Part5.

信頼性向上に向けたシステム開発共通リファレンス

本報告書 Part5 は高信頼システム実装に関する共通開発指針に関するWGでの検討結果を整理したものである。なお、この WG は下記のメンバにより構成されている。

WG メンバ名簿

主査	野中 誠	東洋大学 経営学部 経営学科 准教授
	池 秀典	東京海上日動システムズ㈱ 商品ソリューション本部 火新・海上ソリューションサービス部 ソリューションプロデューサ
	一柳 幹男	信金中央金庫 理事 システム部長
	太田 忠雄	㈱ジャステック 常務執行役員 営業本部長
	沼田 克彦	東京電力㈱ システム企画部 新配電システムプロジェクトグループマネジャー
	長谷川 潔	東日本旅客鉄道㈱ IT・Suica 事業本部 IT・Suica技術部 課長 セキュリティグループ
	広瀬 雅行	㈱東京証券取引所グループ IT 企画部長
	淵 昌彦	東京ガス㈱ 防災・供給部 施設 G マネジャー
	若宮 正男	三菱スペース・ソフトウェア㈱ 技術推進部 技術管理グループ グループマネジャ

（五十音順）

事務局

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

WG 開催記録

第 1 回	2008 年 12 月 8 日(月) 14:00～17:00	(独)情報処理推進機構「会議室A」
第 2 回	2009 年 1 月 19 日(月) 14:00～17:00	(独)情報処理推進機構「会議室A」
第 3 回	2009 年 2 月 16 日(月) 14:00～17:00	経済省本館 17 階東4 第5共用会議室

１．はじめに

ここでは、まず、重要インフラ情報システムとはどのようなものかを説明する。次に、共通リファレンスの概要について説明し、その対象とする範囲を示す。続いて、共通リファレンスを提示することの狙い、期待される効果、および共通リファレンスを検討するにあたって考慮した事項を説明する。

重要インフラ情報システムとは

重要インフラとは、内閣官房情報セキュリティセンター（NISC）が作成した「重要インフラの情報セキュリティ対策に係る第 2 次行動計画（案)」によると、「他に代替することが著しく困難なサービスを提供する事業が形成する国民生活及び社会経済活動の基盤であり、その機能が停止、低下又は利用不可能な状態に陥った場合に、わが国の国民生活又は社会経済活動に多大なる影響を及ぼすおそれが生じるもの」と定義されている（NISC, 2008)。また、同行動計画（案）では、重要インフラ事業における重要サービス（国民生活や社会経済活動に与える影響が大きいために特に防護すべきサービス）を提供する情報システムを、重要システムと呼んでいる。本報告書における「重要インフラ情報システム」とは、同行動計画（案）における「重要システム」と同義であると考えている。

同行動計画（案）では、情報通信、金融、航空、鉄道、電力、ガス、政府・行政サービス、医療、水道、および物流の 10 分野を、防護対象の重要インフラとして指定している。また、これらの分野において重要度の高い情報システムとして、例えば、勘定系システム（金融)、運行システム（航空)、運転監視システム（電力)、プラント制御システム（ガス）などが挙げられている（その他の例は 0 節で提示)。これらは、本報告書が想定する重要インフラ情報システムの例である。

共通リファレンスについて

本報告書では、メトリクスとその参考目標値を称して共通リファレンスと呼ぶ。共通リファレンスという名称の意図は、重要インフラ情報システムの種類によらず、共通して参照されるべき項目であると考えているためである。

メトリクスとは、プロセスまたはソフトウェア成果物のある特性を数値で表現するために、その特性に関する属性を測定する測定方法と、測定結果を表現する際に用いる尺度を定義したものである。文脈の違いによって、測定量、尺度、指標などさまざまな名称で呼ばれるが、本報告書ではすべてメトリクスで統一している。

参照目標値とは、メトリクスに目標値を設定する際の参考値として、共通リファレンスで提示する値である。メトリクスを用いる際には、組織の実績などに基づいて目標値を設定し、実績値と比較しながら定量的コントロールを行う。この目標値を初めて設定するときなどに、参考目標値が利用されることを想定している。共通リファレンスでは、システムが社会に与える影響度の度合いによって、異なる参考目標値を提示する。これは、『組込

みソフトウェア開発向け品質作り込みガイド（以下、ESQR ガイド)』(IPA/SEC, 2008）で示された考え方であり、共通リファレンスでもこの考え方を踏襲する。

本報告書で示したメトリクスは、共通リファレンスの具体的イメージを明確化するために例示したものであり、確定した内容ではない。今後の検討において、メトリクスの追加・変更・削除が行われる予定である。その際には、測定のしやすさ、導入のしやすさ、および実効性を重視してメトリクスを検討する。

なお、本報告書では具体的な参考目標値を提示していない。その代わりに、参考目標値を今後検討する際に役立つと思われる情報を参考情報として提示した。

共通リファレンスの範囲

ここでは、共通リファレンスが対象とする範囲を、ライフサイクルプロセスの観点、測定対象とするソフトウェア特性の観点、および測定を実施するタスクの観点から規定する。これらは本報告書の時点での範囲であり、今後の検討において変わる可能性がある。

(1) 対象とするライフサイクルプロセス

共通リファレンスが扱うライフサイクルプロセスの範囲は、重要インフラ情報システムの企画プロセスを出発点として、ベンダから重要インフラ事業者（以下、事業者）にソフトウェアが引き渡され、情報システムが運用可能となる時点までとする。具体的には、『共通フレーム 2007』(IPA/SEC, 2007）における次のプロセスを対象とする。

- **企画プロセス**
- **要件定義プロセス**
- **開発プロセス**
- **運用プロセス（運用を開始する前のアクティビティまで）**

(2) 測定対象のソフトウェア特性：欠陥に着目

測定対象のソフトウェア特性として、欠陥に関わる特性に着目する。重要インフラ情報システムのソフトウェア特性のうち、定量的コントロールの対象とすべき特性は、欠陥に関わる特性以外にもさまざまなものが考えられる。しかし、共通リファレンスでは、ソフトウェア信頼性に着目しているため、直接的な結びつきのある欠陥に関わる特性を主に扱う。

ここで、欠陥とは、プログラムの実行コード中に含まれる欠陥（バグ）のみを対象とするのではない。システム化構想企画書や要件定義書など、システム化構想から運用準備にいたるライフサイクルプロセスで作成される各種の仕様書に含まれる誤りも、欠陥として扱う。

『ESQR ガイド』では、測定の容易さを重視し、欠陥に関する特性は測定対象として取り上げていなかった。一方、共通リファレンスでは、ライフサイクルプロセスを通じた欠

陥の作り込み・除去状況の把握に基づいた定量的品質コントロールを重視する。そのため、本報告書で示す通り、レビューやテストにおける欠陥摘出数に基づくメトリクスを新たに追加している。

なお、欠陥には、ソフトウェア要件から最終プロダクトに至るメインストリームの成果物における欠陥と、それらの成果物を評価するために作成される副次的成果物（テストケースなど）における欠陥がある。本報告書では、これらの副次的成果物の扱いは言及していない。しかし、ソフトウェア信頼性の向上を考えると、これらの副次的成果物の欠陥もコントロール対象とすべきである。副次的成果物の欠陥については、今後の検討課題とする。

(3) 測定対象のタスク

前項で挙げたライフサイクルプロセスのうち、メトリクスを用いて測定を行うタスクは、表 5－1 に示したタスクを想定している。これらのタスクは、『共通フレーム 2007』で提示されたアクティビティにおいて、重要インフラ情報システムのソフトウェア信頼性向上に大きく寄与すると考えられるタスクである。なお、各タスクには、便宜上、本報告書における独自の ID を付した。

ビジネスニーズの観点からすれば、どのようなシステムを開発するかの決定や、システム投資効果の算定など、表 5－1 のタスク以外にも重要なタスクが存在する。これらのタスクも、定量的コントロールの俎上に載せる必要があるかもしれないが、共通リファレンスではソフトウェア信頼性向上に直接的に結びつくタスクを取り上げることを優先し、表 5－1 のタスクを対象とした。

表 5－1 測定対象のタスク

ID	タスク	タスクが含まれるアクティビティ(プロセス)
SyCR	システム化構想レビュー	1.4.2 システム化構想の立案（1.4 企画プロセス）
SHRR	利害関係者要件レビュー	1.5.3 利害関係者要件の確認（1.5 要件定義プロセス）
SysRR	システム要件レビュー	1.6.2 システム要件定義（1.6 開発プロセス）
SysADR	システム方式設計レビュー	1.6.3 システム方式設計（1.6 開発プロセス）
SwRR	ソフトウェア要件レビュー	1.6.4 ソフトウェア要件定義（1.6 開発プロセス）
SwADR	ソフトウェア方式設計レビュー	1.6.5 ソフトウェア方式設計（1.6 開発プロセス）
SwDDR	ソフトウェア詳細設計レビュー	1.6.6 ソフトウェア詳細設計（1.6 開発プロセス）
SwCR	ソフトウェアコードレビュー	1.6.7 ソフトウェアコード作成及びテスト（1.6 開発プロセス）
SwIT	ソフトウェア結合テストの実施	1.6.8 ソフトウェア結合（1.6 開発プロセス）
SwQT	ソフトウェア適格性確認テストの実施	1.6.9 ソフトウェア適格性確認テスト（1.6 開発プロセス）
SysIT	システム結合テストの実施	1.6.10 システム結合（1.6 開発プロセス）
SysQT	システム適格性確認テストの実施	1.6.11 システム適格性確認テスト（1.6 開発プロセス）
OT	運用テストの実施	1.7.2 運用テスト（1.7 運用プロセス）

※ 表中の項番は、『共通フレーム 2007』の項番を意味する。

共通リファレンスを提示する狙いと期待効果

共通リファレンスを提示する狙いは、定量的品質コントロールメカニズム導入と、システム全体の視点による品質評価の促進である。

(1) 定量的品質コントロールメカニズムの導入促進

共通リファレンス、すなわちメトリクスと参考目標値を提示する狙いは、局所的には、重要インフラ情報システムのソフトウェア開発に定量的品質コントロールメカニズムの導入を促進することである。これにより、次に挙げる効果が期待される。

- 目標値と実績値の比較に基づいたプロジェクト管理の実施
- 品質作り込み活動(設計作業など)および品質評価(レビューやテスト)活動の適正な実施
- 目標値を設定し、その達成状況を確認しながらプロジェクトを進めるという意識の定着化
- システム障害の根本原因分析と、それに基づく再発防止に向けたプロセス改善の実施
- 手戻り工数の減少による開発期間の短期化および開発コストの削減
- 要求されるソフトウェア信頼性水準の達成

また、具体的な参考目標値を提示することで、メトリクスおよび目標値に関する産業界

での議論を誘発し、それがかえって定量的品質コントロールへと向かう道筋になることを期待している。参考目標値はあくまで例示としての値であり、絶対的な基準ではない。また、参考目標値は、定量的品質コントロールを通じてソフトウェアの信頼性を向上させるための手段であり、参考目標値を達成することが目的ではない。エンジニアリングの観点からすれば、十分な根拠のない数値を参考目標値として提示することを懸念する向きもあるだろうが、組織を定量的品質コントロールへとドライブすることの重要性を優先し、参考目標値を意図的に提示するアプローチをとる。運用上は、組織における過去の実績に基づいた妥当な目標値を設定する必要がある。共通リファレンスが示す参照目標値は、その議論の出発点として参考にしてもらえればと考えている。

(2) システム全体の視点による品質評価の促進

共通リファレンスが対象とするライフサイクルプロセスおよび測定対象のタスクは、ベンダが実施するプロセス（タスク）だけではなく、事業者が実施するプロセス（タスク）も含んでいる。その意図は、事業者、ベンダ、さらには協力会社も含めたソフトウェア開発プロセス全体における品質の測定を通じて、システム全体としての適正な品質評価を促進することにある。

例えば、システム要件の誤りが原因でシステム障害が発生した際に、この欠陥に関する責任の所在を事業者側にするのかベンダ側にするのか、その決定は、実務的には契約でコントロールする場合が多い。この場合、一般には責任の明確化とその補償に関心が向けられるが、共通リファレンスにおける考え方はそれとは異なり、システム全体の視点による品質測定および評価の適正化に焦点を当てている。そのため、ベンダだけでなく事業者が実施するタスク（例えば、システム化構想レビュー、利害関係者要件レビューなど）での測定を求めている。

共通リファレンス検討にあたっての考慮点

共通リファレンスを検討する過程で考慮した事項について、以下に述べる。

(1) 信頼性確保のための具体的方策の扱い

信頼性を確保するための具体的方策の例示は、共通リファレンスの範囲外とする。例えば、システムを二重化・三重化する具体的な方法や、フェイルセーフ機能の作り込み例、安全性確保の方策などアイデアはあるだろうが、これらの内容は共通リファレンスには含めない。

ただし、信頼性確保の方策がシステム要件またはソフトウェア要件として定義された以降は、それらの要求事項は定量的コントロールの対象となる。定義された要求事項がシステムライフサイクルを通じて漏れなく実現されていることをフォローするために、メトリクスを用いて評価する。

(2) 信頼性確保に必要なコストの扱い

信頼性確保に必要なコストは、プロジェクトプロファイリングにおける調整の範囲として扱い、システムプロファイリングでは扱わない。例えば、信頼性を確保するためにシステム構成の冗長度を高めたり、ソフトウェア要求を多く取り入れたりすることで、信頼性確保に必要なコストは増加する。これらのコスト要因は、提供側の論理で決まる要因であり、システム障害が生じたときにユーザが受ける影響度合いとは基本的に独立である。したがってこれらの要因はシステムプロファイリングではなく、プロジェクトプロファイリングとして扱う。

信頼性確保のためのソフトウェア要求が増加するのであれば、前項(1)で述べた内容と同一の扱いとなる。一方、ソフトウェア要求は増加しないが、システム構成が複雑になることでレビューのチェックリストやテストケースが増加するために、通常よりも多くの開発コストを必要とする場合がある。このような影響をプロジェクトプロファイリングで扱うことで、メトリクスの目標値を高く設定でき、すなわち、高い目標値を達成するために多くの工数が必要であるという関係性を示すことができる。

(3) プロセスの質および実施内容の扱い

プロセスの質や内容を考慮せずに、メトリクスとその目標値だけで定量的コントロールを実施するのには限界があるのではないかとの指摘がある。これは正しい指摘であり、重要なことではあるが、共通リファレンスでは、プロセスの質や実施内容を担保するための方策は対象外とする。これらの担保は、扱うとすれば、エンジニアリング対策の項目で触れることにする。

プロセス遵守率などのメトリクスを定義し、定量的コントロールの俎上に載せることもできなくはない。しかし、共通リファレンスでは、成果物の測定結果からプロセスの質を評価するというアプローチをとる。したがって、プロセス遵守率などの測定により、プロセスの質や実施内容を測定することはしない。

２．重要インフラ情報システムのソフトウェア開発を取り巻く課題と解決アプローチ

ここでは、重要インフラ情報システムおよびそのソフトウェア開発を取り巻く課題として、本報告書が認識しているものを挙げる。また、それらの課題を解決するアプローチとして、本報告書において採用する方法論の概要を説明する。

課題認識

重要インフラ情報システムのシステム障害事例は、残念ながら皆無ではない。これは、国内外を問わず、歴史的に見ても同じである。しかし、近年国内において特に目立つ傾向として、システム障害が発生したという事象のみが過大にクローズアップされて、その現実的な影響度に関わらず過剰な批判が事業者およびベンダに寄せられることが多い。また、事業者およびベンダにおいて、重要インフラ情報システムのソフトウェア信頼性を十分なレベルで確保できていない、あるいは確保する組織体制が確立していない場合もある。重要インフラ情報システムの信頼性に対するソフトウェア信頼性の占める割合が高まっている今日の状況を考慮すると、要求されるソフトウェア信頼性水準を確実に達成する組織体制の確立が急務である。

このような背景を踏まえた上で、本報告書では、重要インフラ情報システムおよびそのソフトウェア開発に関する課題として、次の 6 点を指摘する。

1. 要求されるソフトウェア信頼性水準に対する社会的共通認識の欠如
2. システム障害がもたらす影響度に対する社会的共通認識の欠如
3. ソフトウェア信頼性の目標値に対する事業者・ベンダ間の合意形成手段の欠如
4. 定量的品質コントロールメカニズムの欠如
5. ソフトウェア開発のダイナミクスに適応した開発マネジメントの欠如
6. 要求されたソフトウェア信頼性水準とコストのバランスをとる方法論の欠如

以下、それぞれについて説明する。

課題1.　ゼロ欠陥のソフトウェア開発は実務的に不可能であることについて、社会的共通認識が形成されていない

重要インフラ情報システムにおいてシステム障害が生じると、その事業者およびベンダに対して、社会から過剰な批判が寄せられることが多い。その際の国民の反応は、多くの場合、「システム障害は皆無であるべき」という前提に基づいている。しかし一方で、情報システムの開発・運用に関わる実務者は、「ゼロ欠陥のソフトウェアを開発することは実務的に不可能」という事実を認識している。コストを無尽蔵に投入したとしても、ソフトウェアという高度に複雑な論理的構築物という性質上、完全なゼロ欠陥のソフトウェアを開発することは極めて困難である。このように、ソフトウェアの信頼性に対する社会的認識のギャップが存在する。

課題2.　システム障害がもたらす影響度について、社会的共通認識が形成されていない

「ゼロ欠陥は実務的に不可能である」という事実が社会的に共通認識されたとしても、続く課題として、システム障害が国民に与える影響度について共通認識が得られていない、また、共通認識を形成する手段が存在しないという課題がある。重要インフラ情報システムにおいてシステム障害が発生すると、一般に、その個別のシステム障害（インシデント）によって直接的に影響を受けた国民の数、直接的な経済的損失、事業者が計上した直接的な損失額などが報道により伝えられる。しかし、これらの情報は個別のシステム障害に依存した内容であり、潜在的には、より大きな影響がもたらされる可能性がある。重要インフラ情報システムのシステム障害による影響度を評価する場合、その重要性の高さから、もっとも被害が大きい場合における影響度を評価すべきと考える。また、事業者側のビジネス判断に基づく損失額（例えばリコールや補償など）は、国民に与える影響度の度合いとは直接的には無関係であるため、これらは切り分けて評価する必要がある。

また、個別のシステム障害についても、その影響度の度合いを分かりやすい尺度で評価できる仕組みがあるとよい。そのような仕組みがあれば、個別のシステム障害について、国民と事業者およびベンダ間でシステムリスクに対するコミュニケーションが促進され、ひいては重要インフラ情報システムの信頼性確保に対する困難さの理解に結びつくことが期待される。

システム障害の影響度を評価する手段は、国民だけでなく、事業者およびベンダにも便益がある。一般に、事業者とベンダがソフトウェア開発契約を締結する際に、システム障害の影響度は、各社独自の方法で見積もられている。共通利用できる影響度評価の手段が提供されれば、事業者とベンダ間のコミュニケーションの促進や、複数ベンダによるシステム提案の比較・検討、ベンダにおける開発計画の策定などにも役立てることが期待される。

課題3.　ソフトウェア信頼性の目標値について、事業者とベンダで合意形成する体系化された方法がない

課題 1 で述べた通り、ゼロ欠陥の達成は実務的に不可能である。そのため、事業者とベンダが開発計画を検討する際には、ソフトウェア信頼性の目標値としてどの水準を設定すべきか決めなければならない。しかし、これを設定する体系化された方法はこれまでに提供されておらず、事業者とベンダのあいだでアドホックに、あるいは組織固有の基準に基づいて取り決められており、一貫性がない。

課題4.　定量的品質コントロールのメカニズムがソフトウェア開発プロセスに適切に組み込まれていない

ソフトウェア信頼性の目標値が設定された場合に、ライフサイクルプロセスを通じて、

この目標値を効率的かつ効果的に達成する定量的品質コントロールの仕組みを構築できている組織は必ずしも多くない。ライフサイクルプロセスの終盤になって信頼性を評価したところ、これが目標値を下回っていることに初めて気がつくような組織には、重要インフラ情報システムのソフトウェア開発を委託するのは心許ない。ライフサイクルプロセスの早い段階からフィードバックをかけて、信頼性の目標値を確実に達成するための戦略的なアプローチが求められる。

ソフトウェア業界では、近年、定量的品質コントロールに対する関心が高まっている。多くのメトリクスを導入し、測定プログラムを導入しているソフトウェア組織も多い。しかし、多くの場合、測定した結果をうまく活用できておらず、適切な定量的品質コントロールに結びつけられていない。最小の測定で最大の効果が出せるような、定量的品質コントロールの実践に向けたガイドが求められる。

課題5.　仕様変更や規模拡張など、ソフトウェア開発のダイナミクスに適応した開発マネジメントが行えていない

課題 4 に関連して、特に、ソフトウェアの仕様変更が生じた場合、手戻りが生じた場合、開発規模がプロジェクトの進行に応じて変化した場合など、ソフトウェア開発におけるダイナミクスの適切なコントロールが行えていない。定量的品質コントロールを実現するためのメトリクスがこれまでに多数提唱され、実務の場において適用されているが、ソフトウェア開発におけるダイナミクスの問題への対処は不十分である。仕様変更が生じた場合には、一般的に、ソフトウェア影響波及解析の実施、回帰テストの実施、インタフェース分析の実施などが求められるが、これらの対策を適切かつ確実に実施することは容易ではない。

重要インフラ情報システムに限らず、ソフトウェアの欠陥は、仕様変更や手戻りが発生した際に混入しやすい。そのため、ソフトウェア開発のダイナミクスに適応した定量的品質コントロールおよび開発マネジメントの実施が求められる。

課題6.　要求されたソフトウェア信頼性水準とコストのバランスをとる体系的な方法論がない

課題 3 に関連して、重要インフラ情報システムといえども、そのソフトウェア開発は事業者およびベンダにとってはビジネス活動であり、コストを無尽蔵にかけられるものではない。要求されたソフトウェア信頼性水準を達成するために、ライフサイクルプロセスにおいてコストを重点投下すべき対象を特定し、要求水準を達成しつつコストの適正化を図る方法論が求められる。

なお、これは、システム障害が社会に与える影響度が極めて甚大であるにも関わらず、達成すべきソフトウェア信頼性水準を事業者とベンダの交渉により下げてもよい、という考えではない。ここで述べているのは、あくまで、要求水準の達成戦略に関する議論である。

課題解決のアプローチ

以上に挙げた課題に対して、本報告書では、

1. 効率的・効果的な定量的品質コントロールメカニズムの導入
2. システムおよびプロジェクトのプロファイリング
3. メトリクスとその参考目標値の提示
4. エンジニアリング対策の提示

の 4 つのアプローチによって課題解決を図ることを提唱する。これは、『ESQR ガイド』が示した考え方におおむね基づいている。以下に、それぞれのアプローチの概略を説明する。それぞれの詳細は、後続する各章で別途説明する。

アプローチ1.　効率的・効果的な定量的品質コントロールメカニズムの導入

課題 4 および課題 5 を解決するために、効率的かつ効果的な定量的品質コントロールのメカニズムを、事業者およびベンダのライフサイクルプロセスに導入する。課題 4 について、導入の容易さを考慮した上で、実践的な定量的品質コントロールの方法、測定したデータの具体的な活用方法、および定量的品質コントロールメカニズムの組織への導入指針について提示する。課題 5 について、仕様変更や規模拡張に対して定量的品質コントロールを適用する方法について提示する。

なお、課題 5 に対応した事項は、本報告書では言及しておらず、今後の検討課題である。

アプローチ2.　システムおよびプロジェクトのプロファイリング

課題 1、課題 2、課題 3、および課題 6 を解決するために、『ESQR ガイド』の考え方に基づき、システムおよびプロジェクトのプロファイリングを導入する。

システムプロファイリングとは、ソフトウェアが原因でシステム障害が発生したときに、その障害がもたらす人的および経済的損失をシステム利用者の視点から算定し、その結果に基づいて対象システムをタイプ分けする作業である（IPA/SEC, 2008）。『ESQR ガイド』では Type-1（Normal）から Type-4（Highly Critical）までの 4 タイプを与えており、各タイプにはそれぞれ異なる品質目標値の参考値が与えられている。重要インフラ情報システムは、その性質上、Type-3（Critical）または Type-4 のいずれかに分類されるものとする。

プロジェクトプロファイリングとは、プロジェクトの特性を開発側の視点から評価する作業である（IPA/SEC, 2008）。『ESQR ガイド』では 10 項目のファクターを挙げており、それぞれのファクターを評定することにより、±10 の補正係数が得られる。得られた補正係数は、システムプロファイリングで得られた品質目標値の参考値の補正に用いられる。

課題 1 について、アプローチ 4 と併せて提示することにより、重要インフラ情報システムのソフトウェア信頼性の現実的な目標値はゼロ欠陥ではないということを国民に伝える。

課題 2 および課題 3 について、システムプロファイリング手法により、システム障害がもたらす影響度に対する社会的共通認識と、影響度に応じた目標とするソフトウェア信頼性水準の合意形成を目指す。その際に、『ESQR ガイド』で示された方法を重要インフラ情報システムに適合するよう調整する。課題 6 について、プロジェクトプロファイリング手法により、開発側の論理により調整可能な範囲でのコストバランスをとる方法を提供する。また、『ESQR ガイド』p.25 が示す通り、ソフトウェアを構成する異なる部分に対してそれぞれ別個にシステムプロファイリングを行うことで、コストを重点投資すべき部分とそうでない部分を切り分け、ソフトウェアシステム全体として要求される信頼性水準を損ねることなく、コストバランスをとる方法を提示する。

アプローチ3.　メトリクスとその参考目標値の提示

課題 4 および課題 5 を解決するために、具体的なメトリクスとその参考目標値を提示する。メトリクスは、一般に、プロセスを評価するメトリクスと、プロダクト（成果物）を評価するメトリクスに分けられる。ここでは、測定のしやすさと、定量的コントロールの有効性を考慮して、プロセスおよびプロダクト評価メトリクスを選定する。すでに『ESQR ガイド』において、定量的品質コントロールのためのプロセスおよびプロダクト評価メトリクスがいくつか提示されている。共通リファレンスでは、ESQR のメトリクスはほぼそのまま取り入れる予定である。これらに加えて、欠陥に関する特性を評価するメトリクスを新たに追加する。

システムプロファイリングの結果、各メトリクスにはシステムタイプによって異なる目標値が設定される。また、その値は、プロジェクトプロファイリングによって補正される。これは『ESQR ガイド』が提示した考え方であり、共通リファレンスにおいてもその考え方を踏襲する。

アプローチ4.　エンジニアリング対策の提示

課題 4 の解決を円滑に行うため、エンジニアリング対策を提示する。エンジニアリング対策とは、定量的コントロールを効果的に実施するための具体的かつ現実的な施策であり、実践上のガイドである。これには、定量的コントロールという枠から離れて、高信頼性を達成するために役立つノウハウも含める。本報告書の範囲では、これまでの議論を通じて出された意見のうちいくつかを例示するにとどめる。

３．定量的品質コントロールメカニズムの導入

定量的コントロールには、プロジェクト内でのコントロール（プロジェクトマネジメント）と、プロセス改善を通じたコントロール（プログラムマネジメント）の二段階がある。図 5－1 に、定量的コントロールの概念図を示す。図 5－1 には例としてソフトウェア要求レビューに着目しているが、測定対象は表 5－1 に示したすべてのタスクである。

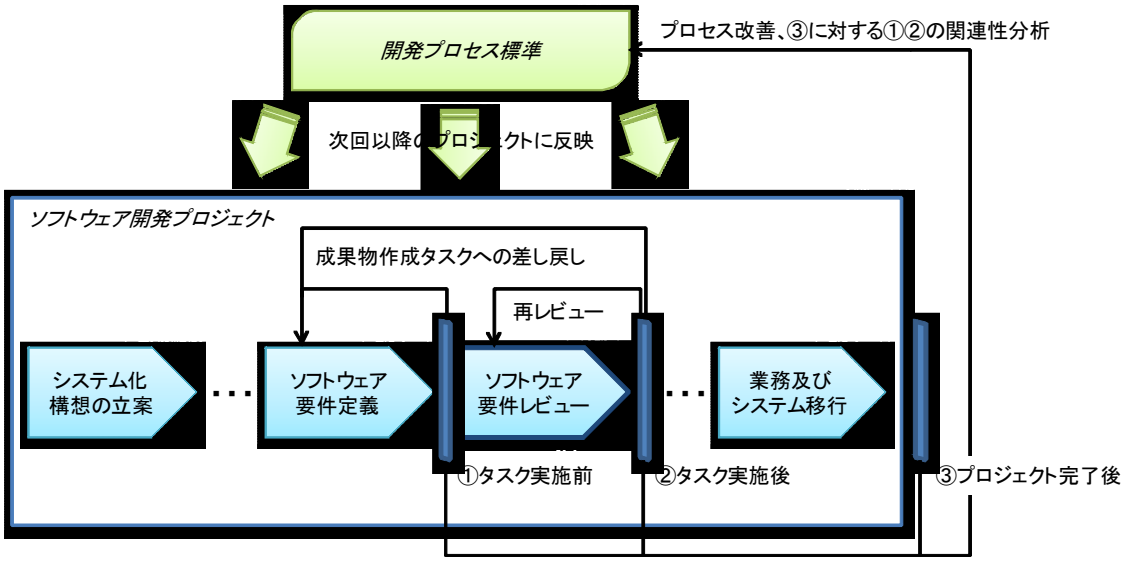

図 5－1　定量的コントロールの概念図（ソフトウェア要求レビューに着目した場合）

図 5－1 では、一連のタスク群からなるソフトウェア開発プロジェクトにおいて、メトリクスの測定結果と目標値の比較により、他のタスクやプロセス標準へとフィードバックする様子を模式的に表している。主なフィードバックは、成果物作成タスクへの差し戻し、レビューまたはテストの再実施、および開発プロセス標準の改善である。短期的にはプロジェクトマネジメントレベルでのフィードバックが重要であり、長期的にはプログラムマネジメントレベルでのフィードバックが重要である。

メトリクスを用いた測定のタイミング、またはあるタスクに対する測定値が最終的に確定するタイミングには、①タスク実施前、②タスク実施後（または実施中）、③プロジェクト完了後の 3 つがある。それぞれの時点で測定した結果を目標値と比較し、定量的品質コントロールを行う。

例として、ソフトウェア要件レビューというタスクに着目した場合を考える。このタスク実施前に測定した結果が目標値（組織で基準として設定した値）を達成していない場合は、達成できなかった原因を分析した上で、成果物作成タスク（ソフトウェア要件定義）へと成果物を差し戻して成果物の品質向上を働きかけたり、あるいは以降の工程で早期に問題の解消を図ったりするなどの対策が求められる。一方、タスク実施後に測定した結果が目標値と大きく乖離していた場合は、同様に、成果物作成タスクへと差し戻したり、成

果物の評価タスク（ソフトウェア要件レビュー）を再実施したりするなど、適切な措置を講じる必要がある。

メトリクスには、プロジェクトが完了した時点、または運用後一定期間（一般的には一年間）が経過した時点にならなければ測定値が確定しないものがある。例えば、欠陥除去率など、出荷後に発見される欠陥の測定を要するメトリクスがこれに相当する。多くの場合、このようなメトリクスで表される特性こそが、事業者やベンダが定量的に把握したい内容である。これらのメトリクスの目標値を達成するには、関連性がある代替メトリクスを用いて定量的品質コントロールを実施する必要がある。すなわち、プロジェクト完了後に測定値が確定するメトリクスと、タスク実施前および実施後のメトリクスの関連性（相関関係や因果関係）を分析し、その結果に基づいてタスク実施前および実施後のメトリクスの目標値を設定する。こうすることで、プロジェクトの早い段階から品質を確保し、結果として、コントロール対象の特性を達成するというアプローチが求められる。

例えば、システム結合テストにおける欠陥密度について、ある一定の目標値以下に抑えたいとする。ここで、欠陥密度とレビュー工数比率に相関関係が確認できたとする。この関係を根拠として、欠陥密度をある一定以下に抑えるために投入すべきレビュー工数の目標値を設定し、次回以降のプロジェクトでの定量的コントロールに結びつける。

あるいは、プロジェクト完了後に測定値が確定するメトリクスであっても、プロジェクトの途中で測定を行い、その値の収束状況を監視することで定量的品質コントロールを行うアプローチを採用してもよい。

このような分析の結果は、当該プロジェクトのコントロールだけではなく、組織の開発プロセス標準へとフィードバックし、プロセス改善に結びつける必要がある。その結果は、次回以降のプロジェクトに対して確実に反映し、継続的改善に向けて取り組む必要がある。この地道な取り組みこそが、要求されたソフトウェア信頼性水準を確実に達成するための近道である。

４．システムおよびプロジェクトのプロファイリング

ここでは、『ESQR』ガイドで示されたシステムプロファイリングおよびプロジェクトプロファイリングを元に、重要インフラ情報システムのプロファイリングに適用する際の考慮点や変更点について説明する。

システムプロファイリングの方法

重要インフラ情報システムにおけるシステムプロファイリングは、『ESQR ガイド』が示すシステムプロファイリングの基準（表５－1）をそのまま適用する。ただし、重要インフラ情報システムとして分類されるのは、システムタイプの Type-3（Critical）または Type-4（Highly Critical）のいずれかとする。すなわち、経済損失が 10 億円未満と算定されるものは、重要インフラ情報システムに分類しない。

表5－1 ESQR システムプロファイリングの基準（Type-3 および Type-4）

システムタイプ	人的損失	経済損失
Type-4 (Highly Critical)	極めて甚大な人的損失	極めて甚大な経済損失
Type-3 (Critical)	人的損失あり	10 億円以上の経済損失

Type-3 と Type-4 にシステムタイプを限定した場合、表５－1 の通り、人的損失または経済損失において「極めて甚大な」損失があるか否かによってタイプ分けされる。本質的に、タイプ分けには絶対的な基準があるわけではなく、社会的に共通認識が得られるタイプ分けができればよいという任意性がある。とはいえ、「極めて甚大」であるか否かという基準はやや漠然としており、タイプ分けの再現性を損ねるリスクが高い。そこで、定性的ながらも、少しでも客観性を向上させるタイプ分けの基準が求められる。

ひとつの方法として、『ESQR ガイド』pp.47-50 に示されている「システム障害に関する影響評価スケール ST-SEISMIC」（表 5－2）を用いてタイプ分けする方法が考えられる。ST-SEISMIC は、もともと個別のシステム障害（インシデント）における影響評価スケールとして提唱されたものであり、タイプ分けの道具として意図されたものではない。しかし、重要インフラ情報システムの潜在的システム障害の中でもっとも悪いケースに対して ST-SEISMIC を適用し、タイプ分けをするというアプローチは妥当であると考える。その際には、ST-SEISMIC では「ユーザ操作への影響」の階級 6 および 7 が空欄となっているので（表 5－2 の太枠部分）、ここに適切な基準を設ける必要がある。

重要インフラ情報システムのシステムタイプ分け例

NISC が作成した「重要インフラの情報セキュリティ対策に係る第 2 次行動計画（案）」では、重要インフラ情報システムの例として 10 分野の 31 システムを挙げている（NISC, 2008）。表 5－3 は、これら 31 システムのタイプ分け例である。このタイプ分け例は、シ

表 5－2　ST-SEISMIC（『ESQR ガイド』p.49 より抜粋引用、一部加筆）

階級		ユーザ操作への影響	ユーザに対する健康被害の影響	ユーザに対する経済的被害の状況	システム周辺の環境への影響	システムタイプ
4	中	ユーザの目的が一部達成できなくなる	一部のユーザに健康被害がでる	一部のサービスが停止し、ユーザの一部で経済損が出る可能性がある		Type-3 (Critical)
5	強	ユーザの目的が一部達成できなくなる	一部のユーザに重大な健康被害がでる	一部または全部のサービスが停止し、ユーザの一部で多大な経済損が出る可能性がある	システムの不具合が限定的ながら、社会的不安を引き起こす、又は利便性を損なう	
6	烈		多くのユーザに重大な健康被害がでる	一部または全部のサービスが停止し、多くのユーザで多大な経済損が出る可能性がある	システムの不具合が、社会的不安を引き起こす	Type-4 (Highly Critical)
7	激		ほとんどのユーザに重大な健康被害がでる	一部または全部のサービスが停止し、ほとんどのユーザで多大な経済損が出る可能性がある	システムの不具合が、大きな社会的不安を引き起こす	

ステムプロファイリングや ST-SEISMIC などの手順を適用した結果ではないので、参考情報として捉えていただきたい。

表 5－3　重要インフラ情報システムのタイプ分け例

Type-3 (Critical)	Type-4 (Highly Critical)
ネットワークシステム（情報通信）、オペレーションサポートシステム（情報通信）、ニュース・番組制作システム（情報通信）、編成・運行システム（情報通信）、資金証券系システム（金融）、国際系システム（金融）、対外接続系システム（金融）、保険業務システム（金融）、整備システム（航空）、貨物システム（航空）、気象情報システム（航空）、座席予約システム（鉄道）、各府省庁及び地方公共団体の情報システム（政府・行政サービス）、診療録等の管理システム（医療）、集配管理システム（物流）、貨物追跡システム（物流）、倉庫管理システム（物流）	勘定系システム（金融）、証券取引システム（金融）、取引所システム（金融）、運行システム（航空）、予約・搭乗システム（航空）、航空管制システム（航空）、列車運行管理システム（鉄道）、電力管理システム（鉄道）、制御システム（電力）、運転監視システム（電力）、プラント制御システム（ガス）、遠隔監視・制御システム（ガス）、水道施設や水道水の監視システム（水道）、水道施設の制御システム等（水道）

プロファイリングによる目標値の設定方法

メトリクスには、システムタイプおよびプロジェクトの特性を考慮した目標値を設定する。その方法は、『ESQR ガイド』が提示した方法に従い、システムプロファイリングおよびプロジェクトプロファイリングによりメトリクスの目標値を設定する。具体的には、次の手順により目標値を設定する。

第 1 ステップ:システムプロファイリングにより、重要インフラ情報システムのタイプを Type-3 または Type-4 に分ける。各メトリクスに与えられたシステムタイプ別の参考目標値を、目標値の初期値とする。

第 2 ステップ:プロジェクトプロファイリングにより、重要インフラ情報システムのソフトウェア開発プロジェクトの特性を評価し、各メトリクスの目標値の補正係数を算出する。

第 3 ステップ:システムプロファイリングにより得られた目標値の初期値に、プロジェクトプロファイリングにより得られた補正係数を適用し、各メトリクスの目標値の補正係数を算出する。

『ESQR ガイド』では、次に挙げたファクターをプロジェクトプロファイリングの対象としている（IPA/SEC, 2008）。今後の検討において、必要に応じてファクターの見直しをはかる。

1. **ソフトウェア規模**
2. **ソフトウェアの複雑さ**
3. **システム制約条件の厳しさ**
4. **仕様の明確度合い**
5. **再利用するソフトウェアの品質レベル**
6. **開発プロセスの整備度合い**
7. **開発組織の分業化・階層化の度合い**
8. **開発メンバのスキル**
9. **プロジェクトマネージャの経験とスキル**
10. **システム障害時のメーカ側損失額**

５．プロセス評価メトリクス

ここでは、プロセス評価メトリクスの例を示す。0 節で述べた通り、これらのメトリクスは共通リファレンスの具体的イメージを明確化するために例示したものであり、確定した内容ではない。

表 5－4 に、プロセス評価メトリクスの例を示す。これらのうち、作業充当率と作業実施率は、『ESQR ガイド』で提示されたメトリクスである。それ以外は、欠陥に関する特性を表すメトリクスとして、新たに追加した。

表 5－4 プロセス評価メトリクスの例

メトリクス名称	概要
参照成果物レビュー完了率（%）	レビューの際に参照する他の成果物のレビュー実施状況を表す。
欠陥摘出密度（欠陥数／規模）	レビューやテストで摘出した欠陥数を規模で正規化した値。
規模あたりレビュー時間（時間／規模）	レビューに要した時間を規模で正規化した値。
作業充当率	※ ESQR で提示されたプロセス評価メトリクス
作業実施率	※ ESQR で提示されたプロセス評価メトリクス
フェーズ欠陥除去率（%）	レビューやテストなど各フェーズにおける欠陥除去の有効性を表す。
プロセス欠陥除去率（%）	開発プロセス全体での欠陥除去の有効性を表す。
累積欠陥除去率（%）	欠陥除去率の測定を簡便化し、欠陥除去状況の概略を示す。
無欠陥コンポーネント率（%）	タスク開始時における無欠陥ソフトウェアコンポーネントの比率。

表 5－5 に、プロセス評価メトリクスと測定対象タスクとの関連を示す。表 5－5 は、各メトリクスをどのタスクに対して適用し、評価するのかを表している。なお、企画プロセスでは成果物の規模を規定することが容易でないので、規模で正規化するメトリクスは企画プロセスのタスクに適用していない。

以降で、それぞれのメトリクスについて説明する。本報告書の時点では、各メトリクスの説明項目が完全ではない。今後の検討により詳細化する。

(1) 参照成果物レビュー完了率

概要:レビューの際に参照する他の成果物（参照成果物）のレビュー実施状況を表す。
定義:レビューが完了した参照成果物の数 ÷ 参照成果物の合計。
測定値が確定するタイミング:タスク実施前。
参考目標値:システムタイプや測定対象タスクによらず、100%であることが望ましい。
使い方:100%未満の場合は、当該レビューの開始を取りやめて、レビューが完了していない。

表 5－5 プロセス評価メトリクスと測定対象タスクとの対応関係

プロセス評価メトリクス ＼ タスク	SyCR	SHRR	SysRR	SysADR	SwRR	SwADR	SwDDR	SwCR	SwIT	SwQT	SysIT	SysQT	OT
参照成果物レビュー完了率	○	○	○	○	○	○	○	○					
欠陥摘出密度			○	○	○	○	○	○	○	○	○	○	○
規模あたりレビュー時間			○	○	○	○	○	○					
作業充当率	○	○	○	○	○	○	○	○	○	○	○	○	○
作業実施率	○	○	○	○	○	○	○	○	○	○	○	○	○
フェーズ欠陥除去率	○	○	○	○	○	○	○	○	○	○	○	○	○
プロセス欠陥除去率	○	○	○	○	○	○	○	○	○	○	○	○	○
累積欠陥除去率	○	○	○	○	○	○	○	○	○	○	○	○	○
無欠陥コンポーネント率							○	○	○	○	○	○	○

参照成果物のレビューを先に実施する。または、レビューが完了していない参照成果物があるという事実を認識し、参照成果物の不確実性というリスクを認識した上で当該レビューを実施する。

参考情報:ソフトウェアインスペクションにおける有効なプラクティスの 1 つに、「インスペクションで参照する成果物が事前にインスペクションされていない場合、インスペクションの実施を禁止する」が挙げられている（Gilb and Graham, 1993)。

(2) 欠陥摘出密度

概要:レビューやテストで摘出した欠陥数を規模で正規化した値。

定義:指摘した欠陥数(主要な欠陥のみ) ÷ 成果物規模(ドキュメントページ数または LOC)。

測定値が確定するタイミング:タスク実施後。

使い方:欠陥摘出密度が目標値よりも大幅に低い場合は、成果物の品質が平均レベルよりも優れているか、またはレビューやテストで十分な量の欠陥を指摘できていない可能性が高いと考えられる。この場合、レビュープロセスまたはテスト項目の設計プロセスを点検し、適切なレビューやテストが実施されたことを確認し、必要に応じて再レビューしたり、テスト項目を設計し直したりする必要がある。

逆に、欠陥摘出密度が目標値よりも大幅に高い場合は、成果物の品質が平均レベルよりも劣っている可能性が高いと考えられる。その場合、レビューやテストの効率と効果が低下し、投入工数が無駄になるリスクが高まる。したがって、この傾向がレビューまたはテストの早期に確認できた場合は、レビューまたはテストを中断し、成果物を作成者に差し戻し、成果物の品質改善を促す必要がある。

参考情報:Humphrey は、TSP (Team Software Process) における品質ガイドラインとして、主要な欠陥について、要求インスペクションで 0.25 欠陥/要求仕様書ページ、高レ

ベル設計インスペクションで 0.1 欠陥/設計仕様書ページ、コンパイルで 10 欠陥/KLOC 未満、単体テストで 5 欠陥/KLOC 未満、統合テストで 0.5 欠陥/KLOC、システムテストで 0.2 欠陥/KLOC を目標値として提示している（Humphrey, 2000）。

（3）規模あたりレビュー時間

概要・定義：レビューに要した時間を規模で正規化した値。

測定値が確定するタイミング：タスク実施後。

参考情報：Humphrey は、TSP における品質ガイドラインとして、要求インスペクションまたはレビューで 0.5 時間/ページ以上、高レベル設計インスペクションまたはレビューで 0.2 時間/ページ以上（ただし、書式が整ったロジック設計文書に対して）、詳細設計インスペクションまたはレビューで約 40 秒/擬似コード行（ただし、擬似コード 1 行は 3LOC 相当）以上、コードインスペクションまたはレビューで約 20 秒/LOC 以上という数値を示している（Humphrey, 2000）。ただし、元々の数値は一時間あたりに換算した逆数で示されており、例えばコードインスペクションまたはレビューの生産性として 200LOC/時間として示されている。

（4）作業充当率

ESQR では、仕様レビュー作業充当率、設計レビュー作業充当率、コードレビュー作業充当率、テストレビュー作業充当率、テスト作業充当率、レビュー作業充当率の 6 項目を定義している。これらのメトリクスが対象とする作業は、表 5－1 に示した対象タスクと一致していないため、今後、擦り合わせが必要である。

目標値の設定にあたっては、『ESQR ガイド』が示す値に加えて、次の情報も参考になる。Humphrey は、TSP における品質ガイドラインとして、要求定義に対する要求インスペクションの工数比率を 0.25 以上、高レベル設計に対する高レベル設計インスペクションの工数比率を 0.5 以上、コーディングに対する詳細設計の工数比率を 1.0 以上、詳細設計に対する詳細設計レビューの工数比率を 0.5 以上、コーディングに対するコードレビューの工数比率を 0.5 以上という数値を示している（Humphrey, 2000）。

（5）作業実施率

ESQR では、仕様レビュー作業実施率、設計レビュー作業実施率、コードレビュー作業実施率、テストレビュー作業実施率、テスト作業実施率、レビュー作業実施率の 6 項目を定義している。これらのメトリクスが対象とする作業は、表 5－1 に示した対象タスクと一致していないため、今後、擦り合わせが必要である。

（6）フェーズ欠陥除去率

概要：レビューやテストなど各フェーズにおける欠陥除去の有効性を表す。

測定値が確定するタイミング:プロジェクト完了後。

参考情報:欠陥除去率（DRE: Defect Removal Efficiency）は、プロセスの効率性を測定する典型的で重要なメトリクスである（Laird and Brennan, 2006）。

Humphrey は、TSP の品質ガイドラインとして、要求インスペクションで 70%、設計レビューおよびインスペクションで 70%、コードレビューおよびインスペクションで 70%、コンパイルで 50%、単体テストで 90%（ただし、5 欠陥/KLOC 程度の品質の成果物に対して）、結合およびシステムテストで 80%（ただし、1 欠陥/KLOC 程度の品質の成果物に対して）という数値を示している（Humphrey, 2000）。

Jones は、米国平均の値として、要求定義で 77%、設計で 85%、コーディングで 95%という数値を示している（Jones, 1996）。文献には明示されていないが、これらの数値は成果物作成とそのレビューの両方を含めた欠陥除去率であると思われる。

Jones は一方で、既定値としては、使い捨て型プロトタイプで 90%、システムソフトウェア設計で 90%、設計インスペクションで平均 65%（最大 95%）、コードインスペクションで 65%という数値を示している（Jones, 1998）。また、ソフトウェア要求定義に着目した場合、実施プラクティスによって 60%から 97%まで欠陥除去率が変動することを示している（Jones, 1998）。

（7）プロセス欠陥除去率

概要:開発プロセス全体での欠陥除去の有効性を表す。

測定値が確定するタイミング:プロジェクト完了後。

参考情報:Jones は、計画よりも早期に完了したプロジェクトでは、プロジェクト規模が 100FP 未満で 98%、100～1,000FP で 96%、1,000～5,000FP で 94%、5,000FP 以上で 92%という数値を示している（Jones, 1996）。

Humphrey は、TSP の品質ガイドラインとして 99%以上という数値を示している（Humphrey, 2000）。

（8）累積欠陥除去率

概要:欠陥除去率の測定を簡便化し、欠陥除去状況の概略を示す。

測定値が確定するタイミング:プロジェクト完了後。

参考情報:野中らは、国内の CMMI レベル 4 またはレベル 5 相当のソフトウェア企業 5 社から平均的な成功プロジェクト合計 9 件をサンプリングし、コードレビュー終了時から出荷後までの累積欠陥除去率を調べた結果、コードレビュー終了時の平均値が 60.996%（中央値 62.136%）、単体テスト終了時で 84.763%（同 84.500%）、統合テスト終了時で 94.878%（同 99.000%）、出荷時点で 99.854%（同 99.947%）という数値を示している（野中・西, 2007）。ただし、この数値は、コードレビュー以降に摘出された欠陥が対象であり、それ以前の摘出欠陥は数値に含まれていない。

Humphrey は、TSP の品質ガイドラインとして、コンパイルまでで 75%以上、単体テストまでで 85%以上、結合テストまでで 97.5%以上、システムテストまでで 99%以上という数値を示している（Humphrey, 2000）。

（9）無欠陥コンポーネント率

概要：タスク開始時における無欠陥ソフトウェアコンポーネントの比率。

測定値が確定するタイミング：プロジェクト完了後。

参考情報：Humphrey は、TSP の品質ガイドラインとして Percent Defect Free（PDF）というメトリクスを示している（Humphrey, 2000）。これによると、コンパイルで 10%以上、単体テストで 50%以上、統合テストで 70%以上、システムテストで 90%以上という数値を示している。

６．プロダクト評価メトリクス

ここでは、プロダクト評価メトリクスの例を示す。0 章と同様、これらのメトリクスは共通リファレンスの具体的イメージを明確化するために例示したものであり、確定した内容ではない。

表 5－6 に、プロダクト評価メトリクスを示す。これらのうち、欠陥密度以外のメトリクスは、『ESQR ガイド』で提示されたメトリクスである。

表 5－6 プロダクト評価メトリクス

メトリクス名称	概要
欠陥密度（欠陥数／規模）	出荷後に発見された欠陥数を規模で正規化した値。
ドキュメントボリューム率	※ ESQR で提示されたプロダクト評価メトリクス
ドキュメントバランス	※ ESQR で提示されたプロダクト評価メトリクス
ファイル行数	※ ESQR で提示されたプロダクト評価メトリクス
関数の行数	※ ESQR で提示されたプロダクト評価メトリクス
制御文記述率	※ ESQR で提示されたプロダクト評価メトリクス
コメント行記述率	※ ESQR で提示されたプロダクト評価メトリクス
コーディングルール逸脱率	※ ESQR で提示されたプロダクト評価メトリクス
テスト密度	※ ESQR で提示されたプロダクト評価メトリクス
不具合収束率	※ ESQR で提示されたプロダクト評価メトリクス
不具合修正率	※ ESQR で提示されたプロダクト評価メトリクス

(1) 欠陥密度

概要:出荷後に発見された欠陥数を規模で正規化した値。成果物の品質（信頼性）を表す。

測定値が確定するタイミング:プロジェクト完了後。

参考情報:Cusumano らは、出荷後一年間におけるソフトウェアの欠陥密度を国際比較した結果、日本の調査対象プロジェクト 27 件の中央値は 0.020 欠陥/KLOC、アメリカの 31 件では 0.400 欠陥/KLOC、インドの 24 件では 0.263 欠陥/KLOC、ヨーロッパその他地域の 22 件では 0.225 欠陥/KLOC という数値を示している（Cusumano, *et. al.*, 2003）。ちなみに、調査対象の国内企業は、日立製作所、日本 IBM、NTT データ、SRA、松下電器産業、オムロン、富士ゼロックス、オリンパスである。

Jones は、引き渡し時の残存欠陥数/FP について、システムソフトウェアで 0.28 欠陥/FP、軍需ソフトウェアで 0.42 欠陥/FP、情報システムソフトウェアで 1.10 欠陥/FP などの数値を示しており、米国平均では 0.75 欠陥/FP という数値を示している（Jones, 1996）。1FP は C 言語で平均約 130LOC、COBOL で約 90LOC である（Jones, 1998）ことを考えると、これらの数値は、Cusumano らが示した値に比べておよそ 10 倍高い欠陥密度である。

Jones は、CMM レベル別に出荷後ソフトウェアの欠陥密度を示している。これによると、CMM レベル 1 で 7.5 欠陥/KLOC、レベル 2 で 6.24 欠陥/KLOC、レベル 3 で 4.73 欠陥/KLOC、レベル 4 で 2.28 欠陥/KLOC、レベル 5 で 1.05 欠陥/KLOC という数値が示されている（Jones, 2000）。これに対して、Davis と Mullaney は、TSP を適用したプロジェクトについて 0.06 欠陥/KLOC という数値を示している（Davis and Mullaney, 2003）。

７．基本メトリクス

プロセスおよびプロダクト評価メトリクスの値を求めるには、規模や欠陥など、要素レベルでの測定が必要になる。このように、プロセスまたは成果物に関する単一の属性を直接的に測定するメトリクスを、本報告書では基本メトリクス（base metrics）と呼ぶ。これは、『ESQR ガイド』における「基礎指標」と同様である（JIS X 0141 では「基本測定量」と呼んでいる)。本報告書の段階では、基本メトリクスの定義について言及するに至っておらず、今後の検討課題である。

例えば、次のような点について検討する必要がある。

(1) 欠陥の定義

欠陥を測定する際には、故障レベルで 1 件とカウントするのか、成果物の訂正箇所レベルで 1 件とカウントするのか。訂正箇所レベルで測定する場合、1 件の故障またはエラーに対して、仕様書やソースコードの複数箇所を訂正した場合でも、それらを欠陥として逐一カウントすべきか。ソフトウェアエンジニアリングの原則からすれば、欠陥（defect）は、成果物に含まれる不適切な部分であるため、逐一カウントすることが求められる。また、Humphrey の PSP/TSP でも、訂正箇所それぞれを欠陥として個別に記録する方法を採用している。一方で、多くのソフトウェア組織では、故障 1 件に対してバグ票 1 枚を起票し、訂正箇所を逐一記録するといった方法はとっていない。実践面も配慮した定義を提示することが求められる。

また、仕様には明記されていなかったが、暗黙知レベルで当然と考えられる仕様上の問題が原因でシステム障害が発生した場合、これを欠陥としてカウントすべきか。例えば、運用時にオペレータが操作ミスをしたためにシステム障害が生じた場合、どこまでをシステムの欠陥として、どこから先は運用上のミスとするのか。実務的には契約などにより限定して運用するが、何らかの指針を与える必要がある。

(2) 規模の定義

ソフトウェアのソースコード規模を測定する場合、流用元の母体規模も含めて評価するのか、あるいは新規開発規模のみを評価するのか。新規開発規模を測定する場合、母体規模に対する修正・削除も含めて測定するのか。ソースコードは論理行数のみをカウントするのか。論理行数の測定対象にはどの要素を含めて、どの要素を除くのか。仕様書のページ数を測定する際に考慮すべき点は何か。テストの評価をするときの規模メトリクスと、生産性を評価するときの規模メトリクスは異なるものを用いるべきではないか。

これらのほかにも多くの課題があり、理論と実践のバランスがとられた統一的基準を設定するのは難しいが、何らかの指針を与える必要がある。

８．エンジニアリング対策

エンジニアリング対策とは、定量的コントロールを効果的に実施するための具体的かつ現実的な施策であり、実践上のガイドである。また、定量的コントロールという枠から離れて、高信頼性を達成するために役立つノウハウもこれに含める。

例えば、以下のようなエンジニアリング対策について記述・検討する必要がある。

- 定量的品質コントロールを実際に運用するにあたっては、どのようなことが課題となるのか。測定した結果を活用するためには、どのような工夫が求められるのか。定量的品質コントロールを継続的に回して行くには、組織としてどのような取り組みが必要になるのか。
- 欠陥を測定する際には、どのような欠陥管理票を用いると測定が容易になるのか。事業者、ベンダ、および協力会社の欠陥データを集約してシステム全体の品質を評価する際には、どのような工夫が求められるのか。
- インスペクションに慣れていないチームがインスペクションを実施する際には、どのような工夫が必要なのか。インスペクションに不慣れなメンバが実施すると、とかく、議論にムラが生じやすい。インスペクションを効率的に実施し、かつ定量的コントロールに必要なデータを正確に記録するには、インスペクション会議の進行、モデレータのリーダーシップ、記録係の割り当てなど、留意点が多数ある。このような点を提示するとよい。
- 要求事項がソースコードに至るまで確実に反映されていることを追跡するには、トレーサビリティマトリクスの適用が有効である。この方法を適用する際の実践的ノウハウは何か。
- 仕様変更が生じた場合には、影響波及解析やインタフェース分析の実施が有効である。この方法を適用する際の実践的ノウハウは何か。仕様変更が発生したときに実施すべきアクション項目は何か。それらは、チェックリストの形式で提示できないか。

９．今後の検討課題

以下に、今後の検討課題を挙げる。すでに本報告書の中で言及してきた項目もここに含まれる。

(1) 共通リファレンスの範囲の検討

0 節で共通リファレンスの範囲を規定したが、ソフトウェア信頼性に影響するタスクが他にも挙げられる。例えば、システム化計画のレビュー、テストケース作成方法の確認、テストケースの確認、ソフトウェア導入（インストール）計画の確認、ソフトウェア受入れ準備の確認などは、ソフトウェア信頼性に影響するタスクである。これらのタスクを、定量的品質コントロールの俎上に載せ、それぞれメトリクスを定義するか、検討が必要である。

本報告書では、共通リファレンスの範囲として保守プロセスを含めていない。しかし、例えばソフトウェアが保守フェーズに移行した後に機能追加が行われる場合、これは、保守フェーズにおいて開発プロセスが実行されることとほぼ同義であり、母体ソフトウェアに機能追加をする開発プロセスとして見なすこともできる。保守プロセスと開発プロセスの扱いについて明確化する必要がある。とくに、共通リファレンスで提示するメトリクスや、定量的品質コントロールのメカニズムがそのまま適用できるのかどうか、検討が必要である。

また、0 節第 2 項で述べた通り、副次的成果物も定量的品質コントロールの俎上に載せるかどうか、検討が必要である。例えば、テストケースという副次的成果物を評価する場合、その網羅性（テストケースの設計カバレッジ）に着目する必要がある。同様に、レビューチェックリストもプロジェクト間で流用される副次的成果物と見なすことができ、その評価も必要になる。これらの扱いに関する検討が必要である。

(2) システムプロファイリング方法の洗練と、タイプ分けの妥当性評価

0 節で、『ESQR ガイド』の ST-SEISMIC を用いたシステムプロファイリングについて言及した。本報告書ではアイデアを提示したのみであり、この方法の了解性や妥当性について検討が必要である。妥当性評価にあたっては、具体的な重要インフラ情報システムに適用し、0 節で示したようなシステムタイプ分け結果を得て、その分類に対する妥当性を可能な範囲で客観的に評価する必要がある。

一方で、すでに『ESQR ガイド』において、統一的基準として用いられるべきシステムプロファイリング手法が IPA/SEC より提示されている。一貫性を考慮すると、これに対する変更はできるだけ避けるべきである。明らかな誤りがあれば訂正が必要であるが、システムのタイプ分けには絶対的な基準はなく、分類には任意性がある。複数の基準ができることによる混乱を回避する努力が求められる。

(3) プロジェクトプロファイリングのファクターおよび参考目標値の調整方法の検討

0 節で述べたプロジェクトプロファイリングについて、『ESQR ガイド』に示されたファクターで十分と考えられるのか、補正係数の算出方法は妥当か、補正係数を用いた参考目標値の調整方法は適正であるかなど、検討が必要である。これらの妥当性を科学的手法により確認することは容易ではないが、客観性を高める努力は必要である。

(4) メトリクスの検討と、参考目標値の設定

0 章、0 章、および 0 章で述べたメトリクスについて、さらなる検討が必要である。とくに、本報告書では参考目標値を提示していないため、参考情報や事例分析などを通じて値を設定する必要がある。また、プロダクト評価メトリクスについて、サイクロマティック数などのように、ソフトウェアエンジニアリングにおける基本的なメトリクスが他にもある。また、MTBF や MTTR などのように、プロダクトの外部特性を測定する典型的なメトリクスがある。共通リファレンスのポリシーとして、最小のメトリクスで最大の効果を得たいと考えており、バランスを考慮した上での検討が必要である。

(5) ソフトウェア開発のダイナミクスへの対応

本報告書で示したメトリクスは、どちらかというと、ウォーターフォール型開発プロセスモデルに即したプロジェクトでの運用を想定している。しかし実際には、0 節の課題 5 で示した通り、仕様変更や手戻りなど、ソフトウェア開発におけるダイナミクスの不可避であり、これを取り込んだ上での定量的品質コントロールが求められる。ソフトウェア開発のダイナミクスに対してメトリクスをどのように適用するか、今後の検討課題である。

メトリクスによる定量的品質コントロールが適用できない領域については、それに対応したエンジニアリング対策を提示する必要がある。

(6) エンジニアリング対策の提示

本報告書では、エンジニアリング対策の必要性のみを述べており、具体的な対策には言及していない。具体的なエンジニアリング対策を提示する必要がある。

参考文献

[1] Cusumano, M., *et. al.* (2003): Software Development Worldwide: The State of the Practice, *IEEE Software*, Vol.20, No.6, pp.28-34.

[2] Davis, N. and Mullaney, J. (2003): The Team Software Process^{SM} (TSP^{SM}) in Practice: A Summary of Recent Results, CMU/SEI-2000-TR-014.

[3] Gilb, T. and Graham, D. (1993): *Software Inspection*, Addison-Wesley.（伊土誠一・富野壽監訳（1999）『ソフトウェアインスペクション』共立出版）.

[4] Humphrey, W. S. (2000): The Team Software Process^{SM} (TSP^{SM}), CMU/SEI-2000-TR-023.

[5] IPA/SEC（2007）『共通フレーム 2007』オーム社.

[6] IPA/SEC（2008）『組込みソフトウェア開発向け品質作り込みガイド』翔泳社.

[7] Jones, T. C. (1996): *Patterns of Software Systems Failure and Success*, International Thomson Computer Press.（伊土誠一・富野壽監訳（1997）『ソフトウェアの成功と失敗』共立出版）.

[8] Jones, T. C. (1998): *Estimating Software Costs*, McGraw-Hill.（富野壽監訳（2001）『ソフトウェア見積りのすべて―規模・品質・工数・工期の予測法』共立出版.）

[9] Jones, T. C. (2000): *Software Assessments, Benchmarks, and Best Practices*, Addison-Wesley.

[10] Laird, L. M. and Brennan, M. C. (2006): *Software Measurement and Estimation: A Practical Approach*, IEEE Computer Society, Wiley-Interscience.

[11] NISC（内閣官房情報セキュリティセンター）（2008）「重要インフラの情報セキュリティ対策に係る第2次行動計画（案）」http://www.nisc.go.jp/active/infra/keikaku2.html

[12] 野中誠・西康晴（2007）「組織的要因がソフトウェア品質に与える影響」『第 26 回ソフトウェア品質シンポジウム発表報文集』日科技連, pp.195-202.