# 情報システム技術論'96 ＃1

久野　靖*

1996.5.7

## 1　はじめに

- 「情報システム技術論」は科目案内によれば「計算機と通信を中心とする情報技術において…問題となる事柄を、ソフトウェア工学、プログラミング言語の両面から論じる」となっていますが…
- 実際にやる方としては、この専攻/コースの科目はすべて今年から始まるので、どのようにするか思案している所ではあります。
- 言い替えれば、学生さんの要望に応じて興味を持ってもらえることをやりたい、という気持ちは多いにあります。
- ただし、博士課程の科目ということで、修士までにあるような入門的な内容の科目にするつもりはありません (入門が聞きたい人は修士の科目を聞きに行きましょう)。専門科目は取りたい人が取ればいいので、無理矢理誰でも聞けるレベルを設定する必要はないと思います。
- 逆に言えば、その専門分野を主に修めている人が面白いと思うレベルは譲れないところです。ただ、そのような人でもちょっと専門外の事柄については知らない部分も多いはずですから、そのようなところを狙えば結構多くの学生さんに興味を持ってもらえるレベルになるのではと思っています。
- もちろん、今日顔を出して「やっぱりパス」にしても何ら問題ありません。気にする必要もありません。
- とりあえず今日は、来て下さった方すべてに興味を持ってもらえることを考えて見ます。
- それは…「計算機分野の教育/研究って具体的に何をやるの?」

## 2　計算機分野とは?

- 計算機科学とは何?
- CS(Computer Science)、IS(Information Systems)、ISE(Information Science/Engineering)の区分?
- 情報システム学?
- 結局、何を理解することが重要か? 具体的に、皆様にとって?

少し、自分の頭で考えて見ましょう… また、自分のバックグラウンドではそれらのうち、どの部分がカバーできていると思いますか?

*企業科学専攻/GSSM

## 3　今後の進め方

- 今後の進め方ですが、うまくまとまるなら皆様の希望を集約して 4 つ程度領域を選び、その部分について私も調べて紹介し、皆様も適当な論文を選んで紹介して頂こうと思います。
- そもそも 1 コマのために毎週出てくるのでは大変そうですから、2 コマ×5 回で済ませ、19:45 以前に来られない人がいるようならその分遅くまで実施したいと思いますがいいですか?
- 紹介するのは論文としては、(1) 英文のもので (2) できれば雑誌記事よりは学術論文、をお願いしたいです。(一応、英文よみの訓練のため。日本語のは渡しておけば読んでもらえますよね。)

## 4　論文紹介: Evaluation of Safety Critical Software

- 計算機を safety critical な用途に使うこと→増えている。
- じゃあ、どうしたらその安全性を保証できるの?
- ソフトウェア→ weak-link behavior (弱いところで決まる)
- s/c なソフトはできるだけ小さく単純に。すべての部品はまた s/c でなけ r ばならない。

### 4.1　なぜソフトウェアは特別?

- 計算機のせいで事故やトラブル…よく知られている
- 「一発で動いたリアルタイムソフト」を募集→どれも不可
- ソフトはエラーでぼろぼろなのが普通
- 他のものよりソフトのエラーが問題になる理由?
- どうしたら改良? 例: クリーンルーム、ランダムテスト

### 4.2　なぜソフトウェアを使うのか?

- そんな信用のおけないものは使わない?

1. システムにもっと論理を組み込める。組み込んでも安価。ソフトによる定期チェック→ハードの信頼性。
2. 変更が簡単。とくに触るのが困難な場所に設置する機器とか。
3. 運用者により多くの情報を提供。

### 4.3　ソフトウェア制御はどう違わない?

- まずは同じところを考える。
- ブラックボックスモデル…(1) 入力関数、(2) 出力関数、(3) 出力と入力の関係、(4) 必要な制約→検証
- ソフトウェアによる制御でも同じ。でも…

### 4.4 ソフトウェア制御はどう違う?

- 複雑さ…システムそのものが。プログラマの頭に入るのに時間が掛かる。
- エラーに対する感度が高い…tolerance の考え方が通用しない
- テストが困難…アナログ機器の中間補間は使えない。
- 非独立…複数の要素が同時に fail する確率は乗算、という考えが使えない。複数機器を搭載してもよくならない。
- 作業者に対する標準がない…誰でも書ければ「ソフトウェアエンジニア」。システムのプロでもソフトウェアはアマチュアだったり。

### 4.5 ソフトウェアのテスト

- ブラックボックステストは不可…内部が線形や多項式ではないから。
- 組み込みテストは無意味…ソフトは劣化しない。しかも複雑にはなる。
- テストではバグの存在は示せるが正しさは示せない。
- 信頼性を予測するのが難しい (連続した運用に入ってからなら取れるが…)
- 可用性 (availability) はもっと難しい。虫取りにどれだけ掛かる?
- trustworthy(パニックしない) と reliable(信頼できる) の違い。テストで trustworthy さは増やせるが信頼性は増えない。
- テストの役割…テストしてはいけない、という人もいるが…
- 独立した検証機関 (Verification & Validation) の必要性。場合によってはライバル企業が受注しても。IBM のクリーンルーム方式では、ソフトを作成した本人は動かしてはいけない (!)…テストする人だけが動かす。乱数でケースを選ぶ。
- どんなテストかは予め分かってはいけない。

### 4.6 ソフトウェアのレビュー可能性

- レビュー可能性 (reviewerbility) とは…
- これまでは、プログラムの書き方は個人の趣味。他人のソフトを調べろとは言われない。
- 今後…s/c なのだから、ドキュメントはもちろん、コードの書き方も標準化。
- これまでの方式でもレビュアーが絶えずレビューをする→ソフトウェアでも同様に。レビューの標準。
- しかしソフトは「見える」点は問題ないが、「正しいか分かる」という方は問題。簡単なソフトで間違いがあるのにプロのレビュアーに分からなかったという実験! (ひどいコードだった。)
- だから「ソフトウェアはアート」は不可。標準に従って。ただし、誰も見ない文書を大量に生成するような標準も不可。
- ソフトウェアに標準がないのが困る。標準があっても曖昧だったり。

#### 4.6.1 何をレビュー?

a. 機能が意図した通り?
b. 保守可能? 理解可能? ドキュメント? 構造?
c. モジュールごとに、データ構造を検証。アルゴリズムに照らして、数学的に。
d. それに基づいてコードを検証。
e. 正確さ…テストの量と質

- ソフトウェア工学のアプローチと同じ。全部まとめてではなく、1度には1つの側面ずつ検討。
- レビュアーの資質に合った側面をレビュー。
- レビューが可能なコード/ドキュメントの品質であること。書いていないことは想像してはいけない。

#### 4.6.2 どんなドキュメント?/レビュー?

- 入出力の規定…(1) 環境の変数、(2) 計算機への入力、(3) 出力
- プログラム構造の文書…(1) 構造、(2) 列挙、(3) インタフェース、(4) 機能。→文書としては (1) 要求仕様、(2) モジュールガイド (informal)、(3) モジュール仕様 (formal)。
- モジュール内部の文書…データ構造、トップダウン設計など…ただしプログラミング言語のプロは必要ない。物理現象のプロ。
- コードレビューの文書…プログラミングのプロ。
- テストの文書…統計的手法も重要。

#### 4.6.3 文書間の関係のレビュー

- 独立した視点からのレビューがうまく行くには、各文書の整合性がなければだめ→そのレビューも?

#### 4.6.4 構成管理

- せっかくレビューしても違う版のを出荷したら…構成管理が重要

### 4.7 モジュール構造

- データ抽象、情報隠蔽、カプセル化、オブジェクト指向などと呼ばれているもの。

### 4.8 信頼性評価

#### 4.8.1 そもそも評価すべきか?

- 信頼性評価の長い歴史
- ソフトウェアの信頼性 $= 1.0$?
- しかしながら、統計的性質はあることが事実。e.g. MTBF
- ある入力が来たためのエラー→入力の統計的性質
- ソフトはシステムの要素ではない?

#### 4.8.2 何を評価すべきか?

- 信頼性 (reliabllity) とは、失敗する入力に出会う確率…厳密にふるまいが規定できていれば、入力の分布からパニックの頻度も分かる…が実際にはむり。そもそも失敗しては困る!
- だから、重大な虫がすり抜けてしまった可能性の低さを評価…trhustworthyness
- 可用性 (availability)…使用可能である時間時間割合。回復に要する時間に強く依存。

### 4.9 有限状態モデル

- ソフトウェアの誤り率は各サブルーチンの誤り率からは計算できない。
- 有限状態機械モデル: 最初にローディングした時が初期状態。next-state と output という 2 つの表で表される。
- ランダムテストで、決められたエラー確率以下になるまでシステムをテスト/虫取りしていく…式 (1)
- これらの値から trustworthyness を計算…式 (2)

#### 4.9.1 3 つのクラスのプログラム…

- バッチで、メモリを残さない…毎回最初からやればいいから簡単
- バッチで、メモリを残す…状態を乱数で生成するか、複数パスを走らせる
- リアルタイム…ずっと走っているから問題。適当な期間で区切り、期間をまたがる状態情報が少なくて済むならバッチと同様にできる。
- 適当なテスト期間/状態を設定するのが肝心。

### 4.10 結論

## 参考文献


[1] デニング図による情報関係専門学科カリキュラムの評価, 計算機教育シンポジウム, pp. 103-110, 情報処理学会, 1991.

[2] P. J. Denning et. al, 木村訳, 学問としての計算機分野, 情報処理, vol. 31, no. 10, pp. 1351-1372, 1990.

[3] 野口, 中森, 大学等における情報処理教育の諸問題, 情報処理, vol. 31, no. 10, pp. 1373-1389, 1990.

[4] 中嶋, 浦, 情報システム学の誕生とその現状, 情報処理, vol. 36, no. 10, pp. 914-920.

[5] Parnas, D. J. , et. al., Evaluation of Safety-Critcal Software, CACM, vol. 33, no. 6, pp. 636-648, 1990.