Software Requirements Specification for Live Neuro: Live Neuro

Bo Liang

February 6, 2025

Contents

Revision History

Date	Version	Notes
Feb 3, 2025	1.0	Initial Draft

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	**unit**	**SI**
t	time	second
j	dipole current vector	ampere and angle with the coordinate axis.
r	location	meter
e	sound	decibel

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the heat transfer literature and with existing documentation for solar water heating systems. The symbols are listed in alphabetical order.

symbol	**unit**	**description**
e_t	db	speech envelope
m	integer	total number of dipole sources
v_t	ampere	brain background activities

[Use your problems actual symbols. The si package is a good idea to use for units. —TPLT]

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
Live Neuro	interactive data visualization tool linking multiple data plots together
TM	Theoretical Model

1.4 Mathematical Notation

- Goal Statement
- Instance Models
- Requirements
- Introduction
- Specific System Description

2 Introduction

Neuron data based on brain activity is often complex and multi-sourced. To clearly describe the activity of different neurons, multidimensional data is typically required. An interactive data visualization tool is particularly useful for effectively presenting these data, allowing users to explore and understand the relationships and transformations between different data points.

2.1 Purpose of Document

The following section provides an overview of the Software Requirements Specification (SRS) for the interactive neural data visualization tool. The developed program will be referred to as "Live Neuro, " based on the original, manually created version. This section explains the purpose of this document, the scope of the requirements, the characteristics of the intended reader, and the organization of the document.

2.2 Scope of Requirements

The scope of requirements include the whole stimuli-response process neural data and the interactive data visualization

2.3 Characteristics of Intended Reader

Reviewers of this documentation should have an undergraduate-level understanding of mathmatics and high school level understanding of neuro science The users of Live Neuro Neuro can have a lower level of expertise, as explained in (Section 3.2)

2.4 Organization of Document

The organization of this document follows the template for an SRS for scientific computing software proposed by koothoor2013, smithLai2005, smithEtAl2007, and smithKoothoor2016. The presentation follows the standard pattern of presenting goals, theories, definitions, and assumptions. For readers that would like a more bottom up approach, they can start reading the instance models and trace back to find any additional information they require.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

Fig1 shows the system context. A circle represents an external entity outside the software, the user in this case. A rectangle represents the software system itself (Live Neuro). Arrows are used to show the data flow

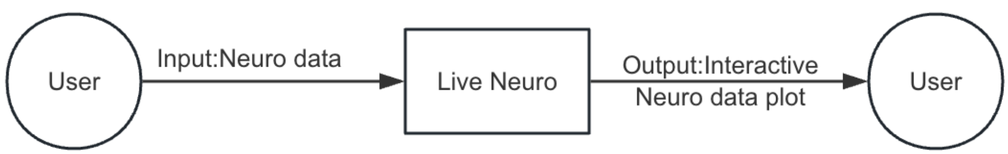

Figure 1: System Context

- User Responsibilities:
  - Provide the input data to the system, ensuring no errors in the data entry
  - Take care that consistent units are used for input variables

- Live Neuro Responsibilities:
  - Detect data type mismatch, such as a string of characters instead of a floating point number
  - Determine if the inputs satisfy the required software constraints
  - Calculate the required outputs

3.2 User Characteristics

The end user of Live Neuro should have an understanding of undergraduate mathmatics and computer science

3.3 System Constraints

There are no system constraints.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, and definitions that are used.

4.1 Problem Description

Live Neuro is intended to solve the problem of interactive neuro data visualization

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- //todo

4.1.2 Physical System Description

The physical system of Live Neuro, as shown in Figure2, includes the following elements:

PS: Current Dipoles can be used to measure the intensity of neuronal activity

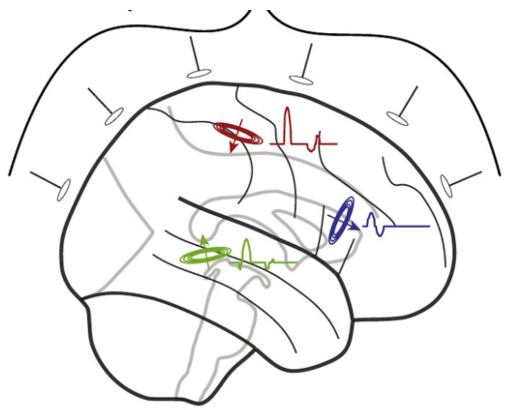

Figure 2: Current Dipoles placed on the scalp

4.1.3 Goal Statements

Given the Acoustic Stimuli and Semantics Stimuli, the goal statements are:

GS1: Predicted MEG/EEG data

GS2: Interactive neural data visualization plots

4.2 Solution Characteristics Specification

4.2.1 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [TM], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: The Stimulus data is continuous

A2: The tester is focused.

A3: The tester's mood was relatively stable before the test.

A4: Dipole sampling data is accurate

A5: The neuron data is complete

A6: The neuron data has no noise or noise has been removed

A7: The neuron data is complete with no missing parts

4.2.2 Theoretical Models

This section focuses on the general equations and laws that Live Neuro is based on.

RefName: **TM:NSM**

Label: Neural stimulation-response model

Equation: $\boldsymbol{j}_{m,t,i} = f_i\left(e_t, e_{t-1}, \cdots, e_1\right) + v_{m,t,i}$

Description: The above equation gives Neural stimulation-response model, where $\boldsymbol{j}_{m,t,i}$ is The dipole current j of source m, at time t, in the i-th direction.$v_{m,t,i}$ is the background activities of source m, at time t, in the i-th direction.$f_i\left(e_t, e_{t-1}, \cdots, e_1\right)$ is the stimulus-driven component.e_t is the speech envelop driven by acoustic and semantics stimuli

Notes: None.

Source: https://doi.org/10.1016/j.neuroimage.2020.116528

Ref. By: GD??

Preconditions for TM:NSM: None

Derivation for TM:NSM: Not Applicable

4.2.3 General Definitions

This section collects the laws and equations that will be used in building the instance models.

Number	GD1
Label	**finite impulse response**
SI Units	A
Equation	$f_i(e_t, e_{t-1}, \cdots, e_1) = \sum_{l=0}^{L-1} \tau_{m,i,l} e_{t-l} = (\boldsymbol{\tau}_{m,i})^\top \mathbf{e}_t, i \in \{R, A, S\},$
Description	Newton's law of cooling describes convective cooling from a surface. The law is stated as: the rate of heat loss from a body is proportional to the difference in temperatures between the body and its surroundings. $q(t)$ is the thermal flux ($\mathrm{W\,m^{-2}}$). h is the heat transfer coefficient, assumed independent of T (A**??**) ($\mathrm{W\,m^{-2}\,{}^\circ C^{-1}}$). $\Delta T(t) = T(t) - T_{\text{env}}(t)$ is the time-dependent thermal gradient between the environment and the object (°C).
Source	Citation here
Ref. By	DD1, DD**??**

Detailed derivation of simplified rate of change of temperature

[This may be necessary when the necessary information does not fit in the description field. —TPLT] [Derivations are important for justifying a given GD. You want it to be clear where the equation came from. —TPLT]

4.2.4 Data Definitions

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given.

Number	DD1
Label	**Heat flux out of coil**
Symbol	q_C
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q_C(t) = h_C(T_C - T_W(t))$, over area A_C
Description	T_C is the temperature of the coil (°C). T_W is the temperature of the water (°C). The heat flux out of the coil, q_C ($\mathrm{W\,m^{-2}}$), is found by assuming that Newton's Law of Cooling applies (A**??**). This law (GD1) is used on the surface of the coil, which has area A_C ($\mathrm{m^2}$) and heat transfer coefficient h_C ($\mathrm{W\,m^{-2}\,{}^{\circ}C^{-1}}$). This equation assumes that the temperature of the coil is constant over time (A**??**) and that it does not vary along the length of the coil (A**??**).
Sources	Citation here
Ref. By	IM1

4.2.5 Data Types

[This section is optional. In many scientific computing programs it isn't necessary, since the inputs and outpus are straightforward types, like reals, integers, and sequences of reals and integers. However, for some problems it is very helpful to capture the type information. —TPLT]

[The data types are not derived; they are simply stated and used by other models. —TPLT]

[All data types must be used by at least one of the models. —TPLT]

[For the mathematical notation for expressing types, the recommendation is to use the notation of **?**. —TPLT]

This section collects and defines all the data types needed to document the models. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Type Name	Name for Type
Type Def	mathematical definition of the type
Description	description here
Sources	Citation here, if the type is borrowed from another source

4.2.6 Instance Models

[The motivation for this section is to reduce the problem defined in "Physical System Description" (Section 4.1.2) to one expressed in mathematical terms. The IMs are built by refining the TMs and/or GDs. This section should remain abstract. The SRS should specify the requirements without considering the implementation. —TPLT]

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.4 to replace the abstract symbols in the models identified in Sections 4.2.2 and 4.2.3.

The goals [reference your goals —TPLT] are solved by [reference your instance models —TPLT]. [other details, with cross-references where appropriate. —TPLT] [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

Number	IM1
Label	**Energy balance on water to find T_W**
Input	m_W, C_W, h_C, A_C, h_P, A_P, t_{final}, T_C, T_{init}, $T_P(t)$ from IM**??** The input is constrained so that $T_{\text{init}} \leq T_C$ (A**??**)
Output	$T_W(t)$, $0 \leq t \leq t_{\text{final}}$, such that $\frac{dT_W}{dt} = \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))]$, $T_W(0) = T_P(0) = T_{\text{init}}$ (A**??**) and $T_P(t)$ from IM**??**
Description	T_W is the water temperature (°C). T_P is the PCM temperature (°C). T_C is the coil temperature (°C). $\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant (s). $\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless). The above equation applies as long as the water is in liquid form, $0 < T_W < 100^o$C, where 0^oC and 100^oC are the melting and boiling points of water, respectively (A**??**, A**??**).
Sources	Citation here
Ref. By	IM**??**

Derivation of ...

[The derivation shows how the IM is derived from the TMs/GDs. In cases where the derivation cannot be described under the Description field, it will be necessary to include this subsection. —TPLT]

4.2.7 Input Data Constraints

Table 1 shows the data constraints on the input output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The column for software constraints restricts the range of inputs to reasonable values. The software constraints will be helpful in the design stage for picking suitable algorithms. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise.

The specification parameters in Table 1 are listed in Table 2.

Table 1: Input Variables

Var	**Physical Constraints**	**Software Constraints**	**Typical Value**	**Uncertainty**
L	$L > 0$	$L_{\text{min}} \leq L \leq L_{\text{max}}$	1.5 m	10%

(*) [you might need to add some notes or clarifications —TPLT]

Table 2: Specification Parameter Values

Var	**Value**
L_{min}	0.1 m

4.2.8 Properties of a Correct Solution

A correct solution must exhibit [fill in the details —TPLT]. [These properties are in addition to the stated requirements. There is no need to repeat the requirements here. These additional properties may not exist for every problem. Examples include conservation laws (like conservation of energy or mass) and known constraints on outputs, which are usually summarized in tabular form. A sample table is shown in Table 3 —TPLT]

[This section is not for test cases or techniques for verification and validation. Those topics will be addressed in the Verification and Validation plan. —TPLT]

Table 3: Output Variables

Var	**Physical Constraints**
T_W	$T_{\text{init}} \leq T_W \leq T_C$ (by A??)

5 Requirements

[The requirements refine the goal statement. They will make heavy use of references to the instance models. —TPLT]

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: [Requirements for the inputs that are supplied by the user. This information has to be explicit. —TPLT]

R2: [It isn't always required, but often echoing the inputs as part of the output is a good idea. —TPLT]

R3: [Calculation related requirements. —TPLT]

R4: [Verification related requirements. —TPLT]

R5: [Output related requirements. —TPLT]

[Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM. —TPLT]

5.2 Nonfunctional Requirements

[List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. —TPLT] [The goal is for the nonfunctional requirements to be unambiguous, abstract and verifiable. This isn't easy to show succinctly, so a good strategy may be to give a "high level" view of the requirement, but allow for the details to be covered in the Verification and Validation document. —TPLT] [An absolute requirement on a quality of the system is rarely needed. For instance, an accuracy of 0.0101 % is likely fine, even if the requirement is for 0.01 % accuracy. Therefore, the emphasis will often be more on describing now well the quality is achieved, through experimentation, and possibly theory, rather than meeting some bar that was defined a priori. —TPLT] [You do not need an entry for correctness in your NFRs. The purpose of the SRS is to record the requirements that need to be satisfied for correctness. Any statement of correctness would just be redundant. Rather than discuss correctness, you can characterize how far away from the correct (true) solution you are allowed to be. This is discussed under accuracy. —TPLT]

NFR1: **Accuracy** [Characterize the accuracy by giving the context/use for the software. Maybe something like, "The accuracy of the computed solutions should meet the level needed for <engineering or scientific application>. The level of accuracy achieved by Live Neuro shall be described following the procedure given in Section X of the Verification and Validation Plan." A link to the VnV plan would be a nice extra. —TPLT]

NFR2: **Usability** [Characterize the usability by giving the context/use for the software. You should likely reference the user characteristics section. The level of usability achieved by the software shall be described following the procedure given in Section X of the Verification and Validation Plan. A link to the VnV plan would be a nice extra. —TPLT]

NFR3: **Maintainability** [The effort required to make any of the likely changes listed for Live Neuro should be less than FRACTION of the original development time. FRACTION is then a symbolic constant that can be defined at the end of the report. —TPLT]

NFR4: **Portability** [This NFR is easier to write than the others. The systems that Live Neuro should run on should be listed here. When possible the specific versions of the potential operating environments should be given. To make the NFR verifiable a statement could be made that the tests from a given section of the VnV plan can be successfully run on all of the possible operating environments. —TPLT]

- Other NFRs that might be discussed include verifiability, understandability and reusability.

5.3 Rationale

[Provide a rationale for the decisions made in the documentation. Rationale should be provided for scope decisions, modelling decisions, assumptions and typical values. —TPLT]

6 Likely Changes

LC1: [Give the likely changes, with a reference to the related assumption (aref), as appropriate. —TPLT]

7 Unlikely Changes

LC2: [Give the unlikely changes. The design can assume that the changes listed will not occur. —TPLT]

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 4 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 5 shows the dependencies of instance models, requirements, and data constraints on each other. Table 6 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

[You will have to modify these tables for your problem. —TPLT]

[The traceability matrix is not generally symmetric. If GD1 uses A1, that means that GD1's derivation or presentation requires invocation of A1. A1 does not use GD1. A1 is "used by" GD1. —TPLT]

[The traceability matrix is challenging to maintain manually. Please do your best. In the future tools (like Drasil) will make this much easier. —TPLT]

	TM??	TM??	TM??	GD1	GD??	DD1	DD??	DD??	DD??	IM1	IM??
TM??											
TM??			X								
TM??											
GD1											
GD??	X										
DD1				X							
DD??				X							
DD??											
DD??								X			
IM1					X	X	X				X
IM??					X		X		X	X	
IM??		X									
IM??		X	X				X	X	X		X

Table 4: Traceability Matrix Showing the Connections Between Items of Different Sections

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed. Figure **??** shows the dependencies of theoretical models, general definitions, data definitions, instance models, likely changes, and assumptions on each other. Figure **??** shows the dependencies of instance models, requirements, and data constraints on each other.

	IM1	IM??	IM??	IM??	4.2.7	R??	R??
IM1		X				X	X
IM??	X			X		X	X
IM??						X	X
IM??		X				X	X
R??							
R??						X	
R??					X		
R2	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R4			X	X			
R??		X					
R??		X					

Table 5: Traceability Matrix Showing the Connections Between Requirements and Instance Models

9 Development Plan

[This section is optional. It is used to explain the plan for developing the software. In particular, this section gives a list of the order in which the requirements will be implemented. In the context of a course this is where you can indicate which requirements will be implemented as part of the course, and which will be "faked" as future work. This section can be organized as a prioritized list of requirements, or it could should the requirements that will be implemented for "phase 1", "phase 2", etc. —TPLT]

10 Values of Auxiliary Constants

[Show the values of the symbolic parameters introduced in the report. —TPLT]

[The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —TPLT]

[The value of FRACTION, for the Maintainability NFR would be given here. —TPLT]

[The following is not part of the template, just some things to consider when filing in the template. —TPLT]
[Grammar, flow and LaTeXadvice:

- For Mac users `*.DS_Store` should be in `.gitignore`
- LaTeX and formatting rules
  - Variables are italic, everything else not, includes subscripts (link to document)
    - Conventions
    - Watch out for implied multiplication
  - Use BibTeX
  - Use cross-referencing
- Grammar and writing rules
  - Acronyms expanded on first usage (not just in table of acronyms)
  - "In order to" should be "to"

—TPLT]
[Advice on using the template:

- Difference between physical and software constraints
- Properties of a correct solution means *additional* properties, not a restating of the requirements (may be "not applicable" for your problem). If you have a table of output constraints, then these are properties of a correct solution.
- Assumptions have to be invoked somewhere
- "Referenced by" implies that there is an explicit reference
- Think of traceability matrix, list of assumption invocations and list of reference by fields as automatically generatable
- If you say the format of the output (plot, table etc), then your requirement could be more abstract

—TPLT]

Appendix — Reflection

[Not required for CAS 741 —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?
4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.
5. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.
6. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge

or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A?
TM??	X																		
TM??																			
TM??																			
GD1		X																	
GD??			X	X	X	X													
DD1							X	X	X										
DD??			X	X						X									
DD??																			
DD??																			
IM1											X	X		X	X	X			X
IM??												X	X			X	X	X	
IM??														X					X
IM??													X					X	
LC??				X															
LC??								X											
LC??									X										
LC??											X								
LC??												X							
LC??															X				

Table 6: Traceability Matrix Showing the Connections Between Assumptions and Other Items