第１分科会Ａグループ

ソフトウェア成果物の品質評価方法の研究

Research of evaluation method for software products

主査　　小笠原　秀人　㈱ 東芝
副主査　三浦　邦彦　　矢崎総業 ㈱
リーダ　石田　芳昭　　㈱ 野村総合研究所
　　小堺　大亮　　Samsung Electronics Co., LTD.
　　篠田　みゆき　TIS ㈱
　　富本　達明　　三菱電機マイコン機器ソフトウエア株式会社
　　渡邊　泰史　　㈱ NTT データ

1　研究概要

近年、社会的に大きな影響を及ぼすシステム障害が多く発生している。以前に比べて、時代の流れとともにデジタル化が進み、ソフトウェアの規模が大きく膨らみつつあることや、インターネットの普及や携帯電話の進化により、ソフトウェアを使用する機会が増加したことなど、ソフトウェアの重要性が高まってきている。

こうした中、ソフトウェア成果物の品質を確保することは重要なことであり、各社取り組んでいる。しかし、開発段階におけるソフトウェア成果物の品質評価は難しく、確信できる方法で行なっているとは言えない状況にある。

当分科会では、ソフトウェア成果物の品質評価方法に注目し、開発現場で活用できる品質評価方法について、研究をおこなった。

Abstract

In recent years, many system troubles which cause big influence socially have occurred.

The importance of software has been increasing compared with before. The reason is that digitization progressed with the current of the times and the scale of software has swollen greatly, and that the chance to use software increased by the spread of the Internet and the evolution of the cellular phone.

In such situation, it means that it is important that we secure quality of software products, and it works in each company. However, quality evaluation of software products in a development stage is difficult, and it is in the situation not necessarily done by the method that can be convinced.

This subcommittee paid attention to the method of evaluating the quality of the software products, and we researched the method of evaluating the quality that is able

to be used on the development site.

2　テーマ選定の理由

研究会メンバは、品質管理部門のメンバや、開発現場で品質に関心が高いメンバである。

各社で業務を携わる中、次のような疑問を持っている。

ソフトウェア成果物（各種設計書・ソースコード・テスト計画書等）の品質評価が感覚的になっていて、果たして正しい品質評価ができているのだろうか。品質評価を評価しているという根拠を示すための手法・手段が明確になっていないのではないか。品質を定量的に測定できるメトリクスは、果たして正しいものだろうか。

これらを解決し、開発現場で実践的に活用できる品質評価方法を提供することはできないだろうか、と考え、このテーマを選定することとした。

3　活動目標

次のことを当グループの活動目標とした。

1)　品質評価のプロセスを明確にする。

・ISO/IEC14598-1「ソフトウェア製品の評価－第１部：全体的概観」（JIS X0133-1）（以下、ISO14598）を活用して、品質評価プロセスを定義する。

・CMMI「測定と分析」プロセスエリアと比較検討する。

2)　ソフトウェア成果物の品質測定法（①プロセスに着目した評価方法と②プロダクトの品質特性に着目した評価方法)を整理し､開発現場で実践的に活用できる情報を提供する。

①プロセスに着目した品質評価方法（プロセス品質測定法）の整理

・メトリクスを整理し、提供する。

②プロダクトの品質特性に着目した品質評価方法（プロダクト品質測定法）の整理

・ISO/IEC9126 ソフトウェア製品の品質－第１部：品質モデル（JIS X 0129-1）（以下、ISO9126）を理解する。

・共通フレーム 98-SLCP-JCF98（1998 版）（以下、SLCP）アクティビティと品質評価 U 字モデルの関係を整理する。

・SLCP アクティビティと ISO9126 品質特性・品質副特性の関係を整理する。

・品質特性に基づいたレビューポイントを整理する。

4　活動内容

この１年の活動内容を「表-1　活動経過」に示す。

表-1　活動経過

日程	活　動　内　容
平成 17 年 4 月 22 日	研究会メンバ自己紹介と検討課題洗い出し
平成 17 年 6 月 03 日	各社の品質評価の仕組み・メトリクスの紹介。良い点・悪い点などについて意見交換を実施。
平成 17 年 7 月 14 日～15 日	SLCP と各社の工程とのマッピングを実施。当研究の目標・研究したいテーマについて議論

平成 17 年 9 月 20 日	SLCP と照らし合わせた各社品質保証活動状況の紹介。 ISO9126 品質特性・品質副特性の観点からの品質評価方法検討。
平成 17 年 11 月 04 日	品質評価プロセスについての検討。２種類の品質測定法（プロセス品質測定法・プロダクト品質測定法）について検討。 SLCP と ISO9126 品質特性・品質副特性とのマッピング検討。
平成 17 年 12 月 1 日 （臨時会）	品質評価プロセスについての再検討。２種類の品質測定法（プロセス品質測定法・プロダクト品質測定法）について再検討。 SLCP アクティビティと ISO9126 品質特性・品質副特性とのマッピング方針検討。品質特性に基づいたレビューポイント検討。
平成 17 年 12 月 16 日	研究論文全体構成についてのレビューを実施。 SLCP アクティビティと ISO9126 品質特性・品質副特性とのマッピングについてレビュー。
平成 18 年 1 月 13 日	研究論文全体の再レビューを実施。 SLCP アクティビティと ISO9126 品質特性・品質副特性とのマッピングについて再レビュー。
平成 18 年 2 月 24 日	研究発表。

5　研究成果及び考察

5.1 品質評価方法の検討

研究会メンバでソフトウェア成果物の品質を評価するよい方法について、さまざまな意見を出し合い検討をおこなった。各社で実施している事例の紹介を持ち寄り、意見交換をおこない、参考にすべき書籍や国際規格・モデルなどはないか、シンポジウムなどで参考にすべき発表がおこなわれていないか、ホームページ上で参考にすべき内容が掲載されていないか等の調査をおこなった。

検討を重ねた結果、まず、品質評価をするための仕事の流れ（以下、品質評価プロセス）を整理することとした。次に、具体的にソフトウェア成果物の品質を測定する方法を整理することとした。品質を測定する方法として、①プロセスに着目した評価方法（プロセス品質測定法）と②プロダクトの品質特性に着目した評価方法（プロダクト品質測定法）の２種類に分類できるのではないか、という議論になった。

5.2 品質評価プロセスの検討

1) ISO14598 の活用

品質評価プロセスを整理するにあたって、共通の理解を得られるモデルが必要であった。

そこで、ISO14598 で規定されている品質評価の枠組みを前提に、品質評価プロセスを検討し、定義することにした。この ISO14598 シリーズは、ソフトウェア製品の品質に関する規格 ISO9126 シリーズと対を成す規格である。ソフトウェア製品の品質評価プロセスに関するモデルとしては他に、CMMI「測定と分析」や ISO15939(JIS X 0141)があるが、品質を測定する方法で ISO9126 を参考にすることから、研究会メンバは ISO14598 を活用することを選んだ。ISO14598 の評価プロセスに、研究会メンバで検討した①プロセス測定法、②プロダクト測定法を含めた品質評価プロセスを『付図１．ISO14598 を参照した品質評価プロセス』に示す。ソフトウェア成果物の品質評価をおこなうためには目的に見合った測定法

を適切に選択することになる。

2) CMMI「測定と分析」との関係

さらに、ISO14598 を参照した品質評価プロセスについての理解を深めるとともに、更なる有効なプロセスとするために、CMMI「測定と分析」の固有プラクティスとの対比を行った。(『付表１．ISO14598 と CMMI「測定と分析」の対比』参照。)

5.3 品質測定法の検討

1) 品質測定法の考え方

ここでは、研究会メンバが検討したソフトウェア成果物の品質の測定法、①プロセスに着目した評価方法（プロセス品質測定法）と②プロダクトの品質特性に着目した評価方法（プロダクト品質測定法）についての２種類について述べる。

まず、プロセス品質測定法とは、レビューでの指摘事項の割合が適正なのか、テスト件数は規模に対して適正なのか、バグの発生率は規模に対して適正なのか等、レビュー管理・テスト管理といったマネジメントのプロセスに着目した評価をいう。例えば、レビュー工数比率、レビュー指摘密度・テスト密度・バグ密度などがある。

次に、プロダクト品質測定法とは、ISO9126 品質特性・品質副特性に基づく評価をいう。例えば、機能性の合目的性に着目した要件網羅率（要件に対する機能網羅の度合い）などがある。

プロセス品質測定法は、比較的に各社で活用しており、すでに定着した評価方法であると言える。一方、プロダクト品質測定法については、ISO9126 の品質特性はよく知られているが、それを活用した品質評価は定着しているとは言えない状況である。そこで研究会メンバは、プロダクト品質測定法に、より着目し、詳細について調査・検討をすることとした。詳細は、5.5 で述べる。

5.4 プロセス品質測定法の検討

1) プロセス品質測定法

ここでは、プロセス品質測定法について、研究会メンバで検討した結果を述べる。

プロセス品質測定法を実施するうえで、考慮すべき点として、ライフサイクルによって評価方法に違いがあることである。各社で工程定義に相違があることから、ここでは SLCP アクティビティを活用することとした。プロセス品質評価法では、メトリクスを設定し、その値が基準に対して、正常な値なのか、異常な値なのかをみることによって、品質が適切なのか判断することが重要である。SLCP アクティビティ別に、メトリクスや判断基準をまとめたものを、『付表３．プロセス品質測定法によるメトリクス』に示す。

付表３．には、各アクティビティにおけるレビューやテストの実施度、成果物の完成度、品質の確保などの観点から、いろいろな具体的なメトリクスを記述した。メトリクスには、明確な判断基準があるものや、結果から見た大まかな品質傾向を示すものがある。また、プロジェクトの特性によって当てはまらない場合もある。プロジェクトの特性を考慮し、目的にあったメトリクスを抽出していくことが必要である。

5.5 プロダクト品質測定法

1) IS09126 について

ここでは、プロダクト品質測定法について、研究会メンバで検討した結果を述べる。

プロダクト品質測定法については、IS09126 を参考にすることとした。研究会メンバは、IS09126 について詳細を理解していなかったため、規格の理解をおこない、整理した結果をここに記述する。

ソフトウェアのライフサイクルによって、異なる品質の視点がある。IS09126 では、「利用時の品質」「外部品質」「内部品質」として定義されている。その関係を『図１．品質評価 U 字モデル』に示す。

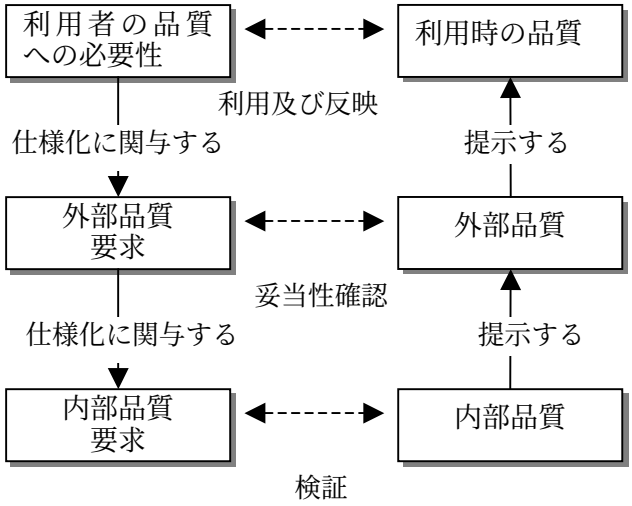


図１．品質評価 U 字モデル

IS09126 では、外部品質および内部品質のための品質モデルが定義されている。品質モデルは、６つの品質特性からなり、それぞれの品質特性は、品質副特性に分割されている。それらを『図２．外部および内部品質モデル』に示す。

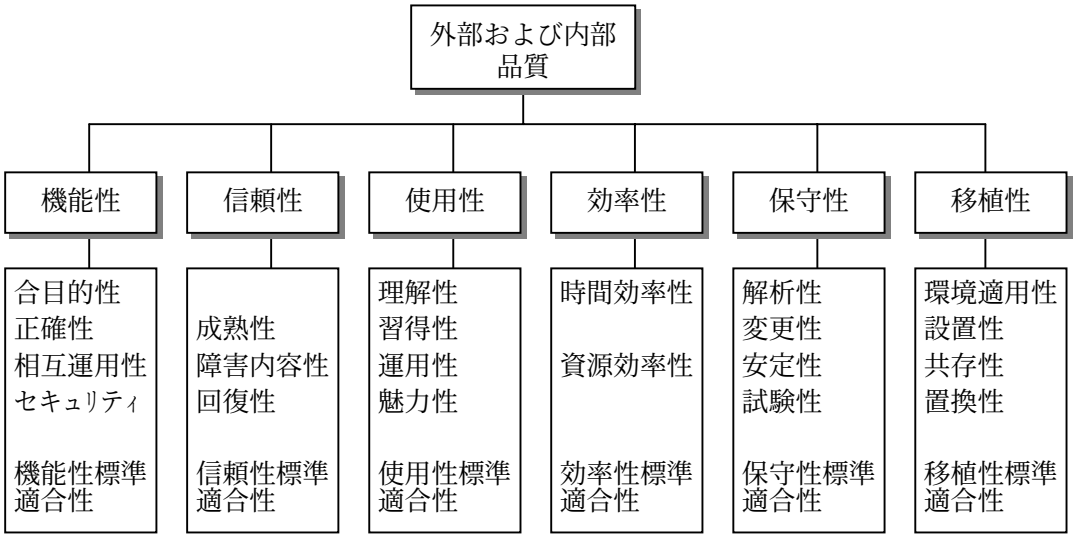


図２．外部および内部品質モデル

「利用時の品質」は、利用者の視点でのソフトウェア製品の品質である。「利用時の品質」は、４つの特性（有効性、生産性、安全性、満足性）に分類されている。

ISO9126 では、これらのことが詳細に定められており、ソフトウェア成果物の品質を評価するうえで非常に重要な存在であることがわかった。

2) SLCP と品質評価 U 字モデルとの関係

次に、前述の品質評価 U 字モデルと SLCP アクティビティがどのような関係になっているか、整理をおこなった。その結果を『図３．SLCP と品質評価 U 字モデルの関係』に示す。

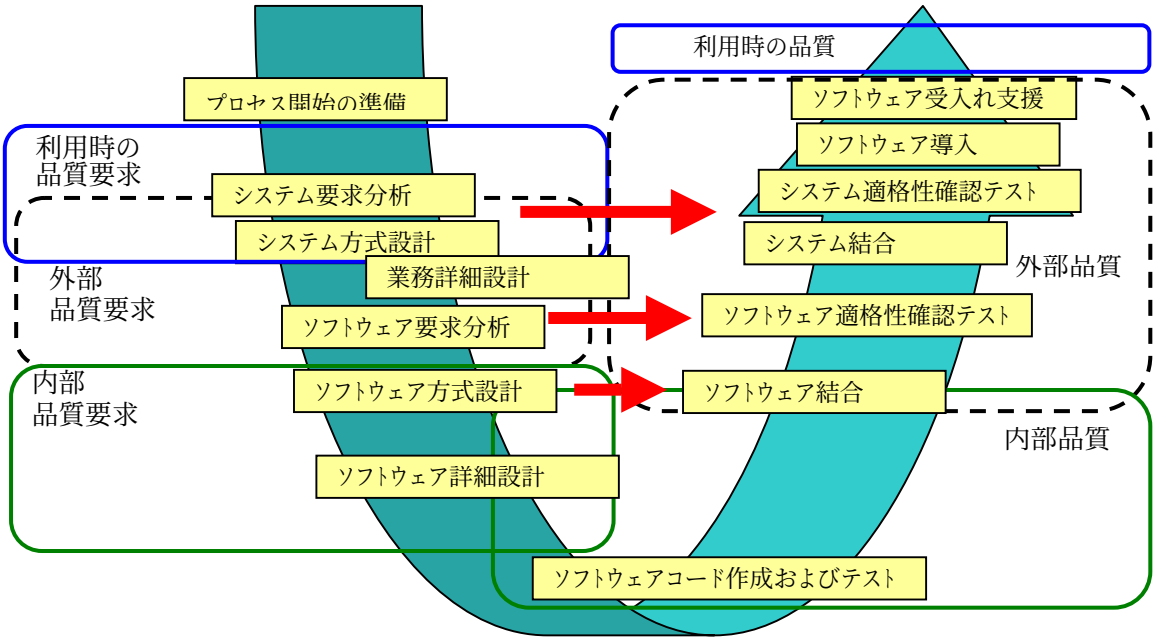


図３．SLCP と品質評価 U 字モデルの関係

この図から、内部品質は比較的限定されたアクティビティ（ソフトウェア方式設計、詳細設計、ソフトウェアコード作成およびテスト）で、作り込みと検証をおこなうが、外部品質には、かなりの多くのアクティビティが関連していることがわかった。また、利用時の品質は、本番運用時の品質の情報が必要になってくると思われる。

3) SLCP アクティビティと品質特性／品質副特性とのマッピング

ここでは、SLCP アクティビティと品質特性／品質副特性との関係について述べる。

研究会メンバにより、「SLCP アクティビティや成果物（実施資料）によって、関係する ISO9126 品質特性・品質副特性に違いがあるのではないか。」ということが議論された。違いを明確にすることにより、ソフトウェア成果物の品質評価を実施する際に、どの SLCP アクティビティで、どの ISO9126 品質特性・品質副特性を考慮すべきなのか明確になり、役立つためである。研究会メンバでは、SLCP アクティビティのレベルでは、ISO9126 品質特性・品質副特性の間の相関において、すべての品質副特性に参照関係が発生するかもしれないが、成果物（実施資料）のレベルなら相関関係の有無や強弱が見出せる（関係する品質副特性があるものや、ないものの存在が確認できる）のでは、と言う仮説を立てて検証を行った。

SLCP アクティビティと ISO9126 品質特性・品質副特性の関係についてまとめた結果を、『付表４.SLCP と品質特性／品質副特性の関係』に示す。

SLCP アクティビティや成果物（実施資料）と ISO9126 品質特性・品質副特性の相関についての結果を述べる。まず、成果物（実施資料）のレベルでは、それほど顕著な相関関係の違いを見出すことはできなかった。次に、一部のアクティビティ（1.4.7.ソフトウェア詳細設計、1.4.8.ソフトウェアコード作成およびテスト等）を除き、ほぼ全てのアクティビティで網羅的に品質副特性を参照していることが確認できた。また、相対する上流と下流（いわゆる V 字モデルの対称のアクティビティ）で、対になっていることが確認できた。

4) 品質特性に基づいたレビューポイント

ここでは、ISO9126 品質特性・品質副特性毎にどのような定性的なチェックをすべきなのか議論し、整理をおこなった。その結果を『付表５．品質特性に着目したレビューポイント』に示す。

実際の開発現場では、このレビューポイントに基づいてチェックし、どこが満足していないかを明確にすることで、品質評価に有効活用できると考えたからである。

5.6 目標達成度の評価

本研究を終えて、目標達成度は以下の通りである。

1) 目標１：品質評価のプロセスを明確にする。

ISO14598 を活用して、また、CMMI「測定と分析」プロセスエリアと比較検討し、品質評価プロセスを定義することができた。

2) 目標２：ソフトウェア成果物の品質測定法（①プロセスに着目した評価方法と②プロダクトの品質特性に着目した評価方法）を整理し、開発現場で実践的に活用できる情報を提供する。

①プロセスに着目した評価方法（プロセス品質測定法）の整理し、メトリクスを提供することができた。

②プロダクトの品質特性に着目した評価方法（プロダクト品質測定法）の整理

ISO9126 の理解、SLCP アクティビティと品質評価 U 字モデルの関係を整理、SLCP アクティビティと ISO9126 品質特性・品質副特性の関係を整理、品質特性に基づいたレビューポイントを整理により、開発現場で実践的に活用できる情報を提供することができた。

5.7 反省と今後の課題

今回の活動について反省点をまとめる。

まず、よかった点として、ISO9126 品質特性・品質副特性のあらためて理解をすることができたことを挙げる。ISO9126 品質特性・品質副特性の理解は非常に難しかったが、SLCP アクティビティとのマッピング、品質特性に基づいたレビューポイントを作成することなどにより、理解することができた。

反省すべき点として、以下を挙げる。

第１に、品質特性に基づいたメトリクスをうまく導き出すことができなかった。もっと早

い時期に ISO9126 品質特性・品質副特性を理解しておけば、メトリクスを導くことができたと思う。

第２に、ソフトウェア成果物の品質に対して、評価の参考になるような手段は提供できたが、客観的評価ができるレベルには至らなかった。

第３に、ISO9126 で記述されている「利用時の品質」について整理することができなかった。

今後の課題として、以下を挙げる。

第１に、ISO9126 品質特性・品質副特性に基づいたメトリクスを導きだすこと。

第２に、SLCP アクティビティと ISO9126 品質特性・品質副特性のマッピングについて、更に強弱をつけるなど改善をおこなうこと。

第３に、今回の成果を社内に持ち帰り具体的に活用すること。

ソフトウェア成果物の品質を客観的に評価することは、重要なことであるが非常に難しいことである。当研究会活動により、それらを解決する手段の一つとして情報提供ができたと思う。今回の研究内容を社内で活用することで、各社のプロセス改善へと導くことができると思う。

6　参考文献

1.「CMMI-SE/SW/IPPD/SS 公式日本語翻訳版　Version 1.1　段階表現」
　独立行政法人 情報処理推進機構、2004

2.「ソフトウェア製品の品質－第１部：品質モデル」
　JIS X 0129-1：2003（ISO/IEC9126-1:2001)、日本規格協会

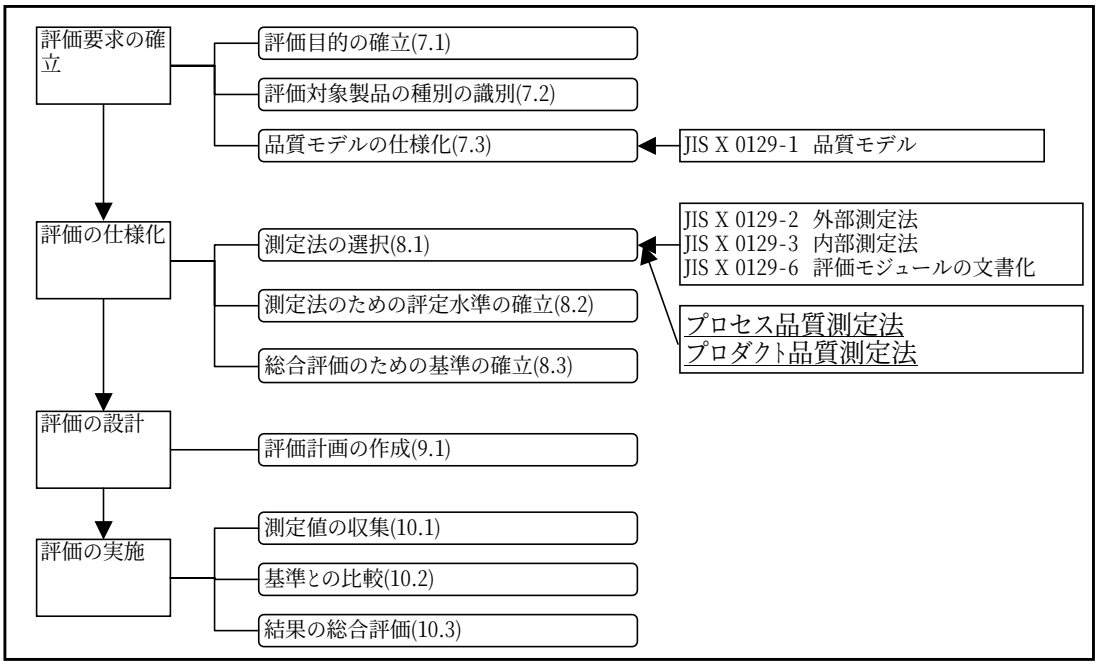
3.「ソフトウェア製品の評価－第１部：全体的概観」
　JIS X 0133-1：1999（ISO/IEC14598-1:1998)、日本規格協会

4.「ソフトウェアの品質の測定と評価　標準化の状況　ISO/IEC9126&14598 から SQuaRE へ」東基衛、2001.5.27

5.「役に立つデザインレビュー　－ソフトウェアにおける考え方と戦略」
　堀内純孝(著)、日科技連、1992

6.「21 世紀へのソフトウェア品質保証技術　－日科技連ソフトウェア品質管理研究会 10 年の成果」吉澤 正監修、日科技連ソフトウェア品質管理研究会編集

7. 共通フレーム 98-SLCP-JCF98（1998 版）
　ソフトウェアを中心としたシステム開発および取引のための共通フレーム
　株式会社　通産資料調査会　1998 年 10 月 28 日



付図1. IS014598を考慮した品質評価プロセス

付表１．IS014598とCMMI「測定と分析」の対比

IS014598			対応する「測定と分析」の固有プラクティス(SP)（付表２．参照）
評価要求の確立	7.1	評価目的の確立	SP1.1
	7.2	評価対象製品の種別の識別	
	7.3	品質モデルの仕様化	
評価の仕様化	8.1 8.1.1 8.1.2	測定法の選択 測定の種別 測定の為の要求	SP1.2、SP1.3
	8.2	測定法の為の評定水準の確立	SP1.4
	8.3	総合評価の為の基準の確立	
評価の設計	9.1	評価計画の作成	SP1.1、SP1.4
評価の実施	10.1	測定値の収集	SP2.1
	10.2	基準との比較	SP2.2
	10.3	結果の総合評価	

付表２．CMMI「測定と分析」の固有プラクティス

固有ゴール（SG）			固有プラクティス（SP）		
SG 1	「測定と分析」活動を整合する	測定の目標及び活動は、特定された情報ニーズ及び目標と整合されている	SP 1.1	測定目標を確立する	特定された情報ニーズ及び目標から導出された測定目標を確立し維持する
			SP 1.2	尺度を明記する	測定目標に対応する尺度を明記する
			SP 1.3	データの収集手順及び格納手段を明記する	測定データが獲得され格納される方法を明記する
			SP 1.4	分析手順を明記する	測定データが分析され報告される方法を明記する
SG 2	測定結果を提供する	特定された情報ニーズ及び目標に対応する測定結果が提供されている	SP 2.1	測定データを集める	明記された測定データを獲得する
			SP 2.2	測定データを分析する	測定データを分析し解釈する
			SP 2.3	データ及び結果を格納する	測定データ、測定仕様、及び分析結果を管理し格納する
			SP 2.4	結果を伝達する	全ての直接利害関係者に、「測定と分析」活動の結果を報告する

付表3．プロセス品質測定法によるメトリクス

SLCPアクティビティ		主な目的	メトリクス	計測項目	判断基準
		ヒヤリングインタビューの充実度の把握	ユーザーヒヤリング実施時間	ユーザーヒヤリング時間　H	多いほうがよい。
1.4.1	プロジェクト開始の準備		ユーザーヒヤリング項目数	ユーザーヒヤリング項目数	多いほうがよい。
1.4.2	システム要求分析		ユーザーヒヤリング回答率	ユーザーヒヤリング回答率	100%
		要求仕様実現度合いの把握	要求仕様実現率	搭載仕様件数/要求仕様件数×100	100%
1.4.3	システム方式設計		誤り率	機能の誤り個数/機能の総和×100	0
		成果物完成度の把握	DR指摘件数	レビュー指摘件数	ある基準値
1.4.4	業務詳細設計		DR指摘処置時間	指摘処置時間　H	少ないほど良い
			DR工数	人員×レビュー時間	多いほど良い
1.4.5	ソフトウェア要求分析		DR実施効率	指摘件数/レビュー工数	基準値より大
			DR工数比率	レビュー工数/工程内全工数×100	基準値
		顧客要求項目のうち、要求仕様に変換されていない項目がどの程度残っているかを示す。	未変換顧客要求項目数	確定顧客要求項目数－SW要求仕様に変換された顧客要求項目数)［個］	顧客要求増減後、指標値が0になるまで
		マイルストーンで予定されている段階とを比較し、予定より遅れていないかを確認	要求開発フェーズ予実比較	実際の要求開発段階vsマイルストーンでの要求開発段階	予定と実績の値を取るのみ
		詳細が固まっておらずSW要求仕様に変換できない顧客要求の項目数	未確定顧客要求項目数	未確定顧客要求項目数［個］	0
		要求変更が頻繁に起こっているかを一定期間変更件数で確認する。	要求変更頻度（時系列）	一定期間に発生した要求変更回数［件］	変更の回数値を取るのみ
		顧客要求が変更されたときの対応（受け入れ可能な変更とその対応方法、あるいは受け入れ不可能な変更）を考慮していない要求仕様の項目数	変更未考慮要求仕様項目数	（要求仕様項目数－顧客要求変更を考慮した要求仕様項目数）［個］	要求仕様増減後、指標値が0になるまで
		要求仕様から顧客要求またはその他の生成源（暗黙的な要求）への追跡ができない要求仕様項目数	生成源追跡不可能要求仕様項目数	（要求仕様項目数－生成源に到達可能な要求仕様項目数）［個］	要求仕様増減後、指標値が0になるまで
		要求仕様に記述された機能／非機能要求の検証（受け入れテスト）方法が文書化されていない項目数	検証不可能要求仕様項目数	（要求仕様項目数－検証方法が文書化された要求仕様項目数）［個］	要求仕様増減後、指標値が0になるまで
		顧客要求引出し段階で、顧客へのインタビューにかけた時間	顧客インタビュー時間	顧客インタビュー時間［時間］vs 確定顧客要求項目数	工数（時間×項目数）値を取るのみ
		要求分析段階で、要求分析のために作成したユースケースシナリオ数を要求規模で正規化した指標	ユースケースシナリオ密度	ユースケースシナリオ数［個］／確定顧客要求項目数［個］	1以上
		要求仕様作成段階で作成されたドキュメント量を要求規模で正規化した指標。	要求仕様のドキュメント密度	要求仕様ドキュメントの頁数［頁］／確定顧客要求項目数［個］	過去のデータや他の分類と比較して、指標値が妥当かどうかを判断する。
		要求変更が承認されているにも関わらず、要求仕様書（あるいは各成果物）に反映されていない項目の（最大）未反映時間。要求仕様書（あるいは各成果物）が最新の要求を反映しているかどうかを示す。	要求変更未反映時間	計測日時－承認済ステータス登録日時［日］	未反映時間が極端に長い項目については、要求変更が反映されていない危険性が高い。
		要求変更が承認されているのに、その変更が成果物に反映されていない要求仕様項目数。各成果物が最新の要求を反映しているかどうかを示す。	要求変更未反映項目数	変更承認済ステータス項目数［個］	指標値が極端に大きい場合やなかなか収束しない場合には、要求変更が成果物に反映されていない危険性が高い。
		要求ステータス（例：提案済、承認済、仕様作成済、設計済、実装済、検証済、削除済、却下済）を、ある一定期間ごとに集計してその分布を時系列に並べた指標。	要求ステータス分布（時系列）	要求ステータス項目数［個］	ステータスが順調に推移しているかを確認する。
		各要求仕様項目から、前方（設計・実装・検証フェーズ）の成果物を追跡できない項目数。	前方追跡不可能要求仕様項目数	（要求仕様項目数－前方成果物に到達できる要求仕様項目数）［個］	指標値が1以上の場合、要求が実現されていない危険性がある。
		ステークホルダー（関係者）のコミットメント（合意）が得られていない要求変更の（最大）未承認時間を示す。	要求変更未承認時間	計測日時－変更提案済ステータス登録日時［日］	未承認時間が極端に長い項目については、要求変更が合意されずに開発が進んでいる、あるいは要求変更が放置されている危険性が高い。
1.4.6	ソフトウェア方式設計	成果物完成度	DR指摘率	実績値/目標値	基準値
			DR指摘密度	DR指摘件数/開発規模KS×100	基準値
			DR実施率	実績回数/予定回数×100	100%
1.4.7	ソフトウェア詳細設計		DR実施効率	指摘件数/DR工数	基準値
			DR工数比率	レビュー工数/工程内全工数×100	100%
			指摘事項改善率	解決事項/指摘件数×100	0、小さいほど良い
			ドキュメント検査不良率	不良件数/ドキュメント枚数	0、小さいほど良い
			ドキュメント不備率	修正ページ数/全ドキュメント枚数×100	基準値
			ドキュメント率	ドキュメント枚数/開発規模KS	基準値より大
		要求仕様の品質の確保	チェックリスト件数	チェック件数/機能項目数	0、小さいほど良い
			仕様変更件数	仕様変更件数	
			仕様実現率	工程内実現件数/仕様件数	100%
			機能誤り率	機能の誤り個数/機能の総数	0

SLCPアクティビティ	主な目的	メトリクス	計測項目	判断基準
	設計内容のドキュメントが、どれだけ記述されているかを定期的に集計し、予定に対してどれだけ遅れているかを、時系列で表現する。	設計書ページ数（予実比較）	(その時点までの)設計書予定ページ数[頁] VS 設計書ページ数[頁]	予定ページ数と比較して、ページ数が著しく少なくないか確認する。
	設計されているプログラム構成要素（クラス、関数）の数を定期的に集計し、予定数と比較する。	プログラム構成要素数（予実比較）	(その時点までの)設計予定プログラム構成要素数[個] VS	設計したプログラム構成要素の数と予定数の差を把握する。
	設計フェーズにおけるマイルストーン到達日と、予定到達日を比較し、遅れ具合を判断する。	マイルストーン（予実比較）	予定到達日数[年月日] VS 到達日数[年月日]	マイルストーン到達の遅れ日数を把握する。
	機能要求がどれだけ設計に反映されているのかを示す指標。	シーケンス作成率	(ユースケースごとの)(シーケンスのあるシナリオ数 / 全シナリオ) × 100 [%]	1以上
	プログラムの構成要素(例えばクラス)に到達できない要求の数。	構成要素トレース未到達要求数	要求仕様数 - 構成要素に到達可能な項目の数 [個]	0
	要求されている不正アクセス対策の数と、そのうち設計に反映された不正アクセス対策の数を集計し、設計に反映されていない不正アクセス対策の数を算出する。	セキュリティ未対策数	不正アクセス対策要求数 - 不正アクセス対策設計数 [件]	0
	要求されている障害（システムに対するアクシデント）対策と、そのうち設計されている障害対策の数を集計し、設計されていない障害対策の数を算出する。	障害未対策数	障害対策要求数- 障害対策設計数 [件]	0
	既存のコードやライブラリ、ソフトウェアを使用する場合、それらの安全性を測るための指標。	使用条件でのテスト合格率	(テストした使用条件 / 使用条件) × 100 [%]	指標値が著しく低いものを検出、また100[%]に満たないものを把握する。
	既知の脆弱性と、そのうち対策が取られた脆弱性の数を集計し、対策が取られていない脆弱性の数を算出する。	脆弱性未対策数	既知の脆弱性の数 - 対策された脆弱性の数 [件]	既知の脆弱性のうち、未対策な脆弱性の数を把握する。
	全シナリオに対する、処理時間を予測したシナリオの割合。	処理時間を予測したシナリオの割合	(処理時間を予測したシナリオの数 / シナリオの数) × 100 [%]	処理時間を予測したシナリオの数が、全シナリオ中に占める割合を把握する。
	設計内容が記述されたドキュメントのページ数を集計単位（例えば機能単位）で集計し、開発規模(FP、SLOC、ELOC)で正規化したもの。	ドキュメント率	設計書ページ数 / FP [頁/FP],SLOC [頁/SLOC],ELOC [頁/ELOC]	ドキュメント率が著しく低い部分を把握する
	設計内容が記述されたドキュメントに対しチェックリスト等に基づいた検査を行い、集計単位（例えば機能単位）ごとに、記述が必要な項目の数と記述されている項目の数から百分率を算出する。	ドキュメント完全率	(設計書記述項目数 / 要設計書記述項目数) × 100 [%]	指標値が低い部分、すなわちドキュメント化が不十分な部分を検出する。
	設計内容が記述されたドキュメント1ページ当たりに、検査(レビュー)により不良と判断された箇所がどれだけあるかを示した指標。	ドキュメント検査不良率	不良件数 / ドキュメントページ数 [件/頁]	ドキュメント検査不良率が高い部分、あるいは著しく低い部分を検出する。
	設計結果を基にして新規開発する規模を見積もり、計画されていた規模と比較する。	新規開発規模(予実比較)	計画されていた新規開発規模 VS 設計結果から見積もった新規開発規模	計画されていた開発規模と、設計結果から見積もった開発規模の差を把握する。
	設計結果をもとにして後工程に必要な工数を見積もり、計画されていた工数と比較する。	後工程にかかる工数(予実比較)	計画されていた後工程の工数[人月] VS 設計結果から見積もった工数[人月]	差を把握
	要求仕様に到達できないプログラム構成要素(例えばクラス)の数。	要求仕様トレース未到達構成要素数	(プログラム構成要素数 / 要求仕様に到達可能な構成要素の数) × 100[%]	100
	集計単位（例えば機能）ごとに設計時間を集計し、それを規模で正規化する。	設計時間	設計時間 / FP(SLOC, ELOC) [H/FP(SLOC,ELOC)]	設計時間が規模に対して著しく少ない部分や、予定よりも少ない部分を検出する。
	各レビュー後に、要求の理解不十分が原因であるレビュー指摘項目数を集計し、時系列で表現する。	要求の理解不十分が原因であるレビュー指摘項目数	要求の理解不十分が原因であるレビュー指摘項目数 [件]	要求の理解不十分が原因であるレビュー指摘項目数が0件、あるいは減少していなければならない。
	変更対策が必要な要求仕様の数と、それらのうち対策が取られたも数を集計し、未対策数を算出する。	要求仕様変更未対策件数	変更対策が必要な要求仕様の数 - 変更対策が取られた要求仕様の数 [件]	0
	内容についてレビューされたシーケンスが、全シーケンスに占める割合。	シーケンスレビュー率	(レビューされたシーケンス数 / シーケンス数) × 100 [%]	十分な数のシーケンスがレビューされているか把握する。
	各レビューごとに指摘項目数を集計し、レビュー対象が適切であったか判断する。	各レビューの要修正レビュー指摘項目数	要修正レビュー指摘項目数 [件]	各レビューにおける指摘項目数の偏りを把握する。
	集計単位（例えば機能）ごとにレビュー回数を集計する。	個別レビュー回数	集計単位ごとのレビュー回数 [回], 時間 [H]	著しくレビュー回数の少ないものを検出する。

SLCPアクティビティ		主な目的	メトリクス	計測項目	判断基準
		レビューした機能(クラス、関数)が全体に占める割合。	レビューカバレッジ	(レビューした機能数 / 全機能数) × 100 [%],○(レビューしたクラス数 / 全クラス数) × 100 [%],○(レビューした関数数 / 全関数数) × 100 [%]	レビューした機能（クラス、関数）が全体に占める割合を把握する。
		レビューで指摘された修正の必要な項目数と、既に修正した項目の数を集計単位（例えば機能）ごとに集計し、未修正な項目の数を算出する。	未修正件数	要修正件数 - 修正件数 [件]	未修正件数が多い部分を検出する。
		機能ごとに要求変更の回数を集計する。	要求変更回数	（機能ごとの）要求変更回数 [回]	変更回数が著しく多い機能を検出する。
1.4.8	ソフトウェアコード作成およびテスト	成果物完成度	DR指摘率	実績値/目標値	基準値
			DR指摘密度	DR指摘件数/開発規模KS×100	基準値
			DR実施率	実績回数/予定回数×100	100%
			DR実施効率	指摘件数/DR工数	基準値
			DR工数比率	レビュー工数/工程内全工数×100	100%
			チェックリスト消化率	実績/予定	100%
			指摘事項改善率	解決事項/指摘件数×100	0、小さいほど良い
			ドキュメント検査不良率	不良件数/ドキュメント枚数	0、小さいほど良い
			ドキュメント不備率	修正ページ数/全ドキュメント枚数×100	基準値
			ドキュメント率	ドキュメント枚数/開発規模KS	基準値より大
		品質の確保	バグ発生	バグ件数/プログラムステップ数KS	0、小さいほど良い
			カバレッジ	%	0、小さいほど良い
			修正量	修正ステップ数KS	0、小さいほど良い
			摘出率	実績値/目標値×100	100%
			残存密度	残存バグ数/開発規模KS×100	0、小さいほど良い
			再利用率	再利用規模KS/開発規模KS	基準値より大
			チェックリスト件数	チェック件数/ステップ数	
		対象がどこまで実装されているかを示す。	実装割合	実装割合 = 100 × 実装が完了した機能数 / 全機能数 [%]	進捗が遅れている箇所を探す。
		ある時点での実装の遅れを，その時点での予定されていた機能に対する未実装機能の割合でしめす。	実装の遅れ	実装遅れ = 100 - 100 × 実装が完了した機能数 / 現在までに実装される予定の機能数 [%]	進捗が遅れている箇所を探す。
		実装後に単体テストを行ったかどうかを示す。	単体テストの有無	機能モジュールに含まれる関数を対象に100 × 単体テストを実施した関数の数 / 機能モジュールに含まれる関数の数 [%]	テストが行われていない箇所を探す。
1.4.9	ソフトウェア結合	実装後に結合テストを行ったかどうかを示す。	結合テストの有無	機能モジュールに含まれる関数を対象に① 100 × 結合テストを実施した機能の数 / 機能モジュールに含まれる機能の数 [%],② 100 × 結合テストを実施した関数の数 / 機能モジュールに含まれる関数の数 [%]	テストが行われていない箇所を探す。
		レビュー・単体テスト・結合テスト・総合テストで発見された不具合の内、未だ修正されていない不具合の件数を表す。	未修正件数	未修正件数 = 要修正件数 ― 修正件数	機能毎に集計し、機能間の比較により値が高い箇所を探す。
1.4.10	ソフトウェア適格性確認テスト	不具合修正後の確認のテストが行われていない件数を示す。	修正確認テスト未実施件数	修正件数 - 修正確認を実施した修正件数	機能毎に集計し、確認テストが行われず残っている箇所を探す。
		多数の開発者によって開発を行う場合，開発者間でソースコードのコーディングスタイルを統一しソースコードの可読性を高めるためと，不具合の出にくいコーディングを目指すために，守るべき指針としてコーディング規約が採用されることがある。	コーディング規約違反件数	コーディング規約に違反している件数	機能毎に集計した上で他の機能と比較し，多くの件数が発見された箇所
		ソースコードの実装量に対する見積り値と，実際の実装量とのずれを表す。	実装量の見積り誤差	見積り誤差 = |SLOC(見積り値) - SLOC(実績値)| / SLOC(見積り値) [%],見積り誤差 = |ELOC(見積り値) - ELOC(実績値)| / ELOC(見積り値) [%]	他の機能・類似のプロジェクトと比較し，突出した値の箇所を確認する。
		一つの関数内に多量の処理を記述した長大な関数は，その関数を理解するのが困難になり不具合が入りこむ可能性が高くなる。	長すぎる関数	関数の行数 (全関数について計測)	一般的には ELOCで100以下程度に抑えるべきと言われている。
		関数内の処理が複雑すぎる場合，その関数を理解するのが困難になり不具合が入りこむ可能性が高くなる。	複雑すぎる関数	関数の複雑度を全関数について計測，関数の複雑度を表す最も一般的なメトリクスは，McCabeの複雑度と呼ばれるもので，通常，分岐(if等)の数＋1で計算される。	McCabeの複雑度の場合、一般的には 10以上の値から不具合が発生し易いと言われているが、10以上の値全てを対象とするより，複雑度が高い関数が多い機能から対処する方ほうがよい。
		関数を新たに作成する時，似たような処理をもつ関数全体，もしくはその一部をコピーし，修正を加えて作成することは，一般的に行われている。	多数の類似した関数	各関数に対し，その関数と一定値以上の類似性をもつ関数の数を計測	他の機能との比較により判定

SLCPアクティビティ		主な目的	メトリクス	計測項目	判断基準
		単体テスト・結合テストで実施されたテスト項目数の十分性を示す。単体テスト・結合テスト等のテスト項目数と，それを正規化する値(コード量・機能数)値から算出する。	実施されたテスト項目数の十分性	正規化の例,○ テスト項目数 / 機能数 ,○ テスト項目数/FP,○ テスト項目数 / SLOC(一般的には KLOC(1000) 単位),○ テスト項目数 / ELOC(一般的には KLOC(1000) 単位)	他の機能・過去のデータと比較し，著しく低い値の箇所を探す
		テスト網羅性を評価するための指標。実施されたテスト項目によって、コード「全体」の何%が実際に動作したかを示す。	テストカバレージ率	集計単位（通常は機能単位）に対して、C0カバレージ [%] 100 × テスト時に実行されるステートメント数/ソースコードのステートメント数 [%] ,C1カバレージ [%] 100 × テスト時に実行済みの分岐数 / ソースコードの全分岐数 [%]	事前に設定した閾値よりも値が小さい箇所を特定する。
		各不具合毎に不具合修正後の確認のテストが行われたかどうかを示す。	修正確認テストの実施割合	修正確認テストを実施した修正件数/修正件数	機能毎に集計し、確認テストが行われず残っている箇所を探す。
		要求管理システム等に登録された要求が，一旦登録された後に変更された回数を示す。	変更回数(要求管理)	要求が登録された後に変更された(変更が登録された)回数	他の箇所に比べ変更が著しく多い箇所を探す。
		ソースコードに対する変更回数を表す。	変更回数(ソースコード)	各ソースコードの一定期間(一週間毎等)毎の変更回数	他の箇所に比べ変更が著しく多い箇所を探す。
1.4.11	システム結合	一定期間毎の各ソースコードに対する変更の量を表す。	変更量(ソースコード)	各ソースコードの期間毎の変更量	他の機能に比べ変更が著しく多い箇所を探す。
		品質の保証	試験項目数設定率	試験項目数/開発規模KS	基準値
1.4.12	システム適格性確認テスト		試験実施率	実行量/予定量×100	100%
			カバレッジ	%	0，小さいほど良い
1.4.13	ソフトウェア導入		バグ発生率	バグ件数/プログラムステップ数KS	0，小さいほど良い
			バグ修正量	修正ステップ数KS	0，小さいほど良い
1.4.14	ソフトウェア受入れ試験		バグ摘出率	実績値/目標値×100	100%
			指摘事項改善率	決定事項/指摘件数×100	
			差し戻り回数	差し戻り回数	0，小さいほど良い
			ドキュメント検査不良率	不良件数/ドキュメント枚数	0
		要求分析／設計／実装フェーズで作成されたプロダクトに対するレビュー実績（実施回数）が，予定に対してどの程度遅れているかを，各フェーズの集計単位（通常は機能単位）ごとに算出する。	レビュー未実施回数（予実比較）	○（その時点までの）レビュー予定回数－レビュー実施回数[回],○（その時点までの）レビュー予定回数[回]vsレビュー実施回数[回],各フェーズの集計単位（通常は機能単位）ごとに集計して算出する。	レビュー実施が予定より遅れている箇所を把握する。
		要求分析／設計／実装フェーズのレビューで指摘された要修正項目数を、集計単位（通常は機能単位）ごとに集計してその規模（コード量、機能数、FPなど）で正規化したもの。	要修正レビュー指摘発生密度	○（要修正レビュー指摘項目数／ELOC[SLOC]） ,○（要修正レビュー指摘項目数／機能数[FP]） ,集計単位（通常は機能単位）ごとに集計して算出する。	（指標「レビュー実施密度」等によりレビューが十分だと判断できる）他の機能や過去のデータと比較して，著しく値の大きい箇所を特定する。
		要求分析／設計／実装フェーズのレビュー予定回数を、集計単位（通常は機能単位）ごとに集計してその規模（コード量、機能数、FPなど）で正規化したもの。	レビュー予定密度	○（レビュー予定回数[時間]／ELOC[SLOC]） ,○（レビュー予定回数[時間]／機能数[FP]） ,集計単位（通常は機能単位）ごとに集計して算出する。	他の機能や過去のデータと比較して，著しく値の小さい箇所を特定する。
		要求分析／設計／実装フェーズのレビュー実施回数を、集計単位（通常は機能単位）ごとに集計してその規模（コード量、機能数、FPなど）で正規化したもの。	レビュー実施密度	○（レビュー実施回数[時間]／ELOC[SLOC]） ,○（レビュー実施回数[時間]／機能数[FP]） ,集計単位（通常は機能単位）ごとに集計して算出する。	他の機能や過去のデータと比較して，著しく値の小さい箇所を特定する。
		ピアレビューや総合テストの対象となる成果物の要件（仕様書等）のドキュメント量を、集計単位（通常は機能単位）ごとに集計してその規模（コード量、機能数、FPなど）で正規化したもの。	要件のドキュメント密度	○（要件ドキュメントの頁数／ELOC[SLOC]）、○（要件ドキュメントの頁数／機能数[FP]） ,集計単位（通常は機能単位）ごとに集計して算出する。	他の機能や過去のデータと比較して，著しく値の小さい箇所を特定する。
		総合テスト項目が全体の何%実施されたかを、集計単位（通常は機能単位）ごとに算出し、時系列で表現する。	総合テスト実施率（時系列）	○(総合テスト実施項目数／総合テスト予定項目数)*100 [%],集計単位（通常は機能単位）ごとに集計して算出する。	総合テストの進行が滞っている箇所を把握する。
		総合テスト項目が全体の何%合格したかを、集計単位（通常は機能単位）ごとに算出し、時系列で表現する。	総合テスト合格率（時系列）	○(総合テスト合格項目数／総合テスト予定項目数)*100 [%],集計単位（通常は機能単位）ごとに集計して算出する。	総合テストによる品質確保が滞っている箇所を把握する。
		総合テストが，予定に対してどの程度遅れているかを、集計単位（通常は機能単位）ごとに集計して比較する。	総合テスト予実比較	○(その時点までの)総合テスト予定項目数[個] vs 総合テスト実施項目数[個]	総合テストの実施が予定より遅れている箇所を把握する。
		総合テストが，予定に対してどの程度遅れているかを、集計単位（通常は機能単位）ごとに集計し、時系列に表示して比較する。	総合テスト予実比較（時系列）	(その時点までの)総合テスト予定項目数[個] vs 総合テスト実施項目数[個]vs 総合テスト合格項目数[個]	総合テストの実施が予定より遅れている箇所を把握する。
		総合テストでの不具合検出数から、残存不具合数を推定する。	推定残存不具合数	○要修正不具合総数（推定）－要修正不具合数	リリース予定時期での残存不具合数を予測し、予定している日時までに十分不具合を除去できそうかを把握
		総合テストでの不具合検出数の収束が予測される日時が，テスト終了予定期日からどれだけ遅れそうかを予測する。	期日遅れ予測日数	○不具合収束予測日－テスト終了予定日	リリース予定時期での残存不具合数を予測し、予定している日時までに不具合が収束しそうかを把握

SLCPアクティビティ		主な目的	メトリクス	計測項目	判断基準
		総合テストで検出された要修正不具合数を、一定期間ごとに集計して時系列に並べて表現したもの。この指標値がしきい値以下になれば、品質が安定したと推測される。	不具合発生頻度	○一定期間（通常は一日単位）に検出された要修正不具合数,○一定工数（あるいは一定テスト項目数）に検出された要修正不具合数	指標値がある値以下の状態が一定期間続けばテスト終了とする。
		総合テストで検出された要修正不具合数を、集計単位（通常は機能単位）ごとに集計してその規模（コード量、機能数、FPなど）で正規化したもの。	不具合発生密度	○（要修正不具合数／ELOC[SLOC]）,○（要修正不具合数／機能数[FP]）	（指標「テスト実施密度」等によりレビューが十分だと判断できる）他の機能や過去のデータと比較して、著しく値の大きい箇所を特定する。
		総合テスト予定項目数を、集計単位（通常は機能単位）ごとに集計してその規模（コード量、機能数、FPなど）で正規化したもの。	テスト予定密度	○（総合テスト予定項目数／ELOC[SLOC]）,○（総合テスト予定項目数／機能数[FP]）	他の機能や過去のデータと比較して、著しく値の小さい箇所を特定する。
		総合テスト実施項目数を、集計単位（通常は機能単位）ごとに集計してその規模（コード量、機能数、FPなど）で正規化したもの。	テスト実施密度	○（総合テスト実施項目数／ELOC[SLOC]）,○（総合テスト実施項目数／機能数[FP]）	他の機能や過去のデータと比較して、著しく値の小さい箇所を特定する。
		用意された全テスト項目のうち、異常系のテスト項目が占める比率。	異常系テスト項目比率（予定）	○（異常系総合テスト予定項目数／総合テスト予定項目数）*100 [%]	他の機能や過去のデータと比較して、著しく値の小さい箇所を特定する。
		実施された全総合テスト項目のうち、異常系のテスト項目が占める比率。	異常系テスト項目比率（実績）	○（異常系総合テスト実施項目数／総合テスト実施項目数）*100 [%]	他の機能や過去のデータと比較して、著しく値の小さい箇所を特定する。
		テスト網羅性を評価するための指標。	テストカバレージ率	集計単位（通常は機能単位）に対して、C0カバレージ [%] 100 × テスト時に実行されるステートメント数/ソースコードのステートメント数 [%] ,C1カバレージ [%] 100 × テスト時に実行済みの分岐数 / ソースコードの全分岐数 [%]	事前に設定した閾値よりも値が小さい箇所を特定する。

付表4. SLCPと品質特性／品質副特性の関係

SLCPアクティビティ		SLCPタスク		成果物		品質特性／品質副特性																											
						機能性					信頼性				使用性					効率性			保守性					移植性				
				計画資料	実施資料	合目的性	正確性	相互運用性	セキュリティ	標準適合性	成熟性	障害許容性	回復性	標準適合性	理解性	習得性	運用性	魅力性	標準適合性	時間効率性	資源効率性	標準適合性	解析性	変更性	安定性	試験性	標準適合性	環境適応性	設置性	共存性	置換性	標準適合性
1.4.1	プロジェクト開始の準備	1.4.1.4	開発プロセス実施計画の策定	開発プロセス実施計画書																												
1.4.2	システム要求分析	1.4.2.1	システム要求事項の分析		システム要求仕様書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
		1.4.2.2	システム要求事項の定義	システム適格性確認テスト計画書																												
		1.4.2.3	システム要求事項の評価		システム要求評価報告書																												
1.4.3	システム方式設計	1.4.3.1	システムの最上位レベルでの方式確立		システム方式設計書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
		1.4.3.2	システム方式の評価		システム方式評価報告書																												
		1.4.3.3	システムのテスト方針策定	システム結合テスト計画書																												
1.4.4	業務詳細設計	1.4.4.3	伝票・帳票の設計		伝票、帳票																												
		1.4.4.4	業務運用の詳細設計		業務運用詳細設計書																												
		1.4.4.5	業務運用手順の作成		業務運用マニュアル	○	○				○	○	○	○	○	○	○	○	○	○	○	○											
					入出力操作マニュアル																												
					業務運用規定																												
1.4.5	ソフトウェア要求分析	1.4.5.1	ソフトウェア要求事項の確立	ソフトウェア的確性確認テスト計画書	ソフトウェア要求仕様書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
		1.4.5.2	ソフトウェア要求事項の評価		ソフトウェア要求事項評価報告書																												
1.4.6	ソフトウェア方式設計	1.4.6.1	ソフトウェア構造とコンポーネントの方式設計	ソフトウェア結合テスト計画書	ソフトウェア方式設計書																												
		1.4.6.2	外部、コンポーネント間の各インターフェースの方式設計		インターフェース方式設計書																												
		1.4.6.3	データベース最上位レベルの設計		データベース方式設計書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
		1.4.6.4	利用者文書（暫定版）の作成		利用者文書																												
		1.4.6.6	ソフトウェア方式設計の評価		ソフトウェア方式評価報告書																												
1.4.7	ソフトウェア詳細設計	1.4.7.1	ソフトウェアコンポーネント詳細設計		ソフトウェア詳細設計書																												
					インターフェース詳細設計書																												
					データベース詳細設計書																												
		1.4.7.5	ソフトウェアユニットのテスト要求事項の更新	ソフトウェアユニットテスト計画書		○	○			○	○	○	○	○						○	○	○										
		1.4.7.7	ソフトウェア詳細設計およびテスト要求事項の評価		ソフトウェア詳細設計評価報告書																												
					ソフトウェア要求事項評価報告書																												
1.4.8	ソフトウェアコード作成およびテス	1.4.8.1	ソフトウェアユニットとデータベースの作成及びテスト手順	ソフトウェアユニットテスト手順書	ソフトウェアコード																												
			とテストデータの作成	データベーステスト手順書	ソフトウェアユニット																												
					データベース																												
		1.4.8.2	テスト手順とテストデータの作成、テストの実施		ソフトウェアユニットテスト報告書	○	○			○	○	○	○	○						○	○	○										
					データベーステスト報告書																												
		1.4.8.5	ソフトウェアコードおよびテスト結果の評価		ソフトウェアコードおよびテスト評価報告書																												
1.4.9	ソフトウェア結合	1.4.9.1	ソフトウェア結合計画の作成	ソフトウェア結合テスト手順書		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
		1.4.9.2	ソフトウェア結合テストの実施		ソフトウェア結合テスト報告書																												
		1.4.9.5	ソフトウェア結合テストの評価		ソフトウェア結合テスト評価報告書																												
1.4.10	ソフトウェア適格性確認テスト	1.4.10.1	ソフトウェア適格性確認テストの実施	システムテスト適格性確認テスト手順	ソフトウェア適格性確認テスト報告書																												
		1.4.10.4	ソフトウェア適格性確認テストの評価		ソフトウェア適格性確認確認テスト評価報告書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
		1.4.10.5	監査の支援		監査報告書																												
1.4.11	システム結合	1.4.11.1	システム結合テストの実施	システム適格性確認テスト手順	システム結合テスト報告書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
		1.4.11.3	システム結合テストの評価		システム結合テスト評価報告書																												
1.4.12	システム適格性確認テスト	1.4.12.1	システム適格性確認テストの実施		システム適格性確認テスト報告書																												
		1.4.12.2	システムの評価		システム評価報告書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
		1.4.12.3	利用者文書の更新		監査報告書																												
1.4.13	ソフトウェア導入	1.4.13.1	ソフトウェア導入（インストール）の計画の作成	ソフトウェア導入計画書	ソフトウェア導入報告書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
1.4.14	ソフトウェア受入れ試験	1.4.14.1	取得者の受入れレビューと受入れテストの支援		受入れレビュー記録書	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
					受入れテスト報告書																												

○：比較的関係がある

付表5. 品質特性に着目したレビューポイント

大項目		中項目		チェック項目	
項番	項目名	項番	項目名		
5.2.1	機能性	1	合目的性	(1)	要求機能・品質要求・性能要求が仕様書に記述されているか
				(2)	前段階の仕様書に記述された機能が現段階の仕様書に記述されているか
				(3)	ユーザの要求に対し機能仕様書のどの機能と合致するか明確になっているか
				(4)	機能仕様書に書かれた機能がプログラムのあるモジュールで実現されているか
				(5)	ドキュメント、マニュアルの機能説明の間に矛盾はないか、
				(6)	ドキュメント、マニュアルの機能説明とプログラムの動作に矛盾はないか
		2	正確性	(1)	入力の時間的制約、物理的場所、入力順序、タイミング、入力属性、入力方法について考慮しているか
				(2)	処理について時間的制約、物理的場所について考慮しているか
				(3)	処理について処理順序、タイミング、アルゴリズムが考慮されているか
				(4)	処理結果について時間的制約、物理的制約、出力順序、タイミング、出力属性、出力方法が考慮されているか
				(5)	入力の組合せ、入力の競合に関して考慮しているか
				(6)	処理の組合せ、処理の競合に関して考慮しているか
				(7)	出力の組合せ、出力の競合に関して考慮しているか
		3	相互運用性	(1)	システムとして通信手順が定義されているか
					(1－1) プロトコル(通信規約)、初め・終わりの合図が定められているか
					(1－2) 障害が発生したことを認識する約束、障害内容を知らせる約束は決められているか
					(1－3) 障害の回復条件が決められているか
				(2)	システムで定義されている通信手順が守られているか
					(2－1) プロトコル、初め・終わりの合図が守られているか
					(2－2) 障害が発生したことを認識する約束、障害内容を知らせる約束は決められているか
				(3)	システムとしてデータ形式が定義されているか
					(3－1) データの変換則、データ変換が不可時の約束、データ順序の相手との約束が定められているか
					(3－2) アドレス付けがユニークに定められているか
					(3－3) データの定義、データ例外の定義が定められているか
				(4)	システムで定義されているデータ形式が守られているか
					(4－1) データの変換則、データ変換が不可時の約束、データ順序の相手との約束が守られているか
					(4－2) アドレス付け、データの定義、データ例外の定義が守られているか
				(5)	相手の能力に合わせて情報量の制御が行えるか
		4	標準適合性	(1)	標準(法規制など)に規定された外部仕様に適合しているか
				(2)	標準に規定されていないものについての処理は定められているか
				(3)	標準に規定されていないものに対して、処理を定める場合は将来の変更に対処できるように設計されているか(たとえばオプショナル仕様化)
				(4)	特定のハードウェア、ソフトウェアに依存していないか
				(5)	安全設計がなされているか
					(5－1) 主体システムの要求される製造物責任を考慮した安全設計がなされているか
					(5－2) フェールセーフ設計、フェール ソフト設計、フールプルーフ設計の概念があるか
					(5－3) フォールト マスキング設計、フォールト アボイダンス設計の概念があるか
		5	セキュリティ	(1)	当該ソフトウェアの持つ、あるいは取り扱う情報に対して、他から破壊されないようになっているか
				(2)	万一、上記(1)が守れない時、破壊が認識できるようになっているか
				(3)	破壊した相手がわかる仕掛け、回復できる仕掛けになっているか
				(4)	上記(2)の時、回復できるようになっているか
				(5)	情報の破壊に備え、事前に情報が保存されているか
				(6)	当該ソフトウェアの持つ、あるいは取り扱う情報が、他から盗まれないようになっているか
				(7)	万一、上記(6)が守れない時、盗まれたことが認識できるようになっているか
				(8)	盗んだ相手を識別できる仕掛けになっているか
				(9)	万一盗まれても、解読できない処置が施されているか
				(10)	当該ソフトウェアの扱う情報が不当に他に漏れないようになっているか
					(10－1) 履歴確保、パスワード チェック、ファイルの二重化を行っているか
5.2.2	信頼性	1	成熟性(無欠陥性)	(1)	機能仕様について、信頼性レビューを実施したか、そのレビュー結果、問題点を解決したか
				(2)	潜在不良予測値を超す不良が摘出されたか
				(3)	テスト経過からみて、確認項目はすべて確認したか
				(4)	テスト条件の多重度、組合せ度(テスト濃度)、摘出不良からみて不良は収束したか
				(5)	テスト時間からみて、品質は確保されたか
				(6)	技術者の技能、能力からみて、品質は確保されたと判断可能か
				(7)	テスト環境からみて、機能確認は完了したと判断可能か
				(8)	不良の摘出状況からみて、品質は確保されたと判断できるか
				(9)	C0、C1などのカバレージは完了しているか
				(10)	列挙された問題点はすべて解決、または処置がなされているか
		2	障害許容性	(1)	フールプルーフ性
					(1－1) ミス オペレーションに対しエラーチェック、エラーの旨の表示を行っているか
					(1－2) システム停止要求といったシステム運行に影響の大きなオペレーションに対しては、問い合わせ応答形式で再確認を求める処理となっているか
					(1－3) 電源断などに対してファイル破壊防止処理があるか
				(2)	フォールト トレランス性
					(2－1) 発生した障害により影響を受けた処理を再度実行して正常に処理する機能になっているか
					(2－2) 発生した障害により影響を受けた部分に対し、代わりの処理で正常に処理する機能になっているか
					(2－3) 発生した障害に対して多重系で処理し、影響を無視できるようになっているか(マルチプロセッサ)
					(2－4) 発生した障害に対して、この障害の影響をくいとめて処理を続行する機能があるか
					(2－5) 発生した障害により影響を受けた部分を修復して、処理を続行可能とするようになっているか(機能回復ルーチン)
					(2－6) 発生した障害により影響を受けた装置を交代して実行する機能となっているか(オペレーション ミス、ソフトウェアの欠陥、ハードウェア故障、ハードウェア欠陥により発した障害を示す)
				(3)	異常箇所の検知ができるか
				(4)	必要情報、履歴がとられているか
				(5)	自動修復ができるか
		3	回復性	(1)	チェックポイント リスタート性

大項目		中項目		チェック項目	
項番	項目名	項番	項目名		
					(1－1)　障害が発生した場合、それまでに実行した結果の保証をするデータを保持する機能があるか
					(1－2)　障害が発生することを予測し、定時間ごとに罹障範囲を短縮するために結果の保持をする機能があるか
					(1－3)　障害が発生した場合、その当該の障害の影響を受けたデータを再度処理できる機能があるか
				(2)	部分閉塞
					(2－1)　障害が発生した場合、影響を受けた部分の処理を閉塞し、他の部分を実行する機能はあるか
					(2－2)　障害が発生した場合、障害を発生した部分を閉塞し、実行を継続する機能はあるか
				(3)	トレース、ロギングなどの障害診断用の記録はとれるか
				(4)	ダンプ解析などの障害解析用の記録はとれるか
5.2.3	使用性	1	理解性	(1)	統一がとれているか
					(1－1)　仕事の始まり方、終り方に統一がとれているか
					(1－2)　情報の与え方、出し方に統一がとれているか
					(1－3)　情報を出す位置、与える順序に統一がとれているか
					(1－4)　メッセージが統一されているか
				(2)	標準化されているか
				(3)	一般的な方法論であるか
				(4)	法則性、例示（デモンストレーション）があるか
				(5)	視覚化されているか
					(5－1)　アイコン表示など工夫、図表化されているか
					(5－2)　概念図があるか
				(6)	説明にレベルが設定されているか
				(7)	説明レベルが選択できるか
				(8)	ＨＥＬＰ機能、支援システム（ソフトウェア、マニュアル、ハードウェアなど）があるか
				(9)	使用者のレベルにあった選択が可能か
		2	習得性	(1)	統一がとれているか
					(1－1)　仕事の始まり方、終り方に統一がとれているか
					(1－2)　情報の与え方、出し方に統一がとれているか
					(1－3)　情報を出す位置、順序に統一がとれているか
					(1－4)　メッセージが統一されているか
				(2)	標準化されているか
				(3)	一般的な方法論であるか
				(4)	法則性、例示（デモンストレーション）があるか
				(5)	視覚化されているか
					(6－1)　アイコン表示などの工夫や、図表化がされているか
					(6－2)　概念図があるか
				(6)	覚えやすいように工夫されているか
					(6－1)　アクセント、変化があるか
					(6－2)　形が変わっているか
					(6－3)　得意な音が出るか（単に音が出るか）
				(7)	ＨＥＬＰ機能、支援システム（ソフトウェア、マニュアル、ハードウェアなど）があるか
				(8)	説明にレベルが設定されているか
				(9)	説明レベルが選択できるか
				(10)	使用者のレベルにあった選択が可能か
		3	運用性	(1)	オペレータへの指示
					(1－1)　指示があるか
					(1－2)　ガイダンスが出るか
					(1－3)　指示が選択できるか
				(2)	オペレーション結果
					(2－1)　結果が報告されるか
				(3)	オペレーション性
					(3－1)　ミスしにくい入力形式であるか
					(3－2)　ガイダンス、重要なオペレーションに対する問い合わが出るか
					(3－3)　前オペレーションに戻るなどの操作が可能か
					(3－4)　スクロールなど画面操作が容易か
					(3－5)　オペレーションの簡略化の工夫がされているか
				(4)	オペレーション結果エラーの場合
					(4－1)　エラーの旨の表示があるか
					(4－2)　エラー後の処理について指示があるか
					(4－3)　再入力、前オペレーションに戻ることが可能か
					(4－4)　メッセージ表示が工夫されているか
				(5)	オペレーションエラー時の処理
					(5－1)　エラーしてもシステムに重大な支障をきたさないか
					(5－2)　エラーしても回復機能、手段があるか
					(5－3)　エラーした結果、システムに支障をきたす場合には安全側に処理がされるか
				(6)	システムの運転状況についての表示、記録があるか
				(7)	システムの状態表示機能、集中制御機能があるか
				(8)	システム障害時に交替装置、交替機能があるか
				(9)	システム障害発生時に警報機能、通報機能があるか
5.2.4	効率性	1	時間効率性	(1)	ターンラウンドタイム、平均起動時間、入出力回数、ダイナミック　ステップ数は要求目標を満たしているか
				(2)	処理速度
					(2－1)　メモリ資源/入出力資源/ＣＰＵ　の各々に余裕がある時に高速処理を実現するか
					(2－2)　回線処理能力が高い時に高速通信可能か
					(2－3)　特定の機能について高速処理を行えるか（例：コンパイラの最適化機能など）
				(3)	処理能力
					(3－1)　メモリ資源/入出力資源/ＣＰＵ/回線　　の各々に余裕がある時に多重に処理を実現するか
					(3－2)　特定の機能について多重処理を行えるか（例：コンパイラの共用利用や共用ファイル機能）
				(4)	最適化処理

大項目		中項目		チェック項目	
項番	項目名	項番	項目名		
				(4－1)	CPU使用特定に合わせたスケジュールができるか (例1) 入出力バウンドジョブとCPUバウンドジョブのスケジュール方法 (例2) 並行して実行できるジョブについての並行実施
				(4－2)	ユーザ要求、データ量に応じた処理の最適化ができるか (例) コンパイラにおけるユーザソースプログラムのDOループ構造の最適展開の実現 (例) データ量に伴うデータ検索方式の実現
		2	資源効率性	(1)	特定資源に関する効率性
					メモリ、CPU、回線、入出力装置、ファイル、データなどの各々の特定な資源に着目した効率性の追求
				(1－1)	メモリ資源に関する資源効率性
					(a) 必要時に最小必要量をとっているか
					(b) 不要になったらすぐに解放しているか
					(c) 連続領域でなく、不連続領域でもよいように処理されているか
					(d) バウンダリ調整は小さな単位で行っているか
				(1－2)	CPU資源に関する効率性
					(a) 処理上、どこで割り込まれてもよい設計か
					(b) CPUを必要としない時、保持する資源数が最小となっているか
				(1－3)	回数資源に関する効率性
					(a) 相手のプロトコルに対応できているか
					(b) 同一アドレスで多重使用ができる設計になっているか
					(c) 異なるアドレスからの通信に対応できるか
					(d) 必要な時のみ通信を行っているか
				(1－4)	入出力資源に関する効率性
					(a) 必要時に最小資源をとっているか
					(b) 不要になったらすぐに解放しているか
					(c) 同一特性での要求を連続して処理するようにしているか (例：プリンタの書式設定や外字設定を頻繁に変更すると切替オーバヘッドが大きくなる)
					(d) CPU割込みのない、あるいは少ない形での入出力要求を出しているか
					(e) 入出力装置の特性、性能に適合した入出力要求を出しているか (例：バッファリングサイズや回転待ち時間の調整や準備完了までの初期動作時間の調整など)
				(1－5)	ファイル、データに関する効率性
					(a) ブロッキングなどにより、無効データを作らないやり方としているか
					(b) データ圧縮機能により効率を上げる機能があるか
					(c) アクセス頻度の高いデータ、ファイルはメモリ常駐あるいはディスクキャッシュに取り込む使い方ができるか
					(d) 逆に不要なデータ、ファイルはメモリに常駐させない機能 (選択的に可能なこと) があるか
					(e) データ、ファイルの特性に合わせたアクセス機能があるか
5.2.5	保守性	1	解析性	(1)	解析情報の形態
				(1－1)	ドキュメント形式での 解析方法、製品概要、作成年月日、作成者名 の各々が与えられているか
				(1－2)	起きていることの 情報がメッセージ、画像、出力リスト、ジョブなど が与えられているか
				(1－3)	処理の進行状況、処理状態が表示されているか
				(2)	解析情報の質／量
				(2－1)	量は段階的に表示可能であるか (例：コード表示、1行表示、サマリー、詳細といった段階表示が必要)
				(3)	解析情報の表示方法
				(3－1)	ブリンク、赤色などの特徴を持つか
				(3－2)	異常が検知されたら、ただちに出力されるか
				(4)	解析方法／対処方法の提示機能
				(4－1)	起きた事象に対して、次に何をすべきかの指示ができるか
		2	変更性	(1)	顧客ニーズの変化に対応する機能面
				(1－1)	処理要求順序、処理要求範囲の変更に耐力があるか (例：画面表示後にプリント出力するのを逆にしたいといった要求例) (例：入力データが整数だったのが実数になっても処理ができるといった例)
				(1－2)	1つの変更に対して多数箇所を変更させずに済む耐力があるか (例：1変数は1箇所のみで定義されていること)
				(2)	使用資源の変化に耐力があること (例：プリンタに出力していたものを画面表示に変えたい、あるいはその逆の場合)
				(2－1)	使用メモリの変化に耐力があるか (例：メモリサイズが小さくなってもプログラムの変更なく処理が行えるか)
				(3)	使用者の変化に対応する機能面
				(3－1)	同じオペレーションで操作が可能か
				(3－2)	プログラムが変わっても同じ感覚でオペレーションが可能か (例：GUIが一致しているか)
					(一貫性、統一性、標準化とも大きな関係あり)
				(4)	変更性情報の形態
				(4－1)	変更方法が与えられているか
				(5)	オンライン中の変更に耐力があるか (例：プログラム実行中の変更に対応した機能があるか)
				(6)	モジュール分割方法、ソース上のコメントは記載されているか
				(7)	機能とモジュールの対応表はあるか
				(8)	変更に対応したパラメータ化がなされているか
		3	安定性	(1)	情報量の変化に対する耐力
				(1－1)	入力データ量の急激な変化、入力イベントの同時発生、データの性質の変化に対して耐力があるか
				(2)	資源量の変化に対する耐力
				(2－1)	メモリやCPUの使用可能量の変化に対して耐力があるか
				(3)	異常発生に対する耐力
				(3－1)	システム障害発生、入出力障害発生、ミス オペレーションに対する耐力があるか
		4	試験性	(1)	試験機能があるか
				(1－1)	どこを試験されているか表示可能な機能、自己診断機能があるか
				(1－2)	性能表示、試験後解析可能なデータの採取が可能か

大項目		中項目		チェック項目	
項番	項目名	項番	項目名		
					(1－3)　試験方法が統一されているか
					(1－4)　試験により本来の能力に影響が出ないか
					(1－5)　被試験者の能力に影響されないか
					(1－6)　試験の進度が表示されるか
5.2.6	移植性	1	環境適用性	(1)	ハードウェア　インタフェース適応性
					(1－1)　ハードウェアにおけるアーキテクチャ、処理能力、種別、メモリサイズに非依存であるか
				(2)	ソフトウェア　インタフェース適応性
					(2－1)　ソフトウェアにおけるアーキテクチャ、処理能力、アルゴリズム、構造に非依存であるか
				(3)	人とのインターフェース適応性
					(3－1)　利用者の能力、慣れ、教育、知識、常識に非依存であるか
				(4)	扱う情報に関する適合性
					(4－1)　データ属性に非依存であるか(例:コード属性、文字種類など)
					(4－2)　データ構造、ファイル属性、ファイル構造に非依存であるか
				(5)	移植先環境に関する適合性
					(5－1)　文化、要求仕様、仕掛(フール)、言語、法律に対して非依存であるか
		2	移植作業性	(1)	ハードウェア　インタフェースに関する移植作業性
					(1－1)　ハードウェアの変更が必要ないか
					(1－2)　ハードウェアの追加、改造、入替えで移植が可能か
				(2)	ソフトウェア　インタフェースに関する移植作業性
					(2－1)　ソフトウェアの変更が必要ないか
					(2－2)　ソフトウェアの追加、改造、入替えで移植が可能か
				(3)	人に関する移植作業性
					(3－1)　人の変更が必要ないか
					(3－2)　人の追加教育、再教育、変更で移植が可能か
				(4)	扱う情報に関する移植作業性
					(4－1)　情報の変更が必要ないか
					(4－2)　情報の変更(コンバータによる)、情報の改造(人手による論理的対応づけ)で移植が可能か
				(5)	移植物そのものに関する作業性
					(5－1)　移植にあたっての例示があるか
					(5－2)　移植が可能な構造であるか
					(5－3)　理解性のあるプログラムであるか
					(a)一貫性、法則性があるか
					(b)標準化されているか
					(5－4)　テストが可能であるプログラムであるか
					(5－5)　言語が同一であるか
					(5－6)　移植に関するドキュメントがあるか
		3	規格準拠性	(1)	外部仕様に関する規格準拠
					(1－1)　表記法、使用コード、文法、エラーチェック、データ、使用言語に関する規格に準拠しているか
					(1－2)　インストール、ライセンス契約、表示方法に関する規格に準拠しているか
				(2)	内部仕様に関する規格準拠
					(2－1)　表記法、処理方法、モジュール規約、プログラム構成、処理順序に関する規格に準拠しているか
				(3)	主体システムの持つ規格準拠性(説明:ソフトウェアが実現する主体システムに要求される規格に対する準拠性、たとえば、原子炉制御システムならばそれに課された安全規則などに準拠しているか)
		4	置換性	(1)	環境(ハードウェア、ソフトウェア、オペレーション、データ)の変更なく動作が可能か