Companion Cube Calculator

Geneva Smith

September 26, 2017

Contents

Table 1: **Revision History**

Date	Version	Notes
September 26, 2017	1.1	Completed the first draft of the General System Description (Section 3)
September 25, 2017	1.0	Completed the first draft of the Introduction (Section 2)

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

The tasks performed by the C^3 tool are unitless.

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_C	m^2	coil surface area
A_{in}	m^2	surface area over which heat is transferred in

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

1.3 Abbreviations and Acronyms

Text	Description
A	Assumption
CRPG	Computer Role-Playing Game
DD	Data Definition
GD	General Definition
GS	Goal Statement
GUI	Graphical User Interface
IM	Instance Model
LC	Likely Change
NPC	Non-Player Character
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
C^3	Companion Cube Calculator
T	Theoretical Model

2 Introduction

This document is an SRS for the Companion Cube Calculator (C^3), a mathematical tool which determines the range of a user-specified equation given the domains of the equation's variables. This tool is being developed to aid in the specification and refinement of GLaDOS, an emotion engine for Non-Player Characters (NPCs) in Computer Role-Playing Games (CRPG) as described by Smith (2017).

2.1 Purpose of Document

This document outlines the requirements identified for the development of the C^3 tool, including the product goals, product scope, and the mathematical models driving the design. It also describes the mathematical assumptions, theories, and models used to create the tool. The purpose of documenting this information is to aid in future use, maintenance, and development of the C^3 tool.

This document is intended for two reader types – those who wish to use the tool and those who wish to expand the tool. Even though the Companion Cube Calculator was created to aid in the development of a specific system (GLaDOS), the tested equations do not have any specific units. This means that it can be used for any equation that exists in the domain of real numbers ($\Re$). Therefore, this document can be used by a user who wishes to use the C^3 tool to determine the range of a specified mathematical equation. Since the initial development of the C^3 tool will be limited to arithmetic operators, this document includes information that will be useful to a developer looking to expand the abilities of the C^3 with additional mathematical models such as trigonometry to suit their project.

2.2 Scope of Requirements

The Companion Cube Calculator calculates the mathematical range of a user-defined equation using the defined variable domains for that equation. In the cases where a variable domain is not provided, it will be assumed that the range is $(-\infty, \infty)$. For the initial version of this product, the mathematical operations allowed in an equation will be limited to the four arithmetic operators ($+$, $-$, $\times$, $\div$, exponents). Future iterations can expand this list to include trigonometric functions (sin, cos, tan, arcsin, arccos, arctan), partial equations, and other useful operations and functions.

The purpose of this product is to aid in the design and tuning of the GLaDOS architecture, a specialized game engine enables game designers to create Non-Player Characters (NPCs) that react to their environment by using models of emotion from psychology. One module in the architecture converts information from the environment into an internal representation that directs an NPC's decision-making. This task requires the specification of numerical equations with multiple variables which must be normalized to a range of $[-1, 1]$. In order to normalize the engine's calculation, the range of the equation must be known. Currently, the encoded equations are not well-informed by observation or scientific research

and must be subjected to an iterative process in order to address this problem. An automated method of calculating the range of a proposed equation will make this process faster and less error-prone.

Although this product is being designed for a specific project, the concepts involved can be used in similar projects because it does not contain any project-specific concepts or models.

2.3 Characteristics of Intended Reader

The intended reader of this document must understand elementary algebra, especially notation and equations, in the domain of real numbers ($\Re$). They must also have an understanding of domain and range with respect to an equation in order to understand the outputs of the product and how it relates to its inputs.

2.4 Organization of Document

This document begins by describing the general description of the Companion Cube Calculator tool (Section 3), which includes the system context, constraints, and the intended users' characteristics. This is followed by a description of the problem to be solved, including the models and assumptions that are used to solve it (Section 4). This description is followed by the functional and non-functional requirements of the C^3 tool (Section 5) and the potential changes that will be made to them in future iterations (Section 6). The last section in this document, traceability matrices and graphs, visually describes the dependencies between different document components (Section 7). This information should be used by making changes to the decisions made for this version of the tool.

3 General System Description

This section identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

The Companion Cube Calculator tool is a stand-alone application for calculating the range of a user-provided equation given the known domains of the equation's component variables. Therefore it is independent and self-contained with respect to external organizations, products, and technologies.

The user interface of the C^3 tool facilitates communication between the product and the user, and must contain the ability to exchange user inputs and system outputs. In general, the user is responsible for ensuring that they have provided a semantically correct equation and that their inputs do not contain mathematical operations that are not implemented in the tool. The Companion Cube Calculator is responsible for providing an information

exchange interface between itself and the user, performing mathematical calculations, and detecting syntactic errors in the system inputs.

- User Responsibilities:
  - Provide an equation from the domain of real numbers ($\Re$) that only contains mathematical operations that are recognized by the system
  - Provide the mathematical domains of the equation's variables if they are known
  - Ensure that the equation given to the tool does not contain semantic errors
- Companion Cube Calculator Responsibilities:
  - Allow the user to enter their equation and known variable domains as system inputs
  - For variables with unknown domains, provide default values to be used in calculations
  - Detect data type mismatch, such as a string of characters instead of a floating point number
  - Detect bracket mismatches
  - Calculate and output the mathematical range corresponding to the user's inputs; if a result cannot be calculated, communicate this to the user

3.2 User Characteristics

The end user of Companion Cube Calculator is also the intended reader (Section 2.3) of this document.

3.3 System Constraints

The use of a Graphical User Interface (GUI) is useful for the C^3 tool because it can help the user visualize their equation during design time and debugging. This has implications on the types of languages that this tool will be developed in and target platform choices. If a GUI is included in the C^3 design, only Windows platforms will be supported in the initial release.

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. [Add any project specific details that are relevant for the section overview. —SS]

4.1 Problem Description

Companion Cube Calculator is [what problem does your program solve? —SS]

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:

-

4.1.2 Physical System Description

The physical system of Companion Cube Calculator, as shown in Figure ?, includes the following elements:

PS1:

PS2: ...

[A figure here may make sense for most SRS documents —SS]

4.1.3 Goal Statements

Given the [inputs —SS], the goal statements are:

GS1: [One sentence description of the goal. There may be more than one. Each Goal should have a meaningful label. —SS]

4.2 Solution Characteristics Specification

The instance models that govern Companion Cube Calculator 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 [T], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: [Short description of each assumption. Each assumption should have a meaningful label. Use cross-references to identify the appropriate traceability to T, GD, DD etc., using commands like dref, ddref etc. —SS]

4.2.2 Theoretical Models

This section focuses on the general equations and laws that Companion Cube Calculator is based on. [Modify the examples below for your problem, and add additional models as appropriate. —SS]

Number	T1
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 ($\mathrm{J\,kg^{-1}\,°C^{-1}}$) 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.
Source	http://www.efunda.com/formulae/heat_transfer/conduction/overview_cond.cfm
Ref. By	GD**??**

4.2.3 General Definitions

This section collects the laws and equations that will be used in deriving the data definitions, which in turn are used to build the instance models. [Some projects may not have any content for this section, but the section heading should be kept. —SS] [Modify the examples below for your problem, and add additional definitions as appropriate. —SS]

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	(**?**, p. 8)
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. —SS]

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. [Modify the examples below for your problem, and add additional definitions as appropriate. —SS]

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	**?**
Ref. By	IM1

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

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	**?**
Ref. By	IM**??**

Derivation of ...

[May be necessary to include this subsection in some cases. —SS]

4.2.6 Data Constraints

Tables 2 and 4 show the data constraints on the input and output variables, respectively. 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 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 2 are listed in Table 3.

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

Table 2: 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%

Table 3: Specification Parameter Values

Var	Value
L_{min}	0.1 m

Table 4: Output Variables

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

4.2.7 Properties of a Correct Solution

A correct solution must exhibit [fill in the details —SS]

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. —SS]

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

R3: [Calculation related requirements. —SS]

R4: [Verification related requirements. —SS]

R5: [Output related requirements. —SS]

5.2 Nonfunctional Requirements

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

6 Likely Changes

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

7 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 5 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 6 shows the dependencies of instance models, requirements, and data constraints on each other. Table 7 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. —SS]

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.

	T1	T??	T??	GD1	GD??	DD1	DD??	DD??	DD??	IM1	IM??	IM??	IM??
T1													
T??			X										
T??													
GD1													
GD??	X												
DD1				X									
DD??				X									
DD??													
DD??								X					
IM1					X	X	X				X		
IM??					X		X		X	X			X
IM??		X											
IM??		X	X				X	X	X		X		

Table 5: 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 6: 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??
T1	X																		
T??																			
T??																			
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 7: Traceability Matrix Showing the Connections Between Assumptions and Other Items

References

Geneva Smith. GLaDOS: Integrating Emotion-Based Behaviours into Non-Player Characters in Computer Role-Playing Games. M.A.Sc. (Software Engineering) Thesis, McMaster University, April 2017. URL http://hdl.handle.net/11375/21369.

8 Appendix

[Your report may require an appendix. For instance, this is a good point to show the values of the symbolic parameters introduced in the report. —SS]

8.1 Symbolic Parameters

$\Re$ Domain of real numbers