 Hardware Interfaces 35

5.3 Software Interfaces 35

5.4 Communications Interfaces 35

6. Other Nonfunctional Requirements 36

6.1 Performance Requirements 36

6.2 Safety Requirements 36

6.3 Security Requirements 36

6.4 Software Quality Attributes 36

7. Other Requirements 37

8. Analysis & Discussion 38

8.1 Discussion of Web and Open Source Alternatives 38

8.2 Recommended Solution Architecture 38

8.3 Proofs of Concept – Completed and Proposed 38

9. Conclusion 38

Appendix A: Glossary 38

Appendix B: Analysis Models 38

Appendix C: Issues List 38

Appendix D: We can make as many appendices as necessary. 39

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Executive Summary

# Introduction

Jordan: Please incorporate FEMA’s feedback in this section. Comments are included.

Hazus provides stakeholders with a dependable view of the potential natural hazard risks they face and thereby encourages hazard mitigation and community resilience activities. 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. As indicated in Table 1 below, the 2017 Hazus Strategic Plan identifies five goals, three of which are directly addressed by this whitepaper, while the other two represent a necessary background for successful implementation.

The purpose of this whitepaper is to define the functionality, interfaces, performance, attributes, and design constraints of OpenHazus, the proposed future version of FEMA’s Hazus risk analysis software. It is based on a Product Requirements Assessment (PRA) focused on the major capabilities and features needed by Hazus stakeholders. This document will not define additional capabilities but focuses on the provision of the capabilities defined in the PRA. 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 whitepaper includes sections addressing architecture, product research, and prototyping.

|  |  |  |
| --- | --- | --- |
| ***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 the 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 | 🗹 |
|  |

## Purpose

Jordan: Please incorporate FEMA’s feedback in this section. Comments are included.

<Identify the product whose requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this Geospatial System Requirement Specification, particularly if this Geospatial System Requirement Specification describes only part of the system or a single subsystem.>

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 modelling tools that are customizable and fast with fully documented methodologies. While Hazus in its current form is not meeting these demands, an open source, web-based version of Hazus could meet these needs and evolve efficiently as new needs arise. This paper explores possible approaches and outcomes for an open source redesign of Hazus aimed at effectively broadening its scope and better supporting its role as a leading publicly available risk modelling 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 modelling capabilities. The development of Hazus as a centralized desktop software requires an update cycle structured to deliver periodic, costly releases. This approach does not support the efficient integration of innovative hazard and loss modelling methodologies at varying stages of validity 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 or pseudo 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 users and developers that, like in many Free and Open Source Software (FOSS) projects, handle a large share of helpdesk duties and act as a conduit between actual users and developers. The increased transparency and accountability should improve stakeholder confidence significantly.

Demand for quantitative risk information has increased alongside a widening user base, evident in new regulatory requirements and incentives offered by FEMA programs for communities or projects that implement data-driven decision making towards the reduction of risk. 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 that will allow planners, project managers, and policymakers to access summarized risk data produced by past Hazus studies at varying geographic levels nationwide. This online portal will make summarized Hazus results discoverable and available for the appropriate model with updates scheduled at a set frequency, thereby considerably widening the user base of Hazus. Standardized 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. In fact, the functionality of Hazus and other risk modelling technologies do not necessitate a GIS at all. 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 out of XXX lines of code, Hazus only employs XXX lines of geospatial analysis. In fact, the earthquake and hurricane models do not make calls to any ESRI geoprocessing tools. And 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. Significant resources are also dedicated to ArcGIS version updates, including an estimated 1,500 developer hours for the recent ArcGIS 10.6 release.

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, precomputed damage factors.

Similarly, a relatively small percentage of Hazus users require a desktop environment for their risk modelling projects. Users with a legal obligation to maintain secure risk modelling data inputs (actuaries, certain military planners, etc.) represent an estimated 5% of Hazus use cases. Given the ongoing shift in computing technologies away from local environments and onto cloud resources, transitioning to a web-based architecture for Hazus could fulfil 95% of user needs while increasing modelling 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 inventory including User Defined Facilities (UDF), best practices, and results. This requires the development of a quality control process and committee to vet new data in collaboration with FEMA Production and Technical Services (PTS) and Community Engagement and Risk Communication (CERC) Contractors and Cooperating Technical Partners (CTP).

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 via an Application Programmer’s Interface or API. 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.

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 modelling capabilities to be embedded within Hazus. These embedded processes have proven costly to maintain and as well consistently lag behind the latest available hazard modelling techniques.

Another beneficial aspect of an open architecture is the reliance on FOSS, which would make it easier to customize Hazus for many uses. 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 modelling 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.

The next steps for OpenHazus include a phased development built around a modular architecture of Hazus analytical capabilities. Lightweight desktop utilities for site-specific analyses will be made available for user download in the short term while larger scale versions of these tools are built on web servers for availability in the OpenHazus platform. OpenHazus nationwide inventory requirements will be addressed by integrating the latest building footprint data with HIFLD Open infrastructure and essential facility data, state and local data and developing strategies for effective attribute mapping and dynamic updates. Existing Risk MAP CS MIP and MSC capabilities will be leveraged to develop live access and discovery of external hazard data sources directly from the OpenHazus platform. A portal for viewing, downloading and sharing authoritative OpenHazus results from across the US will be deployed in order to shift non-technical users away from the creation of duplicative model data and toward summarized risk information ready for communication with a broad audience.

## Document Conventions

Recommend no action for v0.9

In final draft, recommend adding a short list of term definitions in future version, depending on usage in final draft, eg. what is meant by Legacy Hazus vs. OpenHazus, Hazus without (-MH), etc.

<Describe any standards or conventions that were followed when writing this Geospatial System Requirement Specification, such as fonts or highlighting that have special significance.I can’t think of any but, if you do, here is the place to identify them.>

## Intended Audience and Reading Suggestions

Jordan, please ask Suman/Andrew for input here and put a few sentences. I envision our intended audience as FEMA personnel on Hazus Program, plus RiskMAP leadership both on CDS contract and with FEMA

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, etc.>

## Project Scope

No action needed in v0.9

<Provide a short description of the capability being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals and/or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. A specification that specifies the next release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision.>

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 a hypothetical OpenHazus, and exploring available Open Source and web technologies for feasibility in an OpenHazus architecture.

## References

No action recommended for v0.9

Once document is finalized, recommend adding a list of appendices, eg. “Use cases.”

<List any other documents or Web addresses to which this Geospatial System Requirement Specification refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document.>

# Overall Description

## Product Perspective

Jordan: Please incorporate FEMA’s feedback in this section. Comments are included.

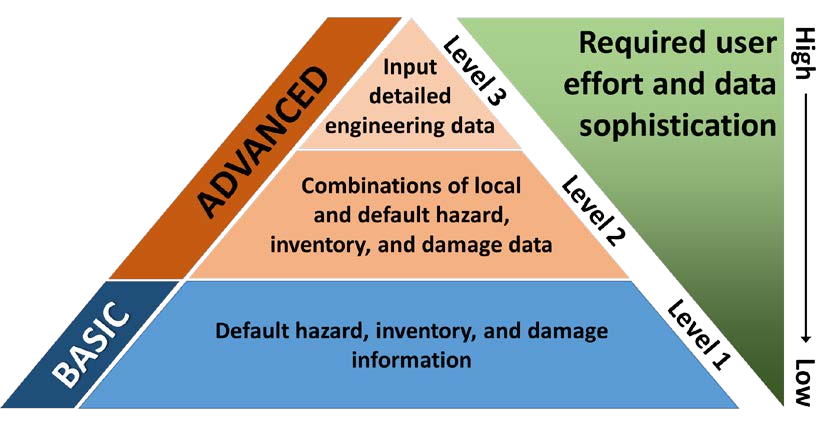
<Describe the context and origin of the product being specified in this Geospatial System Requirement Specification. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the Geospatial System Requirement Specification defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces might be included. This is the section to identify the unique geospatial aspects of the capability >

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 and structural impacts resulting from given hazard information for earthquakes, floods, tsunamis and hurricanes. Hazus results allow members of the emergency management community to develop policies that aim to decrease the risk of future loss due to natural hazards – a capability required by risk reduction policies developed by local, state and federal authorities. 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 a regional scale.

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 the Disaster Mitigation Act of 2000 (DMA 2000) to develop risk assessments leveraged in hazard mitigation plans to provide critical information for decision makers.

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 outlined in Table 2 below. Specific capabilities vary according to user needs, hazard analyzed, and data availability. See Appendix 2 for a detailed breakdown of typical Hazus analytical workflows. 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. An advanced analysis may also incorporate data from previous external risk studies. 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 costs approximately $6,500 per seat, which is 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 (see Section I.1 above), the significant impending cost of rewriting Hazus to maintain compatibility with ArcGIS Pro has precipitated the need to redesign Hazus independent of external GIS viewers, which provides an opportunity to identify key obstacles preventing Hazus from effectively meeting the needs of users and revisit Hazus requirements (outlined in Section I.3).



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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)** | **✓** | **✓** | **✓** |  |

## Product Features

No action needed for v0.9- this is a summary section and better to do after FEMA has given feedback on Section 3.

<Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Details will be provided in Section 3, so only a brief high level summary is needed here. Organize the functions to make them understandable to any reader. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, might be included.>

## User Classes and Characteristics

Andrea will update for v0.9

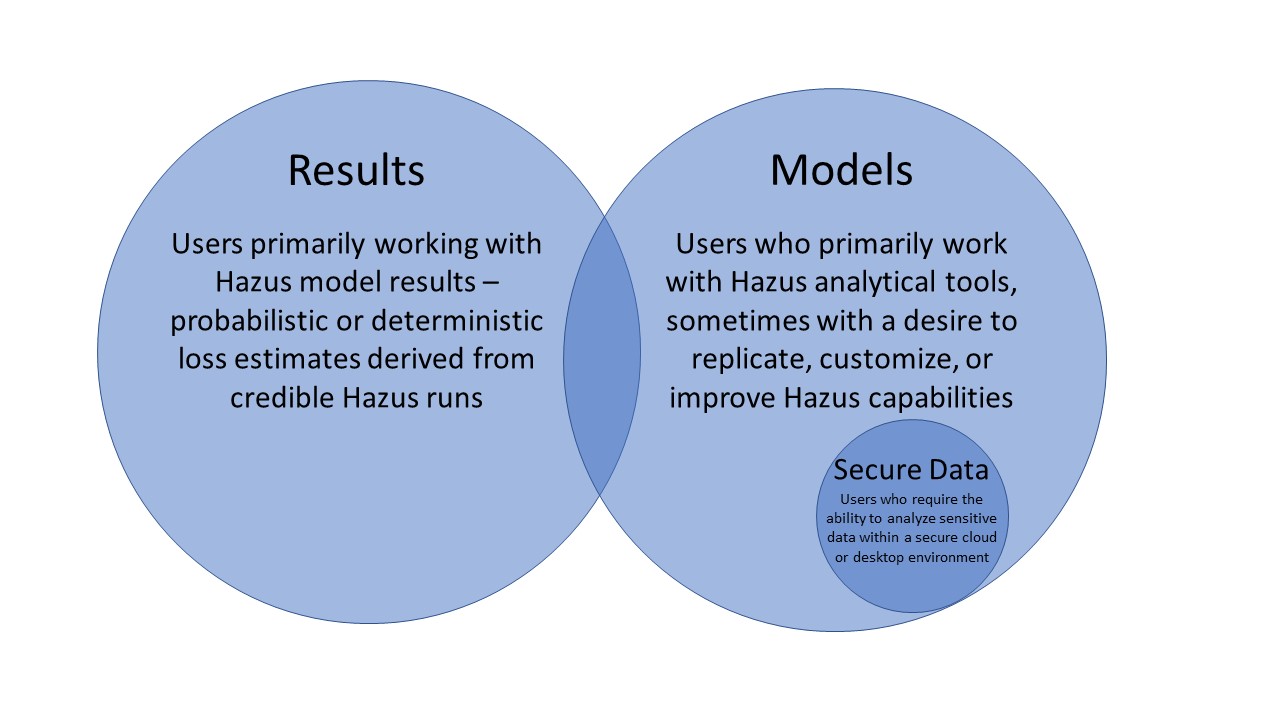
<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the important characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.>

The Hazus user community is large and diverse, with an estimated 10,000 users globally according to a survey completed in 2013. As FEMA’s standardized method for estimating damage and loss due to natural disasters, the intended audience for Hazus is with emergency managers who work in mitigation planning. However, as indicated in Table 1 (Section I.4), 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 Appendix 3 for demographic survey results from Hazus user training courses.

The size and diversity of the Hazus user community makes it difficult to effectively summarize or categorize. Table 2 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 platforms. See Appendix 4 for user profiles which demonstrate how single users can map to multiple categories simultaneously.

OpenHazus provides an opportunity to rethink the classification of Hazus users in light of their overlapping and shifting roles. The more fundamental classification framework in Figure 2 would simplify and improve the efficacy of Hazus development and outreach. Designing OpenHazus to target users needing only authoritative results rather than analytical capabilities as a significant user category will eliminate the “Levels of Analysis” originally prescribed for Hazus users. If baseline Hazus model results are provided in a user-friendly online platform, OpenHazus analytical modules will only be run by users seeking to leverage some combination of customized damage functions, inventory data, or hazard data specific to a scenario or community. An actuary may use default inventory and nationwide probabilistic hazard data, but customize the damage functions available in OpenHazus. A local planner might use default OpenHazus damage functions, but upload custom scenario-based hazard data and updated site-specific inventory information to use as model inputs. Since any user running OpenHazus will be using some kind of customized model input, distinguishing between “levels” of customization would overcomplicate OpenHazus implementation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hazard/Geography** | **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 advanced engineering/damage data |
| Tsunami | NGO/Research | Recovery |  |
|  | Government Contractor |  |  |
|  | Private Sector |  |  |
|  | Academic |  |  |
|  | International |  |  |



User analysis has indicated that users face similar challenges when engaging with the Hazus Program regardless of their user categorization. Requirements (Section III) were developed with the goal of alleviating these challenges. These common challenges are summarized below

## Operating Environment

No action needed for v0.9- this is a summary section and better to do after FEMA has given feedback on Section 4.

<Describe the environment (architecture) in which the system will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must interface and coexist.>

## Design and Implementation Constraints

No action needed for v0.9- this is a summary section and better to do after FEMA has given feedback on Section 4 and Assumptions. Some Assumptions might need to move to this section (eg. Hazus must be free to use)

<Describe any items or issues that will limit the options available to the developers. These might include: business or regulatory policies; hardware limitations; interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements (e.g., GML); communications protocols; security considerations; privacy considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## User Documentation

Andrea will update for v0.9

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

No action needed for v0.9 – but need to think further on this and update/re-organize with constraints before final draft

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in this Geospatial System Requirement Specification. These could include third-party or commercial components and/or capabilities that you plan to use (e.g., Google Maps, etc.), issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

Requirements of OpenHazus development are affected by the following:

* Project funding and schedule
* Changes in project requirements
* Availability of recent, high-resolution demographic and inventory data (e.g. U.S. Census)
* Availability of required features in the Hazus application shell
* Availability of building stock data, damage functions, and loss modelling functions required to perform the loss estimations
* Availability of external sources of high-quality authoritative hazard data
* Operation within a common RDBMS

Key assumptions behind OpenHazus are described below.

* Proposed Hazus software architecture supports the requirements outlined above.
* There is a common interface to items such as the inventory, hazard, analysis and results menus, study region builder, as well as inventory preparation tools and results exporting.
* The Risk MAP CDS Team is responsible for system integration and shall coordinate and support all tasks associated with integrating the hazard models with the overall OpenHazus software.

The following requirements are considered to be desirable but will be pursued after development of the initial versions of OpenHazus:

* Allowing users to define their own custom building types.
* Revising damage functions.
* Optimizing average annualized loss.
* Facilitating ESRI development of Hazus plug-in on newer versions of ArcGIS as well as Windows Operating Systems.
* Optimizing Fire Following Earthquake.
* Data updates when available.
* **1. 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, and building (damages and economic loss) impacts due to natural hazards. This requirement does not include estimating the extent and magnitude of potential hazards. OpenHazus implementation will emphasize the use of external, authoritative hazard datasets.
* **2. Free for Users:** OpenHazus as a methodology toolset should remain free, as should access to core national datasets. Private for-profit entities may be charged a fee for usage that exceeds five percent of the OpenHazus server load..
* **3. Publicly available:** Hazus methodology shall continue to be publicly available in the format of user manuals, technical manuals, and other supporting documentation.

# System Features

Andrea will edit each requirement for consistency in language and level of detail

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

In November 2018, the Hazus Team met in Boulder, Colorado to outline the System Features envisioned for OpenHazus. As a result, the team developed a set of 10 factors from the Boulder meeting:

① Loss estimation tool. This is the core application of OpenHazus that takes precedence over other potential demands in supporting the mission of FEMA 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) and building (damages and economic loss) impacts due to natural hazards. This requirement does not include estimating the extent and magnitude of potential hazards. OpenHazus implementation will emphasize the use of external, authoritative hazard datasets.

② Data/results-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 FGDC and OGC, must be adhered to. A first expansion of these capabilities includes a seamless link between OpenHazus and FEMA’s MIP and MSC to further integrate FEMA’s risk assessment and mitigation programs.

③ Support import, dynamic integration, and discovery of authoritative hazard data. In the spirit of openness, as outlined in section I.2, 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 sandbox, 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.

④ 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. 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.

⑤ 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. 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.

⑥ Secure/private data integration. An estimated ten percent of Hazus stakeholders are working with PCII and PII data that require secure storage, access and communication mechanisms. 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.

⑦ Improve Hazus-provided inventory data. A significant obstacle to a more widespread use of Hazus is the amount of effort it takes to create inventory data. In recognition of this impediment, FEMA has, in cooperation with DHS’s HIFLD support team, 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 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 building-level data, expectations have grown for CDS to provide a centralized national building data set. As specified in requirements ② and ③, 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 ③, a process should be established that allows to merge higher quality inventories with the main repository.

⑧ User must not pay to use Hazus. OpenHazus as a methodology toolset should remain free, as should access to core national datasets. Private for-profit entities may be charged a fee for usage that exceeds five percent of the OpenHazus server load.

⑨ Must be thin client. 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 (web) server. One of the main issues with Hazus is that software updates must be synchronized with updates for the operating system and dependent GIS software, and often these updates conflict with other locally installed software. Many Hazus installations are therefore isolated from the rest of the enterprise workflow, requiring significant hardware and software resources and maintenance and impacting the overall security of installation. A thin client consists of a relatively small application that can be installed on a variety of platforms such as MS Windows, Mac OS, Linux, Chrome, iOS, Android, etc.

⑩ 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.

## System Feature 1 – Desktop Independence

<Don’t actually say “System Feature 1.” State the feature name in just a few words.>

### Description and Priority

Critical to OpenHazus is maintaining the existing functionality while expanding into a 21st century computing environment, which includes seamlessly moving between desktop, mobile, and web-based environments. The inventory data, as well as the current CDS data holdings on flood-related analyses, should now be directly accessible for any visualization, analysis and reporting function of Open Hazus, and have therefore been given the level of a necessity (weight = 100). Not an absolute necessity but highly desirable are non-publicly, 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 B). 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 are on the to-do list for later versions in the near future, which is why these priorities were assigned a middling weight of 50-60.

Finally, 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 Open Hazus cloud repository. These desirables were given a relatively low weight of 15-30.

For the relatively small audience of some ten percent 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.

**Boulder factor addressed: #9 - Must be thin client.; #6 - Secure/private data integration**

**Challenge addressed: Desktop Architecture** Users must download and install Hazus in a desktop environment. Hazus software development and release cycles take place every 4-6 months and accomplish a pre-established set of tasks required to maintain and enhance a large software package.

* 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 days depending on security permissions.
* 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.
* Preservation of user’s loss estimation results and advanced applications of the model is difficult in the desktop environment.

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.

### Stimulus/Response Sequences

Fred, can you recommend if we should drop this section? I’m not sure what to put here.

<Briefly list the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

### Functional Requirements

All – please review each section of requirements and offer feedback on whether these should be placed in an appendix, numbered differently, organized in a different sequence, etc.

<List and briefly discuss the functional requirements associated with this feature. These are the capabilities that must be present in order for the user to execute the use case. Briefly include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

1. Requirements for Desktop Independence
   1. OpenHazus shall operate primarily in a web environment to include the following:
      1. Default inventory data (file format, file size range)
      2. Repository for public, user-defined inventory data (file format, file size range)
      3. Permission-protected storage for private user-defined inventory data (file format, file size range, permission structure)
      4. Access to authoritative hazard input data sources libraries, to include the following:
         1. Flood 🡪 Our own (MIP, MSC) repository (see Req X), OFA’s
         2. Hurricane 🡪 NOAA, Hurrevac, ASCE Windfields (probabilistic only)
         3. Earthquake 🡪 USGS ShakeMap, National Hazard Maps (historic, probabilistic)
         4. Tsunami 🡪 NOAA PMEL, State product libraries
      5. Access to user-defined/generated hazard data where products are not available from authoritative libraries, such as:
         1. Flood 🡪 User-defined
         2. Hurricane 🡪 User-defined
         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)
         4. Tsunami 🡪 User-defined
      6. Access to damage and loss analysis modules, according to hazard type
      7. Ability to integrate input data (default inventory, user-defined inventory, hazard input, or any combination of these) to damage and loss analysis modules
      8. Documentation, for each analysis module, of the required input data type(s)
      9. Repository of vetted source code modules
      10. Repository of results for pre-run, deterministic scenarios
      11. Repository for public, user-defined results
      12. Repository for permission-protected, private, user-defined results
   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. OpenHazus desktop utility shall be a fast running (eg. greater than 10,000 records per second) thin client application based on the OpenHazus site-specific flood POC
      2. OpenHazus desktop utility shall be compatible with the common Windows operating system
      3. OpenHazus desktop utility customizable software script will be available on Github for advanced users to customize, enhance, or update

## System Feature 2 – Esri ArcGIS Independence

### Description and Priority

One of the driving factors for the development of Open Hazus is to gain independence from Esri. It is important, however, to ascertain that this does not come at the price of diminished functionality. Maintaining existing generic GIS functionality and 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 (regardless of scale/granularity), having the continued opportunity to 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 (it turns out that 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 package studied, are not constrained here either; the lesser weight of 70 for these two criteria turned out not to be a serious restriction.

**Boulder factor addressed: # 8 - User must not pay to use Hazus.**

**Challenge addressed: GIS Dependence** The Esri ArcMap interface of Hazus obscures the technically complex processes of hazard risk modeling behind a map display. Users with GIS specializations are expected to undertake risk assessment projects that benefit greatly from 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 skillsets 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 tiny fraction of current Hazus analytical processes are uniquely geospatial. This small fraction includes traditional desktop GIS functions for clipping, mapping, projecting, and developing flood-related rasters. The more extensive 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 OpenHazus can be easily replaced by Open Source geospatial libraries that meet these needs independent from a traditional desktop GIS architecture.

### Stimulus/Response Sequences

Fred, can you recommend if we should drop this section? I’m not sure what to put here.

<Briefly list the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

### Functional Requirements

All – please review each section of requirements and offer feedback on whether these should be placed in an appendix, numbered differently, organized in a different sequence, etc.

<List and briefly discuss the functional requirements associated with this feature. These are the capabilities that must be present in order for the user to execute the use case. Briefly include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

1. Requirements for Esri ArcGIS independence
   1. Where GIS or spatial components are required to maintain existing functionality, no Esri ArcGIS products will be used
   2. OpenHazus will not have any dependency on Esri ArcGIS products
   3. OpenHazus datasets and results will be made available by OpenHazus in a format that is compatible with Open Source GIS for continued analysis outside of Hazus at the user’s discretion
   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
   5. Analysis components that are re-written to meet 2.4 will include site-specific analysis only as a site-specific utility
   6. Analysis components requiring spatial analysis (eg. interpolation of a surface during raster generation) shall be migrated and re-architected to a spatial platform separate from Esri products
   7. Study regions for all hazards shall be created using a spatial or tabular interface
      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. Study region creation shall be global, and only limited by geography where hazard dictates
   8. Study regions and model results for all hazards shall be viewable on an interactive, selectable map interface
   9. Results shall be viewable in an interactive, spatial interface similar or concurrent to the study region interface
   10. Results shall be viewable in tabular format in addition to interactive map format, with options to export both
   11. Results shall be viewable in interactive charts and graphics where users can select results by jurisdictions and input data types
   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

## System Feature 3 – Modularization

<Don’t actually say “System Feature 1.” State the feature name in just a few words.>

### Description and Priority

Hazus has been growing as a family of code developments and has, in the course, become somewhat unwieldy. 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. We envision this code rewrite to occur in phases, i.e., version 1 of Open Hazus will deal with core functionalities and continuation of business, while later versions accommodate interaction with third party software and developers.

Based on these perambulatory notes, 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 as this has to be enforced now even if the benefits will be reaped by the Open Hazus 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, we recommend that calculations of social and economic impacts, 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.

**Boulder factor addressed:** #4 – Modularization

**Challenge addressed: Credibility management. Streamlining development, updates and enhancements.**

Hazus methodologies are developed based on transparent engagement with the academic community in order to ensure state-of-the-art capabilities. However, implementing these methodologies in the Hazus source code software is not transparent or accessible for 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 high quality and lower quality default Hazus results.
* Incorrect, misinterpreted or out-of-context Hazus results can be circulated among the risk management community, which erodes Hazus Program credibility.

Modularization of the existing Hazus architecture and code is preferred from an overall design standpoint for two main reasons: first, Legacy Hazus contains many modules currently that share functionality and code in stored procedures, meaning small code changes in one area can have an unforeseen impact on other, unrelated areas. These dependencies make bug fixes and changes risky, as well as expensive. Similarly, it is difficult for Legacy Hazus developers to quickly push changes out to the user community; releases have traditionally been either service packs or full version upgrades. Recompiling and packaging thousands of lines of code for a single bug fix is inefficient and adds installation burden to the user. It would be preferable to perform updates on small, specialized modules of code, available for download or use in a web environment, making updates instantly available.

Second, the bundled architecture of Legacy Hazus creates functional challenges for users within the software. Being forced to run all or many components of the analysis adds processing time and risk of error or failure in areas where results might not even be needed. Breaking data management and analysis options into smaller modules will allow more options for customization and automation of analyses by the user.

### Stimulus/Response Sequences

Fred, can you recommend if we should drop this section? I’m not sure what to put here.

<Briefly list the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

### Functional Requirements

All – please review each section of requirements and offer feedback on whether these should be placed in an appendix, numbered differently, organized in a different sequence, etc.

<List and briefly discuss the functional requirements associated with this feature. These are the capabilities that must be present in order for the user to execute the use case. Briefly include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

1. Requirements for Modularization
   1. Existing Hazus architecture and source code shall be divided into subsets or modules of existing code, refactored where necessary
   2. Recommended modules include but are not limited to:
      1. Baseline inventory datasets (maintained and updated by FEMA – add file format and size)
      2. User-defined inventory upload/import, classification, storage, and editing for a variety of common tabular and non-proprietary GIS formats
      3. Import of authoritative hazard data for all hazard modules
      4. User-defined hazard upload/import or creation, storage, and editing for a variety of common tabular and GIS formats
      5. Viewing, editing, and selecting for analysis from existing Hazus damage function libraries for a variety of common tabular formats
      6. Upload/import, storage, editing, and selecting for analysis from user-defined damage functions for a variety of common tabular formats
      7. Rapid (eg. greater than 10,000 records per second) calculation of hazard severity values at inventory locations
      8. Rapid (eg. greater than 1,000 records per second) calculation of damage values at inventory locations based on hazard severities and associated damage functions
      9. Calculating social and economic impacts based on hazard severity and building-specific damage functions
      10. Visualizing and interpreting analysis results
   3. Modules shall be catalogued in an online library
   4. Modules shall be re-written as needed in open source languages to allow customization and enhancements by users
   5. The OpenHazus development team will provide a reference implementation for user-defined customization
   6. OpenHazus development activities should address defects and enhancements on a rolling basis across individual modules, without impact to other modules
   7. A uniform process for governance, including vetting and releasing user customizations to modules shall be developed and managed by FEMA
   8. A regular cycle for soliciting module updates from hazard committees and users shall be developed and managed by FEMA
   9. Any and all source code that is also open shall be documented according to industry standard

## System Feature 4 – Data Sharing

<Don’t actually say “System Feature 1.” State the feature name in just a few words.>

### Description and Priority

Parsing the term ‘open’ in OpenHazus, the emphasis is at least as much on the sharing capabilities as it is on laying open any source code. The 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 B list over 40 different requirements. Following the logic of the discussion in previous sections, all requirements that are at the core of existing Hazus functionality are assigned top weights. The higher the dependence on external collaboration, e.g. for legacy third party code for earthquake and tsunami scenarios or even the export to non-standard GIS formats have been assigned relatively light weights to reflect their lower rank.

**Boulder factors addressed:**

#2 – Data/results-sharing platform

#3 – Support import, dynamic integration, and discovery of authoritative hazard data

#6 – Secure/private data integration

#7 – Improve Hazus-provided inventory data

* **Challenge addressed: 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 without SQL or GIS skills 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 basic Level 1 analyses using no hazard or inventory enhancements are run repeatedly by users and not preserved.
* Hazus user community efforts would be best focused on improving the quality of risk assessments provided the baseline results were already readily accessible.
* Hazus user community efforts remain siloed instead of collective, risk assessment efforts are often duplicated, and decision-makers remain largely unaware of Hazus capabilities.

**Challenge addressed: 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 hydrology and hydraulics capability starting in 2003 in order to partially fill this gap. This capability leverages National Elevation Datasets (NED), hydrologic analysis utilizing regional discharge regression equations and a hydraulic model to derive an estimate of flood depths in gridded form for multiple return periods. These methods have become out of date and are in significant need of updating or replacing. Hazus loss estimation based on these internally generated depth grids have been shown to be inaccurate in many areas, which has eroded trust in Hazus as a credible flood risk analysis tool. Since the majority of FEMA Risk MAP programs are centered on quantitative flood assessment, this lack of trust results in a shrinking user base for Hazus flood models. 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 are simple tabular calculations handled easily by lightweight desktop utilities or web services leveraging database tools.

**Challenge addressed: 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 demands both time and resources and is a costly part of performing a risk study, and the work can become out of date quickly in rapidly growing communities. FEMA has developed an incomparable resource of some 16 GB (zipped) 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 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 format and sourced from a variety of freely available national sources. Hazus has yet to leverage the Homeland Infrastructure Foundation Level Data (HIFLD) Open program data. 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 an inefficient and manual process – some of the critical essential facility layers in Hazus were developed in 2001. Hazus specific vulnerability attributes can require special skill sets to develop and are not readily available through data providers. Legacy Hazus datasets do not leverage newer more comprehensive baseline inventory datasets or improved methods of assignment of Hazus vulnerability attributes. Meanwhile, Hazus demographic data updates require waiting several years after the decennial census and implementing a manual update process even though agencies like the Census Bureau provide dynamic data feeds through web APIs, as well as partial annual updates that would ensure constantly up-to-date demographic information.

The Data Sharing requirements are broken into four categories according to the architectural need within OpenHazus for organizing data by type and usage. The first category - Deterministic Results Repository – is separate from the remaining categories in that users will not be able to customize the analysis that produces these results. Any analysis run in Legacy Hazus using baseline inventory and built-in deterministic hazard events or scenarios will be pre-run for OpenHazus and the results made available for viewing and download through the OpenHazus web environment (Baseline Results Repository). No customization of the input data, hazard scenarios, or analysis types will be available. However, the user interface will be interactive, enabling the user to select results by scenario, jurisdictions and types.

The remaining three categories – Baseline Hazus Inventory, Authoritative Hazard Data, and User-Defined Data, will be structured differently to allow user interaction and customization.

### Stimulus/Response Sequences

Fred, can you recommend if we should drop this section? I’m not sure what to put here.

<Briefly list the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

### Functional Requirements

All – please review each section of requirements and offer feedback on whether these should be placed in an appendix, numbered differently, organized in a different sequence, etc.

<List and briefly discuss the functional requirements associated with this feature. These are the capabilities that must be present in order for the user to execute the use case. Briefly include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

1. **Requirements for Data Sharing**
   1. **Baseline Results Repository**
      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.
      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.
      3. Users interested in results described in 4.1.1 and 4.1.2 should be redirected to the results repository
      4. Results for all pre-run scenarios shall be viewable in an interactive spatial interface
      5. Results for all pre-run scenarios shall be queryable and downloadable from the interactive spatial interface, in either spatial or tabular format
      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.
      7. User customized reporting for deterministic results data shall be supported through the use of downloadable results
   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.

* + 1. Baseline inventory data shall be provided by FEMA in a public web interface for viewing and download
    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
    3. Baseline inventory data for OpenHazus should discontinue use of the Legacy Hazus aggregated GBS and substitute nationwide site-specific building stock
    4. Baseline Essential Facility inventory for OpenHazus shall be updated with data available through HIFLD Open
    5. Baseline, 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.).
    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
    7. State data will use an open source database file format, and be subdivided for download by jurisdiction and inventory type
    8. State data will not exceed 10 GB in uncompressed and 2 GB in compressed size to continue to support download and transportability
    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
    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 reupload for user-defined analysis (see section 4.4). These tables shall be small and efficient tabular formats (eg. .csv) easily read by a variety of software.
    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
  1. **Access to Authoritative Hazard Data** 
     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.
     2. Users should be able to upload their own hazard data for integration in their analyses
     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. FEMA should develop a governance process that includes approvals and internal quality control.
     5. **Flood** 
        1. Flood hazard data shall be sourced from users and external data sources (FEMA libraries, NOAA, etc)
        2. Legacy Hazus automated H&H capabilities for generating depth grids will not be migrated to OpenHazus
        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. Depth grids in the repository shall be available in open source formats
        5. Users in the desktop environment should be able to extract from the online reposistory individual flood depths based on lat-lon coordinates or individual structures.
        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
        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)
     6. **Hurricane**
        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
        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
        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. Hurricane data in the repository shall be in open source formats
        5. Typical file sizes for download and upload will continue to be less than 10 MB consistent with Legacy Hazus hurricane hazard data.
     7. **Earthquake**
        1. Earthquake ground motion hazard data – probabilistic, historic, scenario and actual – shall be sourced from the USGS Earthquake Hazard Mapping and ShakeMap programs
        2. Legacy Hazus capability for generating arbitrary, fault-based, and historic epicenter-based earthquake scenarios will not be migrated to OpenHazus
        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. Data in the repository shall be available for download and enhancement by the user in open source file formats (eg. .shp, .xml).
        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 (eg. .shp and .xml)
        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
        7. Typical file sizes for download and upload will continue to be less than 100 MB consistent with Legacy Hazus earthquake hazard data.
     8. **Tsunami**
        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
        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
        3. OpenHazus shall support the capability for user-supplied upload of tsunami hazard data in common open source raster file formats
        4. Integration of supplemental hazard data for evacuation and casualty modelling, 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.
        5. Typical file sizes for download and upload will continue to be less than 100 MB consistent with Legacy Hazus tsunami hazard data.
  2. **User-Defined Data (Inventory, Hazard, Results) Repository**
     1. OpenHazus will provide a repository for user-defined inventory, hazard, and results data, in a web environment
     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
     3. Security permissions and access will be regulated to allow levels of access according to user needs, eg. if a user wishes to make their data public, they can do so.
     4. Public datasets shall be made searchable by location, hazard type, and creator
     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
     6. OpenHazus will provide a recommended framework for documenting user-defined metadata, according to FGDC and OGC standards
     7. OpenHazus will provide lightweight desktop analytical capabilities designed specifically for users who need to model hazard impacts using Personally Identifiable, proprietary, or protected information.
     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.
     9. Authoritative Federal, State and local users and FEMA partners should be especially encouraged to share datasets publicly
     10. Blank default databases will be provided as templates for international Hazus users

## System Feature 5 – Automation

<Don’t actually say “System Feature 1.” State the feature name in just a few words.>

### Description and Priority

Similar to 4.3 modularization, automation is an important modernization aspect of moving to OpenHazus. If OpenHazus is to expand its user base, it needs to provide (geo-) processing templates that novice users can run as defaults, while allowing expert users to modify these to their own needs. At the same time, with increased availability of high-resolution inventory and hazard data and the endeavor to provide ready-made annualized analyses, our vision of OpenHazus mandates near constant runs of scenarios, risk assessments, and automated procedures in response to 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 (weights of 80 and 85 respectively).

**Boulder factor addressed:** #10 – Automation

**Challenge addressed: Labor intensive analysis process. Narrow dissemination of results. Outdated results.**

While advanced analysis is difficult to automate, analyses using online hazard data libraries and baseline inventories described in Section 4.1 can be readily automated. A primary reason why Hazus results are not broadly available is the requirement that users of the program require training and apply several days of labor to develop a complete end to end analysis.

Automation will:

* Provide a streamlined way to update analyses based on improvements and additions to hazard data and/or baseline inventory updates, so results are not frequently outdated.
* Reduce the resources required to update and run nationwide analyses when new USGS earthquake hazard maps are updating for building code applications, ensuring Hazus results reflect the latest authoritative hazard data.
* Support full suite probabilistic flood analyses and provide updates when preliminary and final Risk MAP flood data are produced to support both the Risk MAP product outreach and adoption.
* Reduce the demand/need for expensive and repetitive level 1 updates. Mitigation strategies can result in numerous scenario types, each requiring evaluation against a baseline or “unmitigated” scenario. Evaluation of multiple mitigation strategies, such as “what-if” analyses based on adoption of freeboard, enhanced building code or NFIP regulations, as well as retrofit strategies for essential facilities would be made more efficient through automation.

Although primarily developed to support risk analysis, Hazus is increasingly leveraged to provide situational awareness for real world events. This has largely been beneficial to Hazus since it provides additional opportunities to validate and calibrate the model. However, since Hazus is open and available to users of all skill levels, results are disseminated during events from users with less experience and without coordination with authoritative agencies for critical near real time input of earthquake, flood, wind, surge or tsunami hazard data. Response events require frequent updates based on new forecast advisories, additional ground motion parameters, and/or field observations collected under FEMA Mission Assignments or other efforts. While manual QA/QC processes, as well as interagency coordination is required of the Hazus program during response events, OpenHazus would provide the authoritative run of record and results within its sharing platform for a broad variety of response event users.

### Stimulus/Response Sequences

Fred, can you recommend if we should drop this section? I’m not sure what to put here.

<Briefly list the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

### Functional Requirements

All – please review each section of requirements and offer feedback on whether these should be placed in an appendix, numbered differently, organized in a different sequence, etc.

<List and briefly discuss the functional requirements associated with this feature. These are the capabilities that must be present in order for the user to execute the use case. Briefly include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

1. **Requirements for Automation**
   1. **Automated Updates to Baseline Results Repository**
      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
      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.
   2. **Automated Probabilistic Flood Risk Assessment**
      1. Automation should be established to support multi-frequency probabilistic or uncertainty analyses for flood, supporting multiple depth grids for each frequency
      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.
   3. **Automated Mitigation Strategy Evaluations**
      1. Automated analysis should be established to support multiple mitigation strategy scenarios, including freeboard, NFIP regulations and building code adoption and enforcement
      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.
   4. **Automated Analysis and Results Dissemination for Response Events**
      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.
      2. Automated results export and dissemination should be provided and hosted using OpenHazus providing an authoritative source for the run of record.

# External Interface Requirements

Fred, I think we covered each of these sections in Section 3. Should they be broken out, or left as they are and remove this section?

## User Interfaces

<Describe the logical characteristics of each interface between the system and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components (e.g., Google Latitude). Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a Mashup), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as HTTP. Specify any communication security, data transfer rates, and synchronization mechanisms.>

# Other Nonfunctional Requirements

Fred, I think we covered each of these sections in Section 3. Should they be broken out, or left as they are and remove Sections 5.1, 5.2, 5.3? Possibly combine 5.4 with Section 6 into a “non-technical” and “unresolved” requirements section?

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

In addition to the system features discussed in Chapter 4, 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. Still with a nod towards end users, but also towards future co-developers of Open Hazus modules, the perceived ease of use of each software package analyzed in Chapter 7 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. Rather than assigning this aspect a higher weight (we settled for 50), we have been strict in the actual score that each software package received in the evaluation.

Much of the analysis of how difficult it is to port existing Hazus analyses to an open source environment and the degree to which existing Hazus features are recognizably matched in Open Hazus, has been addressed in 4.2, Esri independence. To avoid double-counting the weights in Chapter 4 and Chapter 6, we assigned these porting characteristics 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.

## Other Requirements

The discussion of system features and software quality attributes revealed a few other requirements that need further discussion with the client before we can address them in our software evaluation. We are following the general sequence of criteria listed in the system features sections of Chapter 4.

### 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) one 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.

It is obviously of utmost 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. This is not a problem per se (and probably even a good thing) but goes beyond the current scope of this white paper.

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.

### Modularization

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 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, a board of directors, a user group, voting members, and financial auditors for the non-FEMA part of Open Hazus. The range of such efforts spans histories like the directed move of GRASS GIS from USACE to the [OSGeo Foundation](https://www.osgeo.org/about/) or the self-organized creation of the [QGIS Association](https://www.qgis.org/en/site/getinvolved/governance/index.html). 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

A number of criteria in the data sharing section of Chapter 4 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](https://earthengine.google.com/) and it would be yet another policy decision to determine whether a contractual agreement with Google would violate the spirit of openness in Open Hazus.

Another technically inspired but in the end more political decision is the notion of data formats supported by Open Hazus. Modern GIS standards such as [GeoPackage](https://www.geopackage.org/), [GeoJSON](http://geojson.org/), and [Cloud-Optimized GeoTIFF](https://www.cogeo.org) are easy to accommodate with the GDAL library. On the technical side, the same holds true for [NetCDF](https://www.unidata.ucar.edu/software/netcdf/) and NASA’s [Hierarchical Data Format](https://eosweb.larc.nasa.gov/HBDOCS/hdf.html). , which come in so many different profiles that the Hazus-specific implementation would have to be determined by a steering committee as indicated in 6.5.2.

Such a committee (or more specific ad hoc committees) would also have to determine the QA/QC procedures for sandboxing 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.

Three other aspects that the authors of this whitepaper have no particular position on but that deserve consideration by its recipients at FEMA are raised in the remainder of this section. The first one is how much server space to allocate to each user – and related to that, may a user have multiple accounts for different projects and how to deal with agency rather than individual accounts. A second is prompted by high resolution datasets (one foot or less nationwide). At such a scale, the boundaries between the conceptual models for raster versus vector data are starting to blur. It is now conceivable to store and process everything in raster format, which then opens the door to software packages like Rasdaman, which is widely used in the European-wide CORINE project. From a coding perspective, this would simplify a lot of rewriting of existing Hazus functionality. However, it would require a change of mind(set)s, who are used to think of building stocks in form of features rather than pixels. Finally, if this idea is too radical, then in light of the very significant resource requirements of a statewide hazards or inventory database for, say California, it might be worthwhile to consider moving away from the safe solution of PostGIS to Hadoop-based OLAP datastores. This idea will be picked up in the following chapter.

# Analysis & Discussion

## Discussion of Web and Open Source Alternatives

The previous chapters outlined characteristics of Open Hazus and alluded to ways how these may be implemented using existing open source software packages. Open Hazus is conceived as being flexible enough to not necessitate an underlying GIS. This does not mean that it is void of GIS functionality (spatial indexing, spatial measurements, topological operations, spatial interpolations, etc.). A GIS as we understand it is a complex family of software modules to ingest, organize, manipulate, analyze, and portray spatially referenced data. Open Hazus will have to do all of this – and more. But rather than having to tie Open Hazus to a particular GIS (or any GIS), we propose to take the best of a number of different open source programs to create hazard management system that is better than any solution that is based on a particular single piece product.

As one of the goals of OpenHazus is to lighten the resource requirements for the majority of its users, it is essential that the bulk of the computing load occurs on web servers. While several commercial and open source GIS now support server-side processing, they have typically not been developed with this server side as their focus. Similarly, GIS databases are weak in comparison with their general purpose brethren, and certainly do not support OLAP transactions or streaming (no-SQL) data. As Open Hazus is trying to reach non-GIS audiences, it would be a mistake to force a GIS-based graphical user interface on users, who have a database or a statistics background. In other words, a proper and thorough evaluation of open source software packages requires to look beyond the realm of, say, [OSGeo projects](https://www.osgeo.org/about/).

The authors of this whitepaper 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!) |

< *Insert diagram describing the relationship between the seven components of the software review* >

In the desktop realm, we investigated PostGIS, SpatiaLite and rasdaman (see Appendix B). For Web GIS, we researched the capabilities of four web mapping server platforms (MapServer, MapGuide, GeoMOOSE, and Mapbender), two true Web GIS (GeoServer and Carto) and one web geprocessing platform (PyWPS). Among the plethora of JavaScript libraries, we concentrated on those that have either been 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. Our analysis of 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. We nevertheless recommend the two others for specialized use cases: SpatiaLite for field work and rasdaman for storing massive multi-scale coverages (rasters and images). Based on the system requirements outlined in Chapter 4, Open Hazus 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 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. We included GeoWave from our geospatial library domain here as well, revealing how difficult it is to drawn sharp boundaries between the categories.

We will be recommending the winners in each of the eight categories in the following section. 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. Section 7.2 will discuss the pros and cons of each recommendation and suggest priorities for different phases of the Open Hazus development.

## Recommended Solution Architecture

Our evaluation consisted of multiple processing steps. The underlying data is documented in appendices B-D. In a first step, we divided our criteria into those discussed as system features (requirements) in Chapter 4 and the software quality attributes discussed in section 6.4. The total of the over 100 requirements of the first group was given twice the weight as the total of the five criteria in the second group. A look at the tables in Appendix C illustrates the process.

## Desktop GIS

### From a system features perspective, QGIS, GRASS, and even gvSIG are very similarly ranked. All three are full-fledged GISs 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 trips up even savvy GIS users) and gvSIG scored just a little lower than QGIS on foreseeable longevity and the matching of existing Hazus features. We are comfortable with our recommendation to use QGIS as the reference desktop implementation of Open Hazus but foresee that several of the other desktop GIS packages might very well be picked up by a future Open Hazus developer community.

## Web GIS

This category of software packages contains three distinct groups of software solutions that cannot really be compared to each other. MapServer, MapGuide, GeoMOOSE and Mapbinder are web mapping servers and are hence not trying to provide the analytical functionality that forms part of the system feature requirements outlined in Chapter 4. Similarly, PyWPS is a specialized software that Open Hazus developers might very well 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 the new kid on the block; 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 (to be discussed in section 7.2.5) representing the exact opposite (high development effort but maximum flexibility and “openness”.

## 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 the not very robust weights) 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 keep an eye out for 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.

## Geospatial Libraries

As in the Web GIS category, the entrants in this group are a very diverse bunch with typically highly specialized functionality. As elaborated on in section 7.1, these libraries are not mutually exclusive. GDAL/OGR got a perfect score in what it was written for 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 version 1 of Open Hazus but with its pole position in both geospatial libraries and no-SQL databases, it cannot escape the attention of anyone who endeavors to support the massive data requirements expounded repeatedly in Chapter 4.

## 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 to extreme opposite end of the development effort spectrum alluded to in 7.2.2. 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, version 1 would most likely take as many years to develop as it would take month 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](https://cloud.r-project.org/web/packages/stars/index.html) 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.

## Geospatial Databases

PostGIS is the clear winner in this category. It will definitely be the basis of version 1 of Open Hazus. 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 discussed in section 7.1, it could be a game changer if Hazus analyses are re-conceptualized in terms of high resolution rasters.

## 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 version 1, will have to be kept in mind for future versions of Open Hazus. 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 reginal governments and in the banking industry but OpenCube fails in the software quality attributes category. It would probably take an NSF 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 whitepaper recommend reaching out to Microsoft to determine their level of interest in a joint development for this part of Open Hazus.

## 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 of the system features side but ranks lowest on the software quality attributes side. In the end, we cannot recommend any of these and fall back onto GeoWave, which is the winner in both categories and was already chosen in the group of geospatial libraries.

## Recommended Software Architecture Conclusions

We propose a phased development of Open Hazus and anticipate the choice of different software packages for each of these phases. 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.

## Proofs of Concept – Completed and Proposed

Doug, please fill this in with your write-ups.

<Discuss the POCs already completed and how these informed the Recommendations for Web/Open Source Alternatives, and Solution Architecture. Describe any proposed POCs to be completed prior to design phase>

Based on the above requirements and analysis, a number of Proof of Concept (POC’s) 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.

**Open Source Utility for Hazus Data Export and Reporting**

Hazus inputs and results are stored in backend SQL 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 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 tables, summarize those data points across a variety of categories, and join them to a shapefile of study region tracts converted from SQL 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 for GIS-independent results reporting in a web-based, Open Source version of Hazus.

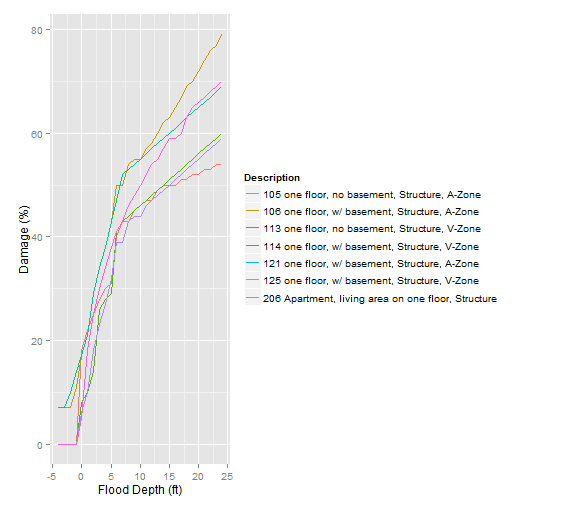
**Nationwide Essential Facility Inventory Updates using HIFLD Open**

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.

**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 includes not only the building and content 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.



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

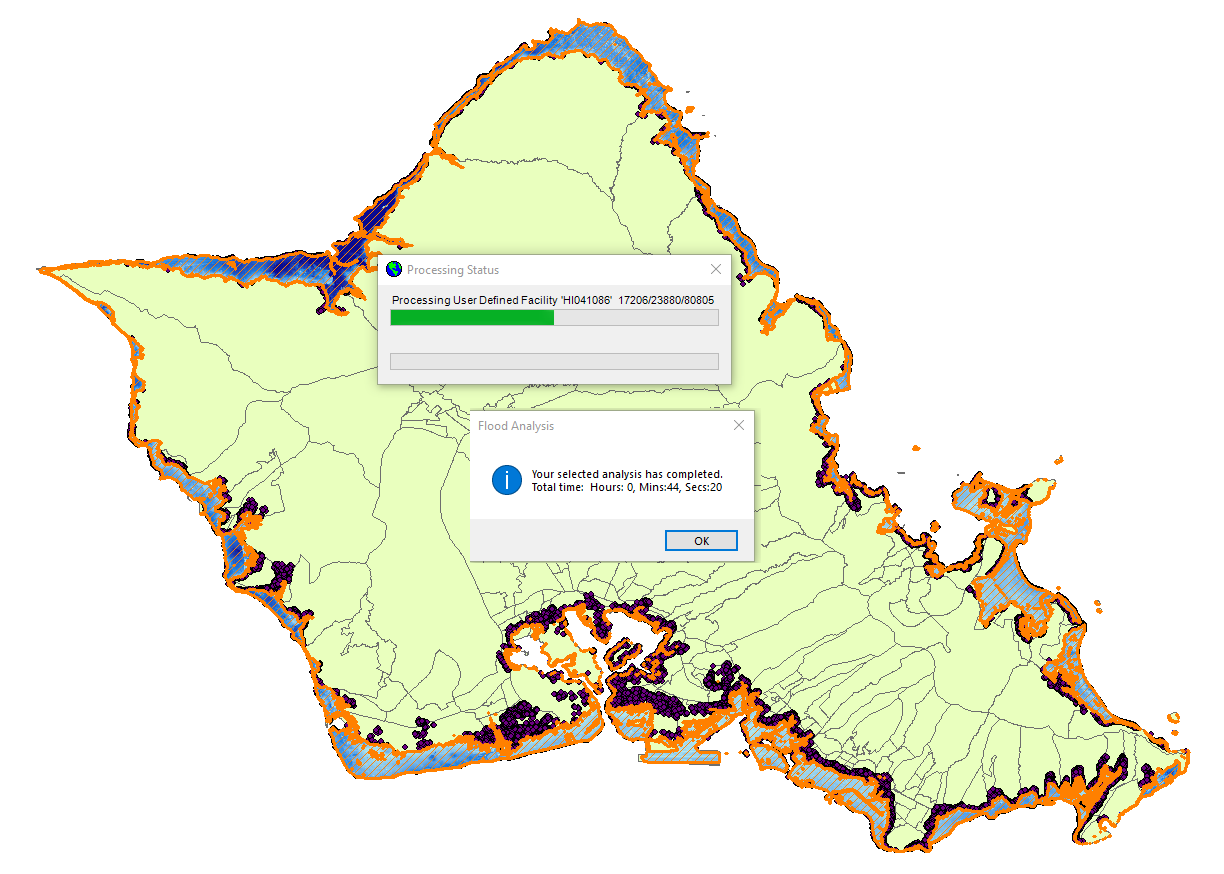
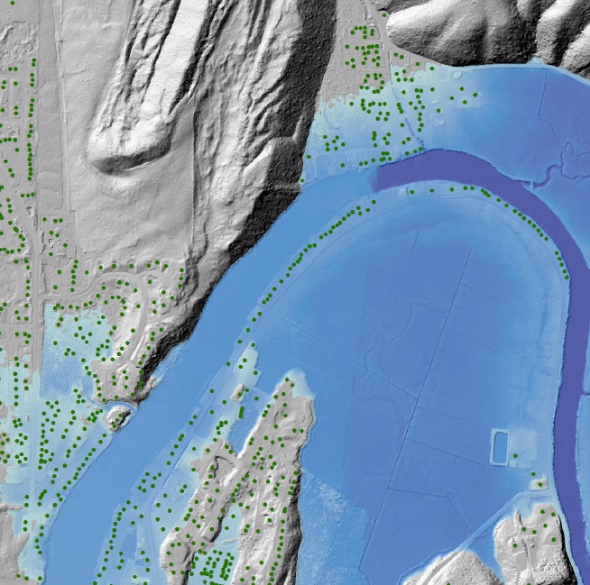
3. Provide both a light weight desktop utility and web-based tool

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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Software** | **Runtime** | **Desktop Utility Size\*** | **Input Format** | **Output Format** |
| Hazus 4.2 | Hazus and ArcGIS with Spatial Analyst | 44 min 20 sec | 12.4GB Hazus; 2.28GB ArcGIS with Spatial Analyst | .mdb or .xls (using CDMS) | SQL tables |
| OpenHazus0 | ArcGIS (DOGAMI) | 84 sec | 2.28GB | .gdb, .mdb, .shp | .gdb |
| OpenHazus1 | QGIS and .shp | 300 sec | 1.95GB | .mdb, .shp | .shp |
| OpenHazus2 | QGIS and .csv | 40 sec | 1.95GB | .mdb, .shp | .shp, .csv |
| OpenHazus3 | GDAL and .shp | 13 sec | 56.5MB GDAL plugins; 206MB Python libraries | .shp | .csv |
| OpenHazus4 | GDAL and .csv | 4 sec | 56.5MB GDAL plugins; 206MB Python libraries | .csv | .csv |
| OpenHazus5 | 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 Aluetain Tsunami. | | | | | |

**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. Income losses, debris estimates, and functionality losses are not provided.



***Figure 1*** *shows screenshots from a typical Hazus UDF flood analysis. Damages are driven primarily by the locational accuracy of site-specific inventory in relation to the flood hazard layer. Processing for analysis on the right took about 45 minutes to complete.*

**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 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. These include lookup tables that provide building, content and inventory losses for all 3 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.

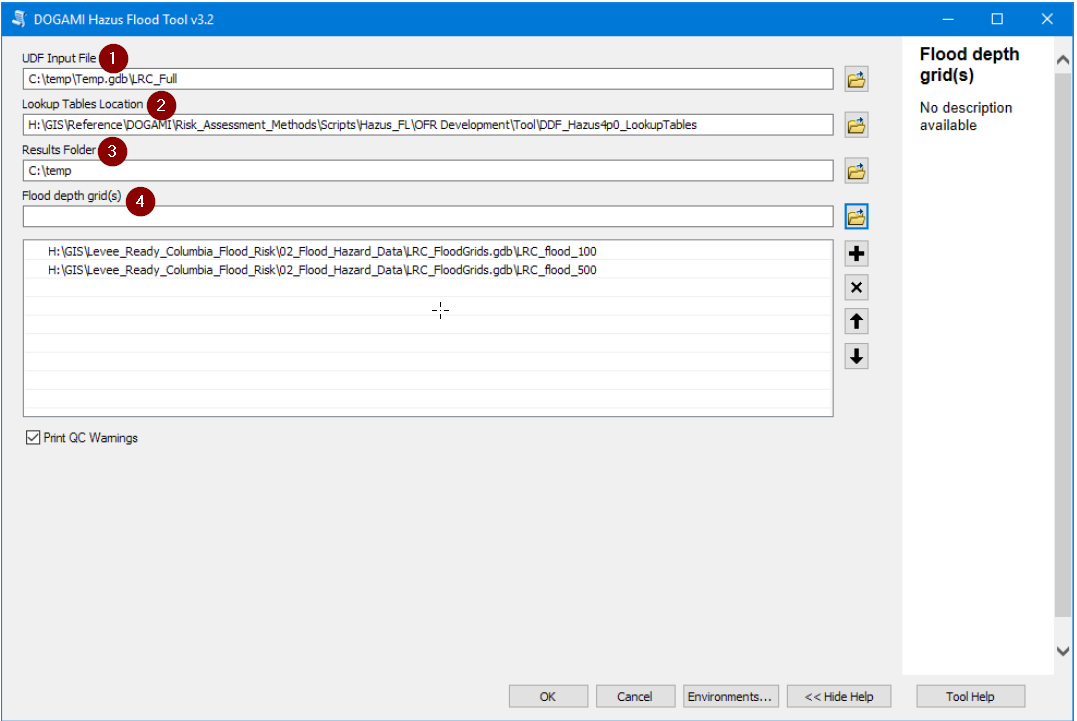
The DOGAMI flood tool requires 4 basic steps:

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



**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.

**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.

**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.

**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.

**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.

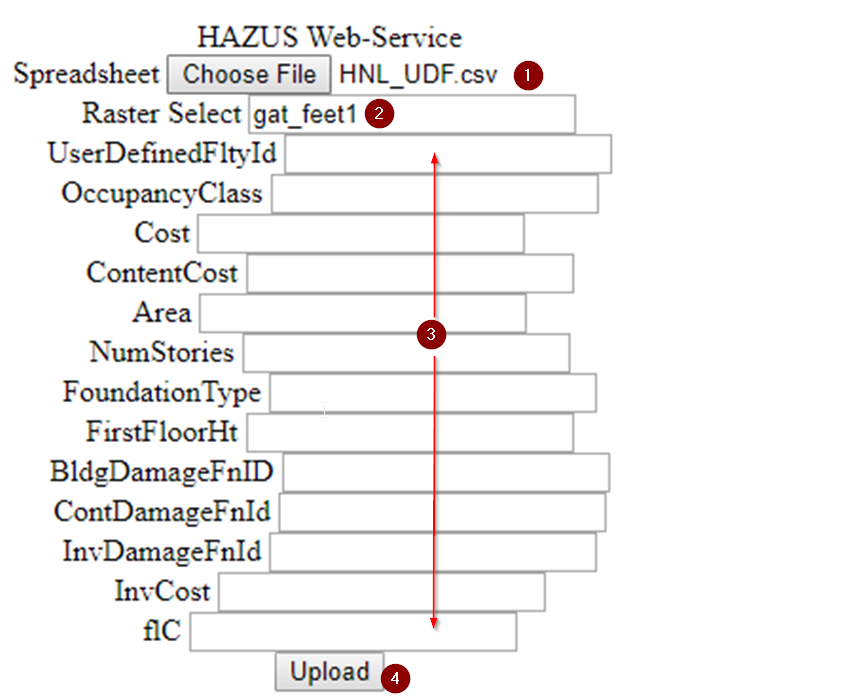
There are 4 steps required for the online version:

1. Browse to input file

2. Select depth grid

3. Use default fieldnames or enter fieldnames if different

4. Select upload



**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:

1. ***Occupancy Type***: based on Hazus 33 model building occupancy type categories
2. ***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.
3. ***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 3 attributes are 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:

1. ***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*).
2. ***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*).
3. ***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.*)

**Microsoft Footprints as OpenHazus Foundational Inventory**

**Importing from the MIP, MSC feasibility**

**Inventory coverage of flood products**

# Conclusion

Appendix A: Glossary

<Define all the terms necessary to properly interpret the Geospatial System Requirement Specification, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each Geospatial System Requirement Specification.>

Appendix B: Open Source Product Review

The purpose of this whitepaper is to define the functionality, external interfaces, performance, attributes, and design constraints of Open Hazus. 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.

## B.1 Description of Open Source Desktop GIS

There are dozens of open source geospatial desktop applications and it is beyond the scope of this whitepaper 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 Open Hazus. Specialty applications, developed for niche application areas (the [Whitebox geospatial analysis tool](http://www.uoguelph.ca/~hydrogeo/Whitebox/) 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](https://grass.osgeo.org/), commonly referred to as GRASS GIS, is used for data management, image processing, graphics production, spatial modelling, 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 B.5.1), 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 modelling, Landscape structure analysis, Solution transport, Watershed analysis.

### Quantum GIS

[Quantum GIS](https://qgis.org/en/site/), 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](http://udig.refractions.net/) 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](http://udig.refractions.net/gallery/) built with uDig illustrates implementations in hydrology, logistics, forest management, and the [Distant Early Warning System for Tsunamis](http://udig.refractions.net/gallery/dews/). uDig is also the underlying platform for DIVA GIS.

[DIVA-GIS](http://www.diva-gis.org/) 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](https://biodiversityinformatics.amnh.org/open_source/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](http://udig.refractions.net/developers/) to write their own GIS (like, for example, [JGRASS](http://udig.refractions.net/gallery/jgrass/)) based on the Eclipse platform.

### System for Automated Geoscientific Analysis (SAGA)

[SAGA](https://sourceforge.net/projects/saga-gis/) 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 to run 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 whitepaper, including non-OSGeo projects like Carto.

### MapServer

[MapServer](http://mapserver.org/) 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 plugin 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](https://www.deegree.org/) 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](http://www.geomoose.org/) 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 organising 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](https://mapbender.org/) 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](https://geoserver.org/) 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 B.4.1), an open source GIS Java toolkit. Together with PostGIS (see B.6.1) and OpenLayers, and with significant support by the company [Boundless](https://boundlessgeo.com/), 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 B.2.8). 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](https://boundlessgeo.com/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](https://github.com/CartoDB). 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](https://www.vizzuality.com/), makes its living by hosting customized Carto GIS server instances and developing specialized front and backends. 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 a Web-GIS to a big geospatial data visualization platform.

### PyWPS

[PyWPS](http://pywps.org/) 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](http://openlayers.org/) 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](https://d3js.org/), 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 domains.

### turf

[Turf](http://turfjs.org/) 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 plugin but a standalone JavaScript library that works with any web mapping package.

### Cesium

[CesiumJS](https://cesiumjs.org/index.html) 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](https://www.unimelb.edu.au/cdmps/research/research-projects/iddss)) is a fine example for that.

### kepler.gl

[Kepler.gl](http://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 whitepaper.

### GeoTools

A classic example for an open source geospatial library is [GeoTools](http://geotools.org/). 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](https://gdal.org/). 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](https://trac.osgeo.org/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](https://geotrellis.io/) 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](http://locationtech.github.io/geowave/index.html) 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 rster 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 plugin 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 plugin 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 B.6.1).

### Spatio Temporal Asset Catalog

The [SpatioTemporal Asset Catalog (STAC)](https://github.com/radiantearth/stac-spec) 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 whitepaper 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](https://www.r-project.org/) is a language and environment for statistical computing and graphics. Readers of this Whitepaper 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](https://cran.r-project.org/web/views/Spatial.html), [operations research](https://cran.r-project.org/web/views/Optimization.html) and [machine learning](https://cran.r-project.org/web/views/MachineLearning.html) tools. Packages like [rattle()](https://cran.r-project.org/web/packages/rattle/index.html) 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](https://cloud.r-project.org/). There are about 13,640 packages available on the Comprehensive R Archive Network ([CRAN](https://www.r-project.org/)) 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](https://www.eclipse.org/) 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, diagraming, 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](https://jupyter.org/) 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[[1]](#footnote-1))

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 (seeB.3.3). The [Urban Data Science Toolkit](http://udst.squarespace.com/) 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 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](https://postgis.net/) 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](https://www.osgeo.org/projects/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](http://www.gaia-gis.it/gaia-sins/) is a [SQLite](http://www.sqlite.org/) 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](http://www.rasdaman.org/), 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](http://www.earthserver.eu/), 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](https://kylin.apache.org/) 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](https://engineering.linkedin.com/teams/data/projects/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](http://spatialhadoop.cs.umn.edu/) 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)](https://github.com/apache/couchdb) 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](https://github.com/orientechnologies/orientdb) is a schema-free, graph-based database, written in Java, with a SQL-like query language but no ability to performs 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](https://orientdb.com/database/spatial-module-orientdb-2-2/) 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 im- 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](https://orientdb.com/database/traffic-management-white-paper/).

### redis

The [Remote Dictionary Server (redis)](https://redis.io/) 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](https://github.com/EverythingMe/geodis) 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](https://ravendb.net/), developed by a company that goes by the name of Hibernating Rhinos, is available on [GitHub in source code](https://github.com/ravendb) 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 datatypes (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](https://www.geomesa.org/documentation/tutorials/geomesa-quickstart-cassandra.html) (see also B.8.6).

### GeoWave

[GeoWave is a OSGeo library](https://www.osgeo.org/projects/geowave/) 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 plugin 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 C: Analysis Models

## C 1 Lookup Table for the Numbering of System Features in Tables C2 – C8

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

## C 2 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 | | Score | Weighted score | Score | | Weighted score | Score | Weighted score | Score | | Weighted score | | Score | Weighted 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 D: 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](http://adcirc.org/) 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](https://www.fema.gov/media-library-data/1521832299221-9e218ec1310c357befe493e534482673/CNMS_Technical_Reference_Feb_2018.pdf)

DEM Digital Elevation Model

DHS (United States) [Department of Homeland Security](https://www.dhs.gov/)

EF Essential Facilities

FEMA [Federal Emergency Management Agency](https://www.fema.gov/), a unit of DHS

FGDC [Federal Geographic Data Committee](https://www.fgdc.gov/)

FIMA [Federal Insurance and Mitigation Administration](https://www.fema.gov/what-mitigation/federal-insurance-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](http://storm.aoml.noaa.gov/hwind/)

H&H Hydrologic and Hydraulic (modelling studies)

HEC [Hydrologic Engineering Center](http://www.hec.usace.army.mil/) (of the U.S. Army Corps of Engineers)

HEC-RAS [HEC’s river Analysis System](http://www.hec.usace.army.mil/software/hec-ras/) that models the hydraulics of water flow through natural streams

HFT Hazard Factor Tables

HIFLD [Homeland Infrastructure Foundation-Level Data](https://hifld-geoplatform.opendata.arcgis.com/datasets)

HPLF High Potential Loss Facilities

HVA Hazard Vulnerability Assessment

HSEEP [Homeland Security Exercise and Evaluation Program](https://www.fema.gov/media-library/assets/documents/32326)

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 Mercali Intensity

MSC Map Service Center; one of five units in CDS (see above)

NAVD North American Vertical Datum

NHC [National Hurricane Center](https://www.nhc.noaa.gov/)

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](http://water.noaa.gov/about/nwm)

OGC [Open Geospatial Consortium](http://www.opengeospatial.org/)

ORNL [Oak Ridge National Laboratory](https://www.ornl.gov/)

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](https://www.tudelft.nl/en/ceg/about-the-faculty/departments/hydraulic-engineering/sections/environmental-fluid-mechanics/research/swan/) wave model (developed at TU Delft, NL)

TIGER [Topologically Integrated Geographic Encoding and Referencing system](https://www.census.gov/geo/maps-data/data/tiger.html)

UDF User-Defined Facility

UI/UX User Interface / User Experience

USGS [United States Geological Survey](https://www.usgs.gov/)

WKT Well-known Text, an ISO and OGC markup language standard for vector geometries

WSEL Water Surface Elevation

# Appendix E: References

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.

1. <http://blog.ibmjstart.net/2015/08/22/dynamic-dashboards-from-jupyter-notebooks/> [↑](#footnote-ref-1)