帝京科学大学紀要 Vol.2 (2006) pp. 45-52

研究ノート

ソフトウェアの信頼性に関する一考察

嶋村　尚吾*

（平成 17 年 10 月 3 日受理）

A Study of software quality

Shogo SHIMAMURA*

The Department of Media and Information Systems of Teikyo University of Science & Technology regards software educations for undergraduate students as important. And so, the quality of teaching materials must be improved.

This note describes two examples of systems which were used as teaching materials for software educations, one is with a very simple structure, and the other is with a complicated structure.

The former is a Windows console application “Shogi Program”, a kind of game program which the author developed as one of teaching materials for the freshmen course “Algorithms and Data Structures”. And the structure of it is very simple, because it has no components to be connected with it and users of it always use it alone. This program looks like a program with high quality, because it has showed no defect since the first print on the text of the course and while the author developed its network versions (Web application and C/S (Client-Server) system application) for the 2004 university campus festival (2004 TUST campus festival). But , in order to improve its quality moreover, object oriented developing method is worth considering.

The latter is a network application system with a reception program of a seminar and a registration program to its roll book. The author developed this system for the 2004 TUST campus festival and used it as one of teaching materials for the third-year students course “Database Application”. This system has Web application version and C/S system version. And the structure of this one is complicated, because it had to be developed with managed development tool and uses DBMS (Microsoft Access), and more than one users use it simultaneously. This note describes the content and the cause of the defect of its Web application version which was caused by one 2-bytes space character specified instead of 1-byte space character by mistake. And this defect was found when the author made the third-year students to use it in the 2005 first term course “Database Application”. This example of defect is educational and shows one of the most important things for developers to obey when developing managed mode applications.

キーワード：将棋プログラム、受付プログラム、出席管理プログラム、データベース、Web アプリケーション、クライアントサーバシステムアプリケーション、品質、Microsoft Visual Studio .NET、SQL 文、Windows

1.　帝京科学大学におけるソフトウェアに関する教育の担当

帝京科学大学メディア情報システム学科におけるソフトウェア教育授業の担当講座の内容は、講義細目、および本学の公式 HP で公開している。

また、それらのほかに、平成 16 年度には、科大祭（帝京科学大学祭）において、「インタネットを使用したソフトウェア教育」のテーマのもとに、それらの講座の講義内容の一部と、講義の中で使用している技術を応用して開発したソフトウェアを紹介した。

そこでは、次の 5 つのテーマを取り上げた。

- 講座受付
- アルゴリズムとデータ構造
- データベース応用
- オペレーティングシステム
- 将棋プログラム

* 理工学部メディア情報システム学科

「アルゴリズムとデータ構造」、「データベース応用」では、定期試験問題の一部を示し、授業のレベルを紹介した。また、「オペレーティングシステム」では、最近の PC でサポートされている仮想空間を取り上げた。これらは授業の内容に沿ったものであったが、これらのほかに、科大祭のために、次のネットワークアプリケーションを用意して展示した。

(1) Web アプリケーション
　① 講座の受付システム
　② 受講状況登録システム
　③ 将棋プログラム

(2) C/S（クライアントサーバ）システムアプリケーション
　① 講座の受付システム
　② 将棋プログラム

これらのシステムのうち、上記（1）②の受講状況登録システムは、本学の講座である「データベース」、「データベース応用」で取り上げ、デモで見せたり、作成法を講義したり、実際の授業の中で使用させたりしているほか、フレッシュセミナー（本学独特の授業で、1 年生の前期に実施するもの）でも使用実績があった「出席管理システム」をベースにしたものであったが、その他のシステムは、科大祭のために準備したシステムで、平成 16 年の夏休み期間すべてを割いて開発した。

これらのシステムを取り上げたのは、本学の講座である「オペレーティングシステム」では、Windows の基礎技術を取り上げていて、Web アプリケーションや C/S アプリケーション等の応用技術は、デモ等で簡単に紹介している程度であることから、本学の学生向きに、これらの応用技術を実際に使わせることを考えたためである。

2. 将棋プログラムの品質について

上記 1 の（1）③、および（2）②の「将棋プログラム」は、本学の講座「アルゴリズムとデータ構造」におけるプログラミング教育の中で講義している対戦ゲームのプログラムである「将棋プログラム」をネットワーク化したものである。

その元の対戦ゲームのプログラムは、講義で取り上げているアプリケーションの開発方法の適用例として、本学就任後に C 言語で開発したものであるが、テキスト掲載後バグが出ず、上で説明したネットワーク化した際にもバグが出なかったと思う。そのことからして、品質が確保できたと考えてもよいのではないかと思われる。

そこで、当プログラムで品質が確保できた理由を以下考えてみたい。

(1) オブジェクト指向技術は使用しなかった。

　「アルゴリズムとデータ構造」は、当初、1 年後期の講座であったことから、オブジェクト指向技術は使用していない。したがって、これまで使い慣れた C 言語で開発することになった。

(2) 最新の開発環境を使用した。

　使用した開発環境は Microsoft Visual Studio .NET で、使用した言語は C++ .NET であることから、強力な型チェック機能（例えば、関数の定義で指定した引数の型と異なる型の実引数を指定した場合等）とか、Intellisence 機能（ソースコード入力時、例えば、関数の呼び出しを指定する際に、関数の定義を参照できるように表示してくれる等、格段に楽になっている）とかの進歩した技術の恩恵を享受することができた。

(3) 分かり易さを追求できた。

　下で述べるアルゴリズム表記法を採用したが、この手法をこの講座で採用したのは、教師の考えを学生に伝えるためのコミュニケーション手段が重要であると考えたからである。

　ただし、このような規則を与えたことにより、規則に縛られたプログラミングという姿勢をとっていると見えなくはないことから、硬い（凝り固まった）開発スタイル*) の教育をしていると見えてしまうという側面は持っていると思われる。

　いずれにしても、将棋プログラムについては、分かり易さを追求して、ごく限られた言語仕様を使用し、取り決めた規則に全くはずれることなしに開発するという硬い開発スタイル*) で開発したことが、高信頼性の確保につながった一因になっていることは否定できないと思われる。

アルゴリズムの表記法を設定する前に、まず、使用する言語仕様例として、次のように、C 言語のサブセットを考えるとした。

- 整数型の変数と定数を扱うことができる。
  - int a,b,c;　・・・・・a、b、c という名称の変数を使うことを宣言する。
    a=1;　・・・・・変数 a に 1 という定数（値）を代入する。
    b=2;
    c=a+b;
- 四則演算をすることができる。（演算した結果を変数に代入することができる。）
  - int a,b,c;
    a=5*4+2;　・・・5×4 に 2 を加えた結果を変数 a に代入する。
    b=a/2;　・・・変数 a の値を 2 で割った結果を変数 b に代入する。
    c=a-b;　・・・変数 a の値から変数 b の値を引いた結果を変数 c に代入する。
- 制御の流れを変えることができる。（制御の流れを変える特別な命令を指定しない限り、命令を書いた順序に従って実行して行く。）
  - 条件を判定して制御の流れを変更することができる。
    - ✧ int a,b;
      if (a>5)
        b=0;
      else
        b=1;
      変数 a の値が 5 より大きければ次の処理を実行する。
        変数 b に 0 を代入する。
      違う場合には次の処理を実行する。
        変数 b に 1 を代入する。
    - ✧ if （式)
      文－1
      else
      文－2
      なる命令で
      - ➢ else 部は省略することができる。
      - ➢ ・・・
      ・
      ・
  - ・・・
  ・
  ・

次に、次のような、アルゴリズムの表記法を設定した。

- 条件判断 1
  - 日本語によるアルゴリズムでの表現
    1　“A”の場合には、次の処理を実行する。
      1.1　処理 B を実行する。
      1.2　処理 C を実行する。

    ｝条件を記述した文よりもポイントシステムで 1 レベル下の番号を付けた文で、その条件を満足した場合に実行する処理を指定する。

2　違う場合には、次の処理を実行する。
　2.1　処理 D を実行する。
　2.2　処理 E を実行する。

“違う場合には”と記述した文よりもポイントシステムで 1 レベル下の番号を付けた文で、“違う場合には”と記述した文に付けた番号より 1 少ない番号の文で指定した条件を満足しなかった場合に実行する処理を指定する。

3　処理 F を実行する。

・・・
・
・
・・・
・
・
・・・
・
・

(4)　「アルゴリズムとデータ構造」で取り上げた他のアプリケーションでの思考が役立った。

「アルゴリズムとデータ構造」では、将棋プログラムのほかに、ます目を処理するプログラムとして、2 次元の幾何の問題（2 次元の平面上において、パラメータにより指定した直線が、x=0、2、4 の 3 本の直線と、y=0、2、4 の 3 本の直線により作られるます目のうち、通過するものの個数を求める問題）と、五目並べのプログラムを取り上げたが、そこで使用したアルゴリズムを部品化して流用するなどは行わなかったものの、そこでの思考過程が役立ったと思われる。

(5)　インメモリの Windows コンソールアプリケーションとして開発した。

「アルゴリズムとデータ構造」で取り上げた将棋プログラムは、Windows のコンソールアプリケーションとして開発し、すべて、メモリ内で処理していることにより、ファイルとか、通信とかという要素が入らず、プログラムロジックが複雑化しないで済んだ。

これらの要素が入ると、プログラムのロジックが複雑になることから、テストしなければならないケースも多くなって、開発に要する時間と労力がケタ違いに大きくなる。その例として、本学において実際に起きた不具合を 3 で紹介する。

以上のような理由が考えられるが、将棋プログラムというアプリケーションを開発するための最良の開発方法をとった結果、信頼性が確保できたというよりは、他の事情により取らざるを得なかった開発スタイルが、たまたま良い結果に結びついたといえるのではないかと思われる。

すなわち、初心者に対する手本となるようなプログラムの開発手順に従って開発することを開発方針としたことにより、結果的に、硬い開発スタイル*) で開発したという形になったが、信頼性は確保できたということだといえるのではないかと思われる。

＊）

この硬い開発スタイル（rigid style of development）という用語は、「The Puzzle of Japanese Software」（Michael A. Cusumano 著）（Communication of the ACM July 2005 Volume 48, Number 7）で使用されている用語を使用した。

そこでは、日米のソフトウェアの違いということで、日米それぞれのプロジェクトにおける、出荷後の最初の 12 ヶ月間に、顧客から報告された事故（不良）の件数が紹介されている。

それによると、1990 年の調査（日本 11 プロジェクト、米国 20 プロジェクト）では、日本の 0.2 に対して米国 0.8 と 4 倍の差であった。さらに、2003 年の調査（日本 27 プロジェクト、米国 31 プロジェクト）では、20 倍の

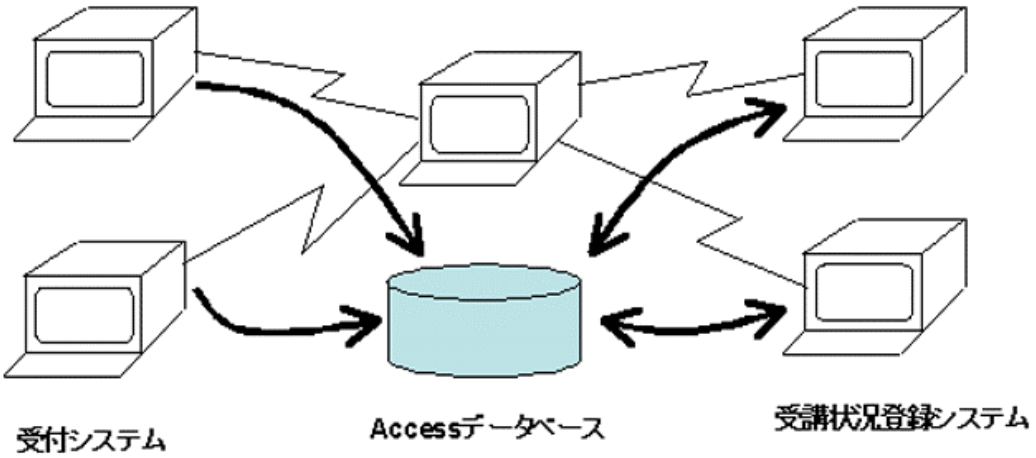
差があり、ソースコード 1000 ステップ当りの平均不良の数が、日本の 0.02 に対して、米国 0.4 であることが分かったとしている。この差については、将来、プロジェクトをもっと詳細に検討することにしたとしているが、この数値から懸念されることとして、少なくとも、サンプルとして選択した会社においては、革新とか、実験的なことよりは、「極度に硬い開発スタイル」と、「不良ゼロ」を重要視していることを示唆しているかも知れないとしている。[1)]

ここでの硬い開発スタイルといっているものに、上で述べた将棋プログラムの開発スタイルそのものが相当しているわけではないと思うが、或る一面は持っているのではないかと思われることから、この用語を使用した。

3. 複数の要素が関係したために生じた動作不具合の例

科大祭用に開発した「受付システム」と「受講状況登録システム」を、本学の「データベース応用」の講座で、学生に使わせてみたが、その際に、以下に示すような不具合が発生した。

(1) 「受付システム」と「受講状況登録システム」の概要



両システムでは、上図に示すように、まず、「受付システム」で、受付けをし（複数人で、同時に受付が可能）、受講を許可した顧客を Access のデータベースに登録する。そして、その顧客は、「受講状況登録システム」で、その登録データを使用して、受講状況を Access のデータベースに登録する。

さらに、これらのシステムは、一時的に、備え付けの受講用の PC の数より多くの顧客を受付けて、待ちが生じることがあることを想定したシステムになっている。

① 「受付システム」

顧客が受付に到着したら、入力フォーム上の入力欄に氏名を入力し、「受付」ボタンをクリックすると、自動的に、全体として上昇順になるような顧客番号を発行して、その顧客に与える。

確認用のフォームに対して、ＯＫボタンをクリックすると、顧客番号と氏名を「待ちテーブル」に登録する。

その後、受講用の PC に空きができたら、「ご案内」ボタンをクリックすると、待っている顧客のうち、顧客番号の一番小さい顧客の情報を「待ちテーブル」から表示して、その顧客を席に案内する。

その後、OK ボタンをクリックすると、待ちテーブル上のその顧客の情報を削除し、その顧客の顧客番号と氏名の情報を、「受講状況登録システム」が管理する「受講者テーブル」に登録する。

上記の操作を通常の操作とするが、案内処理では、特別な場合の対応策として、特定の顧客番号を指定して案内処理を実行することもできるようにしている。

② 「受講状況登録システム」

受講者は、PC から、受付け時に割り当てられた顧客番号とパスワードを使ってログインし、自分の受講状況を入力する。

以上のように、「受付システム」で割り当てる顧客番号は、異なる顧客に対して同一の番号を割り当てることはないようにし、000001 から、受け付けた順番に 000002、000003、・・・と割り当てている。

(2) 不具合の現象

「受付システム」で管理している「待ちテーブル」上に、待っている顧客の顧客番号と氏名が登録されていて、さらに、「受講状況登録システム」が管理している「受講者テーブル」上に、受講者（顧客）の顧客番号と氏名が登録されており、「待ちテーブル」上に登録されている顧客の顧客番号の最大のものが、「受講者テーブル」上に登録されている顧客の顧客番号の最大のものより大きい状態で、いったん終了し、その状態で再開始すると、次のような異常が生じる場合がある。

すなわち、新たに受付けた顧客を「待ちテーブル」に登録しようとする時点で、その顧客番号を持ったデータがすでに「待ちテーブル」上に存在するというエラーになる。

上記のような状態で再開始することは、科大祭当日では無かったことと、事前のテスト、リハーサル等で再開始する場合には、必ずデータベースを初期化していたと思われ、この不具合のケースが発見されずに済んでしまっていたと思われる。

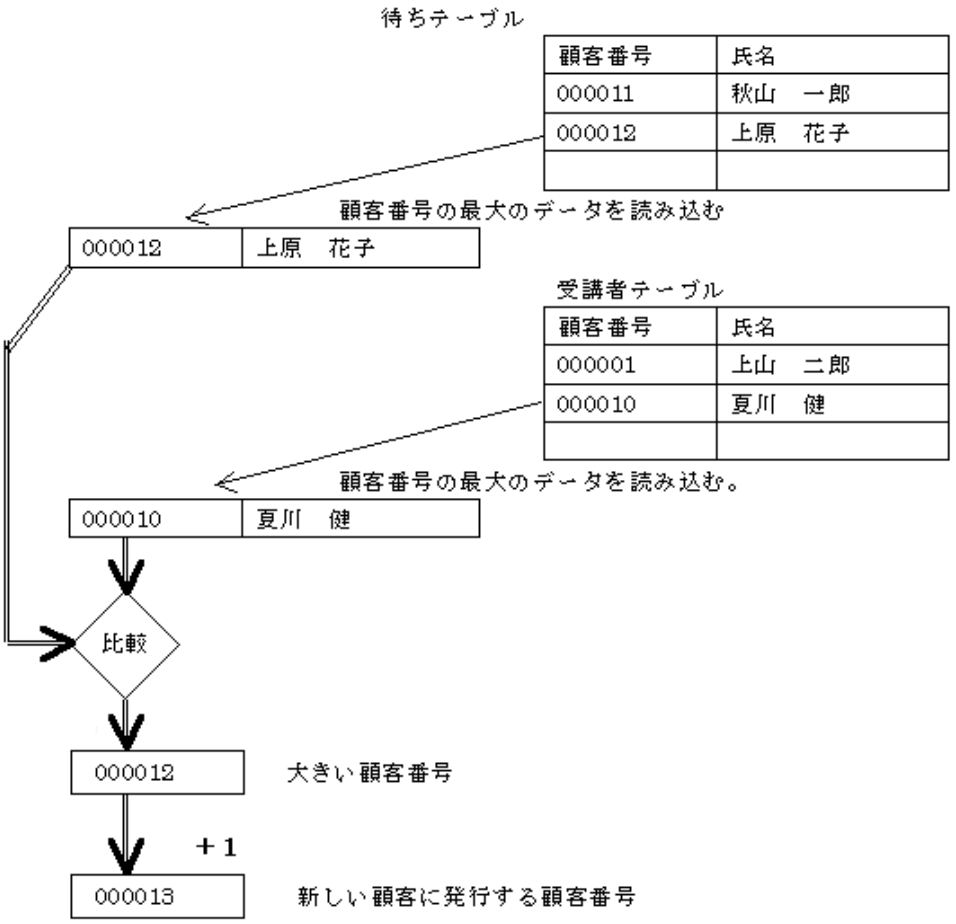
授業で、学生に使用させていて不具合が発生したときには、学生１人１人に、受付処理と案内処理を自分の PC から実行させていて、その際の案内処理では、受付処理時に自分の名前を登録した際に発行された顧客番号を指定して案内させていた。

そこで、授業終了時に、たまたま上記のような状態で終了してしまったために、以降の授業で、その状態のまま再立ち上げをしたことにより不具合が発生したのであり、授業終了時に、「待ちテーブル」上に登録されている顧客の顧客番号の最大のものが、「受講者テーブル」上に登録されている顧客の顧客番号の最大のものより小さかったら、この不具合は、顕在化せずに、潜在したままであったのである。

(3) 不具合の原因

① 不具合を起こした部分の処理手順

「受付システム」を立ち上げた後、最初の顧客を受付けたときに発行する顧客番号を次のように決定している。



上記処理を実行するのに、「待ちテーブル」、「受講者テーブル」とも、Access のデータベース上のテーブルを使用しており、両テーブルから、それぞれの顧客番号の最大のデータを読み込む際に ADO.NET（Microsoft .NET Framework に組み込まれているライブラリで、データベースとのやり取りを行うもの）を使用して読み込んでいるが、その際に、それぞれ、SQL 文（リレーショナルデータベースアクセス用の国際標準の言語仕様に従ったステートメント）を指定している。

② 不具合の直接的な原因

上記①の処理で、「待ちテーブル」から顧客番号の最大のデータを読み込む際に使用している SQL 文にエラーが含まれていた。

SQL 文は、区切り記号として、半角の空白を使用してもよいことになっているが、それぞれの半角の空白のうちの 1 つに、全角の空白を使用してしまっていた。

そのようなエラーがあると、通常は、実行時にその旨のエラーメッセージが出るが、たまたま、その全角の空白が全角の文字の直後にあった場合には、データの読み込み命令は正常終了してしまって、実際には、意図した検索が実施されないという現象が起こる場合があり、今回の不具合では、その場合に相当して、上記例で、「待ちテーブル」上からは、データが読み込まれなかった。従って、「待ちテーブル」上にはデータが存在していないと判断し、新しい顧客の顧客番号として 000011 を発行してしまい、その顧客のデータを「待ちテーブル」に登録しようとして、それと同じ顧客番号のデータがすでに「待ちテーブル」上に存在していたために、エラーになった。

③ 不具合が作り込まれた原因

正規の、プログラム作成手順に従うと、上記の SQL 文は、Web アプリケーションを Visual Studio .NET で開発する時点で、ウィザードを使用して入力したものから、テキストを自動的に生成していて、この際に、全角の空白が含まれていた場合には、半角の空白に修正して、正しい SQL 文に訂正している（これは、この事故の原因が判明した後のテストで分かったことである）。

したがって、この正規の手順に従ってプログラムを開発していれば、この不具合は発生しなかったのである。

実際には、プログラム作成時に指定した SQL 文に対して、実行時に、そのテキストの一部をプログラムで修正していたのである。そして、その修正時に、全角の空白が入ってしまったのが原因である。

しかも、このように、ウィザードが作成したプログラムを、ソースプログラムレベルで編集したり、プログラム実行時に修正してはならない旨のコメントが、コンパイルリストに示されていたのである。それにも拘らず、この部分のロジックを、上で示したようにプログラム化してしまったときには、このコメントのことを忘れていたと思われる。

このように、コンパイラとか、開発環境とか、データベースとか、SQL 文とかの要素が関連してくると、エラーは入り込む割合が増して、設計するとき、プログラムをコーディングするとき、テストするときに細心の注意が必要になるのである。

④ 今回のこの不具合の教訓

規定されている規則、仕様に従って開発することが何より重要であることを実感させられる不良であった。

この不具合のほか、例えば、将棋プログラムを C/S システムアプリケーション化した際に、単純なプログラムミスから、一組目の対戦グループに対する対戦をサポートしている際に、二組目の対戦グループのサポートを始めると異常になるというようなエラーも生じたが、これは、事前のテストで見付かり大事に至らなかったものの、一組目のテストだけ実行して、二組目以降のテストを実施しなかったら確実に事故になったような不良であったことからも、事前のテストの重要性と、それにも増して、不良を作り込まないようにすることの重要性を実感させられた。

特に、この不良に関しては、研究室内では、PC の数から、一組目のテストは簡単にできるが、二組目の場合のテストができないという問題があり、科大祭の前の事前のリハーサルで見付かったものである。このリハーサルの重要性については、当時の学科長の山本先生からも指摘されたこと

である。

これらの不良からも分かるように、関連する要素が多くなると、エラーの現象も複雑になり、原因の追究も大変になるが、この際には、開発環境におけるデバッグのし易さが重要になり、Visual Studio .NET では、この点についての進歩も感じられる。

4. 最後に

本論文で報告した信頼性とか、事故とか、不良とかについては、使用した環境が Windows に限られていたものの、Windows に限っても、OS のバージョンアップが 3.1～XP と行われ、開発環境も、それに従って新しくなって、OS、開発環境ともに、バージョンアップにつれて、新しい技術が取り込まれて進歩していることが感じ取れる。

そのことからも、新しい OS とか、開発環境とかを使用し、対応していくことが、なによりも重要なのではないかと思われる。

このようなことに関しては、2 で紹介したMichaelの論文でも、問題点の 1 つとして、日本の生産者は、WindowsとかUnixとかのグローバルのプラットフォームの採用が遅いと指摘しているのである。[1)]

参考文献

1. Cusumano, Michael A. : The Puzzle of Japanese Software. *Commun. ACM*, 48(7) : 25-27, 2005