Interview with Paul Ramsey
Vice President, Product Management, Boundless
http://cleverelephant.ca
https://www.linkedin.com/profile/view?id=318979
Interviewed January, 2015
Interviewer: Bill Dollins

1. ***Please describe your background in terms of education, training, professional experience, and roles. How has your background informed your leadership style?***

   I completed degrees in mathematics and statistics at universities in British Columbia, but with no particular goal in mind: mathematics was just the undergraduate subject I was best at; statistics was an interesting practical field of applied mathematics, when it became clear that pure mathematics was well beyond my capacities.

   One accidental side effect of taking the graduate degree in statistics was gaining experience with UNIX computer systems, as the program used a lab full of HP/UX Apollo workstations to run the S-Plus statistical programming languages (the makers of the open source R statistics package copied the syntax of S-Plus, so my graduate training has not yet completely faded to irrelevance). I had spend my undergraduate years running DOS and Windows computers, but only as an end user: I wrote essays, played games, did some simple Pascal programming for introductory computer science courses, but nothing more. With the HP/UX systems and S-Plus a wider door seemed open. There were add-on modules for S-Plus I needed (for generalized linear models) and using them required building the source code. So I learned enough basic system administration to build out the modules.

   At around the same time, I discovered Linux (at the urging my wife, who wanted to use it to run GRASS, for her GIS education) which brought the power of UNIX home to my living room. Unlike DOS/Windows, it was infinitely configurable and could do things over a network like connect directly to the internet over a basic terminal dial-up connection (see http://en.wikipedia.org/wiki/Serial_Line_Internet_Protocol). This was pretty magical and very DIY, and I got heavily hooked. Between doing my thesis work computing and my home experiments with internet networking, I had picked up enough UNIX to be pretty dangerous.

   I got a third and final piece of luck in my first job out of school: working for the Ministry of Aboriginal Affairs helping with treaty data analysis. I got the job because I good with numbers, but I kept it because I was good at computers. And fortunately the Ministry had some very good, interesting computers: Solaris workstations for GIS data crunching. All my previous UNIX skills provided a basis for being very competent with these machines in a way none of the other staff were, so I could do cool analysis and data manipulation that helped with my confidence and my internal profile.

   Again, the UNIX philosophy of building up complex solutions from small tools served me well. I build small web applications with perl, and tied together data sources using perl and FME to automate large conversion jobs. What worked in my experience, every time, was integrations quickly realized with open source UNIX tools. Meanwhile, the line staff in the Ministry sweated out hand-building maps using CAD tools like Microstation. Software bought off-the-shelf always seemed very limited in scope, and (worse) hard to integrate with other data flows. Based on this early experience, I began to grow some technology biases.

At the Ministry I was fortunate in another way, in that my Director was adamantly anti-ESRI. So our tools were a weird collection of things -- FME, a locally-made GIS called "Facet", Microstation for cartography -- but no Arc/INFO, that was a "rip-off". I was forced to learn how to solve analysis problems using combinations of tools, and got used to the idea that solutions involve wiring together parts. If I had Arc/INFO at hand, it's likely I would have spent a lot more time "on the reservation", since that particular product was both full-featured and highly scriptable -- it would have been catnip to a prosumer like myself.

As my contract with the Ministry wound down, I found new opportunities working for the local GIS company "Facet", so many that I had enough work to hire an associate for my (then brand new) company. What I looked for was naturally what I had become -- a generalist with a hard sciences background, but not necessarily a computer scientist, someone who could quickly pick up the many and varied tools that were necessary to make a "Facet" GIS analysis sing.

From that first hire, Refractions Research grew as a generalist GIS firm with a specialization in "all things not ESRI". While that ruled out vast swathes of government business, it was at least a narrow niche within which competition was relatively limited.

As with a lot of small companies, staff with the ability to work independently on the whole problem, generalists, were valued more highly than specialists. My ability to grow the company depended a lot on being able to both characterize and solve technical problems, but also to communicate with the customers in non-technical ways. In consulting, there's always room for people who can bridge the gap between worlds and turn a customer's vision into a programmer's high-level architecture.

Being able to write effectively and understand code was very useful in the open source world, and even when I was almost 100% involved in management and sales, I continued to read code in all the languages our company produced it. I couldn't write software, but by skimming through code and reading documentation, I retained enough information to piece together the architectural solutions we needed to sell.

2. ***When did the PostGIS project begin and what was the motivation for it?***

PostGIS began in 2001, while Refractions were still engaged at the Ministry of Aboriginal Affairs. We did treaty negotiations support, mostly analysis, and as a result had to manage a lot of data. The analysis was still done in the "Facet" GIS system, but being able to reproduce and iterate models was getting unwieldy as the number of analyses being requested grew.

http://refractions.net/products/postgis/history/

We had been working with PostgreSQL already, as a database behind another Facet project we worked on for watershed analysis, so we were familiar with it, but only as a tabular database. Storing spatial data was a natural thing to want to add, since we were storing all kinds of non-spatial data, and even storing spatial data as blobs to be read by the Facet system.

The OpenGIS "Simple Features for SQL" specification had been released only a couple years earlier, and it included guidance for how to store spatial data in a relational database, three methods: in relational tables (shred the geometry into records of the form "x1,y1,x2,y2", etc); in blobs; and in "objects". Since the first two could be tested without any new database code, we tested them both right away, using the "JShape" applet to visualize and time the result. They were both really slow.

By this time, we had hired a staff member with an actual computer science background, and he was not intimidated by PostgreSQL internals, so he decided to try the third option, using the PostgreSQL type-extension APIs. If PostgreSQL had not been built with an original design that made type extension relatively easy, we probably would never have done it. But it was, and we did. This time, the result was really fast. We could store large data sets and retrieve subsets in milliseconds.

By now I had been doing consulting for over four years, and built up a small company, and really come to love all the open source tools that made it so easy to do magical things without having to go begging for licenses. I wanted to be one of the cool kids too, and being a cool kid did not mean "try to sell things" because selling was pretty boring and objectionable, it meant "show off your really cool tools": so PostGIS was released a few months after the first tests, as a 0.1 release with basic spatial object and indexes and a handful of functions.

It was a smash hit.

3. ***What was your involvement with open source prior to PostGIS?***

Amazingly, I had almost no involvement whatsoever prior to PostGIS. No community involvement at any rate. I had used open source tools, open source operating systems, and of course PostgreSQL, but I had not engaged with any of the communities, not even in the projects (GDAL, MapServer) that already were up and running in the open source GIS arena.

Once PostGIS was up and running, a natural thing to do was to try and render pictures from it, and we quickly discovered the MapServer project and added a PostGIS driver to it. And since MapServer was already a moderately sized community, experiencing the engagement and feedback from the other community members was really bracing: here was a whole bunch of people who felt more or less exactly the same as I did about software and tools. It turned out, I wasn't a space alien after all. (see Eric Raymond, Cathedral and the Bazaar)

4. ***PostGIS extends another open source platform (the PostgreSQL relational database). What were the motivations for making PostGIS open source itself?***

   After the initial development, we had a little chunk of code. It did some modest stuff in a database with spatial objects, but not a lot. We had used some spare time in the consulting schedule to get this far, but consulting work was going to start up again, and PostGIS wasn't going to pay the bills in the state it was in.

   As an add-on to the already-free PostgreSQL database, PostGIS didn't stand a lot of chance as a proprietary pay-only product. The only people using PostgreSQL at the time were folks who couldn't afford Oracle (in fact, to this day PostgreSQL remains a narrow niche product compared to the alternatives, despite its manifest superiority in so many ways). It's unlikely many of the PostgreSQL user base would buck up for PostGIS. In later years, another local company in Victoria attempted to commercialize some of their PostgreSQL add-on (actually ports of Informix add-ons to PostgreSQL) with little success -- mostly they just succeeded in continuing to support their existing Informix customers as they migrated over to PostgreSQL.

   We could have kept PostGIS in-house, as a consulting differentiator, but since we were only addressing a very small market at that point, having a "custom in-house spatial database" wasn't exactly going to be a selling point (more like the opposite, really).

   That left either letting it die, or sharing it with the world. And sharing with the world had a salutary extra effect in that it allowed us to interact with people who otherwise would have no reason to speak with a small consulting company on an island off the coast of British Columbia. Within the first couple months after release, I'd exchanged e-mails with folks in Fulton County, Georgia, at the City of Boston (with Regina Obe, who is herself a major PostGIS contributor now) and at the LA Unified School District, which led to our first paid PostGIS consulting gig a few months later.

   Retrospectively, looking at the considerable success of PostGIS, it's tempting to say we left a lot of money on the table by open sourcing it, but there's a serious chicken/egg problem with that analysis. The success of PostGIS was very much bound up with the free-and-open-source nature of it. As a proprietary add-on to PostgreSQL, it would have been quickly supplanted by some PostOther, either another extension like ours, or just enhancement of the in-built geometric types that already existed (and still do) in PostgreSQL.

   Understanding software as something that is built more often than sold is useful for understanding the economics of open source (see Eric Raymond, The Magic Cauldron).

5. ***How was the PostGIS project organized at the outset and over its first two years?***

   We started as a "benevolent dictator" model, with myself and whomever at the company was doing PostGIS development being the dictators. About 95% of the development was being done by the company, and that remained true for the first five years, so there was no particular problem running things that way. The biggest outside patch was a change to the well-known text parser, which improved performance somewhat (and was immensely confusing, I was very happy to rip it out during the PostGIS 2.0 work). Otherwise we mostly got small bug fixes, platform support fixes, or an occasional new function.

6. ***How is it organized now and can you describe the evolution of the project's organization to its current state?***

The current organization is structurally a "project steering committee" (PSC) model, with the PSC chosen from amongst the most active committers by the PSC in a self-refreshing model. This organization was adopted when the project moved into the Open Source Geospatial Consortium (OSGeo) and out of corporate control by Refractions. At that point I had left Refractions after getting burned out on consulting, and the company had not had any staff PostGIS developers for a couple years, so most of the development effort was coming from outside the company.

OSGeo insists that all projects under their umbrella be "community driven", so sole corporate ownership and development are not allowed. They also suggest organizing principles similar to those used by Apache projects: PSCs with multiple organizations represented, and open development procedures.

As much as the bureaucratic structure of a PSC, the simple mechanisms of open development are important for a true open project (see Karl Fogel on building open source software):

- Code contributions to an open source code repository, so everyone can see the current state of development
- Open mailing lists for development discussions, so everyone can participate in decisions about development directions
- Open discussion of new contributions, so that new contributors can get a fair hearing about their contributions
- Open ticket tracking system, so that all defects can be easily referenced and the history of development and improvements are transparent

As a single-company project, it would have been easy for PostGIS to retreat away from running in an open fashion, but with the exception of open development discussions (discussions in the office were always too convenient skip in favor of prefer email threads) we already had a fully open project before joining OSGeo.

One thing that is clear in decision-making around the project is that it as much a "do-ocracy" now as it was when Refractions was providing all the developer time. The project members who are currently the most active have a greater say on direction than those who are currently doing other things.

Many of the functions in PostGIS are now "interruptible". Not because the PSC said we wanted interruptible functions, but because the employer of a PSC member wanted them, and the PSC had no objections. In practice, the PSC's strongest power is the power to say "no". Because saying "yes", or rather "we should", is powerless without the resources to effect the decision, and the resources are not controlled by the PSC. Even PSC members mostly have large percentages of their time controlled by their employers, whose interest in PostGIS is part-time at best, and zero-time at worst.

Do-ocracies have their upsides and downsides. On the one hand, project members who are energized are incented for their energy: their energy yields results, and members who are not contributing back off and let progress happen. On the flip side, a result can be short term decision making, particularly when the project is staffed by part-timers or contract programmers. As an example, the internal geometry routines of PostGIS, managed in "liblwgeom" were converted from a static to a shared library primarily to fulfill a consulting contract (doing so allowed geometry cleaning routines from PostGIS to be easily used in SpatiaLite). There were no upsides for PostGIS, only downsides: the build got more complex, and there have extra packaging complexities associated with bundling the library as a system dependency. The upsides were all external to the project, and short term for the consulting engagement. However, collegiality dictated allowing the the work to go ahead rather than blocking it: open source projects run on social capital and conflict is usually avoided.

7. ***We work from a broad definition of leadership as the "ability to motivate others to work toward a common outcome." Within that context, what have been the challenges to managing the contributor community of PostGIS and how has the project's leadership addressed them?***

For a number of years the primary development was corporately controlled by Refractions, so motivation was basically monetary: the corporation set the development goals (mostly completing standards support and supplying functions needed for consulting engagements) and community involvement was limited to enhancements and fixes around the edges of those core priorities.

As the development community became more diverse, the formal processes of describing larger goals before developing them became more important. The direction of development was still driven more by individuals than leadership however, the PSC simply took on a more active role of curating development plans before endorsing them.

Major changes for the 2.0 release provide some examples:

- Changing the on-disk format was a priority of mine, to allow future enhancements and performance work. This was a major change, which would involve touching almost every function in the code base. I documented the change and described it in principle to the PSC before seriously beginning, then did a proof of concept implementation to ensure that the changes did not cause more problems than they solved. At that point the PSC gave full support, and I completed the work to a state where all regression tests passed, so the new work was demonstrated to be backwards compatible with the original implementation. The new code was then merged.

- Adding raster support to PostGIS was done in two stages. Again, the developers first provided documentation, describing how the implementation would work, and the advantages it would provide. To prove the concept, they first developed their code as an add-on to PostGIS. Only once they had completed the add-on (which worked on top 1.5) and demonstrated the utility of it and community support for it, was it brought into the main code base for 2.0.

8. ***How does the PostGIS leadership set the agenda and direction for the project? How is this direction maintained in an open-source setting where contributions may come from any segment of the community?***

   In the early days, the existence of the OGC SFSQL standard (and later the ISO SQL/MM standard) provided a general framework of "desired functionality" that the community could work towards, in the absence of their own priorities. A similar effect could be seen in Apache Java projects which often were test implementations of upcoming Java community standards. The GeoTools/GeoServer project also hung functionality around a core framework of OGC web services standards. Other open source projects have been "fast follower" projects of proprietary offerings: OpenLayers followed Google Maps; Hadoop followed Google's MapReduce; OpenStack follows Amazon AWS. Still other open source projects are themselves re-inventions of existing projects following a particular technical aesthetic: Leaflet can been seen as a reimplementation of OpenLayers with a minimalist aesthetic (also, against a more modern language philosophy); MapNik can be seen as a reimplementation of MapServer, using a "modern C++" design aesthetic.

   In most cases, the core direction is set externally initially, and then elaborations flow in organically from the community. The OGC standards for WFS/WMS/SLD do not contemplate variable substitution, but GeoServer provides the capability because they dramatically enhance the power of the services. The SFSQL standard did not include support for Z/M dimensions, but PostGIS did from early on because higher dimensions were a practical requirement for Refractions in consulting engagements. However, from very early on the SFSQL standard was proposed as a development roadmap (http://lists.osgeo.org/pipermail/postgis-users/2001-June/000031.html)

9. ***What roles and responsibilities do you recognize in the PostGIS project? How do you identify and cultivate emerging talent in its contributor community?***

   Within the project itself, we have only two formal roles: committer and PSC. However, functionally everyone in the PostGIS ecosystem ends up in functional roles depending on how they choose to interact with the project:

   - Users take the software as/where they find it, and run it to their own purposes. They are effectively invisible to the project.
   - Advocates both use the project, and also teach others about it. They effectively grow the base of users, which in turn grows all the other categories above, as people migrate from role

to role over time. Advocates are usually invisible to the project, since they tend to advocate locally, out of view of the larger community.

- Members join one of the community fora and become visible, as participants on the mailing list, or as users of the ticket tracking system. There are 2000 members of the user list.
- Reporters come across issues with PostGIS and report them back to the community, either on the mailing list or as tickets. While users and advocates might also identify bugs and workarounds, the threshold act of testers is returning their knowledge to the common pool. There are about 300 unique reporters in the ticket system.
- Contributors supply code and/or to the project, often as patches attached to trouble tickets, or enhancement tickets. There are about 40 contributors.
- Committers can apply changes directly to the official source code repository. There are 12 committers.
- Project steering committee approves major changes/enhancements to the code base, there are 6 PSC members.

Providing commit access to active contributors of documentation or code is the primary lever we have used to encourage new membership. In general, highly active new contributors have been integrated into the community successfully. (eg, Mark Cave-Ayland, Olivier Courtin, Bborrie Park, Regina Obe, Nikolas Aven). However, granting commit rights is not in itself a spur that will generate a new active member, the real requirement seems to be external professional need for the software. All the new active members who have remained active have professional associations that encourage continued activity.

- Regina Obe consults on PostGIS and has written a book on it.
- Bborrie Park uses the raster subsystem of PostGIS (which he maintains) extensively in his role at the University of California.
- Olivier Courtin consults on PostGIS and his company provides PostGIS expertise.
- Nikolas Aven uses PostGIS in his professional work in forestry.

In cases were professional roles change, activity also changes along with it.

- Mark Cave-Ayland was a substantial contributor while working closely on PostgreSQL projects in his consulting business, but his activity has gone down as his work has moved into other fields.

10. ***As the PostGIS project, its user base, and contributor community have grown, what developments have occurred that you found unexpected or surprising?***

One surprise is that it took off at all, and the speed with which it did. It speaks to the extreme utility of spatial-in-a-database that once the functionality was available users rushed rapidly to it. PostgreSQL community members have commented, correctly I believe, that PostGIS has been a major driver of PostgreSQL adoption. The extension is driving adoption of the core.

Another surprise is that PostgreSQL itself has more or less stayed out of the spatial business, even though it has in-built geometric types that pre-date PostGIS by a decade (they were part of the original Postgres academic implementation by Stonebraker at UC Berkeley).

Perhaps the quietest but most determinative development is the importance of testing infrastructure to the ongoing growth of the project. During the period when Refractions was still funding PostGIS development, the primary developer Sandro Santilli took advantage of some undirected time to put in place a regression testing system, and a fairly complete set of regression tests. These tests dramatically de-risked future development, catching errors in implementation and allowing wholesale changes like the PostGIS 2.0 work to go forward with almost no user-reported issues. Having this huge automatic safety net for new changes allows the leadership team to allow contributions and patches with a lot more freedom and less effort than otherwise. The success of the SQL tests led to a suite of C tests for PostGIS 2.0, and a policy that all new features and all new commits in general come with a test that exercises them or correctness, so the safety net only gets more comprehensive over time.

11. ***Can you describe an example of an innovation that came from the PostGIS community that wasn't necessarily in the project pipeline? How does the project leadership incorporate such developments into the project?***

Raster support was never a priority of the established project leadership, though now of course the leadership does include raster maintainers. Bringing raster to PostGIS was a deliberate project on the part of Pierre Racine, for whom it was a priority of his academic contracts. Again the impetus was external, but since the development was also external and the PSC generally disinterested or hostile, the process is worth looking at.

Racine first compiled a complete design that explained both the philosophy of the implementation and the reasoning behind it. (http://lists.osgeo.org/pipermail/postgis-users/2008-July/020456.html, http://trac.osgeo.org/postgis/wiki/WKTRaster). This not only demonstrated his seriousness about the project, but provided a road-map that would otherwise have been missing. There was no standards base for a raster SQL API, and the existing implementations in products like Oracle had been rejected in the past as not providing any unique capability for raster management.

He engaged the PSC directly (in particular, me) to get objections aired early. (http://lists.osgeo.org/pipermail/postgis-users/2008-July/020511.html)

When he did begin his implementation, he did so outside of PostGIS proper, as an add-on (WKTRaster) that allowed the concept to be proved before the PSC had to decide whether to adopt it (as they did at a meeting at FOSS4G in Barcelona in 2010).

12. ***What have been the key challenges to keeping the PostGIS project sustainable over the long-term? What challenges do you foresee over the next five to ten years?***

PostGIS is a project that is just big enough to have a wide user base and a multiple-developer ecosystem, but it's right on the edge. Most of the time it's effectively a single-developer project, as a particular PSC member works on a particular project. Only at release times, when enough features have built up that a numbered release seems necessary to get the work out to the user base, do all the team members work simultaneously, clearing out old tickets before release.

In software, there are always some problems that are really hard to fix, but that are not critical to fix. PostGIS has had very-good-but-not-perfect handling of geometry overlay and union for many, many years. People only notice the imperfection in those rare cases where some piece of their geometry breaks the code. They usually figure some workaround, perturbing the coordinate or rounding them, to force their recalcitrant geometry to work, then they move on. Fixing the problem for good would require perhaps 6-12 months of concerted effort, on GEOS and PostGIS development. The problem remains, un-worked. The last serious external investment in overlay robustness was in 2008, and that amounted to about $25,000 in total.

Similarly, users find PostGIS so robust and useful for mid-size and large data sets that they move on to run extra large data through it, and then wonder why they can't make it faster by using extra cores. Making PostGIS/PostgreSQL capable of multi-threaded operation is a major development effort involving PostgreSQL and PostGIS developers. No one user organization is willing to fund the development alone, though all users would of course benefit immensely if it were ever done.

Past efforts to aggregate small contributions into larger development support envelopes (http://blog.cleverelephant.ca/2005/10/concurrency-for-postgis.html) have proven extremely difficult, even when the funding envelopes are very small compared to even the lowest levels of proprietary licensing.

As with general leadership and priority setting, the PSC lacks the resources to direct long-term investments in the software. Fortunately PostGIS is relatively small, and leverages development effort from the larger PostgreSQL community -- multi-threaded execution may finally arrive in PostgreSQL and thus be accessible to PostGIS, but not because of PostGIS, even though PostGIS users will be some of the primary beneficiaries.

The most effective open source projects have been ones that provide simple cores and build a community of extensions around the core. The core remains small enough to be tractable, while the extensions can be independently contributed by companies/consultants/contractors who have practical needs for specific functionality. Apache and Apache modules. Leaflet and extensions. Linux and hardware drivers.

As PostGIS becomes more and more "feature complete", this problem only magnifies, as even small feature addition projects become rarer and rarer, the core PSC will be more involved in other projects, and less available for simple maintenance and release work on PostGIS. This is hardly a problem unique to open source, since proprietary product also move from "active development" phases to "maintenance" phases as they age and reach stability, and open source projects generally move into a "maintenance" mode organically, when it ceases to make sense to add new and whizzy extra features.

13. ***What accomplishments of the PostGIS project do you attribute specifically to it being open-source? How could these have not been accomplished otherwise?***

- Existing and achieving the ubiquity it has. There's no way a closed source PostGIS achieves the level of market penetration that we have as an open source project.
- Leveraging the larger open source ecosystem. The use of libraries like Proj4 and GEOS was critical to moving PostGIS ahead fast, and sharing development effort with other organizations. While proprietary software can also make use of open source libraries, there is often a cultural bias towards maintaining IP "purity" that doesn't show up in open source communities. Some aspects of PostGIS were copied directly from other projects at a source code level (some caching and memory management tricks, for example) that in a proprietary project might have gotten the lawyers involved.

14. ***Are there use cases or situations where open-source may not be the ideal approach to solving a technical problem?***

For solving a technical problem, never. Given a set of resources, the philosophies of open development and equal participation are the best way to leverage both internal and external resources to achieve maximum development quality and quantity. However, that assumes a fixed set of resources. (eg, if an organization just wants a particular software capability, and has the budget to create it, and doesn't plan to monetize it, just to use it, why not develop in the open? the plusses are significant and the minuses negligible).

However, if the software has to generate its own resourcing from the marketplace, then open source might not be optimal. Truly unique new capabilities can often be developed more quickly and brought to market more effectively as proprietary software: the short term lifecycle can be resourced much more richly and run much more quickly. However, most software isn't truly unique, and the flip side of IP control and monetization is IP stagnation. Open source excels at finding new and unexpected uses for old software, while proprietary software generally only finds uses expected by the originators.

This is why fast-follower open source projects are so common. An initial implementation has already proven the utility of a concept, and the open source project provides a more accessible version of that capability, with more responsiveness to the technical user community. For pure technical / computing products, the risk of being lapped and rendered irrelevant by a fast follower is no longer trivial, which is why most new proprietary software is highly user facing, and most server-centric technical software is open source, even when the creator is corporate and even venture funded. (e.g. Docker)

15. ***PostGIS has been widely adopted across the geospatial community, supported by numerous open-source projects and leading commercial products alike. It is regarded as a successful project. How is success defined and measured for an open-source project? What are the common attributes of open-source projects that are seen as successful?***

    It doesn't take much to create a successful open source project: it has to be useful to you, and working on it has to give you some sense of satisfaction. Often gaining users will provide extra satisfaction, in ego-gratification terms. Achieving more success than that is a relative rarity.

    The most successful projects will out-live their creators, having gathered a community of users large enough in incubate sufficient reports, contributors and committers to become self-sustaining over the long term. In the geospatial world, a relative handful of projects have achieved that: PostGIS, MapServer, GeoServer, GDAL/OGR, OpenLayers, Leaflet, MapNik, QGIS, GRASS, and some others.

    Open source projects compete in the marketplace of ideas, so to some extent it's possible to use a tool as blunt as Google Trends to gauge progress over time (http://www.google.ca/trends/explore#q=postgis%2C%20geoserver%2C%20openlayers%2C%20mapbox&cmpt=q). Number of users, number of web searches, chatter, conference talks, the progress of open source projects it a 100% memetic competition.

16. ***Based on your experience with PostGIS, as well as open-source software in general, what lessons can traditional organizational structures (such as corporations or governments) draw from successful open-source projects in order to improve outcomes? How could these lessons potentially be applied beyond technology in such organizations?***

    The number one lesson is the value of openness.

    Even organizations which sell closed source software can reap a huge benefit from openness. Atlassian, which makes closed source bug tracking and software development productivity tools, is a customer darling because they run basically an open development shop. Everything is open except for the source code, so end users (and more importantly, power users) feel very close to the process. Atlassian reaps all kinds of benefits:

    - Good will among their users which translates to
    - User engagement with activities like bug identification and tracking which translates to lower maintenance costs for them, and
    - User engagement with activities like helping each other on open fora which lowers support costs.

    Everyone wants user engagement and feedback, but too many organizations think that opening a digital suggestion box is all that is required. In order to incent engaged, high value feedback, you have to engender community, which means you have to open up your processes, and allow everyone to have a visible voice. If users can see their feedback actually being integrated into the open process, they have an incentive to engage.

The second lesson is the value (to openness) of a level playing field. Open source achieves this through the mechanism of the license: everyone has exactly the same legal rights to use and alter the code, nobody has an advantage. In extreme cases, the open source license allows the project to bifurcate and go in different directions depending on the desires of different user communities. This is harder to achieve in a hierarchical organization structure, where differentiated levels of power and control are the raison d'etre, but on a project by project basis has potential. When trying to bring Department A along as a partner in your project, does a mandate work better or a relationship where control is shared? In the short run? In the long run?