SHORT TITLE

YOUR THESIS TITLE, WHICH CAN BE AS LONG AS YOU WANT ON THE TITLE PAGE

BY

JANE DOE, B.Eng.

A Report

submitted to the Department You Belong To

and the School of Graduate Studies

of McMaster University

in partial fulfilment of the requirements

for the degree of

Masters of Engineering

Masters of Engineering (YYYY) McMaster University

(Department You Belong To) Hamilton, Ontario, Canada

TITLE:	Your Thesis Title, Which Can Be As Long As You Want On the Title Page
AUTHOR:	Jane Doe B.Eng. (Software Engineering & Game Design), McMaster University, Hamilton, Canada
SUPERVISOR:	Your Supervisor
NUMBER OF PAGES:	xi, 12

Lay Abstract

A lay abstract of not more 150 words must be included explaining the key goals and contributions of the thesis in lay terms that is accessible to the general public.

Abstract

Abstract here (no more than 300 words)

Your Dedication

Optional second line

Acknowledgements

Acknowledgements go here.

Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Notation

$A \leq B$	A is less than or equal to B

Definitions

Challenge	With respect to video games, a challenge is a set of goals presented to the player that they are tasks with completing; challenges can test a variety of player skills, including accuracy, logical reasoning, and creative problem solving

Abbreviations

AI	Artificial intelligence

Declaration of Academic Achievement

The student will declare his/her research contribution and, as appropriate, those of colleagues or other contributors to the contents of the thesis.

Chapter 1

Introduction

Scientific computing (SC) is an intersection of computer science, mathematics, and science. It is a field that solves complex scientific problems by using computing techniques and tools. Writing documentation is a process of developing scientific software. The role of documentation is to help people better understand the software and to "communicate information to its audience and instil knowledge of the system it describes" [1]. The significance of software documentation has been presented in many papers by previous researchers [2], [3], [4]. It is further shown by Smith et al. [5], [6] that developing scientific computing software (SCS) in a document-driven methodology improves the quality of the software .

Jupyter Notebook is a system for creating and sharing data science and scientific computing documentation. It is a nonprofit, open-source application born out in 2014, providing interactive computing across multiple programming languages, such as Python, Javascript, Matlab, and R. A Jupyter Notebook integrates text, live code, equations, computational outputs, visualizations, and multimedia resources, including images and videos. Jupyter Notebook is one of the most widely used interactive

systems among scientists. Its popularity has grown from 200,000 to 2.5 million public Jupyter Notebooks on GitHub in three years from 2015 to 2018 [7]. It is used in a variety of areas and ways because of its flexibility and added values. For example, the notebook can be used as an educational tool in engineering courses, enhancing teaching and learning efficiency [8], [9].

Even though the importance of documentation is widely recognized, it is often missing or poorly documented in SCS because: i) scientists are not aware of the why, how, and what of documentation [10], [11]; ii) it is time-consuming to produce [12]; iii) scientists generally believe that writing documentation demands more work and effort than they would likely yield in terms of the benefits of it [13].

We are trying to increase the efficiency of documentation development by adopting generative programming. Generative programming is a technique that allows programmers to write the code or document at a higher abstraction level, and the generator produces the desired outputs. Drasil is an application of generative programming, and it is the framework we use to conduct this research. Drasil saves us more time in the documentation development process by letting us encode each piece of information of our scientific problems once and generating the document automatically.

1.1 Background

1.1.1 Drasil

Drasil is a framework that can generate software artifacts, including Software Requirement Specifications (SRS), code (C++, C#, Java, and Python), README, and

Makefile, from a stable knowledge base. The goals of Drasil are reducing knowledge duplication and improving traceability [14]. Drasil captures the knowledge through our hand-made case studies. We currently have 10 case studies that cover different physics problems, such as Projectile and Pendulum. Recipes for scientific problems are encoded in Drasil, and it generates code and documentation for us. Each piece of information only needs to be provided to Drasil once, and that information can be used wherever it is needed (note: add an example). SRS is a template for designing and documenting scientific computing software requirement decisions created by Smith et al [15]. Drasil is capable of generating SRS in document languages HTML and LaTeX. We are looking to extend the capability of Drasil by generating Jupyter Notebook in Drasil.

1.1.2 Jupyter Notebook

Jupyter Notebook is an interactive open-source web application for creating and sharing computational science documentation that contains text, executable code, mathematical equations, graphics, and visualizations.

Structure of a notebook document

A Jupyter Notebook has two components: front-end "cells" and back-end "kernels". The notebook consists of a sequence of cells: code cells, markdown cells, and raw cells. A cell is a multiline text input field. The notebook works by users entering a piece of information (text or programming code) in cells from the web page user interface. That information is then passed to the back-end kernels which execute the code and return the results [16].

Values of Jupyter Notebook

There are several values of Jupyter Notebook: sharable, all-in-one, and live code. First of all, the notebook is easy to share because it can be converted into other formats such as HTML, Markdown, and PDF. Secondly, it combines all aspects of data in one single document, making the document easy to visualize, maintain and modify. In addition, Jupyter Notebook provides an environment of live code and computational equations. Usually, when programmers are running code on some other IDEs, they have to write the entire program before executing it. However, the notebook allows programmers to execute a specific portion of the code without running the whole program. The ability to run a snippet of code and integrate with text highlight the usability of the notebook.

1.2 Problem Statement

Since both Jupyter Notebook and Drasil focus on creating and generating scientific computing documentation, we are interested in extending the values of Jupyter Notebook to Drasil and the kind of knowledge we can manipulate. Following are the three main problems we are trying to solve with Drasil in this paper:

1. Generate documents in JSON format. Jupyter Notebook is a simple JSON document with a .ipynb file extension. A Drasil-generated document needs to have a format that is readable by the notebook. Drasil can only write in HTML and LaTeX; we will need Drasil to be able to generate documents in JSON format so that the documents can be read and written in Jupyter Notebook.

2. Develop the structure of lesson plans and generate them. Not only can Drasil

generate SRS, but we would also like to expand its application by generating lesson plans. As mentioned, Jupyter Notebook is used as an educational tool for teaching engineering courses. We are interested in teaching Drasil a "textbook" structure by starting with generating a simple physics lesson plan. We are trying to capture the elements of textbook chapters, find the family of lesson plans, and classify the knowledge in order to build a general structure in Drasil so that the lesson plan will be able to generalize to a variety of lessons.

3. Generate an interactive notebook. Jupyter Notebook is an interactive application for creating documents that contain formattable text and executable code. However, Drasil doesn't support interactive recipes. We are looking for the possibility of generating a notebook document with a mix of text and code, increasing the ability of Drasil and its potential to solve more scientific problems.

1.3 Thesis Outline

Thesis outline here.

Chapter 2

Conclusion

Every thesis also needs a concluding chapter

Appendix A

Your Appendix

Your appendix goes here.

Appendix B

Long Tables

This appendix demonstrates the use of a long table that spans multiple pages.

Col A	**Col B**	**Col C**	**Col D**
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D

Continued on the next page

Continued from previous page

Col A	**Col B**	**Col C**	**Col D**
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D

Bibliography

[1] Andrew Forward. *Software documentation: Building and maintaining artefacts of communication.* University of Ottawa (Canada), 2002 (cit. on p. 1).

[2] David Lorge Parnas. "Precise documentation: The key to better software". In: *The Future of Software Engineering* (2011), pp. 125–148 (cit. on p. 1).

[3] Vikas S Chomal and Jatinderkumar R Saini. "Significance of software documentation in software development process". In: *International Journal of Engineering Innovations and Research* 3.4 (2014), p. 410 (cit. on p. 1).

[4] Noela Jemutai Kipyegen and William PK Korir. "Importance of software documentation". In: *International Journal of Computer Science Issues (IJCSI)* 10.5 (2013), p. 223 (cit. on p. 1).

[5] Koothoor Nirmitha and Smith Spencer. *Developing Scientific Computing Software: Current Processes and Future Directions.* 2016. URL: `http://hdl.handle.net/11375/13266` (cit. on p. 1).

[6] Yu Wen and Smith Spencer. *A Document Driven Methodology for Improving the Quality of a Parallel Mesh Generation Toolbox.* 2007. URL: `http://hdl.handle.net/11375/21299` (cit. on p. 1).

[7] Jeffrey M. Perkel. "Why Jupyter is data scientists' computational notebook of choice". In: *Nature* 563 (2018), pp. 145–146 (cit. on p. 2).

[8] Alberto Cardoso, Joaquim Leitão, and César Teixeira. "Using the Jupyter notebook as a tool to support the teaching and learning processes in engineering courses". In: *The Challenges of the Digital Transformation in Education: Proceedings of the 21st International Conference on Interactive Collaborative Learning (ICL2018)-Volume 2*. Springer. 2019, pp. 227–236 (cit. on p. 2).

[9] Pengfei Zhao and Junwei Xia. "Use JupyterHub to Enhance the Teaching and Learning Efficiency of Programming Related Courses". In: (2019) (cit. on p. 2).

[10] Sibylle Hermann and Jörg Fehr. "Documenting research software in engineering science". In: *Scientific Reports* 12.1 (2022), p. 6567 (cit. on p. 2).

[11] Jiyoo Chang and Christine Custis. "Understanding Implementation Challenges in Machine Learning Documentation". In: *Equity and Access in Algorithms, Mechanisms, and Optimization*. 2022, pp. 1–8 (cit. on p. 2).

[12] Rebecca Sanders and Diane Kelly. "Dealing with risk in scientific software development". In: *IEEE software* 25.4 (2008), pp. 21–28 (cit. on p. 2).

[13] Spencer Smith, Thulasi Jegatheesan, and Diane Kelly. "Advantages, disadvantages and misunderstandings about document driven design for scientific software". In: *2016 Fourth International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering (SE-HPCCSE)*. IEEE. 2016, pp. 41–48 (cit. on p. 2).

[14] Contributors of Drasil. *Drasil - Generate All the Things!* URL: `https://jacquescarette.github.io/Drasil/` (cit. on p. 3).

[15] W Spencer Smith and Lei Lai. "A new requirements template for scientific computing". In: *Proceedings of the First International Workshop on Situational Requirements Engineering Processes–Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP*. Vol. 5. Citeseer. 2005, pp. 107–121 (cit. on p. 3).

[16] GitHub contributors. *The Jupyter Notebook*. URL: `https://jupyter-notebook.readthedocs.io/en/stable/notebook.html` (cit. on p. 3).