Concordia University

Institute for Co-Operative Education

Software Engineering Co-Op Programme

Integration of FileMaker Databases as a Stepping Stone Towards an Open Tracking System

Production & Marketing Group

Institute for Research in Construction

National Research Council Canada

Ottawa, Ontario

Prepared by:

Marc-André Laverdière

Second Work Term

August 30th, 2002

Table of Contents

List of Tables and Illustrations

1. Introduction

This section aims at providing the reader with the context of the Production & Marketing group, as well as the project's definition and context.

1.1. Production & Marketing Group Within NRC

For this term, I worked with the Production & Marketing group (P&M), a special unit of publishing professionals within the National Research Council.

The unit acts as a solution provider for the Canadian Codes Centre (CCC) and the Canadian Construction Materials Centre (CCMC), offering publishing, editing, and translation services, custom system development and maintenance, as well as advising on a multitude of projects.

1.2. Project Definition

The Intents Databases Integration project was defined as database integration, data reorganization and user interface improvements over the Intents Databases.

1.3. Project Context

The project aims at providing a short-term solution, acting as an intermediary stage facilitating the development and introduction of the *Open Tracking System*, a Document Management System with integrated workflow automation. The project aimed to lower the maintenance overhead of the current database implementation, and was designed in the context of a shortened life expectancy.

1.3.1. Current Database Implementation

The current data is stored within more than 70 FileMaker Pro database files, built on a single file organization. Because of FileMaker Pro's limitations on Interface Design, the current system holds data within 151 fields, complicating all maintenance operations and allows only single user access.

1.3.2. Project Life Expectancy

The project's user interface will be used for at most a year, assuming the integration of the *Open Tracking System* will proceed as scheduled.

2. Project Realization

This section aims at detailing the project's objectives, process and methods needed to accomplish the previously detailed objectives. A closer look will be given to the project's objectives and resources, the process used, the evaluation of the database system, the reorganization plan, as well as its implementation and documentation practices.

2.1. Project Objectives

Besides the objectives of facilitating the migration to a different software solution, this project also aims to lower maintenance effort, offer an improved reliability, ease of use, multi-user capabilities and to facilitate the publishing of its contents.

2.1.1. Migration of Data

The information was to become as normalized as possible, as this data will be imported within the *Open Tracking System* in the early future.

2.1.2. Facilitation of Maintenance

A major objective of this project was to obtain a solid solution that would reduce maintenance effort from 10 persons-hours per week to less than two person-hours weekly effort.

2.1.3. Reliability

It was also determined that the solution would be developed with the goal of obtaining a 10^{-2} reliability level, as well as offering simple mechanisms for failure recovery.

2.1.4. Ease of Use

The solution was to offer superior simplicity as to avoid user training overhead.

2.1.5. Multi-User Organization

The solution had to allow concurrent users to browse through records, as well as record-locking mechanisms to ensure data integrity in this context, through FileMaker's provided mechanisms and supplemental algorithms as required.

2.1.6. Facilitation of Publication Process

The system also had to accelerate common tasks performed by editorial staff, as well as facilitating the bridging of information between the database system and the SGML Fragments reference, which holds the SGML data used for publishing all National Code documents.

2.2. Software Process

This project, although being part of the maintenance phase of the current database systems, was considered as a novel software development effort.

The process has been divided in three different effort plans: the data reorganization, data security and the user interface redesign.

2.2.1. Data Reorganization Effort Organization

Because the current database organization wasn't documented, the data reorganization effort has been mostly spent into documenting the current database structure.

Once the database structure was layed out, I determined, in collaboration with users, the information to be retained and added within the new database system. As such, a table of fields to be converted and added, with all necessary processing, was layed out.

P&M Information Specialists were also consulted in order to resolve intercompatibility issues between the Intents Database and the Code SGML Fragments repository.

This analysis determined that additional effort would be required in order to facilitate the matching of information between the Intents and SGML Fragments databases. It was determined that it would be necessary to develop an automated record analysis technology (please consult §2.3.1 below for more information on this subproject).

Therefore, the data reorganization effort was spent into two activities: the detection of Code sentence hierarchy and the importation script.

2.2.2. Data Security

As CCC staff has technical responsibilities based on different subjects within National Code documents, it was necessary to elaborate a security system that would allow general read access, but would restrict write access to certain parts of the database. It was furthermore needed to restrict certain functions only to administrators, feature non-existent within FileMaker. As such, I designed an access rights model and prototyped a solution using an Access Control Matrix database file, which was fully reused in the implementation of the solution.

2.2.3. User Interface Redesign Effort Organization

The interface redesign activity has been focused on four lines: interface reuse, productivity enhancement and ease of development.

The Software Engineering Group of the Institute for Information Technology contributed greatly to this process by providing training on effort allocation and new feature detection methodologies, centered on the users' common tasks.

The User Interface efforts were facilitated by reusing 90% of a prototype built by CCC. Reallocating field display to our new data structure and client-server organization were the only modifications necessary.

Users were queried through a survey in order to determine which features were desirable, and which operations could be accelerated through interface improvements. A group of users where then interviewed with regards to their work habits. These results were then compiled prioritized to guide the implementation.

Because of a limited timeline and FileMaker's unique interface programming environment, an iterative rapid prototyping strategy was used to implement features as prioritized. As such, one feature at a time was implemented and tested for conformance, which allowed the delivery of prototypes and basic usability testing during the interface development.

2.3. Implementation

2.3.1. Pattern Recognition of Text Extracts

One of the major challenges required to facilitate publication of the database contents was to create a matching from record-based information within the database, to Code sentence-based organization for the SGML Fragments. Since a Code sentence within the database can be spanned into multiple records, knowledge of which clauses and subclauses were analyzed within each individual record allowed the matching of data with the SGML Fragments.

The text was presented with various ellipses and indentation, which made impossible to use this information directly for detection. Because the clauses were numbered with letters and the subclauses used roman numbering, we couldn't use a pure token identification strategy to extract the information needed with a good level of accuracy.

In order to obtain high detection rates, it was thus necessary to combine both techniques. The implementation is summarily described below.

Based on live data, this algorithm had a successful detection rate superior to 90%.

<table>
<tr><th>Step</th><th>Processing</th></tr>
<tr><td>Substring extraction</td><td>The Input String is divided into substrings using the newline character as divider. For each substring Identifiers located before an ending parenthesis ‘)’ and no more than four characters long are kept. The white space preceding these Identifiers is recorded, and the text after the ending parenthesis is evaluated for the presence of an full ellipsis (…) or partial ellipsis (… <some other text>)</td></tr>
<tr><td>Deterministic hierarchy detection</td><td>The table of Identifiers is analyzed and each is flagged with values for the sentence level, clause level, subclause level, or unknown.
The following rule set is used to determine the level:
<table>
<tr><th>Identifier</th><th>Hierarchy Level</th></tr>
<tr><td>Number</td><td>Sentence</td></tr>
<tr><td>Single letter, except ‘i’ and ‘v’</td><td>Clause</td></tr>
<tr><td>Roman numeral, except ‘i’ and ‘v’</td><td>Subclause</td></tr>
<tr><td>‘i’ or ‘v’</td><td>Unknown</td></tr>
</table>
</td></tr>
<tr><td>Context-based hierarchy detection</td><td>Whenever there are unknown values, a second iteration goes over the data and performs a context analysis of the unknown elements, analyzing indentation differences with the neighbours, as well as using their hierarchical context, in order to determine the hierarchical level of the element.</td></tr>
<tr><td>Output String generation and Warnings</td><td>The table of Identifiers and hierarchical values is used to generate an output string, using the underscore ‘_’ as delimiting character. The subclauses are output as being the value of the preceding clause, with their numeral value concatenated. Full ellipses are dismissed at this point.</td></tr>
</table>

Table 1: Hierarchy Detection Algorithm

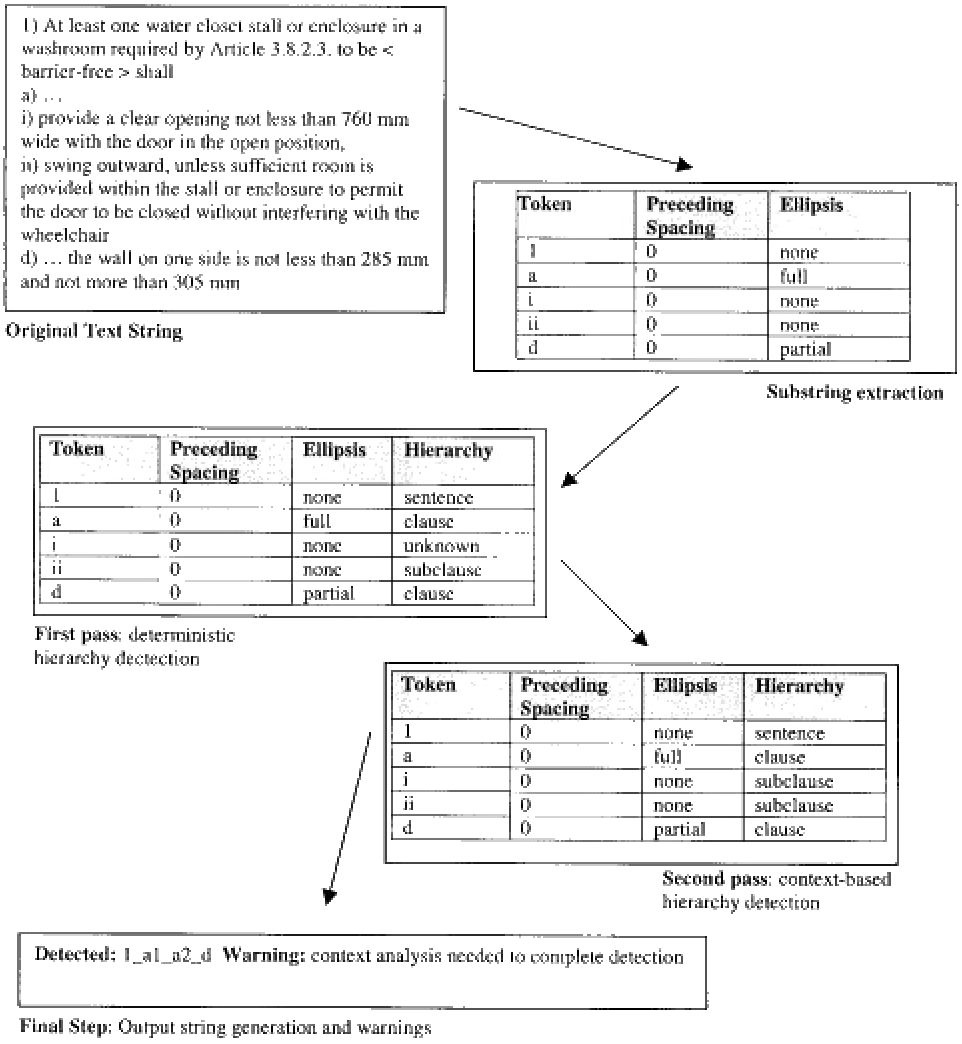

Figure 1: Example of Processing Steps and Results

2.3.2. Development of Importation Script

The script to import and process data from the databases to merge to the new database structure was designed with a 'pipes and filters' approach and implemented procedurally in Perl.

The 'pipes and filters' approach views the program as a series of filters to which will be channelled data to be processed. This approach was particularly appropriate for our importation script, as all the input data was to be directed to become output data, with or without intermediary processing, without user interaction. Furthermore, since each input field required

simple processing in order to become the desired output field, the complexity of the program remained low.

Perl was chosen for this task because of the advanced string processing offered, as well as facilitated memory management. Furthermore, Perl is also a language commonly used within P&M, which would facilitate eventual maintenance operations on the script.

2.3.3. Implementation of Access Control Matrix

The Access Control Matrix (ACM), designed to hold user's access rights to specific sections of the Intents Database, was held on the server, along the data files.

Client User Interface files would access the ACM and set the appropriate user right flags locally. Whenever a restricted operation was launched, the User Interface file would validate the user's access rights before proceeding with the operation, allowing secure and efficient security.

2.3.4. Improved User Interface

After studying the improved user interface provided by CCC, displayed fields were adapted in order to support the new data structure in our client-server organization. Since FileMaker doesn't support a basic 'commit' feature, which makes changes in a field be applied directly in the database, it was necessary to change all displayed layout elements into 'global fields' (FileMaker's equivalent of a local variable). These global fields, in order to be set and saved, needed to be managed by a large set of FileMaker scripts, and required the implementation of browse/edit modes.

Once this step was completed, the two highest priority features were added: the Code reference quick search (as illustrated in the figure below with the "Go to Code Unit" button) as well as the record number quick lookup (illustrated below with the "Go" button).

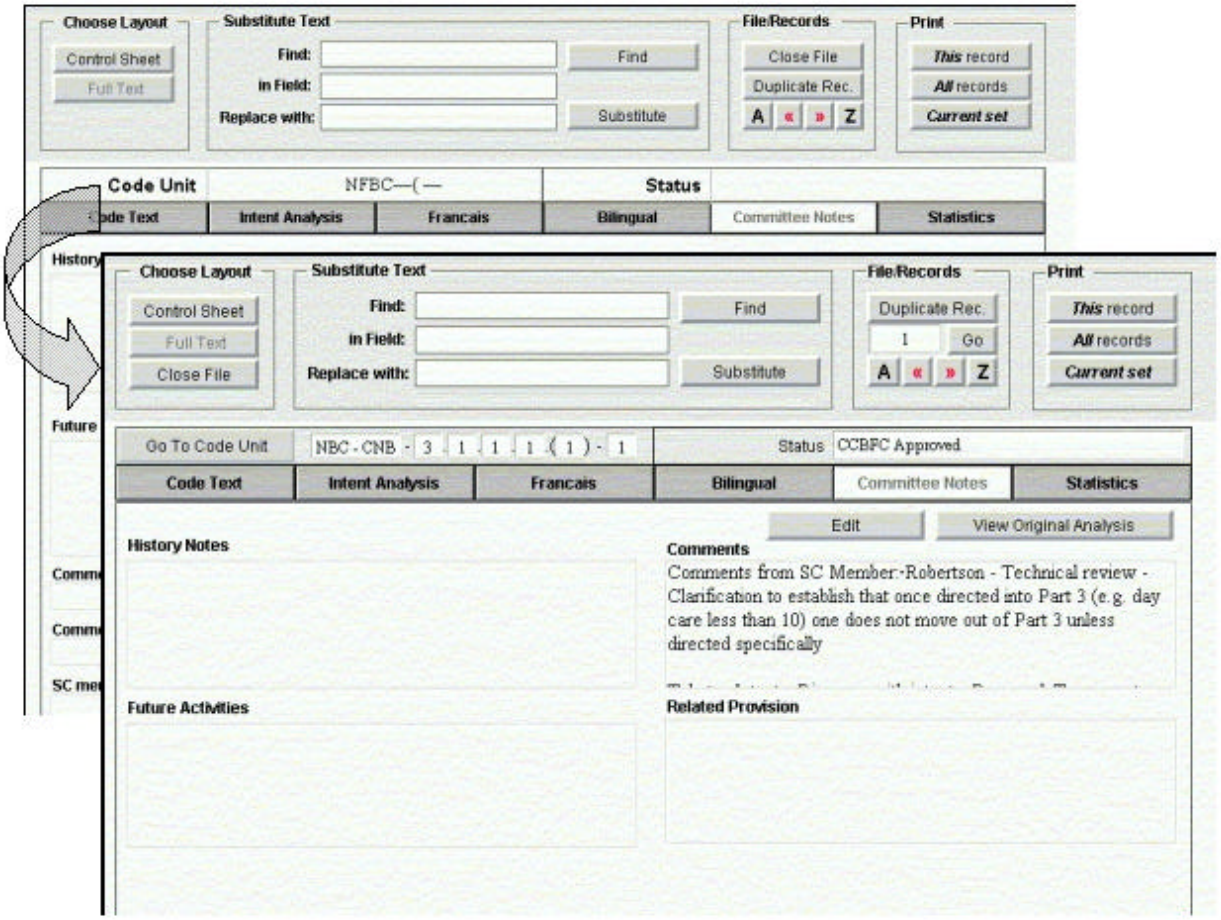

Figure 2: Reuse and Adaptation of User Interface

2.4. Documentation

The project was documented for future development with a maintainers' manual, and a conversion plan, with internal documentation within scripts.

2.4.1. Maintainers' Manual

The maintainers' manual detailed the importation of data and its processing, as well as all the concepts and algorithms needed for the implementation and maintenance of browse/edit modes, access control matrix and all other algorithms added to the User Interface Prototype. A list of functions within the Perl scripts was also provided for reference purposes. Also, the FileMaker scripts developed were detailed in terms of function and field usage.

2.4.2. Conversion Plan

The conversion plan, as introduced in §2.2.1, detailed the databases and the new data model, as well as the steps to undertake to complete the transition.

2.4.3. Script Internal Documentation

The Perl scripts were internally documented with general script description headers, function headers, variable description and block description. Within more complex functions, documentation on a line-by-line basis was also provided.

FileMaker scripts, because of software limitations, were internally described on a block level.

3. Evaluation

This section aims at evaluating the *Software Engineering* programme offered at Concordia University in the context of my employment for NRC, in order to determine strengths and weaknesses in the training offered by Concordia, on academical and personal levels, as well as to critically offer suggestions on programme improvement. As such, we'll focus on the skills obtained from academic training; my professional and personal development; my evaluation of the academic training offered by Concordia, as well as a set of recommendations for programme enhancement.

3.1. Skills Obtained from Academic Training

The courses taught at Concordia University were of great use for me for the completion of this project. My knowledge of data structures and algorithms, technical writing, regular languages, operating systems, pattern recognition and software process were of utmost importance for the realization of this project.

3.1.1. Use of Data Structures and Algorithms Knowledge

Concordia's teaching of data structures and algorithms was offered in a very conceptual manner, which allowed me to easily evaluate and implement algorithms and data structures in new languages.

3.1.2. Use of Technical Writing Knowledge

The Technical Writing training required as part of the programme has been of great use, empowering me to deliver content in a more effective manner.

3.1.3. Use of Regular Languages Knowledge

The use of Perl's regular expressions facilitated tasks for the Importation and Hierarchy Detection scripts. This implementation would've been more challenging without this prior knowledge.

3.1.4. Use of Operating Systems Knowledge

The Operating Systems course introduced critical knowledge of synchronization methods and security systems.

This knowledge allowed me to quickly and efficiently design and implement data integrity mechanisms that were essential in the implementation of our multi-user solution.

Furthermore, the concepts of security taught, particularly the principles and implementation of access control matrices and capabilities-based systems, allowed me to efficiently enforce security objectives within the scope of this project.

3.1.5. Use of Pattern Recognition Knowledge

The pattern recognition course, through the generic presentation of a large variety of algorithms and resource materials, allowed me to critically analyze and to develop an efficient algorithm for the Hierarchy Detection script. This capacity allowed me to determine a solution without the need to deeply research possible algorithms, which greatly accelerated this part of the software development effort of the project.

3.1.6. Use of Software Process Knowledge

The knowledge of *Software Engineering* learn from the Software Process course has been of great support for the completion of the project, as I was able to organize the necessary activities and to set milestones for completion of each part of the development effort.

3.2. Professional and Personal Development

My experience within NRC allowed for multiple rooms of personal and professional development.

On a personal level, I had the opportunity to meet many people and make good friends through the activities organized by NRC. The contact with my colleagues allowed me to learned many new things and to share activities and memories in a very satisfying manner.

On the professional level, I had the chance to work in a complex work environment, within an established hierarchy and dealing with staff of multiple groups. This allowed me to learn about time and task management, as well as learning more on productivity and efficiency in a real work

environment. The contact with users via my technical support duties also taught me a lot on the importance to effective user training and product documentation.

3.3. Evaluation of Academic Training

The academic training offered by Concordia University was quite satisfactory in all technical matters, showing multiple strengths and only a few weaknesses.

On the strong side, the Software Engineering programme, encouraging the learning of technical content during the first year, is quite appropriate in the context of the profession. The skills taught before the first midterm indeed allow students to perform a wide range of programming-related activities.

However, on weaker side, I would comment that the introduction of *Software Engineering* courses is too slow, leaving students with little actual understanding and knowledge of the profession by their second midterm.

As such, the recommended course plan and offerings should focus more on the skills needed by employers for work terms than on generic course requirements, such as *Engineering* core courses. I also believe that the students would be better prepared for work terms if the recommendations of §3.4 were to be implemented.

3.4. Recommendations for Programme Enhancement

My experience within the Intents Databases Integration project allowed me to identify areas of weakness within the *Software Engineering* programme, which would be countered by the teaching of courses in advanced document management, project management, effective communication and advanced technical writing.

3.4.1. Advanced Document Management Instruction

A software effort of the scope and magnitude of the Intents Databases Integration project deals with large amount of information, mainly on database organization.

As to provide a better understanding of the situation I was facing, I'll briefly detail the documentation that was necessary for the completion of this project. The complete information

for this project is concentrated amongst 2 text documents, 2 Perl scripts and multiple FileMaker Pro files. These documents will hold listing of database field structure and usage and algorithms implemented. Fixing of a single error of a field classification, field naming or algorithm description thus becomes a difficult and error-prone maintenance effort.

Concordia University offers no training for document management, except for the fundamentals of revision management in the Document Processing course. Students therefore have no knowledge or skills of methods facilitating the writing and updating of complex database analysis and integration documents, nor maintainer documentation manuals.

With instruction provided on matters of complex project documentation tools and methodologies, a lot of documentation maintenance efforts could've been redirected towards higher documentation quality in the scope of the Intents Database Integration project.

3.4.2. Project Management Instruction

The *Software Engineering* programme, through the *Software Process* and *Software Metrics* courses, offered a quick introduction on issues of software project management and project estimation. This introduction was, however, quite summarily done, providing little to none actual knowledge on managerial activities such as scheduling.

This lack of skill, combined with limited understanding of the technologies used, yielded largely unreliable completion estimates for tasks related to this project.

It is my belief that project management constitutes a fundamental skill that all students should master early, as much as *Technical Writing*, which is a course recommended for first year students.

3.4.3. Effective Communication Instruction

The reality of the workplace requires a large set of communication skills not being taught within the *Software Engineering* programme. The programme requires students to take a single course in technical writing, which briefly introduces other communication skills.

However, the capacity to effectively bridge from the technical realm to stakeholders of varying knowledge and experience is critical in a major project such as the Intents Database Integration.

By offering a course focusing on interaction psychology, presentation techniques, communication tool selection and other 'soft skills', students would better interact between themselves and with coworkers on personal and professional levels.

This is necessary not only in the context of the current workplace expectations, but also as effective multimedia communication is a cornerstone of software engineering. The ability to convince clients, managers and colleagues through effective interaction and presentation is bound to be assets more important to software engineer than technical skills in the near future because of the nature of software development in the current marketplace.

Furthermore, it is essential that professors receive such training, as they act as role models to students. Students generally like professors with greater verbal communication skills, as students feel they are learning more, and that the instructor is approachable. Researchers would further appreciate such sponsored skill development, as presentations in conferences and joint research program efforts would be facilitated by an increased capacity to communicate effectively with peers and members of the public alike.

As such, in the context of today's marketplace and our University's preparatory mandate, it is primordial to put a focus on effective communication at all levels of the University as being complementary to technical skills. Demanding that students enrol in a course as described would likely increase the attractiveness of Concordia graduates to employers and their environment, as well as stimulating an academia more open to the public.

3.4.4. Advanced Technical Writing Instruction

Comments from my supervisor showed good appreciation for my technical writing skills, yet found them lack a fully professional look, which is expected by employers.

Considering that the quality of the documents I wrote so far were always well ranked by our professors, a problem lies in the expectations of the professors towards well written documents, as well as in the instruction for Technical Writing.

Concordia University should introduce a policy requiring that specifically specifically trained technical writers and/or editors evaluate and comment on all term and project reports handed in by students. This solution, considering the cost of such an activity and realizing that it would be financially inapplicable to implement for all submitted work, remains cost-effective and would stimulate students towards improvement of their writing skills.

Courses offered by independent organizations, such as *The Learning Tree*, have offered advanced Technical Writing courses for years, allowing students to develop skills for truly professional and efficient writing. The *Technical Report* course could be changed from a 1.5 credit course to a 2-3 credit course, in which advanced technical writing would be taught to students.