**Ross Video Co-op: Overdrive Developer**

Prepared By: Curtis Collins

Co-operative Education

Computer Engineering Technology – Computing Science

Work Term: Winter 2011

Algonquin College

Employed by: Ross Video

Supervised by: Allan Pepper

Ottawa, Ontario

Submitted: Friday April 8, 2011

Table of Contents

[Executive Summary 3](#_Toc289950278)

[Introduction 4](#_Toc289950279)

[My First Impression 4](#_Toc289950280)

[First Day 4](#_Toc289950281)

[First Feature 5](#_Toc289950282)

[Being an Employee 6](#_Toc289950283)

[Being Part of a Team 6](#_Toc289950284)

[Contributing 6](#_Toc289950285)

[My Next big Project 6](#_Toc289950286)

[Wrapping up the Work Term 7](#_Toc289950287)

[Involvement with Management 7](#_Toc289950288)

[Working with a Timeline 7](#_Toc289950289)

[Getting Ready for NAB 8](#_Toc289950290)

[Conclusion 9](#_Toc289950291)

[Glossary 10](#_Toc289950292)

# Executive Summary

At Ross Video I was a software developer on a product called Overdrive. Not only did I fix bugs, but I took part in creating a couple of features that made their way into the latest release. As a developer should, I participated in regular product meetings as well as PRD reviews. The managers and other developers were very open to ideas, and even though I do not have the same experience in this field as they do, I was able to participate in meaningful discussions about the product, contributing my own thoughts and ideas to the product. Throughout the work term I have been using many different skillsets obtained from my program of study at Algonquin, with object-oriented programming being the greatest asset, considering the nature of my position within Ross Video. The knowledge I have gained from almost every class I have taken at Algonquin, however, has helped me in at least one way or another throughout the term. In contrast, I feel like there was one crucial element that has been overlooked. When I first started my work term I found that I spent a lot of time learning about things that I should already know about, such as version control systems, bug tracking systems, and even simple things like Vnc and remote desktop. These last two I did have previous knowledge of, but I feel that they are essential tools and should be covered in our program of study. Throughout the lifetime of my study period I have found a lack of direction. Each course is very informative and useful by itself, but I just felt it was lacking in its sense of wholeness. Perhaps this is covered later in the program, though, so I won’t dwell on it. All in all, it was a very productive work term, and I have enjoyed it immensely.

# Introduction

Right about now, you are hoping to read a sentence that will excite and thrill you, drawing you deeper and deeper into a report that will leave you utterly speechless. Well, I shall not disappoint you. As far as reports go, they can be awfully boring to read, leaving you fairly unsatisfied and disappointed, and often leave you wondering as to how you might have better spent the last fifteen minutes or so.

The reason you are reading this report is because you are either interested in how my co-op work term has been going, or you are judging it, assessing whether or not I have been productive, using knowledge gained from my program of study at Algonquin. I have.

I have already told you why you *are* reading this report, so now I will tell you why you *should* read this report. This report outlines the chronicles of the past four months of my life as a software developer. What I have done, what I have learned, what I have seen, how much I have earned; knowledge-wise. I believe there is a fair bit of valuable information following this introduction, so I believe we should get right down to business.

# My First Impression

## First Day

My first day at Ross Video was slightly terrifying, seeing as I had never before been a developer of any sort; and gotten paid for doing it, that is. The feeling of terror, however, was short-lived as I quickly got to know everyone, and became comfortable with the work environment. I worked on a product called Overdrive, which was written almost entirely in java, and is built on top of Eclipse using the RCP platform. I did what any developer would do when introduced to a new product; I studied it. Not only was it essential to learn how to use the product, but I had to understand what all that damn code meant!

## First Feature

Overdrive is a fairly large and complex piece of software, and it’s built on a LOT of technologies that I had never even heard of, let alone knew how to use. The product itself is a client-server model, and as a whole it has a few different client applications that are run separately, so even though it is a “product,” it is not just one application.

Getting back to my first experience with the software, I began my long journey to conquering this mountain of code the way anyone would; I fixed bugs. After fixing my first bug, and checking it in (This was only my second or third day as an actual Ross Video employee), I was given my first real challenge. I was to use Java Web Start technology to launch a Vnc Viewer application that could connect to, and view, the Linux OS that was running on the switcher panel connected to Overdrive. You see, in Live Video Production, a Switcher is used to cut between cameras and other devices. Overdrive automates this process so that someone doesn’t have to control the switcher itself. This eliminates a lot of human error in the production of live video.

I’ll be blunt; I had no idea what to do. Being new to the position, I had no idea how I was going to accomplish this task I was given. Well, as it happened, it turned out to be a fairly simple task. The most difficult part was learning the frameworks that the Overdrive web server was built on. After a few days of researching JSP, Struts, JNLP, HTML, CSS and the whole MVC software architecture, I was able to get started. Hooking this all together was still a challenge, seeing as I have never actually *used* these technologies before, but when all was said and done it took little more than a week to complete. By the time I was finished, Allan (who is my supervisor, head honcho, boss man, whatever you would like to call him), was already hinting at my next task. By the look of the sinister grin on his face, and a cackling laugh that was more like something you would hear on Halloween, I knew I wasn’t going to like it. Actually, it wasn’t so bad.

# Being an Employee

## Being Part of a Team

After my first feature, I began to feel like a regular Ross Video employee, doing all the regular employee things. I went to the Overdrive weekly meetings and we would each discuss what we were working on, and how it was coming along. This weekly meeting, however, was far from the only time we would collaborate as a group. It seemed as if we would discuss new ideas and brainstorm the challenges we were facing almost every day, one way or another. We worked alone, and yet we were a team, each of us a cog on the machine that was our product. It was pretty impressive, to say the least.

## Contributing

Perhaps one of the greatest things about Ross Video in particular is in how much I was able to contribute to the company. I am not saying my ideas were any good at all, but I had a say in how things were done! Anytime there was a PRD review regarding something Overdrive related, I was always invited to the meeting. In the review, everyone is given a chance to give their input, and so I took advantage of that any time I had an idea. Another great thing is that the code I wrote was actually being released with the product.

## My Next big Project

After fixing about fifteen bugs, give or take, Allan gave me my next big assignment. To understand what I did, I will give you a little lesson. A Ross Video switcher uses something called an MLE to push video out to wherever it is going. MLEs can be stacked, abstractly, on top of one another, which is useful when you need more effects than one MLE can provide. Before I had got my hands on it, Overdrive was only able to control a switcher that had a maximum of four MLEs. What I did was extend Overdrive’s capabilities so that it would be able to control up to eight MLEs, not just four. This involved a lot of planning, and careful consideration as to what modifications would need to be made. As it turns out, it involved a bit of work, and even the redesign of the protocol used to communicate between Overdrive and the switcher. The details of my weeks of work on this feature are very much repetitive, so I will spare them. The real fun came when I had finally convinced myself that I was ready to start testing the changes. We all know that testing is so very important if we are to have a product worth anything at all, and so I made sure to test every possible case reasonable under the circumstances. Of course we ran into problems, slight errors in my code, but these were easily fixed. All in all, the testing went very smoothly, and everything decided it would just work. The feeling of accomplishment I felt at that moment was very nice indeed, and is one of the rewards of working as a developer. It is a good feeling.

# Wrapping up the Work Term

As I continue to work hard during the final few weeks of my time at Ross Video, I continue to contribute to the Overdrive product.

## Involvement with Management

Throughout the work term, I have had a chance to study the inner workings of the workplace, and have gotten to understand a little of how a business operates. The managers at Ross Video work with developers closely in assuring that progress is made. In one particular instance I was approached by the managers to do some testing that was of critical importance to the company, or so I believed anyway. It was a great experience to be able to be so closely involved with the dynamics of the company, and to learn how to react in situations that may present a reasonable amount of pressure.

## Working with a Timeline

Timelines are important. This is something that project managers know very well, but there are a few things I have learned in my time at Ross Video. The first thing I learned is that it is very important to plan your development, and spend the time to make a proper assessment of the task you are about to undertake. The reason this step is so important is because you almost always need to give your project manager an estimate as to when you will complete your task. If you underestimate, you are stuck asking for more time, or even worse, you are stuck working extra hours. Another thing I have learned is that if for some reason my task is going to take more time than I had originally anticipated, then I should let people know. Honesty has come in very handy, I must say. As a developer it is my job to make sure that the development is done properly and on time.

## Getting Ready for NAB

As the end of my work term draws near, I am far from finished working. In the midst of preparations for NAB, I have been working on providing old functionality to a new version of Overdrive. With a new database design in Overdrive 10, we have had to rethink how the clients are updated when changes are made in the database. This has recently been my number one priority. After doing a fair bit of research I had come up with a few designs as to how this feature could be implemented. Reviewing my ideas with other developers helped me to center out the best approach, and I went with that. I decided on a simple database poll from each of the clients that would continuously ask for changed from the database, and would simply update when it found something new. This was a very simplistic approach, and I admit not the ideal approach, but seeing as we were pressed for time, it was the most reasonable approach. Sometimes it is the developer’s job to decide what seems reasonable to accomplish within the time frame he is given. This was no different.

NAB will be the final test of my development, and will be the ultimate critic as to the quality of my work. Mistakes are made, however, and sometimes that is how things go. I have realized that it is impossible to become a perfect developer. It is, however, possible to be a great developer. A great developer isn’t necessarily the fastest developer, or the developer who codes with the highest quality. A great developer is someone who understands what the situation asks of him, finding a balance between speed and quality. And most importantly, a great developer is someone who will not be afraid of making mistakes, but will do whatever it takes to make sure those mistakes are rectified.

# Conclusion

Throughout the work term I have learned a great deal of things. I have used knowledge from my program of study, as well as gained knowledge that compliments what I already knew. I feel like I have developed a new perspective that will change how I learn as I continue my education. Working with other developers and being able to discuss our ideas has given me a great insight into the development process, and I consider that to be invaluable knowledge in what I hope will be a long and prosperous career in this field. Working as a team, and learning how the different roles of a project come together, is perhaps one of the best things I could have learned through this whole experience. Seeing how everyone works together to make a product something that is actually useful to someone in the world, that is what I will really be taking with me from all this. To be a gear turning the machine is one thing, but to understand how every aspect of the machine works in unison to accomplish a set of goals, that is true knowledge. I know that this report is perhaps less technical than it should be, but I thought it would be more valuable to make it semi-enjoyable to read, meeting the requirements at the same time. Lastly I would like to say how much fun it was to work with Ross Video, and that their reputation, as high as it is, does not do them justice. It was a joy to be part of their team.

# Glossary

**NAB** – (National Association of Broadcasters) NAB Show in Las Vegas, where players in the industry showcase their new products and technologies.

**Java** – An object-oriented programming language. You should know that!

**Eclipse RCP** – (Rich Client Platform) The RCP is basically a core application in which you can build upon in order to build a full application more quickly, using software that already exists as your core.

**Java Web Start** – Web Start is a framework used to launch java native software from a web browser.

**Struts –** A Web framework that allows you to abstract your view from your model. It serves as the controller in a Java web project.

**JSP –** (Java Server Pages) Used to produce dynamic web content.

**MVC –** (Model-View-Controller) Architecture in software development that defines where certain type of code should be contained, separating server logic from the user interface.

**HTML –** (Hyper Text Markup Language) Interpreted by web browsers to display web pages.

**CSS** – (Cascading Style Sheets) Used to create and organize styles associated with web pages.

**PRD** – (Product Requirement Document) Outlines the requirements of a product or feature.