コボル文化の人と共存共栄 V2

汎用機系開発者とオープン系開発者が円満に開発できるために

なぜ、現場では、衝突するのだろう?

Ognac

コボル文化の人と共存共栄 V2

- 自己紹介: Ognac
- 高校の時に取得したアマチュア無線のコールサインがJE3OGNでした。
- じいさん・おじんと揶揄されました。
- 旧郵政省は人を見ているのでしょうか。
- 過去形なのは、うっかり失効で無効にされてしまいました..orz。
  Automatic Computer をつけて名乗ってます。

コボル文化の人と共存共栄 V2

- 阪神間でフリーランスの.NET系システム開発やってます。
- フリーランス仕事無ければフリータ
- 現在浪人中です。
- お仕事、下さい。

- 今日の資料は以下にアップしています。
- http://ognac.wankuma.com/Download/SRC/PPT_DownLoad2.htm

- 拙いBlogを書いています。
- http://blogs.wankuma.com/ognac/

コボル文化の人と共存共栄 V2

- 略歴
- プライベート
- NEC_TK80以来のインテル系ユーザー
- Z80の対の裏レジも好きだった。
- 初期は68系が好きだったのですが....
- パブリック
- 汎用機: S/370VM/MVS・Cobol
- 中型機: S/36・RPGⅡ/RPGⅢ(AS400の前身)
- 小型機: IBM_AT系を経てMS系の開発

コボル文化の人と共存共栄 V2

- 要約概要
- コボル文化と称していますが、汎用機文化との対比です。
- 業務アプリは業務ルールの機械化。
- それを実現する機械はコンピュータ。
- 業務処理 と 汎用機・Web・クラサバの形態とは別の次元

- 現場で、汎用機系PMの開発手順とオープン系PMの開発手順とで温度差を感じた人いませんか。
- なぜ、温度差が生じるのでしょう。
- オープン系開発者にガスが溜まっていく現実を目にします。
- オープン系開発者からみて納得しにくいルールがあるのが、大きい。
- ルールの成立理由を 理解できるとガズも溜まりにくくなります。

コボル文化の人と共存共栄 V2

- 要約概要
- コボル文化と称していますが、汎用機文化との対比です。
- 業務アプリは業務ルールの機械化。
- それを実現する機械はコンピュータ。
- 業務処理 と 汎用機・Web・クラサバの形態とは別の次元

- 現場で、汎用機系PMの開発手順とオープン系PMの開発手順とで温度差を感じた人いませんか。
- なぜ、温度差が生じるのでしょう。
- オープン系開発者にガスが溜まっていく現実を目にします。
- オープン系開発者からみて納得しにくいルールがあるのが、大きい。
- ルールの成立理由を 理解できるとガズも溜まりにくくなります。

コボル文化の人と共存共栄 V2

- 要約概要
- コボル文化と称していますが、汎用機文化との対比です。
- 業務アプリは業務ルールの機械化。
- それを実現する機械はコンピュータ。
- 業務処理 と 汎用機・Web・クラサバの形態とは別の次元

- 現場で、汎用機系PMの開発手順とオープン系PMの開発手順とで温度差を感じた人いませんか。
- なぜ、温度差が生じるのでしょう。
- オープン系開発者にガスが溜まっていく現実を目にします。
- オープン系開発者からみて納得しにくいルールがあるのが、大きい。
- ルールの成立理由を 理解できるとガズも溜まりにくくなります。

コボル文化の人と共存共栄 V2

- 前回は、その話の概論とをコボルの構文から話しました。
- 前回の駒が、半分残してしまいましたので、その後半をします。
- 前回の資料は
- http://ognac.wankuma.com/Download/SRC/PPT_DownLoad1.htm
- 後半を、再編成しました。
-
- (注) 以下の話は、オープン系の立場から見ています。

コボル文化の人と共存共栄 V2

- 温度差はどこから生じるか
- 業務設計から実装までの過程の捉え方
- ウォーターフォールの誤った運用
- 古い言語ルールに起因するルールの堅持

- コボルやフォートランは歴史が長いので一つの文化を形成しています。オープン系も独自文化を形成しつつあります。
- 異なる文化が対峙すれば、摩擦が生じるのは必定。
- 互いに正論であっても、ベクトルが違うので
- 正論*正論 != 正論 になりがち。
- でも顧客には迷惑。
- 互いに理解しあって、よりよいシステムをつくりましょう。

コボル文化の人と共存共栄 V2

- 温度差はどこから生じるか
- 業務設計から実装までの過程の捉え方
- ウォーターフォールの誤った運用
- 古い言語ルールに起因するルールの堅持

- コボルやフォートランは歴史が長いので一つの文化を形成しています。オープン系も独自文化を形成しつつあります。
- 異なる文化が対峙すれば、摩擦が生じるのは必定。
- 互いに正論であっても、ベクトルが違うので
- 正論*正論 != 正論 になりがち。
- でも顧客には迷惑。
- 互いに理解しあって、よりよいシステムをつくりましょう。

コボル文化の人と共存共栄 V2

- 要件定義=>外部設計=>内部設計=>開発
- 　の流れは、同じです。でも汎用機系では、
- 　要件定義を砕いたのが外部設計
- 　外部設計を砕いたのが内部設計と考える人が居ます。

- 開発するプログラム本数を、外部設計で決めて見積もることが多い。
- 現実は、サブシステム間で同類のルーチンや画面があれば、基底クラスにして共通化を図るので、外部設計上の本数と製造本数は不一致です。
- 業務要件から砕いてきた工程管理表には、基底クラスを計上しにくい(表面化すると顧客との話が面倒?)

- 外部設計と内部設計は対等で従属関係ではない。
- しかし、従属関係と捉えるから矛盾が生じる。

コボル文化の人と共存共栄 V2

- 要件定義=>外部設計=>内部設計=>開発
- 　の流れは、同じです。でも汎用機系では、
- 　要件定義を砕いたのが外部設計
- 　外部設計を砕いたのが内部設計と考える人が居ます。

- 開発するプログラム本数を、外部設計で決めて見積もることが多い。
- 現実は、サブシステム間で同類のルーチンや画面があれば、基底クラスにして共通化を図るので、外部設計上の本数と製造本数は不一致です。
- 業務要件から砕いてきた工程管理表には、基底クラスを計上しにくい(表面化すると顧客との話が面倒?)

- 外部設計と内部設計は対等で従属関係ではない。
- しかし、従属関係と捉えるから矛盾が生じる。

コボル文化の人と共存共栄 V2

- ウオータフォールは良くできた手法、
- 適用の仕方が細かいからうまく機能しない。
- オープン系こそ、適切なウォータフォールが必要と考えます。
- 開発工程管理で重要なのは、各行程で業務運用に矛盾がないかをチェックすることです。実装手段を云々して失敗するケースが多い

- 語弊がありますが、下手な作りでもバグがなければ、通ります。
- ソース品質のチェックと業務品質のチェックは別物です。
- OOPの継承云々などは、実装手段なので、内外設計書に記述すると足枷になります。実装設計書に記載すべきでしょう。
- 実装段階で手戻りが生じても設計書に影響させないのが良いです。

コボル文化の人と共存共栄 V2

- 業務設計工程は業務要件のみで設計する。....のが基本
- 外部設計工程は、実現方法(汎用機、Web 、クラサバ)を考慮。
- 内部設計工程は、実装要件を考慮
- 実装設計工程は、実装方法を考慮
- ＝＞役割分担がアヤフヤだと工数ばかりかかり進展しない。

ここが押さえられていたら、上流工程を誰がしても、製造で衝突することは少ないです。

でも越権行為があり、それが温度差となって現れます。

平たく言えば、上流工程者が製造工程に口出したり、
画面などのプログラムを前提に業務設計するからですね。

コボル文化の人と共存共栄 V2

開発者の資質か生産効率か。

書いているソースの効率性・有効性を重視する開発者も有れば、

雛形を手直しして製造することで満足する開発者もいます。

是非でなく資質の問題かもしれません。

ソフト製造という工業工程でみれば、雛形模倣型のほうが管理しやすい面もあります。

遅くても進捗が確実なほうが安心できます。

これは、何所の産業にもいえますが、標準手順、標準化された部門は進歩が遅れても安定供給を選びます。

我々開発者はソース品質を重視しますが、経営的には安定することが優先されるので、ここでもガスが溜まります。

これば、対コボル問題でなく、対技術者なので、これくらいで。

コボル文化の人と共存共栄 V2

■COP文化

一度作成したソースの有効活用は、ソースレベルで浸透してます。

勿論、ライブラリ化してExeを内部からCallすることもできますが比率は少ないようです。

構文は　　/Copy file名　です。

Cの や javasciptの　Include文に相当します。

レコード定義文や、関数に相当する Perform文は /copyばかりだったりします。

Include文や Macro文は C#やVBにも欲しいです。実装して下さいMSさん。

■雛形文化、

新規開発であっても、白紙からソースを起こすことは少なく、既存の雛形を持ってきて、指定された個所を変更して納品することも多いです。

雛形を登録するときは、「xxxをyyyのように変更する」という文書か付きます。

内容やロジックを理解して変更すれば良いのですが、ロジックを読まないで変更だけする人もいます。バグったら大変なことになります。

これはオープン系のコピペ問題もおなじですね。

雛形化手順が徹底しているてバグり難いのは歴史が長く、手法が確立しているから?

コボル文化の人と共存共栄 V2

■occurs文化とSQL

SQLが普及してきて、コボルでも使えるようになりました。

といっても、コボル言語構文にSQLが加わったのではなく、

call exec　sql文　という形で呼びます。

working セクションと呼ばれる変数域に結果が戻ります。

ですので、言語の並びとしてのスッキリ感はなく、外部Module Callというよそよそしさがあるのは、しょうがないです。

そのSQLですが、従来のSAM/VSAM(汎用機の索引ファイル)の単純置換とみなして使用しているケースが未だに散見されるようです。

例えば 8月の売上集計するとき

select sum(売上額) from 売上File　where xxx　group xxxx　の一行でするところを

```
loop
    if(eof) Exit;
    1行取得;
    足し込み;
    goto  Loop;
EXIT:
```

コボル文化の人と共存共栄 V2

と書いて、平然としている人がいたりします。
指摘すると返ってくる答えは、「雛形がそうなっている」
自分で雛形を作ろうと思わないのが不思議ですね。
開発者個人の問題でなく、雛形文化だから、雛形作成者の問題になりますね。

コボル文化の人と共存共栄 V2

便利な機能もあります。配列と領域再定義機能は便利です。
一人の月別年間売上の行は

```
01 予定実績行.
  02  課コード.
  02  課名.
  02  予定実績 occurs 12.
      05 予定額   9(5).
      05 実績額   9(4).
```

のように記述でき、
集計は

```
      PERFORM VARYING MM FROM 1 BY 2 UNTIL I > 12
              COMPUTE SUM = SUM + 実績額[MM]
      END-PERFORM
```

のように使います。配列そのものです。

コボル文化の人と共存共栄 V2

視点はかわりますが、BBSのFAQに 0.1問題があります。

0.1 + 0.1 + 0.1…(10回) !=1.0

とならない。オカシイのでは…というものです。

「当然のことだ、なぜ判らない」という解答を目にしますが、

質問者を責めるのは酷です。

情報処理の学習経験者に対してなら良いのですが、コボル文化の世界では、学習する機会がなかった人もいます。

その人達の頭には、広域変数しか存在しないのと同様に、固定小数点しが存在しません。現実世界の10進数との差を認識しろというのは酷なようです。

0.1は半端な数字で 0.125がスッキリした数字だ..

一般の人には理解を得にくいでしょう。そういう面で汎用機文化の人は一般人に近いとも言えます。

われわれオープン系は、変人なのかも。

情報学を履修した人としてない人の温度差でもありますね。

コボル文化の人と共存共栄 V2

■便利は、基本を隠す。

脱線しますが、便利機能を使うと、基本が疎かになるのは宿命かもしれません。

雛形や標準化も同じことが言えます。

RDBが普及する以前は、複数のシーケンシャルファイルをマッチングさせるソート・マージのロジックの理解は必須でした。

ソートもバブルやシェル・クイックなど多種あり、1:1マッチング、n:nマッチングでロジックが変わったりしましたが、

今は、1行のSQL文で置換できます。

アルゴリズム習得が不要になったわけではありません。

でも知らなくても機能は実現できます。

そこに製造スキル矛盾が生じます。

メソッドやクラスを使うとき、実装方法を知らなくても使えるのがメリットで、仕組みを習得する必要がないのが「売り」です。

確かに、仕組みを知らなくてもテレビや電話は使えてますしね。でも、基本アルゴリズムは知っていて欲しい.....

高望みかなぁ。

コボル文化の人と共存共栄 V2

言語制約がプログラム設計に影響する。(彼らが規約に口出すとおかしくなります。)

前回と重複しますが、コボルには広域変数しかありません。

ローカル変数(スタック変数)がありません。

ローカル変数の有無は、製造設計に多大な影響があります。

関数は、仮引数を介して、スタック領域にメモリ空間を確保するので多彩なことが可能になります。 しかし、スタック変数という概念がないので、一元的なフローの思考になります。

「再帰コールは理解できないから使用禁止」ということになる。

継承関係もインスタンス内にスタック保持し、個別に値を保持しますが、その仕組みが理解できないので

「ややこしくなるので継承禁止」となったりします。

(細部規約になりすが)し 使用変数はプログラムのしで一括宣言する。

出口は一個所にする。

というのがあり、足枷になります。

変数は使う直前に定義して、不要になった段階で解放する..

この原則と相反します。

コボル文化の人と共存共栄 V2

「出口を一個所にせよ。」というのもありますね。

これはgoto文問題も絡み可読性が悪化します。

例外処理という文化が無かったので、例外を使って回避するという仕組みが、書かれていても、ピント外だったりします。

「各メソッド、プロパティには try() catch()を含めよ。
catch()で再度 throw せよ 」

となったりします。

コボル文化の人と共存共栄 V2

画面遷移はfunctionキー

汎用機でのオンラインは Web画面に類似していると言われます。確かに、Javascriptの使えない Webアプリに近いです。

でも、汎用機画面は 80col* 24行の固定画面で構成されますし、

submitに相当するのは enterキーでなく、実行キーだったりします。(今は、実行キーが存在しないので、右下のCtrlキーが相当)

submit機能は、上部のfunctionキーを使うこともあります。

functionキーを使っての画面遷移したいという要求は、未だにあります。

汎用機から置換した際、いままでの操作性を維持したい. 気持ちはわかりますが、それを実装するコストとリスクが見合うか。

Webアプリなら尚更、FunctionキーはBrowser固有のキーがあるのでアプリで弄るのは問題ありでしょう。

でもそのような、要求や設計をしてしまう上流工程者がいます。汎用機文化出身者に多いようです。

コボル文化の人と共存共栄 V2

自分の常識を押し通す意識はないようです。
操作性に対する認識が異なるとこから来ています。
操作性は端末がハード的に持っているもので、
プログラムで操作するものとは考えていないようです。

それ故に、内部設計段階で F4キー: xx機能　F5キー:登録確認....といつた表現が登場することになります。
開発者が指摘しても、開発者の技術不足だと一蹴するのは、問題です。
もっと理解して欲しいですね。
でも、PCの特性と汎用機端末の特性の差を説明して、説得させる技術者が少ないのも問題。

コボル文化の人と共存共栄 V2

■コボル使いは言語知識で仕事をする。
オープン系の人は、OSサービス、フレームワークのサービスを、
プログラムから呼び出して利用しますが、さほど意識しませんよね。
言語機能とフレームワーク機能の垣根が低いとも言えます。
　というより、言語はフレーワーク機能の接着剤であるとも言えます。

コボル系の人は、OSサービスを使うことが少ないのと、使うにしても身構えて使うので、一体感が少なく垣根が高いようです。
そういった背景があるので、ロジックを言語の範囲で考える習慣になったようです。
コボル系の人がオープン系を習得するとき、言語機能とOS機能、フレームワーク機能を学習することになり、一度に習得する項目が多すぎて消化不良を起こすようです。

VBの人がC#に移りにくいのは、事象を言語レベルで捉えるからでしょうね。
情報を加工するという視点で見れば、言語に左右されることはないと思います。
プログラマを言語で色分けするのは、悪習慣です。言語は道具に過ぎません。

コボル文化の人と共存共栄 V2

■最後に
好き勝手なことを言ってきましたが、
文化の差は伝わりましたでしょうか。
互いの文化を理解しようと勤めれば、溝は埋まりす。
しかし、拒否反応を示す人がいるのも事実です。互いの努力が要ります。
新人でも染まってしまうと、自分の環境万歳になり、他に対して拒否反応を示す人がいます。
　JAVA 対 Net
　旧VB 対 Net.VB
　Frame.1.1文化 対 FrameWork 2.0/3.x
も同類の構図を呈して来ました。
環境は、どんどん変わっていきます。VS2010や動的言語がどんどん登場します。
なにが残るかは判りません。
優れたモノでもポシャるかも知れません。

コボル文化の人と共存共栄 V2

基礎知識と言われているモノを押さえておけばなんとかなると信じてます。

新機能は、言語レベルでなくモジュールレベルで登場します。
言語は機能を引き出す接着剤に過ぎません。

コボル文化の人も、オープン系の基礎知識をマスターして欲しいです。
コボル文化も押さえて欲しいです。
オープン系の人も、現行バージョンの機能は押さえて欲しいです。

もっと相互交流して楽しい開発を!!