Wikipedia is closing in on the landmark figure of 40 million articles spread across 292 language editions. It is one of the top most popular sites on the internet and looks to be its most visited reference resource. One aspect of its existence that is indisputable is that it is the product of a global volunteer community effort to create, edit and maintain content that is un-matched. By any standard, Wikipedia is a runaway success, but it has not been a smooth, straight road along which it has traveled, as it has faced charges of being run by an editorial oligarchy rather than a well-managed and unprejudiced community committed to transparent, factual reliability. Communities can, it seems, be problematic. Understanding and managing communities is not all sunshine and roses but when done right it offers its members a great community experience. A successful community manager needs to wear many hats: leader, moderator, advocate, mediator, analyst, friend and so many more.

The word ‘community’ can mean many quite diverse things, but it does specifically refer to a group with common goals or interests. When we use the term ‘language community,’ we really are using an umbrella term, but it’s convenient to group many elements together which share some basic characteristics. Where a community’s rules and regulations come from is also varied as are policy statements and so on. The word ‘volunteer’ is also difficult to confine to a set meaning. Volunteerism occurs in many quite different spheres of activity. However, it is generally understood that whatever the motives a person might have for volunteering their time, services or skills, by volunteering, they are engaged, committed and offer to work free of charge.

The volunteer communities within the Language Industry fall under three distinct categories: developer communities, translator communities and end-user communities. Volunteer engineers donate their time and coding skills to community open source projects such as Okapi, Appertium, OmegaT and many more. Translator volunteers donate their time and translation skills to mostly humanitarian causes such as crisis relief initiatives through Translators without Borders or inspirational community reach through TED talks, just to mention a couple. End-user communities are actual application users, as opposed to those who design and build them, contributing by localizing the application for their locale, examples of which are Mozilla, Facebook and Wikipedia.

One idea that has recently been developed out of voluntary activities involving the online community is ‘crowdsourcing’ often mis-used instead of ‘community.’ Whereas community is about interacting with its members and forging relationships, crowdsourcing is about engaging an audience. A few corporations have managed successfully to deploy crowdsourcing translations, mostly because few brands have the type of crowd-appeal necessary. For crowdsourcing to succeed there needs to be a product so irresistible to the general public that many professional translators will be willing to freely donate their time to localize it in their language. Brands like Facebook can pull it off, not because they don’t want to pay for translation, but because they want their community of end-users to be involved and committed and that multilingual community of users has been more than willing to
be mobilized in a cause to which they are dedicated. The mere volume of some social media cannot be localized in real time unless the community of end-users is willing to do it. Community-driven projects and crowdsourcing are both models of social interaction with end-users and they both have different benefits and strengths.

Localization Developer Communities

Compared to some other open source communities, our localization pool of developers is significantly smaller and all of them are so busy with their day jobs allowing them very little time to contribute to open source projects. Instead of ideology, the primary motive in open source localization projects is a real and specific business need. Code contributors tend to be developers who are using the tools themselves and have thought of a way to improve, enhance and possibly make a certain task more efficient. In some cases, when a developer wishes to combine an open source application with existing proprietary software, there may be a need to make adaptations for compatibility purposes and this code contribution benefits the entire industry.

The most common scenario for further development in open source projects is a specific need from a business unit which is willing to pay for a new feature. Whereas the core contributors to most open source localization projects will fix bugs and make small improvements, these do not move the project forward. Keeping in mind that contributors are using their free time to help, it is no surprise that creating and adhering to a project roadmap is impossible. Timelines are unpredictable and it’s only when funding is available for a dedicated engineer that the project moves forward.

There is real economic benefit in having open source localization projects as there are many common components that automation providers and tool makers need. For example, to extract text from different file types, generate XLIFF files, etc. There is compelling reason to re-circulate code and eliminate redundant repetition by building these components all over again and again.

Chase Tingley from Spartan Software is a member of the small team of contributors to the Okapi project and he believes: “It’s a matter of distributing engineering investment for the portions of the technology stack that are effectively commoditized.”

Tingley, who is currently Vice-President at Spartan Software Inc, started contributing to open source straight out of Harvard University in 1999. At the time there was a great deal of enthusiasm about the internet and how open source was changing software development. Tingley like many others aspired to the idealism and took the opportunity to improve his programming skills through large open source projects. He started by submitting patches to the text editor “Vim” and then moved on to contribute to the Mozilla codebase for another 18 months. Today he coordinates how Spartan Software uses Okapi, as well as the development they contribute.

The Okapi Framework is a set of interface specifications, format definitions, components and applications that provides an environment to build interoperable tools for the different steps of the translation and localization process. The project uses the collaborative code-development platform Bitbucket, and also hosts a wiki that is used as a knowledge base. Anyone is welcome to join and offer contributions by reporting bugs and issues, or by providing feedback, corrections and suggestions on the documentation and the specifications and finally by suggesting improvements to the existing material or recommending new developments.
Contributors who have the appropriate skills are encouraged to participate directly in the development of Okapi libraries, components, and applications, or in developing help and documentation. Bug reports can be filed in Bitbucket, and proposed code changes can be submitted through Bitbucket’s “pull request” feature. Additional discussion and support can be found in two email lists: the okapitools group for users, and the okapi-development group for developers. Both lists are open to anyone.

The material developed under the Okapi Framework project is licensed under the GNU Lesser General Public License agreement (GNU LGPL). This is one of the licenses approved by the Free Software Foundation. In summary, the GNU Lesser General Public License is designed to ensure that the code, as written by the author, must always remain free. However, the library where the code is contained may be used and linked by non-free applications.

Although there is no formal governance, the project is small enough that it allows the members to reach a consensus on technical decisions and to resolve any differences of opinions. Dealing mostly with data processing and transformation, the subject rarely generates strongly conflicting opinions on user experience or feature design. There is no formal process for code review, and the approved members can push changes without review, although informal reviews are common. For code contributions from the community they use pull requests (a method of submitting contributions to an open development project) and usually a couple of the members will sign off. Occasionally they may have to alter it or ask the author to change it. Most of the development is done by US contributors, however recently there is a growing trend of bug reports and code contributions from users and developers in Europe. Looking inside the Okapi project gives us a fairly accurate glimpse of how localization open source projects are today. They may seem to be in their infancy compared with projects like Gnu, Mozilla and Apache, but they are providing a needed functionality and enhance community collaboration.

Mozilla Localization Communities

When it comes to volunteer organization and community management, no one does it better than Mozilla. With a mission to ensure the Internet is a global public resource, open and accessible to all, Mozilla is a community structured as a virtual organization governed by meritocracy. With well-defined roles and responsibilities, Mozilla has policies covering every aspect of digital communication. Mozilla boasts an amazing 10,500 Mozillians (community volunteers) worldwide covering 89 languages and holds hundreds of events annually including the famous Mozilla Festival. Since its inception in 1998 as the Mozilla Project, it now consists of the Mozilla Foundation, its subsidiary the Mozilla Corporation and the Mozilla Reps Council which is fully composed of volunteer Mozillians.

I met with Jeff Beatty, the new Head of Localization for Mozilla whose mission is to lead one of the longest-running community-driven localization programs in the software industry aimed at making the Open Web accessible in all the languages of humanity. Beatty joined Mozilla as a L10n Program Manager in September 2011 and was later made a Localization Engineer. In April 2013 he successfully founded Mozilla Utah, aiming to attract Utah’s linguistic talent to open source projects. Beatty shared that the Mozilla Community has extremely well defined Participation Guidelines covering two parts. The Diversity and Inclusion section encourages everyone’s participation and the Interaction Style section defines expected behaviors. Both sections have clearly defined resolution paths. In addition to the general Mozilla Guidelines
which refer to governance, the Localization community has specific language related Style Guides (based on the Multidimensional Quality Metric framework) which define standards and contain rules that are used to both translate and evaluate a translation’s quality including: Style as in tone, consistency and cultural references; Terminology as in developing term bases and handling difficult concepts; Internationalization as in format, dates, names etc.; Grammar; Fluency; and Accuracy.

The Localization Drivers team consists of a technical group and a project management group while everyone is acting to different extents as community managers. The Mozilla general participation guidelines are adhered to but there is a lot of tolerance for minor violations, as keeping the cohesion of the community is more important than establishing a zero tolerance policy. Repeat violators are handled with care and rarely banished. Having established an autonomous governing policy from the start, the Mozillians are well practiced in discussing, generating feedback and facilitating decisions and resolutions for everyday issues. When there are suggestions for major participation policy changes, there is direct access to the Chairwoman Mitchell Baker.

The technical group has been very busy in developing a new framework and applications to facilitate the numerous communities around the globe, all open source and available to download. L20N is a new open source localization framework for the web allowing localizers to put small bits of logic into localization resources to codify the grammar of the language. In essence L20n’s framework removes the need for developers to thoroughly understand the specifics of a natural language and provides an opportunity for localizers to create better translations. Beyond that, L20n provides an intelligent language fallback for web apps (e.g., if an app module does not yet have es-MX strings, developers can tell the app to fallback to es-AR strings) and will eventually allow for seamlessly updating localizations to end users outside of delivering new software binaries. Another application designed and launched by the localization team is Pontoon, which allows web content to be localized in place, with context and spatial limitations right on the live webpage. Pontoon is a very simple and intuitive tool that does not require advanced technical skills of its users. All that is needed is that the localizer hovers the mouse over the web content, selects a block of text, translates it and finally saves it. Mozilla also uses Pootle, a web software powered by Translate Toolkit 1.13.0, enabling Mozillians to do both translation and translation management. This community tool is Free Software, and any team can use it and contribute to the Pootle community.

The team utilizes a two-tier review model facilitating the entire process within the community. Although the technical review is performed internally by the technical group, in language matters there are some Mozillians that have been vetted and granted permission to contribute without review. The localization team is moving towards automating elements of their manual technical review process in an effort to remove themselves from being dependencies in the review process. This will allow them to focus on understanding cultural differences when they are addressing different locales within the global community. Keeping the global community motivated, engaged and cohesive is a complicated and multi-faceted task that needs to be customized for different locales. For example, communities in Latin America are very proud to
wear Mozilla t-shirts proving their involvement while in India a certification with Mozilla’s badge on their resume is enough to make them happy.

I asked Beatty to explain to me the difference between crowdsourcing and community.

“Community-driven projects are very personal, a community comprises of people you know, work and speak with everyday, know their strengths and weaknesses, are friends and you can count on them. A crowd on the other hand is an amorphous entity removed from you and only engaged through generic messages trying to compel them to act with no inclination of what the response to your “call to action” will be. Whereas in crowdsourcing a marketing strategy may or may not bring in the desired effect, community-driven projects require a large buy-in and create a lot more overhead and expenses in adding that personal touch which is invaluable to finding success in community models.”

Given that our ideas of what a community is and how it behaves are somewhat fluid, it’s no surprise that they are open to change in many and diverse ways. When speaking about the language community, we also need to take into account that languages constantly change and often in quite unpredictable ways. Usage brings about variation. Yet there’s a third factor that we need to take account of and that is technology. As the millennium dawned not so long ago, many of us speculated about the changes we would experience in this new era. With the exception of the Ray Kurzweils of this world, who knew that we would be using real-time automated translation on mobile devices across the globe?

As our communities evolve, we must expect new standards and norms to emerge. With their myriad interdependencies, variable standards and complex datasets, these are notoriously difficult to codify. But we must never lose sight of the fact that while ‘ought’ and ‘is’ are caught in an eternal struggle for dominance, communities are about people. However, that is as things stand right now. At a time when there is a burgeoning debate about whether entities like Siri are a part of our community and whether they have rights, we may face some quite unexpected challenges. Furthermore, as computers develop and ‘talk’ to each other in their own arcane languages, we might find ourselves having to accommodate a completely new set of foreign tongues, as it were. There are those in the AI community who are very diligently pondering these issues right now, but that’s another story.