Software Requirements Specification for : Concrete Remaining Life Prediction

Yi-Leng Chen

February 3, 2024

Contents

Revision History

Date	Version	Notes
February 2	1.0	Initial Documentation

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
s	time	second
°C	temperature	centigrade
A	electric current	ampere

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**
A_d	mm	Accumulated deterioration at time t_y
A_{df}	mm	Amount of damage at failure
A_{cr}	$\mu A/cm^2$	Average corrosion rate of a year
H_r	%	Relative humidity
k_d	mm/year	Factors influencing deterioration at time t_y
M_1	month	Number of months that relative humidity is below 70%
M_2	month	Number of months that relative humidity is between 70 and 100%
M_3	month	Number of months that rain occurs
n	year	Time order
R_d	m/s	Rate of concrete degradation
t_i	year	Age of the concrete at the time of condition inspection
t_r	year	Remaining service life of concrete
t_y	year	Service life of concrete
t_{yf}	year	Time to failure

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
Icorr	Corrosion rates
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
RH	Relative humidity
SRS	Software Requirements Specification
TM	Theoretical Model
UI	User Interface

2 Introduction

In response to the critical need for predicting the remaining life of concrete structures, this project aims to develop a program that implements theories introduced by a United States laboratory in 1992. Their work emphasized the importance of estimating remaining service life to enable property owners to plan for repairs or demolitions proactively. The program's goal is to predict the remaining service life of concrete structures, offering a valuable tool for effective maintenance and decision-making.

The upcoming section will present a roadmap for the Software Requirements Specification (SRS) of the program. This segment elucidates the document's purpose, outlines the scope of the requirements, and describes the characteristics of the target audience for this document.

2.1 Purpose of Document

The primary aim of this document is to outline the requirements of the Concrete Remaining Life Prediction program. This encompasses background information, goals, assumptions, theoretical models, definitions, and other details related to model derivation. These compo-

nents collectively enable the audience to gain a clear understanding and verify the purpose and scientific basis of this program.

2.2 Scope of Requirements

This document explores the impact of climate data and the level of concrete deterioration on concrete structures, with the aim of implementing a predictive program.

2.3 Characteristics of Intended Reader

The intended Reviewers of this documentation should possess an understanding of meteorology or building environments. Additionally, a basic grasp of high-school level mathematics and chemistry is recommended. For users of the prediction program can have a lower level of expertise, as further explained in the "User Characteristics" section (Section 3.2).

2.4 Organization of Document

The organization of this document follows the SRS template provided by Dr. Smith and the GlassBR SRS document.

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 illustrates the system context. In this representation, a circle signifies an external entity, which, in this case, is the user. The rectangle represents the prediction program, and arrows depict the data flow between the system and its environment.

The interaction between the product and the user is through a UI. The responsibilities of the user and the system are as follows:

- User Responsibilities: Provide the input data related to climate and concrete deterioration level, ensuring that the entry data is valid.

- Prediction Program Responsibilities: Determine if the inputs satisfy the required program constraints and predict the remaining service life of concrete.

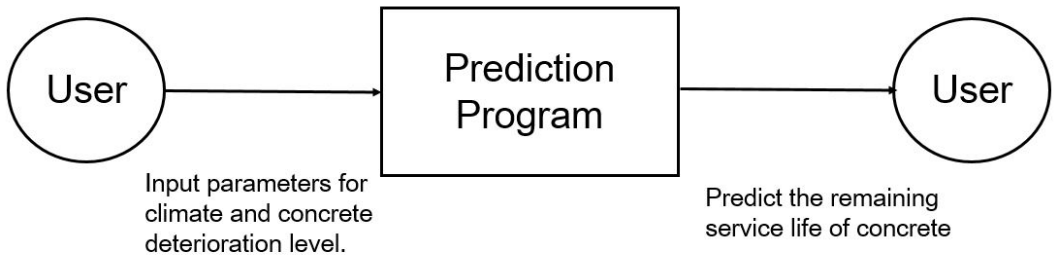

Figure 1: System Context

3.2 User Characteristics

- The end user is expected to possess basic computer literacy for effectively handling the software.
- The end user is expected to have an understanding of the theory behind concrete degradation measurement and its root causes.

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, definitions and finally the instance models.

4.1 Problem Description

A system is required to predict the remaining service life of concrete by integrating climate data and concrete degradation levels.

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:

- Corrosion rates (Icorr): It represents the flow of electric charge associated with the corrosion reactions occurring on a metal surface.

4.1.2 Physical System Description

The physical system of prediction program includes the following elements:

PS1: Concrete.

PS2: Degradation Element: The substance will attach to concrete, causing its degradation.

4.1.3 Goal Statements

Develop a prediction program that implements the theories, with the goal of predicting the remaining service life of concrete structures.

4.2 Solution Characteristics Specification

The instance models that govern this program are presented in Subsection 4.2.5. 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 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 values of k_d remain constant throughout the entire deterioration period.
A2: The only factor influencing the remaining life of concrete and causing degradation is listed in Table 1.
A3: Disregard self-conditions of concrete, such as cover thickness.
A4: Unless specifically mentioned, consider multiple degradation processes occurring simultaneously, making the time order n=1.
A5: If only one degradation process occurs, implicitly refer to Table 1 to determine the exact value of n.
A6: All information obtained from inspections is accurate.
A7: Relative humidity is the only factor that influences Icorr.
A8: Distribute based on M1 , M2 and M3 three periods.
A9: The relative humidity value needs to remain consistently below 70% for a month to be considered in the calculation of the M1 value. The same applies to the M2 and M3 values.
A10: The relationship between M1 and M2 is mutually exclusive to M3. In other words, there is no rain during the M1 and M2 periods.

4.2.2 Theoretical Models

This section focuses on the general equations and laws that this program is based on.

RefName: **TM:PredictLife**

Label: Predict remaining service life of concrete

Equation: $t_r = t_f - t_y$

Description: The above equation calculates the remaining service life of concrete.

Notes:

Source: None.

Ref. By:

Preconditions for TM:PredictLife:

Derivation for TM:PredictLife: Not Applicable

None

4.2.3 General Definitions

There are no general definitions.

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 (GD**??**) 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

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

4.2.5 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 4.2.4 to replace the abstract symbols in the models identified in Sections 4.2.2 and 4.2.3.

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

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.

4.2.6 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

Table 2: Specification Parameter Values

Var	Value
L_{min}	0.1 m

4.2.7 Properties of a Correct Solution

A correct solution must exhibit fill in the details. 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

Table 3: Output Variables

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

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

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: Requirements for the inputs that are supplied by the user. This information has to be explicit.

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

R3: Calculation related requirements.

R4: Verification related requirements.

R5: Output related requirements.

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

5.2 Nonfunctional Requirements

List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. 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. 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. 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.

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 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.

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.

NFR3: **Maintainability** The effort required to make any of the likely changes listed for 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.

NFR4: **Portability** This NFR is easier to write than the others. The systems that 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.

- 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.

6 Likely Changes

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

7 Unlikely Changes

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

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.

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.

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

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??	GD??	GD??	DD1	DD??	DD??	DD??	IM1	IM??	IM??
TM??												
TM??			X									
TM??												
GD??												
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??	4.2.6	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??																			
GD??		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 Values of Auxiliary Constants

Show the values of the symbolic parameters introduced in the report.

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

The value of FRACTION, for the Maintainability NFR would be given here.

References

The following is not part of the template, just some things to consider when filing in the template.
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"

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

Appendix — Reflection

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

1. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

2. 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.

3. 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?