組み込み開発者におくる
MISRA-C:2012

日本語版解説書ができるまで

MISRA－C研究会
(株)デンソー　電子技術1部
池田　元三

目次

1-1. Introduction（目的）

【目的】

今回の講演において、MISRA-C自体およびMISRA-C:2012とどう付き合えばいいかを**ある程度**理解してもらうとともに、

MISRA-C研究会（NPO法人組込みソフトウェア管理者・技術者育成協会:SESSAMEのWG3）の行っている活動および、その活動の成果物である「MISRA-C解説書」についてどういうものか？どう読めばいいか？を**何となく**理解してもらいたいと思っています。

そして、話の中に込められた（特に）**若い人へのメッセージ**を**何となく**感じてもらえればと願っております。

1-2. Introduction（前提）

本セッションを聞いて頂いている方々に対して、以下のような前提とお願いをさせて頂きます。

【前提】 想定した聴講者像

学生、初級者大歓迎ですが、以下の方を想定しました。

- ソフトウェア開発の経験がある
- C言語での開発経験はあるが言語規格に詳しくはない
- 少しは「MISRA-C」について聞いたことがある
- できれば「MISRA-C：2004」解説書を読んだことがある

【お願い】

- 分からないことがあれば、**その時に**何でも聞いて下さい
- 細かな技術的な質問に対する議論は、別途とさせて下さい

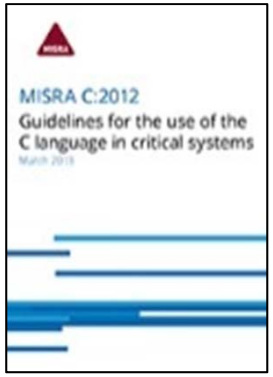
1-3. Introduction（書籍）

組込み開発者におくる MISRA-C 組込みプログラミングの高信頼化ガイド
by MISRA-C研究会

いわゆる解説書

組込み開発者におくる MISRA-C 2004 C言語利用の高信頼化ガイド
by MISRA-C研究会

解説書第2版



MISRA C:2012 Guidelines for the use of the C language in critical systems
by MISRA

最新版MISRA-C

（注）“MISRA”と”MISRA C”は、MIRA Ltd. のトレードマーク（商標）です

1-4. Introduction（自己紹介）

【自己紹介】

(株)デンソー（当時は日本電装）入社以来、エレクトロニクス分野を転々。 研究開発・量産設計・品質保証などに従事しハード・ソフトのみならず、ダイキャスト・プレス・樹脂ケースという部品の設計まで経験し、今は希望して人材育成を担当。

入社時に漠然と願っていた、

- ナビの開発（特許で〇百万円稼ぐ）
- レースの仕事（中島悟、関谷正徳と仕事できた）
- 海外の仕事（米国に計11年駐在した）

をいつのまにか達成。 運を味方にチャンスには飛びつけ！

「願えば叶う」 未知の領域でも、知らない人に囲まれることになっても、めげずに頑張っていれば、必ず道は開ける！

2．MISRA-Cとは？

What is MISRA?

車載ソフトの安全性と
信頼性を支援

To provide assistance to the automotive industry in the application and creation within vehicle systems of safe and reliable software.

MISRA, The Motor Industry Software Reliability Association, is a collaboration between vehicle manufacturers, component suppliers and engineering consultancies which seeks to promote best practice in developing safety-related electronic systems in road vehicles and other embedded systems.

What is MISRA-C?

C言語のサブセット
（ガイドライン）

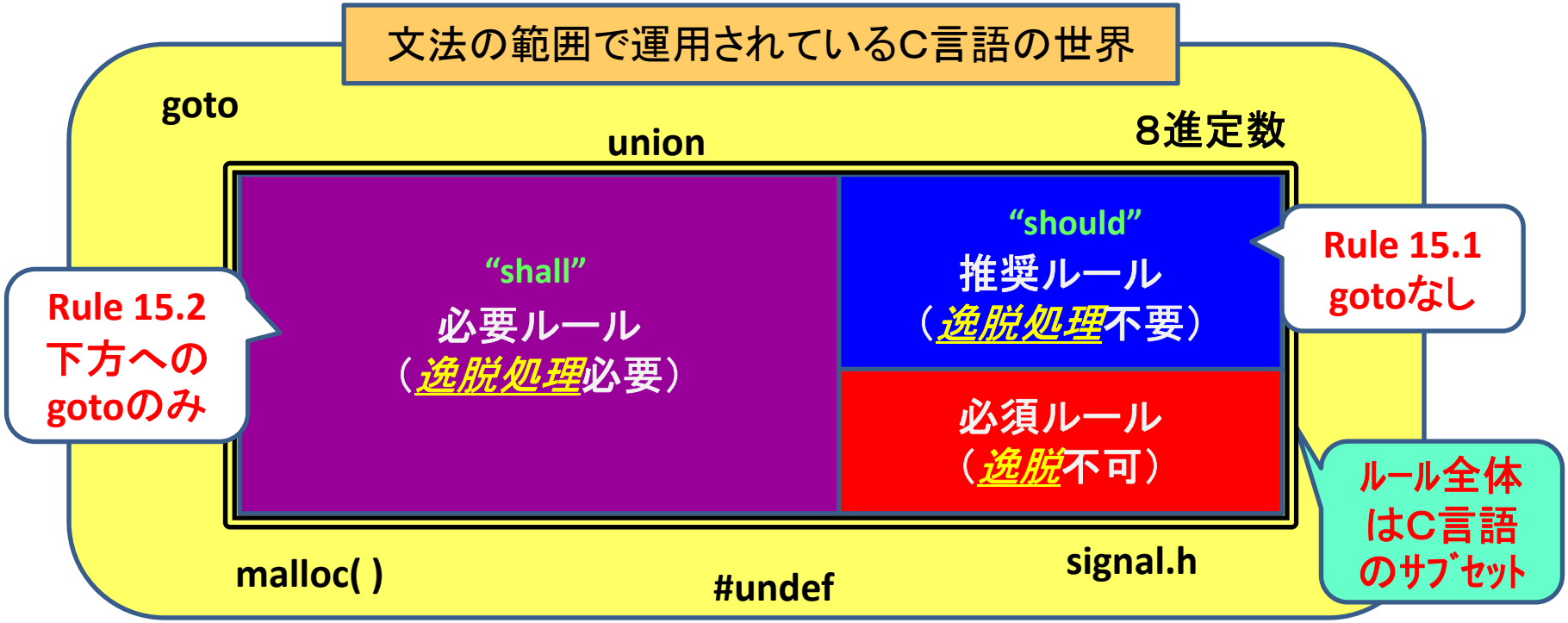
As part of these activities, MISRA C was first published in 1998. The intention was to provide a "restricted subset of a standardized structured language" as required in the 1994 MISRA Guidelines for automotive systems being developed to meet the requirements of Safety Integrity Level (SIL) 2 and above.

出典:MISRA Home Page http://www.misra.org.uk/MISRAHome/WhatisMISRA/tabid/66/Default.aspx

2-1. MISRA-Cの概要

【概要】

C言語を安全に書くためのガイドライン（ルール）であり、車載ソフトウェアにおいては標準コーディングルールになっているとも言え、（特に海外）自動車メーカからはこのルールへの準拠が要求されることも多い。



逸脱処理：ある必要ルールに準拠できない場合（非適合）は、「逸脱」するための処理が必要となる。 この逸脱処理では、その妥当性や安全性などを記録する

参1．MISRA-Cルール間の関係

Rule 15.1（推奨） goto文は、用いるべきではない

Rule 15.2（必要） goto文は、同一関数内で後方（下方）にあるラベルにジャンプしなければならない

Rule 15.3（必要） goto文から参照されるラベルは、同じブロック内またはgoto文を含むブロック内で宣言しなければならない

```
int f ( int loopnm, int loopmx )
{
    int i = 0;
L1:
    ++i;
    if ( i > loopnm )
    {
        goto L2; /* 非適合：Rule 15.1&15.3*/
    }
    if ( i < loopmx )
    {
        goto L1; /* 非適合：Rule 15.1&15.2 */
L2:
        --i;
    }
    return ( i );
}
```

15.2の逸脱処理が必要

上方へのジャンプなので15.2に非適合

推奨ルールなので逸脱処理は不要

gotoによるジャンプなので15.1に非適合

ブロック外へのジャンプ（宣言）なので15.3に非適合

15.3の逸脱処理が必要

2-2. MISRA-Cの歴史

【歴史】

① MISRA-C:1998　日本語版：自技会（TP01002）

Guidelines for the Use of the C Language in Vehicle Based Software

推奨（Advisory）34、必要（Required）93　= 127

ルールの数

② MISRA-C:2004（C2）　日本語版：自技会（tp-01002-06）

Guidelines for the Use of the C Language in Critical Systems

推奨 20、必要 121　= 141

③ MISRA-C:2012（C3）　日本語版：MISRA翻訳（予定）

Guidelines for the Use of the C Language in Critical Systems

推奨 31、必要 121、必須（Mandatory）7　= 159

（Directive 16、Rule 143）

両方合わせてguideline（ガイドライン）と呼ぶ

2-3. MISRA-Cの活用意義

【利用者と目的】

(1) 自動車メーカ

車載ソフトウェアの品質確保のために、電子部品（ECU）メーカにルールの準拠を要求する

非適合判定・逸脱処理の理解

ルールの意味するところの理解

(2) 自動車部品・組み込み製品メーカ/ソフトウェアベンダ

顧客からの指示により、または（特に車載）組み込みソフトウェアの品質確保のため、ソフト開発時の標準コーディングルールとして活用する

適合/非適合判定の理解

(3) 静的解析ツールベンダ

MISRA-C対応ツール（適合確認、非準拠の検出）の開発を行うための規格書として扱う

ルールの意味するところの理解

(4) ソフトウェア開発者/学生

安全で可読性の高い（組み込み）ソフトウェアを開発するためや、C言語自体を正しく理解するための教材とする

2-4. MISRA-Cの活用方法

【お勧めしない活用方法】

・MISRA-C本文または解説書を読まずに（MISRA-Cを理解することなく）適当にコーディングしてツールで警告が出た箇所を修正する

修正すればいいという訳ではない

・顧客指示または組織方針という理由で、深く考えずに非適合と判定された箇所を修正する

修正しなければいいという訳でもない

・非適合と判定された箇所を、その安全性を追求することなく逸脱処理をしてそのままにしてしまう

【お勧めしたい活用方法】

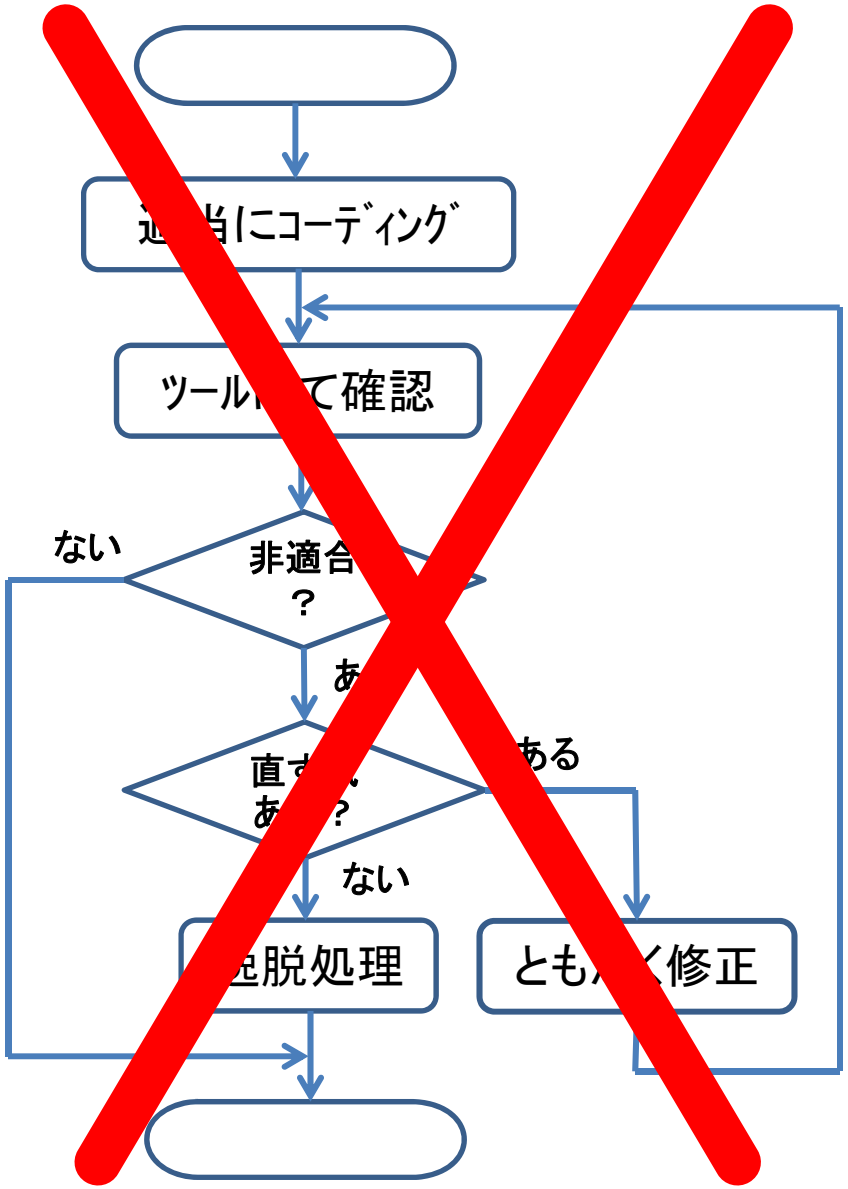
まず勉強

・MISRA-Cの思想/各ルールの意味するところを理解した上でコーディングする

基本は修正（次から楽）

・非適合と判定された箇所は、修正の必要性を吟味してから修正しない場合は安全性の確認と逸脱処理を行う

参2．MISRA-C対応フロー（bad example）



左記フローは、MISRA-Cのプロセスに確かに準拠しています。　でもね、

【問題】

1．「直す気ある」かどうか？で修正の可否を決めてはいけない
（現実は、時間がないとか、直し方が分からない、直すの面倒・・で決まることが多い）

2．「直す気ない」なら、その理由の明確化および妥当性の判断と、安全性の確認 etc. が必要！
それをしないで、安易にいい加減な「逸脱処理」をしてはいけない

３．MISRA-C:2012について

２０１３年３月１８日に第３版（C3）がリリースされた。

【主な特徴】

(1) ガイドライン（ルール）の枠組が変わった

・Ruleに加え、Directive（指針）を新規に追加

・推奨、必要ルールに加え、必須（Mandatory）ルールを追加

(2) 各ルールの構成が変わった

逸脱ができない

・ルール本文（あいまい表現）に加え補足と例外でルールを構成

(3) ツール活用の指針が強化された

・静的解析ツールによる決定可能性や解析範囲などを明示

(4) C99にも対応するようになった

・C90とC99の双方に対応

公式表現　通称表現

（注）RuleとDirectiveの両方をガイドラインまたはルールと呼ぶ

3-1. MISRA-C:2012の特徴1

(1) ガイドライン（ルール）の枠組が変更

・Ruleに加え、Directive（指針）を新規に追加

＊ Directive（指針）:

文法のみでは厳密に定義できないガイドライン。 ソースコードが「Directive」に適合しているか非適合であるかを判断するためには、プロジェクトの規則など、ソースコード以外の追加の情報が必要となる。

（例）Dir 2.1: すべてのソースファイルは，コンパイルエラーなしにコンパイルされなければならない → ソースコードだけでは判定できない

・推奨、必要ルールに加え、必須（Mandatory）ルールを追加

＊ Mandatory（必須）:

MISRA C:2012 への適合を主張するためには、適合していなければならない。逸脱は認められない。

（例）Rule 9.1: 自動記憶域期間を持つオブジェクトは、設定する前に値を読み取ってはいけない → 通常スタックに配置され、毎回初期化が必要

（注）上記外の変更は省略（質問は受け付けます）

3−2．MISRA-C:2012の特徴２ #1

(2) 各ルールの構成が変更

・ルール本文に加え、補足と例外でルールを構成

ルール 13.5（必要）
for 文の三つの式には、ループ制御に関わるものだけを記述しなければならない
“The three expressions of a for statement shall be concerned only with loop control”

Rule 14.2（必要）
forループは良い形式でなければならない

“A for loop shall be well-formed”

MISRA-C：2004

```
for ( x = 0, i = 0; i < 10; i++ )
```

非適合！

MISRA-C:2012

```
for ( x = 0, i = 0; i < 10; i++ )
```

これって良い形式？悪い形式？

3−2. MISRA-C:2012の特徴2 #2

(2) 各ルールの構成が変更

・ルール本文に加え、補足と例外でルールを構成

ルール 13.5（必要）

for 文の三つの式には、ループ制御に関わるものだけを記述しなければならない

“The three expressions of a for statement shall be concerned only with loop control”

for 文の三つの式を記述する場合、次の目的でしか用いてはならない

第1式	ループカウンタの初期化（次の例では ｉ ）
第2式	ループカウンタ（ ｉ ）のテストを記述し、オプションで他のループ制御変数（flag）を記述する
第3式	ループカウンタ（ ｉ ）のインクリメント又はデクリメント

```
for ( x = 0, i = 0; i < 10; i++ )
```

Rule 14.2（必要）

for ループは，（補足に記載の）良い形式でなければならない

“A for loop shall be well-formed”

【適用範囲】C90/C99
【分類】決定不可能, システム
【補足】
for文は以下の3つの節に分けられる：
第1節は，
・空でなければならない，または，
・ループカウンタに値を代入しなければならない，あるいは，
・ループカウンタを定義し，初期化しなければならない．（C99）
第2節は，
・持続的な副作用を持たない式でなければならない，そして，
・ループカウンタを使わなければならず，また追加でループ制御フラグを使用しても
よい，そして，　．．．

【例外】
3つの節すべてが空であってもよい．例えば，無限ループを許すため for (; ;)と記述してもよい．

3−3. MISRA-C:2012の特徴3

(3) ツール活用の指針が強化された

・静的解析ツールによる決定可能性を明示

＊決定可能:
ソースコードのみを静的に解析することによってルールへの適合/非適合が理論的に決定できること

・(適合/非適合の)解析を行う範囲を明示

＊範囲:
ルールへの適合/非適合を判定するために、ツールまたはレビューで解析しなければならないコードの範囲のこと(単一の翻訳単位かシステム全体のいずれかが明示されている)

・ルールの表現がツールに優しくなっている

Rule 19.2(推奨)
unionキーワードを用いるべきではない

ルール 18.4(必要)
共用体は用いてはならない (MISRA-C:2004)

大切なことは、ツールに依存しないで活用すること

ツールで発見しやすくなっている

４．MISRA-C研究会とは？

「MISRA-C」を正しく理解して、実用的な形でまとめて
日本の組み込み業界に紹介するグループ
参加メンバー：規格の専門家、コンパイラ開発者（言語規格の専門家）、組み込みソフト開発者、静的解析ツール開発者など

【歴史】

① MISRA-C:1998：第１期活動　2002-2004
　リーダ：東陽テクニカ、参加２３社/２８名

② MISRA-C:2004：第２期活動　2004-2006
　リーダ：東陽テクニカ、参加３１社/３５名

③ MISRA-C:2012：第３期活動　2013-2015
　リーダ：名古屋市工業研究所、参加１６社/１８名

4−1．MISRA研究会第１期活動

(1) 翻訳の支援

・自動車技術会が担当したMISRA-Cの翻訳を、ソフト開発者の知見を活用して支援した

(2) MISRA-Cの解読

・MISRA-Cが何者かまるで分かっていなかったので、どういう意図でルールが作られたかの理解を一生懸命行った

(3) MISRA-C解説書の発行

・初の試みであったが、日本規格協会の協力を得て解説書の第一弾を発行することになった

すべてが初めてであったが、研究会の基礎ができあがった

4−2. MISRA研究会第2期活動

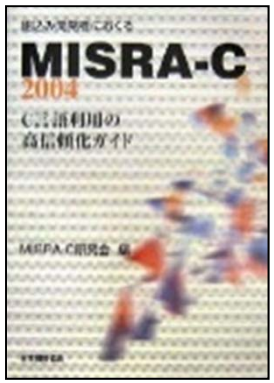
(1) MISRA-C2ドラフト版のレビュー

・ドラフト版を入手して全ルールのレビューを行った

・英国MISRAに出張して、レビュー結果を報告して修正を提案して受け入れられたり、新しい用語などの理解に務めた

(2) 翻訳の支援

・自動車技術会が担当したMISRA-Cの翻訳を、ソフト開発者の知見と過去の経験を活用して支援



(3) MISRA-Cの解読

・新しい概念、用語、第一版との違いの理解に努めた

(4) MISRA-C解説書の発行

・第一版の経験を活かして、加筆、サンプルの充実などを実施

MISRAの信頼も得て、ドラフトから参加して感謝された

4−3．MISRA研究会第3期活動

(1) MISRA-C3ドラフト版のレビュー

・(一部メンバーが)ドラフト版を入手して全ルールのレビューを行い、日本環境には不適切なルールの是正を図った

(2) 翻訳の支援

・MISRAが担当したMISRA-Cの翻訳を、ソフト開発者の知見と過去の経験を活用して改善案を作成

(3) MISRA-Cの解読

・新しい概念は少なかったが、理解度のレベルアップに努めた

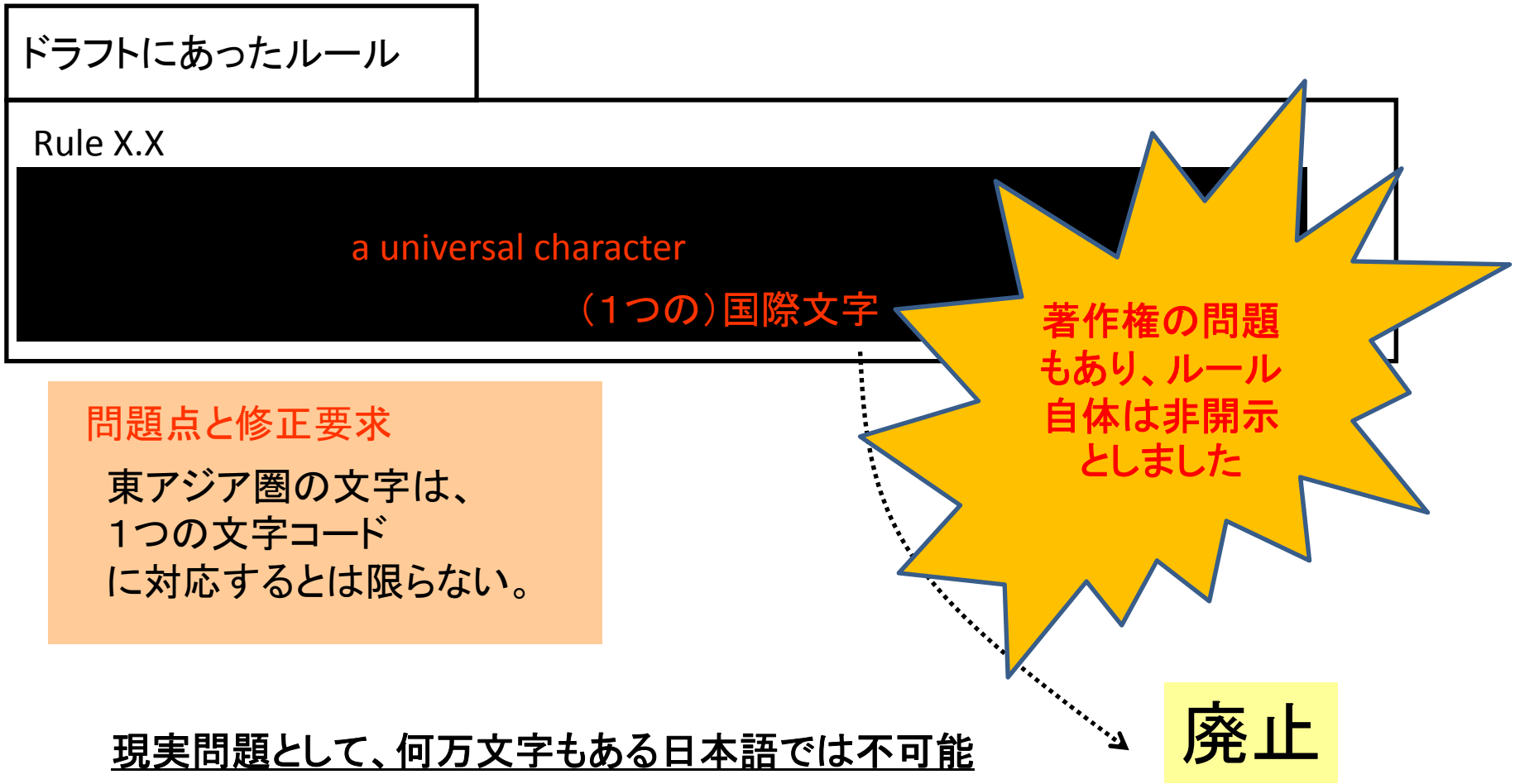
(4) MISRA-C解説書の発行

・更なる改善努力とともに、初めて電子書籍での発行に挑戦

少数精鋭で、過去の解説書を凌ぐものを作ろうと奮闘！

参3．ルール成立前の修正要求

【事前活動による貢献】
研究会メンバーの脇田さんが、MISRA-C:2012の事前レビューに参加し、日本語には不向きなルールの問題点を指摘して、出版前に「廃止」にしてくれた。



現実問題として、何万文字もある日本語では不可能

５．MISRA-C解説書ができるまで

MISRA-C研究会の最大の成果物は、何と言っても「MISRA-C解説書」である。

・その解説書はどうやって作成されてきたの？
・今回の解説書の特徴は何？
・解説書作成にあたってどういうことで苦労してきた？
・研究会活動はメンバーに何をもたらすの？

などについてご説明したいと思います。

まずは、研究会各メンバーの（自己）紹介から・・

5-1. 研究会メンバー紹介 #1

(1) リーダー

・小川　清（名古屋市工業研究所）

工学博士。 規格の専門家であるが電気工学や情報工学にも造詣が深く、「ソフト開発技術者は、１年目からコンパイラを書ける実力が必要である」が持論です。
経済学部の学位も持っている変わり者で、「がんばろう東北」を発信中。

(2) C言語規格の専門家

・川尻　智士（元ルネサスエレクトロニクス）

コンパイラの開発に従事してきたのでC言語規約に詳しく、１期・２期活動の技術面での支柱的存在。 残念ながら3期途中で事情があって涙を飲みながら離脱。

・鹿目　稔（スパンション・イノベイツ）

同様にコンパイラの開発を担当しておりC言語規約に詳しい。 何事にも真面目に取り組み、技術面でどんな変なことを聞いてもちゃんと答えてくれる貴重な存在。

・中村　さおり（ルネサスシステムデザイン）

本人曰くコンパイラ職人。 入社してすぐにコーディングした製品に未だに関わっていることが密かな喜び。 いつも多忙で、週末になったり連休に入ったりした途端に抱えていた仕事が上がってくるような、頑張るママさんエンジニア（今は管理職）。

5-1. 研究会メンバー紹介 #2

(3) 組み込みソフトウェア開発者

・脇田　貴文（矢崎総業）

会社では、組込み系とIT土方系を担当している１期からの古株メンバー。 しかし、古株の中ではまだ若い方なので議事録を書いたりしてくれている。 現場の開発者としては規約にも詳しく、理論と実践のバランスが非常に良く取れている。

・伊藤　久美子（日立オートモティブ）

同様に１期から参加している掛け替えのないメンバーの一人。 会社では、組込み関係の部署にて開発環境などを担当している。 C3では、ツール対応への改善も取り入れられているので、その面でもとても頼りになっている女性エンジニア。

・宇野　結（元パナソニック）

１期から参加のもう一人。 コンパイラの構文解を利用したツール開発などを担当してきて、現在はC言語だけでなくリファクタリング・モデリングなどの講師を務める才女。 筆者の小中高大学のすべての後輩で、何かといじめている。（逆かも？）

・築城　一宏（古河電工）

２期から参加の中堅どころ。 入社当時は表面科学などを担当していたが、途中で自動車関係に異動して、ECU・センサの開発から自社の教育・プロセス見直しまでを担当している。 その明るい性格から、暗くなりがち（？）な研究会を救っている。

5-1. 研究会メンバー紹介 #3

・吉川　直希（トヨタテクニカルデベロップメント）

２期から参加している自動車メーカに最も近い立場の一人。 元々はマイコン関係の会社で基本ソフトの開発を行っていたが、現在は自動車関係のソフト品質などを担当。 昨年生まれたとってもかわいい赤ちゃんにべったりの良きパパさん。

・服部　智幸（東海ソフト）

エンベッデッドという部署で（その名の通りの）組込み関連の開発に従事。 主に、新規顧客向けの新規業務と新人教育を担当している「新しいもの」好き。 研究会には初参加であるが、独学で身につけたMISRA-Cに関する知見は優れている。

・片山　貴晴（サンテック）

自販機向けの組込みソフトの開発を担当している貴重な非自動車分野の技術者。入社以来ソフト開発一筋で、研究会に入るまでMISRA-Cのことを知らなかったにも関わらず猛勉強で貢献してくれている。 一見真面目で、実も真面目。（きっと）

・森川　聡久（ヴィッツ）

機能安全の開発支援やそのコンサルが主な業務の機能安全のスペシャリスト。研究会においても、トレーサビリティにうるさい。（ウソです） MISRA-Cを身につけたことで、機能安全のコンサルに磨きがかかることは間違いなしです。（本当です）

5-1. 研究会メンバー紹介 #4

・尾仲　洋和（サニー技研）

半導体テスタやマイコン開発支援ツールの開発から、車載ソフトウェアの開発まで何でもこなします。 真面目でおとなしそうに見えましたが、実は関西出身でたまに関西人らしさが顔を出し、2次会でも合宿でも何でも付き合ってくれます。

・後藤　文康（アイシン・コムクルーズ）

車載セキュリティの導入やプロセス改善が主な担当業務で、研究会の忘年会幹事というとても重要な責務も担っています。 立派な体格に見合った広い心の持ち主で、大変な業務も笑顔で乗り越えてくれます。（忘年会のことではありません）

・池田　元三（デンソー）

一番態度がでかいのに、実は一番C言語プログラミングから離れたところにいます。最後にコードを書いたのは25年以上前という有様なので。 それでも、メンバーをいじめまくっているのは、解説書を少しでも良くしたいからです。（と信じたい・・）

(4) 静的解析ツール開発者

・宮野　学（東陽テクニカ）

研究会をずっと支えてきた東陽テクニカ期待の若手の星です。 どんどん若返りが図れる会社は、老体にむち打つような筆者の会社からはうらやましい限りです。
技術に秀でているだけでなく折衝事も引き受けてくれて、本当に不可欠な存在です。

5-2. 解説書作成のプロセス #1

(1) 方針決め

いきなり取りかかるのではなく、まずは方針を明確にした。

① 読者はC言語経験１年以上を想定し、初級者にも分かり易い内容にするとともに、熟練者にも価値あるものとする
② MISRA-C:2012の2004からの変更点を理解して、それに見合った解説書の構成にする
③ すでにかなり定着していると思われる MISRA-C:2004 との相違点をできるだけ明確に示すようにする
④ 不明点はMISRAに確認し、MISRAの趣旨を正しく理解した上で研究会独自の考え方を加えるようにする
⑤ MISRA翻訳に対して、明らかな誤訳や専門語や日本語として不適切な表現をできる限り改善して適用する
（MISRAにも、翻訳版に反映してもらうよう働きかける）
⑥ 解説書出版の早期化および改訂の容易さに加え、持ち運びの便利さを考慮し従来の紙での発行に先んじて電子書籍での発行も検討する

5-2. 解説書作成のプロセス #2

(2) 2004解説書との違い検討

読み慣れた2004解説書をどう発展させてより良いものにするかの検討を行った。

① ルールが、「ルール本文」と「補足（Amplification）」に「例外（Exception）」の３点セットになっているので、その３点を一箇所にまとめてルールを理解し易いようにした。（注：「補足」と「例外」がないルールもある）

Rule 8.3（必要）
オブジェクトまたは関数の宣言はすべて、同じ名前と型修飾子を使わなければならない。

C90[未定義 10]、C99[未定義 14]、[コーニグ 59-62] 【適用対象】C90/C99 【分類】決定可能, システム
【補足】 記憶域クラス指定子は、このルールを適用しない。また、関数へのポインタの宣言は、関数定義と異なっていても、このルールの違反にならない。
【例外】 同じ基本型の互換性のある記述は、相互に用いてもよい。例えば、int、signed および signed int はすべて、同等である。

5-2. 解説書作成のプロセス #3

(2) 2004解説書との違い

② 2004解説書の「潜在的問題」は、解説で説明されていることと内容が重複していることも多く、また全体的に文章が長い傾向があった。 そこで、短い文章でそのルールの意義を説明することを目的とした「ねらい」を新規に追加した。

Rule 3.2（必要）
//コメント内では、行接合を用いてはならない [C99]

参考例

【ねらい】
//コメントの次の行が意図せずにコメントになってしまうことを防止する。

③ MISRA-C:2004に準拠した実装が行われている場合に、どう対応すればいいかを明確にするために、「MISRA-C:2004との関係」を新規に追加した。

参考例

【MISRA C:2004との関係】
2004のRule 9.3.
2004では列挙子の値を指定しないか、列挙子の値を全て指定するかの何れかの方法のみを許したが、2012では暗黙的な列挙子の値が他と重複しなければ、部分的な値を指定可能とした。

5-2. 解説書作成のプロセス #4

(3) MISRA-Cの理解と解説の作成

前章（21P）、Directives（16P）、Rules（141P）、References/Appendix（49P）を15人で分担した。 Ruleは大分類で22あるので、1人が2～3個ずつの大分類を受け持つことになった。（全159ルール）

【手順】

① 各人が担当ルールを詳細まで理解して解説書を書く
② 月例の研究会MTGを開催し、3人1組のチームで各解説書をレビューする
③ そのレビューを全チームが完了するまで継続する
④ 完了後、ルールの割り当てとチーム編成を変更して同じことを行う
⑤ そのサイクルを3回廻して、解説書のレビューを完了
⑥ その後、一部メンバーによる合宿での議論により全体での整合性改善を開始
（例えば「逸脱」がルール全体で同じような内容・トーンになるように修正。
また、遅れていた前章やReference（Appendix）の仕上げにも着手）
⑦ 出版が電子書籍化に決定したことを受け、その課題対応も実施
⑧ 全体での研究会活動を7/31に完了し、一部メンバーの最終校正を実施中
早く出版して打ち上げを行おう！

参4. 研究会活動のプロセス

方針決め

MISRA-Cの
理解と解説作成

グループレビュー

全員でレビュー

見直し1回目

悩みながら

最後の見直し

たまには2人で

合宿

各人最終
見直し

最終会議

最終レビュー
と校正

完成 !!

電子（Kindle）版解説書が完成！

かんぱーい！

5-2. 解説書作成のプロセス #5

(4) MISRA-C解説書の作成

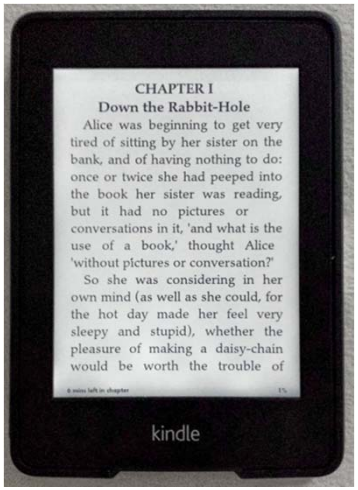
2012版は、電子書籍 ”Kindle” にて発行することが決定。

【利点】

① 発刊までの時間が短縮できる
② 改訂が容易で、読者の負担なし（無料で）更新が可能である
③ （現時点では）日本語の単語で検索ができる
④ 書籍を持ち運ぶよりは軽くて容易に持ち運べる
⑤ 文字の拡大・縮小が可能である（老眼にも優しい）
⑥ 暗い場所でも読むことができる

【欠点】

⑦ 書籍では容易な、違う場所（頁）との行き来が簡単ではない
⑧ 一冊の解説書を複数の人で共用することができない
⑨ PCとかタブレットがないと読むことができない
⑩ 紙ではない（本好きには残念・・）



5-3. 解説書の読み方#1

MISRA-C:2012の解説書には、以下に述べるような工夫がしてあります。 是非、その点に留意して読んで下さい。

(1) 前章

① MISRA-C:2012（本体）と同じ章立てにしながらも、翻訳には終わらずに研究会ならではのメッセージを加えた

② 分かりやすくするために本体の6章の内容は他の章に織り込んだが、章番号がずれないように、6章はコラム欄として「研究会各位の思いを、少しくだけた形で読者に伝える」ようにした

(2) ねらい

③ 当該ルールの説明を行うのではなく、ルールの存在意義として「どういうリスクを防ごうとしているか」を中心に、さらにそれを簡潔に書くように心がけた

④ 従来は「可読性の向上」で済ませていたところを、なるべく「可読性」という言葉を使わずに、読み違えたら何が起こる？というところまで書くように努めた

参5. コラム

【コラムの例】 研究会メンバーの思いを伝える

1. コラム1：決定可能（Decidable）と決定不可能（Undecidable）

今回、「決定可能」と「決定不可能」という用語が使われるようになりました。 これは一見すると、「決定可能＝ガイドラインに非適合な場所がツールにてすべて検出可能」「決定不可能＝非適合がツールでは検出不可能」というように解釈してしまいそうですが、必ずしもそういう訳ではありません。

・・・

従って大切なことは、適当にコーディングして後からツールで非適合を見つけて修正するという方法ではなく、ガイドラインを正しく理解して適合するようなコーディングをした上で念のためにツールをかけて見るという方法が望ましいということです。

MISRA-Cの解説からもう少し踏み込んだ話題が中心です

2. コラム2：int あ;

近年、上記のようなプログラムを受け付けつけるコンパイラが出てきています。なかには絵文字を利用できてしまうコンパイラも存在します。C99以降、識別子に処理系依存文字を利用することが認められたことから、「int あ;」は、ISO-Cとして許容される記法となっています。

・・・

MISRA-Cは特定の国や地域のためのガイドラインではないので、日本特有の問題については十分に議論されてはいません。だから、日本語の識別子を明確に禁止するようなルールは不足しています。そう考えるべきでしょう。 ・・・

参6. ねらい

【ねらいの例】 本当のリスクを見極める

Rule 2.1（必要） プロジェクトは、到達しないコードを含んではならない

このルールの目的は、一見すると次のように思われる。
「（到達しなくても動作上問題がない）不要なコードを見つけ出して削除する」

でも、不要なコードがあることがそんなに問題なのだろうか？　それを削除することが本当に重要なことなのだろうか？　そのためにわざわざルールを作った？

これは、Fail Safe の処理などではあるかも

そこで逆のことを考えてみた。
「一見不要なコードって本当に不要なのだろうか？」
「ひょっとしたら必要なのに、誤って到達しないようになったのでは？」

すなわちこのルールの価値は、「ツールで到達しないコードを見つけ出して必要なのに到達（実行）しないコードになっていることを気づかせる」

ということで、ねらいは
「必要なコードが誤って実行されなくなってしまうことを防止する」となった。

5-3. 解説書の読み方#2

(3) 解説本文

① MISRA-C本体や翻訳版を読むことなく、この解説書だけを所有する読者のことを考え、本文自体の説明やC言語規約の解説までも織り込むことに配慮した（「未規定である」で切り捨てたりしないように努めた）

② 読者層として想定した初級者向けに解説をしているが、熟練者が読んだとしても「へー」と思ってもらえることを目指して技術的に正確であることを重要視した

(4) サンプル

③ 第2版（C2）に引き続いて、本体に載っているサンプル以外のサンプルコードを付加して、ルールの理解が更に進むように配慮しただけでなく、適合/非適合の理由と、非適合になった時にリスクにまで言及している

④ 机上の空論にならないように、ありそうなサンプルコードになるように努力するとともに、実際にC90とC99のコンパイラをかけてみて、ちゃんと動く（または、ちゃんとエラーが出る）ことをすべて確認した

参7. 解説文

【解説文の例】 ルールの心を説明する

Rule 13.2（必要） 式の値と持続的な副作用は、すべての許可された評価順序のもとで同じでなければならない

【解説】

C言語規格は、副作用完了点から副作用完了点の間での式の評価の順序および副作用の生じる順序を規定していない（未規定）。 たとえば「x + (y++) 」という式においてxを評価する前にyをインクリメントするのかxを評価した後にインクリメントするかはコンパイラに委ねられている。 このことはほとんどの場合問題ないが、たとえば「y + (y++) 」のような式においては問題を引き起こす。 左オペランドのyが評価される前に右オペランドyがインクリメントされるか、左オペランドのyが評価された後にインクリメントされるかによって式の値が異なる。このような式「y + (y++) 」の値は未定義として扱われ、コンパイルエラーとならないことがある。式の記述を Rule 13.2 に適合させることによって、このような記述を避けることができる。

例えば、「**式の演算結果が未定義だからダメ**」からという紋切り型の説明ではなく、初級者でも理解できるように、「ねらい」で簡単に述べたことをていねいに説明するような努力がなされている。

参8. サンプル

【サンプルの例】 ルールの理解を助ける

サンプルコードには、適合例と非適合例を示しているが、それだけではなくルールを理解しやすくなるような事例とその説明をできるだけ付加している

Rule 13.2（必要）【補足】1. オブジェクトは複数回変更してはならない

【サンプル】

x = x = 0; // 非適合： 注2（Rule 13.4 にも非適合）

注2）完全式内でオブジェクト x を2回変更しているので補足1に非適合となる。x の値は 0 と一意に決まるが、そうであれば「 x = 0; 」でよい。この記述は誤りの可能性がある。

ツールで検出されれば、誤りに気が付く可能性が高まる

x = x + 1; // 適合： 注3

注3）オブジェクト x に対して変更と読み取りの両方を行っているが、代入演算子 = の右オペランドのオブジェクト x の値の読み取りは x 内に保存される値の計算に必要なので適合となる。たとえば元の x の値がゼロであればこの式の値および x の値は 1 となり、一意に決まる。

適合、非適合だけでは済ますことなく、その理由やさらにリスクも示すような努力もなされています。 上の例では、「意味のないコードを書くな」ではなく、「こんな変なコードは誤っているのは？ 調べてみましょう」という意図があるルールだと。

5-3. 解説書の読み方#3

(5) 逸脱

① 第2版（C2）では「こう記述すれば、本ルールからの逸脱である」という項目を設けたが、今回は「逸脱の可能性とその際の処置」について言及している

② また、（特に推奨ルールについては）この「逸脱」の項目に記載しないで解説文の中にそういう記述を行うことで、そのルールの理解を助長するようにもした（例：なぜ推奨になっているかの理由を、サンプルと合わせて説明している）

(6) References（Appendix）

③ MISRA-C本体の9章 References（参考文献）を10章にして、研究会独自の略語一覧も付加し、Appendix（付属書）を9章にした

④ 本体および解説書内で使用されている用語は、本体で示されているもの以外にも研究会で必要と判断したものは付加した

参９．逸脱

【逸脱の例】 ルールを逸脱する事例や処置方法を伝える

Rule 2.1（必要）
プロジェクトは、到達しないコードを含んではならない

【逸脱】
プログラマは，防衛的プログラムを実現するために、Cソース上に記述された論理からは、到達しないことが明らかなソースコードを記述する事がある。
例えば、ノイズによるRAM化けが発生し、設計値と異なる異常値を検する条件（switch文のdefault節や，if文の条件式)を設計した場合、Cソース上に記述された論理からは、**異常値検出ロジックは常に未到達なロジックになる**。
このような場合に限り、プログラマは逸脱の手続きをしなければならない。
このルールの逸脱の手続きとして、**以下に示す内容をプログラマは確認**しなければならない。
(a) 防衛的プログラミングを意図して設計されたソースコードであること
(b) ソースコードの箇所を特定できている事
(c)(b)で特定した箇所が防衛的プログラミングとして意図通り動作すること

安全のために必要な逸脱の事例を提示

参10. 用語・略語一覧

【用語の例】 解説書の理解を助ける

10.1 略語一覧

研究会が独自で略語一覧を付記

ABI: Application binary interface（アプリケーションバイナリインターフェイス）
AC: AutoCode（オートコード、自動コード）
AGC: Auto Generated Code（自動生成コード）
AMD: Amendments（追補、補遺）
ANSI: American National Standard Institute（アメリカ規格協会）
BSI: British Standard Institute（イギリス規格協会）
C2011:ISO/IEC 9899:2011 Programming Language C, MISRA-C:2012の対象外
C90:ISO/IEC 9899:1990 Programming Language C, ANSI-C:1989と同等
・・・
MIRA: Motor Industry Research Association（自動車産業調査会）
MISRA：Motor Industry Software Reliability Association
（自動車産業ソフトウェア信頼性協会）
MISRA掲示板:MISRA-UKが主催するMISRA C等の英語の掲示板
・・・

5-4. 研究会活動について#1

MISRA-C研究会の活動はやりがいがあるとともに大変でした。
何がどう大変で、何に喜びを見いだしたかを報告します。

(1) 大変だったところ

① ２期に比べるとメンバーは半減となったので、１人１人の負荷は大きくなった
② 基本的にMISRA-Cに関係しそうな会社の寄り合い所帯で、さらにボランティア的な活動なため、メンバーの業務が多忙になったり都合がつかなくなると、担当の分野の研究会業務が滞ったり会議に出席できなくなったりした
③ 翻訳ができるまで、また翻訳ができても（誤訳や、あまり良くない訳が多かったので）原文の英語を読む機会が多く、正しい理解の妨げとなった
④ 他のメンバーが書かれた解説を引き継いだ場合などで、その解説を正しく理解することが難しいことがあった（他社の人が苦労して書いたものに対して、「これはダメでしょう」なんて気軽に言えないし・・）
⑤ 知識・経験や会社の考え方の違いから、ある人の常識と他の人の常識に違いがあって、万人に分かりやすい説明をすることが非常に難しいと痛感した
⑥ 多忙な中でも何とか時間を捻出するために自宅で作業をする人も多かったが、睡眠不足になったり、座椅子に長時間座っていて腰痛になったりしてしまった

5-4. 研究会活動について#2

(2) 楽しかったところ

① コストとか戦略的なことを抜きにした、純粋に技術的なことだけを議論することができた（現場にいる人は自分自身の業務への反映を考えながら、現場を離れた人は昔を思い出したり部下の業務のことなどを考えながら・・）

② ルールの意図が良く分からなかったりした場合に、自分で調べて考え抜いたり専門家の方から意見をもらったりしてそれが解明できた時に、大きな達成感を感じることができた

③ 時には技術論を激しく戦わせることがあっても、その場を離れた席では本当になごかやに会話を楽しむことができた（忘年会や合宿は楽しかった・・）

④ 一番は、何と言っても通常では面識を持つ可能性も低い方々（例えば、競合会社の人とか）とお付き合いする機会を持てたこと

研究会活動は、最高でした!!

参11. 合宿の風景

6. Summary

1. 2年間に渡った研究会活動におけるメンバーの思いはただ一つでした。 それは「MISRA-Cを理解したいと考える人々に最も優れた解説書を届けたい」ということです。

   本当に最高かどうかは分かりませんが、メンバーの全員が限られた時間の中で全力を尽くしたことだけは間違いないと思っています。

2. 今回は電子書籍という形で皆さんに解説書を届けることになりましたので、読んで頂いてご意見やご要望などがあれば遠慮なく連絡して下さい。（改訂を検討しますので）

   今後も継続してレベルアップに取り組んでいきます！

参12. 研究会のデータ

① メンバー
合計：20名、常時活動：15名

工事中です・・

② 活動期間
2013年7月～現在（2年1ヶ月～）

③ 月例研究会
2013年7月～2015年7月　合計24回、1000±100人時

④ 合計工数
8000±2000人時

⑤ 合宿
2015年4月16日～17日、参加7名

⑥ 忘年会
2013年12月20日 & 2014年12月19日、参加15名以上