To improve understandability, maintainability and reproducibility

This is too broad. Give examples of why the papers consider documentation significant.

Chapter 1

Introduction

Scientific C~~c~~omputing (SC) is ~~an~~ at the intersection of computer science, mathematics, and science. ~~It is a field that solves~~ SC analyses and simulates mathematical models of complex scientific and engineering problems by using computing techniques and tools. ~~W~~writing documentation ~~is a~~ should be part of the 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],~~ [2--4] [4]. ~~It is further shown by~~ Smith et al. [5], [6] shows that developing ~~scientific computing~~ SC software ~~(SCS)~~ in a document-driven methodology potentially improves the quality of the software.

You need a sentence to connect these paragraphs. You have argued that doc is important, you can then say that a frequently used approach to document SC software is to use Jupyter notebooks.

Jupyter Notebook is a system for creating and sharing data science and scientific computing documentation and code. It is a nonprofit, open-source application born ~~out~~ in 2014~~,~~. Jupyter Notebook provid~~ing~~es 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~~ Notebooks can be used as an educational tool in engineering courses, enhancing teaching and learning efficiency [8], [9].

At the end of section intro blurb give a "roadmap" for the chapter.

Even though the importance of documentation is widely recognized, it is often missing or poorly ~~documented~~ realized 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].

The connection to Jupyter is missing.

Insert a paragraph here saying the benefits of Jupyter notebooks for making SC documentation easier to maintain because the code and documentation are maintained together, etc. → then point out that even Jupyter notebooks are problematic because of the time to create them, and the difficulty of making large changes to the code. This motivates Drasil!

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

Put text b/w headings. You could give a roadmap of the sub-sections

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 motion and double Pendulum simulation. Recipes for scientific problems are encoded in Drasil, ~~and~~ which then 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. The SRS is built using a template for designing and documenting scientific computing software requirements as ~~decisions~~ created by Smith et al [15]. Drasil is capable of generating can SRS in the document languages HTML and LaTeX. We are looking to extend the capability of Drasil by generating Jupyter Notebooks in Drasil.

add an example

T good idea ✓

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].

→ point to a figure with a small sample of a Jupyter notebook

The Value of Jupyter Notebook

There are several advantages 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.

Is there a source you can cite on the benefits of Jupyter?

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 Jupyter Notebooks. To acheive this, we will have to generate documents in notebook format. Jupyter Notebook is a simple JSON document with a .ipynb file extension. Notebook contents are either code or Markdown. Therefore, non-code contents must be in Markdown format with JSON layout. Currently Drasil can only write in HTML and LaTeX. We are building a notebook printer in Drasil for generating documents that are readable and writable in Jupyter

Notebook.

2. Develop the structure of lesson plans and generate them. As mentioned, Jupyter Notebook is used as an educational tool for teaching engineering courses. When it comes to teaching, lesson plans are often brought up because they help teachers to organize the daily activities in each class time. We are interested in teaching Drasil a “textbook” structure by starting with generating a simple physics lesson plan and expanding Drasil’s application. We aim to capture the ✓ elements of textbook chapters, identify the family of lesson plans, and classify the knowledge to build a general structure in Drasil, which will enable the lesson plan to generalize to a variety of lessons.

Currently

3. Generate notebooks that mix text and code. Jupyter Notebook is an interactive application for creating documents that contain formattable text and executable code. However, Drasil doesn’t support interactive recipes. There is no code in SRS documents, and text and code are generated separately in Drasil. We are looking for the possibility of generating a notebook document that incorporate both text and code, thereby enhancing the capabilities of Drasil and its potential to solve more scientific problems.

1.3 Thesis Outline

Thesis outline here. Yes