Deliverable 1

Team_01 (GitHub):
Abel Debalkew
Carl Alvares
Jaya Thirugnanasampanthan
Kemar Harris
Lan Yao
Seemin Syed

Table of Contents

Software Architecture 3

- Overview 3
- Interesting Aspects 5
- Quality 5
- Improvements 6

Chosen Software Development Process 7

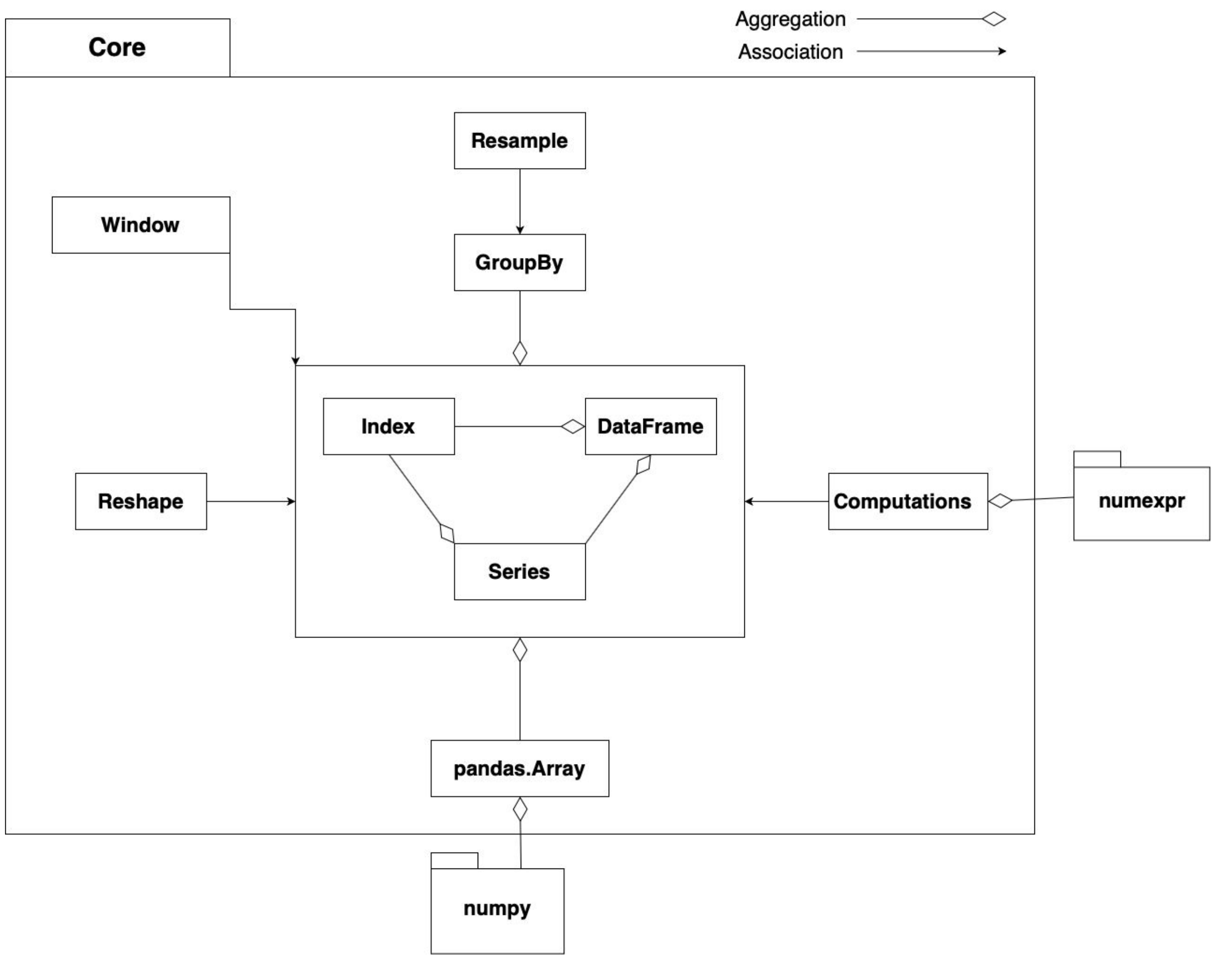
- WIP Limits & Its Benefits 8
- Kanban Board & Its Benefits 8
- Kanban’s Flexibility 9
- Drawbacks to Kanban 9
- Modifications Made To Kanban 9
- Reuse-Oriented Model: Process Activities 10
- Drawback to the Reuse-Oriented Model 11
- Modification Made to the Reuse-Oriented Model 11

Consideration of Other Software Development Processes and Models 11

- Extreme Programming (XP) 11
- Plan-Driven Development/Waterfall Model 11
- Rational Unified Process (RUP) 12
- Scrum 12
- Spiral Model 12
- V-Model 12
- Conclusion 12

Software Architecture

Figure 1



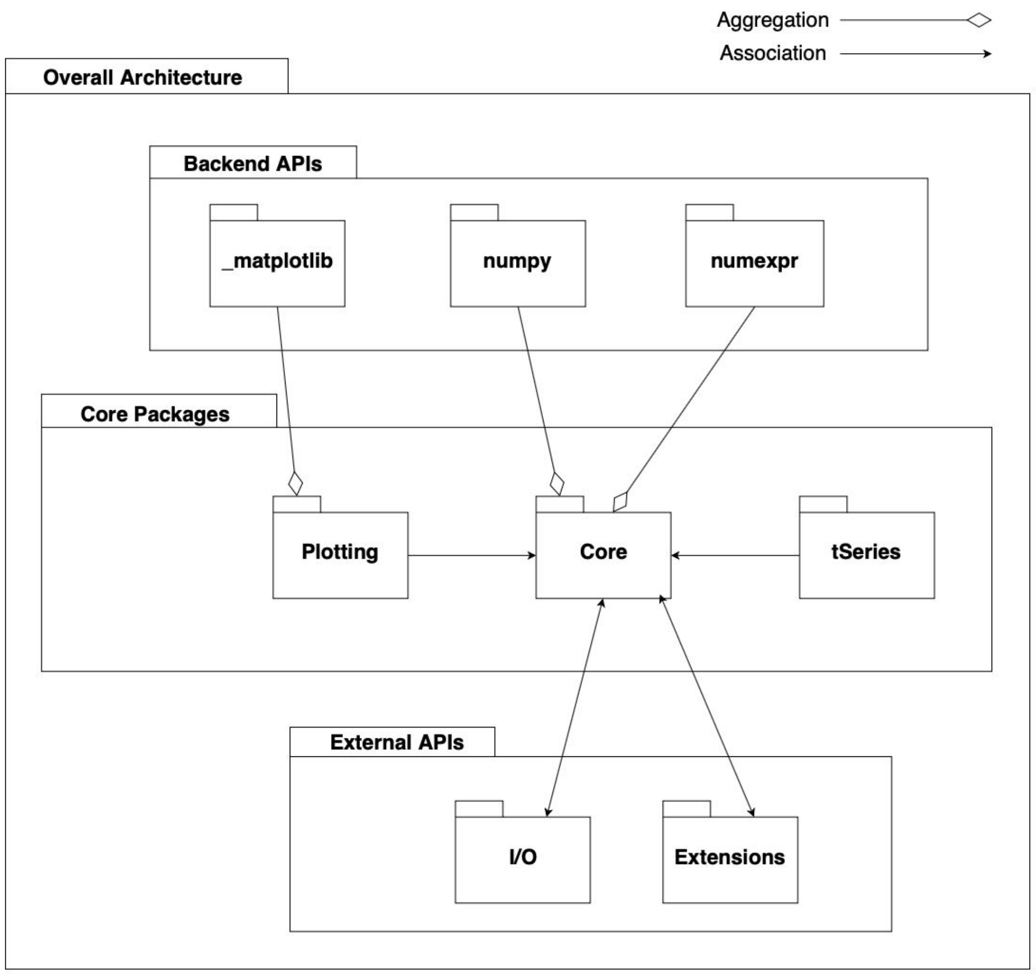
Overview

The core package is where the bulk of pandas' functionality is contained (Figure 1). At the base of this is Series, a pandas data structure which represents one-dimensional cross-sectional and time series data. Series extends the pandas Array object, which in turn is an extension of the Array object in the numpy library, a dependency of pandas. The DataFrame data structure is a dictionary of Series objects, and as such, is a two-dimensional container. Both Series and DataFrame make use of the different Index objects, depending on the type of data being analyzed, in order to characterize their data (for example, with labels). These basic objects are used throughout pandas.

Also within the core package are other classes which provide functionality for these basic objects. Reshape is a collection of data manipulation routines, such as pivot and merge, which can be used on the basic objects. GroupBy operations allow data within Series and DataFrame objects to be split, manipulated, and then grouped. Resampling contains functionality to repeatedly take samples from data contained in the basic objects or in

GroupBy objects. Computations make use of the numexpr library (another dependency) to perform efficient calculations and evaluations of Python expressions within pandas, mainly for use on columns in DataFrame objects. The Window class contains functions that allow for the calculation of statistics from multiple types of subsets of a larger dataset. These functions can be used on the basic objects or GroupBy objects.

Figure 2



Outside of the core, pandas is split into three layers (Figure 2). There is a backend layer, where libraries that pandas depends on are contained. In the core packages layer, there are the main APIs that users will interact with when using pandas. Finally, there are the external APIs, which allow pandas objects and functionalities to be imported, exported and extended to and from other sources.

In the core layer, there are the plotting, core, and tseries packages. The core package is described above. The plotting folder provides the API for using Matplotlib functionality in order to plot data contained in Series and DataFrame objects. It also provides an abstraction layer between Matplotlib and pandas, creating a separation between Matplotlib on the backend and this API, which is in the core layer of pandas. Matplotlib is meant to be a soft dependency. The tseries package separates calculation of time related data entirely from other types of data. It provides classes and implementations of time related concepts, such as weeks, months, business days, etc.

The backend layer contains the _matplotlib, numpy, and numexpr packages. Matplotlib is a library used for plotting data. pandas uses it to provide viewers with a visualization of their data. Numpy is a package in python for scientific computing with Python. pandas bases its main structures, Series and DataFrame, on this library. This allows pandas to focus its core functionality on shaping, modifying, and viewing data, while leaving the core computing functionality to numpy. Numexpr is a numerical expression evaluator for numpy. Expressions that operate on arrays are accelerated when using numexpr. Python can be slow when evaluating complex expressions on data structures like DataFrames, and since pandas focuses on data manipulation, numexpr is used in conjunction with numpy to speed up calculation.

Lastly, the External API layer has input/output and extensions packages. The input/output module provides functionality to read and write data from pandas to and from various sources, such as CSV, HTML, Excel, and more. It also allows pandas objects to be converted to and from other formats. The input/output module interacts with the core module, as this is where objects it converts to and from are defined. The extensions package allows users to extend pandas objects in their own libraries and projects.

Interesting Aspects

An interesting part of the pandas architecture that the team found was that the plotting module allows for the use of third-party backends instead of the default (Matplotlib), effectively creating a soft dependency. This is done through a separate API layer which defines the functionality that plotting requires from third-party libraries, allowing users to bring in their backends of choice as long as they meet these requirements. This was a recent change by pandas contributors to offer greater flexibility to users.

Another interesting fact that the team noticed was that the plotting module and the tseries module are separate modules, despite performing similar functions. This is due to the fact that there are numerous differences between time-series graphs and other forms of graphs, and as such, pandas developers decided that there was little overlap in execution between the two.

Quality

Essentially, the pandas library uses DataFrames and Series as the main data structures users interact with. DataFrames are composed of the Series and Index classes, while Series are composed of Index. Designing their most important data structures like this was clever, as it uses favors composition over inheritance. However, because of the nature of how DataFrames utilize Series, the classes are tightly coupled. This results in users not being able to use DataFrames to their full capability without having a fundamental understanding of how Series work. The level of abstraction for index is higher, as they are abstracted in DataFrames implementation of .iat, .loc, and .iloc. This improves the overall architecture of the core module.

The core module also contains functionality for modifying and working with data that uses DataFrames as its base. Each of the packages are separate from each other and are defined in what they are used for. Pandas. Array and Computation place an abstraction layer between numpy and numexpr respectively. Although this abstraction layer is present, the coupling between pandas and numpy is so tight that usually when working with pandas, one would have to import numpy to use certain types of data in their DataFrame. For example, a common way to generate random data while using pandas is done directly through the use of numpy.random.randn().

APIs that do not modify data lie outside of the core module. As pandas has two primary uses (viewing and modifying data), the decoupling of their implementations shows good architectural design. The code base is split into layers, with the backend layer containing other libraries, the core packages layer containing the main APIs, and the external API layer containing modules that are unrelated to working with data. All libraries from the backend have an abstraction layer between the library and pandas, which decreases coupling, which in turn improves the overall architecture. The external API layer keeps “data manipulation” and “not data manipulation” separate, once again separating functionality of code.

Improvements

A possible improvement to the architecture that the team considered was to bring the modularization of the plotting backend to other dependencies in the project. The Computations module within core, for example, uses the numexpr library, but there might be other evaluation backends that users may be more familiar with and may prefer. The flexibility offered by choosing your own backend is something that would be appreciated by users who use other libraries or write their own custom libraries in order to meet their needs. The more modularized pandas is, the more easily it will fit into other users’ workflows, allowing for increased adoption.

Also, there seems to be new methods and classes in pandas that are intended to replace certain dependencies, but are not yet implemented in all files. For example, there is a pandas class to detect and maintain NaN values in a DataFrame, *pandas.isna*, but some places in the code base still use numpy functions, *numpy.isnan*. Another possible improvement might be to unify these changes across the project, with systems in place to update old code as new pandas functions become available.

Chosen Software Development Process

One of the team goals is to deliver value to users in a limited time. As such, the team has chosen an agile software development process to achieve this goal. This process uses Kanban and a modified version of the reuse-oriented model.

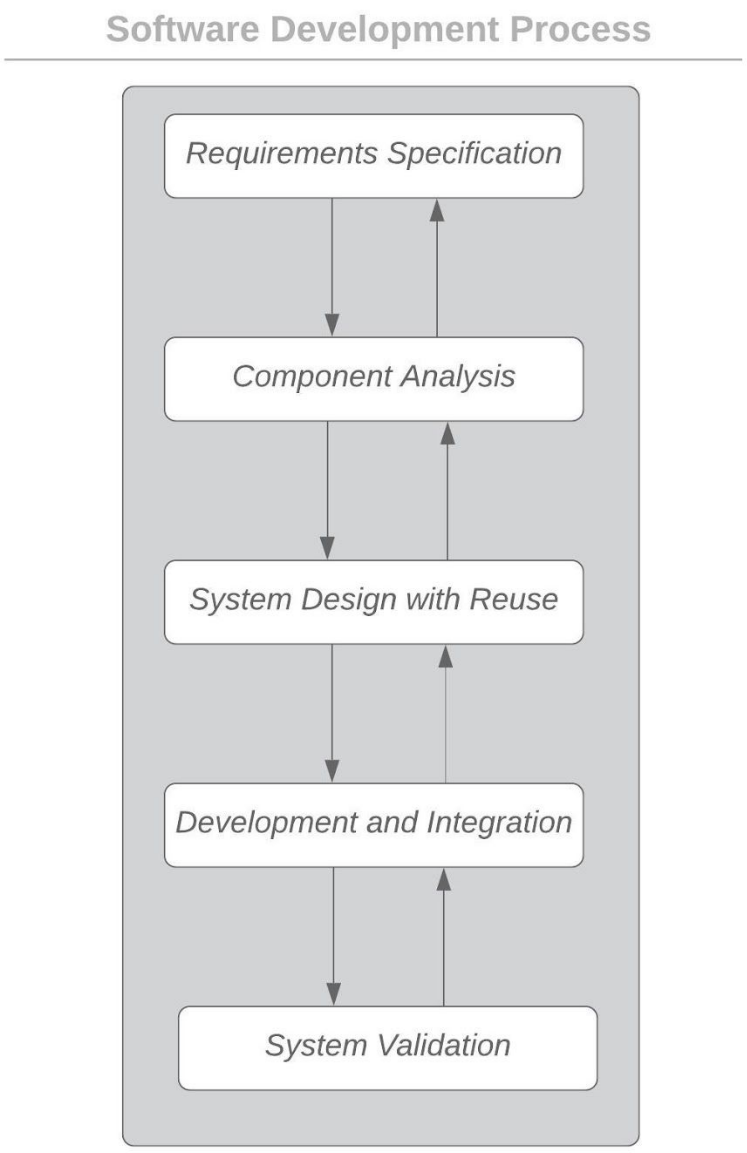

Software Development Process
Requirements Specification
Component Analysis
System Design with Reuse
Development and Integration
System Validation

Figure 3 Illustration of FOSSkateers' Software Development Process
Based on a modified version of Sommervile's reuse-oriented model.

WIP Limits & Its Benefits

A component of Kanban that the team thought would help reach its goal is the WIP (Work-In-Progress) limits. These limits encourage the team to only "pull" a task when there is no pile-up of unfinished tasks in any other stage. This results in value being delivered as soon as possible to users.

In dealing with an accumulation of unfinished work, the team has agreed that:

- If there is a pile-up of unreviewed code, as regulated by the WIP limits on the *Code Review* column on the Kanban board, a member will prioritize code review before they pull a new task.

- If there is no code to be reviewed and there is a pile-up of in-progress tasks, a member will take initiative by offering to pair program with another member to wrap up the task. Pair programming is, of course, a key practice borrowed from XP (Extreme Programming).

To ensure the WIP limits keep up with the team's performance and progress, the team will analyze data from the *Control Chart* and *Cumulative Flow Chart*. One will allow the team to observe the life cycles of tasks, and the time taken by them during each phase of development; and the other will give the team an overview of its progress and performance as time passes. These metrics will show the team the bottlenecks in the team's software development process and inform changes to the WIP limits, so that the team can maintain a healthy workflow.

Kanban Board & Its Benefits

As the team has limited time and want to deliver value to users quickly, it is important that the team spends as much time as possible on development. This means unnecessary activities should be eliminated and members should be encouraged to take initiative at any time. These are made possible with the team's choice of process since Kanban does not force members to adopt additional roles and responsibilities. In addition, the Kanban board allows all members to see who is doing what and whether there is a pile-up of tasks in a stage at any time.

In the team's process, the Kanban board will be used to guide the daily stand-ups (i.e., "walk the board"). Due to the team's varying schedules, it is not possible to schedule daily in-person meetings. As such, the team has identified a digital solution (Jira's Kanban board) which allows everyone to participate online on a regular basis (ideally everyday). In helping the team to understand what is happening at a glance, a task will be represented as a card in which:

- The front side of the card displays the assignee and a short description of the task.

- After clicking on a card, a more detailed description of the task is displayed, including any requirements specification, links to technical discussions, estimate, etc.

During a daily stand-up, the team will identify blockers, check that there is no pile-up of unfinished tasks in any stage, and ensure that the team is working on/pulling tasks with the highest priority. Task prioritization will be shown on the Kanban board using differently coloured cards. The team will decide on prioritization based on business value, the difficulty of tasks (i.e., riskier tasks should be tackled early), and input from the teaching assistant. When a team member pulls a task, they will do so from choosing a card found at the top the *Selected for Development* column. All cards in this column are guaranteed to be worked on and will be ordered based on prioritization, with the card of highest importance on top.

Kanban's Flexibility

Another aspect of Kanban that suits the team is its flexibility. For the next deliverable, the team will attempt to work on bugs in which components (in the pandas code base) can be reused, which means applying the reuse-oriented model. As such, the Kanban board has been configured to reflect the process activities in the reuse-oriented model. However, the choice of process model can change in the future; Kanban allows for this change with minimal disruption to the team's workflow—the team can simply add and remove columns.

Drawbacks to Kanban

A drawback to Kanban is that the process relies heavily on keeping the board updated, as the visibility of progress is hugely important to the way Kanban works.

Due to the lack of time frames in Kanban, it becomes difficult to judge progress made under real-world time constraints. While being highly visible at any given moment during development, the long term visibility of the project is reduced. This problem can be mitigated, however, by using metrics[1] to track progress made in the project as tasks are completed.

Modifications Made To Kanban

By default, Jira's Kanban board comes with the columns: *Backlog*, *Selected for Development*, *In Progress*, and *Done*. However, due to the team's need for greater visibility for items that need *peer review*, and enabled by Kanban's flexibility, the team has decided to add that column. The *Backlog* column has been moved off the board so that the focus is on the work in progress, and issues can be prioritized without distracting the entire team.

The team will not be holding all seven Kanban cadences[2] formally primarily due to time constraints. However, issues that these Kanban cadences will be discussed in one capacity or another.

[1] Metrics are discussed in "WIP Limits & Its Benefits".
[2] https://kanbanzone.com/2019/improve-your-information-flow-with-kanban-cadences/

Reuse-Oriented Model: Process Activities

The process activities encapsulated in the reuse-oriented model complement Kanban well since these activities reduce the amount of code developed. This leads to value being delivered sooner to users.

The activities that the process involves are[3]:

Requirements Specification: Similar to other software development processes, the initial requirements of the system are laid out.The team will find these specifications on GitHub, by reproducing bugs locally, and talking with the bug reporter and other core pandas members. There are two types of requirements that can be included:

- User requirements: Abstract statements of how the system should work from a client's perspective.
- System requirements: More detailed statements about the functionality of specific components used to build the system.

Here, requirements will be recorded and will be made accessible to all.

Component Analysis: Here, the team will search for components in the pandas code base that will help implement the specification.

System Design with Reuse: During this stage, the framework is designed or an existing framework is used. The team will take into account the reusable components discovered in the previous stage. New code may need to be developed if components are not available.

At this point, bugs will be broken down into tasks and prioritized[4].

Development and Integration: New code is developed (if applicable) and the reusable components (and new code) are integrated to create a new system. The team will adhere to the development practices stated in Deliverable 0. This includes creating branches, writing and running unit tests, and reviewing pull-requests before merging them to the master branch.

System Validation: The team will verify that the system created works as specified by the client. Different types of tests (specifically, rerunning all tests made and running the master branch) can be used, which may reveal that individual components do not work as specified, new code breaks other functionalities, there are non-functional requirements needed for integrating parts of the system, or there are missing system requirements.

[3] Software Engineering (9th Edition), Sommerville 2010.
[4] Prioritization is discussed in "Kanban Board and Its Benefits".

Stages will be interleaved since the team cannot reliably anticipate changes. Therefore, this interleaving ensures that the team can respond to changing requirements easily and without having to do a significant amount of rework.

Drawback to the Reuse-Oriented Model

A drawback to the reuse-oriented model is that, in theory, its *requirements modification* activity may result in functionality that does not meet user's needs. The team has attempted to reduce the risk of this by modifying the reuse-oriented model (see following section).

Modification Made to the Reuse-Oriented Model

The team has decided not to have *requirements modifications* as a process activity at this time. It is unlikely that the team will have input on the requirements specification if it is discovered that the components analyzed during the *component analysis* activity do not fully satisfy the requirements activity. In this case, team will likely do one of the following:

- Develop new code to fully meet the requirements specification.
- Go through the *component analysis* activity again.

Consideration of Other Software Development Processes and Models

The pros and cons of other processes and models have been identified below.

Extreme Programming (XP)
XP is well-suited for continuous integration, which was considered a plus for the team's choice of project (pandas requires that fixes are well tested before being merged to the code base). In addition, this process aligns with the team's goal of delivering value to users sooner; in XP, work is prioritized so that business value is delivered first.

However, this process was almost immediately dismissed as it is very prescribed (though for good reason). It would have been difficult for the team to adopt strict engineering practices in a limited period of time. In addition, the team does not have access to the customer (who prioritizes work for the development team and helps write test cases), which is a huge aspect of XP.

Plan-Driven Development/Waterfall Model
This process was considered by the team because it is easy to measure progress. All outputs are planned in advance (they are simply products produced at the end of a stage) and so progress is simply measured against this plan.

However, a significant drawback to this process is that, in principle, one phase must be completed before progress to the next phase is possible. This results in significant rework if changes are made and as mentioned previously, the team cannot reliably anticipate changes. In addition, because this process is broken down into separate sequential phases, the cost of changing and fixing software increases across the software development cycle.

Rational Unified Process (RUP)
The team appreciates how iterative this process is, which means that it allows for changes. However, RUP is not widely used in industry, and the team is interested in gaining experience with a more widely used software development process. Another drawback of this process is that it encourages a lot of documentation, which is an activity that the team is not interested in spending substantial time doing.

Scrum
The team considered this process because it caters to changes in specifications. In addition, the team is familiar with this process, which means that the team can focus more on delivering value to users.

However, it was ultimately eliminated as a choice because the strengths of Scrum in responding to changes are not fully applicable to the current project. In addition, the team does not have access to a product owner and the team does not want to force some members to take on additional roles and responsibilities (e.g., Scrum Master). These elements are core to Scrum, which means that the team could not modify this process without losing the process's essence.

Spiral Model
This model emphasizes the early elimination of risk, which is important in achieving the team's goals in delivering value to users sooner and reducing the amount of rework done. As such, the team will try to identify such risks in future deliverables.

However, the model prescribes that activities are put in place to reduce key risks at every phase of the software development process (e.g., prototyping), which the team finds difficult to carry out in this project, primarily due to time constraints. In addition, the model is rarely used in exact form to begin with.

V-Model
The team considered this model because it considers testing right from the beginning, which ensures that the software is being developed correctly and the right software is being developed. In other words, validation and verification are thought about at every phase of the software development process.

A drawback to this model for the team is that it does not easily allow for change. Each software development phase requires the output of the previous phase to ensure robust testing. This is why the model is best suited for a plan-driven software process (e.g., Waterfall process) in which changes are expected to be limited, thus eliminating a significant amount of rework needed. However, the team is more interested in responding to changes quickly and easily, as communication with issue reporters and core-developers may change requirements.

Conclusion

To conclude, several software development processes and models were carefully considered for our project. In the end, we chose a development process which embraces changes and emphasizes the delivery of value to users sooner rather than later.