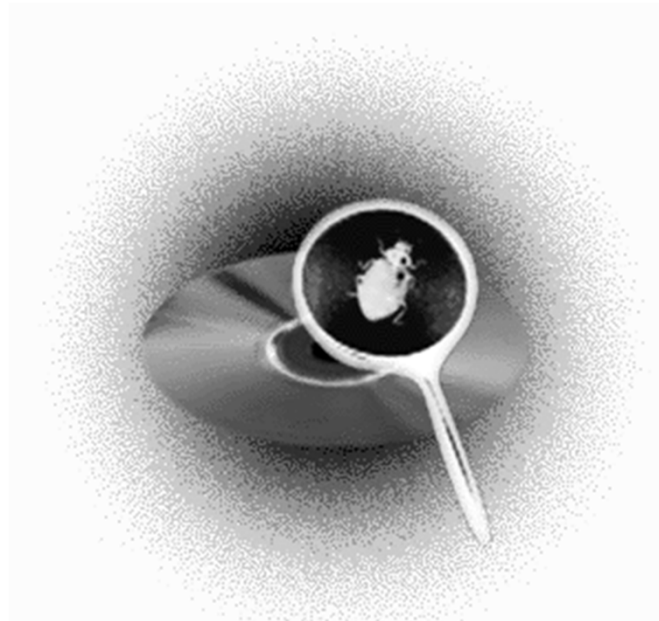


なぜソフトが組み込まれると 品質が悪化するのか？



JSQC 第114回シンポジウム(本部)

2007/7/3(火)

電気通信大学 電気通信学部 システム工学科

西 康晴

自己紹介

- 身分

- ソフトウェア工学の研究者
 - » 電気通信大学 電気通信学部 システム工学科
- 元・ソフトウェアのよろず品質コンサルタント



- 専門分野

- ソフトウェアテスト／プロジェクトマネジメント
／QA／ソフトウェア品質学／TQM全般／教育

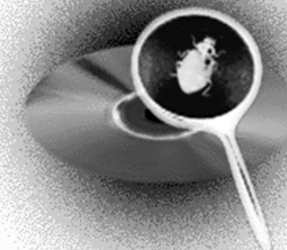
- 共訳書

- 実践ソフトウェア・エンジニアリング／日科技連出版
- 基本から学ぶソフトウェアテスト／日経BP
- ソフトウェアテスト293の鉄則／日経BP
- 基本から学ぶテストプロセス管理／日経BP



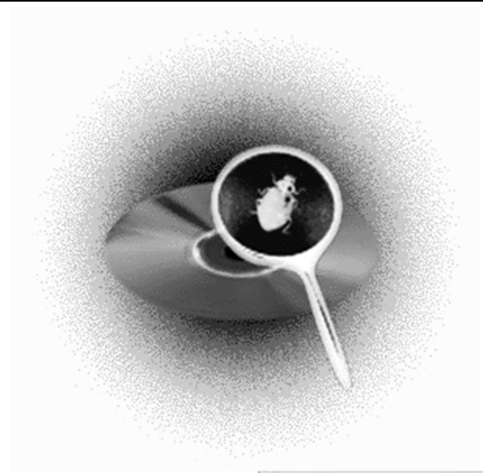
- もろもろ

- TEF: テスト技術者交流会 / NPO ASTER: テスト技術振興協会
- NPO SESSAME: 組込みソフトウェア管理者技術者育成研究会
- 日本品質管理学会 ソフトウェア部会
- 日本科学技術連盟 ソフトウェア品質管理研究委員会
- 情報処理学会 ソフトウェアエンジニアリング教育委員会
- 経済産業省 組込みソフトウェア開発力強化推進委員会



TEF: Testing Engineer's Forum

- ソフトウェアテスト技術者交流会
 - 1998年9月に活動開始
 - » 現在1300名弱の登録
 - » MLベースの議論と、たまの会合
 - <http://www.swtest.jp/forum.html>
 - お金は無いけど熱意はあるテスト技術者を無償で応援する集まり
 - “JaSST:ソフトウェアテストシンポジウム”も開催している
 - » 実行委員は手弁当／参加費は実費＋α
 - » 毎年4Qに東京で開催／今年はこのべ約1500名の参加者
 - » 昨年は大阪・札幌でも開催／会場は満席
 - 「基本から学ぶソフトウェアテスト」や「ソフトウェアテスト293の鉄則」の翻訳も手がける
 - » ほぼMLとWebをインフラとした珍しいオンライン翻訳チーム



SESSAME: 組み込みソフトの育成研究会

- 組み込みソフトウェア技術者管理者育成研究会

- Society for Embedded Software Skill Acquisition for Managers and Engineers
- 2000年12月に活動開始
 - » 200名強の会員／MLベースの議論と、月イチの会合
- <http://www.sesame.jp/>

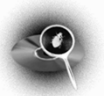


- 中級の技術者を10万人育てる

- PCソフトウェアのような「そこそこ品質」ではダメ
 - » 創造性型産業において米国に劣り、コスト競争型産業でアジアに負ける
 - » ハードウェアとの協調という点で日本に勝機があるはず
- 育成に必要なすべてを開発する
- オープンプロダクト／ベストエフォート
 - » 文献ポイント集、知識体系(用語集)、初級者向けテキスト、スキル標準など
 - » 7つのワークグループ: 組み込みMOT・演習・MISRA-C・ETSS・子供・高信頼性
- セミナーだけでなく、講師用セミナーも実施

講演の流れ

- 日本の組込みソフトウェアの品質は良い？悪い？
- ソフトウェアの品質事故の多発
- 組込みソフトウェアの受難
- ソフトウェア開発の特徴・
ソフトウェアの不具合の特徴
- ソフトウェア開発の問題点
- 組込みソフトウェアの品質向上に必要なこと



日本のソフトウェア開発は品質が良い？

Performance data

	India	Japan	US	Europe & other	Total
Number of projects	24	27	31	22	104
Median output ¹	209	469	270	436	374
Median defect rate ²	.263	.020	.400	.225	.150

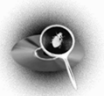
1. No. of new lines of code / (avg. no. of staff × no. of programmer-months).

2. No. of defects reported by customers in 12 months after implementation / total source LOC. We adjusted this ratio for projects with less than 12 months of data.

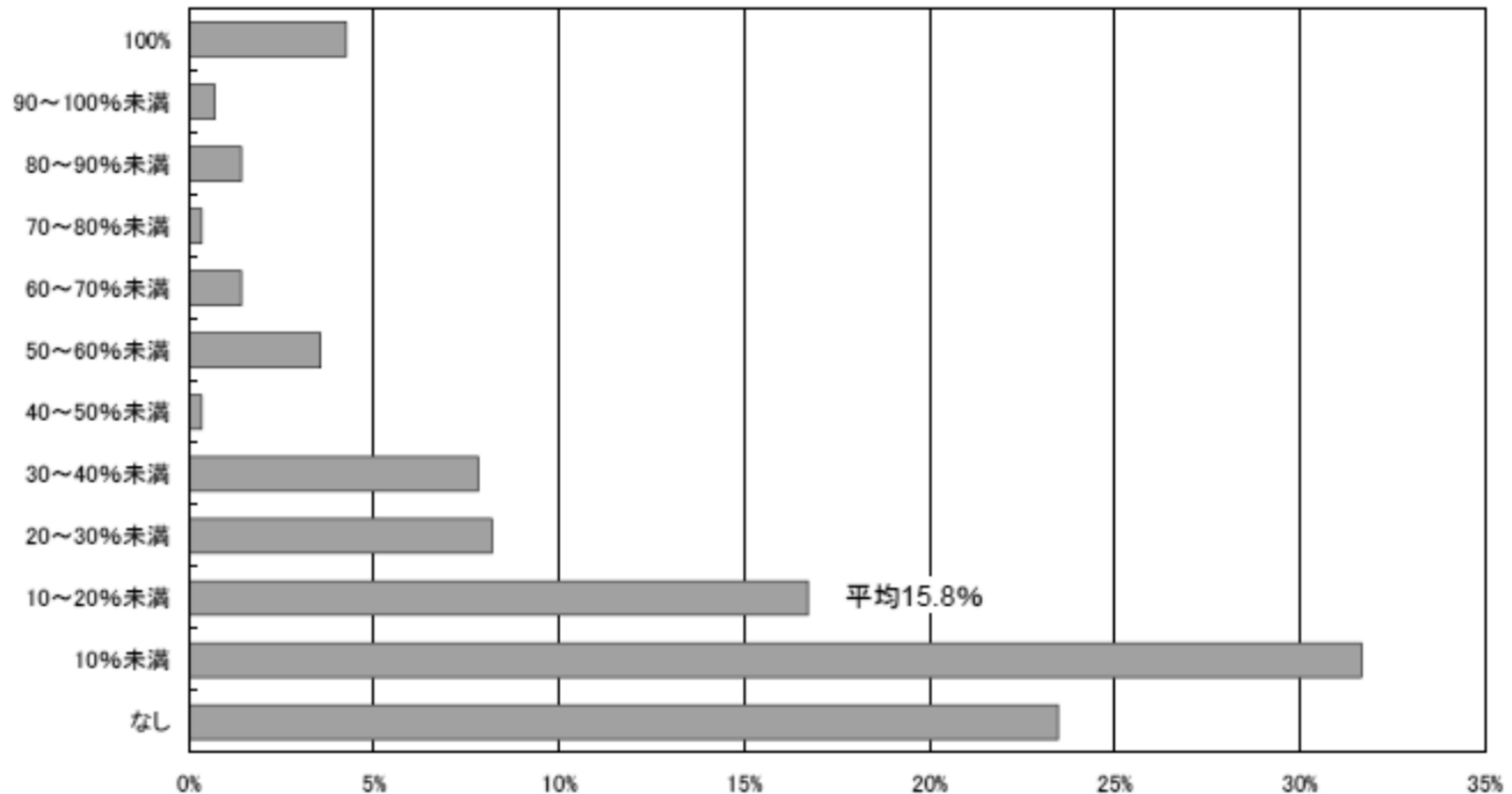
Michael Cusumano, et.al:
“Software Development Worldwide:
The State of the Practice”
IEEE Software, Vol.20.Issue 6, pp.28-34(2003)

日本の組み込みソフトウェアは品質が悪い？

- 組み込みソフトに起因するリコールの多発

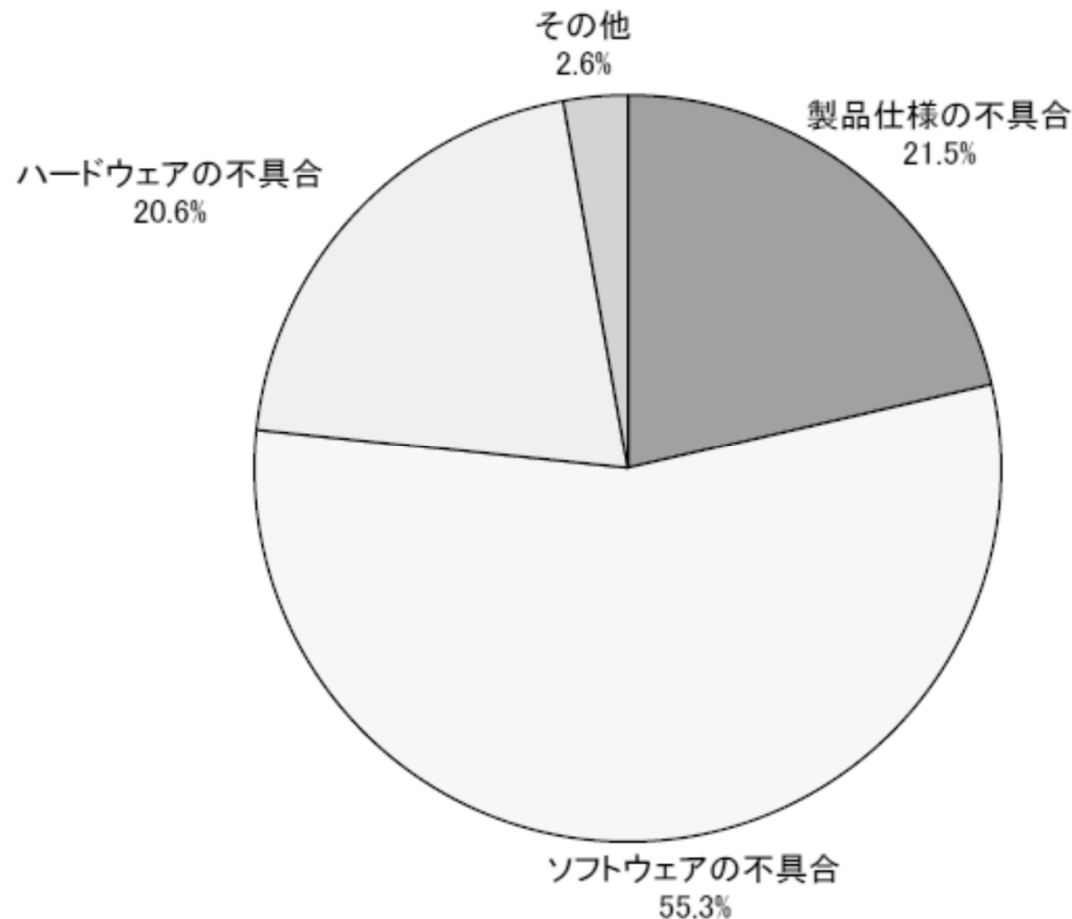


製品出荷後の設計問題の発生率



「2006年版組込みソフトウェア産業実態調査報告書：
経営者・事業責任者向け調査」より

製品出荷後の設計品質問題の主な原因

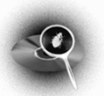


製品不具合の
原因の半分以上が
組込みソフト

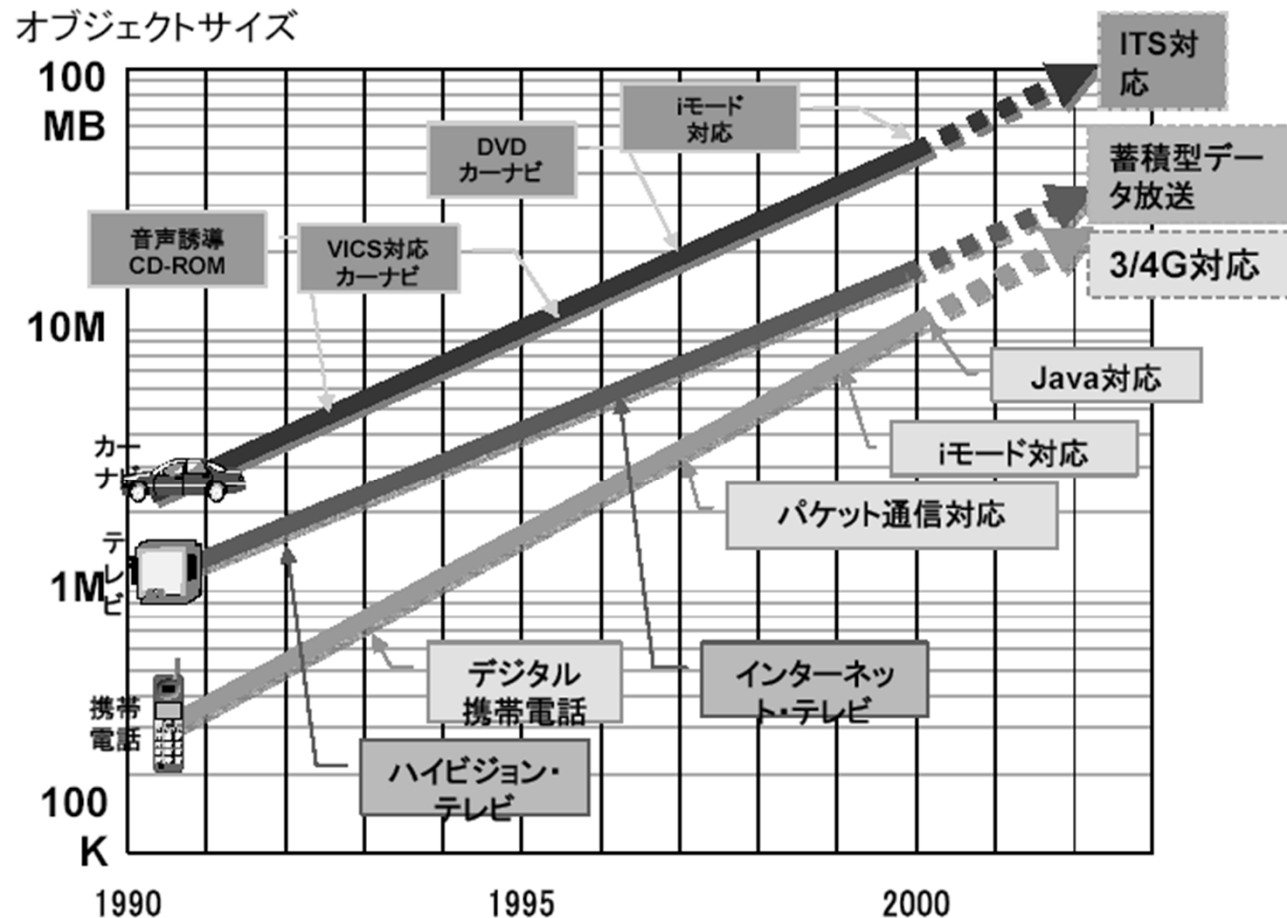
では、ハードの
設計品質は
ソフトに比べて
十分良いのか？

「2006年版組込みソフトウェア産業実態調査報告書：
経営者・事業責任者向け調査」より

© NISHI, Yasuharu



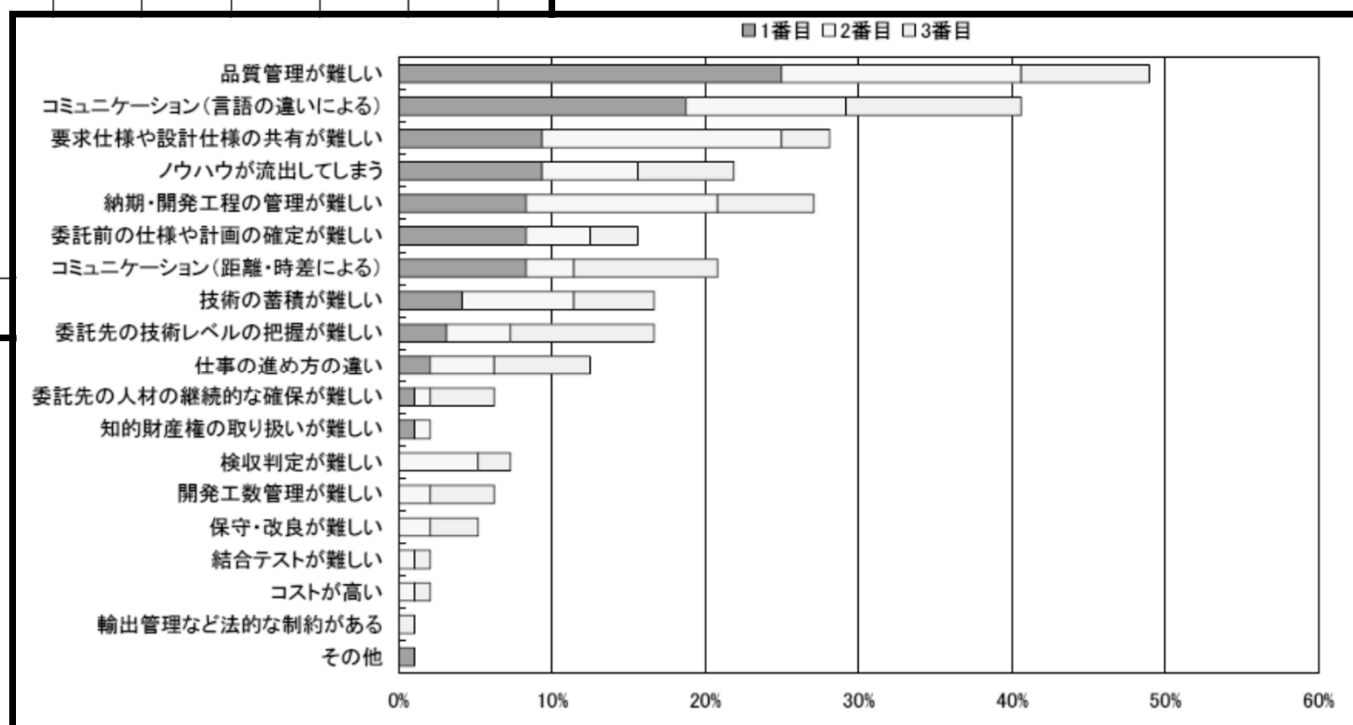
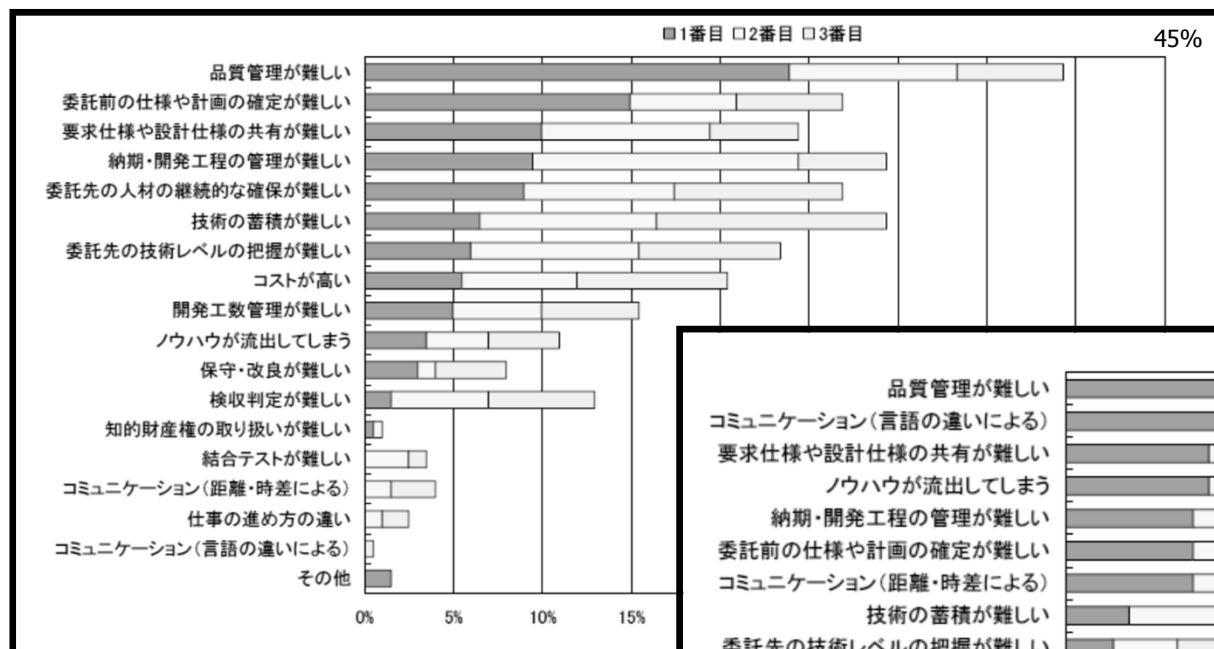
組込みソフトの変化：日経エレクトロニクスより



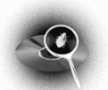
出展：日経エレクトロニクス 2000 9-11(no.778)をベースに追加、修正。携帯電話の増加率は変更してある

経済産業省 組込みソフトウェア開発力強化
推進委員会 組込みソフトウェア開発力強化
推進委員会の活動(2004年6月)より

外部委託の課題

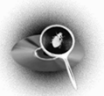


「2006年版組込みソフトウェア産業実態調査報告書:
経営者・事業責任者向け調査」より



講演の流れ

- 日本の組込みソフトウェアの品質は良い？悪い？
- ソフトウェアの品質事故の多発
- 組込みソフトウェアの受難
- ソフトウェア開発の特徴・
ソフトウェアの不具合の特徴
- ソフトウェア開発の問題点
- 組込みソフトウェアの品質向上に必要なこと



ソフトウェアの品質事故の多発

- エンタープライズ系システム

- メガバンクの情報システム統合に伴う品質事故は記憶に新しい
- 東京証券取引所の情報システムは度重なる品質事故
 - » 売買システムの処理増強に伴う品質事故により全銘柄で午前中の取引停止
 - » ジェイコム株誤発注事件では品質事故により400億円超の損害賠償請求訴訟
- 羽田空港の航空管制システムの品質事故は30万人以上が足止め
- 成功するプロジェクトはわずか26.7%(日経コンピュータ)



- 組み込み系システム

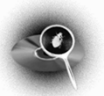
- 国内の携帯電話:65万台が回収・無償交換、131億円にのぼる損害
- ドイツの高級車のブレーキシステム:68万台がリコール
- 鉄道:ATCの誤作動により1ヶ月に50回以上の速度超過、19件の緊急停止措置
- ダム:ゲートが開き、総量約2万立米の貯水が流出

- 安全性が重視されるシステムが増えていくが...

- 自動車はX-by-wireになり、ITSにより道路状況を運転に反映させるようになる
- 軌道式公共交通機関は無人になっていくかもしれない
- 人間を殺傷する動力を持つ家庭用ロボットは、事前訓練を受けずに動かされる

講演の流れ

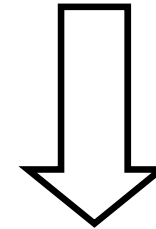
- 日本の組込みソフトウェアの品質は良い？悪い？
- ソフトウェアの品質事故の多発
- 組込みソフトウェアの受難
- ソフトウェア開発の特徴・
ソフトウェアの不具合の特徴
- ソフトウェア開発の問題点
- 組込みソフトウェアの品質向上に必要なこと



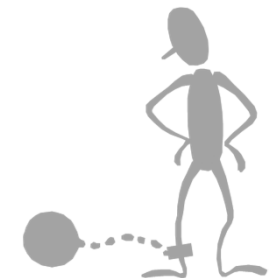
組込みソフトウェアの受難

- 機能要求の増大
 - － 制御の複雑化
 - － ソフトウェアによる機能の増加
 - » 携帯電話≡デジカメ≡コピー機？
- CPU/ECU 数の増加
 - － 比例してソフトウェアの数も増加
- ネットワークによる協調動作
 - － 自動車の重量の1割は通信ケーブル
 - － TCP/IPの搭載によるセキュリティの考慮
- プラットフォームが多様
 - － チップもOSも多種多様で当分統一されない
- 階層の増加
 - － ファームウェアからOS+ミドル+アプリへ
- PCアプリの発生
 - － PC上で組込み機器を管理／ソリューションの提供も
- 低品質な開発資産の流用
 - － ドキュメントの無い開発資産
 - － グローバル変数を多用した設計

ついこないだまで
リレーに毛の生えた
ものだったのに...



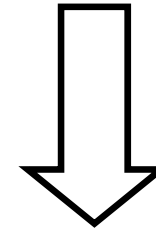
気が付くと
数百万行にのぼる



組込みソフトウェアの受難

- 開発期間の短縮
 - 携帯機器は半年サイクル
- 信頼性要求の劇的な向上
 - 財布や鍵になる携帯電話
- コストダウン圧力の増大
 - アウトソーシング単価下落やオフショア増大
 - オープンソースなどを自社で検証せずに利用
- しかし開発体制は旧来どおり
 - 体制が整う前に複雑化・大規模化しプロジェクトがデスマーチ化してしまう
 - ソフト部隊とハード部隊の連携に乏しい
- 規模増加によって全体像が見えない
 - ポテンヒットやモチベーション低下が発生
- 要員の離脱は珍しい話ではない
 - 過労による病気だけでなく、メンタルトラブルも
- 品質事故・納期遅延・赤字プロジェクトが多発している
 - 携帯、家電、OA、車載、ME、ダム...

ついこないだまで
たった1人で
作ってたのに...

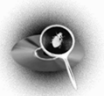


気が付くと
百人超のプロジェクトに



講演の流れ

- 日本の組込みソフトウェアの品質は良い？悪い？
- ソフトウェアの品質事故の多発
- 組込みソフトウェアの受難
- ソフトウェア開発の特徴・
ソフトウェアの不具合の特徴
- ソフトウェア開発の問題点
- 組込みソフトウェアの品質向上に必要なこと



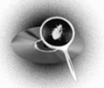
ソフトウェア開発の特徴

- ソフトウェアには「生産」プロセスがない
 - － 実装／プログラミングは、ハードにおける詳細設計プロセスである
 - » 完成したプログラムをCDやROMに焼くプロセスはソフトウェア開発として考慮しない
- ソフトウェアは劣化しない
 - － 物理的実体の性質変動を考慮する必要が(あまり)無い
 - » 組込みにおけるハードウェア、エンブラにおけるプラットフォームは考慮が必要
- ソフトウェアは物理化学法則に従わない
 - － 不具合の原因は論理性の欠如であり、物理化学法則に対応づかない
 - » 社会学や認知科学的法則、モチベーションが大きく関係する
 - » FMEAやFTAなどの故障解析的手法は難しい
- ハードウェアの信頼性測定の理論体系が適用できない
 - － 統計的な信頼度推定が難しい
 - » 材料的観点からの信頼度が存在しないので、ワイブル解析などの寿命推定は適用できない
 - » 利用頻度の推定は正確にできようはずもない
 - » 不具合が作り込まれる確率を推定するのは困難を極める
 - » 局所化できない不具合がありえるため、影響度の予測も難しい



ソフトウェア開発の特徴

- ソフトウェアには「検査」プロセスがない
 - 物理的実体のばらつきを公差の範囲内に収めるために、何かの値を測定しある基準値によって振り分けるような、単純作業は無い
 - ソフトウェアのテストは検査ではなくデザインレビューと本質的に同じである
- ソフトウェアのレビューやテストは難しい
 - ソフトウェアのレビューやテストは、本質的にハードのDRに対応する
 - » ハードのDRも難しいが
 - レビューやテストの工数は開発工数全体の4割を超える
 - レビューの指摘項目の設計の方法論はほとんど存在しない
 - テスト項目の設計の方法論はまだまだ貧弱である
 - テストを網羅的に実施しようとする、発散してしまう
 - » 機能、構造、データ、状態、環境などの組み合わせで乗数的に増えていく
 - 設計の不具合や弱点を探すことは、本質的に難しい
 - » 物理化学的手がかりが無い



ソフトウェア開発の特徴

- 大規模、多数・多種類、多人数、短納期
 - － 747の部品点数は600万点(?), カーナビのステップ数は数百万行超
 - － 乗用車に搭載されるECUは数年後に200を超えていると言われている
 - － 組込みの場合は、プラットフォームが極めて多様である
 - － ある携帯電話は200万ステップで1800人月であった
- 分割統治原則が必ずしも機能しない
 - － 空間的ないし物理的な分割が不可能である
 - » 例えばOSにメモリ保護などの機能はあるが、不具合の伝播を物理的に食い止めることができない
 - － 性能や省メモリのために、実装の詳細が隠蔽できない場合が少なくない
- 論理性が高い
 - － ロジックが入り組んでおり、条件組み合わせが複雑で、並行動作を行う
 - － バグは開発フェーズを経るごとに指数関数的に増加していく
- 自由度が高い
 - － 物理化学的な制約に従うとこう作るしかない、というわけではない
 - － こういうモノはこうやって作るべき、という定石が少なく、知らない技術者も多い



ソフトウェア開発の特徴

- 見積もりが難しい
 - － 生産が無いのでハードで言うと設計開発のみの見積りになる
 - － 成熟度が低い現場が多いため、リスク(製品品質リスク・プロジェクトリスク)の予測ができず、ばらつきが大きくなってしまう。
- 確立した開発方法論が無い
 - － 組込みでは構造化、エンブラではオブジェクト指向のはずだが...
 - － いわゆるSQC手法はほぼ全く適用できない
- 開発環境が変化しやすい
 - － プラットフォームが欧米製のため、彼らのビジネスに合わせて動作環境が変わってしまい、ノウハウを継続できる穏やかな変更にならない
 - － 洋モノをありがたがる文化のため、開発環境もすぐに変えてしまう
- 要求が曖昧で記述しにくく、かつ変化してしまう
 - － 暗黙の要求や「使ってみて始めて分かる」要求が少ない
 - － ビジネスのスピードが速くなり、ソフトは簡単に追加・変更できていると思われる
 - － 組込みの場合は、ハードウェアのしわよせを最終的に受けてしまう



ソフトウェア開発の特徴

- 再利用が難しい
 - － 再利用に最適な機能分割・構造分割・粒度設定が難しい
 - － 開発技術やプラットフォームがどんどん変化する
 - － もともと再利用を想定していない成果物をベースに再利用しようとする
 - － 全くドキュメントが残されていないのに再利用せざるを得ない場合がある
 - － もともと品質の低いコードをベースに再利用しようとする
- 品質が低く、性能が予測しにくい部品が多い
 - － 部品の受け入れテストや部品との組み合わせテスト、個々の部品向けのパッチコードが必要となる
 - － 外部から購入したソフトウェア部品の品質が低い場合がある
 - » 品質を保証することが難しいという側面もある
 - － 品質保証されていない「オープンソース」を使う場合も増えてきている
 - » オープンソース側も品質保証をする動きもある
 - － 成熟度が低い組織では動かすまで性能が分からないという背景から、そもそも複雑なので上流から性能を作り込めないと思っている場合もある



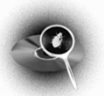
ソフトウェアの不具合の特徴

- (ほとんど) 全てがSystematic Failure(設計不具合)である
 - － したがって条件・状態が揃えば 100% 発生する
 - » システム全体で見ると、ハードの劣化 (宇宙線やノイズの影響を含む) とソフトの不具合が複合して起こることがある
 - － 開発者の認知的側面・組織的側面が原因である
 - » 認知的側面: 考え違い、狭い視野 etc.
 - » 組織的側面: 伝達不足、お見合いによるポテンヒット etc.
 - » ソフトの不具合は物理化学的法則の破れで整理できないため、故障モードが蓄積しにくい
- 複合的な原因
 - － プロダク的な原因とプロジェクト・プロセス的な原因が複合している
 - » 別にハードでも同じだと思うが...
- 影響が大きく広くなりがち
 - － そもそも不具合の影響の大きさを考慮して開発していないキライがある
 - － 影響を局所化するような設計が難しい
 - － 設計で局所化しても、暴走した場合の物理的な障壁は作りにくい



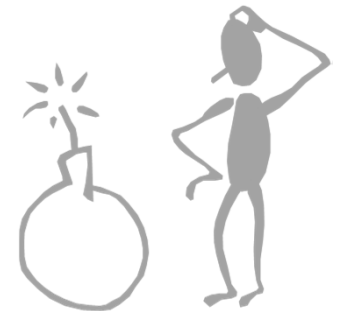
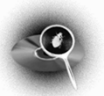
講演の流れ

- 日本の組込みソフトウェアの品質は良い？悪い？
- ソフトウェアの品質事故の多発
- 組込みソフトウェアの受難
- ソフトウェア開発の特徴・
ソフトウェアの不具合の特徴
- ソフトウェア開発の問題点
- 組込みソフトウェアの品質向上に必要なこと



ソフトウェア開発の問題点

- 品質・信頼性・安全性の向上がビジネスの成功（コストダウン・納期低減）に結びつくと思われていない
 - － 品質を上げるとコストは（中長期的に）下がるという法則を知らない
 - » 品質向上はテストや監査を増やしたりドキュメントを増やす活動だと誤解されやすい
 - » PM概念の浸透により、目の前のプロジェクトのQCDだけが目標だと誤解されやすい
 - » エンプラ系は量産が無いため、開発時の品質向上コストを売り上げ向上に転嫁できない
 - － 90年代にGood Enough Quality（そこそこ品質）パラダイムが浸透した
 - » 品質ではなくスピード「だけ」が利益の源泉だと思われるようになった
 - » ハードは欠陥ゼロにできるがソフトは不具合ゼロにできないという甘えがある
 - » 市場もそれを許容しているフシがある

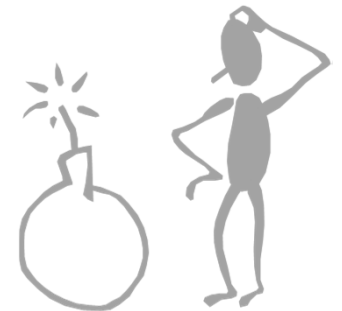
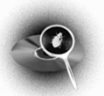


ソフトウェア開発の問題点

- 品質・信頼性・安全性の向上がビジネスの成功（コストダウン・納期低減）に結びつくと思われていない
 - － 多層下請け構造と工数精算システムが諸悪の根源
 - » 能力評価が難しいので、品質意識に乏しい
 - » 納期が低減すると残業が減って利益が低減する
 - » 納期が低減すると契約が終了するので稼働損リスクが増加してしまう
 - － （純粋な）ソフトウェアにはPL法が適用されない
 - » 信頼性や安全性は関係ないと思っている現場もある
- すなわち、品質・信頼性・安全性の向上により組織能力を鍛え、持続的にコストを下げ続けていくという発想に乏しい
 - － ユーザ圧力に負け、目先の安易なコスト低減策に飛びつくというデススパイラルに陥る
 - » 単体テストやレビュー、ドキュメント作成のコストを省く
 - » 十分な準備もせずにオフショア開発を始める
 - － 地道な改善をしている現場もあるが、そうでない現場も多い

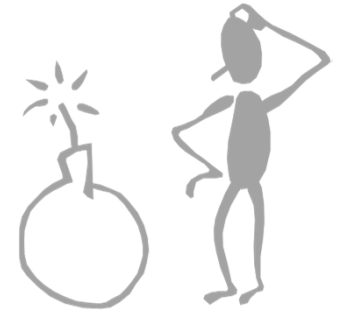
ソフトウェア開発の問題点

- 人間的側面でのアプローチをあまり取らず（特に研究者）、
属人的なノウハウとして済ませてしまうことが多い
 - － 認知的側面にあまり着目しない
 - ≫ 設計不具合
 - － 組織的側面
 - ≫ プロジェクトマネジメント
 - ≫ コミュニケーション
 - ≫ 多重下請け構造
 - ≫ 土農工商メカエレソフト
 - ≫ オフショア
 - － 心理的側面
 - ≫ ソフトウェア開発は3Kの職場という認識が定着している
 - ≫ モチベーション／うつ病／自殺
 - ≫ ソフトウェア開発者の労働安全はほとんど考慮されていない
- 他業界にあまり学ばない
 - － ハードウェアの設計論や品質管理・生産管理と接点があり無い
 - ≫ 表面的な真似をしている事例はそれなりにある



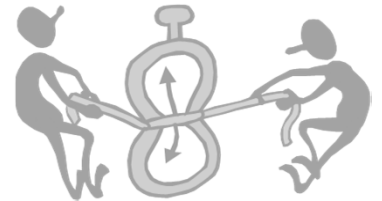
ソフトウェア開発の問題点

- ソフトウェア工学の教育が貧弱である
 - － 能力の評価体系が確立していない
 - » アサイン時・下請け選定時に適切な技術が選べない
 - » そのため頭数(工数)で見積もりをせねばならず、付加価値を価格に転嫁しにくい
 - － ソフトウェア工学を体系的に教えられた学生はほとんどおらず、社内できちんと研修を受けたエンジニアも多くない
 - » 特に下請けになればなるほど教育が足りない
 - » 基本的に工賃精算であり、能力評価がされないため、教育すればするほど稼働率が下がり損をするビジネスモデルになっている
 - » 大企業ではそれなりに体系的な教育をしているはずだが、彼らは工数管理に追われており、実際に開発をするのは下請けである
 - » 企業では OJT と称して体系的な教育もせずプロジェクトに投入し、(間違っているかもしれない)慣れたやり方を一子相伝的に伝えることが多い
 - － 大学には教えられる教員が少なく、コースも無い
 - » ソフトウェア工学科は無く、情報系学科ではほとんどソフトウェア工学を扱わない
 - » 教えている学科でも、テストやレビュー、品質、プロセス、マネジメントは教えない
 - » 初めからちゃんと作れば良いモノができていると思っている学者が多い
 - » プログラミング演習をソフトウェア工学と称する学科すらある
 - » 最近やっと「先導的」「実践的」コースが設置されてきた



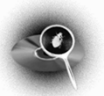
組込みソフトウェアの品質が低い組織の特徴

- 企画・メカ・エレキの変更や納期のしわ寄せがソフトに来てしまう
 - － ギリギリまでマーケットを反映できたり不具合を回避できるのは良いことなのだが
 - － 最近ではエレキもギリギリまで変更できるようになりつつある
- メカ・エレキ・ソフトの壁が高く、気を遣い合う開発になっていない
 - － 「最初から言ってくればよかったのに」ということも多い
- ソフトを理解していない／ソフトのスキルが低い
 - － 製品マネジャーがメカ出身で、ソフトのことがほとんど分からない
 - － 品質保証部がハード由来で、印鑑監査型保証しかしていない
 - － 製品メーカーなのにソフトの仕様書しか書けず、実は外注任せ
- 改善できない／していない／する仕組みになっていない
 - － ソフトで価値を生み出すという発想や雰囲気無く、コストダウンの方策ばかりが現場に押しつけられる
 - － 急激なソフト量の増加に現場が追いつかず、改善する間もなく、みな疲弊している
 - － 協力会社も含めた教育や巻き込んだ改善活動をしていない
 - － ソフト開発部隊の子会社化はどんな影響を及ぼすのか？



講演の流れ

- 日本の組込みソフトウェアの品質は良い？悪い？
- ソフトウェアの品質事故の多発
- 組込みソフトウェアの受難
- ソフトウェア開発の特徴・
ソフトウェアの不具合の特徴
- ソフトウェア開発の問題点
- 組込みソフトウェアの品質向上に必要なこと



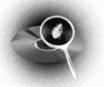
開発全体として必要なこと

- ソフトの開発の難しさを理解し、頼被りしない
 - － ソフトのことを(それなりに)理解しようとし、聞く耳を持つ
 - － 現場や製品マネジャー、QAだけでなく、事業責任者や経営陣の責任も大きい
 - － 「ハードウェア出身のマネージャに分かっておいてほしい7つのこと」を読む
» <http://www.sesame.jp/>
- メカやエレキの設計との共通点・相違点を明らかにし、それぞれのベストプラクティスを他分野に展開する
 - － 一緒に考えることが必要であり重要
 - － 進んでいるメカやエレキの設計品質向上活動を加速させる
- メカ・エレキ・ソフトの「すり合わせ」によって「全体最適」を実現できる組織文化にする
 - － 風通しのよい組織文化を醸成し、他の部隊に気を遣える協調型の開発プロセスを構築する
 - － 製品全体の開発のボトルネックは何かをしっかりと考え直す
 - － メカ・エレキ・ソフトの違いを理解した上で協調できるQMSを構築し、プロセス改善の「ころ」を伝え合い、各分野の改善プラクティスに落とし込む

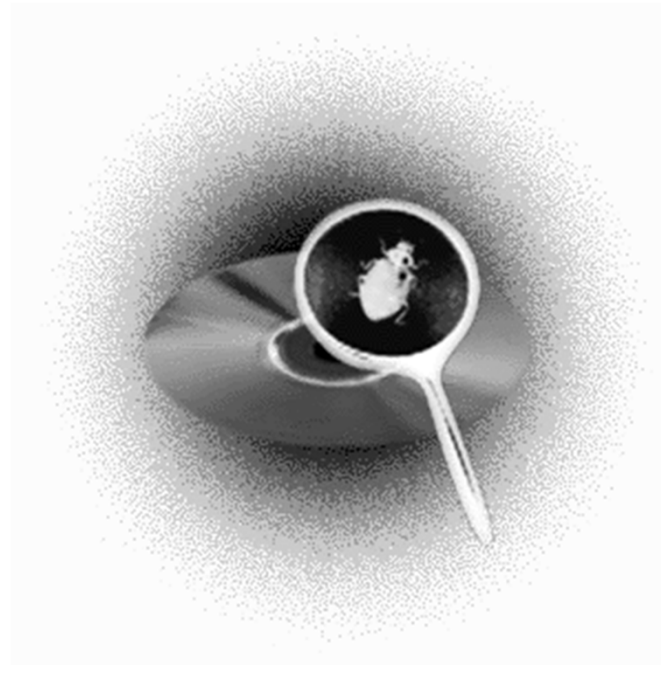


ソフトウェア部隊に必要なこと

- 甘えない
 - － ソフトウェアはバグがあっても仕方がない、とは思わない
 - » ハードだって設計不具合は出るが、あっても仕方がない、とは口が裂けても言わない
 - － 品質とコストをトレードオフと捉えず、両立するように知恵を絞る
- 自分のことばかり考えない
 - － 次工程・他工程はお客様だと考える
 - » 被害者意識を持たない
 - － メカやエレキの特性をきちんと理解する
 - － 小難しいソフトの専門用語を振り回さない
- しっかり考えて努力する
 - － 銀の弾丸を探さない
 - » ソフトの品質確保の全体像を理解し、実践する
 - － 足元を見る
 - » 開発方法論やプロセス改善モデルを鵜呑みにしない
 - » 不具合やプロジェクトリスクの分析をしっかり行い、水平展開をし、再発防止から未然防止につなげ、しっかり教育する
 - － 組織風土を高める
 - » 全員参加、ねばちっこさ、自発性、斟酌する力(ソフトウェア品質シンポジウム2007)



ご静聴ありがとうございました



電気通信大学 電気通信学部 システム工学科
西 康晴
nishi@se.uec.ac.jp