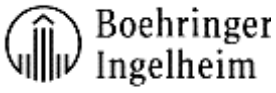
Conception of a Small-Molecule Conformational Analysis Module for NMR Scientists

Prepared by Marc-André Laverdière

Software Engineering Student

1st Work Term



Research & Development Division

Laval (Qc)

Institute for Co-operative Education

January 21st, 2002

Table of Contents

Table of Contents ..... *ii*

Table of Figures ..... *iv*

Definitions ..... *v*

Notes ..... *v*

Introduction *1*

Corporate Background **1**

My Mandate **1**

Software Development Methodology *2*

Software Project *3*

Development Environment **3**

The War on Viruses **3**

The Nature of Viral Reproduction 3

Inhibitors 4

The Role of NMR Spectroscopists 4

High Temperature Molecular Dynamics **4**

Implementation Specifics 4

Restraints & Pseudo Atoms **4**

Implementation Specifics 5

Simulated Annealing **5**

Implementation Specifics 6

Preclustering **6**

Implementation Specifics 6

Clustering Matrix Analysis **6**

Implementation Specifics 7

Visualization of Families **7**

Implementation Specifics 8

NOE Analysis **8**

Implementation Specifics 9

Libraries **9**

Analysis of Work Term *10*

Day to Day Duties **10**

Benefits From Concordia's Education **10**

Operating System Concepts 10

Algorithmics 10

Documentation 11

Unix/Linux Environments 11

Strength And Weaknesses of Concordia's Training **11**

Strengths 11

Weaknesses 11

Improvement to Software Engineering Programme Recommended 11

Focus of Studies 12
Teaching of Basic Methodology 12
Wider Base of Knowledge 12
Project Design and Realization 12

Contribution to Professional and Personal Development 13

Professional Skills 13
Personal Skills 13

Table of Figures

Figure 1: Restraint Manager User Interface _____ *5*
Figure 3: Cluster Matrix _____ *7*
Figure 4: Family Visualization Tool _____ *8*
Figure 5: NOE Analysis User Interface _____ *9*

Definitions

For the scope of this document, we take the following terms to be defined as follows:

NMR Spectrometer: Nuclear Magnetic Resonance. Allows the spectral analysis of molecules.

NMR Spectroscopist: Scientist using *NMR Spectra* results to determine the composition and structure of an analyzed molecule.

NMR Spectra: The output of a *NMR* spectrometer.

NOE Data: Distance information between two atoms obtained from NMR spectra,

Inhibitor: A Molecule that inhibits a protein's activity.

MOE: Molecular Operating Environment (software)

SVL: Scientific Vector Language: programming language inside the MOE

Pseudo Atom: "Dummy" atom representing a group of atoms.

Conformational Determination Process: Methodology used to determine the conformation of a given molecule, based on *NMR Data*.

Aromatic Ring: closed series of linked atoms where all electrons are delocalized.

Prototypal Development Strategy: Software Development approach based on the creation of prototypes of increasing complexity. Normally, the prototypes are used for user's evaluation.

Notes

The software package developed is under the intellectual property of Boehringer Ingelheim (Canada) Ltd. As such, all implementation specifics, novel algorithms and actual code are protected from publication as per non-disclosure agreements.

The screenshots in this document are based on results of the analysis of a fake molecule that do not represent Boehringer Ingelheim's research.

Introduction

During my work-term at Boehringer Ingelheim (Canada) ltd., I had the opportunity to develop a full software suite to be used by *NMR Spectroscopists* in the task of small-molecule conformational determination, under the supervision of Dr. Araz Jakalian. Even tough wide resources were available, many challenges arose during the accomplishment of this project, mainly due to the uniqueness of the developing environment, notably on the scientific specificities, the *User Interface* handling and algorithmic design. In the scope of this report, we will overview the unique environment in which I evolved trough the corporate background and my overall mandate. Moreover, we will analyze the development methodology used, as well as the specifics of the project itself.

Corporate Background

"The Boehringer Ingelheim group of companies, headquartered in Ingelheim, Germany, is one of the 20 leading pharmaceutical corporations in the world. In 2000, it posted revenues of more than EUR 6 billion. Boehringer Ingelheim, which has some 140 affiliated companies worldwide, focuses on human pharmaceuticals and animal health. The human pharmaceuticals business, which accounts for 95% of sales, is comprised of prescription medicines, consumer health care products, chemicals and biopharmaceuticals for industrial customers. Research and development, production, and distribution facilities are located around the globe. In 2000, Boehringer Ingelheim spent almost EUR 1.0 billion on R&D, equivalent to 16% of net sales"[1].

My Mandate

My mandate as a software engineering student was to develop a set of software modules to be used by *NMR Spectroscopists* as a replacement and expansion of another software package. I based myself mainly on already available algorithms and supplied code inside the MOE. I had to query the users, learn about the scientific aspects of the software, train the users and establish a development strategy that would allow efficient development in a user-integrating manner.

I was granted full decisive power over all phases of development, as well as a comprehensive access to facilities, tools, resources and personnel to assist me in my task.

[1] Quote extracted from Boehringer Ingelheim's main site, available at http://www.boehringer-ingelheim.com/corporate/home/home.asp

Software Development Methodology

The NMR Spectroscopists, as main users and clients of the project, wanted to play a comprehensive role in the development of the software package in order to obtain a set of tools well tailored to their needs. As such, I decided to sacrifice efficiency in order to apply a *Prototypal* approach, enabling early feedback and exchange with the users. The project, as such, was considered as the sum of the modules built, and each module had its own development phase.

The overall development strategy used the *Incremental Development Methodology*, and the modules were built either using *Iterative* or *Spiral* methodologies. This combination of strategies, combined with the strong prototypal approach, ensured maximal user satisfaction, as well as offering testing embedded with the development cycle.

This, however, implied that major changes in the software's specifications and architecture would happen, and design steps were taken to lighten this overhead.

In this manner, I was able to fulfill the projects' objectives on user integration, quality, efficiency and performance, as well as providing a comprehensive set of tools to facilitate research.

Software Project

The core component of the project was the creation of specialized software modules inside the Molecular Operating Environment to be used by NMR Spectroscopists in the task of *Conformational Determination*, a powerful tool of *Structure-Based Drug Design*. We will first discover more about the unique development environment I was in, as well as the specifics of virologic research, before proceeding to describe the software itself: the *High Temperature Dynamics*, *Restraint Management*, *Simulated Annealing*, *Clustering* (Preclustering, Clustering, Visualization) and *Analysis* modules.

Development Environment

All the software developed was inside the Molecular Operating Environment (MOE), a software suite from the Chemical Computing Group. MOE offers a kernel of functions and Graphical User Interface to deal with molecules and apply simulations and environments to molecular data, offering specialized databases and a unique programming language: the *Scientific Vector Language* (SVL).

SVL differs from classical programming language concepts by encouraging vectorial data management. An SVL function optimizes the computation of large amounts of data in single function calls and allows high-level processing in single lines of code. Furthermore, SVL allows data organization as vectors and matrices, allowing direct implementation of mathematical concepts such as matrix addition and transposition. This implies a totally different approach to problem solving, on both grounds of memory organization and algorithmic design.

The War on Viruses

Our modern society is plagued with various viral illnesses, such as AIDS and Hepatitis C. Researchers try to find vaccines or to develop drugs assisting the body's natural defense system to eliminate a viral infestation.

The Nature of Viral Reproduction

A virus will typically reproduce itself by inserting its genetic code into an infected cell. The host cell will then generate proteins and will assemble those proteins to create copies of the infecting virus. After this massive multiplication by the cell, the new viruses are then released to infect other nearby cells.

Inhibitors

Inhibitors are molecules that have the property to block the active site(s) of the proteins related to the virus. An effective inhibitor simply blocks the virus' replication process by inhibiting the cell's capacity to put together the building blocks of the virus.

The Role of NMR Spectroscopists

NMR Spectroscopists are scientists who study the composition and conformation of a given inhibitor. The conformation of a molecule represents its actually 3D positioning, impacting on the molecule's solubility and activity. In order to determine an inhibitor's conformation, an *NMR Spectroscopist* must analyze the molecule using *NMR Hydrogen Spectra* in order to find distance ranges between pairs of hydrogen atoms. Once this information is available, the *NMR Spectroscopist* will generate random conformations, and then apply the distance restraints on the conformations, based on the NOE data. The conformations are then cooled down with the distance restraints, in order to obtain realistic low-energy families of conformations. Then, visualization tools will cluster the conformations in order to detect the families of conformations, which will, in turn, tell the *NMR Spectroscopist* which is the likeliest conformation for the studied molecule.

High Temperature Molecular Dynamics

The High Temperature Molecular Dynamics module is a simulation of the molecule's internal movements at high levels of energy (in our case, at 1000K). During this simulation, the molecule's conformation is written to a database at regular intervals.

Implementation Specifics

General code provided with the MOE for a temperature dynamics was already present, but it did not offer the database output we needed. As such, I optimized the code for our needs, as well as adding the required database output.

Restraints & Pseudo Atoms

In order to facilitate the task of entering distance restraints based on the NOE data, a graphical prompter, called *Restraint Manager*, was created, allowing easy restraint and pseudo atom management, as well as saving/importing capabilities.

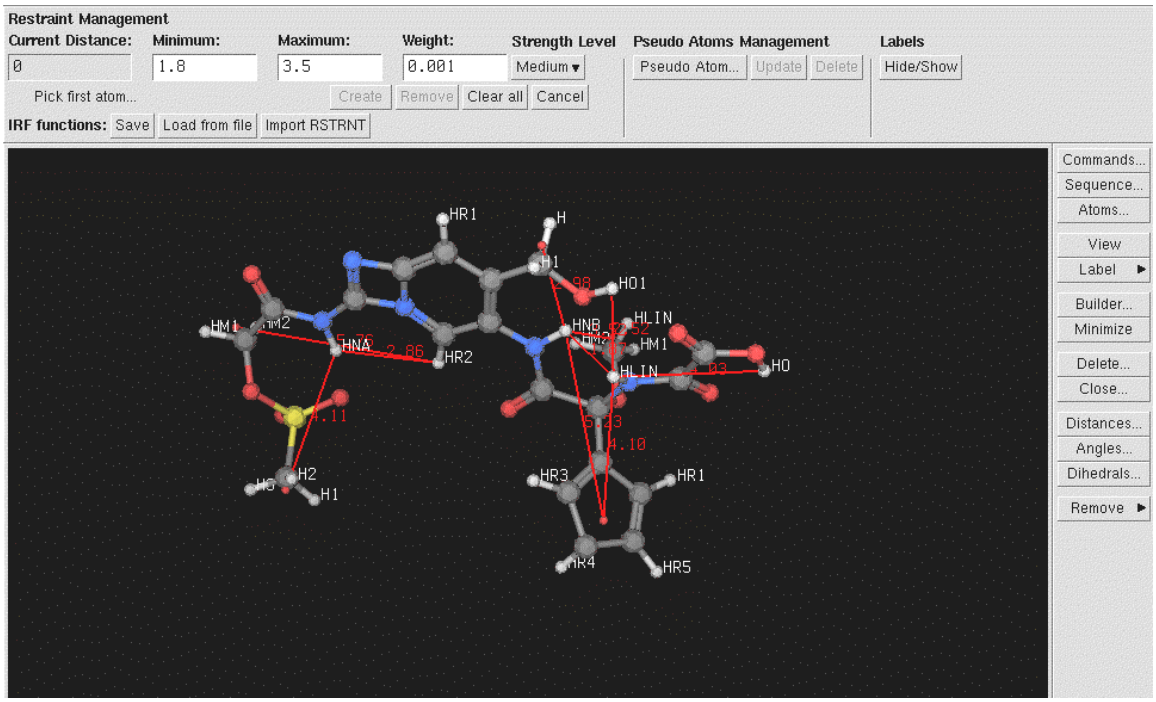


Figure 1: Restraint Manager User Interface

Implementation Specifics

I had to design a new file format: the Interactomic Restraint File (.irf), and expand on a simple prompter already provided inside MOE, so it would become a central, easy to use, graphical tool.

Because the user can kill the prompter's task, dismissing any opportunity of data handling, the *Restraint Manager Prompter* runs as a child process, where the parent process handles the necessary tasks after the execution of the *Prompter*.

Simulated Annealing

The *Simulated Annealing Module* is a crunching function that simulates the slow lowering of temperature of the conformations, from 1000 Kelvin to 50 Kelvin, while imposing the distance restraints and pseudo atoms defined earlier (see Restraints & Pseudo Atoms above for details). The resulting conformation is then minimized to its lowest energy, then restored and re-minimized without any distance restraints. The restrained and unrestrained conformations are then superposed in order to validate the result of the algorithm. All the conformations and various measurements are added to the database.

Implementation Specifics

The original Simulated Annealing protocol, provided to me by Dr. Jakalian, consisted in three series of cooldowns (each cooldown step was followed by *Molecular Dynamics*), with the imposition of restraints between each series.

However, since MOE's biharmonic flat-bottomed potential implementation differs from other software in the industry, the classical protocol has proven to yield useless results.

As such, I recommended alternative algorithms and implemented a series of test programs in order to determine the most efficient protocol to implement as default. This new algorithm cannot be detailed as per non-disclosure agreements with the employer.

Preclustering

The *Preclustering* submodule will prompt the user to select a set of atoms to act as the base for superposition. A series of superpositions will then compare every conformation with each other and output the resulting structures and analysis results in a secondary database.

Implementation Specifics

MOE already provided a module that allowed the superposition of conformations in the same database, based on a user-defined base of superposition. Thus, I had to modify the existing code so that two database files could be handled, as well as updating the *User Interface* for our needs.

Clustering Matrix Analysis

The *Clustering Matrix Analysis* submodule allows the user to visualize the results of the *Preclustering* submodule by creating a user-defined layered dot matrix (see Figure 3: Cluster Matrix) representing families of conformations, grouped by physical closeness.

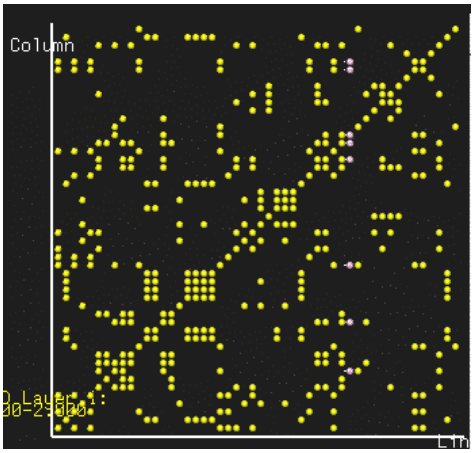


Figure 3: Cluster Matrix

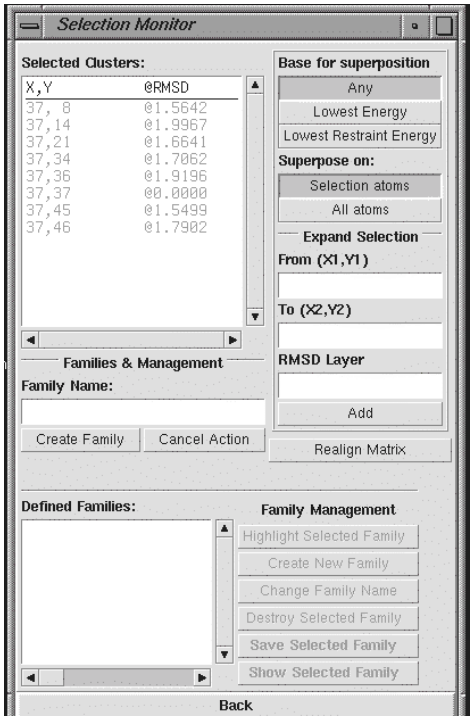


Figure 2: Family Management Panel

The user can then select rows or columns of dots representing families of data, and use the *Family Management Panel* to create and manage families of data, as well as saving the families into separate databases.

Implementation Specifics

This module's GUI has been the biggest technical challenge of the project, partly because of its multithreaded nature. The windows' functionalities were implemented into subprocesses, using shared memory space and message passing to remain synchronized with the core program, features not offered by the MOE, which only provides basic forking capabilities. The clustering matrix itself has been a challenge to implement, since MOE doesn't provide fundamental mouse monitoring APIs. The details of the implementation of the cluster matrix remain Boehringer Ingelheim's intellectual property.

Visualization of Families

The *Family Visualization* tool is a multithreaded program that interacts with the *Clustering Matrix Analysis sub*module, displaying a set of families and offering a set of tools enabling the users to efficiently visualize and update the families (see Figure 4: Family Visualization Tool for details).

Implementation Specifics

This submodule's GUI had the same challenges as in the *Clustering Matrix Analysis* submodule, combined with the challenge to develop an efficient way to provide the family updating functionality. I modified code in the *Clustering Matrix Analysis* submodule to enable the monitoring of multiple conformations.

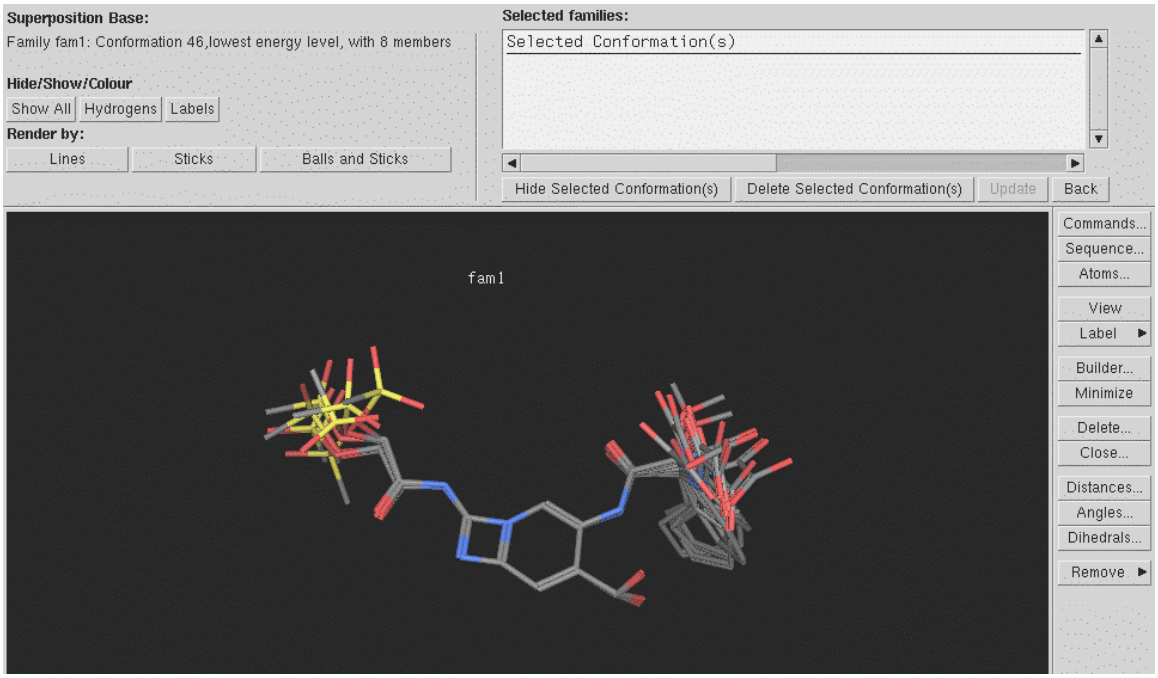


Figure 4: Family Visualization Tool

NOE Analysis

The NOE analysis module has been one of the most complex on the technical level, and constitutes a major improvement over other software packages, enabling the user to perform a quick analysis of the conformations. This analysis tells the user which conformations comply with all the provided NOE data, as well as a list of all atom pairs that could constitute an undeclared NOE (nicknamed "Missing NOEs"). The sensitivity of the detection and conformity thresholds can also be determined by the user. The Missing NOEs algorithm is based on important chemical properties, discarding atom pairs that are interacting as well as atoms inside *aromatic rings*. Therefore, the user is able to quickly and trustingly exclude certain families as potential final conformations, accelerating overall analysis.

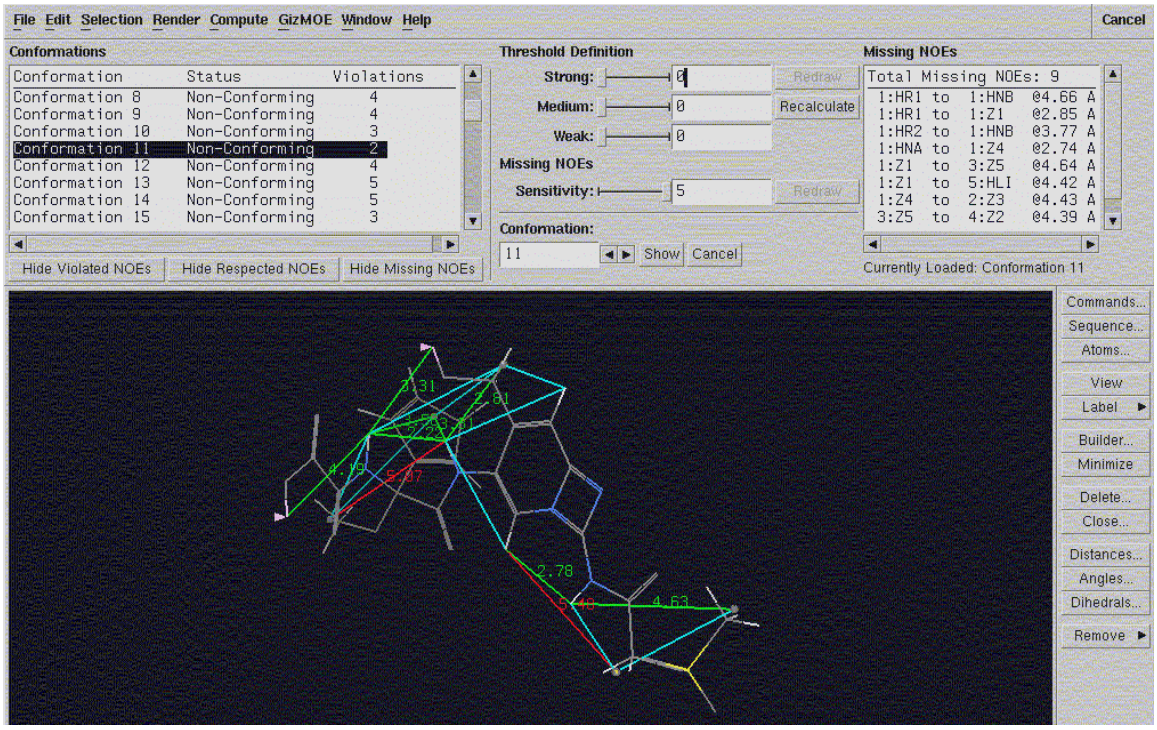


Figure 5: NOE Analysis User Interface

Implementation Specifics

This module consists of two programs: the first program performs the crunching, creating an external database holding the computing results, and the second program provides the GUI over the results. However, the GUI program allows the user to redefine the thresholds and sensitivities of the analysis, therefore using the first program to rebuild the results' database, then refreshing the GUI, based on the new results.

Libraries

In order to facilitate *Code Reuse*, the implementation of the modules and the new features added, a few libraries had to be implemented. The design of these libraries has been facilitated by the fact that MOE will create a copy of the families for each subprocess using them, eliminating the need to develop memory protection techniques and user/process differentiation.

Analysis of Work Term

It is important, as a Co-op student, to effectuate a post-mortem of my activities and preparation for this work term, in order to improve my skills as well as making younger students benefit from my experience.

Day to Day Duties

Since I was the sole developer of the project, I had to fulfill all parts of the *Software Engineering Cycle*, including the development of specifications, design, implementation, testing, etc.

During the early weeks of the project, I spent a lot of time learning about chemistry and about the programming language I was using, even thought this knowledge continued to grow during the work term, as my familiarity and experience allowed me to develop more efficient algorithms.

My routine activities were mainly focused on coding and testing, as well as interacting with the users, querying about scientific information and evaluating the software's functionality.

During some parts of the work term, my work's emphasis shifted on the development of documentation and the preparation of oral presentations.

Benefits From Concordia's Education

This section of the work term's evaluation consists in evaluating which set of skills Concordia University provided me that were useful in the completion of the project.

Operating System Concepts

Trough COMP229, I was able to understand important concepts such as multithreading and APIs that proven to be quite useful in the implementation of the project. Also, this course offered a very large programming project, teaching me a lot about internal documentation.

Algorithmics

Comp352 introduced me to all the concepts of algorithmic design and memory management. Even tough the knowledge was highly specific to C/C++, the skills and reflexes I developed during this course were proven useful for my work term, mainly as a way to establish a basic algorithmic solution that I could further develop as vector-based algorithm.

Documentation

The *Technical Writing* and *Document Processing* courses gave me tools for efficient version management of my code, as well as providing skills helping me express the specifics of the software I developed in a clear and eligible manner, fit for the target audience.

Unix/Linux Environments

The training provided by Concordia allowed me to integrate myself easily in the Unix environment provided by the employer.

Strength And Weaknesses of Concordia's Training

The training provided by Concordia University has proven to be an excellent preparation for the work term I completed, even tough many weaknesses were also highlighted.

Strengths

- The introductory courses of *Computer Science* provide both an overview and a detailed look about important elements enabling the student to solve a wide range of problems.
- The *Technical Writing* course gives a good preparation for the redaction of documents of all natures (user manuals, product documentation, *how to's*, etc.)

Weaknesses

- All the interactions in the course are mainly text-based, which barely develops the student's verbal communication skills. The only oral presentation was done in ENCS281 and did not reflect the wide nature of oral presentations. The student, hired as a *Software Engineering* student, has little or no knowledge in *Software Engineering* as for the first work term. This diminishes the student's ability to provide valuable input and might disappoint employers.
- The student's skills are mainly centered on coding. A few courses teach about methodology, but no training on the issues of *Requirement Analysis* and *Design* is provided.

Improvement to Software Engineering Programme Recommended

This section emphasizes on the gap between the education provided by Concordia University and the actual needs that I had to face to fulfill my project. As such, I will analyze various elements of the courses' scheduling and implementation that could be improved in order to empower the student to

face critical decisions. As such, aspects of studies' focus, methodology, knowledge base and project management will be covered.

Focus of Studies

Before the first work term, the student is subjected to many theoretical and practical courses in *Computer Science*, but practically none in *Software Engineering*. As such, the student has a very limited set of tools in order to make critical decisions. Also, the student hasn't developed the correct methodological reflexes to develop the product as effectively as possible.

As such, I recommend that the students' course planning avoids generic Engineering courses (such as EMAT), in order to allow the insertion of SOEN courses during the 3rd term, which would be more beneficial than theoretical Computer Science courses.

Teaching of Basic Methodology

Learning basics of efficient methodology early on would be greatly beneficial to all students, as we could prevent bad programming habits to develop. As such, I recommend that real emphasis on programming methodology be done in the COMP248 and COMP249 classes.

Wider Base of Knowledge

The education received from the *Computer Science Department* was highly focused on C/C++, restricting wider knowledge to purely theoretical courses, such as *Mathematics for Computer Science*. Alien concepts such as vectorial programming languages may be challenging to students. I encourage the Department to include an introduction to non-classical programming languages.

Furthermore, I encourage the Department to implement oral evaluations and, if possible, on a wider range of audiences. These could take the form of public seminars on topics related to *Computer Science*, use of computer, programming, etc. By this initiative, students could learn more about effective targeting of audiences, preparation and public speaking.

Project Design and Realization

During the basic introductory courses, the student is "taken by the hand" by the instructors, usually trough overdetailed specifications or simplistic assignments. As such, I recommend that all programming courses, instead of having small programming assignments to evaluate the student's

comprehension of the theory, evaluate the student trough a self-managed project. Assignments could still be requested, but as a module, or design document, etc.

Contribution to Professional and Personal Development

The work-term I completed for Boehringer Ingelheim (Canada) ltd. proven to a great learning opportunity for me on both personal and professional levels.

Professional Skills

Efficient *Software Engineering* requires the ability to fully understand the issues at hand in order to make the best decision. As such, my experience taught me a lot about the gathering and clarification of specifications and technical information for product design.

Furthermore, the experience in actual project management and development methodology contributes a lot to my capacity to seek out the best course of action. I also had the opportunity to review the impact of the decisions I made onto the overall development.

As such, I believe that, thanks to the professional skills I learned from this work term, I am in a better position to effectively manage a software management project and evaluate all related work.

Furthermore, since I was developing in an alien environment, I had to learn a lot about chemistry and specifics of the *Conformational Determination Process*. This developed my capability to model situation outside of the traditional realms of *Computer Science*.

Personal Skills

Software Engineers require a lot of interpersonal skills in order to efficiently manage a project, which has proven to be critical to my project's success.

Since the users had no experience in Software Development, I refined my vulgarization skills in order to integrate them in the process, but also to provide efficient training in the software's use.

I also had to query the users about the project's requirements, as well as making presentations to the research scientists. I thus further developed my verbal communication skills and capacity to interact with people of different background and interest. This skill also improved a lot in my capacity to explain algorithmic issues to my supervisor, allowing team-based design.