McMaster University

SmartServe

Software & Mechatronics Capstone

# Development Process & Implementation

*Authors:*
Christopher McDonald
Harit Patel
Janak Patel
Jared Rayner
Nisarg Patel
Sam Hamel
Sharon Platkin

*Professor:*
Dr. Alan Wassyng

*Teaching Assistants:*
Bennett Mackenzie
Nicholas Annable
Stephen Wynn-Williams
Viktor Smirnov

Last compiled on November 19, 2017

# Contents



# List of Figures



| Date | Revision | Comments | Author(s) |
|---|---|---|---|
| October 28, 2017 | 1.0 | Added structure and content | Christopher McDonald |
| October 30, 2017 | 1.1 | Added detail into Version Control section and revised document | Sharon Platkin & Christopher McDonald |

Figure 1: Revision History

# 1 Introduction

## 1.1 Project Overview

SmartServe is an autonomous table tennis training system for a wide range of table tennis players to aid in diagnosing and improving their performance. The system does so by shooting balls toward the player and detecting successful returns. It can then adapt to the player's weaknesses and expose them by playing more to them over time. This alleviates problems of finding and working with a coach or training many players in a short amount of time. The system will be deemed a success if table tennis players enjoy and can see value in using our system in place of or in tandem to a coach.

The development started at the beginning of the Fall 2017 academic turn and will conclude at the end of the Winter 2018 term. The team was assembled in the capstone course for Software and Mechatronics Engineering disciplines.

## 1.2 Naming Conventions & Terminology

- **Git**: a distributed versioning control system
- **Master Branch**: the main branch of the GitHub repository
- **PR**: pull request, a request to add changes from one branch into another done via a merge commit or rebase
- **Slack**: a messaging platform for teams to subscribe and publish to channels or send messages between two or more team members
- **DevOps**: development operations

# 2 Team Members

The team members are as follows:

- Christopher McDonald
- Harit Patel
- Janak Patel
- Jared Rayner
- Nisarg Patel
- Sam Hamel

- Sharon Platkin

In addition to the list above, Dr. Alan Wassyng will be the Project Advisor. He will be assisted by Bennett Mackenzie, Nicholas Annable, Stephen Wynn-Williams and Viktor Smirnov for marking and providing advice to the team.

## 2.1 Roles & Responsibilities

See Table 1 for a breakdown of roles and responsibilities. When a backup member is needed, they will be listed in italics.

Table 1: Roles & Responsibilities Breakdown

| Role | Member(s) | Responsibility |
|---|---|---|
| Scribe | Sharon Platkin<br>*Jared Rayner* | Taking notes for all meetings<br>Posting notes to Slack after meetings<br>Booking rooms for team meetings |
| DevOps Developer | Christopher McDonald<br>*Sharon Platkin* | Solving issues related to GitHub<br>Configuring CI Tools<br>Merging code changes into the *master* branch |
| Team Contact | Christopher McDonald<br>*Nisarg Patel* | Handle a majority of communication with Project Advisor and Teaching Assistants |
| Hardware Team Member | Janak Patel<br>Jared Rayner<br>Nisarg Patel | Handle most hardware focused work tasks |
| Software Team Member | Christopher McDonald<br>Harit Patel<br>Sam Hamel<br>Sharon Platkin | Handle most software focused work tasks |

## 2.2 Meeting Schedule

The team currently has weekly meetings scheduled on Tuesdays at 16:30 and Fridays at 16:30 for a duration of 50 minutes. Every member of the team is expected to attend. Every other Tuesday meeting must cover reflecting on the previous 2 weeks and planning the following 2 weeks for what work needs to be done in that time. All other meetings will be for progress updates, clarifying any work going forward and removing roadblocks which inhibit optimal performance.

The Hardware and Software team will schedule meetings on an ad hoc basis, inviting any members which have a stake in the content being discussed. In the event Sharon Platkin is not present and therefore cannot be the Scribe, a member will be elected by and among the attendees.

### 2.3 Communication Pipelines

All inter-team communication will be done through Slack. This will allow team members to communicate via channels or groups. Examples of the channels currently used are *hardware*, *software*, *documentation*, *meetings*, *deliverables* and *git*. The *git* channel will give updates on every open PR (Pull Request) and the *deliverables* channel gives reminders 2 days prior to deliverable deadlines. Meeting notes will be posted in the *meetings* channel.

When communicating with the TAs or Project Advisor, email will be used and the Team Contact will relay all relevant information to all other team members.

### 2.4 Handling Change

In the event any aspect of the project must be changed, all members can be involved in the decision making. In the absence of a unanimous decision, the team must decide if the change is paramount to the outcome of the project. If not, the decision will be left to a vote by all meeting attendees. If so, a teaching assistant will be involved in the decision making to aid in arriving to a unanimous decision. The Project Advisor will also be involved at the discretion of the teaching assistant.

Once the change has been decided, all documentation and code changes will be given the highest priority to reflect the change made. This will ensure no further work will be done using documentation or code that does not reflect the team's ideals.

## 3 Version Control

### 3.1 Technology

The technology used for version control will be *git*. The *git* repository will be hosted by GitHub and can be found here. *Git* allows developers to host a decentralized version of the source code for any project. This is in addition to a copy hosted on a remote repository. This technology allows any number of developers to perform work on their own copy of the source without impeding on the other developers. Once a developer makes a complete set of changes, they can include these changes in a commit and push them to the remote repository.

### 3.2 Workflow

The workflow we will be using is a feature-branch workflow with the addition of a two-stage release process. For all releases, changes will be made via a PR into the *master* branch from the *develop* branch. Therefore, any demonstrations and deliverables can be found by checking out the *master* branch. When making any changes, an ad hoc branch will be made off the *dev* branch. Once completed, a PR can be made from that branch into the *develop* branch. This pertains to documentation and source code. A flowchart for this can be found in Figure 2. Once a developer begins working on any item, they will start by making a branch based off of *develop*.

In the event commits have been added to *develop* after making your branch and before merging it into develop, a rebase will be required. This will effectively place all your commits made to your feature branch, onto the current *develop* branch.

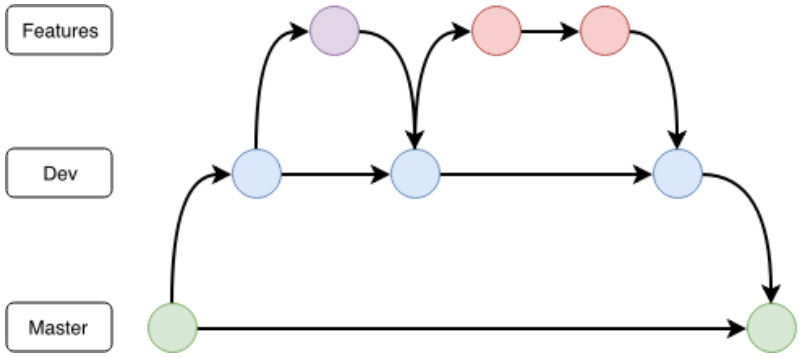


Figure 2: Feature-Branch and Develop Git Workflow

### 3.3 Setup

The following steps must be taken to contribute to the source code.

- Clone the Project

```
git clone https://github.com/ChristopherMcDonald/SoftwareTronCapstone.git
```

- Add a Feature Branch

```
cd /path/to/repo
git fetch --all
git checkout develop
git checkout -b BRANCH_NAME
```

- Push Changes to Repository

```
git add FILE_NAME
git commit -m "MESSAGE"
git push # for first push, git push --set-upstream origin BRANCH_NAME
```

- Rebase Against *develop* Branch

```
git fetch --all
git rebase develop
# deal with any conflicts
git push -f
```

### 3.4 Versioning Scheme

When items of this system are added or revised, the version number of the system will be updated. This number will follow the*major.minor.patch* versioning scheme. This means when anything major has changed, such as a feature being added or a specification being changed, it will increment the current *major* number by one. *Minor* updates will include changes of items in order to refine a feature or meet a specification and will also be incremented by one. For any bug fixes or commentary changes, the *patch* number will be incremented by one.

## 4 Process Workflow

In order to facilitate the development of the system, work will be organized into 2-week long sessions. At the beginning of the week, the team will consider the next 2-3 deliverables and schedule them in accordingly. The team will attempt to finish each deliverable a week or more before the deadline. This allows the team to be ahead of schedule and gives them the ability to respond to other items should they arise. During meetings before the end of each 2-week session, any priorities will be adjusted as needed. When the work session is completed, items will be placed into the next session or placed in the backlog until further consideration. This task tracking is achieved through a real-time collaboration software called Trello. Tello informs every team member on what's being worked on, who's working on what and the status of all tasks.

For the hardware design and assembly of the system, the team is currently using a rapid prototyping approach similar to the cycle in Figure 3. This allows the team to quickly develop and build prototypes for parts of the system. Instead of building everything at once, portions of the system will be built incrementally and integrated into each other to form our complete system. Using the agile methodology again in conjunction with rapid

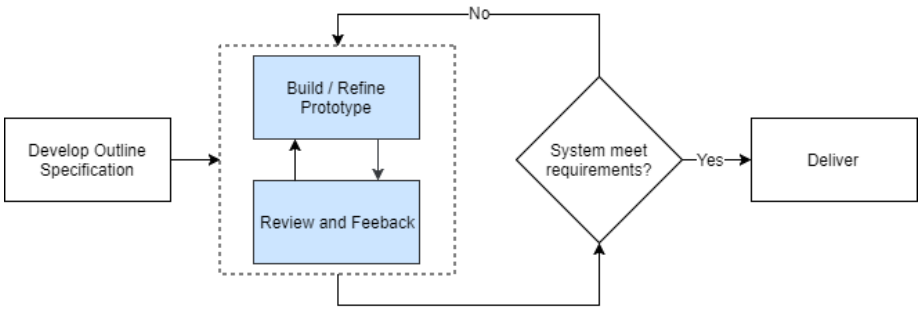


Figure 3: Rapid Prototyping Development Processes

prototyping allows sections of the system to be designed and built simultaneously. This approach is beneficial because it gives more opportunities to obtain feedback from TAs and/or Dr. Alan Wassyng to further refine the prototype. The team is using ready made parts and 3D printing for prototype development because it allows for quick changes and improvements to the prototype which will eventually be implement into the final system.

# 5 Onboarding Process

The following administrative work must be done to onboard another team member:

- Invite them to Slack team
- Share team's Google Calendar with them
- Add them as a collaborator on GitHub
- Add them to the list of authors on the documentation template

The new member will be expected to read all current revisions of documentation made for the project. Any questions which arise can be asked during meetings or on Slack. After a comprehensive understanding of the project, tasks can be assigned to them and any education will be done on an ad hoc basis.

The following tools will need to be installed by the member with the help of the DevOps developer:

- Git version 2.14.0 found here
- LaTeX version LaTeX2$_\varepsilon$ found here