# 7/21/2013

# Discussion paper regarding survey and count modules of RTC TMAP

By T. Götschi and Joshua Moody

For Tracy Hadden Loh, Director of Research, RTC

Joshua and I (TG) met a couple times and discussed the count and survey modules of TMAP. Below is an outline of our thinking, clearly work in progress, which we would like to run by you to decide on next steps. Some general thoughts (bulleted) are followed by a statement on how Joshua suggests going about programming a count app. Our main goal with this document is to introduce you to Joshua and give you a sense of what it would be like to work with him. So it is less focused on actual content of the work.

If at all possible, we would like to touch base with you by Wednesday the latest to discuss this and ideally decide on first steps (mainly with regards to count app).

Terminology:

* Survey module: everything related to tablet based intercept surveys of trail users. This module may in addition include option for online surveys.
* Count module: this includes automated and manual (app based) counts. Automated counts are not addressed here (yet). The mobile app may also include a limited survey component; a follow-up survey for the counter to record their observations of local conditions.

Key topics to tackle early on:

* Understanding the roles of each TMAP element
* Understanding use cases of TMAP
* Understanding overall architecture
* Understanding process of how to get from start to final product

In this document:

* Understanding role of survey module as part of trail travel forecast module
* Understanding role of manual count module as part of trail travel forecast module
* Some thoughts and questions on use cases
* Some thoughts on architecture of survey and count modules
* Proposal on how to proceed with survey and count modules

## Trail travel forecast module:

* Which aspects of trail travel are of interest?
  + Total usage (absolute number of miles, people, etc.)
  + Changes over time
  + Attributes of usage and users
    - Purpose, etc.
    - Age, gender, etc.
    - Perceptions
    - Spending
    - Etc.
* How can these be measured/estimated/modeled?
* What data needs to be collected and how?
* Develop a prediction model which will require a minimal amount of data collection
  + Which data is easiest to collect?
  + Which data has most prediction power?

### Survey module:

* Used to collect all manner of user information through on-site and online surveys
* Since such surveys can easily become labor intense and logistically challenging, keeping things simple and manageable is a priority for this tool (as for TMAP in general)
  + Automated data handling, transfer into database
  + Flexibility with regards to survey contents and use cases
  + Dynamic evolution of survey contents (trying to guess it right from the start is pointless – instead, the tool must be able to accommodate changes at low cost)

### Manual counts module:

* Complements continuous, automated counters on trails where these are installed
  + Provides spatial variation, e.g. on different trail sections, access points/branches
  + Provides basic attribute information, such types of users
  + Serve calibration purposes of automated counters, e.g. over/undercounting, gender split, etc.
* Serve as tool to engage volunteers and the general public in trail assessment. Crowd sourcing is an option.
* Provides basic usage data on trails without continuous counters. (This data will be time-adjusted with continuous count data from other trails)

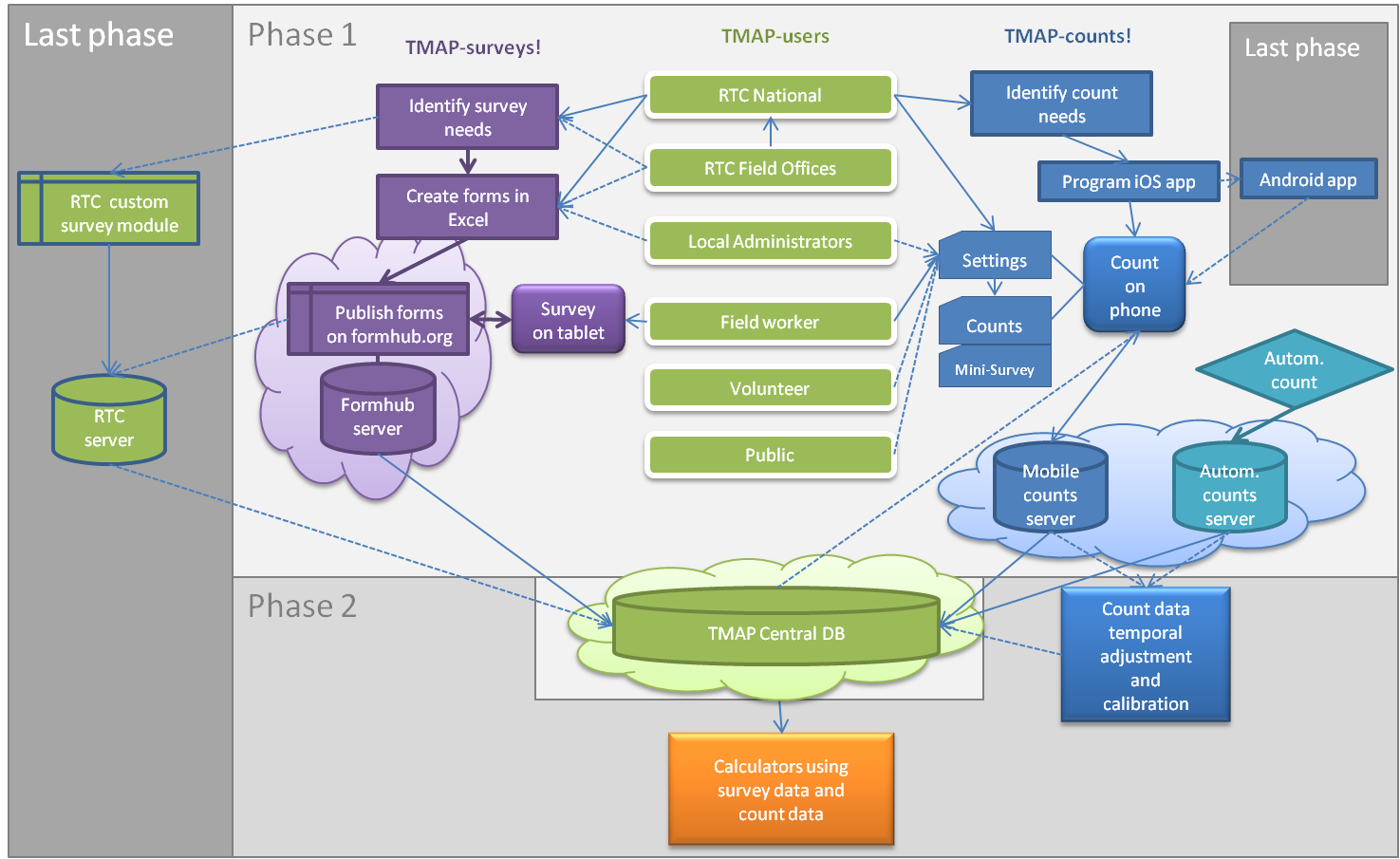
## Use cases:

A set of different use cases, what their goals are, how RTC is engaging in them should be defined

* Top priority trail campaign/project coordinated by RTC national or field offices
  + Serve to develop TMAP models and are subject to detailed TMAP applications
* Trail campaign/project coordinated by RTC national or field offices
  + Etc.
* Trail campaign/project in consultation with RTC national or field offices
* Trail campaign/project following initial detailed/in person instruction by RTC national or field offices
* Trail campaign/project initiated based on available online resources provided by RTC
* Etc.

## Initial thoughts about architecture of TMAP:

* Develop a detailed scheme which shows
  + Each data collection tool
    - Data/variables to be collected
  + Web-service or similar which link to the tool
    - Display dependences to other tools
  + Data base for storage
  + Analysis modules (calculators)
    - Input data they require
    - Outputs they produce
  + Communication modules (Reports, etc.)
    - Input data they require
    - Products they produce
* Below a draft reflecting current thinking about possible architecture.
  + - Grey phases indicate project stages
    - Dashed suggests “optional”, although don’t read 1:1



## Thoughts on development process:

* We suggest an iterative process which develops TMAP in independent, digestible pieces
* We suggest distinguishing several “grades of maturation”, from an opportunistic mix of available technologies at the beginning towards a custom designed, comprehensive RTC-branded platform at the end.
  + The main reasons for this approach are:
    - Focus on functionality and contents first
    - Identify challenges and problems using available, cheap technology rather than (mis)investing in “from scratch efforts”
    - Enable collection of data as early as possible to provide input to develop models
    - Collect user experiences, pilot tests, etc. to improve features
    - Save funds for design and programming from scratch for once when contents and features of platform are clearly defined
  + For each module, steps of maturation should be defined, to the extent possible:
    - E.g. for survey platform
      * Work with formhub.org
        + Focus on survey contents
      * Install on own server
        + Integrate with data base
      * Reprogram user interface and surveys to match RTC-TMAP, possibly integrate in iOS app, etc.
    - E.g. for manual count app
      * Build on existing iOS BikeCount app
        + Adjust features to serve TMAP
        + Enable server functionality
      * Add features, optimize to TMAP use cases
      * Create an Android version
    - Etc.
* For both survey module and manual counts we think from a technical perspective good starting points are in place with formhub.org and BikeCount app. Focus now should be on the process of identifying and describing use cases and corresponding features.
  + For case on why to go with IOS, see Joshua’s summary of points.
  + Case for building on BikeCount:
    - Several key pieces of code can be recycled.
    - Several BikeCount features will be needed for TMAP-counts
  + Case for formhub:
    - Only open source platform available for tablet forms (?)
    - Creating forms in Excel is much more efficient than with the more common menu-based user-interfaces
    - TG has quite a bit of experience with the platform by now.

## Introductory statement by Joshua

I have read the RTC T-MAP and I am excited about working on a mobile counting module for the T-MAP system. This is an informal introduction to me, my approach to software development, and my ideas about how such a module could fit into the T-MAP system.

About Me

In 2011 Thomas Götschi and I got a small grant to make BikeCount - an iPhone app and web site that lets users collect and share manual bicycle counts. It is a simple app and an even simpler web site; a proof of concept that we put together in a month or so on a shoe string budget. Despite its simplicity, it taught us a lot about what a proper bicycle counting platform should look like.

In the early 2000's, I used to bike RTC's Norwottuck Rail-Trail almost every day to commute to and from UMass Amherst and my apartment in Northampton; in the winter, when the conditions were right, I would ski. That was the first time I had ever used a bicycle trail and I loved it. Years later, I moved to Los Angeles and became involved with the exploding popularity of bicycles. I bike commuted to and from downtown LA to places like Marina del Rey and El Segundo, volunteered at the Bicyle Kitchen/La Bici Cocina, and helped organize mass bicycle rides like Midnight Ridazz. I like bikes and like what bikes and bicycle infrastructure can do for a community.

I started my software development career as a research scientist at USC's ISI studying Artificial Intelligence. I have a great deal of experience designing software for collecting data to support scientific research and I have a strong background in statistics. These days, I am a freelance iOS, MacOS, and Rails ([link](http://rubyonrails.org/)) developer. I am excited to work with RTC on the T-MAP project because I have seen how the right trail or the right trail connection can change a life or community for the better. For example, my parents live near Augusta, Maine and they went from not exercising at all to exercising daily, in no small part because the Kennebec River Trail opened near their house.

How can we make good software, on time, and within budget?

Over the years, I have worked on many software projects ranging from $25 million dollar 5-year projects for DARPA to two-week, thousand dollar contracts with individuals. I have learned a bunch of important lessons along the way and have found that traditional software engineering approaches will almost certainly fail to produce the right software, on schedule, and on budget. With a traditional approach your team would meet with me a few times to compile a list of requirements (the stuff that the software should and should not do) and then I would strike off on my own with the specs in hand and promise you that I would deliver the product you need in a couple of months for X amount of money.

It turns out that at the beginning of a software project you know the least about what the ultimate product should be. If we follow a traditional software development cycle and try to define all the requirements up front we'll likely end up with a product that took too long to make, cost too much, doesn't do the stuff you need it to do, and is hard to update.

This mobile counting module needs to be integrated with other parts of the T-MAP system - parts that don't even exist yet. Budgets and priorities are bound to change - maybe we decide to start by focusing on use the module to calibrate the automated counters, but after a couple of weeks, we realize that the supporting specific RTC campaigns need to come first.

My best, most successful, and most enjoyable software projects have been ones where I collaborate with my customers nearly every day, deliver a couple of working versions at least two times a week, and my salary was based on week-to-week productivity. I think I can hear you panicking - Thomas does every time I mention talking every day. This collaboration does not have to be hour long meetings or lengthy emails exchanges; most often they are quick chats or brief notes to make sure we are all on the same page. And we don’t have to slavishly meet every day; you can set the pace.

I propose that we adopt an Agile approach ([link)](http://en.wikipedia.org/wiki/Agile_software_development) for developing the T-MAP mobile counting module. Agile software development can be summed up in four statements ([link](http://agilemanifesto.org/)):

* **Individuals and interactions** over processes and tools
* **Working software** over comprehensive documentation
* **Customer collaboration** over contract negotiation
* **Responding to change** over following a plan

Make no mistake, the stuff on the right-hand side of those statements is important, but I believe it is much more important to focus on the things on the left.

The way we would start is with a document like this. Then we would have a couple of in-depth meetings where we talk about the broad goals of the project and finally settle on the most important features of the software. From that list of important features, we’ll decide which one you need first - that one thing that if you had it today would the biggest impact. We’ll decompose that one feature into some milestones that can each be completed in a several days. By the end of a week of work, you should have a working piece of software. Using that actual, working, functional piece of software we’ll evaluate if we are heading in the right direction and choose the next feature to work on. And so on. At every step in the process, you’ll have a working piece of software (a tangible product) and a body of source code that you can hand off to another developer.

Okay! That’s my schtick. I hope I piqued your interest. Now let’s get into some details.

Why iOS? Why not android? Why not both?

iOS holds ~40% of the mobile phone market, with android at ~48%. For tablets (iPad, Google Nexus, etc.), the market is evenly split. There is some research that shows that iOS users tend to be more active on their devices ([link](http://www.experian.com/blogs/marketing-forward/2013/05/28/americans-spend-58-minutes-a-day-on-their-smartphones/)). Okay, from these numbers looks like a tie - you could choose either platform and reach about the same number of users.

The Android platform has a large number of operating system versions that should be targeted to reach a reasonable number of users ([link](http://developer.android.com/about/dashboards/index.html)). iOS, by comparison has just 2 active operating systems to target. Less is more here - the fewer OSes you have to target, the faster and cheaper your development cycle is going to be. If you look at that link, you'll see that Android users do not update their OSes very often - 35% are using an OS that was released in 2011. The latest iOS version, 6.0, has a 93% adoption rate and iOS 5 has a 6% adoption rate. Even if you supported only iOS 6.0, you'd reach nearly all of your users.

There is a huge variety of screen sizes, pixel densities, across the various Android devices (~4,000 different models). Apple has three basic screen sizes and two pixel densities. This matters, because when you are designing your app you want to focus on the important stuff - what does this app actually do, rather than on why you can't get your images aligned correctly on the HTC 7" tablet. Apple provides a consistent look-and-feel across all devices; images on the iPhone look the same as those on the iPad. This is often not the case across Android devices and chasing down inconsistencies can be costly.

The iOS and Android market shares are nearly equal, but iOS offers some real advantages, especially initially. My advice is to start with an iOS version of your app and learn as much as you can about the app before attempting an Android version.

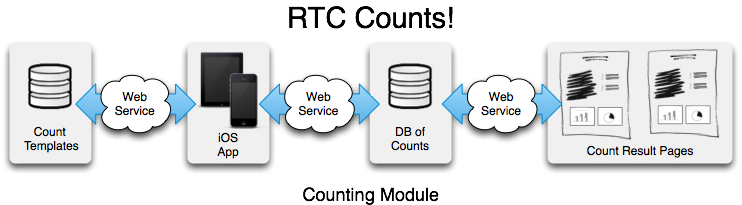
Just to be clear, the web services will be compatible with any mobile OS.

What is RTC Counts!?!

Let’s start talking about features, drawing pictures, and naming things. I just want to throw out some ideas - something that we can poke at when we meet.

'Mobile counting module for TMAP' is a mouthful, so let’s pick a name that we can use to talk about it. How about T-MAP Counts? Or RTC Counts! I kind of like that second one, so I’ll use it until something else is suggested.

What do we mean by a module? RTC Counts! will have two major pieces: a mobile application and a set of web services. The app part is pretty easy to understand - it is the app that will run on the phone or tablet. We can imagine someone standing or sitting next a trail or intersection using the app to record trail usage. But what are these web services? Here’s a simplified picture of what an architecture might look like:



The web services are the communication glue that connect the various tangible pieces (mobile app, databases, web pages) together. These services might be all one program that run on a single server alongside the database or they might be many different programs on many different servers.

Let's try to summarize what the module will do in a couple of sentences. Here is what I came up with:

'RTC Counts!' will help planners, trail administrators, and advocates count trail users and collect attributes of trail users by providing a flexible counting platform. RTC Counts! will be one of the primary public faces and tangible products of the RTC T-MAP project.

What will we count?

Here are some things we might want to count:

* walkers
* bikers
* jogger/runners
* rollerbladers
* dog walkers
* strollers

We may also be interested in distinguishing groups or categories:

* commuters/recreational users
* adults/children/families
* type of bike, helmet use, etc.
* age
* ethnicity
* gender

There are probably more things we could count, but this list seems like a pretty good start.

Who is counting?

Who do we expect will be doing these counts? Broadly speaking, I distinguish three kinds of counters: paid staff, volunteers, and everyone else. Paid staff can be sent out to specific locations with concrete count goals and they can be trained to perform complicated counts. Volunteers, on the other hand, will have varying amounts of commitment and aptitude. For example, a seventy year old volunteer might have great attention to detail, but might be frustrated by a count form that has small, hard-to-touch buttons. You might be able provide some training to volunteers and you might be able to schedule them to do counts at specific locations and times, but then again, you might not. The everyone-else category might be best described as casual counters or ad hoc volunteers. We don't know why they are counting, but we want to give them an opportunity to contribute.

But contribute to what? We'll come back to that question in a moment. It seems clear that a one-size-fits-all count form is not going to work. Even on the largest tablet, it would be impossible to fit all the things we might want to count on the screen in a way that is accessible to all of the potential users. And would we even want to try? The kind of things you want to count on the Kennebec River Trail will probably be different from what you want to count at the Bluemont Junction Trail or the NorPac Trail.

Different Kinds of Counts

There will probably be lots of different kinds of counts. On one end of the spectrum I see counts that are part of directed, highly organized campaigns that focus on answering particular usage questions across several trails. This suggests that we might want to allow users to build customized count forms that capture very specific kinds of information. At the other end of spectrum there will be one-off counts that are not part of any particular organized effort. To support these kinds of counts, we should probably have a bunch of predefined count templates that a casual counter could choose from.

## Business aspects:

* Formal contracts?
* Billing formalities?

## Next steps:

* Advancing count app seems most time sensitive. What do we need to get programming started?
* Various other issues