ソフトウェア工学　94－2
（1993. 9. 21）

Quality and Process Improvement Database (QPID)

新谷　勝利
岡田　光隆
松原　雅美
岡本　順子
日本アイ・ビー・エム（株）

要約

ソフトウェア開発において「品質と生産性の向上」は永遠の課題である。筆者等は開発中のデータをいかに効果的に活用するかがキーであると考え、今までの経験をベースにＱＰＩＤというシステムを開発し運用している。当報告はＱＰＩＤの設計方針、システムの構成等を解説するものである。

Quality and Process Improvement Database (QPID)

Katsutoshi Shintani, Mitsutaka Okada, Masami Matsubara, Yoriko Okamoto
Asia Pacific Products, IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken, 242 Japan

Abstract

With a growing need in our software development where our efforts will be effectively concentrated on improving both quality and productivity, a set of tools we use at various levels of organizational structures and at various stages of our development life cycle has been integrated around a set of databases so that not only in-process data, but also after-ship data including financial data are accumulated for managing our development activities. In this paper, the authors explain how this system, named QPID for Quality and Process Improvement Database, is developed with its objectives and contents. This system has been in use since last summer at our development, and project reviews by management are using the output reports from QPID.

Keywards

Software development process, quality, productivity, project management

はじめに

Ｆ．ブルックスがソフトウェア開発に伴う諸問題を一冊の本[1]にまとめたのは１９７４年であり、その後のソフトウェア工学の進歩あるいはその実際への適用をまとめ、今後への示唆を与える論文[2]を発表したのは１９８４年である。この著書で述べられているソフトウェア開発の古典的問題（品質、コスト、スケジュールなどの諸問題）は現在でも簡単に解決する方法は無く、やはり地道な研究、活動が重要と思われる。

当小論は、問題を解決する方法を指摘するものではなく、ソフトウェア開発過程で発生するデータを収集、分析し、活用することにより何処に問題があるのかを明確にし、さらにプロセスの改善、品質、生産性を向上させる目的で構築されたシステムを説明するものである。筆者は当システムをQuality and Process Improvement Database (QPID)と命名した。

ＱＰＩＤのシステム構成

現在のＱＰＩＤは、使用者別にＱＰＩＤ１からＱＰＩＤ４までの４つのツール群に分けられている。

ＱＰＩＤ１
　　一般の開発者および協力会社の方々用
ＱＰＩＤ２
　　開発責任者、プロジェクト・リーダー用
ＱＰＩＤ３
　　製品企画者用
ＱＰＩＤ４
　　複数のプロジェクト上位管理者用

以下にこれらの各ツール群の機能を紹介する。

ＱＰＩＤ１

ＱＰＩＤ１には以下の６つのツールがあり、一般開発者および協力会社の方が個々のデータを扱うための機能をもっている。これらのツールは、データ収集と共に、開発の効率を上げることを目的にしている。以下に、ＱＰＩＤ１の各ツールの概略を述べる。

ＲＩＤ（レビュー／インスペクション・データ）
　　ＲＩＤはデザイン、コード、テストプラン、テストケース、出版物のレビューおよびインスペクションのミーティング情報および障害を記録し管理するためのツール

ＤＣＲ（設計変更要求）
　　ＤＣＲは開発を通じて発生する設計変更要求を記録し管理するためのツール

ＰＴＭ（プログラムの障害管理）
　　ＰＴＭは開発を通じて発生するプログラムの障害を記録し管理するためのツール

ＴＣＬ（テストケース・ライブラリー）
　　ＴＣＬは次に示す機能を持つ、テストケース管理のためのツール
- テストケースの作成
- テスト結果の入力
- テストの進捗の把握

ＤＣＬ（文書管理）
　　ＤＣＬは開発中に作成される文書を管理し、承認依頼、承認処理を行うためのツール

ＤＰＳ（再発防止システム）
　　再発防止活動とは、開発中に発生した障害および製品出荷後のお客様からの苦情などを原因分析し、同種の問題を再発させないための施策を実施することである。ＤＰＳは、原因分析の結果や、様々な施策に関するデータを一元管理し、知識を共有することにより再発防止活動を支援するためのツール

---

[1] Ｆ．ブルックス，「ソフトウェア開発の神話」（日本語訳），企画センター

[2] Ｆ．ブルックス，「ソフトウェア開発に王道なし」（日本語訳），Unix Magagine 1988.5 p.87-99

ＱＰＩＤ２

ＱＰＩＤ２は主として開発管理／品質管理の各機能から構成されている。ＱＰＩＤ２は、開発責任者あるいはプロジェクト・リーダーがプロジェクト全体のデータを管理するための機能を有している。以下に、ＱＰＩＤ２の各機能の概略を述べる。

「開発管理」

開発費見積り
　　開発における種々の開発費の予測および実績を管理するための機能
サイズ見積り
　　プログラム・サイズおよび出版物のページ数などの予測および実績を管理するための機能
開発キー・デート
　　開発スケジュールを最初のプラン、現在のプランおよび実データに分けて管理するための機能
工程別進捗管理
　　各工程ごとの作業項目の進捗度および作業予定、発生した問題および解決状況を管理するための機能
使用ツール／手法一覧
　　プロジェクトの開発中に採用するツール／手法名およびその効果を記録する。プロジェクトの開始時に、他のプロジェクトにおけるツール／手法の使用実績が参照するための機能

「品質管理」

設計段階における障害数の目標と実績
　　レビューおよびインスペクションで発生する障害の目標と実績を管理するための機能
テスト段階における障害数の目標と実績
　　開発中のテスト段階で発生する障害の目標と実績を管理するための機能
製品出荷後の障害情報
　　製品出荷後に発生する障害の目標と実績を管理するための機能

「レポート機能」：ＱＰＩＤ１、ＱＰＩＤ２のデータをもとに各種のレポートを出力する。

ＱＰＩＤ３

ＱＰＩＤ３は製品企画者が製品の企画データおよびビジネス・データを扱うための機能。ＱＰＩＤ３は以下の機能をもつ。

- 製品情報
- 責任者／組織一覧
- 売上げ予測／売上げ実績
- ビジネス・キーデート

ＱＰＩＤ４

ＱＰＩＤ４は上位管理者が複数のプロジェクトを総括して管理するための各種のレポートをＱＰＩＤ１、ＱＰＩＤ２、ＱＰＩＤ３のデータをもとに出力する。

ＱＰＩＤ１の設計方針

ここでＱＰＩＤの設計方針について述べる。

ＰＴＭの設計方針

筆者はソフトウェア開発の諸問題解決のアプローチを決めるためには、開発過程の各工程で発生するデータを収集蓄積し、それを分析し、問題があることを認識することから始めなければならないと考えた。この時対象となるデータは開発者にとってその原因はともかく状況そのものが自明であるものが最初に選ばれた。即ち、一般的にバグと称されているものである。

今日でも多くの開発者は、自分の作成したコードを自分あるいは第三者がテストした時に判明する障害をバグ・レポートとして文書化した時に「問題の存在」を認識している。多くの開発者はこの時点で自分のコードを幅広く見直す。従って、このバグ・レポート（筆者はその対象範囲を考え「問題レポート」と称している）は開発者がコードを見直す時、その助けとならなければならないと考えた。

同様に同種の障害を再発させないためのデータとなるように、原因は何で、その原因にどうアプローチし、どのような修正を加えたかを可能な限り記述できるように考えた。単純なコーディング・エラーでさえもそれを引き起こしたのには理由があり、ましてや設計の基本的な問題に起因する場合、その理由を詳述しておかなければ、後日修正した本人のみならず、第三者が修正されたコードから「問題の本

質」を理解するのは至難の技であるからである。これを支援するシステムが問題レポート支援システムであり、今日のＱＰＩＤ１の原点となっている。

ＲＩＤおよびＤＣＲの設計方針

問題レポートを分析して判ることの1つに開発者が「自分の起こしたバグではない」というものが多いことがある。これを non-defect-oriented problem (NDOP) と称し、そうではない defect-oriented problem (DOP) と区別している。NDOPが全体の過半数になることは一般的に見られる現象である。コーディングの世界はシンタックスは当然のことセマンティックスも明確なものであるのに、何故このような現象が見られるのであろうか。テストした人はコードを実行した結果、障害と認識したのにである。テストした人の期待は何をベースにしたのか？一つは使用者としての漠然としたものであり、他はコードがそのベースとした設計書、あるいは設計書がそのベースとした仕様書にまで遡ることになる。前者は使用容易性等の様々な要因による設計変更要求に関する支援システムの必要性を、後者は仕様書、設計書に対するレビューおよびインスペクションに関する支援システムの必要性を示唆している。ＱＰＩＤが問題レポート支援システムを拡張して設計変更、レビュー／インスペクション支援システムを作るに至ったのはこの点にある。

ＴＣＬの設計方針

更にテストする人の期待値を明確にするためにテストケース作成において、入力ー処理ー結果を様式化して記述できるようにした。このことはコードの適切性を管理すると同様にテストケースの適切性も管理できるように拡張したことになる。開発者はテストケースをテスト実施前にレビューし、コードがそのような挙動をするかどうかを確認できる。

ＤＣＬの設計方針

ＲＩＤ（レビュー／インスペクション）やＤＣＲ（設計変更）は、仕様書や設計書に対する参照、修正およびその承認を伴う。この仕様書や設計書を登録、削除、承認依頼および承認処理をシステムで支援する必要がある。文書管理システムを作るに至ったのはこの点にある。

ＤＰＳの設計方針

再発防止活動とは、エラー修正をした後にどのような歯止めを考え、対策を実施するかを検討し、実施することである。１つ１つのエラー修正をした直後より、ある程度の数が集まった時に関連する複数の人間が集まってエラーを分析して、根本原因にまで遡る反省会を実施することが有効な再発防止活動と考えられる。種々の観点からエラー・データの分析を可能にする支援システムが問題レポート支援システムの拡張機能として作られる必要がある。

以上のように、ＱＰＩＤ１は問題レポート支援システムの拡張という観点から設計された。

ＱＰＩＤ２の設計方針

ソフトウェア開発において、プロジェクト管理の側面から要求されることとして開発管理と品質管理がある。筆者等は開発責任者、プロジェクト・リーダーが自分のプロジェクトを効果的に管理するためのツールとしてＱＰＩＤ２を設計した。

開発管理では、開発の進捗状況、問題点などを管理できるようにし、品質管理では開発中および開発後の障害数の目標および実績を各工程で管理できるようにしている。

開発管理および品質管理の実績データとして、ＱＰＩＤ１の各機能で収集されたデータは統計処理され、ＱＰＩＤ２のレポートとして利用される。また、製品出荷後の実績データは他システムのデータベースから搬入できるようにした。

以下にＱＰＩＤが提供している障害の管理(目標と実績)のグラフの例を示す。

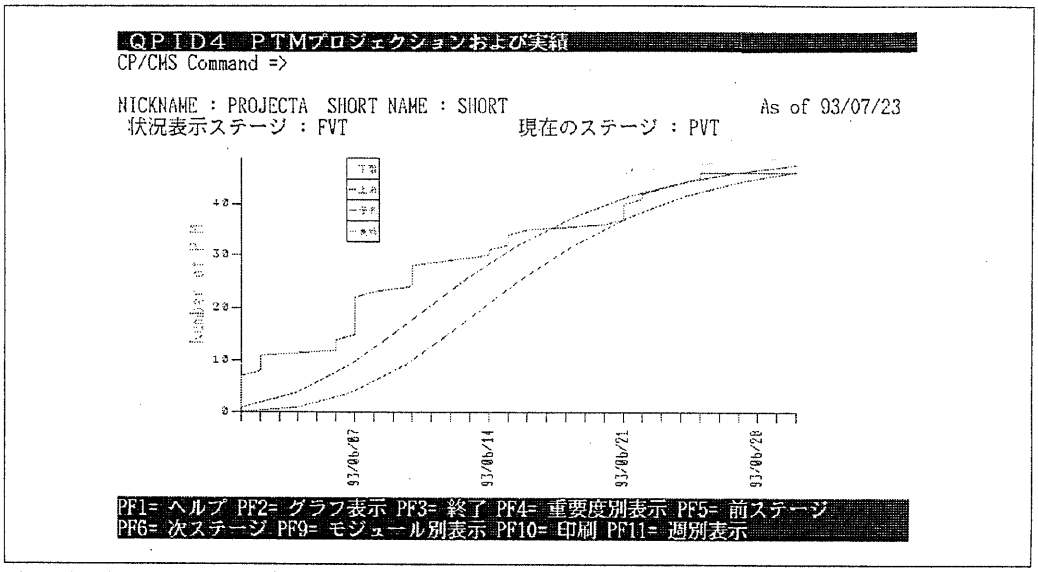

図 1. 障害管理曲線（例）

４つの曲線で管理される。目標の上限曲線、下限曲線、成長曲線そして実績の曲線から表現される。実績の曲線が目標の上限と下限の間にあることが条件で、その範囲を越えた時は何らかの対策を講じる必要がある。

ＱＰＩＤ３の設計方針

ソフトウェア開発で発生するデータとして上記のＱＰＩＤ１およびＱＰＩＤ２で得られるデータ以外にビジネス・データがある。このデータは一般には、製品企画者によって管理され、上位管理者に利用される。

このビジネス・データ（販売予測、売上げ実績、ビジネス・キーデートなど）の管理を支援し、ソフトウェア開発中のデータと結び付ける役割をＱＰＩＤ３で実現できるようにした。

ＱＰＩＤ４の設計方針

複数のプロジェクトを管理する上位管理者には、各プロジェクトの進捗状況、問題点、売上げ、出荷後の障害数など種々のデータが要約されて把握できると共に、プロジェクトの診断情報もシステムで生成できることが望ましい。以上を目的としてＱＰＩＤ４は設計されている。

統合的データベース・システム

データの一元管理

データを長期に渡り蓄積し、活用していくためには、データを一元的に管理し、どのプロジェクトのデータも他のプロジェクトのデータと比較、統計的処理および分析ができることが要求される。

ＱＰＩＤが各層のレベルの人に「自分の問題解決への支援システム」として使用されるためには、論理的拡張性のみでは不充分である。各々の局面でのデータの収集、蓄積、分析支援のみだけでなく、統一的なデータベース・システムとして設計構築されなければならない。

データベースの設計には次のように考慮した。ソフトウェア開発者にとっては詳細レベルのデータと、集約されて分析が容易にできるようにされたデータの双方のデータが必要である。また、プロジェクト・リーダーあるいは開発責任者、上位管理者、更には品質保証を担当するグループへの支援も可能なようにデータベースの構造が階層化されている必要がある。

動作環境

ＱＰＩＤデータベース

ＱＰＩＤがデータを長期に渡り蓄積していくＱＰＩＤ累積データベースはホスト・システム（ＶＭ）上に置いている。これは、筆者等のネットワークが全世界的規模でＶＭ上でサポートされており、ＶＭを経由して全ての使用者がデータの入出力が可能であるためである。

ＱＰＩＤ１の動作環境

ＱＰＩＤ１の動作環境としては、現在ＶＭとＯＳ／２をサポートしている。ＱＰＩＤ１は一般開発者および協力会社の方々によって使用されるツール群であるため、理想的には、開発に使用しているオペレーティング・システム上で稼働することが望ましい。現在は、ＶＭとＯＳ／２以外の環境に対しては、バッチ・インターフェースによるサポートをしているが、今後は各オペレーティング・システムのサポートが必要である。

ＱＰＩＤ２、ＱＰＩＤ３、ＱＰＩＤ４の動作環境

ＱＰＩＤ２、ＱＰＩＤ３、ＱＰＩＤ４の利用者は開発責任者、上位管理者、製品企画者であり、作業環境としてＶＭを使用している。このため、これらのツール群の稼働環境をＶＭとした。しかし、グラフ，表などはワークステーション上で表示する方がパフォーマンス、コストなどの点で効率が良いことから、今後の拡張を考えている。

以上の設計方針をふまえたＱＰＩＤのシステム構成を次に図示する。

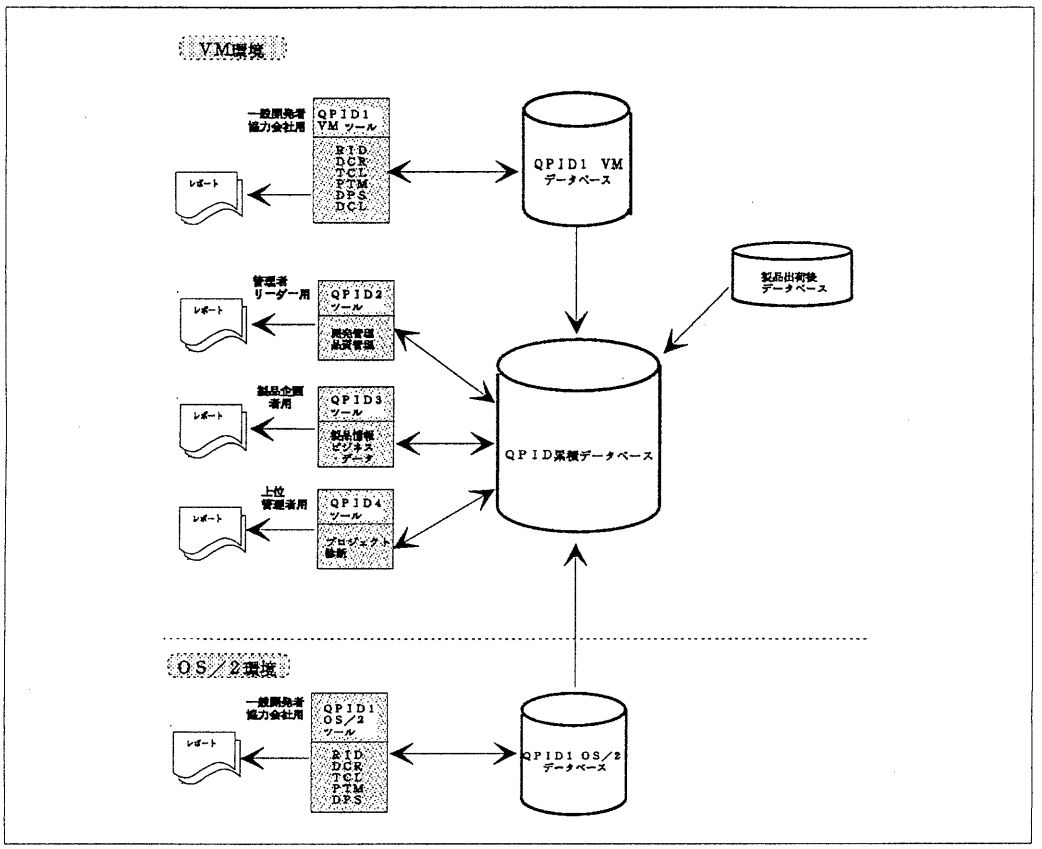

図 2. ＱＰＩＤ１ システム構成

ＱＰＩＤプロセスと品質確認プロセス

筆者等は、ＱＰＩＤとして４つのツール群とデータベースを提供すると共にＱＰＩＤプロセスと品質確認プロセスという２つのプロセスを提供している。

ＱＰＩＤプロセス

当部門では、「ソフトウェア開発プロセス」を規定し、それに基づいてソフトウェアを開発している。ＱＰＩＤはツールの集まりであり、各ツールがソフトウェア開発の中で有効にかつ効果的に使用されるためには、各ツールが「ソフトウェア開発プロセス」で述べられている各活動項目のどれに利用され、「誰が」「どの時点」で「どのような目的と役割」で使用するのかを明確に示すことが重要だと考えた。さらに、ツール間のデータの流れ、オペレーションの流れを図示することにより、より理解しやすいようにした。

ここに、ＱＰＩＤ１の中の問題レポート支援システム（ＰＴＭ）の処理手順を一例として図3に示す。問題発生から解決までの手順は次の通りである。

ステップ　1. ＰＴＭ新規作成ーテスト結果に同意できない場合、１事象１つのＰＴＭを新規に作成する。

ステップ　2. ＰＴＭ回答ー不具合に対する開発者の見解、対処法を対応するＰＴＭに答える。

ステップ　3. ＰＴＭ終了ーＰＴＭを作成した者がそれに対する開発者の回答内容と修正結果を検証し、同意すればそのＰＴＭは終了する。

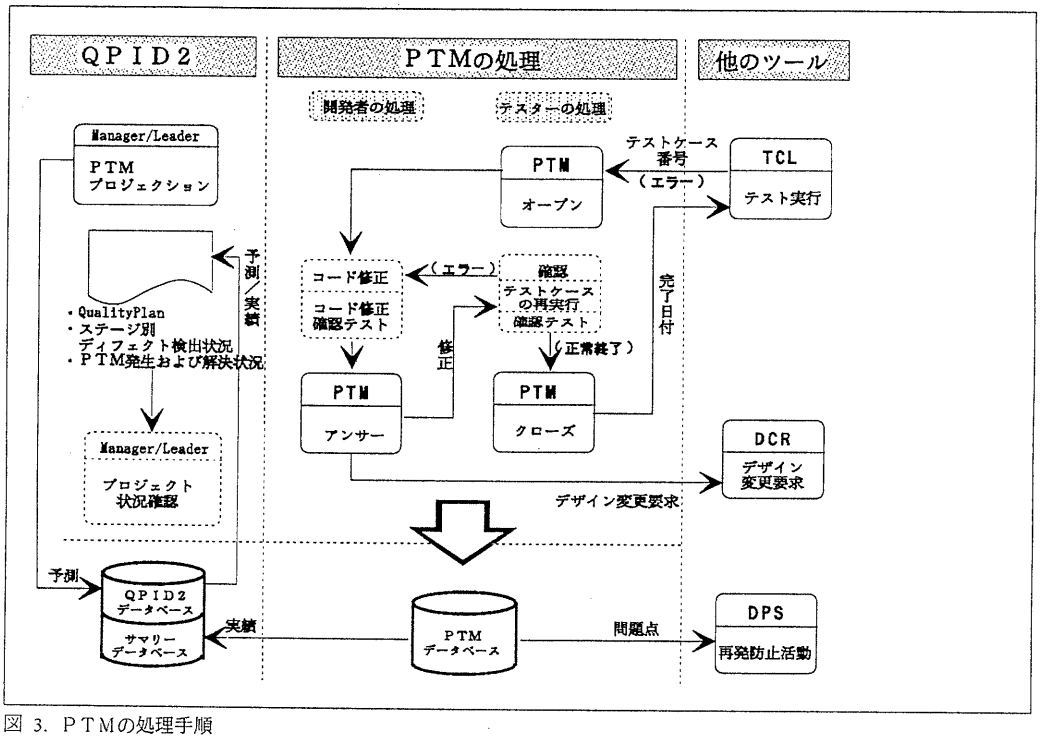

図 3. ＰＴＭの処理手順

さらにＱＰＩＤプロセスでは、ツールの処理手順と共に、関連する他のツールとのデータおよび処理の流れを解説している。図3に示すように、必要に応じて設計変更または再発防止活動の処理を実施することになる。また、開発責任者はＱＰＩＤ２でＰＴＭの状況を参照することができる。

品質確認プロセス

品質を目標と実績で管理し、品質をソフトウェア開発中に作り込んでいくためには各工程でその品質の達成度を確認する必要がある。このために、筆者等はどのＱＰＩＤのデータを使用し、いつ、誰が確認を実施するかを明確にした品質確認プロセスを提供し、適用することを推進している。

ＱＰＩＤの今後

現在、ＱＰＩＤの推進・運用により、データの収集／分析／活用が達成できる方向にある。また、ＱＰＩＤプロセス、品質確認プロセスの適用により、プロジェクトの開発管理／品質管理がＱＰＩＤを中心に実施される方向にある。ただし、普及・定着度合いはまだ不十分であり、今後の課題として次のことを挙げられている。

- 宣伝活動（教育の実施）
- データ分析およびフィードバックの強化
- 要望の迅速な反映

おわりに

「自分の慣れた開発方法を変えたくない」、「スケジュール優先で、データの収集・活用の暇がない」と多くのソフトウェア開発者は思っているのではないであろうか？このメンタリティを自らの意志で変えない限り、ソフトウェア開発現場は、「職人芸」の域をでることはないであろう。品質および生産性を向上させるうえで「個人の技術」に依存するのではなく、プロジェクトの経験・知識の共有化を図ることによって、重複した作業をなくし、同じ過ちを繰り返さないことが重要である。

ソフトウェア開発過程の測定方法については、今後もさらに研究されるであろう。分析およびそれを通して根本原因にまでさかのぼることが、測定そのものより重要である。開発の工程を経る毎に形状を変えるソフトウェアの場合、ある時点の測定値から前の工程へと自動的にバックトラック出来る状況になるのは長い期間が必要となろう。ソフトウェア・メトリックスの今後の進展を期待すると共に、今できることは実行すべきと考える。ＱＰＩＤは当分の間、今の有用性に変化はないであろう。問題は、最初に述べたようなメンタリティをどう変えてゆくかであろう。