Software Requirements Specification

for

Super Surveyor

Version 0.1 candidate

Prepared by Baldur Thorgilsson

Pi Technology

2018-02-15

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
First draft	2018-02-15	Initial version draft sent out for comments	Version 0.1

1. Introduction

1.1 Identification

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

The software described in this document is currently called Super Surveyor and is here after referred to as **the SW**. This name might change later.
The first release of the SW is projected to have the version number 1.0. This document is to describe its functionality.
The version of this document is 0.1

1.2 Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

This template is fetched at https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc.
Defined terms are highlighted with **bolding**.
Requirements will come with priority to indicate in which order they will be implemented. Versions might be released with only some of the total list of requirements implemented.

1.3 Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

This document is written for aerial survey persons (users and testers of the system) and software developer to clarify the function and look of the software. Also, people planning surveys or using results might have wishes for functionality. Administrators of the SW will be able to perform special and rare functions with less obvious user interface.

Here is a list of collaborators to this document. The draft will be sent to this email list for feedbacks and comments.

Jack Lawson	John.Lawson@dfo-mpo.gc.ca
Debra Palka	debra.palka@noaa.gov
Jean-Francois Gosselin	Jean-Francois.Gosselin@dfo-mpo.gc.ca

Daniel Pike	kinguq@gmail.com
Gísli Víkingsson	gisli.vikingsson@hafogvatn.is
Phil Hammond	psh2@st-andrews.ac.uk
Thomas Doniol-Valcroze	Thomas.Doniol-Valcroze@dfo-mpo.gc.ca
Baldur Thorgilsson	baldur@pitemp.com

1.4 Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

(I need some input on why the surveys are performed, who wants them)

The SW is an integrated tool for aerial surveyors to plan an aerial survey, enter data during the survey and postprocess and extract information from the survey.

- Planning involves adding lines to maps.
- On board data entry involves either vocal messages (recorded by the SW) and transcribing later or real time transcribing.
- Postprocessing involves error detection, calculating covered area, and calculating densities.

The software runs on a Windows computer and gets input from hardware devices like Geometers, microphones and keyboard to present data on maps.
The SW can check data for errors and output data in files for further processing.
The SW needs to be able to run without internet connection.
Data is maintained in database and in log files.
The SW can be extended by plugins and can connect to the cloud.
The SW needs to be very robust against data loss.
The SW can calculate the densities (animals per square kilometer) for the whole survey area and subdivisions of it. It can calculate the differences between subdivisions and variation of a subdivision between years.

1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>
https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc
https://code.msdn.microsoft.com/windowsdesktop/Creating-a-simple-plugin-b6174b62

1.6 Definitions, acronyms and abbreviations

Term	**Definition**
User	A person entering or extracting data from the SW

Tester	A person testing the functionality of the SW
Developer	A person constructing the SW
Administrator	A person managing the SW (setup, update, rare or complex functions)
GPS	Global positioning system
Transect	A straight path along which one counts and records occurrences of the species of study
Waypoint	A point on the map, typically endpoints of Transects
Distance sampling	The data collected are the distances of the objects being surveyed from transects, and the objective is to estimate the average density of the objects within a region.
Aerial survey	A project that can take several days. It involves planning of transects and flight on predefined transects performing distance sampling
Segment	Sub-section of a Transect
Period of effort	
Circling	To re-fly a segment in which a sighting was made
Sighting	To enter data (vocally, by keyboard or otherwise) about an observation (can be an individual or group)
Sighting data	The data to enter for each sighting
Segment of effort	A segment where sighting has been performed
Effort	A period of active searching effort
Navigator	Transcriber
Transcriber	Person coding the information of a sighting into the SW
Observer	Person reporting sightings
g(0)	probability of detection on the transect line, usually assumed to be 1
survey status	All environmental conditions
Protolegs	A period of time (or distance?) where the survey status is the same
Leg	For practical analysis purposes, protolegs are later amalgamated into “legs”, which combine consecutive protolegs for which a subset of conditions (is condition same as survey status?) was the same
Flight	An unbroken period of transit along a transect during which all the observers were in the same position. (Is this a good word? Could be confused with an over-flight or mission?)
Records	A list of sightings, voice, GPS or other data from a survey
Effort area	The total area of a survey
Effort sub area	A part of a survey area, typically compared to other sub areas or same sub are between years.
Web portal	A web application which stores and shows data
The cloud	Web portal
DFO	
SCANS	

2. Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

The SW is a further development of a software system called VOR. Jack Lawson supplied a bundle of files that was investigated. The VOR is a collection of programs:

- The main VOR program written in C language. Main emphasis is to enter information about sightings and conditions during the survey.
- The prop02.mdb Access database with Visual basic code. The usage is for processing the data after the survey.
- The stitch.exe and circle.exe written in Fortran. These are executive files without source (one of the documents disclosed that it was written in Fortran) and are either used by one of the other programs above or run separately.
- BWCC.DLL, NAGE04.DLL, NAGSX.DLL file's function is unclear and no source found for them.

Since C-source code for VOR main program is available and one executive version is run-able, it should be possible to reconstruct the look and functionality. There is no project file, so the current source code cannot be compiled as is.
A document is available describing the usage of the Access data base, but not the process. The document mention functions like stitching, tandem and porpoise without explaining the purpose of those routines. Digging into the Visual Basic code should explain the functionality.
For stitch.exe and circle.exe, BWCC.DLL, NAGE04.DLL, NAGSX.DLL there seems to be no description and no source code to read. Here more information is needed (not necessary code, but the idea or purpose)

From the documents in the bundle, it seems like a rather complex procedure to install and run all these programs together. Also the current VOR program is not

The SW is to replace the VOR ensemble of programs into one new up-to-date program that can record and process the data as needed for a broad group of surveyors. The VOR program was designed from the standpoint of real-time transcription. Other groups like to record the sightings as audio and transcript after the overflight. Both options should be possible in the SW. Other applications like sampling over land should also be possible.

Current VOR is designed for a setup where one PC is recording from several observers. The SW should be able to do this, but also handle a situation where each observer has its own PC, tablet or phone. Time sync is accomplished e.g. by GPS.

There is a list of issues to fix for the VOR program, this list must be included in the requirements for the SW (see next chapter).

A new device for angular measurement, the Geometer, seem to be beneficial. Also, there are suggestions that the observers can enter codes for what they see, say by phone, tablet or special buttons. On-land surveys use height measurement and over-sea might use surface temperature. The SW should be able to receive data from such devices.

Maps are used for designing transects, indicating observations and presenting results. These maps are fetched by free map suppliers like Google maps or OpenStreetView.

Speech generation is a proven technology. Reading instructions or measurement results to the observers might help them in better reporting.
It would be interesting to be able to transcript audible records automatically by speech recognition technology. Most likely this is an external service only available during internet connection and might take longer time than a person interpreting the information. Automatic transcript might thus be more likely as post processing. And as speech recognition is an emerging technology, the employment might be a challenge in the noisy environment on board airplanes.

A web portal (cloud) might be beneficial to fetch previous transects, store, share and pool data between groups.

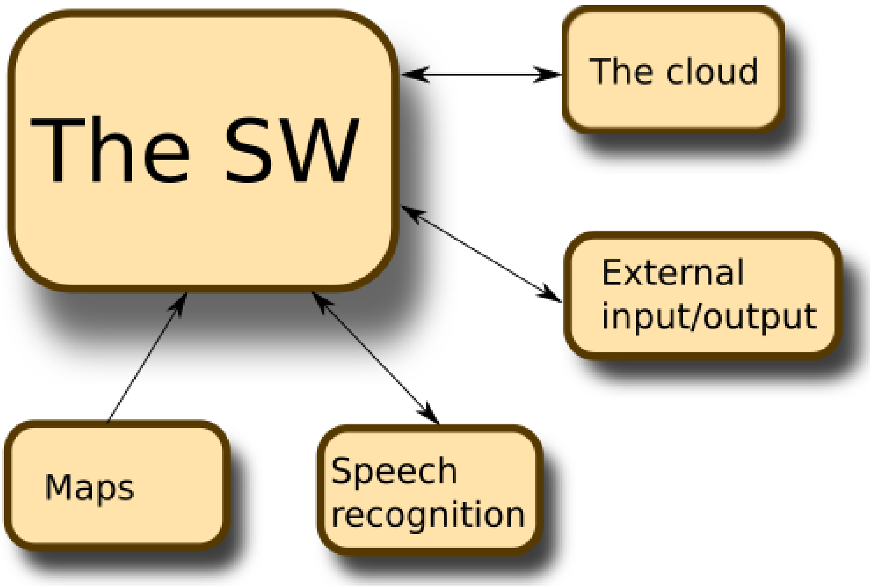

Figure 1, Software structure

For a single PC setup, each observer has a headset (microphone and speaker), a Geometer and optionally other input/output devices. If there is a transcriber on-board he also has the keyboard to the common PC running the SW. (does the pilot have to be in the audio loop?)

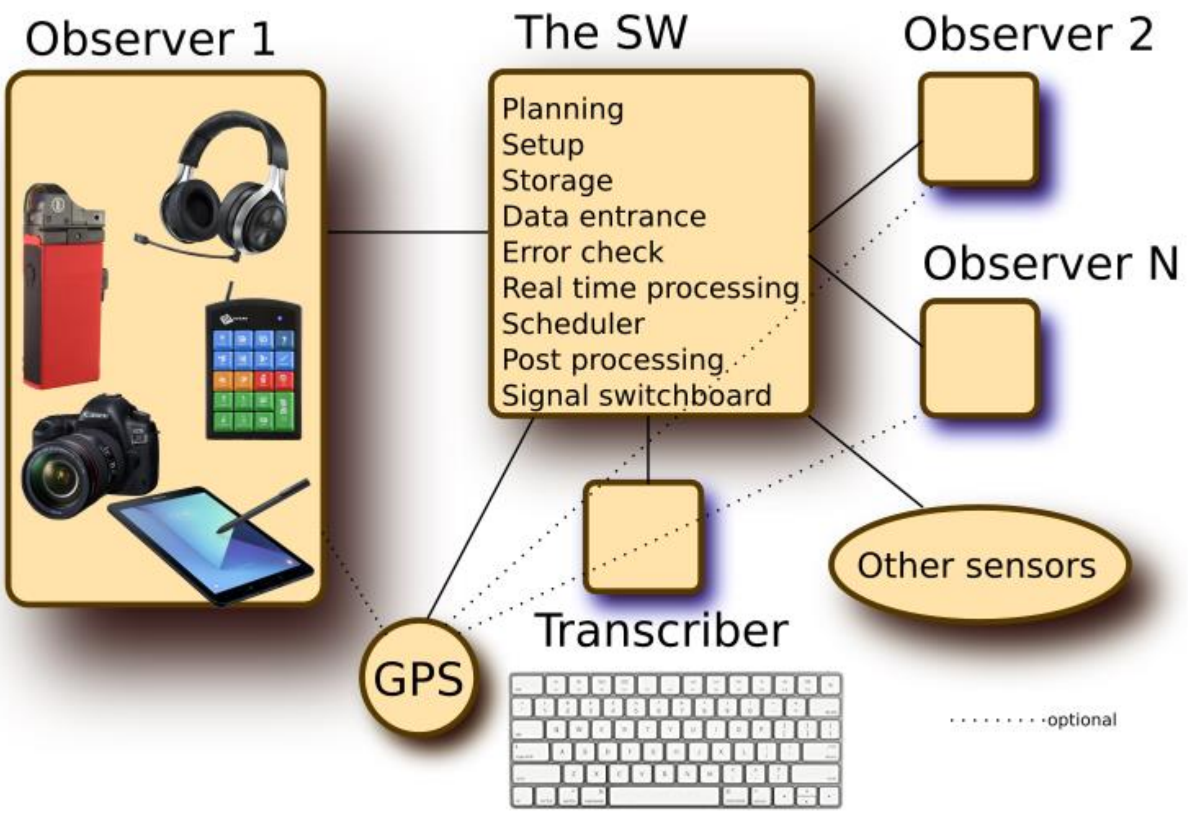

Figure 2, hardware structure

In case of one-PC-per-observer, each observer has a keyboard. In mixed mode some observers share several PC's. All the PC's need GPS for time synchronization. Any of the PC's or a separate PC (with GPS) can take the other sensors input.

In some surveys, cameras are fired on regular intervals. The SW can generate image triggers (scheduler) at a specific time, time interval or coordinate. (is this interesting? I heard this from a guy that used to do reindeer counting, here there were no observers, just cameras and images at regular intervals and all the counting was performed on ground)

Before a survey can take place, it is planned. New transects are designed in the SW and stored; or old transects retrieved from storage (e.g. cloud). The stack of transects selected for a survey is bundled in **survey** (or what is it called?). Each survey can be sub-divided into smaller sub-areas of interest (what are they called?)
One survey is completed in one or more **missions** (takeoff and landing of airplane, what is it called?)
The setup contains the **site-plan** which describes the observers (e.g. initials and place in the airplane), and who has what equipment (for signal switchboard).

While the airplane is getting to the first transect it is **off-effort**. When the airplane is at a transect and sampling has started, the state is **in-effort**.
During a mission the airplane flies along the transects. A part of a transect is called **segment**. A segment where the environmental conditions are the same is called a **protoleg**. A segment with all observers sitting in the same position is called a **flight**. (is this a common notation?)

During an effort the SW records speech, sensor signals, the sighting codes and stores them in a way that is robust to PC failure. An error checker checks if data is out of bound. Parameters for errors are editable. During effort the SW might do some real time data processing and signal switching. Transects, current position and sightings are shown in real time on a map. The signals from the sensors (including Geometer, headset, custom switches and tablets) goes through a switchboard where code translation can take place.
From the setup it is known which observer has which Geometer and headset. When a Geometer reading is generated, it can be translated into speech and fed to the observer headset that has it. This way the observer can add vocal comments about the reading. It is editable which parameters are read to the observer.

An option in the survey is to do cycles. This is a procedure where a protoleg is re-flown to generate estimate for a probability function.

There are 4 types of missions:

Missions	Real time transcribing	Off-line transcribing
Single pc setup	Type 1 (as old VOR)	Type 2 (as Daniel)
Pc-per-observer setup	Type 3	Type 4

Type 1: A transcriber enters the information about environmental conditions and other comments reported via speech from each observer.
Type 2: Transcribing is done on ground after the flight by listening to audio recordings and entering the coded data.
Type 3: Each observer also codes the conditions and adds comments in real time.
Type 4: After the over-flight the data is pooled and transcribed as for type 2.

The process of transcribing could be made automatic with speech recognition, but maybe the airplane noise makes it difficult. This is a new technology in a difficult environment that is worth investigating but is risky.

When the transcribing is finished the data is processed. The total survey area is calculated and the number of sighting for each species. From this the total density is found. The same thing is done for sub areas of the survey and comparison might be done between sub areas.
Also, the total survey or a sub area densisties can be compared to itself between years. This is done by retrieving data from the web portal or from older surveys. The result can be delivered in tables or as colors on a map.

(are there more outputs to be considered?)

When all data is considered correct, it can be uploaded to a web portal.

The Geometer can measure declination, so it substitutes mechanical inclinometers. It also has a button that makes it clear when a sighting takes place. This option might take some of the load off the transcriber. In addition, the Geometer can measure yaw. When tests have shown this to be accurate enough, it might change the way sightings are performed. In old VOR a sighting where reported when the object was abeam (in direction perpendicular to the flying direction). With the addition of yaw, sighting can take place at other times. Of cause the Geometer can still be used as before, taking the sightings at abeam and neglect the yaw.

2.2 Change requests for old VOR

Feedback 1:
“I have three versions: (1) the original version and source code, which I sent to you, (2) a version that we fly with that I paid a programmer to make a few mods to, and (3) a “new” version that Deb Palka and NOAA fly with - I am not sure what changes were made as when I open it up it seems to run the same. I stick with the version that I have, and run it on small and large boats and aircraft.

I am not sure how many people use this software. It is robust (writes every record to disc so even if there’s a crash you lose only the current record; was originally designed to run off a FLOPPY disc, so small footprint and only one small library to install).

Limitations are a few:

(1) would like it to be 64-bit so no fiddling to work with new machines (not critical)
(2) ability to map and use real digital maps (not coastal outlines as currently - that was to keep disc footprint small), plus maybe save the map and track/sightings overlays when wanted
(3) allow unlimited characters in the Comments field, plus a couple of additional columns for Heading and user-defined fields
(4) ideally, an input from a digital clinometer would trigger a new event line, and auto-insert the declination, heading and the usual GPS-based data that the software puts in now (and you could have one of the VOR windows be a clinker monitoring view such as your software does currently)
(5) maybe be able to link an event ID to an external audio recording when an observer pressed the clinometer button

These are just some ideas; for me the key would be (1), (3), and (4).

I do not think there is any copyright on this as Lex was paid years ago to create it for NOAA, and DFO has paid to have it tweaked a couple of times.

So an all-in-one solution such as I envision would make your clinometers even more useful, and allow much better navigation and situational awareness for teams in the field.
Jack”

Feedback 2:
“I agree that at this stage a fresh start might be the best.

I use the VOR programme for a component of navigation and data entry, then manually error-check and merge all the sightings and “effects” comments into the files that I then bring into Distance.

I know that Debi Palka in NOAA uses the Access database to import her VOR data. My understanding is that this Access database checks her data for errors and out-of-range codings, and I think also create the line segments needed for effort calculations in Distance. For her it seems a use of a multiple programme process to get the VOR data into the final form needed for Distance. Debi may be able to provide the missing project file, as I sent you a link to all the software files that I have.

If we could have a SINGLE survey programme that did the following, in no particular order of importance, I would be thrilled:

(1) provide a better navigation map background while flying the survey (instead of having to make custom coastline views in low resolution). If I could load the same marine navigation maps I use in my second mapping programme (and that is currently run on a second laptop and paired to a display for the pilots), then I could output this VOR display to the pilots and eliminate and entire separate computer system and cables!

(2) increase the character length of the comments fields - perhaps a moot point if we can associate an optional audio file with a sighting

(3) handle direct input from the clinometers to both trigger the new records and correctly identify MMO/MMO stations

(4) work better with USB interfaces such that multiple clinometers AND the sea surface temperature probe AND the GPS stream would all come into the system without the juggling of serial and USB interfaces (although I realize this is more of a Windows issue than VOR)

(5) an option to record audio in association with the sightings entry - this would be particularly useful when the sightings rate exceeds the speed of the data recorder, or when there is NOT a data recorder in the aircraft or vessel

(6) I have used the VOR programme on vessels as well (although not for surveying), but perhaps a “toggle” in the interface could allow the data inputs to be identified in terms of the shipboard approach as well (although in the text fields I note the position of all observers so on a vessel would still know which is primary and secondary).

(7) an “out of range” filter implemented during data entry could be something we could consider, but pop-up dialogues stating that I have mis-spelled species codes or used incorrect/improbable distance might end up being too disruptive (I think this is one of the functions of the Access programme)

I am sure that I and others can think of more features we would like to implement.

I think there are many people that would really benefit from this new system - in my opinion a highly-capable programme, offered to users for free or cheaply could be as revolutionary as the free Distance analysis programme. Since this would be useful to myself and other researchers at DFO, and depending on development costs, we could be the core funders for this effort. As we are coming up to our fiscal year end at the end of March, we will have to seek funds in April if this is a project we want to attempt.

Once we get the specifications ironed out, perhaps you could provide us with a cost estimate and timeline to produce such a new programme?

I am excited to try the new clinometers, and better yet, integrate them with a new survey programme!

regards Jack”

Feedback 3:
“When we used the circle back data collection process I used the access database. But for years I’ve just used it for a single team. So I just take the output files read them into excel where the observers edit things. We have two teams in the plane with 2 separate computers running vor. Then I read it into Splus or R to check for errors, post process to merge the two teams data and then output a format to be read by distance.

So I’m interested in your new start. I would think we could help contribute funds.

Debi”

Feedback 4:
“I have had some thoughts about this lately, about doing and transcribing vocal recordings. I am not sure this is feasible but the idea I had was that the observer would take the observations using the geometer, which would trigger an answer/response system. So, after recording the angles the observer would hear "Species?", and the system would wait until the observer said the species name. Then "Group Size?", and so on. The system would not move on until the observer gave a response. The idea here would be that the observer would not forget anything because the system would trigger the responses. But a further development would be that the responses could possibly be transcribed automatically by vocal recognition software. The observer would only be saying simple words, like "minke" or "Three" or "Ten" so recognition might not be too challenging.

That said it is a noisy environment so maybe it isn't possible. On the other hand there is good live noise cancellation available now and airplane sounds should be amenable to that.

Just an idea as it would save some transcription time and further eliminate the need for a data transcriber in the plane.

Daniel”

2.3 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high-level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top-level data flow diagram or object class diagram, is often effective.>

The SW has functions for:

- Planning surveys, that is design transcripts. This is done by adding endpoints to a map or enter coordinates.

- Sightings. The observers have headsets, Geometers and possibly other input/output devices. Information about each sighting is read loud into the microphone or entered as code (transcription). Also survey status (e.g. weather condition) and other information is stored. GPS is automatically stored. In case of real-time transcription, data is checked for error. If in doubt during transcription, images, sound files and other data at the Geometer (or other coded data) event can be found (when an event is selected, the audio recording is automatically advanced to the same time and can be played).
- Postprocessing. (please check here and change if necessary) Observation coordinates are calculated, if positions are further away than defined limits they are marked as errors (what else?), probability function is calculated if circling is used, total and sub division areas are calculated and densities for each specie for each part calculated. (what else output is needed?).

User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from

another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3. External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

Multilingual

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>
Geometers
Programmable pushbutton (e.g. sending Fxx-keys or Alt-x codes)
Headsets
GPS
Surface temperature
Video
Photos
Plane height

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>
Maps are essential part of the presentation of data. For this two main options are visible: Google Maps (GM) and OpenStreetMap (OSM) (http://geoawesomeness.com/why-would-you-use-openstreetmap-if-there-is-google-maps/).
ESRI maps

Speech recognition
Cloud interface

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

4.2 System Feature 2 (and so on)

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>