Two-Way Streets: How Transit Developers, Agencies, and Enthusiasts are Collaborating to Build a Better Trip Planner

Nicholas Bergson-Shilcock, The Open Planning Project

GIS in Transit Conference, 11/17/09

(slide notes are not complete, nor are they meant to be read as a script)
What we do, exactly

We create software, build open source communities, and advocate for smarter, more livable cities

- GeoServer
- PostGIS
- GeoExt
- OpenLayers
- OpenTripPlanner
- Open311
- StreetsBlog.org
- StreetFilms.org
- GothamSchools
- Community Almanac
- Block Party NYC
Trip planning
I’m not actually going to talk that much about trip planning today. During the Q&A, I’m happy to talk about data structures and algorithms and the nitty-gritty aspects of how we’re coding our trip planner, but I want to focus the core part of my talk today on something else.

I want to ask a pretty broad question.
How should we develop software?
How should we develop transit software?
Based on experience developing OpenTripPlanner

My presentation will be grounded in my experiences working on the OpenTripPlanner project.
How should we develop transit software?

Want to explore this question
How are we developing OpenTripPlanner?
OpenTripPlanner:
an open source, multi-modal trip planner

First some background. What is OpenTripPlanner? Three key parts here.
OpenTripPlanner:
an open source, multi-modal trip planner

It’s a trip planner: It answers the question of how best to get from point A to point B.
OpenTripPlanner:
an open source, *multi-modal* trip planner

It’s multi-modal, which means it takes into account multiple modes of transportation -- bus, subway, walking, biking -- when figuring out what the best trip is.
OpenTripPlanner: an open source, multi-modal trip planner

It’s open source: the software is built by a community of software developers who make both the software and the underlying source code freely available for others to use, modify, and distribute.
More about OpenTripPlanner

- Written in Java
- Built using GeoTools, JTS Topology
- Standards-based (GTFS, NED, etc)
- RESTful Jersey-based API
- True multi-modal (not just bike/walk to transit)

Still a very young project.
How did it start?
Like a lot of transit innovations, it started at TriMet.
TriMet’s current trip planner. They’ve developed the front-end and interface in house, but the core router on the back end is proprietary -- a blackbox that doesn’t meet their needs. Not a true multi-modal system, doesn’t do biking directions, agency has no way to improve it.

TriMet wants to replace the routing engine with an open source trip planner that better meets their requirements.
In July, TriMet hosted a kick-off workshop for the project. It brought together half a dozen transit agencies, the lead developers of several open source trip planners, and several of us from The Open Planning Project.
In July, TriMet hosted a kick-off workshop for the project. It brought together half a dozen transit agencies, the lead developers of several open source trip planners, and several of us from The Open Planning Project.
The group spent three days talking about frustrations with existing trip planners, what features were most important, and what an ideal open source trip planner would look like.
What makes OpenTripPlanner different? It’s probably already clear that this project hasn’t followed the standard trajectory of software development.
To better highlight these differences, it's good to speak in comparison to something else. I want to quickly outline two other more traditional forms of software development to illustrate these differences.
“Traditional” open source software most frequently comes about because a developer has an itch, and he decides to scratch it himself. He wants to do something and there isn’t a good piece of free software to do it.
Linux is probably the best known example of this. It started as a single developer scratching his own itch and then posting some code to a mailing list.

This is why so many open source projects have as their leadership and governance model the "benevolent dictator." There's a founding developer who naturally runs the project since it was originally "his" baby.

As projects mature they tend to develop more sophisticated decision making processes. These can take place via informal mailing lists or even formal legal entities. But most open source at least starts with a single developer calling the shots.
With traditional proprietary software development, software is typically written because a company sees a market that they think isn't being served well or even at all. Sometimes the software is written under contract, perhaps in response to an RFP.

The developers are given a spec or design document either by their own management or by the client, a timeline, and then start working. The final product is closed: users of the software aren’t free to fix or improve the software themselves, nor are they free to hire anyone other than the original developer to fix or improve it. The software is entirely controlled by the company that developed it.
Open source. With a twist.

I think it’s more accurate to say we’re doing open source with a twist.
Freely licensed
code is free to use, modify, and distribute
Open dev process
developed by a community, not just a company
What does that really mean?
Discussion takes place on public mailing list
All are welcome to join
Tickets, milestones, timeline online for all to see

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Summary</th>
<th>Component</th>
<th>Version</th>
<th>Milestone</th>
<th>Type</th>
<th>Owner</th>
<th>Status</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Make transportation extremely easy</td>
<td></td>
<td></td>
<td>enhancement</td>
<td></td>
<td></td>
<td></td>
<td>07/17/09</td>
</tr>
<tr>
<td>#43</td>
<td>TestShapeFileLoader broken</td>
<td></td>
<td></td>
<td>defect</td>
<td></td>
<td></td>
<td></td>
<td>11/13/09</td>
</tr>
<tr>
<td>#46</td>
<td>Remove mutable methods from Vertex</td>
<td></td>
<td></td>
<td>defect</td>
<td></td>
<td></td>
<td></td>
<td>11/13/09</td>
</tr>
<tr>
<td>#47</td>
<td>Change Vertex.getOutgoing() and getIncoming() should return Iterable&lt;Edge&gt;</td>
<td></td>
<td></td>
<td>defect</td>
<td></td>
<td></td>
<td></td>
<td>11/13/09</td>
</tr>
<tr>
<td>#23</td>
<td>Bump from Java 1.5 to 1.6</td>
<td>0.1</td>
<td></td>
<td>enhancement</td>
<td></td>
<td></td>
<td></td>
<td>10/16/09</td>
</tr>
<tr>
<td>#36</td>
<td>Choose license for code</td>
<td>0.1</td>
<td></td>
<td>task</td>
<td></td>
<td></td>
<td></td>
<td>10/30/09</td>
</tr>
<tr>
<td>#16</td>
<td>OTP runs out of memory when loading large GTFS files</td>
<td>0.3</td>
<td></td>
<td>defect</td>
<td></td>
<td></td>
<td></td>
<td>09/28/09</td>
</tr>
<tr>
<td>#17</td>
<td>Port OSM importer to OTP</td>
<td>0.3</td>
<td></td>
<td>defect</td>
<td></td>
<td></td>
<td></td>
<td>09/29/09</td>
</tr>
<tr>
<td>#18</td>
<td>OTP should support NED</td>
<td>0.3</td>
<td></td>
<td>enhancement</td>
<td></td>
<td>demory</td>
<td></td>
<td>10/09/09</td>
</tr>
<tr>
<td>#41</td>
<td>FivePoints-style topo map</td>
<td>1.0</td>
<td></td>
<td>task</td>
<td></td>
<td></td>
<td></td>
<td>11/08/09</td>
</tr>
<tr>
<td>#4</td>
<td>Define public-facing HTTP API</td>
<td>0.1</td>
<td></td>
<td>enhancement</td>
<td></td>
<td>fpurcell</td>
<td>assigned</td>
<td>07/17/09</td>
</tr>
<tr>
<td>#25</td>
<td>Add narrative generation</td>
<td>0.2</td>
<td></td>
<td>enhancement</td>
<td></td>
<td>novalis</td>
<td></td>
<td>10/19/09</td>
</tr>
<tr>
<td>#26</td>
<td>Get Portland centerline file</td>
<td>0.2</td>
<td></td>
<td>task</td>
<td></td>
<td></td>
<td></td>
<td>10/19/09</td>
</tr>
</tbody>
</table>
What’s the twist?

It’s definitely an open source project, but there are two things that make it different from most other open source projects.
Founded by many, not one
OpenTripPlanner has had a strong institutional backing from the start by both TriMet and The Open Planning Project, so there's no risk of the software dying because the original developer loses interest in it.
Challenges?

What are some of the challenges we’ve faced and continue to face as a result of these differences?
With many stakeholders, who’s our audience?

This is an interesting challenge, and I think it’s mostly a good problem to have because it results from the diversity of supporters the project enjoys.

Obviously includes TriMet and other transit agencies. But we’ve also seen interest from others ranging from academia to environmental advocacy. The challenge is how how incorporate and prioritize these different interests.
Who makes decisions?

- No “benevolent dictator” because project started collaboratively
- How to bootstrap decision making process in nascent community?

One of the challenges we faced early on is that of bootstrapping the decision making process. Most open source projects start with a single developer, and he or she will call all the shots. Eventually, as the project becomes more mature and more people start contributing, a more formal structure may (or may not) develop to allow for group decision making. But OpenTripPlanner is different: It’s not any “one” developer’s project, nor is it any one company’s. It’s been a collaborative effort from the start. But new communities don’t come with built-in structure and rules.
Proposal and informal voting process

Decisions are made by the OpenTripPlanner community through a proposal and informal voting process on the project mailing list.

While we do vote on proposals, we don't vote in a strict democratic sense, but rather as a way to “easily register opinions, foster discussion, and move toward consensus.”

We borrowed this process from many open source projects that have used it effectively over the years.
Conference calls

We have weekly technical conference calls to discuss the project and more quickly reach agreement.
Should be clear that OpenTripPlanner is still young -- we're only a few months out from the official start of the project -- and it still has a long way to mature into a polished piece of software that's ready and easy to deploy. Nevertheless, the past months have made me optimistic that we will get there, sooner rather than later. Here's why:
One of the great things about open source is the kinds of people that are attracted to it. What you frequently end up with are die-hard transit-hackers. People so enamored with transit software that they choose to spend their free time writing it.

A couple of them are actually at this conference presenting. Both Brian Ferris and David Emory are developers working on OpenTripPlanner who also have their own open source transit projects (OneBusAway and Five Points). Other excellent transit developers -- including Brandon Martin-Anderson and Wyatt Baldwin, who developed Graphserver and byCycle.org, respectively -- are also integral to the project.
Built on and influenced by other excellent projects
Built on and influenced by other excellent projects
Agency backing
intimate relationship with client as partner, not developing in vacuum

Feedback cycle and close relationship with the people who will actually be using and deploying the software.

E.g., TriMet employees contribute significantly to the weekly conference calls and the developer mailing list.
Process informs product

What does this mean? The way in which we're building the software -- collaboratively, with a wide range of varied stakeholders -- will have important effects on the actual code we write.
When thinking about trip planning problems, it’s generally easiest to think about them in terms of transit systems you’re most familiar with -- typically the one in the city where you live. One advantage of having project contributors from all over the country is that it means we have people working on the project who are thinking about trip planning in terms of very different transit systems. New York City is different from Atlanta which is different from Portland.

This diversity is forcing us to keep the software flexible from the start. Minimally, it’s got us thinking about and discussing how best to handle a wide variety of systems and attributes, from the synchronized transfers of BART to the complex, multi-block station geometries of New York City.

I think the situation would be very different if we were just one company, in one city, developing the software. This advantage is natural result of the way the project is structured.
I titled this talk, "two-way streets." I didn't just mean that as a cute metaphor and pun; I actually see that as part of the core value of what I've talked about today. Transit software is complex, and I haven't met anyone who thinks they have the perfect solution. We need better software, and I think that in part means better development methods, through talking to each other, going back and forth, being willing to be flexible. Through sharing best practices.

Conferences like this are wonderful, but we only get these every two years. What if we built this kind of best-practice sharing and collaboration into our development process, communicating every couple days instead of every couple years?
I hope this talk has inspired you to think about different ways of developing transit software.

Neither the software nor the development process I've discussed today is static. They're both constantly evolving and improving, and I'd be amiss if I claimed we've solved every problem or figured out the absolute best way to do things. It's an open process, and I welcome your feedback, your questions, and your constructive criticism.
Thank you.
Keep in touch!

- The Open Planning Project: [http://openplans.org](http://openplans.org)
- Nicholas Bergson-Shilcock: nicholasbs@openplans.org @nicholasbs on Twitter
- OpenTripPlanner: [http://opentripplanner.org](http://opentripplanner.org)