How Perspective-Based Reading Can Improve Requirements Inspections
視点ベース解釈はどのように要求検査を改善可能か

(前置き)

Perspective-based reading gives developers a set of procedures to inspect software products for defects.

視点ベース解釈は，開発者にソフトウェア製品の欠陥を検査する一連の手順を提供する.

Detecting and correcting these defects early in the development process can save a lot of time and money, and possibly avoid some embarrassment.

開発プロセスの初期段階でそれらの欠陥を検出し，修正することは，

多くの時間とお金を節約でき，そしておそらくいくつかの金銭的困難(厄介な問題)を避けることができる.

——本文ここから——

Because defects constitute an unavoidable aspect of software development, discovering and removing them early is crucial.

欠陥は，ソフトウェア開発のやむを得ない点を構成するため，

欠陥を早期に発見し，除去することは，重要である.

Overlooked defects (like faults in the software system requirements, design, or code) propagate to subsequent development phases, and detecting and correcting them become more difficult.[1]

(ソフトウェアシステムの要件，設計，もしくはコードの欠陥のような)見過ごされた欠陥は，

後続の開発段階に伝播し，そして，欠陥を検出し，修正することをより困難にする.

At best, developers will eventually catch the defects, but at the expense of schedule delays and additional product-development costs.

良くても(せいぜい)，開発者は最終的に欠陥を見つけるだろうが，

スケジュールの遅れやさらなる製品開発コストを犠牲にする.

At worst, the defects will remain, and customers will receive a faulty product.

最悪の場合，欠陥が残ったままになり，顧客は障害のある製品を受け取ることになるだろう.

— 2 段落目 —

Providing complete, consistent, unambiguous, and correct documents throughout the life cycle improves the chances for a higher quality software system.
ライフサイクルを通して，完全に，一貫性があり，正確で正しい文書を提供することは，
高品質のソフトウェアシステムが生じる可能性を向上する.

Therefore, checking software documents for defects before proceeding to the next development phase contributes to overall system quality.
したがって，次の開発段階へ進む前にソフトウェア文書の欠陥をチェックすることは，
システム全体の品質に貢献する.

An inspection by qualified personnel is the best way to accomplish this.
有能な人材(有資格者)による検査は，これ(品質に貢献できる欠陥チェック)を達成するための最良の方法である.

Robert Glass stated that “inspections, by all accounts, do a better job of error removal than any competing technology (that is, inspections tend to find more errors), and they do it at a lower cost (the cost per error found is lower).”[2]
ロバート・グラスは，
「いくつかの競合する技術(すなわち，多くのエラーを見つける傾向がある検査)よりもより良いエラー除去を行い，そして，それらは低コスト(エラー発見あたりのコストがより低い)でエラー除去を行う，すべてのアカウントによる検査」
と述べた.

Most published work shows that inspections are both effective and efficient.[3]
ほとんどの公開された研究は，インスペクションが効果的かつ効率的であると示している.

In particular, inspections in the requirements specification phase[4,5] can catch inconsistent or incorrect requirements for the system before they form the basis for design or implementation, which would necessitate rework.
具体的には，要求仕様の段階におけるインスペクションは，
手直しを必要とする設計や実装のための基礎を形成する前に，システムに対して一貫性のない，もしくは間違った要求を見つけることができる.

— 3 段落目 —
Perspective-based reading provides a set of procedures that can help developers solve software requirements inspection problems.

PBRは，開発者がソフトウェア要求インスペクションの問題を解決するのに役立つことができる，一連の手順を提供する．

PBR reviewers stand in for specific stakeholders in the document (such as designers or testers) to verify the quality of requirements specifications.
PBRの査読者は，要求仕様書の品質を確認するための文書における特定のステークホルダ(設計者，もしくはテスターのような)の代理を務める．

A PBR review ensures that requirements are sufficient to support all the necessary later stages of software development.
PBRのレビューは，
ソフトウェア開発の全ての欠くことのできない後の工程をサポートするのに十分な要求を保証する．

PBR offers several benefits compared with other inspection approaches, and development organizations can customize PBR to fit their needs.
PBRは，他のインスペクションの手法と比較して，いくつかの利点を提供する．
そして，開発組織は，彼らのニーズに合うようにPBRをカスタマイズすることができる．

TECHNIQUES FOR IMPROVING INSPECTIONS　インスペクションを改善するための技術
Proper inspection implementation requires an accurate understanding of the related tasks and organization contexts and of the roles of those conducting the inspection.[5]
適切なインスペクションの実施は，
関連するタスクと組織コンテキストと，それらを実施するインスペクションのルール の正確な理解を必要とする．

Usually, the inspection process has several phases: planning, overview, defect detection, defect collection, defect correction, and follow-up.
通常，インスペクションの工程はいくつかのフェーズがある：
計画，概要，欠陥検出，欠陥収集，フォローアップ

— 2 段落目 —
The defect detection phase is the most significant.
欠陥検出のフェーズは，最も重要である．

Reviewers (typically software developers) read a software document and apply some

technique (formal or informal) to uncover defects.
査読者(通常はソフトウェア開発者)は，ソフトウェア文書を読み，
欠陥を明らかにするために，いくつかのテクニック(公式，もしくは非公式の)を適用する.

The "**Techniques for Reading Requirements Documents**" sidebar provides a brief comparative description of techniques reviewers can use to read requirements documents.
その「**要求文書を読むためのテクニック**」の補足記事は，
査読者が要求文書を読むために利用することができるテクニックの簡単な比較説明を提供する.
※詳しくは最下部にある補足資料を参照のこと

— 3 段落目 —
A complete description of inspections must address five dimensions:[3]
インスペクションの完全な説明は，5つの側面に対処する必要がある：

- technical　技術
- managerial　経営
- organizational　組織
- assessment　アセスメント
- tool support　ツールのサポート

— 4 段落目 —
Choosing to focus on organizational aspects such as the number of participants and the structure and frequency of meetings, many publications[6,7] have neglected the technical dimension—the review techniques and how reviewers actually use them.
参加者の数や構造やミーティングの頻度のような組織的な側面に焦点を当てることを選択するため，
多くの出版物は，技術的な次元を無視する.
すなわち，レビューのテクニックと，査読者が実際にそれらをどう使うのかを無視している.

— 5 段落目 —
However, technical considerations are important, and significant problems in this area need attention.
しかしながら，技術的な考慮事項は，
注意が必要なこの分野において，重要かつ重大な問題である.

While reviewers may know how to *write* software documents, they may have little expertise

in *reading* them.
査読者は，ソフトウェア文書の記述方法を知っているかもしれないが，
彼らは，それらを読むための専門知識をほとんど持っていないかもしれない．

Reviewers typically rely on ad hoc reading techniques, with no well-defined procedure, learning largely by doing and gaining significant expertise only gradually.
査読者は通常，十分に定義されていない手順が原因で，
少しずつ重要な専門知識を得て，間に合わせることによって，おおかたを学ぶ
その場しのぎの読解技術に依存している．

The difficulty in providing training for a poorly defined or undefined process such as an ad hoc review further compounds the problem.
その場しのぎのレビューのような，不十分に定義された，もしくは未定義のプロセスに対して
教訓を提示することの難しさは，さらに深く問題を悪化させる．

For large software projects, improving the review process requires understanding what defects escaped the review and targeting them more effectively.
大規模ソフトウェアプロジェクトのために，
レビュープロセスを改善するためには，
欠陥がレビューを逃れることを理解することと，欠陥をより効果的にターゲットにすること
が必要である．

Letting every reviewer develop a review process makes the communication of effective review strategies more difficult, hampering the widespread dissemination of developed expertise.
すべてのレビュー担当者にレビュープロセスを開発させることは，
開発の専門知識の広範な普及を妨げる，より困難な効果的なレビューの戦略のコミュニケーションを作る．

— 6 段落目 —
A software reading technique provides a concrete set of instructions that explain how to read a software document and what a reviewer should look for.[5]
ソフトウェアの読解技術は，
ソフトウェア文その読み方と，レビュー担当者が探すべきものを説明する，具体的な一連の使用説明書(指示)を提供する．

Reviewers can use these guidelines during the inspection's preparation phase to examine a given software document for defects.
レビュー担当者は,
与えられたソフトウェア文書の欠陥を検査するための，インスペクションの準備(用意，予習，作成)の段階の間，それらのガイドラインを利用することができる.

Rather than leaving reviewers to their own devices, as in ad hoc reviews, software reading techniques collect knowledge about best practices for defect detection into a single procedure.
例えばその場しのぎのレビューなどで,
レビュー担当者に，それら自身のアドバイスを任せるよりはむしろ,
ソフトウェアの読解技術が，単一の手順の中に欠陥検出のための最良の実践についての知識を収集する.

— 7 段落目 —
Researchers at the Experimental Software Engineering Group at the University of Maryland, College Park, created PBR to provide a set of software reading techniques for finding defects in English-language requirements documents.
カレッジパーク，マリーランドの大学の実践的な(経験に基づく)ソフトウェア工学グループの研究者は,
英語の要求文書における欠陥を見つけるために，一連のソフトウェア読解テクニックを提供するための，PBRを作った.

Widely applicable and customizable to particular situations, PBR is not strictly formalized into a definitive set of procedures.
広く応用でき(応用できる範囲が広く)，特定の状況においてカスタマイズ可能なPBRは,
最も信頼のおける一連の手順の中に，厳密に形式化されていない.
(※一般的なレビューの中のどこにPBRを位置付けるかがされていないってこと？)

Rather, using PBR entails
正しくは，PBRの利用は，次のものを引き起こす(伴う，必要とする).

- selecting a set of perspectives for reviewing the requirements document;
  要求文書をレビューするための一連の視点を選択すること

- creating or tailoring procedures for each perspective usable for building a model of the

relevant requirements information;
関連性のある要求の情報のモデルを構築するために使用可能な,
それぞれの視点に対する手順を作成する，もしくは調整すること

- augmenting each procedure with questions for finding defects while creating the model; and
  モデルを作成する間に，欠陥を見つけるための質問と同時に，
  それぞれの手順を増やすこと

- applying the procedures to review the document.
  文書をレビューするために，手順を適用すること

PBR OVERVIEW　PBRの全体像

PBR helps reviewers answer two important questions about the requirements they inspect:
PBRは，
彼らの検査する要求についての，2つの重要な質問に答えるレビュー担当者を助ける.

What information in these requirements should they check?
それらの要求の中の情報の何を，彼らがチェックすべきか？

How do they identify defects in that information?
彼らは情報の中の欠陥をどのように特定するのか？

— 2 段落目 —

Most inspection techniques do not help the reader focus on a particular aspect of the document.
多くのインスペクション技術は，読者が文書の特定の側面に焦点を当てるのを支援しない.

They treat all requirements information as equally important because it all represents real functional requirements and constraints on that functionality.
彼らは，
それら全て(要求情報)は，実在する機能要求と，機能における制約を扱うため，
等しく重要であるとして，全ての要求情報を扱う.

But then document reviewers end up with ill-defined responsibilities for finding all types of defects in the entire document.

とはいうものの，文書のレビュー担当者は，
文書全体における欠陥のすべてのタイプを見つけるために不明確な責任で終わる.

Different perspectives　異なる視点

PBR operates under the premise that different information in the requirements is more or less important for the different uses of the document.
PBRは，要求における異なる情報が，文書の用途が異なるために，多かれ少なかれ重要であるという前提の下で動作する.

Many different people use a requirements document to support tasks throughout the development life cycle.
様々な人が，開発のライフサイクルを通して，タスクをサポートするための要求文書を利用する.

Conceivably, each person finds different aspects of the requirements important for accomplishing a particular task.
考えられる限りでは，それぞれの人は，特定のタスクを達成するために重要な要求の異なる側面に気付く.

Therefore, PBR provides a set of individual reviews, each from a particular requirements user's point of view, that collectively cover the document's relevant aspects.
それゆえに，PBRは，
文書の関連する側面をトータルでカバーする，特定の要求のユーザの視点からそれぞれの，
個々の一連のレビューを提供する.

— 2 段落目 —

PBR requires identifying the users of a specific software artifact (here, the requirements).
PBRは，特定のソフトウェアの成果物(ここでは，要求)のユーザを識別する必要がある.

This process is similar to constructing system use cases, which requires identifying who will use the system and in what way.
このプロセスは，
誰がシステムを使うのか，どのように使うのかを特定することを要求する，
システムのユースケースを作図することと似ている.

This selection of users varies according to organization or project needs.
ユーザのこの選択は，プロジェクトのニーズもしくは組織のニーズに応じて異なる.

In our environment, we identified three major uses of the requirements at later stages of the life cycle:
我々の環境で，我々は，ライフサイクルの後工程における要求の3つの主要な用途を特定した.

- *A description of the customer's needs*.
  The requirements describe the set of functionality and performance constraints that the final system must meet.
  顧客のニーズの説明
  要求は，最終的なシステムが満たすべき一連のパフォーマンス制約と機能要求を記述する.

- *A basis for the system design.*
  The system designer must create a design that can function according to the requirements and within the allowed constraints.
  システム設計のための基礎
  システムの設計者は，制約の範囲内で，要求に応じて機能することができる設計を作成しなければならない.

- *A point of comparison for system test*.
  The system's test plan must ensure that the software correctly implements functionality and performance requirements.
  システムテストのための比較ポイント
  システムのテスト計画は，ソフトウェアが機能要求とパフォーマンス要求を正しく実装していることを確認する必要がある.

— 3 段落目 —
These uses suggest perspectives for reviewing the requirements document:
これらの仕様は，要求文書をレビューするための視点を示唆している:

The designer needs correct requirements with sufficient detail for the major system components under review.
設計者は，レビューの中で，主要なシステムコンポーネントに対して，十分に詳細で正確な要求が必要である.

The tester, concerned about requirements testability, needs to see sufficient detail to construct test plans.

テスト容易性の要求に関心を持つテスト担当者は，テスト計画を構築するために，十分に詳細を理解する必要がある.

The customer (or user) of the system requires that the requirements completely and correctly capture the necessary system functionality.
システムの顧客，もしくはユーザは，要求が完全かつ正確に必要なシステムの機能を取り込んでいることを要求する.

A failure to satisfy any of these needs constitutes a *requirements defect*—a deficiency in the requirements quality that can hamper software development.
これらのニーズのいずれかを満たすための障害が，要求欠陥を構成する.
すなわち，要求品質における欠陥が，ソフトウェア開発を妨げる可能性がある.

— 4 段落目 —
Other perspectives can lead to defect detection as well; the examples we've given simply represent the perspectives we've experienced.
その他の視点は，さらに欠陥検出につなげることができる.
我々が単に与えた例は，我々が経験した視点を表している.

Depending on the environment in which users apply PBR, they may find a different set of perspectives more applicable.
ユーザがPBRを適用する環境に応じて，
それら(の視点)は，より適用可能な一連の異なる視点を見つけることが出来るだろう.

For example, a system expected to have a long operational life span could also be reviewed from a maintainer's perspective (which would be concerned with verifying that requirements are easily extensible).
例えば，長い運用年数の寿命を持つことを期待されているシステムは，
メンテナンスする人の視点から見直されることも可能である.

Deciding on the most appropriate set of perspectives is one way of tailoring PBR to a particular environment.
視点の中で最も適切なセットを決定することは，
PBRを特定の環境に合うように調整する方法の1つである.

— 5 段落目 —

Thus, in a PBR inspection, each reviewer on a team assumes a specific user's perspective.
したがって，PBRインスペクションにおいて，
チームにおけるそれぞれのレビュー担当者は，特定のユーザの視点と仮定する.

The reviewer creates a high-level version of the work products typical of what the user would normally produce.
レビュー担当者は，
ユーザが正常に作成するだろう典型的な，作業成果物の高レベルのバージョンを作成する.

From the designer, tester, and customer perspectives, the relevant work products would be a high-level system design, a system test plan, and an enumeration of the described functionality, respectively.
設計者，テスター，そして顧客の視点から，
関連する作業成果物は，それぞれ，高レベルのシステム設計，システムのテスト計画，そして説明された機能の列挙であろう.

— 6 段落目 —
The objective is to avoid duplicating work during the software development process by creating representations of the system capable of supporting two distinct types of activities.
その目的(作業成果物)は，
活動の異なる2つのタイプをサポートすることができるシステムの表現を作ることによって，
ソフトウェア開発ポロ背巣の間の作業の重複を避けることである.

First, reviewers can use the representations as a basis for the later creation of more specific work products.
最初に，レビュー担当者は，
より具体的な作業成果物の後の作成のための基礎としての表現を利用できる.

Second, reviewers use the representations to analyze how well the requirements can support the necessary tasks.
次に. レビュー担当者は，いかに良く要求が必要なタスクをサポートすることができるか，を分析するための表現を利用する.

— 7 段落目 —
An appropriate level of detail helps reviewers construct the relevant system representations.
適切な詳細レベルは，レビュー担当者が関連するシステム表現を構築するのに役立つ.

Organizations vary these levels to further tailor PBR.
組織は，PBRをさらに調整するために，これらのレベルを変化させる.

When used by more experienced reviewers, the PBR procedures should rely mainly on a reviewer's previous experience in creating design plans, test plans, and user manuals.
より経験豊富なレビュー担当者によって利用された場合，
PBRの手順は，設計計画，テスト計画，ユーザマニュアルを作成する際の，
レビュー担当者の以前の経験を中心に頼るべきである.

Alternately, a reviewer can select a very specific type of representation for each perspective (for example, structured design, equivalence partitioning test plans, and use cases, respectively); more specific procedures are appropriate for less-experienced reviewers.
代わる代わる，レビュー担当者は，
それぞれの視点(例えば，構造化設計，同値分割テスト計画，ユースケースのそれぞれの視点)に対する表現の非常に特定のタイプを選択することができる.

— 8 段落目 —
Using different perspectives offers a number of beneficial attributes:
異なる視点を利用することは，有益な属性の数を提供する:

- *Systematic.　体系*
  Explicitly identifying the different uses for the requirements gives reviewers a definite procedure for verifying whether those uses are achievable.
  要求に対する様々な用途を明確に識別することは，
  レビュー担当者に，それらの用途が達成可能であるかどうかを検証するための明確な手順を与える.

- *Focused.　集中的である*
  PBR helps reviewers concentrate more effectively on certain types of defects, rather than having to look for all possible types.
  PBRは，レビュー担当者が可能性のある全てのタイプを探すよりも，
  より効果的に欠陥の特定のタイプに集中させるのに役立つ.

  A study of PBR's effectiveness[5] showed that this additional focus helped reviewers find more defects than if they used a less structured approach, even though they were concentrating on specific aspects of the document while using PBR.

PBRの有効性の研究は，この追加の焦点が，
レビュー担当者達がPBRを利用している間，文書の特定の側面に集中していたにもかかわらず，
レビュー担当者達がより少数の体系的なアプローチを利用した場合よりも多くの欠陥を発見するのに役立ったことを示した.

One reviewer summarized this benefit by saying,
あるレビュー担当者は，この利益を次のように言ってまとめた.

“It really helps to have a perspective because it focuses my questions.
「それは私の質問に焦点を当てているため，それが視点を持つことは本当に役に立つ.

I get confused trying to wear all the hats!”
(さもないと，)私は全ての帽子を被ろうとして混乱してしまう！」
※体系がないと，何から手をつけていいかわからないから，やたらめったレビューしてしまう.
体系があると，視点に沿ってレビューすればいいので，やるべきことをやれば良い.

Additionally, this focus helps avoid duplicated effort among team members.
さらに，この焦点は，チームメンバ間の重複した努力を避けることに役立つ.

- *Goal-oriented and customizable.　ゴール指向，そしてカスタマイズ可能*
  Reviewers select the perspectives used by PBR to reflect the requirements’ uses.
  レビュー担当者は，要求の用途を反映させるために，PBRが使用する視点を選択する.

  In a new organization, the perspectives change to reflect how that organization uses its requirements documents and the inspection’s specific goals.
  新しい組織において，
  視点は，組織がインスペクションの具体的な目標と，要求文書をどのように利用するかを反映するために変化する.

  Because each perspective gives a specific procedure, reviewers can tailor the procedure to the organization’s needs.
  それぞれの視点は，特定の手順が与えられているため，
  レビュー担当者は，手順を組織のニーズに調整することができる.

  For example, the procedure can be more or less specific based on the reviewers’

expertise.
例えば，手順は，多かれ少なかれ，特定のレビュー担当者の専門知識に基づいている.

- *Transferable via training.　訓練を通じて譲渡可能*
  Because PBR works from a definite procedure, and not the reviewer's own experience with recognizing defects, new reviewers can receive training in the procedure's steps.
  PBRは，欠陥を認識したレビュー担当者自身の経験ではなく，明確な手順から動作するため，
  新たなレビュー担当者は，手順のステップにおいて，訓練を受けることができる.

  Additionally, because PBR uses work products for other life-cycle stages, reviewers can apply their training and experience to tasks that may seem more natural to them.
  さらに，PBRは，その他のライフサイクルの段階に対する作業成果物を利用するため，
  レビュー担当者は，彼らにより自然に見えるだろうタスクの，彼らの訓練と経験を適用することができる.

Identifying defects　欠陥の識別

Once reviewers have created relevant representations of the requirements, they still need to determine what defects may exist.
レビュー担当者は要求の関連表現を作成したら，
彼らは，欠陥が存在するかもしれないかをさらに決定する必要がある.

PBR techniques provide questions tailored to each step of the procedure for creating the representation.
PBRの技術は，表現を作成するための手順のそれぞれのステップに合わせた質問を提供する.

As the reviewers construct the representation, they answer a series of questions about the work products.
レビュー担当者は表現を構築しながら，彼らは作業成果物に関する一連の質問に答える.

Requirements that do not provide enough information to answer the questions usually do not provide enough information to support the user.
質問に答えるために十分な情報を提供していない 要求は，
通常，ユーザを支援するのに十分な情報を提供しない.

Thus, reviewers can identify and fix defects beforehand, so that requirements are ready to

support that task later in the product life cycle.
したがって,
要求が, 製品のライフサイクルの後半のタスクをすぐにサポートすることができるように
レビュー担当者は, 事前に欠陥を識別し, 修正することができる.

— 2 段落目 —

We defined a taxonomy of requirements defects to assure the sufficiency of the questions at each phase, and we developed the PBR questions to detect each type of defect.
我々は, それぞれの段階で十分な数の質問を確保するために, 要求の欠陥の分類を定義し,
そして, 我々は, 欠陥のそれぞれのタイプを検出するために, PBRの質問を開発した.

We based this taxonomy on the IEEE definition of necessary attributes of good requirements documents,[8] and our definition is similar to others that help track requirements defects.[9]
我々は, 良い要求文書に必要な属性のIEEEの定義におけるこの分類を基礎にしており,
さらに, 我々の定義は, 他の人が連続の要求欠陥を助けるのに似ている.

Table 1 lists the types of defects that PBR helps detect.
表1は, PBRが検出を助ける欠陥のタイプのリストである.

In practice, these classifications are not orthogonal; a given defect could conceivably fit into more than one category, depending on the interpretation of the classifier.
実際のところは, これらの分類は, 直交していない.
すなわち, 分類の解釈に応じて,
指定された欠陥は, 考えられる限りでは2つ以上のカテゴリに属することができる.

Nor is this taxonomy definitive; an organization can add other categories, depending on its needs.
この分類は決定的なものではない;
すなわち, 組織は, 組織のニーズに応じて, その他のカテゴリを追加することができる.

— 3 段落目 —

The sidebar, "**A Sample PBR Procedure from the Tester's Perspective**," presents an example of how a tester uses the PBR procedure. (We provide additional instantiations for other perspectives at http://fc-md.umd.edu/reading/reading. html.)
補足記事「テスト担当者の視点からの実例PBRの手順」は,

どのようにテスト担当者が，PBRの手順を利用するのかの例を示している.
(我々は，示したURLにその他の視点に対する追加的な事例を提供する．)

This example includes a series of questions tailored to each step of the procedure for developing an equivalence-partitioning test plan.
この例は，同値分割テストの計画を開発するための手順のそれぞれのステップに合わせた一連の質問を含む.

For example, Part b includes guidelines for identifying test cases for each equivalence set.
例えば，
パートBは，それぞれの同値セットに対するテストケースを識別するためのガイドラインを含む.

A series of questions check for missing information (Qb.1), ambiguous information (Qb.2), inconsistencies between requirements (Qb.3), and incorrect facts (Qb.4).
情報が不足していることをチェックするための一連の質問(Qb.1),
曖昧な情報をチェックするための一連の質問(Qb.2),
要求間の矛盾をチェックするための一連の質問(Qb.3),
誤った事実をチェックするための一連の質問(Qb.4).

The question list does not include defects of extraneous information or other miscellaneous reasons (such as document structure) because we did not think them relevant for this step.
我々は，このステップに対してそれら(無関係な情報～)が関連すると考えていなかったため，
質問のリストは，無関係な情報，もしくはその他の様々な理由(文書構造のような)の欠陥が含まれていない.

— 4 段落目 —
Overall, PBR's detailed questions have the following benefits:
全体的に見れば，PBRの詳細な質問は，次のような利点がある:

- *Allow controlled improvement*:　管理された改善を許す
  Reviewers can reword, add, or delete specific questions.
  レビュー担当者は，具体的な質問を書き換え，追加，削除が可能である.

- *Allow training*:　訓練を許す
  Reviewers can train to better understand the parts of a representation or work product that correspond to particular questions.

レビュー担当者は，特定の質問に相当する表現の一部や作業成果物をより理解するために，訓練することができる.

APPLYING PBR　PBRの応用

PBR does not predicate a specific format for the requirements.
PBRは，要求に対する特定の形式に基礎をおいていない.

We tailored it to requirements that use English (or some other natural language) to describe the system functionality, but users could easily tailor it to suit a formal requirements specification language, such as Software Cost Reduction.
我々は，それ(PBR)をシステムの機能を記述するために，英語を使用した要求に合わせた.
しかし，ユーザは，ソフトウェアのコスト削減のように，
それ(PBR)を簡単に形式要求仕様言語に適合することに合わせることが出来た.

Building on experience　経験の上に築く(に基づいて事を進める)

PBR does not assume that developers already possess the skills for effectively analyzing requirements documents.
PBRは，開発者がすでに要求文書を効果的に分析するためのスキルを保有しているということを想定していない.

Rather, PBR helps developers build on their experience in other phases of software development, letting them use skills they already have (such as for creating designs, test plans, or user manuals) to help them better understand requirements documents.
むしろ，彼らが要求文書をより理解するのを助けるために，
彼らがすでに持っている(設計，テスト計画，ユーザマニュアルを作るためのような)スキルを彼らに使わせることで，
PBRは，ソフトウェア開発のその他のフェーズにおける開発者の経験を当てにする開発者を支援する.

Thus, some reviewers (especially less-experienced ones) may find that PBR appears to be a more natural approach to reviewing requirements than other methods, such as checklists.
したがって，あるレビュー担当者(特に経験の少ない者)は，
チェックリストのような他の方法よりも，PBRは要求を見直すためのより自然なアプローチのように見えると感じるかもしれない.

For this reason, PBR's usefulness may lie in how it trains inexperienced reviewers.

このような理由から，PBRの有用性は，
それ(PBR)を経験の浅いレビュー担当者に，どのように訓練するかにあるのかもしれない．

Of course, reviewers must at least have *some* experience or training with the techniques for creating the relevant work products they'll have to apply (such as creating test plans).
もちろん，レビュー担当者は，
彼らが適用するだろう関連する作業成果物を作成するための技術を使った，
少なくともいくつかのトレーニングや経験がある必要がある．

— 2 段落目 —
An organizational culture accustomed to performing reviews is useful for effective application of PBR.
レビューを実施することに慣れた組織の文化は，PBRの効果的な適用に対して有益である．

Novice reviewers who have no experience with looking for defects are not used to thinking about the effect of faulty requirements on later stages of the life cycle.
欠陥を探す経験のない初心者のレビュー担当者は，
ライフサイクルの後工程における，障害のある要求の影響について考えることに慣れていない．

Therefore, conveying to novice reviewers the importance of providing a definition of a requirements defect sufficiently detailed for identification is difficult.
したがって，初心者のレビュー担当者に，
識別するために十分に詳細化された要求欠陥の定義を提供することの重要性を伝えることは，
難しい．

PBR attempts to provide guidelines in this area, and it uses the defect taxonomy to flag problem areas for reviewers.
PBRは，この領域におけるガイドラインを提供しようと試みており，
さらに，それ(PBR)は，レビュー担当者に対して，問題領域に注意(警告)を与えるために，欠陥の分類を利用する．

But reviewers still need to build up their experience in this area.
しかし，レビュー担当者は，この分野における彼らの経験を増大させる必要がまだあります．

Having reviews in place also means that an organization has probably already dealt with the difficulties inherent in reviews themselves, such as motivating reviewers to perform

satisfactory reviews and scheduling team meetings as part of the development cycle.
組織がすでに

場所でレビューを持つことも、組織はおそらくすでにこのような開発サイクルの一部として満足レビューとスケジューリングチームミーティングを行うためにレビューを動機としてレビューに内在する困難自身を扱ったことを意味します。

— 3 段落目 —
Perhaps the most serious constraint on the use of PBR is the amount of reviewer effort required.
おそらく，PBRの利用における最も重大な制約は，要求されるレビュー担当者の努力の量である.

The improved rate of defect detection comes at the price of a higher amount of effort on the part of the reviewers.
欠陥検出の改善率は，レビュー担当者の一部において，それより高い努力の量に相当する代償を払うという状態になる.

For example, in a study of software professionals,[5] the average review time required for PBR ranged from almost the same as for the professionals' usual approach to 30 percent more, depending on the document being reviewed.
例えば，ソフトウェアの専門家の研究において，
PBRに要する平均レビュー時間は，レビューされた文書に応じて，
専門家のお決まりのアプローチとほぼ同じ(から)30%以上に及ぶ.

However, much of this extra effort actually produces the high-level representations of the system that may save effort at later stages of the life cycle.
しかしながら，この多くの余分な努力は，実際には
ライフサイクルの後期段階で手間を省くことができるシステムの高いレベルの表現を提供する.

For example, the high-level test plan developed during PBR can serve as the basis for the actual test plan used to evaluate the implementation.
例えば，PBRの間に開発された高レベルのテスト計画は，
実装を評価するために利用される実際のテスト計画に対する基礎として役立つことができる.

Lessons learned　教訓

Researchers have conducted studies using more than 150 software engineering students in university classes to evaluate and evolve PBR techniques.[10, 11]
研究者は，PBRのテクニックを評価し，進化させるために，
大学の授業において，150以上のソフトウェア工学の学生を用いた研究を行った.

These students range from undergraduates with little previous review experience to professionals with more than 20 years' experience in industry who were returning for advanced degrees.
これらの学生は，少し以前の学部の学生のレビュー経験から，
学士号より上の学位のために帰国された業界で20年以上経験を持つ専門家に及ぶ.

In 1995 and 1996, researchers from the Experimental Software Engineering Group ran studies using 25 professional developers from the NASA Goddard Space Flight Center.[5]
1995年と1996年に，実験的ソフトウェア工学グループからの研究者は，
NASAゴダード宇宙飛行センターから25人の専門家の開発者を用いた研究を行った.

These developers first applied the requirements review technique they used at NASA, and then were trained in PBR and applied the new technique to a similar requirements document.
これらの開発者は，彼らがNASAで利用した要求レビューテクニックを最初に適用し，
その後，彼らはPBRを学ばされ，同様の要求文書に新しい技術を適用した.

In this way, the researchers could assess how well these professionals performed using PBR compared with how they performed using their usual review technique.
このようにして，研究者は，
これらの研究者が，彼らの通常のレビュー技術を用いて行った方法と比較したPBRを用いて行った方法を
評価することができる.

— 2 段落目 —

These studies supported the notion that PBR leads to improved defect detection rates for both individual reviewers and review teams working with unfamiliar application domains.
これらの研究は，PBRが，
不慣れなアプリケーションドメインを扱う仕事をするレビューチームと個々のレビュー担当者のどち

らに対しても，改善された欠陥検出率をもたらす，という考えを支援する.

These studies also showed that, for a familiar application domain, experienced reviewers sometimes ignore the PBR procedure and revert to using previously acquired heuristics.
これらの研究はまた，馴染みのあるアプリケーションドメインに対して，
経験豊かなレビュー担当者が，時々PBRの手順を無視し，
以前に獲得した経験則の利用に立ち戻る
ということを示した.

Training and reinforcement can overcome this tendency.
訓練と補強(強化)することは，この傾向(以前の経験則の利用)を克服することができる.

— 3 段落目 —
By observing the use of PBR in varied environments by several reviewers, these studies also helped researchers better understand the effects of PBR in different contexts and for different types of users.
それぞれのレビュー担当者による様々な環境において，PBRの使用を観察することによって，
これらの研究はまた，研究者に，異なるコンテキストにおける，そしてユーザの異なるタイプに対して，PBRの影響をよりよく理解させることに役立つ.

For example, PBR seems best suited for reviewers with a certain range of experience.
例えば，PBRは，ある範囲の経験でのレビュー担当者に最もふさわしいようだ.

Reviewers who have previously inspected requirements documents on multiple industrial projects have, over time, typically developed their own approaches and do not benefit significantly from the introduction of PBR.
複数の工業プロジェクトにおける要求文書を前もって検査したレビュー担当者は，
ゆっくり時間を掛けて，大体は彼ら自身のアプローチを開発し，
そして，PBRの導入から大いに恩恵を受けない.

Reviewers who have little experience (those who have never trained, or have trained but never applied their skills on a real project) with the relevant representations (such as designs or test plans) need to receive sufficient training before they can effectively apply PBR.[10]
関連する表現(設計やテスト計画のような)とともに，
少ししか経験がない(訓練したことがない，もしくは訓練は受けているが実際のプロジェクトにおい

て自分のスキルを適用したことがない)レビュー担当者は,
PBRを効果的に適用することができることに先立ち，十分な訓練を受ける必要がある.

This training seems necessary so that the difficulties of creating the representation of the system do not distract from the process of checking for defects.
このトレーニングは,
システムの表現を作ることの難しさが，欠陥を検査する工程から遠ざけないように,
必要であると思われる.

— まとめ？(最初の文字が大きくなっていた) —
PBR provides a framework that represents an improved approach for conducting requirements reviews.
PBRは，要求のレビューを実施するために改善されたアプローチを表すフレームワークを提供する.

However, such an approach will only be effective when an organization tailors the framework to its own needs and uses feedback from its reviewers to continually improve and refine the techniques.
しかしながら，そのようなアプローチは,
組織が，組織自身のニーズのためのフレームワークを調整し,
継続的に技術を改善し，洗練するために，その組織のレビュー担当者からのフィードバックを利用する
時にのみ，有効になるだろう.

— 2 段落目 —
Studies have shown that development teams that use PBR to inspect requirements documents tend to detect more defects than they do using other, less-structured approaches.
研究は,
彼らがその他のアプローチ，あまり構造化されていないアプローチを利用することよりも,
より多くの欠陥を検出する傾向のある，要求文書を検査するためのPBRを利用する
開発チームを示している.

Although PBR requires more effort, it offers a number of advantages compared with current practices.
PBRは，より多くの努力が必要であるが，PBRは，現在の慣行と比較して多くの利点を提供する.

Relatively novice reviewers can use PBR techniques to apply their expertise in other development tasks to defect detection.
比較的に，初心者のレビュー担当者は，欠陥検出のためにその他の開発タスクにおいて，彼らの経験を適用するために，PBRの技術を利用することができる.

Using PBR improves team meetings by helping team members build up expertise in different aspects of a requirements document.
PBRを使用することは，
チームメンバーが，要求文書の様々な側面における専門知識を増大させることを支援することによって，チームミーティングを改善する.

It creates high-level representations of the software system, usable as a basis for work products in later stages of development.
(PBRを使用することは，)開発の後期段階における作業成果物に対する基礎として使用できるソフトウェアシステムの高レベルの表現を生成する.

Each development organization can customize PBR's set of perspectives, level of detail, and types of questions.
それぞれの開発組織は，PBRの一連の視点，詳細レベル，および質問のタイプをカスタマイズすることができる.

PBR facilitates controlled improvement, providing a definite procedure, alterable according to project metrics and reviewer feedback.
PBRは，成果物のメトリクスと，レビュー担当者のフィードバックに照らして変更可能な明確な手順を提供し，制御された(管理された)改善を容易にする.

【補足資料】(背景が紫で示してある資料)

Techniques for Reading Requirements Documents

Several techniques exist for individually reviewing requirements documents.

At one extreme is the *ad hoc* review, a review with no formal, systematic procedure, based only on individual experience.

A *checklist* review makes the inspection process slightly better defined by providing reviewers with a list of items on which to focus.

Defect-based reading provides a set of systematic procedures that reviewers can follow, which are tailored to the formal software cost reduction notation.

Like DBR, *perspective-based reading* is a scenario-based technique that provides procedural guidance, tailored to requirements expressed in natural language (for example, English).

— 2 段落目 —
Table A presents some characteristics of these techniques.

We present the requirements languages for which each technique is usable and compare them according to the following criteria:

- *Is systematic.*
  Are the specific steps of the individual review process definable?

- *Is focused.*
  Must different reviewers focus on different aspects of the document?

- *Allows controlled improvement.*
  Based on feedback, can reviewers identify and improve specific aspects of the technique?

- *Customizable.*
  Can reviewers customize the technique to a specific project or organization?

- *Allows training.*
  Can reviewers use a set of steps to train themselves in applying the technique?

Table A Characteristics of requirements reading technique.

Technique	Requirements language	Systematic?	Focused?	Controlled improvement?	Customizable?	Training?
Ad hoc	Any	No	No	No	No	No
Checklist	Any	Partially	No	Partially	Yes	Partially
Defect-based	Software	Yes	Yes	Yes	Yes	Yes

reading	Cost Reduction (formal notation)					
Perspective-based reading	Natural language	Yes	Yes	Yes	Yes	Yes

---

（欄外のコメント）

Because PBR works from a definite procedure, and not the reviewer's own experience with recognizing defects, new reviewers can receive training in the procedure's steps.

A Sample PBR Procedure from the Tester's Perspective

For each requirement, generate a test or set of test cases that let you ensure that a system implementation satisfies the requirement.

Follow the procedure below to generate the test cases, using the questions provided to identify faults in the requirements.

General questions

Read each requirement once and record the number and page along with the inputs to the requirement.

- Q1. Does the requirement make sense from what you know about the application or from what is specified in the general description?
- Q2. Do you have all the information necessary to identify the inputs to the requirement? Based on the general requirements and your domain knowledge, are these inputs correct for this requirement?
- Q3. Have any of the necessary inputs been omitted?
- Q4. Are any inputs specified that are not needed for this requirement?
- Q5. Is this requirement in the appropriate section of the document?

Part a: Building equivalence sets.

For each input, divide the input domain into sets of data (called equivalence sets); all values in each set will cause the system to behave similarly.

Determine the equivalence sets for a particular input by understanding the sets of conditions that affect the requirement's behavior.

You may find it helpful to keep the following guidelines in mind when creating equivalence classes:

- If an input condition specifies a range, at least one valid (the set of values in the range) and two invalid equivalence sets (the set of values less than the lowest extreme of the range, and the set of values greater than the largest extreme) are defined.

- If an input condition specifies a set's member, at least one valid (the set itself) and one invalid equivalence set (the valid set's complement) are defined.

- If an input condition requires a specific value, then one valid (the set containing the value itself) and two invalid equivalence sets (the set of values less than and the set greater than the value) are defined.

Each equivalence set should be recorded under the appropriate input.

- Qa.1. Do you have enough information to construct the equivalence sets for each input?

  Can you specify the boundaries of the sets at an appropriate level of detail?

- Qa.2. According to the information in the requirements, are the sets constructed so that no value appears in more than one set?

- Qa.3. Do the requirements state that a particular value should appear in more than one equivalence set? (That is, do they specify more than one type of response for the same value?)

Do the requirements specify that a value should appear in the wrong equivalence set?

Part b: Testing equivalence sets.

For each equivalence set, write test cases, and record them with the associated equivalence set.

Select typical test cases as well as values at and near the boundaries of the sets.

For example, if the requirement expects input values in the 0 to 100 range, the test cases selected might be 0, 1, 56, 99, and 100.

Finally, for each equivalence set, record the expected resulting behavior.

(That is, how do you expect the system to respond to the test cases you just made up?)

- Qb.1. Do you have enough information to create test cases for each equivalence set?
- Qb.2. Are there other interpretations of this requirement that the implementer might make on the basis of the description given? Will this affect the tests you generate?
- Qb.3. Is there another requirement for which you would generate a similar test case but would get a contradictory result?
- Qb.4. Can you be sure that the tests generated will yield the correct values in the correct units? Is the resulting behavior specified appropriately?

Table 1 Requirements defects that PBR helps detect.

Missing information	Any significant requirement related to functionality, performance, design constraints, attributes, or external interface not included Undefined software responses to all realizable classes of input data in all realizable classes of situations

	Sections of the requirements document Figure labels and references, tables, and diagrams Definitions of terms and units of measure
Ambiguous information	Multiple interpretations caused by using multiple terms for the same characteristic or multiple meanings of a term in a particular context
Inconsistent information	Two or more requirements that conflict with one another
Incorrect fact	A requirement-asserted fact that cannot be true under the conditions specified for the system.
Extraneous information	Unnecessary or unused information (at best, it is irrelevant; at worst, it may confuse requirements users)
Miscellaneous defects	Other errors, such as including a requirement in the wrong section

---

（欄外のコメント）

PBR leads to improved defect detection rates for both individual reviewers and review teams working with unfamiliar application domains.