GEM IT Review - Reviewer's Report
20 July 2010
Final

Notes area

by Stephen Arnold, ed.

v 0.2 – 20 July 2010

With contributions included from:

Steve Arnold, Gentoo Foundation , Allan Hancock College

Shoaib Burq, Geoscience Australia

Jessy Cowan-Sharp, Sunlight Foundation

Stuart Gill, World Bank

Jocelyn Guilbert, Commissariat à l'Énergie Atomique

Jano van Hemert, Edinburgh University

Andreas Hocevar, OpenGeo

Chris Holmes, OpenGeo

Heiner Igel, Munich University

Linus Kamb, European-Mediterranean Seismological Centre

Phil Maechling, Southern California Earthquake Center

Joshua McKenty, NASA Nebula

Kevin Milner, Southern California Earthquake Center

Timo Multamäki, Eigenor Oy

Alessandro Spinuso, Observatories and Research Facilities for European Seismology

Frank Warmerdam, Open Source Geospatial Foundation

Notes area

Table of contents

Notes area

Notes area

Notes area

1 - Overview and consensus recommendations

The primary (IT) goal of GEM is to extend the scope, but not necessarily the state-of-the-art, of risk and hazard calculation, as related to earthquakes. In order to achieve this, we expect our IT architecture to be:

- Open (Data, Source, Protocols, Standards, etc.)
- Pluggable (Modular, Loosely-Coupled)
- Dynamic (Fault-tolerant, Distributed, Constantly Updated)

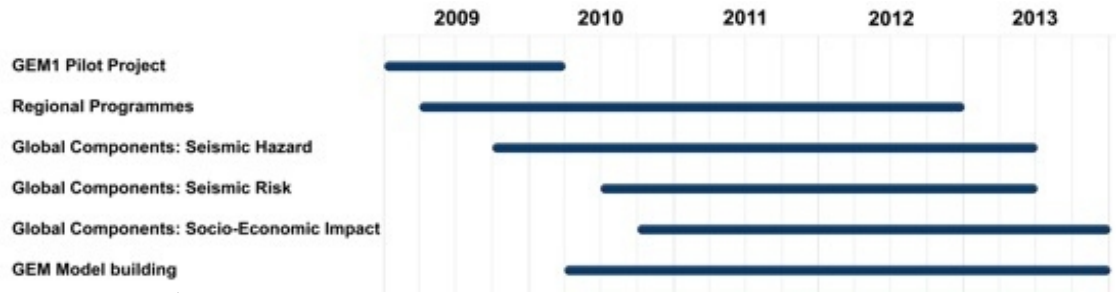

Figure 1. GEM Project Schedule

Note that the organization of this document, including the Section and Subsection naming convention, was more-or-less taken directly from the wiki page structure. The sections were roughly organized by content, with the notes section still in alphabetical order of reviewer's names (same order as the wiki). This document also contains some hyperlinks back to the wiki pages, which may be newer than what was current at the time this report was drafted. That said, any updates made directly to this report may not be reflected in the wiki.

The OpenGEM project has been disected into the following pieces:

- Over-arching architecture
- Database Schema
- Application Layer (including DAC and SOAP-based Web Services)
- Risk Engine (TBD)

Notes area

- Hazard Engine (including SHAML spec)
- Portlets
- Experimental UIs
- MapServer Stack (consumed by Portlets, currently)

Review outcome

What GEM did very well:

There is plenty in the GEM project which is good and commendable. We have listed some specific points, where we jointly agreed on an above average performance.

Looking for help from the outside (us) – in a formal way:

- Putting effort to get expert involvement. (Doesn't happen in most of the other European projects.)
- There is a will to understand the best approaches within GEM.
- The GEM secretariat has managed to find a good cross-section of different backgrounds for the technical reviewers
- Despite having vague requirements, GEM1 has already shown that the existing GEM team is capable of making results and not just idle talk
- Future-looking at an unprecedented scale. (Great potential)

Openess:

- Made a lot of good decisions vis-a-vis being open. Want the code to be FOSS. Thought about licenses. Open to criticism and feedback.
- GEM is a learning organization and it shows. Not afraid to question each other, try to do better the next time. (Vis-a-vis Ben's prototypes)
- WILLINGNESS to throw out the "built one to throw away"
- Hazard side really came together (around openSHA) with very little IT support. (Made good use of OpenSHA external team).
- Trying to do something great, haven't collapsed under their own ambitions.
- Willing to expose themselves for criticism (very atypical).

Other issues:

- Funding is appropriately arranged, which is rather rare in scientific related large projects. As project is not only getting funds from a single source, the probability of actually getting commonly accepted results is vastly larger. What they did

Notes area

accomplish (SHAML spec, etc) is very impressive.

Productivity:

- Produced the global hazard map.
- Doing int'l distributed software development (which ain't easy to begin with).
- GC and RP organization seems well defined.

History and Context

We did not feel there was enough reference to outcome of Canberra meeting including recommendation that we focus on users that can verify and validate calculations done by system. In future IT reviews, we recommend a concerted effort be made to make explicit reference to previous reviews.

We recommend making public a roadmap with specific milestones, both to guide internal development as well as an aid to coordination and collaboration with external communities. Additionally, the decision tree or review process that's guiding prioritization of development should be explicit and published as well.

Ideally, both of these artifacts would be presented as a comparison against specific, existing projects and programs.

Management recommendations

Team management:

- Hire a full time IT manager with experience in FOSS development
- Manage as a single, distributed team
- Distributed is helpful – it will encourage the use of good FOSS practices
- Force good FOSS practices now (internally), before opening up project
- Frequent project reviews (architecture, code, spec, test plans, continous integration)

Unified and cohesive Team:

- Frequent intercommunication and meetings (face-to-face, virtual)
- Consider FOSS methodologies and tools to promote distributed team cohesion
- One set of tools
- Simplify (use the tools you set up)

Notes area

- One repo, one document store
- Allow easy creation of branches and merging to allow flexibility
- It could be a good ideas to organize GEM workshops with scientists and 'users' to be sure that the DB and software (or portal) are in good correlation with the needs.

Hiring:

- Consider a UI/UX designer early on
- Technical Manager (see above)
- Part-time FOSS community engagement role (also see section on Community – FOSS and User Community). Continue to have IT reviews (small, targeted, frequent).

Testing

We recommend that a strong emphasis be placed on testing during subsequent development. This includes load testing (to identify potential scalability issues), regression testing (to achieve verification of scientific validity), and user testing (which will involve getting rapidly-developed prototypes in the hands of real users and collecting detailed feedback.)

(and don't forget unit testing as part of the personal software process)

Establishing the Continuous Integration environment is a top-priority. Test-driven development without CI, isn't really test driven development.

Commit to a methodology:

- Might be agile or waterfall, choose and then stick to it.
- Use the same methodology for the entire project.
- Let the selected methodology inform the architecture.
- Follow the full set of practices. E.g:
    - If agile, perform force-ranking of captured user stories. Hold SCRUM meetings, and collect user test data at the end of every sprint.
    - If SDLC, fully decompose identified use cases to a functional specification.

Prioritize Requirements: (Tighter Focus)

Get a clear set of prioritized requirements (user stories, gurkin, etc.) This does not preclude rapid, iterative, or test-driven development, but it requires very clear ranking – even if for short periods of time.

We recommend you consider the GC teams as your initial, high-priority set of system users. As well, from a management perspective, if these

Notes area

teams are developing software, they should be considered and coordinated as if FOSS external contributors. Use FOSS project management, team communication, peer review, etc.

The existing prioritization of users is weak. The concept of actors may add confusion – Actors seemed to include both end-users, decision makers, and funding organizations. More specifically, however, these Actors were not referenced in later presentations.
If you are going to advance these as a development tool, consider developing an end-to-end use case example (scenario) for each type of actor.

Again, understanding requirements does not mean spending years writing – short, iterative releases (in weeks or months) with real user testing are a highly effective tool for gathering requirements. Ensure the ICT manager has full responsibility for these use cases; she can present them in the next review, and explain why design decisions match the requirements of these cases.

Leverage Existing Projects More

- For difficult code, esp. where there are existing well-tested components
- For collaboration, community building – contribute to them, don't just use
- I.e. GeoNode
- Go visit users and ICT experts in person
- Exploit userbases of other projects, e.g., NERIES mentioned the users of their operational services

Community (FOSS and User Community)

- Don't defer this
- Adopt FOSS practices internally (Fogel's 'Producing Open Source Software' is a great handbook for all the best FOSS practices)
- Regarding community management:
    - Each team member should be participating in the community now, even if the source is not yet open:
    - A very rough estimate of time might be 3-4 hours/week for developers to engage in community activities
    - Build in exception notification to promote accountability for the code base
    - GEM Blog: Blogging about the project, interesting technical challenges, etc.
    - External participation – attend meetings and meetups.

Notes area

Consider a training and conference budget for each developer to get out into the community, learn, and simultaneously increase awareness of the project.
- Organize development sprints -
  - within the team, and with the broader community.
  - sprints can seed projects GEM will build on
- Contribute to other (external) FOSS projects – provide patches, help out their docs, make examples for them, earn commit rights
- Encourage taking ownership of tasks – by ticketing, nominating domain experts etc.
- All decisions taken by the project should be done in archivable manner, on mailing lists, irc, or at least documented if decisions have to be done in person. Even if project is not 'open source' for next 6 months then archives should be opened when it gets opened.

- On the question of hiring a full or half time community manager, may consider someone half time, but more important is for each member of the team to have at least 4-8 hours a week where their priority is doing the above activities.
- It could be a good idea to organize GEM workshops with scientists and 'users' to be sure that the DB and software (or portal) are in good correlation with the needs – these can be arranged in concert with the development sprints mentioned above.

FOSS Development Process:

- Use a style guide for writing/contributing code, writing tutorials, using the collaborative space, etc.
- Commit to a consistent naming scheme (current naming schemes have collisions and confusions)
- Plan for ticketing, reviewing, accepting contributions
- Don't call it FOSS until it's FOSS. See http://en.wikipedia.org/wiki/Free_and_open_source_software
- Specify a date at which code will be released, community will become open.

Social Engagement and Media

There was an early discussion of crowd sourcing, and the significance of the GEM work. But the review did not return to this topic of hazard and risk and how GEM can help with this during later discussion.

Notes area

- If Crowdsourcing is a key requirement, it needs to be addressed earlier in the design
    - Crowd-sourced dataset generation (such as fault or exposure databases) is a role that should be supported in the system – build social tools based on those use cases, not abstract 'social' ideas.
    - Proper metadata and formats might be needed in order to have these user generated datasets interoperable and searchable.. consider the Linked Data and persistent URIs approach for non-sensitive datasets
    - Make the "Haiti Children" aspects of "why we're doing this" a more central part of the messaging (esp. in the FOSS community side)
- Suggested frameworks include django (http://www.djangoproject.com/)
- Exploit existing social networks, how many seismologists are on LinkedIn?
- Regarding "semantic wikis"
    - This was mentioned but not clear what the role is. Why semantic over any other kind of wiki? What does this mean?
    - Efforts around ontology and semantics formalization would be good to start as a public facing, community effort now.
    - With respect to GEM as a system, focusing on developing ontologies and semantics might be out of scope.

Licensing

- CC-zero for content http://creativecommons.org/choose/zero/ (PDDL is a similar option, but CC has nicer awareness)
- A statement on community norms, like http://www.opendatacommons.org/norms/odc-by-sa/
- Add to that a statement that all data contributed to GEM will be open for ever
- Statement on community-contributed content (will be public, open for ever)
- LGPL is good enough, revamp if necessary (be open to relicense if requested)
- Collect Contributor License Agreements (copyright assignment)
- Use PPK to sign authoritative, GEM-certified data assets. (In the future, you could extend this capability to allow any user to sign their data products, both for authenticity and data integrity).
- Design nice logo for GEM-certified data assets, trademark it and

control its use

Notes area

Data Schema / Database:

- Address the concerns from the scientific community regarding schema
- Abandon a monolithic database
- May need to support distributed querying
- Decide if it's important to optimize for the most remote, poor-connectivity regions
- Defer lightweight/standalone clients until primary capability exists.
- Develop tools for DB visualization in terms of quality of the inputs and harmonization of the different tables
- Be sure that the DB embeds all the needs of users in terms of computation or inputs for the computation
- See a first scheme of the organization concerning the DB management (if the DB is distributed this point will became more and more critic)
- Resolve conflicts between SDLC-style data modelling, ORM-based modelling, and the necessary schema evolution (additional mapping layers may not be the ideal solution)
- Consider schema-less, column-oriented, and document-oriented alternatives to RDBMS
    - To deal with Schema extension / churn
    - For scale of certain data types

Data Exchange Formats:

- Various reviewer notes in Section 3 will include specific commentary around the data exchange formats
- SHAML, QuakeML adoption, etc. need more prominent attention as specific IT efforts
- Define the minimum requirement, keep to the minimum
- Plan out extension of SHAML to cover risk and SEI
- Be cautious regarding the level of effort involved in support of these formats
- Several reviewers have concerns about the scalability or suitability of an XML format to the type of data exchange being proposed. Look for various viewpoints in Sections 2 & 3.

Front end

- Central issue: "JSR168/286-compliant frameworks (i.e. portlets)"

Notes area

vs “non-JSR168/286 framworks”, see also JanoNotes and Django recommendations
- Emphasize requirements, not implementation spec (eg. JSR168/286)
- Not as important to choose portlet versus non-portlet; instead requirements should dictate appropriate (and help identify inappropriate) tools and technologies
- Doesn’t seem like the requirements for front-end are well understood
- Disagreement: majority felt portlets are not the right technology, but there were dissenters. See JanoNotes for more on JSR and portlets.
- Make use of tools to build portlets, do not manually develop everything, this is too expensive and too expensive to maintain in the future
- The rapid prototyping effort (in javascript, python, or other high productivity languages) seemed valuable to explore potential features for implementation. In the context of JSR development, tools such as Rapid will speed up prototyping to days rather than weeks or months.
- Front-end capabilities should be analyzed from two directions
    - bottom-up analysis: portal should expose key functionalities of the system
    - top down: UI mockups need to be done which identify desired user-facing capabilities; those are likely to highlight additional technical functionalities which need to be added to the spec. e.g. user sharing of content, ability to edit/fork models, etc.
- Understanding the front-end

Calculation Engine / distributed computing:

- Don’t buy or build a cluster, consider using free solutions first, e.g., clusters and Grids (EGEE, D-Grid, UK-NGS, TeraGrid, NorduGrid, TGCC-CCRT, etc.), then move on to HPC, if you have HPC requirements (PRACE, DEISA2), and cloud vendors (Amazon EC2 and many others).
- CUDA / GPU processing / MPI is all premature optimization, may not fit nature of calculations
- Condor-type, embarrassingly parallel (high-throughput) solution is appropriate, but should not tie into one ‘job submission engine’. Keep dependency on these systems as low as possible (See Section 3 for specific recommendations)

Notes area

Distributed System / Standalone OpenGEM

- Defer addressing construction of lightweight client, standalone system, or federated/distributed system for 2 years
- However, develop calculation engines as standalone applications **now**
- Consider CLI interfaces to these components (as well as WS/REST), many tools exist to take CLI and then wrap these into services. Also many tools exist to web-ify CLIs, including interfacing with cluster/HPC/Grid resources
- Keep engine, front-end and datastore loosely-coupled and independent
- Consider these as potentially distributed systems vs. simply distributed data sources, but defer design decisions (specific recommendations in Section 3)
- Modular design (stand alone calculators, decoupled web applications) implicitly distributes computing resources

Deal with Scale

- GEM's scale distinguishes it from similar efforts. Support for scale seems to be a critical requirement
- For RDBMS, generate example data sets, populate tables and evaluate usability. Modify design of database, or design of data storage if system does not support the required global scale.
- Scale consideration include more than just RDBMS. Consider thinking about future data-intensive needs
- Project management courage is required when taking decisions as future scale is difficult to assess

Rapid Prototyping

- Adopt YAGNI: "You ain't gonna need it"…don't add functionality until needed
- Release early and often, get real user feedback
- Look for and use appropriate tools for rapid prototyping (including pencils)
- don't set up tools for the sake of setting up tools. For example, GEM TRAC system is installed but does not seem to be used..if you're not going to use it, abandon it.

Web Services

- Publicly accessible, versioned service API should be included

Notes area

- Consider using REST – and use it to access services from web applications
- Consider using Web Services Resource Frameworks (WSRF) as the project is exposing resources
- Dogfooding (that is, using the tools that you create) is important (ideally done by different people, and with a different programming language. etc.)
- Make sure an external (or at least distinct) user is also consuming the API, preferably in a second language (javascript is ideal, since it has lots of constraints and is likely in mashups)
- Current system design may overuse web services internally (only make sense internally when accessed from decoupled web applications)
- The current granularity of the web service does not seem quite right. It would likely be a mistake to convert to XML format between seismic application and database, and to use only web service interfaces to access database.

Single Sign On (SSO)

- Most felt the existing SSO model was not quite right – too tightly coupled to portlets, did not adequately protect data sources.
- Adopt an existing standard mechanism, don't reinvent the wheel
- (Specific recommendations in Section 3, many favor SAML/Shibboleth)

Additional resources and links to development tools

- Software for distributed agile project management: pivotal tracker
- Language for user stories: Gherkin
- Collaborative document editing: etherpad
- Test Driven Development TDD
- Behaviour Driven Development BDD a logical extension to TDD
- Grinder – Java Load Testing Framework
- UI Mockup Tools: GoMockingbird and Balsamiq Mockups
- O'Reilly RESTful Web Services
- Fogel's Producing Open Source Sofware
- ReviewBoard – A tool to make code reviews fun…
- Find Bugs FindBugs is one example of a static analysis tool for finding bugs in Java code.
- Schematron to valdate XML
- Rapid to quickly build (without traditional programming) JSP servlets and JSR portlets for running back-end computing jobs

Notes area

- RelaxNG keeps XML readable
- JING make screencasts to show users how it works and what they can expect and share these

Acronymns

- REST Representational State Transfer Article explaining REST in non-techy speak (slightly sexist)

Notes area

2 - Specific supporting topics

DataSchema - OpenGEM Database Model Ver 1.5

Overview

ORM mapping using Hibernate in JAVA.
ERD Deconstruction
Classic 3rd Normal Form

DB SERVER: gemsun01.ethz.ch

Tools:

- Java Topology Suite
- QuantumGIS
- PgAdmin III
- Hibernate Tools, including Hibernate Spatial

Concerns

- System-level tuning (e.g. complex sharding schemes, postgres-specific DB tuning efforts, expensive hardware, etc) is not necessarily portable for software that should be also run locally.
- The GEM system data may be best expressed in a combination of SQL and NoSQL formats. Innovative approaches to data representation may be necessary, esp. for the (potentially large sets of) point data.

References

- LTree Module

NoSQL Options

“CouchDB”: http://couchdb.apache.org/

MongoDB

Notes area

- Example of (crazy) people using it for GIS: http://www.paolocorti.net/2009/12/06/using-mongodb-to-store-geographic-data/

Cassandra

Riak

MonetDB has OpenGIS included. They are very willing to help. Jano can provide contacts,

Is it feasible to have GEO NoSQL? Some folks say yes…

Quantified Userbase

How many users will GEM initially be supporting, and of what types?

Model Builders: Global Components

- There are currently 5 Global Component consortia on “hazard”
- There are currently 5 Global Component consortia on “risk”
- There will likely be 1-2 Global Component consortia on “SEI”
- Each of these is estimated to represent between 40-100 users.
- (Total of 440-1200 users.)

Model Builders: Regional Programmes

- There are about 10 Regional Programmes
- Each is estimated to represent around 250 **active** users.
- They may require i18n, l8n, training and capacity building.
- (Total of 2500 users.)

Minimum Requirements

Due to opportunities for capacity building, etc., it is reasonable to develop the GEM platform targetting modern browsers (e.g., HTML5 and WebSockets).

GEM use of Web2.0 and Social Media

Much of the data required for a global understanding of risk is currently

Notes area

unavailable.

Crowdsourcing is considered a key approach to collecting this data. This may also include aspects of contesting or other incentive schemes (ala DARPA's red balloon).

Methodology fight: Agile Vs. Waterfall

This section was started by Steve (who wanted others to contribute) and was intended to capture some thoughts about agile methods vs. the more traditional "waterfall" model of software engineering, but focused on the context of the GEM requirements for both "openness" and scientific verification.

Although agile methods hold the promise of both rapid development and a way of handling evolving/unknown requirements, a more traditional approach to requirements engineering, including some form of verification and validation of the software, may lend itself better to the goals of V&V in the context of GEM. The key from my point of view will be integrating a solid V&V effort with an appropriately "agile" project environment.

Areas of emphasis that should always be included:

- Frequent communication among relevant project members (up, down, and sideways).
- Use of the proper tools to understand (and document) source code. Someone else will struggle to figure it out at some point (and it may even be you). See the StaticCodeAnalysis page for details.
- Don't let anyone struggle alone; hold peer reviews, assign mentors and/or partners, and make sure your co-workers aren't stuck on something or waiting for something (and don't be afraid to throw away painful code and re-implement it to get it right).

Static Code Analysis

You should always analyze your own code, including generating metrics & documentation, as well as making full use of static analysis tools. Many great free tools are available, and most provide both GUI front-ends (e.g., an Eclipse plugin) and command-line interfaces (for backend automation). A short list of tools will give some examples:

Notes area

Statistics and metrics

- SLOCCount – SLOCCount is a handy tool for counting source code and basic project estimation. Too useful not to use it.
- CCCC – CCCC generates detailed metrics and points out potential problem areas in your code (with nice html formatting). Another extremely useful tool…

Documentation and design extraction

- Doxygen – Doxygen is a source code documentation, analysis, and reverse-engineering tool for several languages, and can be extremely useful when incorporated into the software development process (ie, by defining your coding style to include doxygen-style comments; can also use JaveDoc style) and incorporating your own internal docs as well.
- JavaDoc – JavaDoc comes for free with the JDK. Use it.

Software engineering and analysis

- Eclipse – Eclipse is an all-in-one software environment for multiple languages, with plugins for pretty much everything from analysis to visualization to test and deployment. Yes, it can get bloated, but it's also the best interface between human and code ever invented.
- Plugins – There are so many plugins for Eclipse they needed eclipseplugincentral.com…
- FindBugs – FindBugs is a tool for finding bugs in Java code. If you write (or work with) more than 10 lines of Java, you should really be using this tool…
- PMD – PMD is another analysis tool for Java (not at all orthogonal to FindBugs). Also highly recommended.

Software engineering tools at tigris.org

- Subversion – Clients, plugins, docs, etc (in the process of moving from tigris.org to apache.org).
- ArgoUML – The best free UML editing system around (supports DoDAF notation).

Other

- Source Navigator – Source Navigator is a cross-platform software development and code analysis tool with multi-language and multiple toolchain support.

A short presentation on this topic is also available on SlideShare

Notes area

Outstanding Questions

Editor's note: Some of these seem to be answered in the individual write-ups.

- What existing open source projects (with attendant developer communities) could be co-opt'd / merged with GEM?
- What primary test case (with attendant full-scale data set, etc) can be used to further develop / validate requirements during the next 6 months of development?
- What sort of "requirements roadmap" can help guide this "agile" process?
- Why Latitude X Longitude instead of Geocode?
- Is it feasible to achieve a unified data format (for exchange as well as internal representations), or will the internal representation always be more complex than exchange format?

Notes area

3 - Individual reviewer's notes:

AlessandroNotes

Great Experience, IHMO it was about the time to have such a cross domain/religion/philosophy meeting related to the IT developments within the seismological community, with the support of an extremely updated group of experts with different backgrounds and experience. Hence, first of all, thanks to the GEM group that made it happen!

The Requirements

It seems that one of the topics where the reviewers reached a general consensus is the need for the accomplishment of an important target: The Definition of Use Cases and Functional Requirements. And this point is crucial.

Accordingly on my personal experience as a software developer on big projects having as a customer a scientific community, quite often we are requested to have the role of the user, the coder, the designer, the architect, the tester, and last but not the least, the sales manager.
We have to realize whether there`s a need or a problem, solve it and then sell either the problem or the solution :-).

There`s an interesting article talking about scientific portals or VREs (Virtual Research Enviroment) which says:

“The development and presentation of a VRE must be embedded and owned by the communities served and cannot realistically be developed for the research communities by others in isolation. Since the intention is to improve the research process and not simply to pilot technologies for their own sake, the research must drive the requirements”

“*A VRE which stands isolated from existing infrastructure and the research way of life will not be a research environment but probably only another underused Web portal”*

Michael Fraser Co-ordinator, Research Technologies Service, University of Oxford http://www.ariadne.ac.uk/issue44/fraser/

The approach of GEM should consider to involve as much as possible the users community in the process of defining the requirements. This will help the developers on focusing on the right solution adopting the technique that more fits their needs and their skills. Moreover in the light of the extremely useful tips and guidelines proposed by the reviewers, those skills that are currently lacking can be quickly improved whether

Notes area

an accurate analysis of the problem and the understanding of the available technologies motivates a change of direction.

GEM1

The design of the GEM1 project depicts an extreme modularization and independence of each piece of the architecture, from the database design to the service and presentation layer. A database expert might not be as good on front end or UI development, so “sometimes” the decoupling is a good thing. This obviously required the ability to gain a deep knowledge on a wide and sometimes complex stack of technologies, standards and formats.. the question is, does this lead to the over-engineering of the project? Is over-engineering always bad? IMHO, the lack of clear requirements leads to over-engineered software.

In general, for all the Java developments I`d suggest to consider the adoption of the **Spring framework** which helps in keeping the code clean, modular, testable and lightweight thanks to massive adoption of the IOC and singleton patterns. It provides the glue to integrate several small components in a bigger architecture, keeping the small things.. small and testable. unit testing with spring and the available IDEs works just great.

The webservice approach

Given the Use Cases, there`s the need to understand which of them has to be implemented as a webservice (in its general definition of a programming API available on the web) and which on the other hand doesn`t need to be a webservice. For instance, talking about a browser based product, most of the user interaction could be implemented through a normal MVC pattern where the front end is “directly” connected with the database, better saying to the objects stored in it. Would it be possible and worthwhile to model every use cases as an aggregation of REST webservices? Probably if you start a project from the scratch that`s the way to go..

Some different considerations have to be done in respect of asynchronous call to processing facilities. It`s advisable to use a queue based system where the fire and forget approach is given by definition and the queue can retry a number of times if the delivery fails, submitting a certain number of job in parallel or in a chain. A webservice communication over HTTP implies that the requester needs a response back., Do you want to model such a protocol from the scratch? Is there anything already available? A combination of the two might require to get into a better understanding of the **WPS OGC** specs. This link provides some interesting thoughts http://www.cadmaps.com/gisblog/?p=28.

Notes area

To Portal or Not To Portal

JSR portlets are complex stuff if you don`t rely on the proper framework, but the general concepts behind a JSR portal, such as personalization, component-based development and the reuse of tools are spreading among several web frameworks implemented also with other technologies, which is worthwhile to investigate better. On the other hand, several successful projects and teams in e-science adopt the JSR-168 solution, suggesting that this approach might enable a better integration of external expertise on scientific portal development.

Portals provide the ability to aggregate access to applications. In many development environment or collaboration, these applications are owned and maintained by disparate groups where the coordination of release schedules can be difficult or impossible. In such a scenario, WSRP provides the advantage of allowing a portlet or group of portlets to be released independently of the main portal application. This, together with the opportunity of cross domain collaborations, was a fundamental feature within the NERIES project. OpenSource projects like Jetspeed provide a very lightweight stack, while Liferay is moving towards a cross platform solution which is extremely interesting, besides many new social features provided off the shelf.

If the mentioned motivations are not an issue in GEM, probably considering a framework such as **GeoNode**, where many interactive features related to a collaborative manipulation of GIS products are implemented out of the box, could be helpful, also in terms of future collaboration and contributions to the Risk Assessment web development community.

I`d suggest though to wait for a stable release rather than rely to the existing Beta.

Interoperable Metadata

To what extent some data product should made publicly available on the web, and in which format? Investigating on proper metadata and publication philosophies such as LinkedData http://linkeddata.org/ might lead to a cross domain interoperability and discovery of public data products. This is also related to some concerns expressed by Fabian during the QuakeML talk.

Being Social within and beyond GEM

Datasets, data-products, discussions, GEM portal activities. Might be useful to push all these meaningful information outside the fences of the GEM infrastructure, in order to achieve ease of access to interesting

Notes area

results and wider visibility among the domain field community and beyond.

For this purpose consider beside the implementation of a professional network within the GEM portal, the adoption of well established web2.0 platforms and tools widely used by millions of users, such as iGoogle and Twitter.

AndreasNotes

Data Storage

Well established file formats for spatial data:

- vector: Shapefiles
- raster (generic): GeoTIFF
- raster (hazard related): AME

Developer Community

After talking with some of the team members, it seems that most of them are not expecting to attract developers.

One important thing to consider here is: setting up a FOSS style software development process is a win no matter what. And once this process is in place, accepting external contributions is not a burden, but a help.

FrankNotes

Gridded Data Management.

- Keep gridded data products outside the database (ie. hazard maps, hazard curves).
- Keep gridded data products as managed raster files, with references by filename in the database.
- Support accessing gridded data products by WCS, including implementation of shaML support in one or both of GeoServer and MapServer, and clients like GDAL.
- WCS “references” could be the primary means by which gridded

Notes area

data products are used from remote locations when copying is discouraged.

Processing Engine

The existing risk processing engine does not seem to address some concerns I have:

- approaches to distributing the work over a cluster. Try to avoid complexity, or deep ties into specific clustering technology.
- Look into an engine capability to split up large product calcuations into chunks (ie. great a global calculation into smaller tiles)
- How to integrate existing processing algorithms (possibly like OpenSHA) that do not work at the same fine grained level as the processing engine. For instance, some algorithms may not be easily broken down into stackable filters, and may not support the virtualized access to input data.
- I think they need to have a distinct configuration file input format to drive the processing engine(s) so the processing engine is quite distinct from the web services. The configuration file (referencing other input files) becomes the definition of a processing run.
- For local processing what they should distribute is the engine + modest tools to prepare the “run” configuration file.

Portal, Web Services

Not my area of specialty, but I am doubtful about the use of SOAP/WSDL for the web services. It is a heavy approach which is clumsy for clients. I would contemplate a lighter weight ReST approach for the web services **instead** of the SOAP/WSDL approach.

Further, I would consider a portal development approach that is more organized around JavaScript client technology (as was done in the Pavia prototyping) built against the ReST API for web services rather than the Java Portlet approach.

I do think that an effort be made to avoid passing large objects (like whole hazard maps, complex logic trees, etc) between web services. Instead identified large objects should be referenced, possibly from the database or from files on disk.

Data Formats

- Spend time defining, and documenting file formats to be used

Notes area

for import/export of data with the GEM system, including the formats used as input to the processing engine(s).

- Utilize pixel interleaved GeoTIFF with specific metadata extensions (possibly parts of shaml in a tag) as a working, interchange and archive format for hazard curves and hazard maps. Such a format is compact (binary), efficiently accessable, and already supported by many existing software packages. I can assist.
- Use of shaml as-is for processing inputs seems ok.
- Try to avoid creating specific formats where a simple/specific profile of an existing format (like GML) would do (for faults, etc)
- Put some effort into identifying existing adhoc tools for preparing system inputs, and visualizing system output products.
- Put some effort into developing additional adhoc tools for working with the data formats, possibly including development of GDAL/OGR drivers (for the C/C++ stack), and GeoTools format handlers (for the Java stack).

Building Modelling

Future work around capturing information on building characteristics globally was not discussed in any depth during the IT review, but I believe there needs to be some thought applied to how the information is stored, accessed, and managed.

- Consider developing a simple GML profile for building models.
- Consider storing in a distinct building model database lest the volume of building models eventually collected eventually overwhelms the general purpose GEM database.
- Consider offering WFS access for read and update to the building model database.

Strong Points

- For the most part the normalized database structure seems sensible (bulky data notwithstanding).
- The LDAP auth architecture with users/groups seems good.
- The development of shaML as a core working format for interchange, and data input/output from the calculation engine seems good.
- It looks like excellent work was done building on OpenSHA.

Auxiliary Points

- LGPL is an ok license for the developed code, though a non-

Notes area

reciprocal license (like BSD, MIT, etc) would allow folks like insurance companies to make proprietary improvements to the modelling code without an obligation to release them.
- SVN is adequate for source control, though a distributed VCS like Git has some minor advantages.
- I don't see any compelling reason to move development from Java to Python with the possible exception of adopting a technology like Django. In any event, having the processing engine in Java is fine.

MapServer vs. GeoServer

- Both are ok, so it would likely be best to let the team pick whichever seems like the best fit.
- Be prepared to invest some effort back in support for GEM oriented file formats for whichever server technology is selected.
- Note that GeoServer is well suited to web based feature update vis WFS-T and a client like OpenLayers. MapServer does **not** support WFS-T (update via WFS).
- It is possible it will make sense to deploy both GeoServer and MapServer for particular purposes.
- Ensure that service deployment is based off the product definition within the GEM DB, not having to dump all the data to disk in duplicated forms. For instance, via some appropriate wrappers it should be possible to access any hazard map in the GEM DB without having to constantly dump the GEM DB map list out to some particular file format (MapServer .map, or GeoServer configuration file). In MapServer this would normally be accomplished with dynamic map technology using MapServer to lookup details in the GEM DB. Some similar mechanism no doubt exists for GeoServer.

Open Source Contribution

- It is unlikely that there will be a great deal of outside contribution to the core DB and webservices of OpenGEM.
- There might be some contribution of portlets for the web site.
- There will almost certainly be some contribution of adhoc tools for preparing, visualizing, translating and managing the various inputs and outputs of GEM. Some should be "captured" by GEM, while others will exist in other homes (ie. GDAL/OGR) and should just be referenced as available resources.
- Likely there will be scientists wishing to development experimental/local variations on the modelling code and configurations. These cannot be upstreamed without great care

Notes area

(to avoid invalidating the global modelling code), but it would nice to be able to share these effectively. Use of a distributed source control system likeMercurial or git might be helpful in this regard.

- Some contributions will come as improvements to packages like GeoServer, and GDAL used by GEM.

HeinerNotes

- My comments on HPC (and regarding benchmarking, reproducibility) are well incorporated in the general review.

Some general comments (not directly IT related)

- Make sure that the portals offer specific applications for students (or even make specific reference to tutorials for students) this may help developing a generation of young Earth scientists who know the project/structures from the begining and can actas multipliers

- GEM is extremely ambitious as is with the specific goals set in connection with attenuation relations etc. In my view real progress in preparing societies for impending earthquakes will come more and more from time-dependent hazard analysis in all it s various forms (fault interaction, remote triggering, stress transfer, superswarms). I can not give specific recommendations on how to incorporate this somehow, but maybe GEM can over the years also develop in this direction. Some of the things I was wondering about during the meeting are now addressed in a paper just publishedi n SRL:
Opinion: Operational Earthquake Forecasting: Some Thoughts on Why and How <http://www.seismosoc.org/publications/SRL/SRL_81/srl_81-4_op.html>(Thomas H. Jordan and Lucile M. Jones)

JanoNotes

Background of Jano van Hemert

Dr Jano van Hemert has a PhD in Mathematics and Physical Sciences from the Leiden University, The Netherlands (2002). Since 2007, he is a Research Fellow in the School of Informatics of the University of Edinburgh and since 2005, a visiting researcher at the Human Genetics Unit in Edinburgh of the United Kingdom's Medical Research Council. He leads the UK's National e-Science Centre, supported by an EPSRC

Notes area

Platform Grant.

His personal research group, Edinburgh Data-Intensive Research, comprises 6 post-doctoral researchers and 5 PhD students. He currently leads 4 projects and is involved in 6 more, all of which are national and international projects, with active collaborations in seismology, brain imaging, developmental and evolutionary biology, fire safety engineering, nano-engineering, urban water management, molecular medicine and neuro-informatics, funded by the Engineering and Physical Sciences Research Council (EPSRC), the Biotechnology and Biological Sciences Research Council (BBSRC), the European Commission (EC), the Scottish Funding Council and the Joint Information Systems Council (JISC).

van Hemert has held research positions at the Leiden University (NL), the Vienna University of Technology (AT) and the National Research Institute for Mathematics and Computer Science (NL). In 2004, he was awarded the Talented Young Researcher Fellowship by the Netherlands Organization for Scientific Research. In 2009, he was recognised as a promising young research leader with a Scottish Crucicible. All of his projects are interdisciplinary collaborations and many of his research projects have included partners from industry.

van Hemert is an editor of five international computer science journals. In the past five years, he was the programme chair of five international conferences and workshops in computer science. In 2008, he has published a book on Recent Advances in Evolutionary Computation for Combinatorial Optimization as part of Springer’s Studies in Computational Intelligence series.

His research output includes over eighty published papers and software on optimisation, constraint satisfaction, evolutionary computation, data mining, scheduling, problem difficulty, dynamic optimisation, distributed computing, web portals, experiment design, e-Infrastructures and e-Science applications.

Data-intensive refers to huge volumes of data, complex patterns of data integration and analysis and intricate interactions between combinations of users and systems that deal with these data. The mission of my research group is to advance methods that harness the power of data and computation in collaborative environments. The goal is to support the life cycle of data to information to knowledge in a multi-disciplinary and multi-organisational context. To achieve this we pursue research in e-Science and Informatics and apply our methods in several scientific and industrial domains.

Notes area

High-level recommendations

JSR168/286 Portal Frameworks

I am a big supporter of the JSR portal framework, it is tried and tested in many scientific communities and is already in operation in NERIES. The standard has been around since 2001. Good tools available to speed up the development, which are not used here yet. See for example this video on producing portlets in minutes

There are plenty of papers that describe successful use cases with JSR. Two examples, an American-based one and European-based one

The National e-Science Centre has much expertise in this area. They also use JSR as the preferred framework for development. The actual solution (implementation of the framework) used has changed over time, previously we used GridSphere, now we use LifeRay. Fortunately, any developed portlets can fit into any of these, so when the solution changes, no additional effort is needed. They have several projects based around scientific portals (sometimes called gateways) for instance on Hazard forecasting , Microscopy , Genetics , Supercomputing , Seismology , Developmental Biology , Chemistry and Brain Imaging

Modern implementations of the JSR framework offer many social networking features. Try installing LifeRay (it takes less than a minute to try out). On the first page is shows many of its features; which include direct use of Facebook and Google features.

Data scaling

The right solutions must be chosen to address the data scaling, user demand scaling, heterogeneous user base and distributed data requirements in terms of computing models and technology. How much compute/data power is needed here? How much distribution due to politics/licenses? How many users will use the system at the same time? What processes/tasks will each user want to fulfill? We will advocate (possibly conflicting solutions), but the project must choose. Perhaps an ICT board is required to influence decision making.

Openness

Openness of everything (not just code!) SHAML schemas, Postgress DB schema, tutorials on use, minutes, etc. in a well-organised place. (now there are multiple places, wiki, file server, svn, trac, etc.). Especially open services to the outside world. Look at Biocatalogue for

inspiration

Notes area

Working closer with the community

Report on the use of existing apps in the community and working processes and decide which ones must and can realistically be supported in GEM. The final presentation was just that! “I use this Matlab toolbox regurarly to do task X and I want it supported in a web-based environment so that more people can do it”.

Real agile development that incorporates actual users. Build feature-light apps quickly using appropriate tools (CLIs to services, portal development tools) then keep have weekly feedback cycles; travel around! Visit users and ICT experts. Get more embedded.

Identify the most important aspects to be addressed in ICT (pursuing use cases) and then hire the appropriate **dedicated** team manager to drive the development and especially the agile process

More specific recommendations

Recommend to choose one computational model, now OpenSHA works with Condor/Globus, risk engine uses multithreading. What is the scaling required and what is the best model, e.g., HPC-like parallel messaging, Grid/Cloud-like distributed computing, GPGPUs, data-optimised architectures.

Recommend to use OGSA-DAI in the systems architecture for data access (and integration)! Can solve the data size scaling problem. Also a problem exists when trying to use WS in workflows as they overwhelm the workflow engine (as they have the pattern of query/answer only). Last, a solution for a federated data model is needed, which OGSA-DAI offers in the form of distributed query processing.

Recommend to look at Web Services Resources Framework to deal with the type of services on offer here. Many open-source solutions out there that support it (e.g., Globus,

Recommend to look at frameworks that turn command-line programs into web services. Many are around. This could speed up the production of web services. Also, make use of tools to build portlets rather than by manually programming (e.g., Rapid was used for RapidSeis)

Recommend to open the services to the outside. That is the academic dream of service-oriented. Look at MyGRID’s Biocatalogue for example. By building a small community of developers that build modular services, they can then serve a large community of service users.

Recommend to look as Shibboleth for federated authentication and

Notes area

authorisations.

Recommend to use quick-win rapid-prototyping tools to build prototypes that are then evaluated by **REAL** end-users (agile programming). This requires a good definition of the these users are and a realistic view on whether these users can be drawn into the agile development process.

Recommend to look into column-store databases to deal with the large amount of data from models, e.g., MonetDB has OpenGIS included. It is unlikely Postgress can handle the amount of data expected. Building a home-grown balanced set of databases is a nightmare.

Recommend a more open and coordinated effort of the development. Not only for the source code, but for all artifacts, e.g., database schema, postgress sql code, wsdl files, XML definitions of SHAML, QuakeML, etc. All these should be put somewhere that is easy to navigate by external parties to see a) GEM is very active in certain directions and b) allow feedback on all artifiacts (e.g., the database schema misses this field).

Recommend unit testing all over the project's code base. Codes (portal, portlets, DB-code, OpenSHA, OpenGEM, etc.) are now loosely coupled (even in different svn modules), so how are the processes tested of the integration of these codes?

Notes on the fly

09:30–09:45 (Dominico)
- Seismology funded by Oil exploration, nuclear detection and hazard risk (building and social).
- Distance within the community, seismic hazard (including theoretical physics) to socio-economic impact (world bank)
- GEM1 is the result of the first 15 month of GEM, 3 months of discussion + 12 months implementation
- "Dealing with 1000s of databases all over the world"

09:45–10:30 (Helen Crowley, Marco Pagani)
- Model for risk and hazard should be developed and accessed via a web-based user interface (portal). Will use a number of databases
? How are these databases accessed? Standards?
? What does building a model mean
- Model builders: current focus = global component and regional programmes
- Mode users: current focus hazard and risk experts
- Federated database model (to allow ownership)
- IPR issues (for example when using existing software and services, e.g., Google's geocodes)
- Regional programmes can access the computing infrastructure at GEM Model Facility

Notes area

? Who pays for this? What infrastructure will be provided?

10:30–13:00 (Andrea Cerisara)
- engine is a loop with 'listeners', listeners are communicating via 'pipes', what is a pipe?
- Python or Java?
- SOA useful here? Not looked at yet. Hard to decide as it is not clear how tight the components fit together.
- Make sure to discourage WSDL and encourage OGSA standards, also mention OGC standards, implementations exist
- computation can take longs (weeks?), so push scientific portals hard
- service level agreements and queueing systems? What about emergency computing and violating SLAs?
- should create service (computer) interface and web-based user interface.

14:00–15:00 (? San Diego SC)
- unclear what the system/model/comp.engine architecture is, so what would be the best ICT architecture is?

15:00–16:00 (Rui Pinho)
- IPR and responsibility
- data and software licensing should be separately discussed (but are linked!)
- differentiate between GEM validated data and non-validated data created with GEM tools.

16:00-17: (Philipp Kästi)
- Lots of use cases for different groups
? Some portlets exist, but what cases do these serve?
- SHARE project on Seismic Hazard http://www.share-eu.org/ is co-resourcing the effort in the Hazard direction
! development of service may not be the forteit of the community!

09:00–10:00 (Moemke)
- http://gemwiki.ethz.ch/wiki/doku.php
- database design
- Is Postgres able to handle this amount of data? Why not go column-store? Problematic with GIS-aspects

10:30–11:30
- Web service architecture
? Why go this route if you are not planning to make these services public? Makes extra overhead and potential problems

11:30–13:00
- Existing solutions

14:00–17:00
- More portlets.

Notes area

- Portal framework is the typical framework used in scientific computing communities

JoshNotes

Objective: Avoid D ^{3} – "Death, Destruction, Downtime"

Software / Architecture:

- Consider Concurrent Collections (Intel) for engine internals.
- Consider storing decision trees internally in B-trees (ala partial MapReduce in CouchDB)
- Consider KVS with MapReduce
- In similar fashion, invalidate computed products when new source data is available (ala Make/SCons), using hash of input files
- Support annotation of data, controllers, etc. (Meta-data as a first-class object).

Scheduling / Batch Queue

- Make sure jobs run with priority

Notes area

Auth and Identity

- Consider SAML / OAuth / OpenID for auth and tokens, with (LDAP or another federated scheme)
- Define Admin as a first-class user type
- Require API keys for developed applications (for quota and priority enforcement, if required).

Open Source Community Management:

- Consider Canonical-style developer's summits (UDS)
- Keep community collaboration tools limited in scope to encourage rich discussion

Requirements and Users

- Consider a modern browser (HTML5 / WebSockets) as a minimum requirement
- Draw a clear line between what is "configuration" (what format should be applied for which region), and what is "code" (the parsing logic for a particular format).

Metadata, Attribution, and Validation

- System configuration needs to be baked into all system output.
- GEM needs to "sign" validated and authoritative data products. Don't do this in the data product itself, do it in a separate metadata document (referencing the data by SHA), and sign it with PPK.

Nits (small comments)

- Make sure ulimit is running on all the compute nodes (limit users from crashing nodes by running out of ram)

Testing and Validation

- Use Grinder (or something similar) to load test, at least ad-hoc if not within a CI system.

Referenced Architectures

- NERIES

Notes area

- ORCHESTRA
- QUAKEML
- GML

Existing Hardware

- Four – Sun Fire X4600

LinusNotes

It is quite clear that a lot of good work has been done, even in the stated absence of requirements. My over-arching comment is that there needs to be a clear, consistent, and shared vision of the fundamental purpose, and its supporting requirements and use cases, of the GEM project.

It is good to see management buy-in to the notion of a true prototype development, although perhaps it was a bit heavy-weight for a true prototype. The willingness to throw it away is commendable, although you should not necessarily throw it all away just because. Take what works, and be willing to throw what has proved unworkable.

There was considerable discussion among the reviewers about the technology choice for the web interface, and very little on the underlying core systems. This was unfortunate, as this core is a vitally important component. If the core abilities to provide hazard and risk calculations and to include as appropriate the regional component contributions are not adequately met, the project will not be a success, and no amount of User Interface dressing will conceal that.

It appears that there could be fairly divergent requirements and UI needs for the different categories of users. It may be that different user classes will need their own largely distinct interfaces. Whether this can be achieved within a single UI architecture and technology choice is something to bear in mind. This does, however, increase the importance of a well designed core and API.

It is an intriguing idea to enable and capture discussion and input from expert users, but I think that the notion that you will gather significant and reliable information from the average public is flawed. That comment is perhaps outside the scope of IT review, but I would encourage you not to make technology decisions driven by that particular target.

It was not clear to me how the regional components will interact with the GEM site / application suite. Are they to provide their own models? their own databases? web interfaces to their local implementations ? Are

Notes area

their data and/or model services or outputs to be integrated into the main GEM process chain? Is this to be an architecture of pluggable models and remote databases? This is very different from the capture of user comments and contributions.

There appears to be a lack of explicit requirements and use cases, at least as expressed in the presentations and discussions. You must understand your requirements as best you can, and the prioritize them to manage the incremental development of the system. The user needs, requirements, and use cases – derived from actual users and not speculated by the GEM team – should drive the technology and design decisions. It seems the biggest issues relate to the lack of a uniform, clearly understood and communicated – and shared – vision of the GEM project interface requirements and use cases.

Along the same lines, it is important to understand your scalability requirements. Is there a real need for HPC, and if so, in what form? However, that being said, it is also import to not spend ages developing these requirements, especially if that occurs in a vacuum devoid of user feedback. And it is very hard to get reliable user feedback without examples to discuss. Thus the agile/iterative development approach of quick development cycles and mandatory user feedback.

Although I cannot gauge what level of user contributors there would be to GEM as an open-source project, as was mentioned in the meeting even without such a significant contributing user base there are advantages to managing the project in a FOSS manner, particularly in light of the distributed development environment. I also think, if you do have a larger number of contributors, it will require greater diligence in vetting/verifying the software.

There are certainly use cases that would be well served by semi-interactive, user-based contributions such as discussion threads and comments, and similar sorts of things, I don’t believe that any of the proposed users are going to have a need for a full-blown “social site.” The true need for such social elements must be clearly understood and real before you embark down the path of developing a site based on social networking components. It is true there are a lot of available components, and as demonstrated, it is possible to quickly put together something that looks flashy and appealing. But will it serve the core user base?

This may have been covered in the database schema that was presented, I’m not sure, but I would like to emphasize the need, particularly for GEM, to reliably and definitively track the provenance metadata for all generated products and artifacts. If you intend to provide “officially sanctioned” GEM products, I think it would be prudent to also consider watermarking and other sorts of authentication and validation techniques.

Notes area

PhilNotes

Recommendations for GEM IT Development (July 2010)
from Philip Maechling (maechlin (at) scec.org)

I believe GEM wants GEM IT to perform SHA and risk calculations on a global scale so the following recommendations emphasize what I believe will help GEM IT do these types of rigorous scientific and engineering calculations.

I think GEM IT should specialize in using existing scientific SHA and engineering risk codes and in integrating those codes so that they work together. I believe it is the responsibility of GEM IT to show that GEM software can perform scientifically defensible calculations. I believe GEM IT has to lead the way within GEM on scientific rigor in computing.

From the GEM IT review meeting, it looks to me like GEM has assembled a team with the skills needed to run rigorous scientific calculations. You have scientists, engineers, and software people. The key now is to get the three groups (SHA, risk, and IT) to work on a real science calculation. GEM IT must show that its software and hardware and data sets can produce research results in both SHA and risk.

I think GEM IT should specialize in calculating scientifically valid results with complex scientific and engineering software. The GEM IT group should put the GEM science codes under test, with reference inputs, automated test suite execution, and reference results for each science code. I would not make open-source, or social networking, or portlet development, or xml standards become the main expertise of GEM IT. These are all worthwhile activities, however, these may be what external groups can contribute, or activities that your group focuses on after you have completed significant calculations with the system.

Establishing a core IT team competency running scientific codes can be difficult for many reasons. Science code is often legacy code written in Fortran often integrated with other codes through awk and sed scripts. Many developers want to work on the newest technologies, and scientific codes are rarely based on newest technologies. Scientific codes written by earth science grad students are often difficult for software engineers to understand, run, and modify. Software developers often would rather write their own software than run someone else. Scientific codes tend to quickly become unusable, often due to lack of support and documentation, unless they are instrumented with a re-usable set of test problems and reference results that form acceptance, or regression, tests for the codes.

As soon as possible, GEM science and engineering groups should

Notes area

define baseline or reference SHA and risk calculations that can be started immediately and can be done at publication quality. This could be the Global Seismic Hazard Map you have shown. The baseline research calculation you select should produce a data product that has multi-year value.

Once you have selected the target calculation, and the SHA and risk groups have defined what calculations and data processing needs to be done, then you should let the GEM IT group figure out how to do the calculations. I recommend minimizing the number of scientific codes used and minimizing the modifications made to the codes. It is the responsibility of GEM IT to figure out what codes need to be re-written, what new data stores and formats must be introduced, to perform the reference calculation. GEM IT must show their engineering judgment designing the reference calculation for correctness, repeatability, and for minimum time to solution. GEM science and engineering groups are expected to defend the calculation in public so they must be closely involved and must have final approval on calculation results.

For this first research, let your GEM IT people in close consultation with the GEM science people, do the calculations the most efficient way they can. You should drop all the web services, and portal, xml, interoperability, and outreach IT requirements for this first baseline calculation. To be clear, I am saying this reference GEM calculation may be run from a command line not through a portal or a web service.

On this first reference calculation, two things should be required from the GEM IT group: 1) they need to automate the calculation, and 2) they need to show the calculation is repeatable.

Once the GEM groups all agree the calculation is done, return to the portal, and web services, and ask your IT group to deliver the research data products from this reference calculation through your portal, or through web services, in xml format. GEM IT can use, or add to their current design, based on actual research data.

I believe that a focus on automating large-scale SHA and risk calculations will generate some GEM IT design changes. In the following sections, I outline some more technical comments on technology that I think will be needed for GEM IT.

It is my view that the key to large science calculation is scripting, or automating, long series of calculations. What GEM IT should construct is a system that can call a series of separate scientific codes, one after another, until you get a final research result. By automating calculations, it is possible to achieve the reliable repeatability of complex calculations.

I believe the GEM idea of a computational engine is to run significant SHA and risk calculations. If this is true, then I believe GEM IT should refine your computational engine ideas. First, the computation engine

Notes area

concept of listeners should be replaced with the concept of scientific workflows. Second, the computational engine concepts of event loops and threads should be replaced with an interface to a job scheduler like Condor, PBS, or Maui.

Scientific workflows are sets of software programs that have data dependencies. This technology enables you to script multi-stage calculations that run separate scientific programs one after another.

Programs, scripts, and users on the command line should be able to submit program, or workflows, to a GEM job scheduler, and check on the status of their jobs. If you base the computational engine on a job scheduler, you can use this design with any scale computational resources, from local servers, to cloud computing environments, to remote supercomputers. This approach does have an impact on your scientific software development. The GEM IT group will need to focus on getting your key scientific codes to run in a form that can be submitted to a job scheduler, and run on a shared computer and disk system.

I recommend a principle of pushing web service interfaces out of your core calculations and minimizing both your web services interfaces and the number of xml formats used. Define research calculations that require multiple scientific codes. Let your science groups and IT groups get these scientific calculations running through scripts and from the command line as scientific workflows. Let the scientific programs do their calculations with minimal modification and let them output their results in their preferred format. Write file format converter programs if needed.

Preparing the GEM system to run these large research calculations will force changes onto the current GEM software architecture. I believe the GEM IT team is strong enough to make any necessary changes needed to support these calculations. Once you have the calculation going, if needed, you could implement web services to invoke the calculation and to retrieve a research result, in which case the only XML formats needed define a calculation and deliver a final data product. Portal development should focus on providing user easy access to the data products from this GEM calculation.

I believe that scientific open source software development always involves distribution of both source code and data. Contributions to GEM should always provide both code and data. Data contributions should provide code to extract example data sets, or implement example queries. Code contributions should provide reference input data and expected results data sets.

GEM scientists and GEM IT group can model your calculations conceptually as a series of programs that use file-based inputs and produce file-based outputs. Using this model, your groups can quickly model complex calculations. Then workflows can be built to automate

Notes area

the running of extended calculations. Script these up and get them to work on the systems.

I am convinced that the current impulse to integrate codes through web services will prove much too slow to start. I recommend learning to run the key scientific codes with minimum modifications and exchanging data between programs using existing output formats and format converter programs where necessary.

The computational engine interface should be modeled as submitting a job to a job scheduler. Job schedulers provide users with standard commands to run a job, to cancel a job, to check on status of a job.

I have limited advice for GEM on open-source licenses, portlets, xml web services, and social network sites except don't focus the GEM IT development efforts on these things. These issues will become important later in GEM system development. After GEM IT is producing significant scientific results, you can return to development of these items. I believe interoperability and data access capabilities have value only after the GEM IT systems is producing rigorous and useful SHA and risk results.

ShoaibNotes

Options for handling scalability and performance in large volume databases

It is important to explore creative and elegant solutions

Trying out the QuakeML (without XML)

URL: http://www.seismicportal.eu/services/event?info

Resource URL (time dependent):
http://www.seismicportal.eu/services/event/latest?num=2&format=json

```
quake =
JSON.parse(open('http://www.seismicportal.eu/services
/event/latest?num=2&format=json').first)
pp quake
{"nbEvents"=>2,
 "unids"=>
  [{"lon"=>"24.89",
    "flynn_region"=>"AEGEAN SEA",
```

Notes area


```
    "unid"=>"20100629_0000012",
    "auth"=>"NOA",
    "magtype"=>"ml",
    "depth"=>"140",
    "mag"=>"2.5",
    "lat"=>"38.99",
    "datetime"=>"2010-06-29T07:51:55Z",
    "orid"=>550286},
   {"lon"=>"22.64",
    "flynn_region"=>"SOUTHERN GREECE",
    "unid"=>"20100629_0000010",
    "auth"=>"NOA",
    "magtype"=>"ml",
    "depth"=>"10",
    "mag"=>"2.1",
    "lat"=>"36.16",
    "datetime"=>"2010-06-29T05:24:11Z",
    "orid"=>550268}]}
```


SteveNotes

These notes are more about internal practices rather than external (FOSS) practices per se, although any internal process should still make use of some of the same FOSS community tools (perhaps with a different front-end) as well as the ones listed here, and more (eg, ReviewBoard). Internal practices are important for a number of reasons, some of which are illustrated below and in the additional links.

For specific thoughts on personal/group software process and some of the associated free tools, see StaticCodeAnalysis and the areas of emphasis in AgileVsWaterfall (as well as the SlideShare thing). Whether agile or traditional, modern software engineering really does require at least some kind of basic personal/team processes in order to be both successful and repeatable (see the Watts Humphrey PSP reference on IT Review or Section 4 References and then keep going). They can be very simple (as described above) and easily documented, and are well-supported by the available tools. Coding styles can be easily enforced via custom rule-sets and the example analysis tools. Documenting, following, and improving your processes will not only get you a high CMMI rating, but will also greatly increase your chances at making a successful delivery. Keeping a good “team-player” attitude with respect to such things will also help a lot, i.e., the more you follow your own Doxygen/JavaDoc guidelines, the better your documentation will be. Simple, no?

Notes area

User stories or other clear definition of user requirements clearly needed, including involvement of real users (both technical and non-technical).

Think about design patterns and pre-existing solutions (but don't get carried away and implement every pattern under the sun).

Apply static code analysis tools (eg, FindBugs and PMD for Java code) to your development and V&V processes (regardless of the chosen methodology) and address the important issues before you move on (including justification for **not** fixing something). This can be done in various ways, e.g., by each developer and peer reviewer as they work, automated to run on each check-in of new code, or incorporated as part of the test & evaluation piece (or all of the above).

Use a proper license for code, data, and other artifacts, optimally one that supports the vision of open and transparent (preferably one that can enforce these things, but the specific license "philosophy" should at least be compatible with the stated GEM philosophy).

If it makes sense, steal directly from NERIES, etc.

Data/model validation tools are needed, ie, ensuring the "correctness" of vector topologies, hazard and risk algorithms etc.

During the presentations, we heard about an emphasis on testing, but saw very little in the way of documented requirements (sort of shoots itself in the foot). Must at least document some basic functional and high-level system requirements (or user stories) which should drive system test and evaluation. The goals of scientific verification would seem to point to a more detailed approach to requirements engineering.

Multiple languages are okay, especially bindings/wrappers for the (hopefully) CLI processing tools/libraries. Python is generally a much more productive and flexible language to work in, where both Object Oriented and procedural styles are well-supported, and it becomes almost trivially easy to write useful code. There are several good free books, a helpful developer/user community, and lots of re-usable python code, recipes, etc. I would definitely support moving largely to Python (and away from Java) due to the many development, runtime, and deployment issues associated with the latter. That said, as long as you're working with Java, please do define some appropriate coding guidelines, and follow the StaticCodeAnalysis suggestions (and don't forget to listen to what the tools are telling you).

Why Python? Maybe this brief explanation will help… Although a few of the points are (literally) inside jokes, most of this is useful if read carefully.

Why Ada? If it wasn't for the screaming, I might even recommend Ada for the underlying processing tools. Joachim Schüeth recently won the

Notes area

National Museum of Computing's Colossus Cipher Challenge using Ada, so maybe it's worth a look (the article is a good introduction to why).

As Frank suggested, storing large data objects (eg, raster files) on disk rather than in a database is probably the best approach from an overall data management and performance standpoint. Appropriate filesystem tuning for large/sparse files (and reasonably fast hardware disk I/O) should provide reasonable performance. For modern Linux systems using extfs, check which "types" are defined (for the -T command) in mke2fs.conf and make sure sparse_super is enabled when creating the filesystems. Loop-mounting a large sparse file for the dynamic file area (on the data server side) should provide an additional performance boost (any data files to be archived should be moved off of this mountpoint). In any case, make sure you have enough inodes if using an EXT filesystem.

Lastly, I would strongly support using Python and Django, as discussed in Ben's presentation, along with the GeoNode framework (there are good technical reasons that allign well with Stu's points below). Extending (or even replacing) the Python interface to Geoserver is probably well within the GEM project scope (if using both Django and Geoserver).

Stu's Notes

I'll keep it short. The Bank is making a investment around the GeoNode (geonode.org) as an system to manage country spatial data for disaster risk reduction. Further, many of our software developments for Probabilistic Risk Modeling are occurring within this framework. It is in the interest of the Bank to align our development efforts with GEM. The GeoNode is based on FOSS4G and django a widely used python web dev framework.

Timos Notes

Some personal comments concerning the GEM IT REVIEW by
Timo Multamäki (timo.multamaki@eigenor.com)
+358 40 5479523

First of all, it was positive, constructive and from my view point, needed, meeting. I also believe that it had results.

Comments below are not prioritized and some concern smaller

Notes area

nuances, where as some are concerning larger concepts. Bear with me :)

1) Quantifiable and prioritized high level specifications are entirely missing. Its very hard to say whether a project is successful or not, when you don't know which of the features are mandatory and which are just wishful thinking. These specifications should not go deep in to the technology, but be more on feature level. This document should not be very long, either, probably 1-2 pages with something like 10-20 points, which are further explained in other documents. This is the document that board and management should understand what the project is actually doing and which are its task priorities.

Don't make all priorities to be 'high' or 'mandatory'. Be realistic. There must be differences between features, some are certainly more important than the others.

2) Social networking aspects are not defined. It was stated as 'important goal', but pretty much nothing points to that direction, currently.

- Requirements: What are the actual needs of GEM for social networking? Why? What kind of goals can be achieved by social networking? Once the goals are identified, it is within possibilities to find what is needed to achieve these goals. Make the goals quantifiable.
- Priority: These social networking features need also prioritization, not only between each other, but between all features.

3) Much of the IT work seems to be redundant from other projects. Search similar projects. Utilize all existing projects. Don't re-invent the wheel.

4) Computational engine development? There was very little about that and very likely its among the computationally heaviest tasks. Define API for engine development. Engines are certainly something that only very small minory are doing, but would benefit large part of the community, if the development actually takes place. Fact is that, if the system is too complex to use, it will not happen. Make sure, that you, internally, use the same API that you force the community to use.

5) Purchase hardware as late as possible. Spending money for HW when user base is zero doesn't make sense. Current needs can be met with very modest computer power.

6) Using a single user interface (portal) as it more complex, cumbersome, often cluttered solution. First develop an open API for all kinds of user interfaces. Allow everyone to build an interface which they like. Make the 'portal' as demonstration UI that is easy to reach and can

Notes area

be extended by varying user groups to their liking.

This allows command line support and makes the further development much more flexible.

7) Security aspects. Some (most likely national institutes) will eventually see especially the building inventories and structural datas as a possible threat. There is not that much that you can help that with the exception of being aware and writing some sort of non-liability clause (eg. that GEM is not responsible of the end products that end-users are doing with the data derived from GEM systems).

8) Privacy issues. Just like google streetview run in to troubles, so can GEM as well in countries where privacy issues are high on the list (Northern Europe, at very least). These concerns have primarily effect in structure catalogue, which seems to be causing more trouble than it likely is worth. I'd recommend, especially in the beginning to develop the building models based on city block level information, where each separate house is not modeled. I would consider getting even such information with a city block granularity an unlikely case. This underlines the need for realistic default 'a priori' information of varying structures. It should be at very least defined in basis of country and building type (block house, small houses, shops, industrial complexes). Making these building types wide enough, it is possible to just draw varying building type areas and thus allocate the default values to areas. In many areas this data is already available on this level, as flood models are using similar information.

Notes area

4 - References and supporting materials

External reference materials

- Personal Software Process – PSP is a must-read.
- Doxygen Manual
- CCCC on Sourceforge and the associated PhD research
- Property Lists
- OpenCL
- Rover (Exposure Database via Crowdsourcing) – Unreleased
- Translucent LDAP Proxy
- XDB-IPG
- WebHooks
- PubSubHubbub
- QuakeML

RoundTable discussion (mostly unedited)

One-two minutes overview / summary:

Frank:

- Creative Data Management
  – Raster Formats
  – WCS Mechanism (for remote / gridded data sets)
- Processing Engine (3500 lines)
  – Doesn't distribute
  – Doesn't parallelize
  – Not easy to integrate other code (from other sources)
- DB schema seems fine
- LDAP seems fine
- SHAML

Shoab:

- Scalability issue is forefront
- Modelling output maybe shouldn't go into RDBMS
- Run some trials and tests on scalability (does it stand up):
  Spend a month running models into DB, see if it dies
  – Possibly test NoSQL / Schema

Notes area

Linus:

- Think about:
- Will RP be tightly coupled? E.g. unique content, unique modules.
- To what degree will the system be available for modification by outside partners
  – Pull in data from this into other systems, vice versa
- Granularity issues
- Don't go overboard, have every DB write go through a service call
- Consider different user use cases, should they have different interfaces:
  – General Pub one
  – Scientist one (close interaction with model), Insurance (Pick tech after)

Alessandro:

- Does the WS layer get opened to other users?
- If no, consider single-tier stack. Conversely, they don't want to throw away what they have (feeling),
  so consider what the learning curve is, how much of what they have is worth preserving?
- Divide the backend and frontend in discussion
- Perhaps new python frontend, leave WS Java backend
- Sustainability approach

Stuart Gill:

- Don't try and do this by themselves
- Identify existing projects, leverage them
- Walk through the agenda so we make sure we cover everything in our response
  (In our presentation as well as in the report)

Andreas Acheva:

- Gather developer community – the stack they have now is too strongly coupled, too complex, to make it easy for external contributors to get involved.
- Their WS with SOAP – reuse them with a simpler, RESTful interface
- Keep serverside and clientside separate
- They're afraid of abandoning portels, but longterm solution will definitely be to replace portlets with an existing portal solution (GeoNode or comparable)
- In transition, imbed portlets as iFrames (replace in the future)

Notes area

Chris Holmes:

- Web-centric framework for portal (python)
- Existing WS investment, add REST to SOAP
- Have an external group consuming that API, providing feedback
- Perhaps keep portlets around, if there are other groups that really want to use them
- Use pure JS widgets, wrap into portlets if needed (e.g. existing mapping components)
- Licensing – DB licensing, stamp for auth with key signing, make statement on openness for community-contributed DB (CC0)
- Software licensing – get CLAs, relicense if there's a reason to do it later, don't bother now (LGPL is good enough)
- Federation discussion
- People run it themselves but also, architecture of participation – social and tech
- User testing, user testing, iterative development, user testing (release early and often)
- Stop gathering reqs and making mocks – small, incremental releases, get actual use
- Open Dev sprints, open community govt
- Open SVN ASAP

Jocelyn Guilbert:

- Would like to clearly see the needs
- What were the previous tech (NERIES)
- No clear roadmap and milestones
- DB – need a tool for viz., quality checks on input
- Normalization of inputs (different tables and overlap, builtin sources and computed product)
- Scientists want to see verification of DB contents
- Organize workshop for different kinds of users (user testing, etc).

Heiner Igel:

- Large scale parallel computing simulations expertise
- Coding close to hardware question:
  – Orders of M away from needing GPU, etc.
  – Exploit embarassing parallelism
  – Don't get involved in hardware-close development (yet), remain open for that later
- No regression testing?
- Need benchmarks
- Need a strategy for testing (scenarios, validation)
- Outside IT training and documentation

Notes area

- Reproducability of results (which models, which data was used)
- Italian legal case (manslaughter)
- Standalone is an important question (increases complexity enormously) – discuss?

Jessy Cowan-Sharp:

- Participatory Elements in Architecture
  – General Architecture: not that well understood (by the dev team)
  – Expose regional programme, other types of risks and hazard – not well thought out yet
- Review of how the layers are coupled – simplify
- API endpoint decisions – understand the implications
- Incorporate this into layering, dogfood them
- Expose these to community – let comm. build different front ends
  – Mobile phone apps vs. scientist interface
- Do some UI mockups, consider some full workflows
- Tools for FOSS development (github, twitter)
  – What it means to engage with the dev community
- Bigger emphasis on testing and benchmarking, test to fail (on different tools)

Timo:

- Quantifiable and Prioritized Specs
- Everything was equally important (doesn't work in real life)
- Social networking aspect is really important, got only two minutes of discussion
  – importance hasn't been explained
  – reqs haven't been explained
  – this needs specification
- IT work is redundant to other european science projects
- Bits and pieces could be aquired for free, should be looking for these
- Now thinking of buying new hardware (with no users)
- Buy hardware when you need it

Kevin Milner:

- DB scalability issues
- Spec'd out all this stuff, haven't connected any of it yet
- Put 4M hazard curves into DB (use fakes if needed)
- Needs to get used and tested
- Portlets concern me
  – Scientists and insurance guys will use it no matter what

Notes area

– engaging the broader community don't want clunky, slow, frustrating
- Switch tech if necessary

Jano:

- Data scaling, distributed issues
- We will recommend models, might have conflicting advice
- Need openness of everything:
  – SHAML schema (can email you)
- Too many places,
- Bioinfo. services catalog
- Report on existing apps
- Mimick the working ones
- Real agile development, uses ACTUAL users
- Appropriate tools
- Quick feedback cycles
- Come see the experts if necessary
- Big fan of JAVA portal framework, lots of tools they're not using
- Already being used in NERIES
- ICT space – hire the right team manager to manage dev, and the agile process (focused goal)
- Look at the report, then find the right team
  (Chris agrees, its the most important thing)

Steve:

- Lack of specific requirements
  – Need functional requirements, didn't see docs
  – Have vague ideas in some areas
- Don't need multiple volumes of reqs, but have something to test against
- Hire someone with good free software exp., as well as managed project experience
- ARCHITECTURE:
  – Saying agile, did not see arch that looked like agile
  – Ben's presentation was the best of the bunch (vis-a-vis agile)
- Doing test driven dev. without something to test against
- Some tools and tech aren't useful without knowing how to use them
- Right tools would help, if they use the right process with them
- Understand more of what agile really means, and how to support them from mgmt side
- Understand reqs.
- if they have JAVA people, that's not inherently wrong – flexibility is key, really understand what they're trying to do and using the tools appropriate for the job at hand.

Notes area

Phil:

- Is it in trouble? (Not sure) Success is vague (in this type of project).
- Do you have use cases? Some said yes, some said no.
- Pointing to global hazard map as an end product, not sure it was produced within a single system. (Was patched together).
- Don't represent it as a system product if it was manually constructed.
- Find something that's useful, help someone, somehow. Find 1-2 things that provide value, hang project on it.
- Will wander otherwise.
- System should do verification testing, results should go to people who could confirm them. (Didn't hear this restated).
- Think the WS are a big mistake in this case. (Objects are too big to represent in ASCII). Esp. internally.
- Didn't like the comp. engine software stack. (Thread layers seems wrong.) Consider PubSub message passing instead.

GOOD:

There is plenty in GEM project which is good and commendable. We have listed couple of notes, where
be jointly agreed an above average performance.

Looking help from the outside (us) – in a formal way:

- Putting effort to get expert involvement. (Doesn't happen in most of the other European projects.)
- There is a will to understand the best approaches within GEM.
- The GEM secretariat has managed to find a good cross-section of different backgrounds for the technical reviewers
- Despite having vague requirements, they GEM1 has already shown that GEM is capable of making results and not just idle talk
- Future-looking at an unprecedented scale. (Great potential)

Openess:

- Made a lot of good decisions vis-a-vis being open. Want the code to be FOSS. Thought about licenses. Open to criticism and feedback.
- GEM is a learning organization and it shows. Not afraid to question each other, try to do better the next time. (Vis-a-vis Ben's prototypes)
- WILLINGNESS to throw out the "built one to throw away"
- Hazard side really came together (around openSHA) with very little IT support. (Made good use of OpenSHA external team).
- Trying to do something great, haven't collapsed under their own

Notes area

ambitions.

- Willing to expose themselves for criticism (very atypical).

Other issues:

- Funding is appropriately arranged, which is rather rare in scientific related large projects. As project is not only getting funds from a single source, the probability of actually getting commonly accepted results is vastly larger.
  What they did accomplish (SHAML spec, etc) is very impressive.

Productivity:

- Produced the global hazard map.
- Doing int'l distributed software dev.
- GC and RP organization seems well defined.

JOSH:
Don't make arch spec part of vision statement,.

(FIND POSITIVE COMMENTS)

Agenda Review

we haven't made any comments on formats
- pushing forward shaml, developing quakeml – good
- some thought binary format was needed, others didn't

HPC – should they consider using it?
– there are easy wins in focusing on the massively/embarrassingly parallel techniques
– not so much focus on HPC but definitely need to think about high throughput and parallelization techniques.
– don't get distracted building a cluster— use existing cloud options (EU scientific computing clouds, ec2, european grids, etc.)

Continue to have IT Reviews
Do some smaller and more focused reviews

Leverage this team and others as an IT advisory board.

Code, repos – centralize, simplify.

Naming conventions for different components is confusing. gem and opengem are even confusing

there were some feedback from scientists that the schema didn't accurately capture hazard objects they were using.

Notes area

WednesdayAgenda

Breakout Session?

- GIS experts
- IT-sec

Overall Architecture Review

- quick component review
- review/add to open issues/questions

Scale

- Estimates on user count, data size, concurrency and processing complexity
- Sharding /

GIS Thoughts

- Breakout session with GIS experts
- options for what and how to store
- options for what and how to render (polygons vs points, raster data)

IT Security Concerns

- Signing / PKI infrastructure
- Federated Auth / SSO approach
- Access layer implementations

Holy Wars

- JAVA and/or Python
- Web Service format(s): SOAP vs. REST, XML vs. JSON
- RDBMS and/or NoSQL / DocumentDB
- Appropriate Development Methodology – “JAVA methodology” vs. Agile

Notes area

RESTian interface proposals (if any)

Distributed Application Questions

- Registry – IF-MAP, others
- SSO – SAML, OpenID, OAuth
- a single server with browser access?
- include ability to run a local client?
    - running local client on local and remote data
    - distribute as configurable packages that can be installed and run anywhere (in theory)
    - expose APIs and let users build custom apps/interfaces as desired

Database decisions

- centralized, distributed and/or sharding options
- combine relational with nosql?

Front-End

- architecture (portlets, non-portlets)
- social media tools/focus
- primary UI style (dashboard, user-types, etc.)
- user-facing features

API

- at what levels and to what services should APIs be available?
- what kind of consumers are expected/should be supported?
- does this affect architecture (eat your own dogfood)

Requirements Document

- should we try to assemble one based on our understanding?

Actors

- New roles
    - Admin role
    - Developer role?
    - Individual/social media user