

UNIFYING THE CITY OF NEIGHBOURHOODS:

how ideas behind open source software development tools
may benefit Toronto city government

by

Grant Patten

g.patten@utoronto.ca

Matthew Mandula

matthew.mandula@utoronto.ca

Second Draft

INF 1004

April 11, 2010

Many of the shortcomings associated with municipal governments in Canada may be ameliorated through the careful consultation of an ostensibly unrelated area – open source software development. More specifically, this paper will show how the City of Toronto government could make policy development improvements in fairness, efficiency and communication if officials were to study distributed version control systems and emulate some of their qualities in the development of new tools for policymaking. If Toronto government officials were to study certain tools popularly used in the development of open source software, they could use them as models for creating new tools to support both policy development and complaint reporting.

To provide an overview of the paper, the idea of improving the fairness of policy development in Toronto through the creation of a distributed version control system will first be discussed. Secondly, the idea of improving efficiency by breaking down traditional hierarchies in Toronto city government will be discussed. Thirdly, the idea of improving both neighbourhood-to-neighbourhood and neighbourhood-to-government communication will be discussed. Finally, the idea of improving public services through the creation of a bug tracking tool for complaint reporting will be discussed.

Canadians ought to care about these ideas because – if properly adopted – these new software development-inspired tools for policymaking and complaint reporting could very well help to improve the democracy of their society. As it is the “city of neighbourhoods,” (Kern, 2005, p. 361) Toronto is an appropriate framework to use in this paper; however, these ideas are in no way exclusive to Toronto city government. These ideas may be applied to any municipal government that is struggling with similar issues. Despite the amalgamation of municipalities that occurred in 1998, (Gilbert, 2004, p. 248) Toronto city government must still deal with over

240 distinct neighbourhoods that have their own individual concerns. A policy that one neighbourhood agrees with might be entirely counter to the interests of another neighbourhood, for instance. Additionally, residents must struggle to cut through a lot of bureaucratic red tape (Access Toronto, 2007, p. 2) in order to even get their policy opinions voiced on any kind of substantial level. In fact, the status quo of the Toronto municipal government may be likened to a centralized version control system such as Subversion for this very reason. In a Subversion open source project, a select few make the final decisions and all subordinate users must get their permission before any changes can be implemented. Programmer James Golick writes about Subversion, “anybody wishing to make changes to the source must first check out a working copy from version control, submit a patch, and *hope that it is accepted.*” (2008) Users who have been privileged with this level of control are called *committers*, and unless a less privileged user can convince a committer that their code is worthwhile, it may never see the light of day. It is in this sense that committers in Subversion are a lot like city councillors in Toronto. Residents in a neighbourhood must go to their area’s councillor and attempt to convince them – usually by speaking at a committee meeting – that their policy idea is worth considering. (Access Toronto, 2007, p. 2) One problem with this model is that a few individuals get to decide what is best for the users and the residents. A more democratic model would no doubt allow the users and the residents to decide what is best for themselves. A distributed version control system such as Git would break down this hierarchy, (Google, 2007) thereby creating a more democratic environment. With Git, there are no individually specified committers (*ibid.*) a user must go through in order to get their code integrated into the project. Rather, the community itself decides whose code gets integrated and whose code does not. A user no longer has to convince one very important person; rather, all they have to do is convince another user, and then another user, and

so on, with the expectation that if the code is any good, it will become adopted into enough users' repositories to be made very public and eventually integrated into the best version of the code, (Golick, 2008) wherever that may be in the system. Golick writes, "if the original author continues to maintain the best version of the code, great; if not, users of that code can begin to pull from whoever does have the best version." (*ibid.*) A distributed version control system, then, eliminates the hierarchy present in centralized version control and allows more of the community rather than a set group of individuals to have a say in what gets implemented. Extending this concept to the domain of policy development, decisions could be made by the neighbourhood itself rather than the set group of city councillors. It is conceivable that a Git-like version control system be designed for the purpose of allowing citizens to publicize their policy ideas and any other concerns they may have relating to Toronto city government. Rather than writing code, though, neighbourhood residents would write documents that detail their policy concerns and submit their writing to the government's distributed version control system. The documents could be provided to residents in standardized templates to help organize the writing. Like within Git, if just one other resident finds the policy idea worthwhile, they may adopt it into their policy repository, perhaps applying a few revisions of their own, and then another resident may do the same, and so on, until a chain reaction is created and the policy suggestion becomes so widespread throughout the community that it is impossible to ignore. After such a clear, objective indication of community interest has been made, it would be very difficult for city government to still not approve the policy. This is simply a fairer, more democratic means of going about policy development, as the residents within individual neighbourhoods would be able to collectively decide which policies get the most attention, as opposed to leaving this decision in the hands of government representatives with ambiguous motivations.

A distributed version control system for Toronto policy development would also increase the overall efficiency of the city government. Like the committers in an open source project with a centralized version control system, the city councillors are sometimes confronted with hundreds of suggestions in a short time frame. Naturally, when it is left up to a small number of people to examine each suggestion, it takes a long time to formulate a response, no matter how qualified those people are. In order to speed up the process, then, the unfortunate reality is that many potentially intelligent suggestions will simply be ignored under this hierarchical framework. A well-documented example of this phenomenon is when Linus Torvalds, the benevolent dictator (Fogel, 2009, p. 68) of Linux, could not keep up with all of the patches that were being submitted to him by contributors to the Linux kernel and was eventually suspected of flat-out ignoring many of the submissions. (McVoy, 1998) The solution was to incorporate a distributed version control system into the project, thus flattening the hierarchy and creating a “network of trust” (Google, 2007) around Torvalds so that the entire Linux community may contribute toward the vetting process and only the most worthwhile patches would reach him for ultimate approval. With residents continually contributing policies, a system similar to the network of trust could work for Toronto city government. Certain residents will eventually emerge as standout contributors, continually proposing policies that get widely accepted by their entire neighbourhood. In turn, the city councillors should have an easier time trusting the value of proposals from individuals who have emerged as standout contributors. In the hierarchical status quo of Toronto, it is entirely likely that policy suggestions will similarly fall by the wayside and be ignored, as expecting a few councillors to efficiently manage a city of millions who may be voicing their concerns at any given time is quite impractical. With a distributed version control system for policy development, however, the city of millions will be able to

manage the city of millions, thereby increasing productivity and efficiency because more will get done at a faster rate, turnaround time will be quicker, and fewer suggestions will be ignored.

Following Linus' Law that "given enough eyeballs, all bugs are shallow," (Raymond, 2000) the distributed version control system for policy development will provide Toronto city government with the opportunity to recognize which policies concern residents the most at any given time.

With a distributed version control system for policy development, communication may be enhanced not only between residents and city officials, but also between the residents themselves. This enhancement in communication may be facilitated through the branching capability that a distributed version control system such as Git provides. It is true that branching may also be done in a centralized version control system such as Subversion, but the frustrations associated with implementing this capability in Subversion are well-documented. (Collins-Sussman et al., 2008, p. 84) It is much easier to implement branching, then, in a distributed version control system. The value of branching is that multiple lines of development may be maintained simultaneously on the same project. (*ibid.*) In a 2007 presentation at Google, Torvalds spoke about how with a distributed system an open source project may be neatly parsed out into separate units for users to work on. That is, one group of users may work exclusively on the established parts of the code, while another group of users may work exclusively on the more experimental parts of the code. (Google, 2007) In a distributed system, the expectation is that the changes that are made separately will eventually become merged together if indeed the changes are considered worthwhile by enough users in the community. There are many neighbourhoods within Toronto that share policy commonalities but also have their own individual concerns, and this is why a Git-like version control system for policymaking would enhance communication amongst residents in these various neighbourhoods. Policies that must be shared by all

neighbourhoods may be included in the originating branch, e.g. Toronto-wide laws that have to be abided by under any circumstance. But from there, residents within various neighbourhoods may begin to create their own “policy repositories” based on this originating branch, adding their own policy concerns that relate to their specific neighbourhood. For instance, residents in the Chinatown neighbourhood may work on their repositories, adding their unique policy concerns for their area, while residents in the Forest Hill neighbourhood may work on their repositories, adding policy concerns that will no doubt be quite different from those submitted by people living in Chinatown. As the Chinatown residents’ repositories are communicated out to the Forest Hill residents’ repositories, policy conflicts may emerge. A policy requested by Chinatown residents might somehow directly or indirectly conflict with a policy requested by Forest Hill residents. This is where the branching of the distributed version control system facilitates communication between residents. Once the conflict is spotted, it will be left up to the residents to manually resolve or “merge” (Collins-Sussman et al., 2008, p. 90) the conflict themselves by talking and perhaps arriving at some sort of a compromise. Without a distributed version control system in place for policy development, then, this conflict between the communities may have remained hidden or emerged far later in the process, thus making it much harder to resolve. Additionally, the branching capability will allow for enhanced communication between the residents and the city councillors because once the residents think they have arrived at a possible solution, they will propose it to the city government for official implementation. The mere existence of an officially mandated distributed version control system will actually encourage residents to communicate with city government, and provide a more organized and efficient means for doing so.

In order to fully understand how tools used in the development of open source software may be applied to Toronto city government, one must think about bugs as public problems, not merely technical problems. The notion of bugs as public problems may be applied to many of the public services offered in Toronto, such as public transportation. Commuters complain about the Toronto Transit Commission (TTC) on a regular basis, especially when it comes to the TTC's subway service. (TTC Complaints) These complaints can be deemed "bugs" because they commonly represent legitimate problems with the particular public service. It has to be a key precept for the public to interact with the government promptly on these problems if they are ever going to get resolved, and a bug tracking tool for complaint reporting would no doubt facilitate this interaction. Simon Tatham, a programmer, discusses how to report bugs effectively in a seminal essay from 1999. He writes, "in a nutshell, the aim of a bug report is to enable the programmer to see the program failing in front of them... when you report a bug, you are doing so because you want the bug fixed." Similarly, bug reporting for the TTC would enable employees to clearly see what the problems are, thereby positioning them to fix the bugs at a faster rate. A column published in the Toronto Star on April 8th, 2010 perfectly illustrates that a major problem with the TTC is this lack of a clear communication channel between TTC management and the public. It reads, "...a TTC spokesperson noted that labour and management have the same goal – crafting better relations with the public." (Airing) The bug tracking tool could even be combined with the aforementioned distributed version control system in order to ameliorate potential issues of flooding. After a commuter submits their bug, the report may be fed into a distributed version control system. Here, the same vetting process that was outlined earlier for policy development may be applied. This system may be comprised of thousands of other concerned commuters. Any duplicate complaints that come into the system may be filtered

through the users of this distributed system. The duplicate complaints could even be tallied in order to see how many commuters are complaining about the same issue, and this could perhaps go toward strengthening its importance in the system. Bugs that are seen as important by many users will become adopted into those users' bug repositories, eventually accumulating into one democratically-constructed repository of bug reports that may be submitted to TTC management.

The TTC is a great example of a public service with “bugs” because it is subsidized and ultimately controlled by Toronto city government. (Bolton, 2003) Tatham writes, “in bug reports, try to make very clear what are actual facts and what are speculations. Leave out speculations.” (1999) After the bugs are reported, they could become logged into a distributed version control system that would allow for the community to eliminate reports that appear to be mere speculations. “Standards, categories, technologies, and phenomenology are increasingly converging in large-scale information infrastructure...” (Bowker and Star, 1999, p. 47) The TTC can be seen as this kind of large-scale information infrastructure that is increasingly dependent on technology for the improvement of their system. It seems only a natural step, then, to relate this infrastructure to the technological domain of open source software development tools.

One specific tool for bug tracking that would serve as an appropriate model for Toronto city government is Bugzilla. The fifth chapter of the Bugzilla guide discusses the anatomy of a bug, the life cycle of a bug, searching for bugs and reporting them. (Bugzilla Team, 2010) If the government imposes a similar bug tracking tool on the TTC, issues will no doubt come to light more quickly. Even though Toronto city government essentially controls the TTC, (Bolton, 2003) that does not mean government officials are currently aware of everything that citizens consider wrong with the system. If there were a conspicuous tool put in place akin to Bugzilla whereby each and every citizen could make their complaints heard, officials will more quickly

become aware of the problems that need fixing. “[The internet is] a technology with the capacity to engage and enable interaction across geographies and boundaries, both physical and cultural, and to support initiatives from the ‘bottom up’ as well as the ‘top down.’” (Gurstein, 2003, p. 2) Similarly, in order to facilitate greater interaction between TTC management and the public, a “bottom up” initiative would allow commuters to submit their complaints through the bug tracking tool. The “top down” perspective would factor in later, as users of the distributed system would ultimately submit their developed complaint reports to management for approval.

Consequently, with input from Simon Tatham and the Bugzilla guide, it is clear that bugs may not be seen merely as technical problems, but complaints from the public concerning the state of various government services. “Information is, by its essence, primarily a public good, and it is only by means of technology and aggressive regulatory control that its private nature can be made dominant.” (Gandy, 2002, p. 450) This means that the public should be made aware of the inner-workings of the system, and should be provided with the necessary technology to publicly critique the system. In this sense, a tool for bug tracking is integral to the survival of the system in place. Tatham writes, “tell them exactly what you did... wherever possible, you should provide a verbatim transcript of the session, showing what commands you typed and what the computer output in response.” The bug tracking tool for complaint reporting could be developed into hardware and installed on various subway station walls. Security could be built around the tool to prevent against vandalism. The hardware could provide the commuter with a clear interface and standardized process for going about reporting their bug. As the TTC system is currently, when a problem situation occurs, people often have no idea what to do and how to document it. A commuter will therefore have to find their own individual way of voicing a complaint, whether that be yelling at an employee or keeping quiet until they get home, then

sending off an angry e-mail. However, if a specific tool similar to Bugzilla were noticeably put in place for complaint collection, a standardized process could then be publicized throughout the entire TTC on how to report complaints. When a grievance is issued, people would be more mindful of providing all the information necessary with a formal tool already in place to cater to their needs. Therefore, the creation of a complaint reporting tool that emulates the strict structure and process required for effectively using the bug tracking tool Bugzilla would no doubt benefit the government of Toronto, as it would help them to improve their public services.

As these creative suggestions have illustrated, Toronto city government would likely benefit from looking at domains that may on first glance seem entirely unrelated to government and thinking about how their processes could be appropriated. By creating a distributed version control system for policy development, fairness will be enhanced because the power of the city councillor role will become more evenly dispersed across residents in the neighbourhood. Efficiency will be enhanced because entire neighbourhoods will be working toward resolving problems, rather than smaller groups of city councillors. Communication will be enhanced between neighbourhoods because residents will now have a tool that actively encourages them to talk to each other in order to resolve conflicts. By creating a bug tracking tool for complaint reporting, consumers of public services will be able to voice their concerns in a more organized fashion, thus leading to faster improvement of these services. For these reasons, city officials should seriously consider how the appropriation of ideas from ostensibly unrelated domains may improve not only the state of Toronto government, but the city as a whole.

REFERENCES

- Access Toronto. (2007). "Participate in your local government." [PDF file]. City of Toronto, Public Information, Chief Corporate Office. 1-28. Available at <http://www.toronto.ca>
- "Airing TTC's problems." (2010, April 8). *Toronto Star*, p. A20.
- Bolton, Marilyn. (2003). "TTC vs federal government – Court date April 22, 2003." Retrieved April 3, 2010, from <http://transit.toronto.on.ca/archives/data/200304080840.shtml>
- Bowker, G.C. & S. Star. (1999). *Sorting Things Out: Classification and its Consequences*. Cambridge, MA: MIT Press.
- Bugzilla Team. (2010). "The Bugzilla Guide – 3.5.3 Development Release." [PDF file]. 1-113. Available at <http://www.bugzilla.org/docs>
- Collins-Sussman, B., B.W. Fitzpatrick, & C.M. Pilato. (2009). "Version Control with Subversion: For Subversion 1.6." [PDF file]. Available at <http://svnbook.red-bean.com>
- Fogel, K. (2006). "Producing Open Source Software." [PDF file]. O'Reilly Media. Available at <http://producingoss.com>
- Gandy Jr., O. (2002). "The Real Digital Divide: Citizens vs. Consumers." In Lievrouw, L. & Livingstone, S. (Eds.), *Handbook of New Media*. London, UK: Sage. 448-60.
- Gilbert, L. (2004). "At the core and on the edge: justice discourses in metropolitan Toronto." *Space and Polity*, 8(2), 245-60.
- Golick, James. (2008). "Why distributed version control matters to you, today." Retrieved April 3, 2010, from <http://jamesgolick.com/2008/1/20/why-distributed-version-control-matters-to-you-today.html>
- Google. (2007). "Tech Talk: Linus Torvalds on git." [Video file]. Video retrieved on April 3, 2010, posted to <http://www.youtube.com/watch?v=4XpnKHJAok8>
- Gurstein, M. (2003). "Effective use: A community informatics strategy beyond the digital divide." *First Monday*, 8(12).
- Kern, L. (2005). "In Place and At Home in the City: Connecting privilege, safety and belonging for women in Toronto." *Gender, Place & Culture*, 12(3), 357-77.
- McVoy, Larry. (1998). "A solution for growing pains." Retrieved April 3, 2010, from [lkml.indiana.edu: http://lkml.indiana.edu/hypermail/linux/kernel/9809.3/0957.html](http://lkml.indiana.edu:lkml.indiana.edu/hypermail/linux/kernel/9809.3/0957.html)
- Raymond, E. (2001). "The Cathedral and the Bazaar." In *The Cathedral and the Bazaar: musings on Linux and open source by an accidental revolutionary*. Sebastopol: O'Reilly

Media. Retrieved April 3, 2010, from <http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar>

Tatham, S. (1999/2008) "How to Report Bugs Effectively." Retrieved April 3, 2010, from <http://www.chiark.greenend.org.uk/~sgtatham/bugs.html>

TTC Complaints. (2010). "Welcome to TTC Complaints!" Retrieved April 3, 2010, from <http://www.ttccomplaints.com>