Software Requirements Specification for Attitude Check: IMU-based Attitude Estimation

Adrian Sochaniwsky

January 25, 2024

Contents

Revision History

Date	Version	Notes
Date 1	1.0	Initial Release.

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**
m	length	metre
rad	angle	radian
s	time	second
Hz	frequency	hertz
T	magnetic field	tesla

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 attitude and heading reference systems literature and existing documentation. The symbols are listed in alphabetical order.

symbol	**unit**	**description**
v	m/s	linear velocity
a	m/s^2	linear acceleration
ω	rad/s	angular velocity
g_0	m/s^2	gravitational constant
b	T	earth's magnetic field

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
TM	Theoretical Model
IMU	Inertial Measurement Unit
MEMS	Micro-electromechanical System

1.4 Mathematical Notation

Vectors will be distinguised via **bold** face font such as, **b**.

Matricies will be distinguised via **bold** face font and capitalization such as, **R**.

Orientation will be represented by the quaternion,

$$q = w + ix + jy + zk \tag{1}$$

where

$$i^2 = j^2 = k^2 = ijk = -1 \tag{2}$$

please see [1] for more information.

2 Introduction

The goal of attitude estimation is to determine the rotation of an object relative to a reference frame using sensor measurements. Attitude estimation is essential for many applications, such as navigation and control of spacecraft, drones, or robots. Once only possible with expensive hardware, advances in Micro-Electro-Mechanical systems (MEMS) allow an inexpensive Inertial Measurement Unit (IMU) to provide the necessary measurements for attitude estimation. Typical IMU sensors contain a 3-axis accerlerometer, 3-axis magnetometer, and 3-axis gyroscope. However, challenges arise from the presence of noise, bias, drift, and uncertainty in the sensor data, as well as the nonlinearity of the object dynamics [2].

2.1 Purpose of Document

This document defines the scope, requirements, and functionality, of a software product. It serves as a contract between the developers and the stakeholders, ensuring that both parties have a clear and common understanding of what the software should do and how it should perform. This SRS also helps the developers to plan, design, test, and maintain the software according to the agreed-upon requirements. This document can reduce the risk of errors, delays, and conflicts during the software development process. [3]

2.2 Scope of Requirements

In the dynamics models of this project will only consider a flat local earth, and the effect of the Earth's rotation will be ignored [4].

For MEMS sensor modelling we will simplify the measurement error characteristics. Additioanlly, magnetometers can experince significant local magnetic disturbances from nearby sources such as drone motors. However, we will assume there are no disturbances.

The IMU is assumed to be mounted to a rigid body, the IMU orientation will be the orientation of the object it is attached to.

All measurements are assumed to be in the range of the sensors. For example, the accelration value of the object will not exceed the range of the accerlerometer.

2.3 Characteristics of Intended Reader

The reader should have an understanding of university-level math including matrix and vector operations, numerical methods, and state estimation.

2.4 Organization of Document

The rest of this document is structured as follows. Section 3 discusses the general context and descriptio of the system. Section 4 details the specific system description, goals, and definitions. Section 5 covers system requirements. Section 6 outlines likely changes for

the system. Section 7 discusses unlikely changes. Section 8 covers the traceability of the requirements. Section 9 discusses the system development plan.

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

Figure 1...

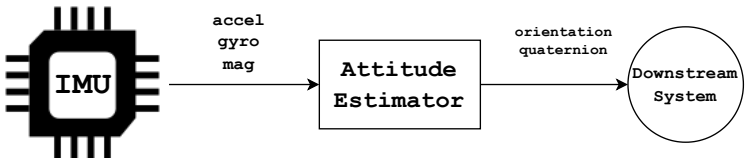


Figure 1: System Context

- User Responsibilities:
  - Provide IMU measurements.
- Attitude Check Responsibilities:
  - Detect data type mismatch, such as a string of characters instead of a floating point number.
  - Return orientation value for each set of measurements.

3.2 User Characteristics

The typical user of Attitude Check is expected to have an understanding of the purpose, inputs, and output of an attitiude estimator. This project is designed for developers of robotics and aerospace software systems looking to process IMU data. High-school level kinematics is required.

3.3 System Constraints

In typical attitude estimation system is expected to operate at high frequencies (100 Hz or more) and consume minimum computational resources. Many systems that use this kind of software are embedded systems with contrianed hardware.

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, definitions and finally the instance models.

4.1 Problem Description

Attitude Check is intended to estimate the attitude of an IMU sensor, given noisy measurements.

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:

- Coordinate frame: the frame of reference for a particular rigid body, comprised of 3 orthoganal vectors. Atitude (orientation) is the angle of an object relative to another reference coordiante frame.

4.1.2 Physical System Description

The physical system of Attitude Check, as shown in Figure 2 includes the following elements:

PS1: Accerlation measurement model.

PS2: Gyroscope measurement model.

PS3: Magnetometer measurement model.

4.1.3 Goal Statements

Given the , the goal statements are:

GS1:

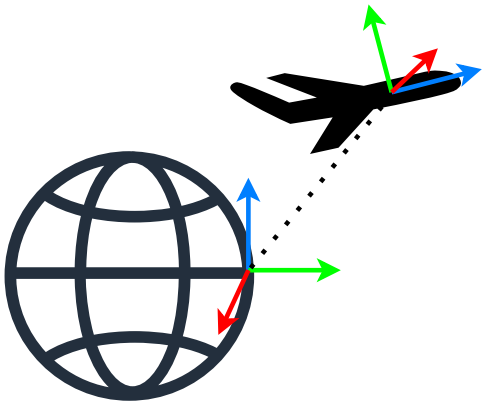
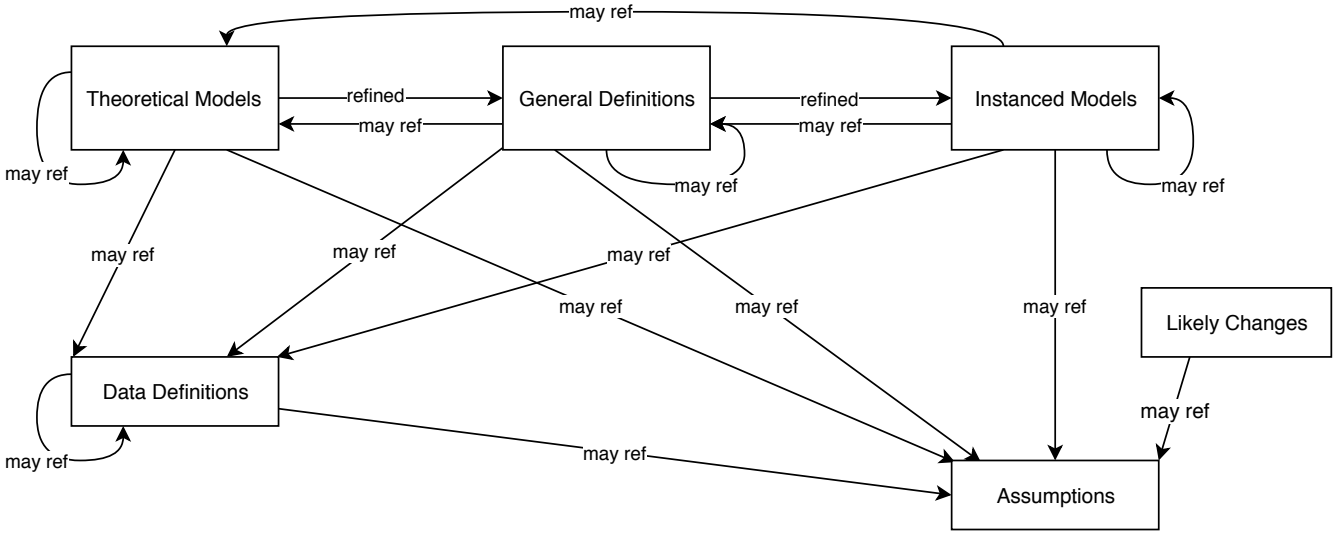


Figure 2: System Context

4.2 Solution Characteristics Specification



The instance models that govern Attitude Check are presented in Subsection 4.4.6. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Types

4.3 Scope Decisions

4.4 Modelling Decisions

4.4.1 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:

4.4.2 Theoretical Models

This section focuses on the general equations and laws that Attitude Check is based on.

RefName: **TM:COE**

Label: Conservation of thermal energy

Equation: $-\nabla \cdot \mathbf{q} + g = \rho C \frac{\partial T}{\partial t}$

Description: The above equation gives the conservation of energy for transient heat transfer in a material of specific heat capacity C ($\mathrm{J\,kg^{-1}\,{}^{\circ}C^{-1}}$) and density ρ ($\mathrm{kg\,m^{-3}}$), where $\mathbf{q}$ is the thermal flux vector ($\mathrm{W\,m^{-2}}$), g is the volumetric heat generation ($\mathrm{W\,m^{-3}}$), T is the temperature (°C), t is time (s), and ∇ is the gradient operator. For this equation to apply, other forms of energy, such as mechanical energy, are assumed to be negligible in the system (A**??**). In general, the material properties (ρ and C) depend on temperature.

Notes: None.

Source: http://www.efunda.com/formulae/heat_transfer/conduction/overview_cond.cfm

Ref. By: GD**??**

Preconditions for TM:COE: None

Derivation for TM:COE: Not Applicable

4.4.3 General Definitions

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

Number	GD1
Label	**Newton's law of cooling**
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q(t) = h\Delta T(t)$
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**??**) ($\mathrm{W\,m^{-2}\,{}^{\circ}C^{-1}}$). $\Delta T(t) = T(t) - T_{\mathrm{env}}(t)$ 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

4.4.4 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 ($\mathrm{W\,m^{-2}\,°C^{-1}}$). 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

4.4.5 Data Types

This section collects and defines all the data types needed to document the models.

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

4.4.6 Instance Models

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 4.4.4 to replace the abstract symbols in the models identified in Sections 4.4.2 and 4.4.3.

The goals are solved by .

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 ...

4.4.7 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	$L > 0$	$L_{\min} \leq L \leq L_{\max}$	1.5 m	10%

Table 2: Specification Parameter Values

Var	Value
$L_{\min}$	0.1 m

4.4.8 Properties of a Correct Solution

A correct solution must exhibit .

Table 3: Output Variables

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

5 Requirements

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:

R2:

R3:

R4:

R5:

5.2 Nonfunctional Requirements

NFR1: **Accuracy**

NFR2: **Usability**

NFR3: **Maintainability**

NFR4: **Portability**

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

5.3 Rationale

6 Likely Changes

LC1:

7 Unlikely Changes

LC2:

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.

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.

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

	IM1	IM??	IM??	IM??	4.4.7	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

	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

Figure **??** shows the dependencies of instance models, requirements, and data constraints on each other.

9 Development Plan

10 Values of Auxiliary Constants

References

[1] Bing Chat with GPT-4. attitude representation: Quaternion. https://ahrs.readthedocs.io/en/latest/quaternion/quaternion.html. Accessed: 2024-01-23.

[2] Bing Chat with GPT-4. "can you create a problem statement for a attitude estimation project". https://sl.bing.net/hakggcilB2y. Accessed: 2024-01-16.

[3] Bing Chat with GPT-4. "write a paragraph that descibes th purpose of a software requirements specification for software engineering". https://sl.bing.net/e5UwAYRf5wa. Accessed: 2024-01-23.

[4] Hussein Al-Jlailaty and Mohammad M. Mansour. Efficient Attitude Estimators: A Tutorial and Survey, December 2020. arXiv:2012.04075 [cs, eess].