ソフトウェアの階層的開発

独立行政法人自動車技術総合機構
交通安全環境研究所　鉄道認証室

森　　崇

0. 登場人物

鉄道信号メーカー　カバ興業チーム

カバ興業　社長
座右の銘:技術と直感

カバ興業　営業　カバお
「怒られてナンボの毎日」

カバ興業　技術　オタかば
「面白くなければ技術じゃない」

カバ興業　プログラマ
ハッキングカバ
「俺しかできないことをやる」

カバ興業設計課長　カバ実
「全体のレベルアップ」

謎な奴

謎のフリーコンサル
なぞカバ
「知識は力！」

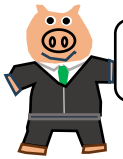
バチバチ

鉄道事業者　カバ鉄道チーム

カバ鉄道　社長
品格の経営、根拠ある経営

カバ鉄道　電気課長
口癖:安くて工工もん持って来い！

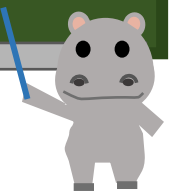
カバ鉄道　乗務員　カバどん
いつかは自動化されるかも。ドキドキ



ブタ工業　社長
カバ興業を凌ぐ強い体質づくり

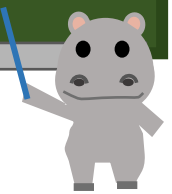
1. はじめに

- 1984年、私はパソコンを入手しました。アホみたいにはまりました。インターネットを導入したのと同じくらい衝撃でした。
- BASIC/Z80のアセンブリ言語/PASCAL/C/C++と順々にはまりました。
- 電子回路設計とは違い、いきなり始められて、いきなりアイディアを試すことができる。圧倒的なスピード感がありますよね。
- でも、後で見てそれは分かるでしょうか？他の人が見てわかるでしょうか？他の人が見て試験ができるでしょうか？
- 自分でも分からなくなって、全部消して、一から作ったことないですか？私はあります。これって仕事としては、どうなんでしょうね。。。



1. はじめに（本日の項目）

- ずばり、なんでこんなにめんどくさい要求が多い規格のいうこと聞かなければアカンのか、なんでGeneric Softwareはこんなにめんどくさいこと要求されてるのか
について話しします。
- データてなんや、ソフトがちゃんとしてればいいんやないか　についてもお話しします。



2. 前回あった議論

社長！いつもいつもしつこくてゴメン。でもな、やっぱりゆうとかなアカンことはあるねん。御社の保安装置、安全機能はSIL 4で作っとるよな。

そりゃまあ、そうしてますよ。そこについては私も口を酸っぱくして、ちゃんとせい、妥協するな！と社内で言っていますよ。フェールセーフのカバ興業ですし。

そこやねんな。この前ウチの社長から、「お前ら二言目にはフェールセーフフェールセーフゆうけどな。ワシはうたがっとんねん。」って言われたわ。

それは誠に遺憾ですな。社長さん  にはちゃんと理解していただかなくては。

いやー、この前な、「そんなもん、電子連動ゆうても、ちゅーりんぐましんやろ。言われたとおりしか処理せーへんモン、データ間違いやったら、どうしようもないやろ。そもそもソフトが間違っとっても、どうしようもないやろ。なんでそれがフェールセーフやねん。ワシまちごうてるか？」って言われたわ。大体ちゅーりんぐましんってなんや！てなったわ。

そこについては私も口を酸っぱくして、ちゃんとせい、妥協するな！と社内で言っていますよ。なんせウチは優秀なスタッフたちが。。。

いやー、多分社長は、すでにあらかた答え持っとんねん。あれはかなりタヌキやからな。「電子連動止めろとは言ってない。安全関連機能はSIL 4やろ。確率論だけやない。」ともゆうとったからな。
あれは地雷や。間違うと爆発する。タヌキは事務系のくせにヤバいで。ちゃんと理屈まとめて、報告頼むで。

（ま、丸投げやん。。）

2. 前回を受けての議論

おいハッキングカバ、ソフトな、ちゃんとせい、妥協するな！バク出すな！

ハァ？なんやそれ。バクってなんや。バグやろ。夢でも食うとけや。それで、ソフトは品質を買えってゆうたら、どうせまた「管理や管理、ワシが管理する」てゆうだけやろ。また団交しなあかんみたいやな。ちゃんと何が原因で何をせなアカンのか、しっかり考えなアカンのちゃうんか。

規格守れ！ 9001やれ、62279やれ！

ハ～！それってISO 9001のトップマネジメントか。呆れ果ててカバの開いた口がふさがらんわ。だいたい、なんでIEC 62279であんなめんどくさいことせなアカンか説明できるか？

それは、今までのベストを集めてきたからや。妥協するなベストをやれ！

あーあ、もう辞めたなってきたわ。技術屋っていうのはな、腑に落ちない限り、やったふりする場合あるねん。ちゃんと合理的に理由を説明して、なぜその努力が必要なんか、ちゃんと理解させない限り、品質向上なんて掛け声だけにしかならへんねん。しょうがない。天才プログラマのオレが、凡人に戻った気持ちで、今日はできるだけ課長と一緒に説明するしかないんか。

それがお前さんの仕事やろ。

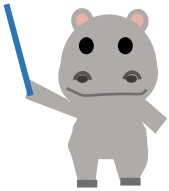
ハ～！なんでカバ興業労働組合の委員長のオレが経営者の仕事せなアカンねん。そういうの経営者の仕事ちゃうんか。まちごうてるか？労使のけじめ付けろや。ストするで。

ストはやめてくれ。満額回答するから。。。

3. ソフトウェア製作にどんなハザードがあるか

- システム要求仕様と安全要求仕様が不足
- ミッションプロファイルなど数値的な仕様が不足
- ソフトウェア仕様トレース不足で間違っている
- データが来ていないのに計算を始めたり実行順序依存やデータのセマフォ*などの定義が不足
- ハードウェアのポートアドレスやICへのデータプロトコルなど、インターフェースが間違っている
- ハードウェアの故障により、不正なデータが入力される対応がぬけている
- ハードウェア故障やノイズにより、処理異常、データの完全性をどうするかの定義と実装がない
- 用意されているライブラリや作成した関数の呼び出し方法、返り値処理が間違っている
- ソフトウェアコンポーネント(関数など)の仕様が間違っている
- ソフトウェアコンポーネント(関数など)のコーディングが間違っている
- ソフトウェアコンポーネント(関数など)の例外処理がコーディングされていない
- コンポーネントレベルの定義外例外処理のハンドリング定義がない
- Application data(駅や車両特有のパラメータや設定)が間違っている

*複数のプログラムなどが共有資源にアクセスする際の管理手法のこと



3. 事柄別に再整理

そもそも必要な機能の実現に応えていない

- システム要求仕様と安全要求仕様が不足
- ミッションプロファイルなど数値的な仕様が不足
- ソフトウェア仕様トレース不足で間違っている

安全要求が不足している

- ハードウェアの故障により、不正なデータが入力される対応がぬけている
- ハードウェア故障やノイズにより、処理異常、データの完全性をどうするかの定義と実装がない

ソフトウェアの構成に対して配慮不足

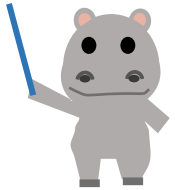
- データが来ていないのに計算を始めたり実行順序依存やデータのセマフォなどの定義が不足
- ハードウェアのポートアドレスやICへのデータプロトコルなど、インターフェースが間違っている
- 用意されているライブラリや作成した関数の呼び出し方法、返り値処理が間違っている

コンポーネント、関数における定義不足、ミス

- ソフトウェアコンポーネント(関数など)の仕様が間違っている
- コンポーネントレベルの定義外例外処理のハンドリング定義がない
- ソフトウェアコンポーネント(関数など)の例外処理がコーディングされていない
- ソフトウェアコンポーネント(関数など)のコーディングが間違っている

個別設定の定義不足、ミス

- アプリケーションデータが間違っている



3. 再整理を受けての議論

他にもハザードはあるかもしれないけれど、だいたいこんなもんかな。

こういうのはワシは頭に入っとる。言わずもがなや。でも全員がオレみたいな経験者やないから、こういうのもいるんやろうなあ。ワシはいらんけど。

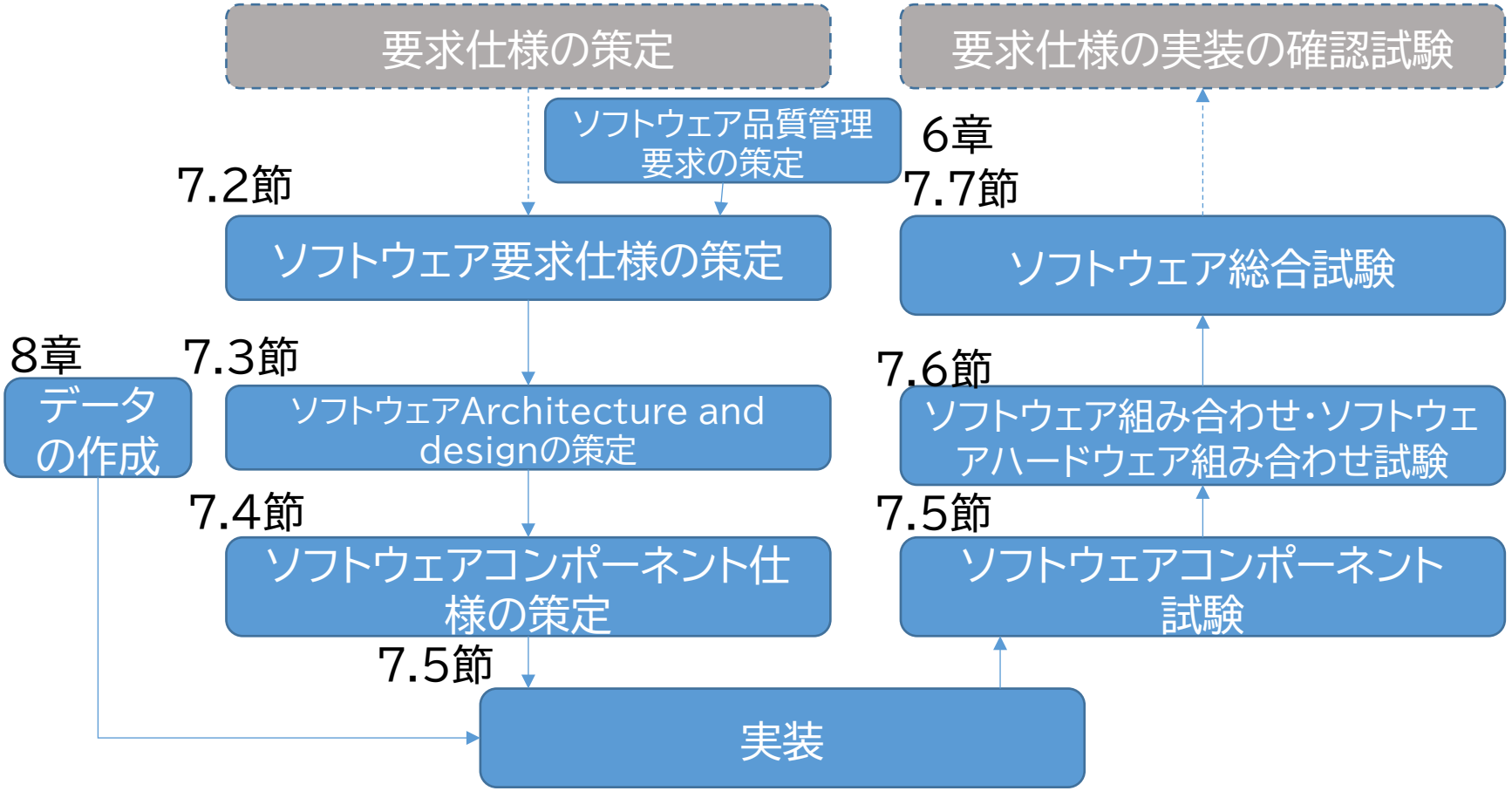
ハッキングカバは労働組合の委員長ならみんなのことを考えるべきでは。。

ハ～！職制に言われんでも分かっとるわい。支配介入かそれ！（冗談や）

次にIEC 62279のGeneric softwareの開発との対比をしたらいいかも。

社長が要求満額回答したからやるわ。

4. どのような手順で開発するのか（要求仕様から実装、試験）



上位で決まったことが下位にすべて含まれ、矛盾がないことを確認することで、要求仕様を実現する。

5. ソフトウェア要求について

IEC 62279 7.2章のうちソフトウェア要求に関する主な要求

- 完全で確認可能、試験可能な仕様とすること
- 入力図書に対しトレースバックできること
- 他のシステムのインターフェースを明らかにすること
- 関係するモードを明らかにし、かつ故障時のモードを定義すること
- 故障に検出に留意すること
- 安全機能を試験できる機能を要求に応じて持つこと
- 安全機能、非安全機能双方明確に同定すること

そもそも必要な機能の実現に応えていない

システム要求仕様と安全要求仕様が不足

ミッションプロファイルなど数値的な仕様が不足

これらはシステム仕様の問題

ソフトウェア仕様トレース不足で間違っている

安全要求が不足している

ハードウェアの故障により、不正なデータが入力される対応がぬけている

ハードウェア故障やノイズにより、処理異常、データの完全性をどうするかの定義と実装がない

ソフトウェアの構成に対して配慮不足

ハードウェアのポートアドレスやICへのデータプロトコルなど、インターフェースが間違っている

5. ソフトウェアアーキテクチャ、インターフェース仕様について(7.3章)

IEC 62279 7.3章のうち、アーキテクチャとインターフェースについての主な要求

- すべてのソフトウェア-ハードウェア間の相互関係の意味を明らかにすること
- どのようにソフトウェアを構成するかの戦略とそのソフトウェアコンポーネントを明らかにすること
- 以前から存在するソフトウェアの要件を定義すること
- 完全で確認可能、試験可能な仕様とすること
- 入力図書に対しトレースバックできること
- 故障のハンドリングをSILに応じて定義すること
- コンポーネントにおけるインターフェースの境界値内外の振る舞い、例外処理やタイミング、順番、同期、時間による出力を定義すること

安全要求が不足している

- ハードウェアの故障により、不正なデータが入力される対応がぬけている
- ハードウェア故障やノイズにより、処理異常、データの完全性をどうするかの定義と実装がない

ソフトウェアの構成に対して配慮不足

- データが来ていないのに計算を始めたり実行順序依存やデータのセマフォなどの定義が不足
- ハードウェアのポートアドレスやICへのデータプロトコルなど、インターフェースが間違っている
- 用意されているライブラリや作成した関数の呼び出し方法、返り値処理が間違っている

5. ソフトウェアデザイン仕様について

IEC 62279 7.3章のうち、ソフトウェアでサイン仕様についての主な要求

- ソフトウェアコンポーネントの機能、インターフェースを決めること
- 使用するデータの形式を決めること
- ソフトウェアコンポーネントのアルゴリズム、例外処理を決めること
- ソフトウェアアーキテクチャ仕様へトレースバックできること

だんだんコマかくなってきたな。で、なんかコンポーネントの話しになってきてるけど、これこの段階なんかが素朴な疑問やんな。

ソフトウェアの構成に対して配慮不足

データが来ていないのに計算を始めたり実行順序依存やデータのセマフォなどの定義が不足
ハードウェアのポートアドレスやICへのデータプロトコルなど、インターフェースが間違っている
用意されているライブラリや作成した関数の呼び出し方法、返り値処理が間違っている

コンポーネント、関数における定義不足、ミス

ソフトウェアコンポーネント(関数など)の仕様が間違っている
コンポーネントレベルの定義外例外処理のハンドリング定義がない

5. ソフトウェアコンポーネントデザイン仕様について

IEC 62279 7.4章のうち、コンポーネントデザインに関する主な要求

- コンポーネントがどのような関数で構成されるのか同定すること
- 詳細にインターフェースを定義すること
- 詳細にアルゴリズムとデータ構造を決めること
- ここで割り当てられたSILはそれ以上の細かい割り当てはしないこと
- ここに書かれている情報のみで、矛盾なくコーディングでき、試験仕様を作成できること

これは！この資料を見て、コードや構造体、試験仕様ができるレベルでないといかんということやな！

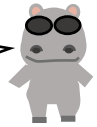
コンポーネント、関数における定義不足、ミス

ソフトウェアコンポーネント(関数など)の仕様が間違っている
コンポーネントレベルの定義外例外処理のハンドリング定義がない
ソフトウェアコンポーネント(関数など)の例外処理がコーディングされていない
ソフトウェアコンポーネント(関数など)のコーディングが間違っている

5. ちょっとまった！デザイン仕様とコンポーネント仕様はどう違うの？

あのさー、こういうのお客さんに説明する営業からしたら絶対言われることがあるんだよね。。

客なんかどうせ大したこと言わへんやろ。「なんか難しそうですね、すごいことやってますね。」くらいなもんやろ。



そこまでアホなお客さんはいないよ。なんか絶対突っ込まれるのは、「ソフトウェアデザイン仕様もコンポーネントを明らかにしてるんやから、ソフトウェアコンポーネント仕様はどういう目的なんや、カバおくんこの場で10秒で説明して。」ってカバ鉄道さんに言われるにきまっているんだよ。

そんなもん、細分化や！外部仕様と内部仕様やないか。カバでも分かる！

そんなの普通分かるわけないんだよ。それはハッキングカバだからそういうんだよ。みんな分かってないし、ハッキングカバも一度カバ鉄道さんに怒られてみたらいいんだよ。

5. ちょっとまった！デザイン仕様とコンポーネント仕様はどう違うの？

あのさー、ぼくのやっている見積もりとかそういう例やったら分かるんやけど。

この中身がコンポーネント仕様

見積もりソフトウェアデザイン仕様

入力：予定工数、人工単価、材料費、管理費率

コンポーネント1
見積もり案を計算する

与えられた条件をもとに単純に計算

出力：人工総額、材料費総額、管理費総額、合計金額、工期

コンポーネント2
技術側で調整

期間、材料費を調整

出力：人工総額、材料費総額、管理費総額、合計金額、工期

コンポーネント3
社長の判断

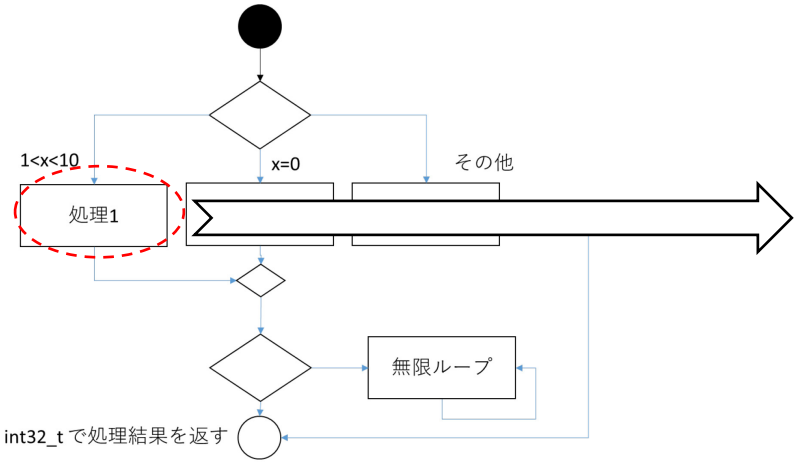
値引き、割り増しを調整

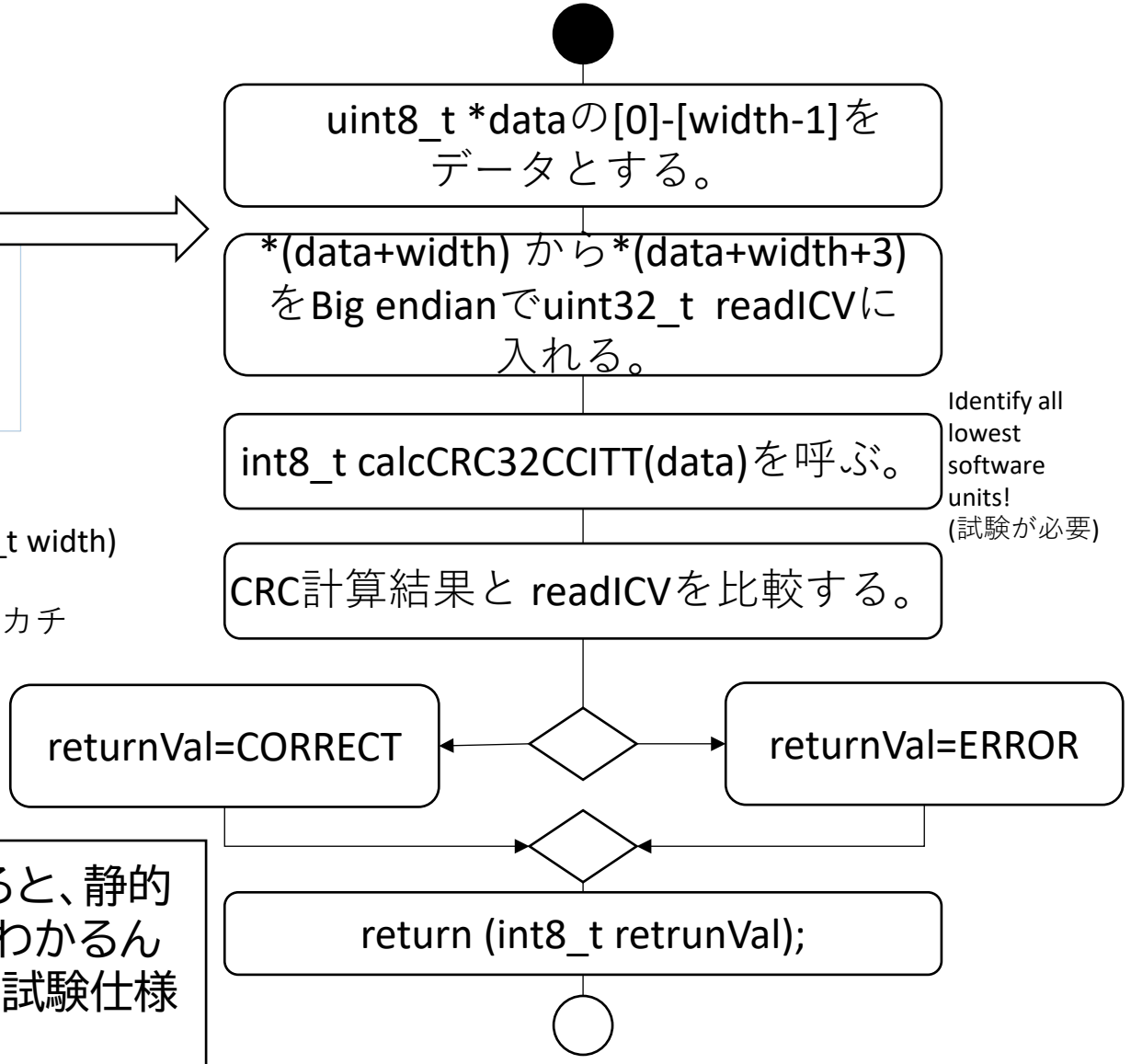
出力：合計金額、工期、アピールポイント

5. ソフトウェアデザイン仕様のモデリング

動作の概要と試験ができる情報、引数と出力がわかるようなモデルを選択するのがええな。

5. ソフトウェアコンポーネント仕様のモデリング





int8_t compareICV(uint8_t* data, uint16_t width)

*data ペイロードとCCITT-CRC32をコンカチネートしたもの。Big Endian
width データの大きさ(byte)
返り値 正常 CORRECT
　　異常 ERROR

このレベルまで落ちると、静的解析のための要素がわかるんじゃないかと思うで。試験仕様も作れる。

5. ちょっとまった！デザイン仕様とコンポーネント仕様はどう違うの？

まあまあ、なんかだまされたみたい。
10秒で説明せいと言われたらどうしよう。

ズバリお答えします！

ソフトウェアデザイン仕様

- コンポーネントをどういう順番で、使っていくかの仕様

ソフトウェアコンポーネントデザイン仕様

- コンポーネントの中身をどう作るかの仕様

6. Application dataとは

だいたいね、技術屋はみんな勝手なんだよ。お客さんはね、「ATC」なんかいらないんだよ。ソフトウェア設計出来たらオワリだと思ってる。

ええっ。何でいらないの？

どんなスゴイATCだってね、お客さんの線路条件とか、駅配置とか、運行条件とか、車両性能が反映されていなければ、「へー、フーン」で終わりなんだよ。「カバ鉄道のATC」がいるんだよ。

それは、アプリケーションデータがあればいいということですよね。

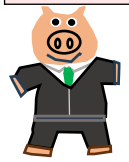
だからそこなんだよ。お客さんのデータがあればいいんじゃなくて、お客さんは要求通りちゃんと動作するかしか見ていないんだよ。視点が違うの、視点が！この辺はホント、ブタ工業が上手に営業していて、ヤバい状態になっているよ。だからお客さん第一主義、ブタ工業は、「お客さんに合わせてソフトしっかり作ってます！」なんてアピールしているよ。

6. Application dataとは

だいたいね、ブタ工業はこういうこと言ってアピールしているんだよ。
ウチはカバ鉄道様のためだけに、開発してますってね。

カバ鉄道殿専用設計ソフトウェアと混ぜ込んだアプリケーションデータ

カバ鉄道殿専用設計ハードウェア



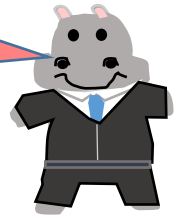
毎度おおきにブタ工業です。カバさんとこ高いですよね。
ウチはもっと勉強しますよ！

ええっ。それは、かえってよくないんじゃ。力の入れ方が違うと、リソース配分を誤り、品質が低下するのでは。メンテナンス性も疑問ですよ。。

6. Application dataの位置づけ

安全は、ランダム故障低減、システマティックエラー低減がキモ。その中でもソフトウェアはGeneric softwareとApplication dataが車の両輪。

お客様のニーズにお応えするのがここの部分です



Application data for Specific Applications

A線区 B駅-C駅間　ATC速度定義ファイルなど

Application data for Generic Applications

速度信号⇔ATC周波数テーブルなど

Generic Application Software

Application data for Generic system

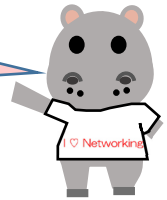
動作周期定義など

ATC論理関係

Generic product Software

FS共通動作、I/O関係など

多くのユーザに使用してもらえるよう必要なATC機能を開発実装しました！



当社の汎用FSコントローラを活用しました！

Generic Product hardware

FS CPU部　I/O部

Specific hardware

ATC送受信関係回路

これを見て分かって頂けるように、これらすべてにおいてSILに応じたtechnics and measuresを活用しなければならないのは一目瞭然です

7. まとめ

- 今回は階層的なソフトウェア開発についてお話ししました。
- 階層は、開発の細分化だけではなく、Generic Product, Generic Application Product, Application Specific Productという観点からも考えることができます。
- IEC 62279は、7章において、Generic Softwareの細分化を行っています。これは、ハザードを低減するために役立ちます。
- 8章において、Generic softwareと同じくらい重要なApplication dataの議論がされています。これも同じくハザードを低減します。
- GP/GA/SAの区分は、IEC 62425 Ed.2にあります。一度また説明する機会があれば実施します。