コーディング作法とその効用
実用的ソフトウェアエンジニアリング目指して

2004年9月10日

組込みソフトウェア開発力強化推進委員会
門田　浩

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

組込みソフトウェア開発に関する課題の整理

準備委員会活動報告より

課題整理の枠組み

(1) 組込みソフトプロセス技術

組込みソフトウェア開発に関するプロセス的な側面

・開発プロセスの定義
・開発プロセスの実施状況の評価改善

(2) 開発技術

組込みソフトウェア開発における

・要求獲得・定義手法
・設計手法,実装手法
・テスト検証技術

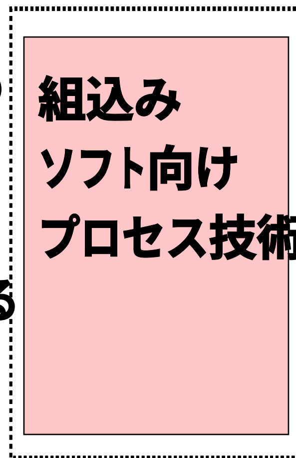
(3) 管理技術

組込みソフトウェア開発における

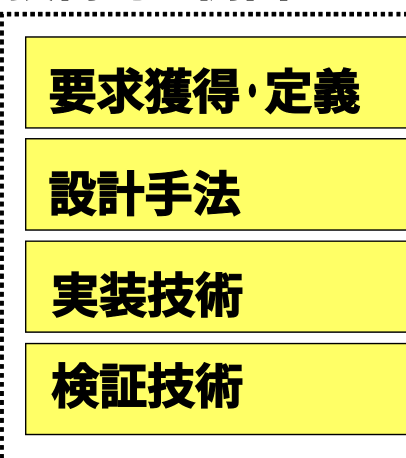
・品質管理技術
・進捗管理やプロジェクト管理などの開発管理技術

現状の開発

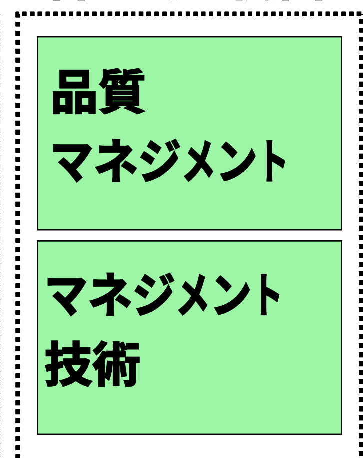
プロセス的な側面



技術的な側面



管理的な側面



プロセス面の解決策を提供

組込みソフトプロセス評価・改善技術

技術面の解決策を提供

組込みソフト開発技術

管理面の解決策を提供

組込みソフト管理技術

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

課題解決策の重要度評価結果

	Short	Mid	Long
Gr-A	+レビュー/インスペクション +コーディング規約整備	+プロセス評価手法 +設計可視化技術 +再利用技術 +開発管理手法/可視化 +見積り技術	+プロセス定義/設計
Gr-B	+プログラム可視化技術	+モデリング技術 +部品化技術 +テスト項目生成技術 +品質計測可視化技術 +重点テスト	+要求を記述整理する技術 +プロセス改善手法の整備 +運用の仕組み
Gr-C	+情報共有	+コンポーネント評価技術 +汎用テスト環境 +テスト自動化	+要求を獲得する技術 +要求を検証する技術 +コデザインとの連携 +コード自動生成技術 +静的検証 +リファクタリング

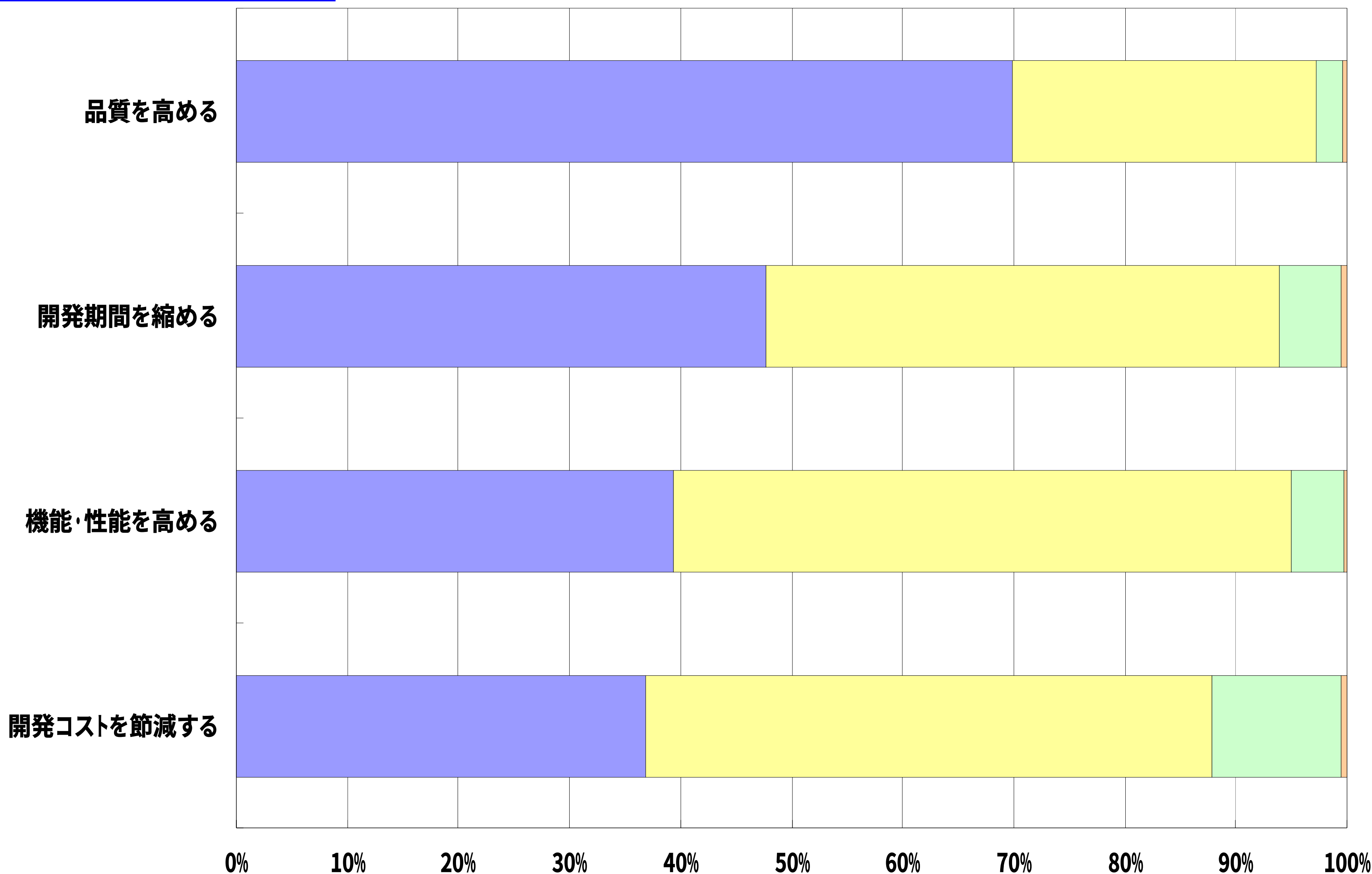
研究/開発スパン

準備委員会活動報告より

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

Q2-1　組込みソフトウェアの改善するべき領域

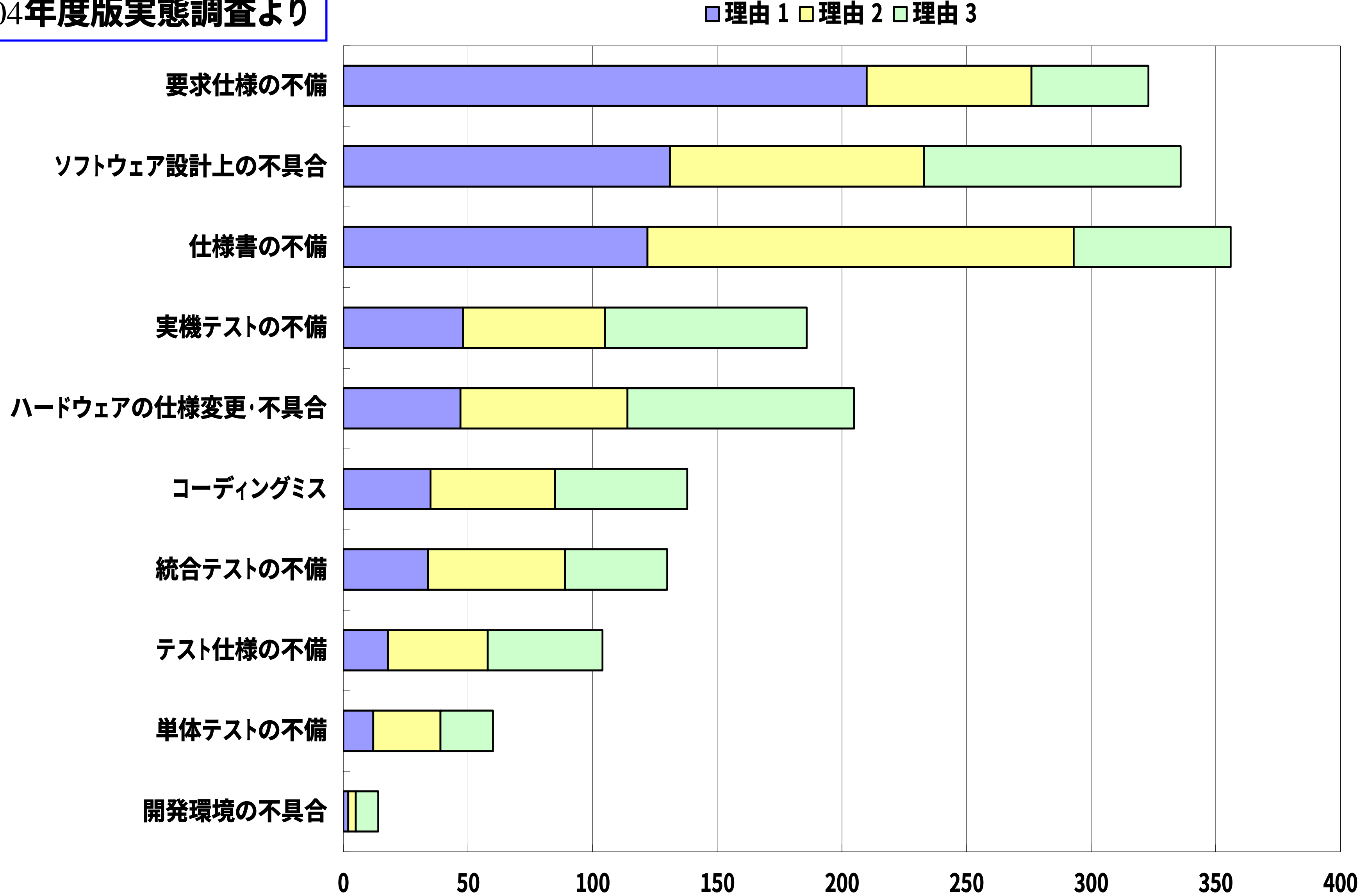
2004年度版実態調査より



Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

Q4-2 組込みソフトウェア手戻りの原因

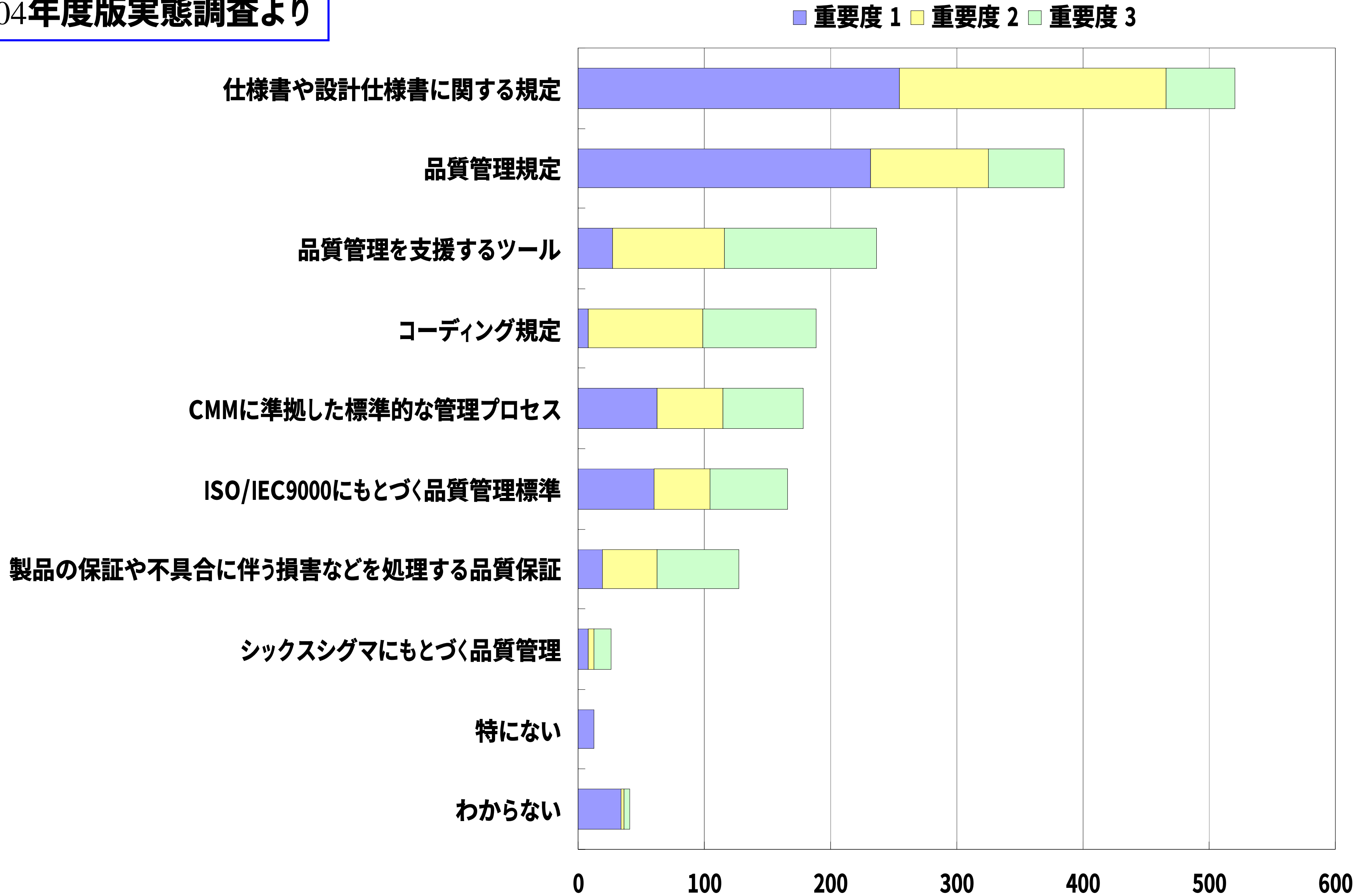
2004年度版実態調査より



Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

Q4-9　品質管理方法として重要と考えること

2004年度版実態調査より

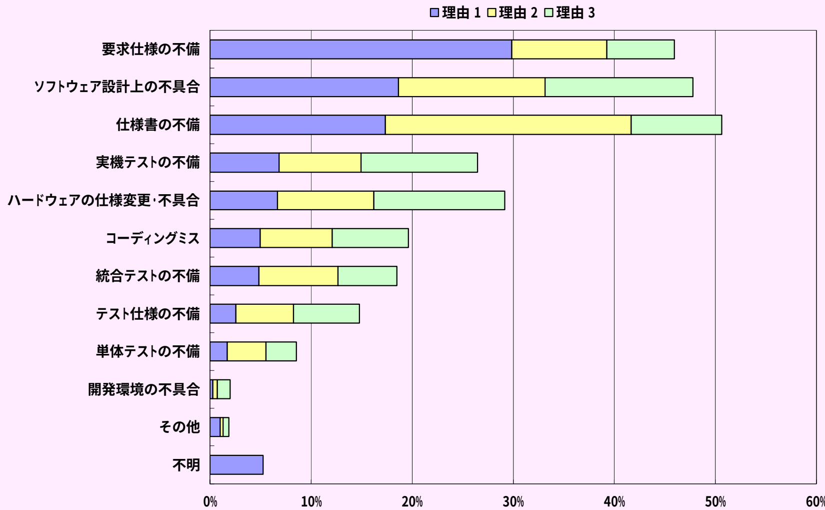


Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

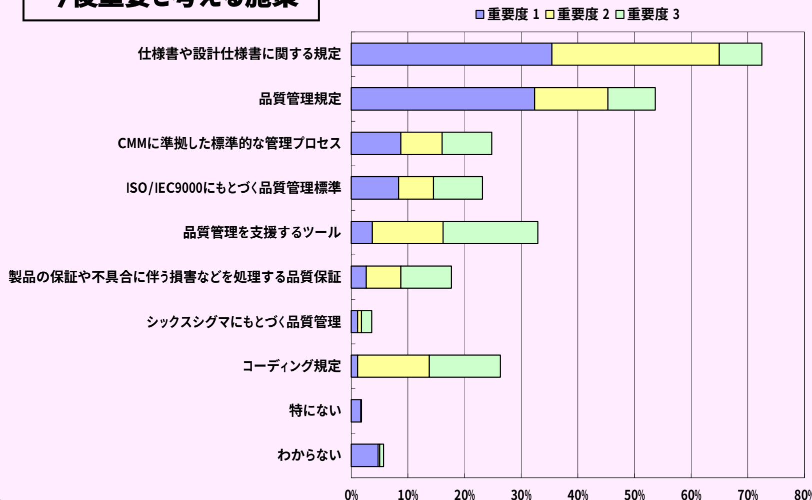
■ 部会テーマ選定理由

組込みソフトウェア産業実態調査

不具合発生の原因



今後重要と考える施策



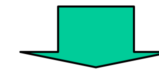
組込みソフトウェア設計力強化準備委員会

解決策開発の優先順位

	Short	Mid	Long
Gr-A	+レビュー/インスペクション +コーディング規約整備	+プロセス評価手法 +設計可視化技術 +再利用技術 +開発管理手法/可視化 +見積り技術	+プロセス定義/設計
Gr-B	+プログラム可視化技術	+モデリング技術 +部品化技術 +テスト項目生成技術 +品質計測可視化技術 +重点テスト	+要求を記述整理する技術 +プロセス改善手法の整備 +運用の仕組み
Gr-C	+情報共有	+コンポーネント評価技術 +汎用テスト環境 +テスト自動化	+要求を獲得する技術 +要求を検証する技術 +コデザインとの連携 +コード自動生成技術 +静的検証 +リファクタリング

研究/開発スパン

Short: 比較的短期間(1～2年)で実現
Mid : 中期間(3年～5年程度)で実現
Long: 長期間(5年以上)での実現



Phase-1　テーマとして
品質向上技術
プロジェクト管理技術
開発プロセス技術
を選定

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

アウトプットの考え方

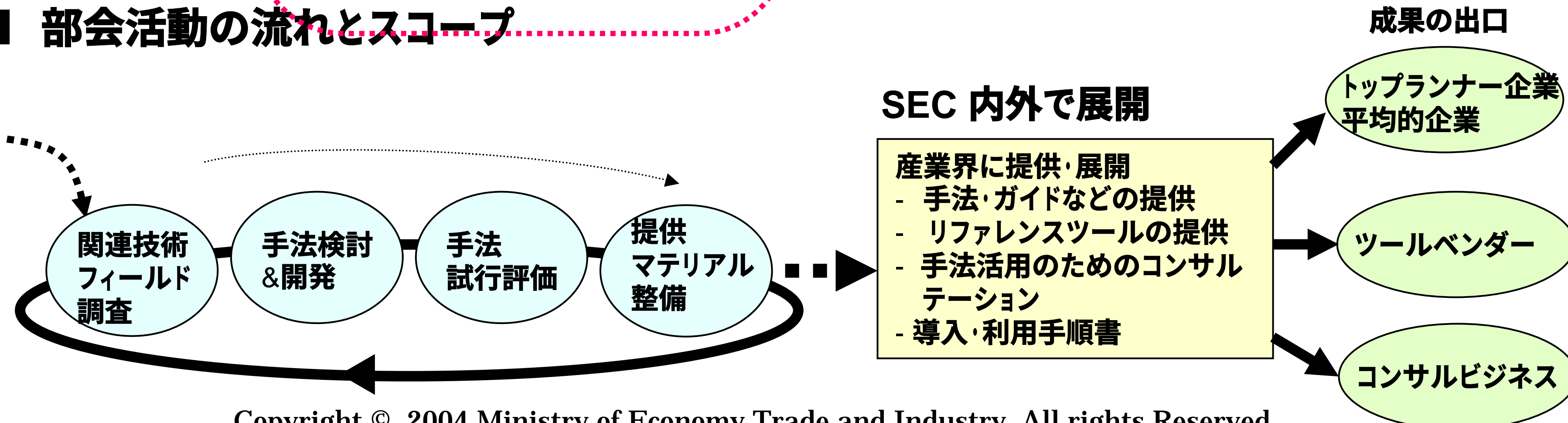
METI

■ 部会全体計画と予定Output

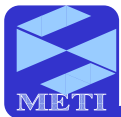
	16年度	17年度		18年度	
	組込みソフトウェア開発基礎体力の向上	組込みシステムとしての設計&管理精度の向上		組込みシステムの開発プロセスの最適化	
品質向上技術部会	•コーディング規約ガイド	•インスペクション/レビューチェックリスト	•設計記述テンプレート	•設計品質ガイド	
プロジェクト管理技術部会	•計画立案チェックリスト •計画書標準フォーム		•予実管理マニュアル&チェックリスト	•PJシミュレータ（リファレンス）	•見積り手法ガイド
開発プロセス技術部会		•プロセスガイド	•プロセス評価手法ガイド		•プロセス改善ガイド

*各ガイド・マニュアルについてはツール（リファレンスツール）による支援方式も適宜検討

■ 部会活動の流れとスコープ



Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved



品質部会スケジュール

WG-Goal & Mission

組込みソフトウェアの品質を確保・向上するための手法・ツールを整備する

Theme

プログラム品質の確保
- コーディング規約整備
- レビュー/インスペクション手法整備

設計品質の向上
- 設計手法の標準化
- 設計の可視化手法整備

Activity

①現状調査（インタビューなど）
②標準的手法の検討・開発
　ガイド/マニュアル整備
③開発した手法の試行適用

Outputs

- コーディング規約&ガイド
- レビュー/インスペクション標準チェックリスト
- 設計記述フレーム（テンプレート）
- 設計品質評価メトリクス＆ガイド

Main Effects

- コード&設計書のチェック・レビュー精度を向上
- コーディングミスや設計書の不備に起因する不具合を削減

Medium -term Schedule

凡例：（斜線ボックス）実態調査項目検討

			16年度	17年度	18年度
SEC主体	プログラム品質	コーディング規約整備	調査(国内外) → Ver.1 検討 → ▲ 規約&ガイド 内部トライアル →	公開・試行評価 → * プロセスWGと連動	
		レビュー/インスペクション	チェックポイント調査 →	標準チェックリスト検討 → トライアル公開 → ▲ インスペクションチェックリスト	
	設計品質	設計記述フレーム	設計記述フォーム/フレーム調査 ‥→	記述フレーム検討 → EML: Embedded Modeling Language	公開・試行評価 → ▲ 設計記述テンプレート * プロセスWGと連動
		設計品質可視化手法		設計可視化手法調査 ‥→ 設計品質メトリクス検討 →	トライアル&評価基準設定 → 公開・試行評価 → ▲ 設計品質ガイド
委託	検討中		記述フレーム先行検討 → プロダクトライン技術先行調査 →	設計検証方式基礎検討 → プロダクトライン実用化-手法先行検討 →	

Comments

- ✓ 再利用技術については組込みソフトウェア向けプロダクトライン関連技術の外部研究機関（大学など）へ研究委託を通して推進
- ✓ 設計記述フレームは将来的に設計検証技術への布石
- ✓ 国内関連大学との連携による先行検討を進める

igh

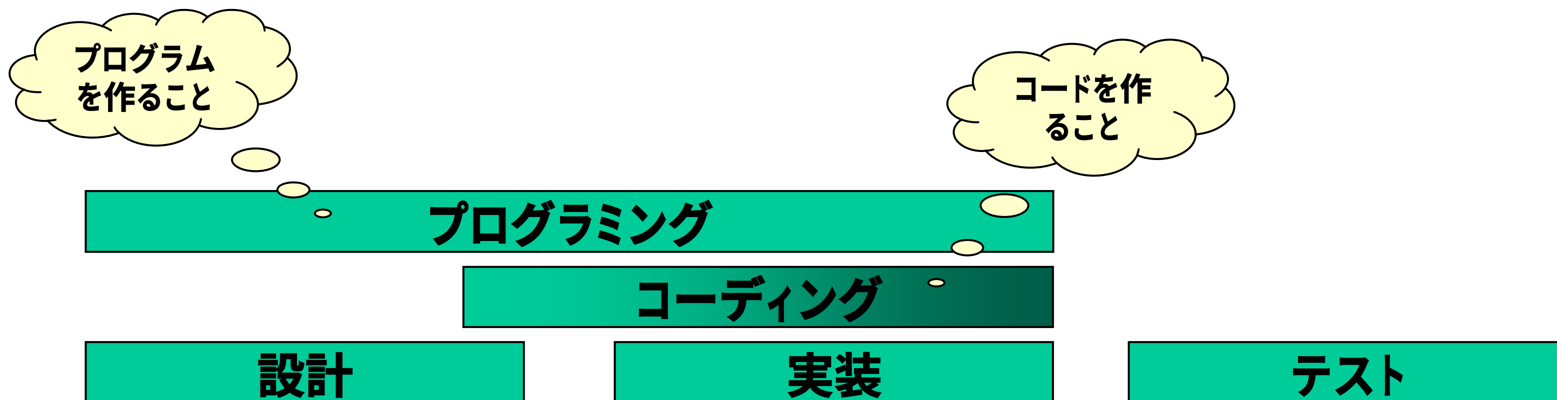
アウトプットイメージ

Outputs Detail

成果物名	概要	対象ユーザ	主たる利用シーン
コーディング規約&ガイド	•組込みソフトに適したコーディング規約作成ガイドを開発する •全般に共通するルールとドメインを考慮した特化ルールを層別 •コーディング規約策定のためのメタルール的なものも検討 •特にハード/ソフトの境界領域など不具合が多い部分に注目	応用開発企業	コードチェック（受け入れ検査含む）
		受託開発企業	納品時前のコードチェック
		ツール開発	コードチェックツール開発
レビュー&インスペクション標準チェックリスト	•コードレビュー&インスペクションなどを行う際のチェックポイントなどをチェックリストとして整備する. •コードレベルの単純なチェック項目とともに,設計の意味的な要素まで加味したチェック項目を整備. •また,インスペクションなどの標準的な作業手順や帳票などの整備も合わせて行う	応用開発企業	•内部でのコードレビューやインスペクションに利用 •成果物のレビュー、インスペクション方法策定時にも利用
		受託開発企業	
		ツール開発	レビュー＆インスペクション支援ツール開発
設計記述フレーム	•設計品質の均一化と可視化のベースとして位置づける •組込みソフトの設計として必要な情報を網羅する枠組みや組込みシステムに適したモデリング手法と表記法を決める •ハード情報の扱いとソフト/ハードの境界領域の記述方法がポイント	応用開発企業	•設計書の参照フォームとして利用 •ハードウェア開発者とのすり合わせにも利用 •関係者間の設計ドキュメント書式を統一
		受託開発企業	
		ツール開発	設計記述支援ツール開発
設計品質評価メトリクス&ガイド	設計品質記述フレームをベースにした設計品質定量評価のために HW&SWの接点にフォーカス ① メトリクスを制定する ② メトリクスの計測方法と評価基準値なども制定する ③ 評価結果の設計へのフィードバック方法を提案する	応用開発企業	設計レビュー時の参考として設計品質確認
		受託開発企業	設計レベルでの品質確認用資料として利用
		ツール開発	設計品質評価ツール開発および品質コンサルテーション

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

品質作りこみの流れ

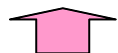
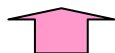


コーディング作法

開発者：

--基本的には良いコードを
開発者が作るための作法

機械チェックできる項目
機械チェックできない項目

設計インスペクション

コードインスペクション

意味的側面
設計的要素

設計上の
チェックポイントなどを提示

結構重なるところも
あるのかな？

インスペクションガイド

管理者:

--第３者が各プロセスの成果物を
チェックするためのガイド
-- チェックのやり方（手順）と
チェックの観点を提示

でも管理者などのチェックは
重箱の隅つつくのではなく
もっと本質的な部分の確認を
重視したいような。。。

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

コーディング規約（ガイド）とインスペクションガイド

	コーディング規約（ガイド）	インスペクションガイド
狙い	• コーディングのポカミスをなくす • 設計者自身による品質を意識したコーディングの実現 • コードのシンタクティクス面での品質保証 • ツールなどによるチェックを前提	• 設計（コード）の意味的な側面からの品質確認を実施 • 設計者などによる人間系による目視確認が前提
主たるユーザ	• プログラマ（初中級） • ツールベンダー	• 担当以外の第3者（管理者など）
対象物	• ソースコード	• ソースコード •設計書 / 仕様書
内容	• コーディングの作法（基本的な心構え） • 品質を意識したコードの記述形	• システム全体をインスペクションできないことを前提に注意すべき点/重点的にレビューすべき点/重点的にテスト・検証すべき点などをリストアップ ・設計書としてあるいは仕様書として注意すべき点/ 考慮されていなければならない点 ・インスペクションの必要性と標準的な手順

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

コーディング規定策定の基本方針

METI

- 広く使われる規定
  - 現場技術者にも受け入れられる規定 (管理の都合だけでない規定)
    - 規定を守れるようになるとスキルが上がる
    - 規定を守るとムダが減り効率が上がる
    - コミュニケーションがスムーズになる、など
  - 柔らかい規定 (規定の作り方と規定サンプル)
    - 導入の目的、組織の実力、技術者のレベル、開発するソフトウェアの特性などにより適切なルールを定義できる枠組み
- スキル標準とリンクした規定
  - 技術者のスキルレベルに応じたルール設定・適用範囲
    - レベル0：基本セットの適用 (まずは心得から)
    - レベル1：標準セットの適用
    - レベル2：拡張セットの適用 (一部の例外も許すルール適用)
    - レベル3：例外も認めて応用動作も許す (ルールの改訂もできる)
    - レベル4：ルールは適用しない？
- 設計プロセスに合った規定
  - コーディング工程より前の工程で定義しておくこと
    - 名前付けルール (ハードウェア関連の名前付けも考慮して)
    - 主要なデータ構造・広域変数・関数・定数などの定義 (名前と機能)
    - 使用するOS、ミドルウェア、ライブラリ、ツールに合わせたローカルルール
  - コーディング工程の後の工程で使われ方
    - チューニング、テスト、ドキュメンテーション、再利用

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

段階的な適用を考慮したルール階層

- 基本セット
  - ソフトウェア開発一般で有効なルール
  - プログラミング言語非依存、処理系非依存
  - ローカルルール無し
- 標準セット
  - 組込み系ソフトウェアの開発で有効な要素
  - プログラミング言語依存、処理系非依存
- 拡張セット
  - 応用製品の開発で有効な要素
  - 例外の規定（実行時間、コードサイズの優先など）
  - 処理系依存
  - ローカルルール有り（プロジェクト依存）

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

コーディング作法-1

対象者：　①　初級/中級のスキルをもつ開発者（プログラマ）
　　　　　②　上記担当者のコードをレビューする開発リーダなど

期待効果：　①　コーディングのミスを削減
　　　　　　② 比較的レベルの高い技術者のスキルを真似て身につけることができる
　　　　　　⇒作業者にとって
　　　　　　　　作業誤りが減り時間の無駄がなくなる
　　　　　　　　スキルアップが図れる

作法とルール：
　　作法：基本的な心構えや考え方
　　ルール：作法の実現手段としての詳細な決め事

ツールでチェックできるルール ツールでチェックできないルール	×	コーディング前に考慮すべきルール コーディング中に注意すべきルール 検査時に注意すべきルール

×

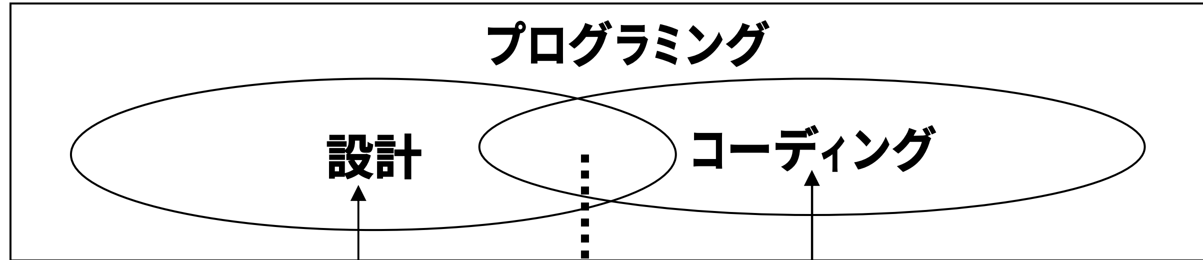
業界ルール　/　社内ルール　/ プロジェクトルール に層別

＊意味的側面を重視

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

コーディング作法-2

Scope:



設計作法/ルール

コーディング作法/ルール

現実にはコーディング作業中に
一部機能の設計を平行して行う
⇒ コーディング作法の一部には
設計的な要素も加えておく

コーディング作法で抑えるべき特性

結局は ソースコードの質の確保

ファイル名なども含めた
ソースコード

移植のし易さ/ 保守のし易さ
性能 (効率) のよさ / 信頼性

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

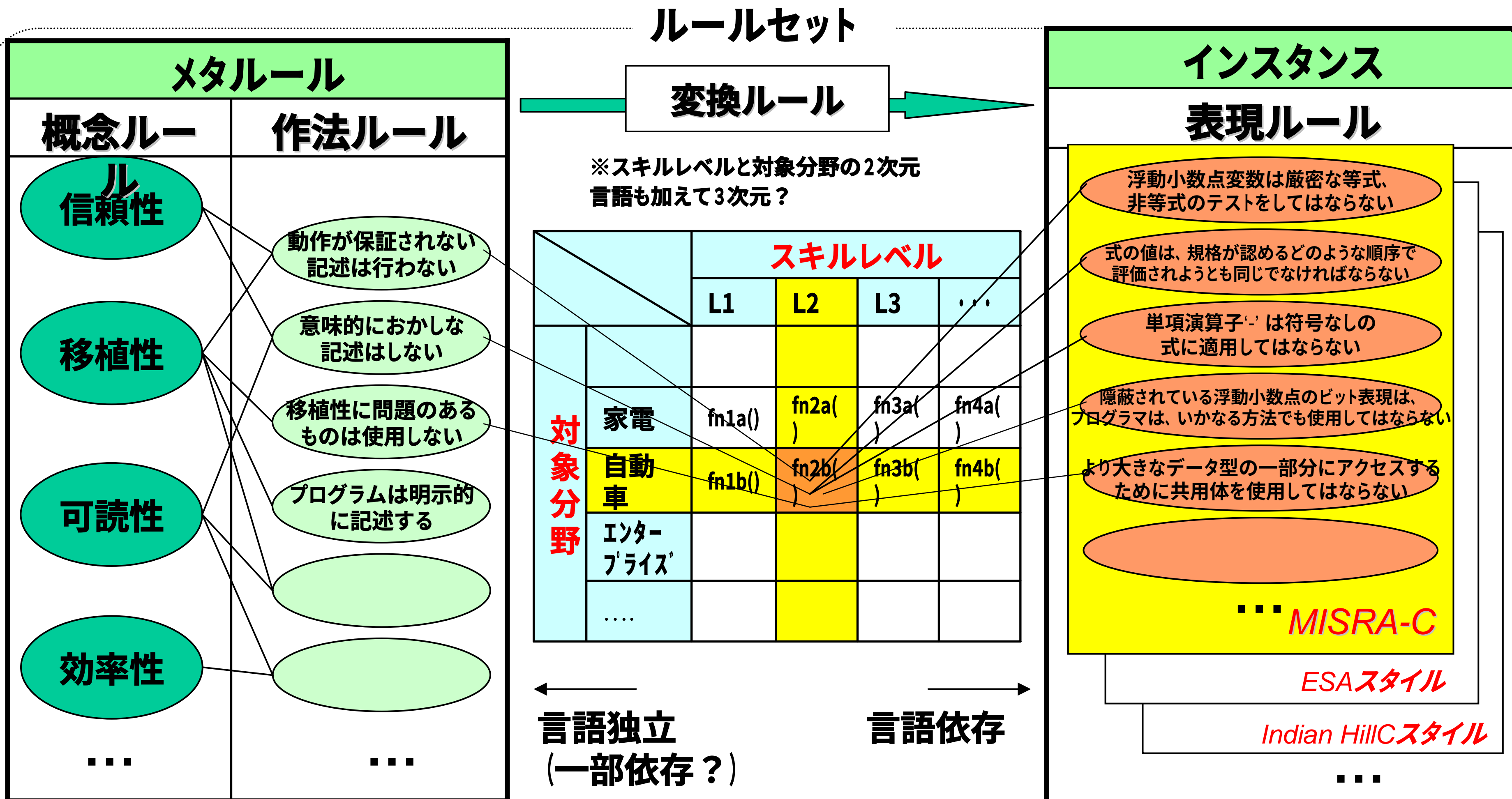
Goalはメタルール

- メタルールを用い、インスタンスを導く仕掛け
- 概念カテゴリは, ISO/IEC-9126の品質特性から採用
  - 信頼性
  - 効率性
    - 実行効率性
    - 資源効率性
  - 保守性
  - 移植性
- 作法レベルは, 既存ルールを参考に今後策定
  - ツールチェックの可能性の明示
  - 適用工程との関係の明示
    例：コーディング工程より前の工程で明示が必要なルール
    - 名前付けルールを決定する

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

フレームワーク(案)と検討アプローチ

①メタルール抽出：具体的インスタンス(例：MISRA-C)からメタルールを抽出 (BOTTOM-UP)
②メタルール検討：メタルールの必要十分性を検討 (TOP-DOWN)
③洗練　　　　　：対象分野毎にメタルールとインスタンスとの対応を検証し完成度を向上



OUTPUT：具体ルールセット例を作成

Copyright © ... hts Reserved

MISRA-Cをインスタンスとしたメタルール抽出例

概念ルール	作法ルール	MISRA Rule No	MISRA 類型	MISRA-Cルール/その他のルール（インスタンスルール）	信頼	堅牢	可読	保守	移植	効率
信頼性	動作が保証されない記述は行わない。	30	必要	全ての自動変数は使用する前に値を代入していなければならない。	○					
		46	必要	式の値は、規格が認めるどのような順序で評価されようとも同じでなければならない。	○				○	
		50	必要	浮動小数点変数は厳密な等式、非等式のテストをしてはならない。	○			○	○	
		65	必要	浮動小数点変数はループカウンタとして使用してはならない。	○				○	
		79	必要	void関数が返した値は使用してはならない。	○					
		80	必要	void式は関数の引数として引き渡してはならない。	○	○				
		103	必要	2つの被演算子が同じ型で、同じ配列や構造体、共用体を指し示す場合を除いては、関係演算子はポインタには適用してはならない。	○				○	
		106	必要	自動領域にあるオブジェクトのアドレスをオブジェクトの存在が終了した後にまで持続期間をもつオブジェクトに代入してはならない。	○					
		107	必要	ヌルポインタは参照を解除してはならない。	○	○				
	意味的におかしな記述はしない。	36	推奨	論理演算子はビット演算子と混同すべきではない。	○		○			
		51	推奨	符号なし整数定数式の評価は結果の型にはまるべきである。	○		○			
		39	必要	単項演算子'-' は符号なしの式に適用してはならない。	○		○			

⋮

※本サンプルはイメージを伝えるもので個々の内容については今後検討予定

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

METI

WGでの話題1

堅い話と柔らかい話

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

「覚え易い」「使い易い」ルールにしよう

- 段階的に覚えられる
  - 概念ルール＞作法ルール＞表現ルール
  - スキルレベル毎のセット
- 「書かせない」ではなく「書かせたい」ことをまとめる
  - 例：「8進数は使わない」ではなく
    - 定数は１０進数を使う
    - ただし、ビットフィールド（I/Oレジスタなど）には１６進数を使う
- 例外も認める
  - 例：「goto文は使わない」
    - 定義された異常処理への分岐には使用できる
- 語呂合わせで覚えられる
  - 平方根の「人並みにオゴレや」「富士山麓にオウム鳴く」
  - 味付けの「さしすせそ」（ちょっと年寄りじみている）

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

ルールベースのコーディングを推進する枠組み

- 新人プログラマのための教育
  - ルールの前に心得を修得
- 既存プログラマのための矯正
  - 癖の把握とアドバイス (生活指導)
  - スキル判定とスキルに合ったルールの選択 (処方箋)
- ツールも変えよう
  - 現場の状況に合わせられるツールに
    - 会社あるいはプロジェクト毎にルールを定義できる
    - 個人ごとに適用範囲を変更できる (スキルに応じて)
    - 関数あるいはファイルごとに例外項目を指定できる
  - ルール違反を検出するツールからコーディングをアドバイスするツールに (プロにはうるさいかも)
    - 適用ルール (定義されたルール) の評価
    - 個人ごとの「癖」データベースとそれに基づくアドバイス
    - ルール達成度 (ルールをどの程度守れたかの指標) の表示

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

話題その2
ボトムアップ的討議アプローチ

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

METI

プログラマに守らせるルールと検査すべきルールは分けて考えよう

- コーディング規約の最終目的を明確にしよう (局所最適にならないように)
  例:「一定以上のコーディング品質を確保することにより、要求品質・要求性能達成までのソフトウェア開発工数を減らすと共に、プログラムに潜在する不具合を減らす」
- 目指したいコーディング規約
  - プログラマが守るべきルールは減らしたいが、最終品質は確保したい
    - 機械的に検査すれば充分なルールは、プログラマが守るルールからは除く
      - 機械的な検査をし易くするためのルールは設定
  - ルールを守ってコツ (本質) をつかむとスキルが上達できるようにしたい
    - プログラミングの達人技をルール化したルール (定石ルール) も設定
      - 組込みシステム色を出したい部分、チューニングも考慮
  - 例外的なルールの逸脱は許したいが、常に必要では意味が無い
    - 逸脱は緊急避難措置のみに限定したい
      - 限定しても充分なようにルール設定できるようにしたい＝現場に合わせてよく考えられたルール
- 工程毎にルールを分けて定義しよう
  - コーディング工程以前に定めるルール (ルール定義規約)
    - プロジェクトで適用するルールの定義
    - コーディング工程に関連するプロセスの定義
  - プログラマが守るルール (コーディング規約)
    - コーディングスキルに依存せず全員が守るルール (スキル非依存ルール)
    - コーディングスキルに依存したルール (スキル依存ルール)
  - 検査工程で検査するルール (検査規約)
    - 機械的に検査できるルール
    - 目視もしくは実行テストでなければ検査できないルール
- ルールの階層を明確にしよう
  - 応用分野あるいは業界で定められたルール (業界標準ルール)
  - 会社あるいは組織で定められたルール (社内標準ルール)
  - プロジェク毎に定められたルール (プロジェクトルール)

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

ルールの目的を明確にして適用範囲を考えよう

- ミス防止
  - 人間の勘違いなどのミスを未然に防ぐための機械的検査をしやすくする
    - 例：8進数は使用しない
  - コンピュータの基礎知識が不充分なことによる誤りを未然に防ぐ（スキルが高ければ不要なルール）
    - 例：浮動小数点の等号による比較をしない
- 移植性確保＋人間のミス防止（複数のプロジェクトを抱えるプログラマ）
  - プロセッサアーキテクチャ依存の相違を隠蔽する（限定すれば不要なルール）
    - 例：構造体の一部を別の方法でアクセスしない
  - プロセッサ品種依存の相違を隠蔽する（限定すれば不要なルール）
    - 例：プロセッサ制御レジスタにマップされた変数を直接アクセスしない
  - コンパイラ（処理系）依存の相違を隠蔽する（限定すれば不要なルール）
    - 例：条件式の評価順に依存するコードは書かない
  - システム構成（メモリやI/O構成など）依存の相違を隠蔽する（限定すれば不要なルール）
    - 例：メモリにマップされたI/O変数を演算式で使用しない
- 効率向上（プロセスも一緒に定義しないと意味がない）
  - デバッグ・テストをし易くする
  - コーディングレビューをし易くする
  - 保守をし易くする
- 性能向上（定石ルール）
  - メモリ効率を高める
  - リアルタイム性能を高める・安定させる
- 環境制約（使用するプロセッサ、コンパイラ、OSなどに依存して設定せざるをえないルール）
  - プロセッサアーキテクチャ、プロセッサ品種、コンパイラ、システム構成、リアルタイムOS依存など
    - 例：タスクの主ルーチンとなる関数にはTASKを指定する

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved

METI

ご清聴、ありがとうございました

今後も成果は継続的に発信し、皆様のご意見を反映いたします。

Copyright © 2004 Ministry of Economy Trade and Industry, All rights Reserved