HOT OSM Task Manager 3 Tech Challenge

Application: Nilenso Software LLP

Introduction

Throughout this application, we hope to address all questions and concerns outlined in the Task Manager 3 (TM3) Tech Challenge (TM3TC). Our team is familiar with HOT, participates in the OpenStreetMap community, and is particularly well-equipped to handle the both the technical and non-technical challenges of the TM3TC. Our company also has a very specific vision of progressively delivering more of our work as open source, preferably around open data, with each project we take on. When we evaluate, sell, and staff projects at nilenso it is rare to come across one so well-aligned as this. We are very excited to apply to the challenge!

We have arranged our application in a slightly different order than the format of the TM3TC document or the project Terms of Reference (ToR). Because it affects absolutely every aspect of the project, collaboration and communication is mentioned multiple times in both of these documents. We feel this is a key issue, and we have started our application with this topic and how we plan to address it.

After collaboration, we brief our company's skillsets and then dive into our experience in, and approach to, project management, design, and development. We intend to staff people from each of these three roles on this project.

Finally, we will directly address how we will achieve the goals outlined in the TM3TC and present our company's qualifications for taking on such a project.

Collaboration

Nilenso is a boutique software consultancy based primarily in Bangalore (India). In our 3-year history, we have built large machine learning, streaming data, experimentation, and mapping/transportation systems. We continue to run the largest of these systems in prouction for our clients today.

No two customers are ever alike and we do not have a strict approach to collaboration with our clients' teams. Our work has moved us around the country and around the globe, operating in and across many different timezones, with many different project management styles, in a variety of toolchains[1]. Our default delivery style is based on the classic Extreme Programming methodologies with 1-week iterations, daily standups, TDD, pair programming, and heavy communication. However, our PM and communication has spanned heavier up-front design, kanban pull queues, or (in rare cases) no process at all. Some clients prefer we speak to them as much as physicaly possible... some prefer to leave us with high-level technical requirements and check in every couple weeks.

What matters most in our collaboration with customers is not the particular strategy used, project management style, or tools chosen. The essential qualities of a successful project are open lines of communication and planning/projection which employs strict measurement and reduces guesswork. The ToR Methodology bullet "A." diretly addresses this first point by identifying the varied channels which stakeholders will be employing. To take this a step further, one of our first objectives would be to document all stakeholders (or as many as possible) to ensure the graph of communication was not only open along the channels themselves, but also transparent from the outside, as a whole.

Planning and projection demands a heartbeat of some sort, and even when using a kanban pull queue to prioritize work, some period (often one week) must be used to measure progress and project into the future with the same metric. Progress on individual features/stories/cards would be updated in real time (daily) so stakeholders can watch work in-flight. The one-week period would be a summarization window for work accomplished, since daily progress is usually too narrow a window to make meaningful predictions.

See: TM3TC "Scope of Work" 1. Collaboration

[1] We are intimately familiar with most of the project management tools out there: Pivotal Tracker, Mingle, JIRA, Trello, paper-stickies-on-a-wall, and of course, Github.

The other side of the reporting and planning coin is, of course, the approval process for new work. The TM3TC is somewhat ambiguous as to whether anyone from the HOT community will take on the responsibility of owning final approval and prioritization or if this will fall to the development team. This is one area where we do strongly feel an authority is required, even if this person (or people) act as a facilitator rather than a gatekeeper. A group approval and prioritization process is possible if it is executed in real time, and we would strongly recommend one such process which we have used to great success on many projects: a project *Inception*.

An Inception is a high-intensity kickoff to a project which captures a shared understanding of the high level goals, their priorities, their budgets, and their limitations. The remainder and majority of the Inception then attempts to capture features (or "user stories", preferably, as these frame each feature in terms of how it benefits the end user), estimate them if necessary, and prioritize them. Whether stories are individually estimated or not, they are then packed into a few sample weeks of the project to rough out the expected pace of development. With this initial projection, 4 months worth of TM3 stories can be laid out in priority order and we will know roughly how much the team feels they can accomplish in that period.

Stories can be reprioritized as the project progresses and the initial projection will be updated every week with the team's actual rate of progress. The project's scope is adjusted every week accordingly.

While not every project we execute begins with an Inception, there should be some form of project kickoff which ensures stakeholders are in agreement about what they hope to achieve within their budget and timeframe and what the development team anticipates they can deliver. Inception or otherwise, this initial project kickoff should leave teh development team confident they clearly understand the deliverables and dates – and leave the stakeholders clear and unambiguous about high-level goals and what they are likely to see at the end of 4 months. If either side is confused or unhappy at this very easly stage, we would go back and clarify or re-evaluate the project. We do not start projects without the confidence we can deliver what is being asked of us.

During the project, new feature requests can of course come from the stakeholders through the various channels into a structured approval and prioritization process. If all cards/stories are ordered by priority from the beginning, it will be easy for the team to collectively prioritize (or for an authority to individually prioritize) individual stories against that total ordering.

See: TM3TC "Scope of Work" 2. Proposal / Approval of feature implementation

Another important aspect of collaboration will be that of nilenso with any other contributors from within the community. As TM3TC encourages individual community contributors to apply, we similarly want to encourage a mixed-mode award of the contract. We would be more than happy to work with a contributor(s) who has more experience with HOT or OSM in general, either as an active community member or active contributor to HOT tools.

While we are applying for the entire Scope of Work (SoW), we happily acknowledge that this may not be possible or optimal.

See: TM3TC "Scope of Work" See: TM3TC "How to Participate" 1. Identification, 3. SoW Fit

Our collaboration strategy involves user testing and user feedback of new features, solidifcation of existing features, User Experience, and design in general. While known stakeholders can be actively communicated with, one of the most important goals of TM3 is the onboarding of new Mappers. The design for a new Mapper begins outside the scope of the project itself: How did they hear about OSM? How did they learn about HOT? What do they hope to achieve? How easy/difficult is it for them and how long does it take? How can we make that entire transition smoother from the very beginning?

These questions are best addressed by our Design team and on a continuing basis. Passive feedback may involve providing an easy feedback mechanism through HOT and OSM channels, within the website or within the app. Active feedback could involve cafe testing, explicit new user solicitation exercises, and surveys.

User Experience feedback from experienced Mappers, Validators, and Project Managers will be easier, as we can reach out to them directly and afford them multiple channels to reach the TM3 team. The team can then centralize and collate this feedback and look to formalize changes (in the form of stories) based on problems identified or positive UI/UX/documentation patterns users would like to see repeated elsewhere.

One consideration for this latter case is how such an open feedback system will work once the 4 months of the TM3 project have ended. If the feedback mechanisms are too open or too ad-hoc for the community to manage, we may need to consider a more structured approach, such as a single-channel "report an issue" feature within the app itself or Github issues (though these can be offputting for less technical users).

See: TM3TC "How to Participate" 7. UI Testing and UI Feedback [Note: "UI Testing" and "QA Testing" are separate, and largely unrelated, topics and we address the latter below, under "Development"]

The last topic to address within the scope of Collaboration is scalability, sustainability, and community engagement. Scalability from an architecture perspective we will leave until the "Development" section, below. However, scalability and sustainability are also questions of collaboration with respect to language, environment, and ecosystem choices.

It is quite clear that a substantial portion of the OSM and HOT software ecosystem runs well on JavaScript and Python. TM2 is no exception, and although under "Development" we will propose two alternate approaches to the tech stack (continuing with Python or replacing the Python backend with one written in Clojure), as far as collaboration is concerned we have a few issues to address:

First is the community's appetite for alternate technologies. If the HOT/OSM community is, generally speaking, more comfortable with widely-adopted technologies, we would hate to jeopardize our application by insisting on a language with a narrower user base (even if the JVM framework itself has a much larger following).

Second is the burden we might create by adding a new language and framework to the mix. Again, we wouldn't want to excessively slow down future progress if every new contributor to TM3 had to learn Clojure from scratch.

Last is the more positive spin on the possibility of writing TM3 in a LISP on the JVM. A LISP affords a much more disciplined approach to functional programming than Python. Though the value of statelessness, immutability, concretized notions of time, and an STM can be partly captured even in object-oriented and dynamic languages like Python, Clojure provides very clear delineations between pure functions and those which produce side effects. The JVM provides a plethora of advantages, from libraries (though the Python is generally healthy in this regard) to tunable/swappable garbage collectors to plug-and-play analytics and production monitoring to pluggable profiling tools. Clojure's ecosystem has also quickly grown to be one of the most mature with respect to testing, documentation, and low-dependency modularity. All of these features make Clojure a surprisingly accessible language, even for programmers new to the environment, and an incredibly *productive* language for those who are even somewhat familiar with it.

Skillsets

Nilenso is a unique technology consultancy. While many corporations admire or actively contribute to open source and open data, they often lack an ethos which is itself compatible with the fundamental principles which make open source a wise choice in the first place. Open source has become the default mode of operation for much of the software industry, and many of us have begun to take "openness", in the form of source code at least, for granted.

Our unique position is that we are an employee-owned co-operative. While we are still a profit-making corporation, as a knowledge worker co-op we have a vision and direction which is collectively and collaboratively defined by our employees (members). Collaboration and cooperative decision-making were the very genesis of our organization. Thus, when nilenso states that we want to spend all our time building open source software in a collaborative environment, an observer can be certain that participation in Open Culture is not just a means to an end, but the organization's modus operandi.

Nilenso offers a collections of skills which will all dovetail nicely on this project. Though we are primarily a software development company, and the majority of our members are software developers, we also have a decade of project management experience and a skilled design team. All aspects of our team's skillset influence, and are influenced by, open systems: Our project management strategy is entirely about transparency, open communication, and reducing risk by failing fast. Our design team is not only familiar with the OSM ecosystem, participants of HOT, and dedicated tree mappers of Bangalore – but they understand and have worked with open data systems for many years. Openness and progress go hand-in-hand, and designing instruction-free, intuitive systems are their passion.

Our software developers are the reason for the majority of our acclaim, however. Though nilenso has only existed for 3 years, we have decades of experience delivering modular and abstract code which has evolved with the industry over the past two decades. "Reuse" for us is much more about clean design – well-defined service boundaries, appropriate library usage, and logical abstration – than it is an goal unto itself. Software and team performance both are consequences of a clean design, not the other way around. Our recent technical achievements can be read about here: http://nilenso.com/recent-tech.html

We'll next take a look at these three disciplines in order of increasing specificity: Project Management, Design, and Development.

See: “Evaluation of Proposals” 2. Reuse of Code

Project Management

Though the TM3TC requests “A timeline overview of the expected major activities and milestones over the 4 month period” (“How to Participate” 2.), this isn’t really a viable approach without a project kickoff of some sort. As an alternative, we’ll provide a sample set of stories from the requirements we’ve gathered from the TM3TC and the ToR.

We have intentionally *excluded* the nearly 300 Github issues for this exercise. Though the Github issues list provides some fantastic examples of work the project will want to prioritize, particularly given those very incisive and clearly defined issues (ex. “Prevent editing until ‘instructions’ page is read”: https://github.com/hotosm/osm-tasking-manager2/issues/890), the list is not consistent in its entirety and scrubbing all 300 issues in an attempt to come up with an exhaustive list of stories without any stakeholder participation will quickly provide diminishing returns.

Instead, the high-level requirements from the TM3TC and the ToR (as well as conversations from #task_manager_3 in Slack) provide a handy sample-sized story list which is still representative of the project goals on the whole.

The sample story list is as follows:

> *NB: Since one of the Github issues listed is “Consistency in terminology: tasks, tiles, and squares” (https://github.com/hotosm/osm-tasking-manager2/issues/912), we will use “tiles” below to mean all three.*

- Validator can query list of hints in TM3 of tiles where OSMA anticipates an error/mistake
- Mapper will be automatically notified when OSMA anticipates a common mistake on the tile the Mapper is currently editing
- PM/Validator can see # of changesets for a Mapper
- PM/Validator can see # of complete tiles for a Mapper
- PM/Validator can see days since account activation for a Mapper
- PM/Validator can see # of days of active mapping for a Mapper
- PM can view an aerial map of historicaly, current, and new project space by geography
- PM can view a map displaying active and historical projects
- PM can clone a project to create a new project with defaults

- PM/Validator can observe changes made by a Mapper in real time to provide early feedback
- Mapper can self-assign a tile to herself
- PM can assign a Mapper to a tile
- PM can mark a Mapper with a "Validator" role for a project
- PM can create a project group
- PM can add/remove projects to/from a project group
- PM can see current and historical project groups
- PM can mark a project group completed
- PM can mark a project completed
- PM can see nearby/intersecting projects on the project creation screen
- PM can see nearby/intersecting projects on the project summary screen
- Mapper can provide in-app feedback about aspects of mapping they find confusing
- Mapper can make a generic request for help to a Validator
- Validator is notified when a Mapper is identified by OSMA as needing help
- Validator can see a prioritized list of tiles requring validation
- Validator sees likely error-prone tiles highlighted by OSMA in the prioritized list of tiles requiring validation
- Validator can send Mapper quick feedback from a predefined set of common issues
- Validating a tile acknowledges the Validator for that task
- Validator can mark a tile validated in JOSM (implies corresponding API)
- Mapper can optionally add an email id to her profile
- Validator feedback is sent to Mapper's email, if available
- [EPIC] TM3 should provide an API for custom analytics queries (See ToR note "Laura O'Grady 12:38 PM Nov 24")
- PM can set expiration period on a tile; tile auto-expires at the end of the period
- PM can assign a difficulty to a tile
- PM will see auto suggestions from OSMA when assigning difficulty to a tile

- Any User can see aggregate analytics on tiles for all of OSM
- Any User can see aggregate analytics on tiles for a project group
- Any User can see aggregate analytics on tiles for a project
- Any User can see aggregate analytics on tiles for a user
- PM can see a list of Mappers for a project
- TM3 should support 100,000 simultaneous users on XYZ hardware (this card may be defined in terms of N users per M hardware nodes, assuming performance tests are published displaying linear horizontal scalability and the failure conditions for horizontal scaling)
- [NFR] Any API user can read a live data stream of activity from TM3
- PM can view a summary of projects & statuses by geographical area
- PM can adjust tile size to accommodate varying complexity/difficulty
- [NFR] Finish pending TM3 API endpoints for existing features (See TM3TC "Goals" 5.)
- Validator can ask a question of a Mapper for tile she is validating
- Mapper can view a TODO/checklist for her current tile
- Mapper can see a prioritized list of task squares available to work on
- project management proposal: 2, 6 - timeline & feature list proposal: 4 - discovery process evaluation: 5 - features (community strategy)
- design sow: [2], 6 - ux "tests" goals: 3 - ui/ux
- development sow: [2], 3, 4, 5 - development, docs, tests goals: 5 - apis proposal: 5, 7 (testing) - stack & testing evaluation: [4, 6] - scalable, sustainable shiny vs. new
- achievement of goals goals: 1 - 4 goals 5 under "development"
- who/what is nilenso? proposal: 8 - our qualifications evaluation: 1, [2], 3 - nilenso, reuse, familiarity