

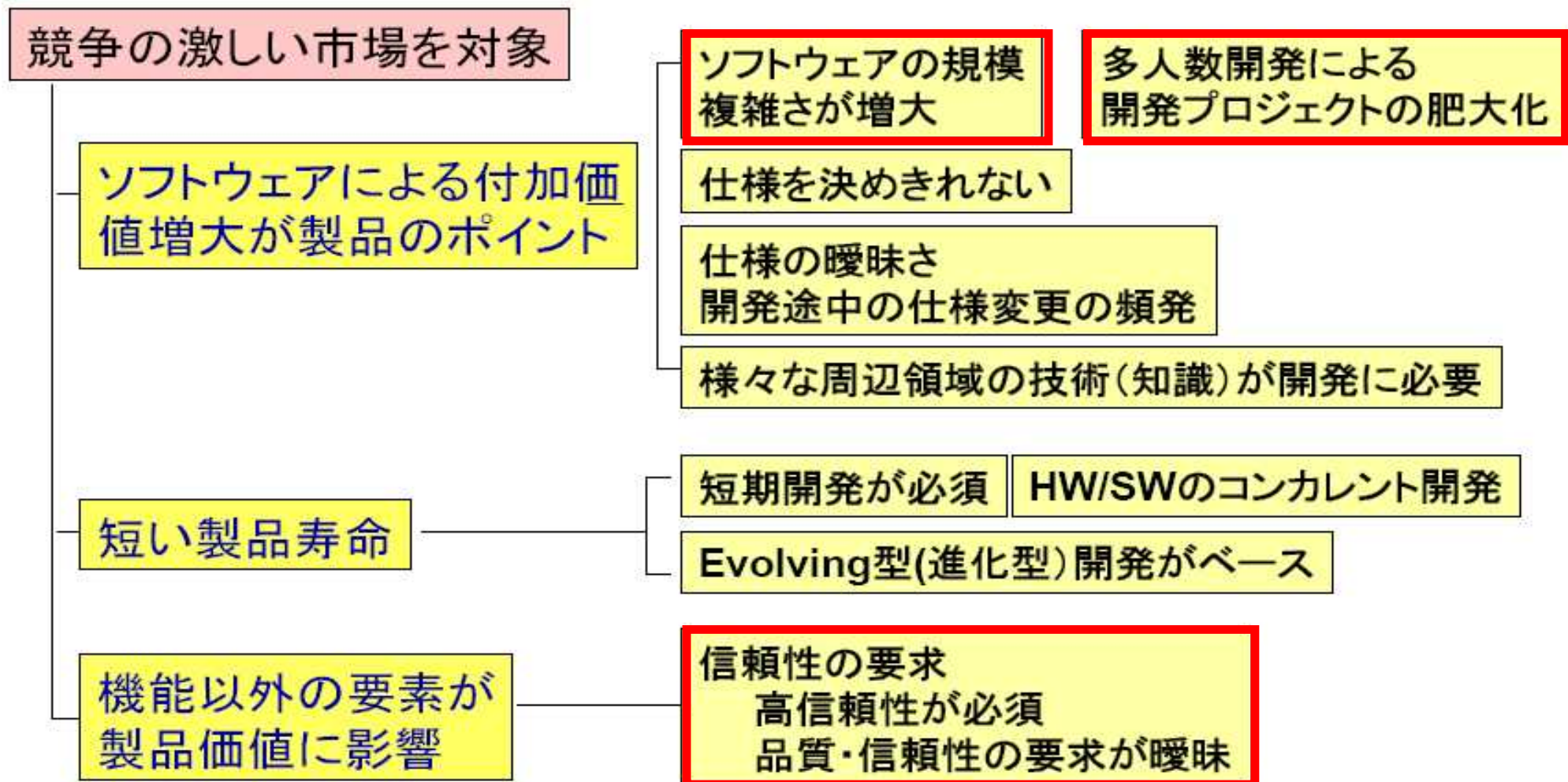
# 高品質実装のためのコーディング作法

大野 克巳

独立行政法人情報処理推進機構  
ソフトウェア・エンジニアリング・センター (SEC)  
株式会社 トヨタコミュニケーションシステム

平成17年2月22日(火)  
組込みソフトウェアフォーラムin名古屋

# 組み込み開発を取り巻く環境

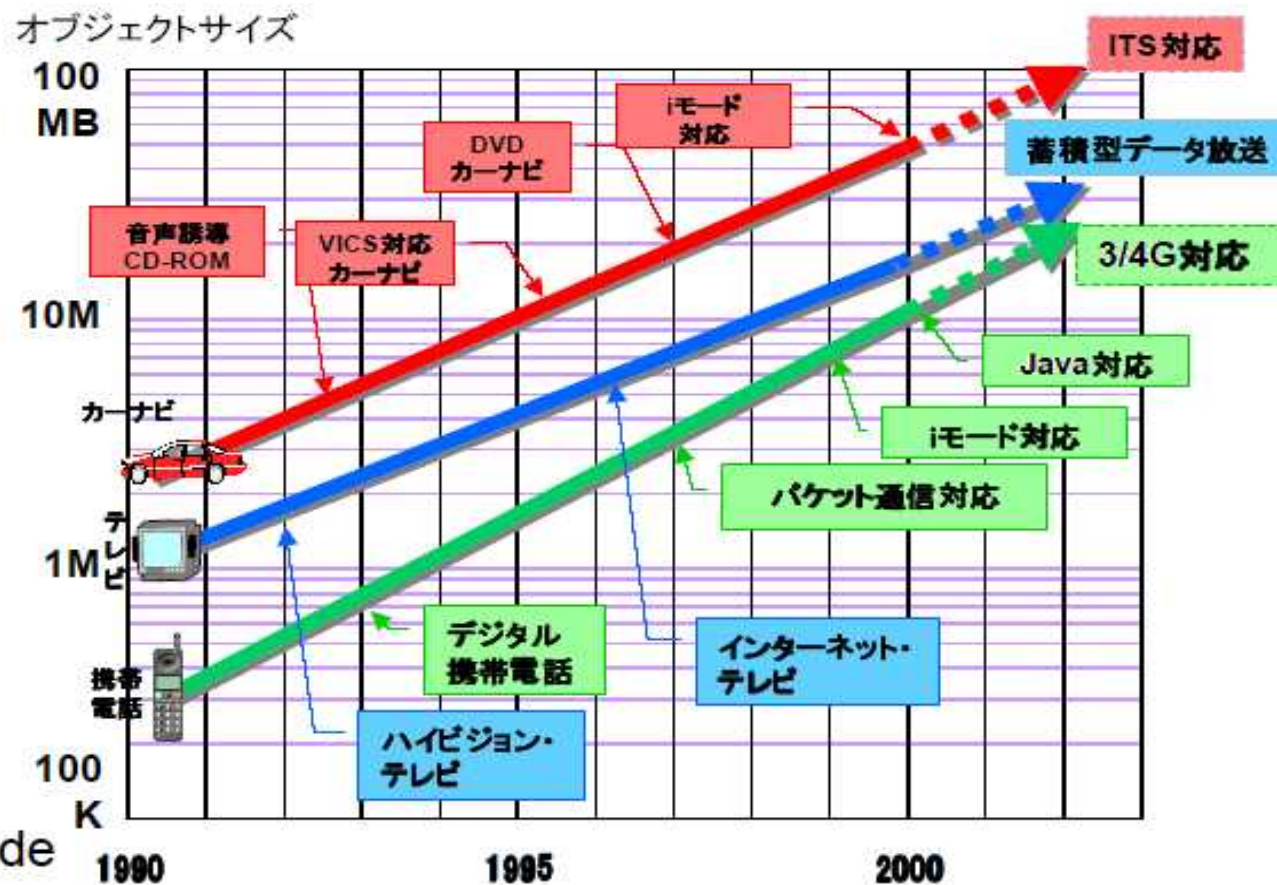


# 組み込み開発を取り巻く環境

## 規模例 (SLOC\*)

- 通信機能搭載型カーナビ
  - 300万行
- 薄型テレビ
  - 60万行
- HDD内蔵DVDレコーダ
  - 100万行

\* SLOC: Source Lines of Code



出典: 日経エレクトロニクス 2000 9-11(no.778)をベースに追加、修正。

# 2004年(昨年)の組み込みソフトウェア不具合の事例 1

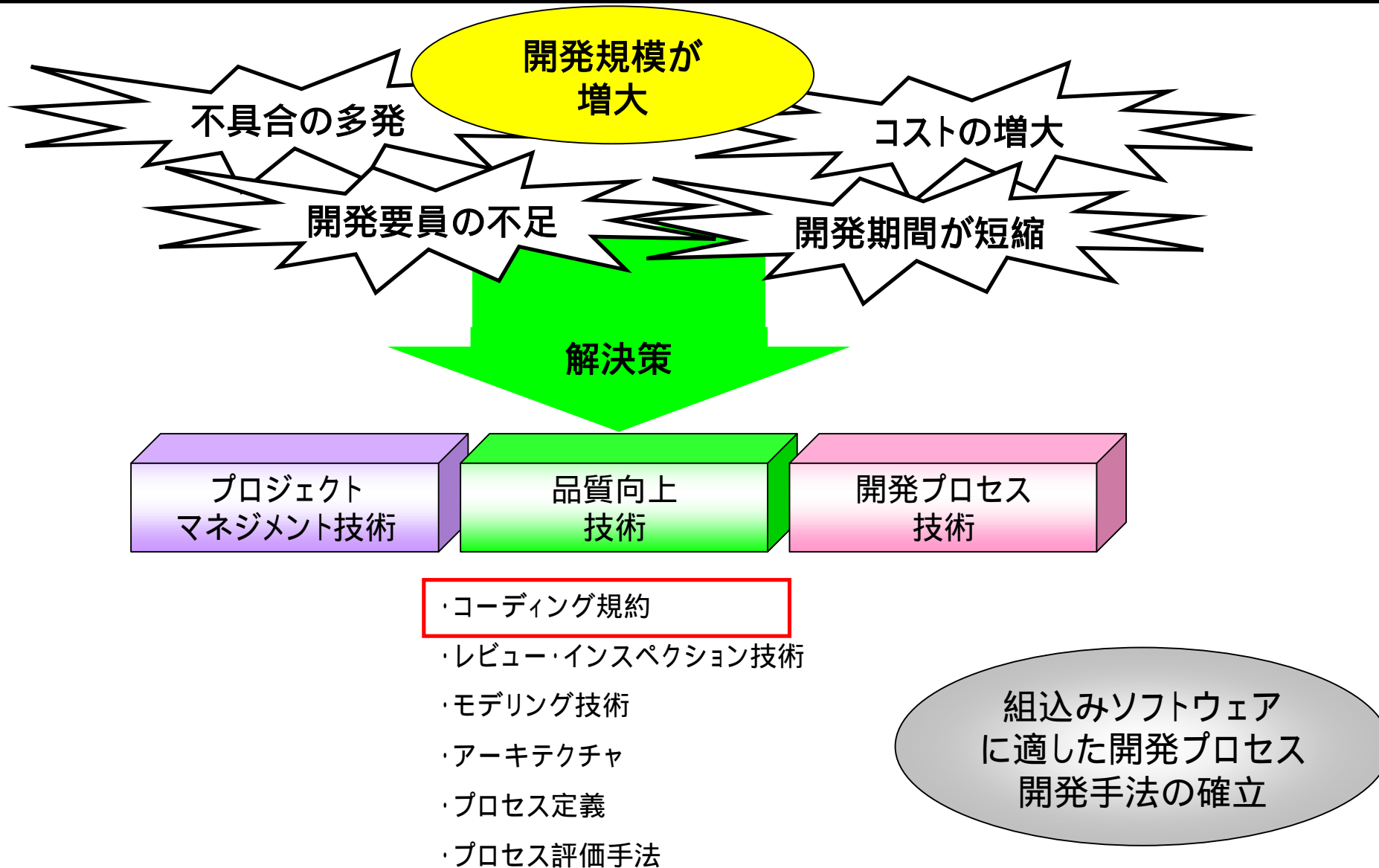
## 携帯電話やAV機器等の故障を引き起こした事例

製品名	不具合の内容
HDDビデオレコーダ	250Gバイト・ハードディスクの112Gバイト以降に録画/再生できない
DVDレコーダ	DVDレコーダで電源がオン状態の時に録画予約メールを受信できない
携帯電話	メールの題名入力後すばやく本文入力に移ろうとすると操作できなくなる
デジタルカメラ	連続撮影中や、オートパワーオフ状態でレンズを着脱後、デジタルカメラが操作不能になる
携帯電話	電話を発信するとほぼ同時に着信があった場合、電話帳の内容が消失する

## 自動車のリコールにつながった事例

製品名	不具合の内容
国産車	DPF(ディーゼル微粒子除去装置)が異常加熱し、規制量を超える粒子を排出量するおそれがある。
輸入車	電源制御装置の不具合により、リモートコントロールキーでドアロックを作動させるとホーンが作動し、止まらなくなることがある。
輸入車	エアバッグコントロールユニットのソフトウェアが不適切なため、着座センサーの情報を感知できない場合があり、事故時に、助手席エアバッグが作動しないおそれがある。

日経コンピュータ 2004.12.27 NO.616 「組み込みソフトの巨大化に立ち向かう」より抜粋



# 高品質な開発とは何か？

---

- 誰が作っても同じものが製造できる
- 試験が可能なものを設計できる
- 動作原理がわかるものを設計できる
- 起こりうるエラーを緩和できる

二上貴夫 「ユービキタス時代の組込みソフトウェアの高信頼性について」 第23回ソフトウェア生産における品質管理シンポジウム、日科技連、2004年

結果的に 「バグがなく、保守性や再利用性が高いソフトウェア」

## コード(プログラム)の記述スタイルのためのガイドライン！

### コーディング規約の効果

#### 可読性の向上

記述スタイルに従ってコードを作成することにより、作成者の癖を排除

#### バグの混入を防止

バグの作り込みにくさをコントロールするための施策、命名規則や、リスクの高い言語仕様を避けるためのルールを用意

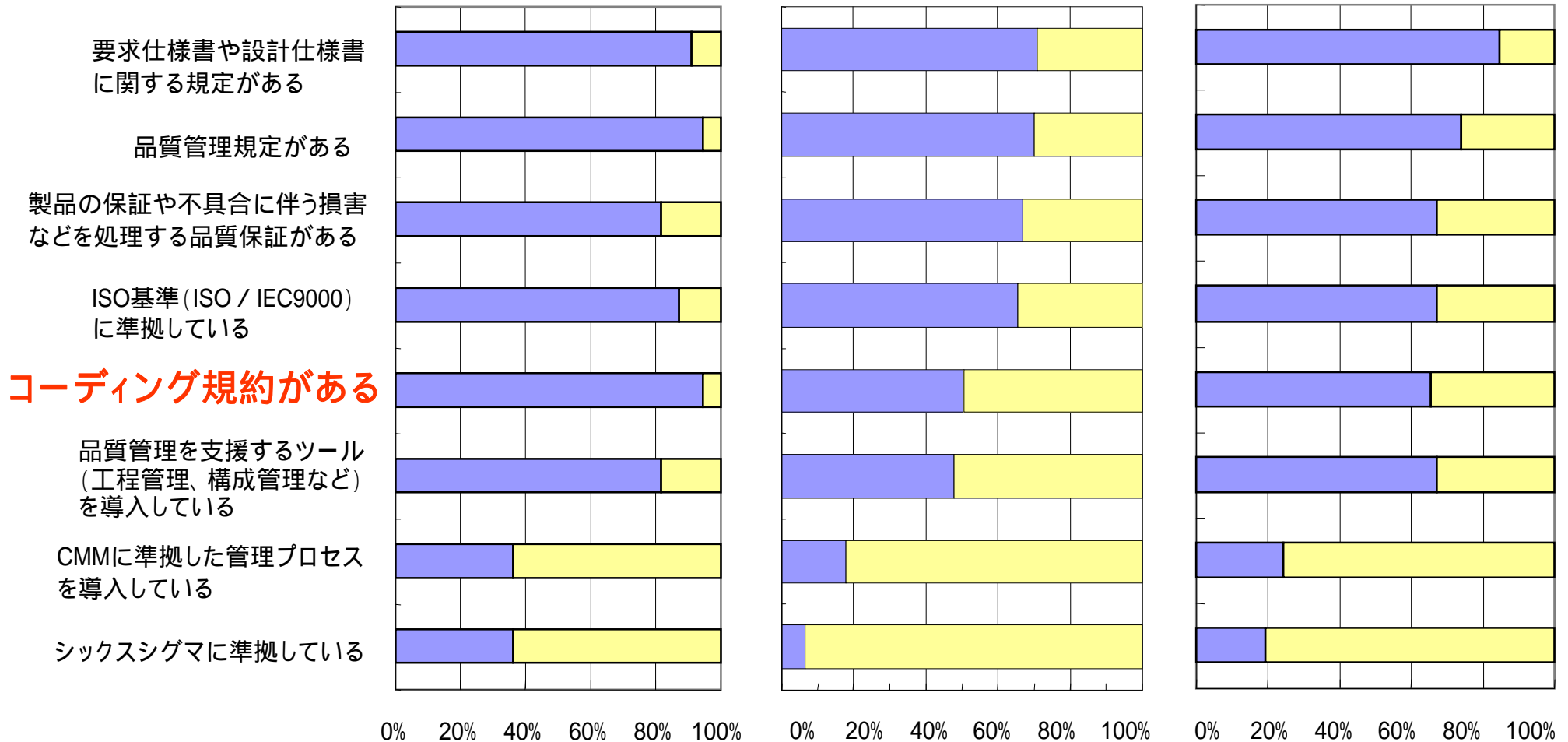
# コーディング規約の有無

欧州

日本

米国

■ はい ■ いいえ



経済産業省 2004年版組込みソフトウェア産業実態調査

たとえコーディング規約が存在していても

## 現状のコーディング規約の問題点

- 運用・仕組みの問題

チェックの仕組み 形骸化

- 規約の内容の問題

漏れ抜け 多すぎる 少なすぎる

- 人・スキルの問題

理解度 こだわりや抵抗

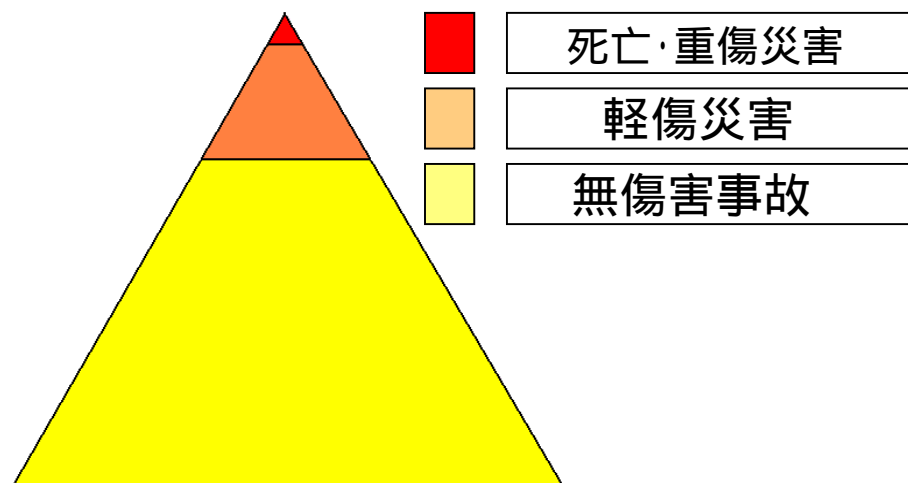
- ツールの問題

ノイズ 解釈の違い

課題は多い

# ハインリッヒの法則

「1件の重大な事故の陰に29件の小さい事故があり、さらにその奥に300件の小さな異常が隠れている」



医療現場での1:29:300の法則  
 300件の小さなミス(カルテの記載ミスや申し送りの伝達漏れ)  
 29件の軽微な医療ミス(注射のmg違いや薬剤の過剰投与)  
 1件の重大な医療事故

ハインリッヒの法則は  
 コーディング規約のルール違  
 反とバグの数の関係にも当て  
 はまるのではないか？

1:29:300の法則は失敗発生率としてあらゆるエラーと事故の直結を断ち切る工夫することの重要性・有効性を示唆している。

ハインリッヒの法則とは、アメリカの技師ハインリッヒ(H. W. Heinrich)が労働災害の事故を統計学上に調べ、計算した結果出した法則である。

# 何故、ルール(規約)違反は起きるか？

身近な事例:

草サッカーチームとルール(規約)

コーディングに置き換えると

初心者にはオフサイドは言葉だけでは教えられない

ルール(規約)を正しく理解していないが故に犯してしまう違反

初心者にはポイントは言葉だけでは教えられない

中級者は「スーパースターに学ぶサッカーテクニック集」は読んでも、「競技規則集」は……

意外と曖昧な「ルール(規約)の細かい理解」

中級者は「テクニック集」は読んでも、「言語の規格書」は……

「サッカーとフットサル」

似て非なるもの、混同・誤解するルール(規約)

「C と C++ あるいは C#」

審判のレベルもいろいろ(寛大?・誤解釈?・ローカル規約?)

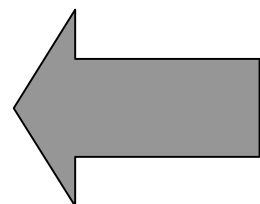
体験により擦り込まれるルール(規約)の記憶(普遍なものであれば良いが)

コンパイラやツールのレベルもいろいろ(寛大?・誤解釈?・処理系の違い?)

# ルールは覚えているか？

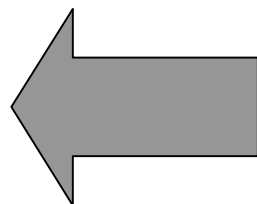
---

身近な事例：理解しているはずの道路標識でも



この標識は何？

身近な事例：いつの間にか見かけるようになったマーク



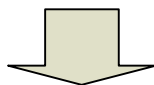
このマークは何？（見た事ありますか？）

# コーディング作法ガイド作成の背景

背景1 コーディングミスの多いプログラムには他の重要なミスも多い  
防止には「コーディング規約」が有効

背景2 コーディング規約に関する問題点

- コーディング規約そのものがない(有用な規約は,過去の経験の積み重ね)
- コーディング規約は存在しても,規則が多すぎる,個々の規則の必要性が理解されないなど,運用に苦労していることも多い



コーディング規約の効果的な運用には,  
意義のわかる,整理されたリファレンスが必要

「コーディング作法ガイド」

- 規則の意義を作法(品質向上のために守るべき具体的な実装の考え方)で明確化
- 現場技術者が受け入れられる規約運用を促進

有用な規約は,PJによって異なる。  
そこで,規約作成のための作法ガイドという形で提供。

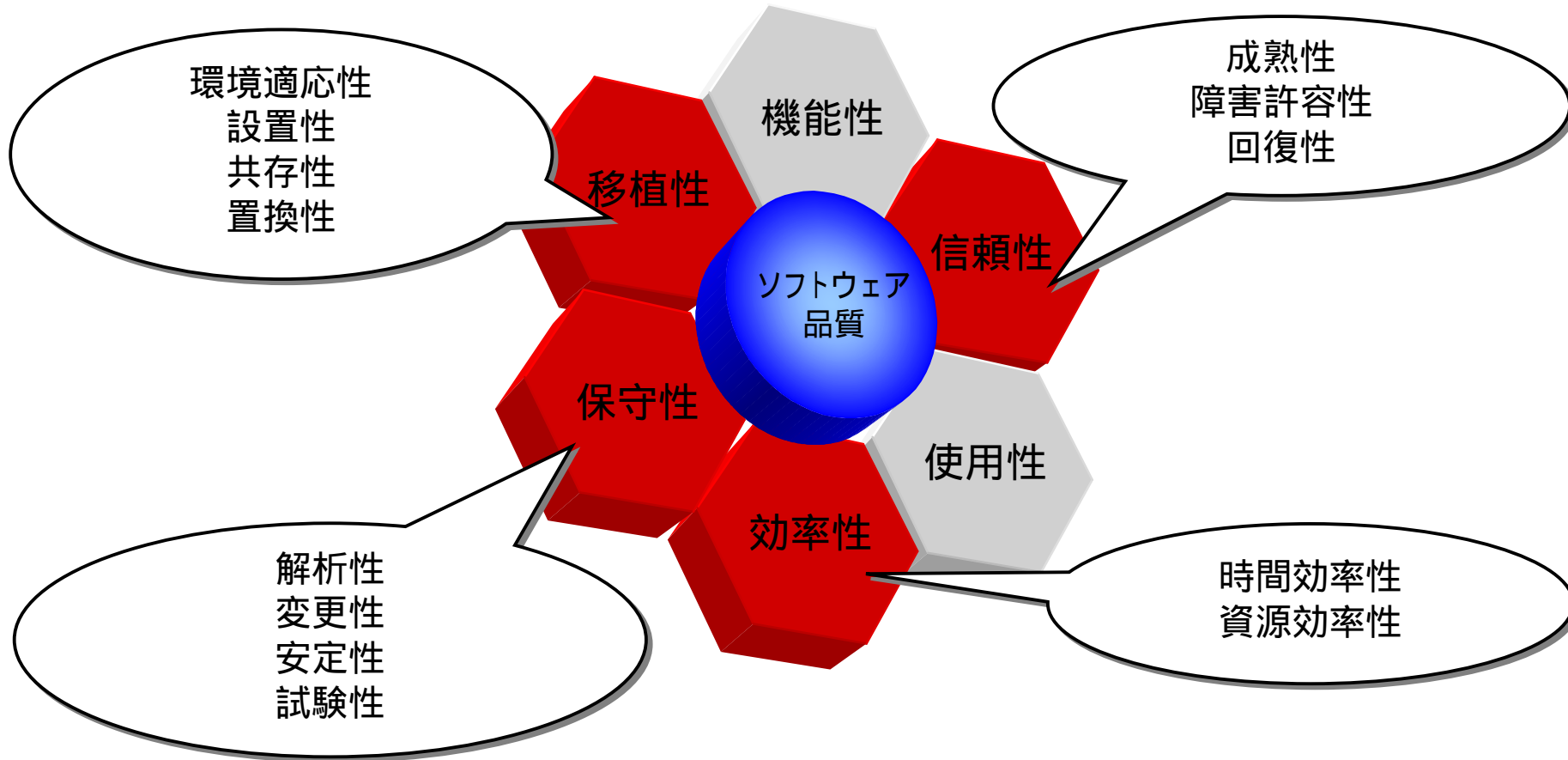
コーディング  
作法ガイド



各プロジェクトの  
コーディング規約

ガイドを参考にして作成

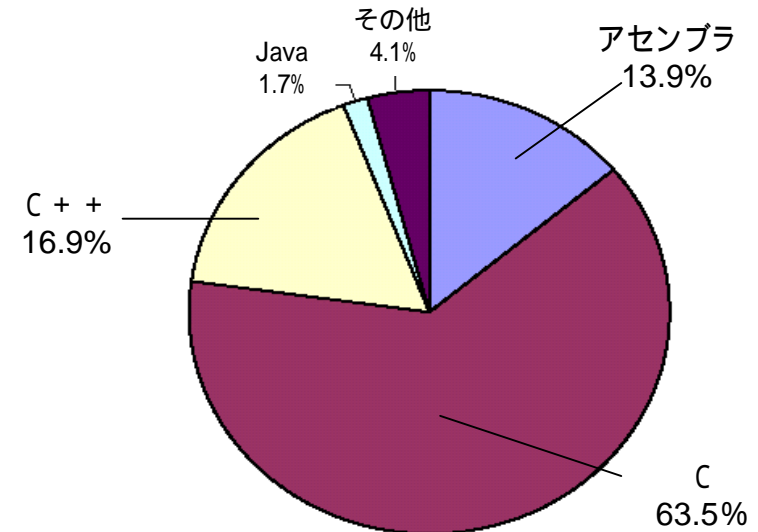
# コーディングの観点からの品質特性 (ISO/IEC-9126)



信頼性・移植性・保守性・効率性の4つの品質特性に着目

# C言語

組み込みソフトウェアで使用しているプログラミング言語



経済産業省 2004年版組み込みソフトウェア産業実態調査

## C言語の危険性

C言語規格書(JIS X3010 - 1933)の巻末 付録「付属書G 可搬書」より

- ・静的記憶域の初期化の方法および時期は未規定
- ・関数指示子および関数呼び出しの実引数が評価される順序は未規定
- ・ビット・フィールドの型がint・signed int または unsigned int いずれでもない場合は未定義
- ・関数が値を返していないのに、呼び出し元で関数呼び出しの値を使用している場合は未定義
- ・switch文におけるcase値の最大数は処理系が定義する

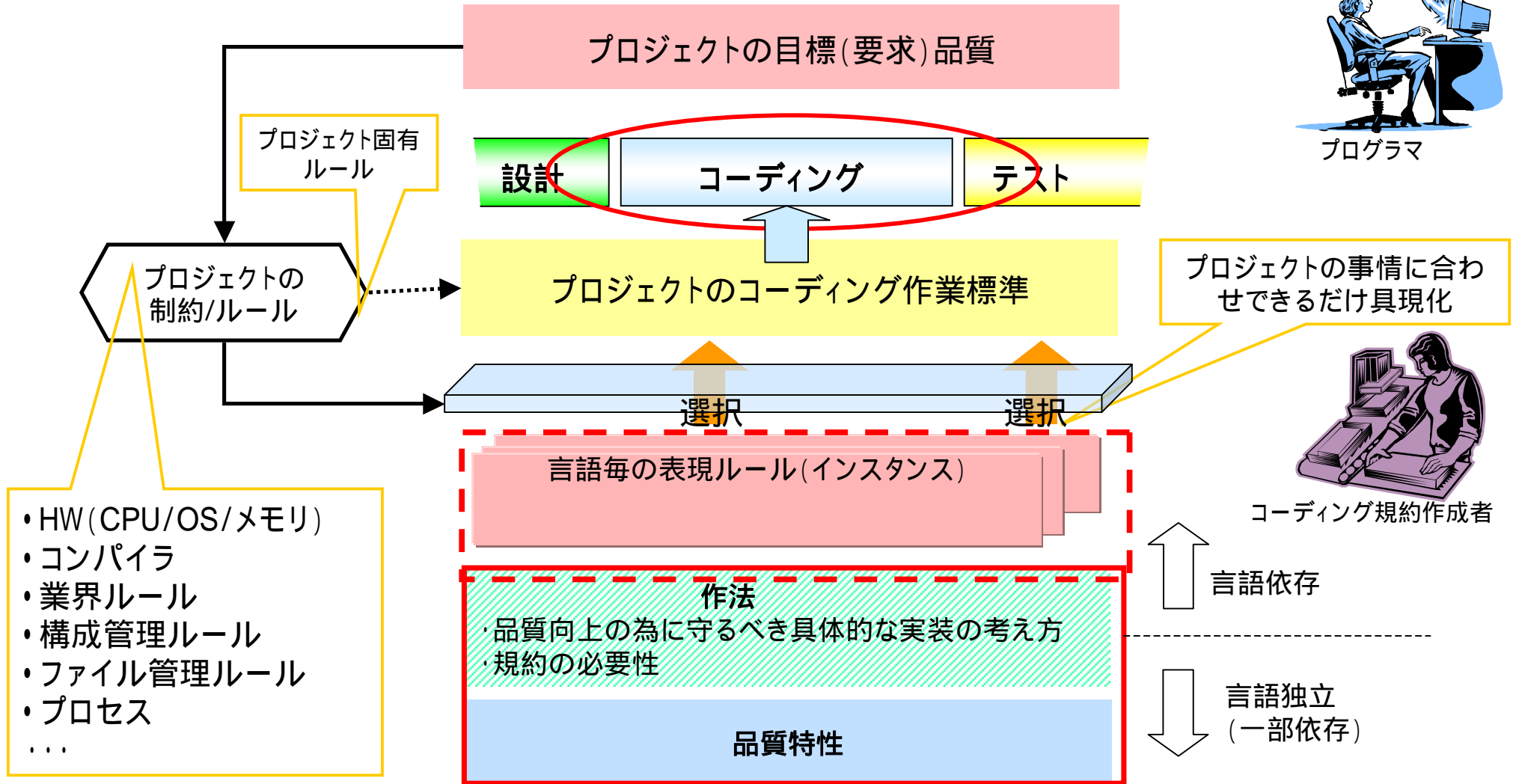
などなど

参考 DesignWaveMagazine 2004年9月号

# コーディング作法とは



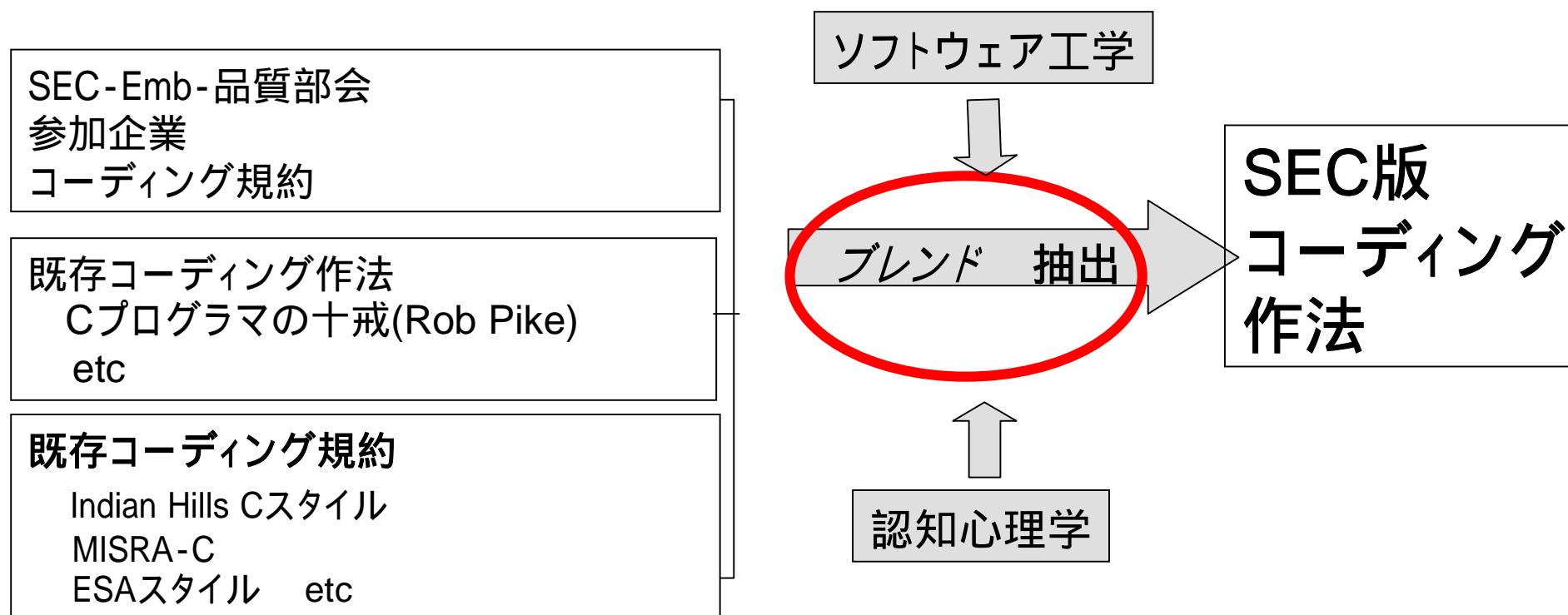
プログラマ



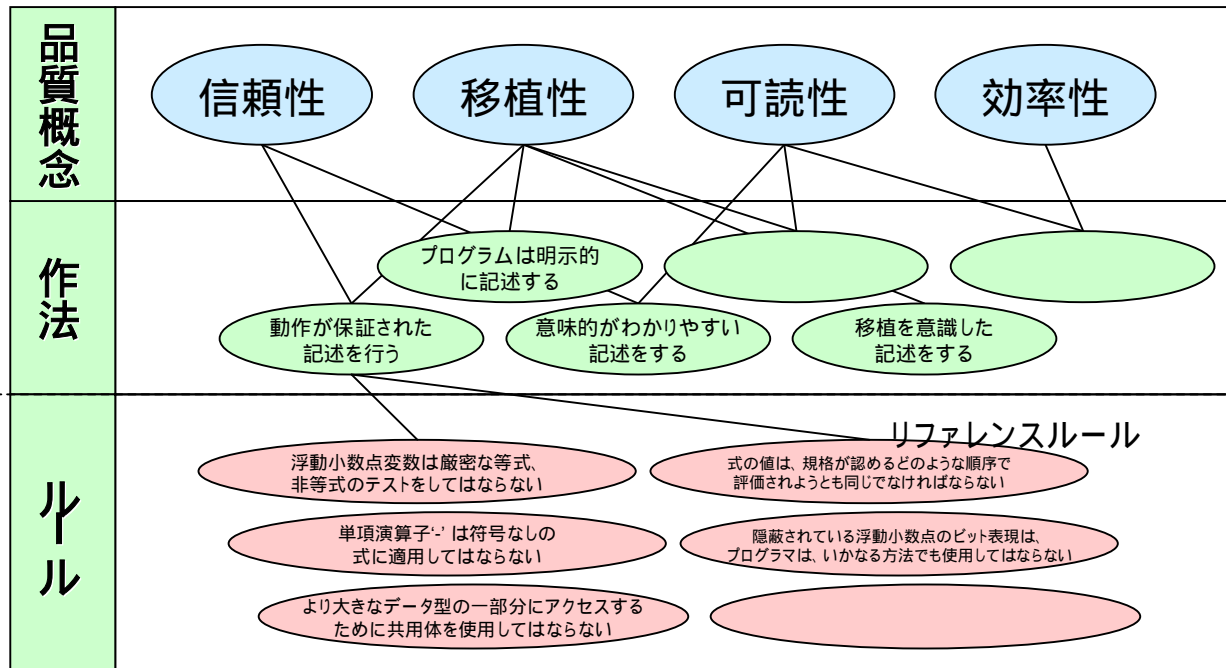
コーディング作法 良いソフトウェアをつくる為の考え方・上位概念

# SEC版-コーディング作法策定の流れ

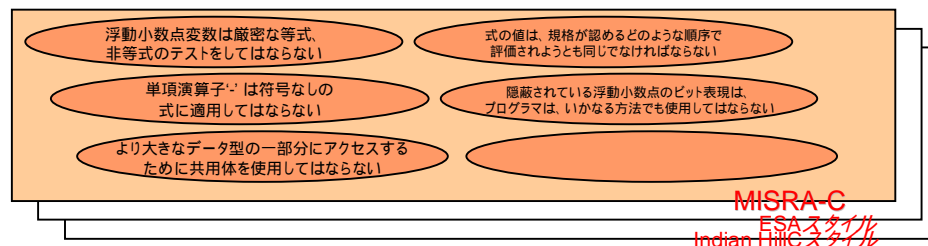
- 既存規約を参考に, 委員会で策定



Input



↓  
 各PJのコーディング規約 作法ガイドを参考にして作成



# コーディング作法の一部紹介(例)

## 信頼性向上の作法

作法		表現ルール
概要	詳細	
領域はきちんと初期化する	領域は、初期化してから使用する。	自動変数は初期化してから使用する。または使用する直前に初期値を代入する。
		const型変数は、宣言時に初期化する。
データの範囲、サイズや表現を意識する	内部表現に依存しない比較を行う。	浮動小数点型変数は厳密な等式、非等式の比較はしない。
		浮動小数点型変数はループカウンタとして使用しない。
		構造体や共用体の比較にmemcmpを使用しない。
	データ型を揃えた演算や比較を行う。	符号なし整数定数式の比較は、結果の型で表現できる範囲内で行う。
適切な型を示す接尾語が使用できる定数記述には、接尾語をつけて記述する。		

# 高品質実装のためのコーディング作法とは？

## 上級技術の問題点

技術天才を作ることにはできるが、

誰も10万人つくれるとは言わない

本来数学センスは無い方が当たり前

組込みシステムへのニーズは10万人ではまかないきれないものがある

- 誰が見ても納得する
- 見る事が実に退屈な
- 興奮しようのないプログラム



**コーディング作法の必要性！！**

二上貴夫 「ユービキタス時代の組込みソフトウェアの高信頼性について」 第23回ソフトウェア生産における品質管理シンポジウム、日科技連、2004年

# コーディング作法 運用に当たって…

- ・プロジェクト特性に応じた最適な規約の策定
- ・目標の設定
- ・規約のメンテナンス

- ・教育
- ・普及
- ・意識改善

- ・静的コードチェックツール
- ・リバースツール
- ・SCM評価
- ・レビュー・インスペクション

決める

×

守る

×

確かめる

どれかが0であれば効果は無い！

=

高品質ソフトウェア

- ・信頼性向上
- ・保守性向上
- ・移植性向上

ご静聴ありがとうございました。