Cover Page

Final PEY Co-op Report 2018-2019
Chaojian Zhang 1000865361, ECE, OTPP

Disclosure statement

```
This is to confirm that I have read the report and that the
information enclosed is correct and contains no confidential
information.

University of Toronto PEY Co-op Student’s Name: Chaojian Zhang
Company: Ontario Teachers' Pension Plan
Supervisor's Name & Title: Warren Peng - Senior Principal
Supervisor's Signature:
Date:
```

Introduction

I had the pleasure to work at OTPP with Total Fund Risk (TFR) team as a Co-op Student for one year starting from late August 2018 to the end of August 2019. During my time with TFR, I was mostly responsible for developing two pieces of softwares which were used regularly by the team. The first one, Funding Liquidity at Risk Engine (FLaRE), is an in-house system to measure funding liquidity risk at OTPP. As a critical component of the funding liquidity risk management process, a what-if tool is required to be built outside FLaRE. The FLaRE what-if tool needs to be flexible in code structure and efficient in computation to support business users such as Total Fund Management. TFM is mandated to make asset mix decisions and manage investment risks including market risk and funding liquidity. I developed the FLaRE what-if capability at OTPP. Another significant project I worked on was RiskHub, a tool to facilitate risk insight and healthy risk dialogue at the fund. The other daily responsibilities included helping my manager with a variety of reporting and data analysis tasks.

To summarize, I was responsible for:

1. Creating a **software** that can efficiently perform FLaRE what-if analysis;
2. Developing a **website** that hosts a variety of charts and data for risk insight and analysis, facilitating a healthy **dialogue** between risk department and risk investment division;
3. Facilitating my manager to study, understand and apply data for solving **risk management** problems.

The purpose of this report is to give an overview of the major project(s) I have worked on at OTPP, and to reflect at high level on my personal experience throughout this year both in terms of **technical aspects** and **non-technical** aspects. I will enumerate the major chanllenges I faced and how I managed to solve them, with the help of my manager and colleagues of course. Also very importantly, I will talk about what specific **methods and knowledge** I gained along the way that may be helpful to fellow readers. A copy of this report will be presented on my personal website (http://totalimagine.com) to showcase what I have done during my work terms.

Main Contents

In this section I will talk specifically about my experience with **Flare What-If**, a server application for risk analysis purpose.

As mentioned in the Introduction, FLaRE is a funding liquidity risk measurement system we use at OTPP, and FLaRE What-If is an **extension** that enables what-if calculation. The what-if capability facilitates a conversation regarding a **potential decision** - whether it is to remove an existing trade or it is to add a new trade. It appears to be a simple elaboration of existing functionalities of FLaRE. In reality, FLaRE What-If is actually a separate piece of software and doesn't depend on existing FLaRE code base because of its unique requirements - it's considered an "extension" insofar as business process is concerned, not in terms of software programming. For instance, the existing FLaRE implementation was done by our team at the time I joined, and is running and maintained by an enterprise team. However, the FLaRE What-If tool is a brand new piece of server application that handles both business logics, hosting and provides service APIs to clients. I was responsible for developing that piece of software.

On the Subject of Conceptual Design vs Engineering Practicality

Before I studied engineering, I didn't appreciate the difference between a programmer, a computer engineer and a software architect, much as I didn't know the difference between a civil engineer and an architectural designer. Some of my friends who are not in the ECE discipline still think people who work as software designers are just programmers who sit all day typing on the keyboard or who are clever with math and can exploit the secrets of telegram messages. Richard Feynman once said (rephrasing his "What I cannot create, I do not understand") "You don't really understand until you can create". It's one thing that you come up with an idea, another thing that you furnish your ideas with details, and a totally different thing that you actually implement your idea and see it work as you expected. Everyone can work out some solutions that **just work**, but it turns out to be more challenging to get things work as you **expect it to be**.

One thing I learnt, or more precisely, strengthened with my understanding, was the criticality of **getting things done** in a **correct and efficient** way. Unlike homework or lab assignments which were mark-based, in a professional setting it seemed all tasks we were assigned were either "satisfying" or "not satisfying", in latter cases one needed to simply **re-work on it** until it's properly done. Also

unlike homework or lab assignments, nothing seemed really "done" because as soon as you had finished some part of the job, new tasks come, and there was always **room for improvement**. RiskHub, which was an internal website we developed for reporting purpose, was a very good example of this: at the very beginning it was a simple prototype showcasing the concepts of presenting our numbers on websites in a real-time fashion instead of using traditional Power Points, then it got more attention and more development and evolve into a platform supporting more transparent communication between our team and our clients. Another good example, as I will elaborate below, is the development of Flare What-if tool.

Flare What-If tool, or Flare What-If Service, is a **RESTful service API**. It is also a web application which allows manual interaction for the purpose of doing what-if analysis with our risk data. Originally I took it over from a colleague who had the idea of a functional prototype for the application which wasn't optimized yet, then we soon realized many problems with that original conception due to **limit of hardware and computational resources** (surprisingly, resources at OTPP still required a budget), and we had to adapt to a more sophisticated mechanism in order to obtain more efficiency.

To quickly summarize the main points, this is the part of *practicality* involved during the development of this tool which were not participated beforehand:

1. In order to support running on a **VM (Virtual machine) server** which didn't have latest version of required software libraries for automatically fetch the data we require (and which we didn't have the full access privilege), we needed to develop **separate data layer** and **presentation layer** as distinct programs running on separate machines - one on VM, the other on the local;
2. In order to support more efficient **human-entered** what-if case specifications, instead of using plain JSON formatted inputs, we developed a very simple-to-learn **domain specific language** called Whistle - the What if Specification Language - which allowed much more efficient and readable inputs for human users;
3. In order to more efficiently **maintain and transfer data files** between server and client, we developed a variety of internal data specifications so everything can be defined in a consistent manner;
4. In order to allow efficient in-memory caching of calculation results, a **dependency-graph-like custom data structure** was introduced to manage the complexity of multi-user many-session cases of running the tool;
5. In order to run a large number of what-if simulation runs with limited resources for different risk dates each of which was very large (many dozen

gigabytes), we developed sophisticated risk data switching logics and session request queuing mechanism - i.e. **session management**; This didn't work well due to technical issues, so we needed some more elaborate and well-designed methods for handling this task laster;
6. In order to support **RESTful API** and more versatile access, we deployed the tool as an ASP.Net Core application instead of WCF application.

To give a more solid context, here are some approximate numbers that define the constraints for development: 1) we need the program to run almost **instantly**, i.e. within *a few seconds*; 2) the program when fully initialized will require more than *30GB memory*; 3) we have only less than *60GB memory* during the first few months of development; 4) the loading from network drives for source data takes more than *5 minutes* for each initialization; 5) we will frequently need to switch risk dates for the running instance, i.e. completely unload then reload different sets of data; 6) the most of time (i.e. bottleneck) spent during program run-time (for the original conceptual prototype) is for **loading and unloading**, i.e. actual underlying business logic is not the bottleneck; 7) We simply just don't have a more powerful machine yet. Now the problem is quite obvious: in order to support **quick** what-if sessions, which will change some of the input data while not affecting others and will totally change the generated results, we can either 1) unload and reload the whole set of data every time, which will definitely fail to meet the speed requirement, or 2) develop some mechanism to cache all the data in memory so the bottleneck of reading data is minimized or completely eliminated. However we cannot just **make a copy of in-memory data** every time we need to perform a what-if session because since each set of data is over *30GB* and we have less than *60GB* memory there is not enough memory for a full copy. However we cannot tell beforehand **which part** of that 30GB data is going to be changed by the user - and also because even if only a subset of data is affected, all of the data will be needed to evaluate the final generated result - so we cannot tell which subset of the data we need to copy beforehand. A more sophisticated way of telling *which data are changed*, *which are not changed*, and possibly *reverting changes* is needed in order to minimize the need to make redundant copy of in-memory data.

Hopefully above statements give a fuller context as regard to the challenge at that time for me and our team. When I say this problem is a matter of "conceptual design", I mean it actually involves a lot of components and one cannot just expect to implement bit by bit until finally figuring out **what is actually needed** - a careful examination of the whole system is needed and an analysis must be given beforehand so implementation can follow with a plan in a timely fashion. When I say this problem is a matter of "engineering practicality", I mean **iterations** and **robustness** of foundational work goes a long way, and one

constantly improves and make sure the quality of his work so later progress can be made based on previous results.

It's also a matter of **discussion** with teammates, asking for advice and decisions from manager, learning **extra skills** and **necessary tools** that helps making sure every step made is solid and useful. Sometimes we have multiple ways of achieving similar results, both comes with respective pros and cons, and it's at this time we need to ask for help from colleagues, and in the end **let the manager make a final call on how to proceed** - with his foresight we can rely on his experience to deal with uncertainty in the future, or simply because he is more familiar with the actual usage cases and can thus make better tradeoffs than those who are merely developing the tools. *Each feature of a tool takes time to implement, and not every feature needs to be treated equally*, that's one take away for a practical tool that's meant to be used, rather than merely conceptualized (e.g. personal and school projects).

In the end, FLaRE what-if took less than **1 second** for a typical small scale run, and around **10 seconds** for moderate to large size runs (i.e. changing more than 25% percent of parameters); It's also used by another project called SWifT, which stands for Strategic What-If Tool. FLaRE what-if is integrated as part of SWifT for more robust risk measurement. As to RiskHub, with features and users growing, it's quickly evolving from a website presenting data into a platform where people can communicate, post requests, and aid in better risk analysis.

Reflection

In this section I will talk about some of the items on my **self-reflection** forms.

In general, I am quite happy with my overall performance at OTPP, except my **communication skills** (aka. English and Chinese), a lot of times even though I know rather clearly what I was thinking and the rationale behind my actions I find it **hard to explain concisely and clearly** to others what I mean. As my colleague and friend Jerry puts it: *I just "can't get to the point"*. Well I do enjoy complicated thoughts than simple ideas, but I figure I could do better with more approachable conversations.

There are some issues with **timing** and completing work assignments on time, adaptability (i.e. speed) to new information and tools (e.g. Excel), adapting to and navigate workplace environments (e.g. floor layout), manage job expectations (i.e. **career goals**) etc. which I am not going to elaborate.

I learnt a lot of **books and various subjects** - most of which are technical, but I didn't find it a good chance to sharpen my skills and seek new development while working on real projects. Such experience is invaluable for a person's technical

growth and it's nowhere available if we are merely working at school or developing our personal projects - there is just not enough **discipline** and **incentive** to keep us going in a positive manner.

Conclusion

In conclusion, it has been a long, fulfilling and at the same time, exhausting year. Surprisingly when I first joined work I enjoyed a lot the balanced daily work and was excited about all the practical information I got a chance to learn, but during later months I began to miss the feeling of going back to school and voyaging into the total unknown - the overwhelming amount of completely new engineering knowledge can seem killingly dangerous but at the same time exciting and intriguing. In fact, I took a brief week off for studying physics, and that was great fun. It's not just about things new though, it's also about the mastery of familiar knowledge, and a **craft of perfection** - sometimes I feel I forgot so much some **fundamental knowledge** I wish I could start over university again so everything I do can be just a bit more efficient and intelligent.

If I was given a chance to **start over and do it (i.e. co-op at OTPP) again**, those are the things I think I could have done better - assuming of course I still possess the knowledge of this first pass (otherwise I doubt I would be clever enough to make any change): I will try better to obtain pass **CFA level one** and study that **Hull's book on financial pricing**. There are some hardcore knowledge that seemed very promising in terms of both personal understanding and utility in daily work but I will not elaborate due to limit of article length; If I get a chance to join this company after graduation to continue my work, then I think I could seriously use some **financial knowledge** to make up the missing pieces of the background and underlying logics going on here at OTPP - of course nowadays I think it's **my greatest strength** that I am trained in software engineering so it actually *doesn't make too much sense to shift completely into a finance role*, but it would be *immensely helpful if I can figure out some ways to combine those two skills*, and make best use of my programming skills and knowledge of computational methods to aid in the analysis and underlying of financial activities and navigate inside the enterprise.

Last but not least, I could seriously get to know people more, instead of immersing in working and developing softwares only.

Recommendations

I definitely recommend whoever is just starting to find a job or thinking about starting her own business, to try to join **something unexpected** like finance

(e.g. for a computer engineering background), the **exposure to unknown realms** can be surprisingly inspiring.

There is something else that intrigued me near the end of my term when I look back what I have done related to **task management and reporting**. From time to time during my work I needed to refer back to previous work and results and make adjustments for new tasks that are based on previous ones, at such times a **file management system** can come very handy and I have invested quite some effort into optimizing this process. The cumulative result is documented in detail in my personal article on my personal portfolio website.