RUP における 10 の重要な要素: 効果的な開発プロセスの重要な要素

Leslee Probasco

Rational Software ホワイト・ペーパー

TP177, 9/00

Rational®
the software development company

目次

要約

Rational Unified Process[1] (RUP という名前で知られています) などのソフトウェア開発プロセスを効果的に適用するには、まずその主要な目的について理解し、それぞれが重要である理由と、開発チームが利害関係者の「真の」ニーズを満たす高品質な製品を開発する際にこれらを結びつけて役立てる方法を理解する必要があります。

支援の要請

ある日の夕方、隣人のランディーがやってきて支援を求めました。彼は週末に教会のグループとキャンプに出かける準備をしていて、この旅行に持っていくためにどんな「ギア」[1]を荷物に詰めたらいいのかを決めようとしていました。既に衣類と用具の一部は用意してあり、さらにいくつかのものを買うつもりのようでした。彼は「ギア・リストを貸していただけませんか?」と聞いてきました。

私のギア・リストということでしょうか。どうやら、そのようです。以前に私がバックパッキングと登山の準備をしているのを彼は見ていたようです。彼は私が自然ツアーを率いたり参加した経験が豊富であることを知っていて、手持ちの用具と衣類のリストを見ながら限られたスペースに荷物をてきぱきと収納したことにとても感心していました。そういう理由で、彼は私のギア・リストを貸してほしいと要請してきました。

私のギア・リストを貸すのはかまいませんが、彼にはあまり役に立たないと思いました。なぜでしょうか。そのリストには彼に必要なすべてのものが記載されていないからでしょうか。

本当のところ、彼に必要だと思われるいくつかのものは、私のリストに記載されているもので代用できましたが、その旅行に実際に必要なものはリストにありませんでした。例えば、彼の靴のサイズが私に比べて大きいだけでなく、彼が好む食品の種類 (それにその量) は、私が持っていくために選んだものと異なるのは間違いありません。

詳細さに圧倒される

私の旅行には、バックパッキングや登山から、スキー、雪道のトレッキング、アイス・クライミング、カヤックまでさまざまな目的があり、旅行日程も 1 日のものから何日もかかる遠征まであるため、私のアウトドア・ギア・リストには実際に何百もの項目があります。また、ある団体のために旅行を率いる場合 (旅行参加署名シート、権利放棄申請書など) と、仲間の何人かでちょっと大自然に出かけるだけの場合では持っていくものが異なります。私が主に心配しているのは、彼がリストの数多くの項目を苦労して調べても、その比較的簡単な旅行に本当に必要なものを見つけることはできないだろうということでした。

ランディーは頭に来て準備を断念するかもしれません。そうすれば、出発の時間までに旅行の準備を整えることはできないでしょう。では、彼はどのようにして持っていくものを決めることができるのでしょうか。彼は先週荷造りを始めました。インターネットを広範囲に調べ、素敵な新しいブーツ、ジャケット、衣類を購入しました。 荷物は既にいっぱいです。

しかも食料はまだ何も荷造りしていません。水も持っていく必要はないのでしょうか。何かがうまくいかなかった場合はどうするのでしょうか。迷ってしまう場合もあります。あるいは、誰かが怪我をした場合にはどう対処するのでしょうか。

重要な要素に的を絞る

ランディーが既に荷造りを終わったものを整理してみると、大自然を旅行するために必要なものについてバランスよく把握していないことがすぐわかりました。

---

[1] 「ギア」は、登山家やその他のアウトドア・スポーツを楽しむ人々が使用する用語で、さまざまなスポーツに使用するすべての装備、道具、衣類、履物、またはその他の製品を表します。RUP も多くの成果物があります。「ギア」については、www.mgear.com を参照してください。

彼に「10 個の必需品が揃っていますか?」とたずねました。彼は次のように答えました。「10 個の必需品? 何ですか、それは?」

そこで私は「必要なものをここに書き出してみます。」と言い、白紙を取り出して一番上に「10 個の必需品」と記入しました。記入した 10 の項目は次のとおりです。

1. 地図
2. 磁石
3. サングラスと日焼け止め
4. 予備の衣類
5. 予備の食料と水
6. ヘッドランプ
7. 救急箱
8. 発火装置
9. マッチ
10. ナイフ[2]

「短いリスト」から開始

これだけでしょうか。そうです。これで終わりです。これらの 10 個の必需品から始め、それぞれの領域に当てはまるものを確認すると、そのほかに必要なものが決まってきます。もちろん、その旅行に合わせてこれらの「必需品」それぞれを調整しますが、「短いリスト」から始め、必要に応じて項目を増やすほうが、長いリストから始めてどれがいらないかを決めるよりも簡単です。

経験を積めば、しだいに目的に合った独自の「長いリスト」になります。その時点になると、私のリストとほかのリストを見て、自分独自のリストを適切に拡張できる方法がわかります。ただし、(彼らがカンニングしない限りは) 2 人のリストがまったく同じになることはありません。

私は最初に登山を始めた何年か前に、これと同じリストを覚えていて、準備している旅行の種類や日数にかかわらず、今でもそのリストを参照しています。誰か経験を積んだ登山家に「10 個の必需品がありますか?」とたずねれば、彼らの大部分が何について聞かれているか正確にわかるでしょう。実際には、さまざまなグループまたは個人が異なる「10 個の必需品」のリストを使用していますが、本質的にそれらはすべて同じです。また、それぞれの旅行で実際に荷造りするものはまったく異なる場合がありますが、それでも「10 個の必需品」が揃っていると言えます。

リストに入れるべきものは何でしょう。「状況に応じて…」

それでは、これが RUP とどのように関係するのでしょうか。私はプロジェクト・チームが RUP の多くの要素を分類するのを手伝うことがありますが、よく次のような質問を耳にします。「これらの項目をすべて分類して、どれがプロジェクトに必要かを決定するにはどうすればいいですか。」、「これは必要ですか。」、「RUP は大規模プロジェクト専用ではないのですか。」などの質問です。
最も多い回答は、「状況に応じて…」です。

---

[2] この「10 の必需品」のリストの詳細な分析については、次の文献を参照してください。"Mountaineering – The Freedom of the Hills", 6th Edition, The Mountaineers, Seattle, WA, 1997 35-41.

本当に必要なものは、友人のランディーにしたことと同じように、プロセスの説明を求めている人々に示すことができる「RUP の 10 の重要な要素」のリストです。このリストは、すべてのプロジェクトで、正しい要素を決定するための適切な出発点として利用することができます。これを、短期のハイキング (非常に小規模のプロジェクト)、一泊のバック・パッキング、スキー旅行またはカヤックの旅行 (異なる分野の中規模のプロジェクト)、大掛かりなエベレスト・クラスの遠征 (非常に大規模で重要なプロジェクト) に応用することができます。この目的は、RUP で「不可欠なもの」と呼んでいるもの、つまり本当に有効なソフトウェア・プロセスに重点を置くことにあります。またこれは、ソフトウェア開発の「最善の実践原則」[3]の概念にも結びついてきます。

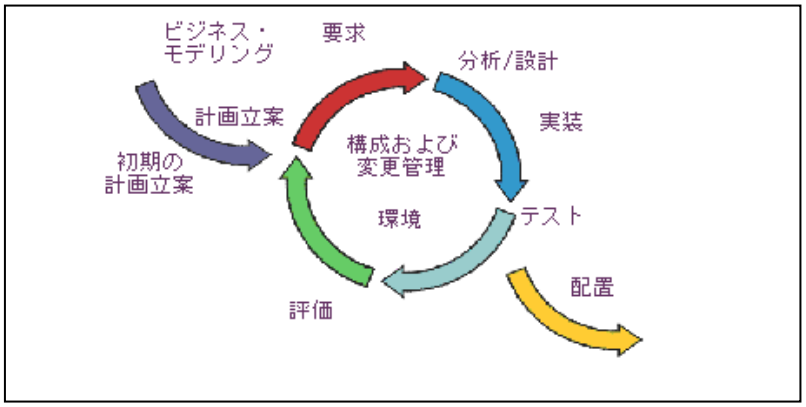
最初にフレームワークを作成

多くのプロジェクトで見かける共通の問題としては、質の高い製品を生産するためのプロセスのライフ・サイクル全体に含まれる「重要」な要素が分からないままに、ある特定の分野ばかりに力を注ぎ、その些細な事柄にこだわりすぎるということが挙げられます。

そこで、プロジェクトが遅れると、彼らは「やはり、要求管理のせいで遅れてしまった。」と答えます。この「要求管理」は、「ユースケース」、「プロジェクト・メトリックスの収集」、「構成管理の使用」、「欠陥追跡ツールの使用」、「ステータス会議の開催」、またはよく知られている多くの言葉のいずれかに置き換えることができます。

何か 1 つ特定の問題分野に的を絞ってしまう前に、より系統的で全体論的な手法を採用して、プロセスの重要な要素が揃っていること (例えばアーキテクチャー) を確認する方がさらに効果的です。

質の高いソフトウェア・プロセスのためのこのフレームワーク (またはアーキテクチャー) が作成済みになると、プロジェクトでは、一番問題になると識別された特定分野に効率よく力を入れることができます (通常は、要求管理がリストの一番上に挙げられることがよくあります[4]) 。ただし、RUP の手法は、プロジェクトのリスクを割り出して優先順位付けし、これらの識別されたリスクに対する早期の軽減戦略を決定する選択内容に基づくことを覚えておいてください。



[3] 「最善の実践原則」は、この数年にわたって Rational が推奨してきている次の項目です。(1) 反復的な開発、(2) 要求管理、(3) アーキテクチャーとコンポーネントの利用、(4) ビジュアルにモデリング、(5) 品質の検証、(6) 変更管理

[4]要求管理は能力成熟度モデル (CMM) で初期の「重要なプロセス領域」の 1 つで、これはソフトウェア開発プロセスの改善を決定した関係者にとってよくあるシナリオです。

RUP の 10 の重要な要素

それでは、「RUP の 10 の重要な要素」とは何でしょうか。次の一覧は、プロジェクトが Rational Unified Process の「重要な要素」に実際に従っている場合に、そのプロジェクトにそろえる最低限の項目のセットを示します。

1. 開発構想書
2. 計画
3. リスク
4. 問題点
5. 開発企画書
6. アーキテクチャー
7. 製品
8. 評価
9. 変更依頼
10. ユーザー・サポート

これからこれらの項目を個別に見て、RUP でその項目が当てはまる分野を調べ、それぞれが「短いリスト」の重要な項目になっている理由を理解します。

開発構想書 - 開発構想書の開発

開発構想書を明確にすることは、利害関係者の「*真の* ニーズ」を満たす製品を開発する上で重要です[5] 。

開発構想書は、RUP *での要求の作業分野* の「重要な要素」、つまり、問題の分析、利害関係者のニーズの理解、システムの定義、変化に応じた要求の管理などを取り入れたものです。

開発構想書により、より詳細な技術的要求に対する高レベルで時として契約的な基礎が提供されます。開発構想書によって、高レベルの要求と設計上の制約が把握され、これを利用することで開発するシステムが理解できるようになります。また、プロジェクトの承認プロセスに情報を提供するため、開発企画書と密接に関連します。開発構想書は、プロジェクトに関連する基本的な「なぜ」と「何」と密接に関係し、今後下される決定の中でどれが有効なものなのかを判断する尺度となります。

開発構想書は、次の質問に答えるものでなければなりません。これらの質問は、必要に応じて個別の詳細な成果物に分けられる場合があります。

- 重要な用語は何であるか。(用語集)
- 解決しようとしている問題は何であるか。(問題説明)
- 誰が利害関係者であるか。誰がユーザーであるか。ユーザーのニーズは何であるか。
- 製品機能は何であるか。
- 機能要求は何であるか。(ユースケース)

---

[5] 「目標は、利害関係者の「真のニーズ」を満たす高品質の製品を、期限内に予算内で開発することである」— *溺 anaging Software Requirements* , Dean Leffingwell& Don Widrig, Addison-Wesley Longman, January, 2000. (邦訳:「ソフトウェア要求管理 新世代の統一アプローチ」日本ラショナルソフトウェア訳、石塚 圭樹、荒川 三枝子 監訳)

- 機能外要求は何であるか。
- 設計上の制約は何であるか。

計画 - 計画に対する管理

「製品の品質は、その製品の計画に完全に依存するものである」[6]

RUP では、ソフトウェア開発計画書 (SDP) には、プロジェクトの管理に必要なすべての情報が集められます。ソフトウェア開発計画書は、方向づけフェーズで開発された複数の個別の成果物を含み、プロジェクト全体を通じて維持されます。

SDP は、プロジェクトのスケジュールを計画し、必要な要員を確保し、スケジュールに対する進捗を追跡するために使用します。SDP が扱う領域は、プロジェクト組織、スケジュール (プロジェクト計画、反復計画、要員、ツール)、要求管理計画書、構成管理計画書、問題分析計画書、品質保証計画書、テスト計画書、テスト・ケース、評価計画書、製品検収計画書などです。

簡単なプロジェクトでは、これらはそれぞれ 1 つまたは 2 つの文にまとめられる可能性があります。例えば、構成管理計画書では次のように簡単に述べています :「毎日の終わりに、プロジェクト・ディレクトリー構造の内容は、圧縮され、日付付きでラベルの付いた圧縮ディスクにコピーされ、バージョン番号を付与されて、中央のファイル・キャビネットに配置されます」。

ソフトウェア開発計画書のフォーマット自体は、計画書を作成するアクティビティーや思考ほど重要ではありません。したがって、フォーマットはどのようなものでも構いませんし、どのようなツールを使用しても構いません。Dwight D. Eisenhower 氏は、「計画書そのものには意味はない。計画を立てることこそが重要である。」と述べています。

重要な項目の 2、3、4、5、8 番は、RUP における*プロジェクト管理作業分野* の「重要な要素」、すなわち、新規プロジェクトの構想、範囲とリスクの評価、プロジェクトのモニターと管理、各反復とフェーズの計画と評価について記述しています。

リスク - リスクの割り出しと軽減

RUP の重要な役割は、プロジェクトの初期に、最も高いリスク項目を割り出して除去することです。リスク・リストは、リスクを把握してプロジェクトを成功に導くことを目的とします。ここでは、深刻な結果を生む可能性のあるイベントが、重要度の高い順に整理されます。

各リスクと共に、そのリスクを軽減する計画が含まれる必要があります。リスク・リストは、プロジェクトのアクティビティーの中心となり、組織化された反復を決定する基準となります。

問題点 - 問題の割り当てと追跡

進行中のアクティビティーから直接発生した客観的なデータを継続的に参照すること、製品構成を展開することは、どのようなプロジェクトにおいても重要です。RUP では、管理と技術上の問題やプロジェクトのリスクを特定し、伝達し、解決するメカニズムを提供する、通常のステータス評価書でこの作業を行います。問題の把握に加え、各問題には、解決に責任を持つ責任者と共に期限日を割り当てる必要があります。問題は定期的に追跡し、必要により更新する必要があります。

これらのプロジェクトのスナップショットにより、管理における注意力が強化されます。期間はさまざまなため、強制機能ではプロジェクトの履歴が把握され、解決されて、進行を妨げる障害やボトルネックが除去される必要があります。

---

[6] Johnson Space Center Shuttle Software Group、"They Write the Right Stuff", Charles Fishman, *Fastcompany,* Issue 6, p. 95, December, 1996.

開発企画書 - 開発企画書の検証

開発企画書はビジネスの観点から、このプロジェクトに投資する価値があるかどうかを決定するのに必要な情報を提供します。

開発企画書の主な目的は、プロジェクトの開発構想書を実現するための、経済的な計画を作成することです。作成された開発企画書は、プロジェクトの投資効果 (ROI) を正しく評価するために使用します。開発企画書によって、プロジェクトの正当性を明確にし、経済的な制約を確立します。また、経済面での意思決定者に、プロジェクトの経済効果について情報を提供し、プロジェクトを推進するかどうかの判断材料とします。

この説明では問題の詳細に深く立ち入るべきではなく、どうしてこの製品が必要かという説得力のある議論を展開すべきです。ただし、これは短くし、プロジェクト・チームの全メンバーが理解して覚えられるよう、十分簡単にする必要があります。重要なマイルストーンに達するたびに、開発企画書を調べてコストと収益の見積もりが正しいかチェックし、プロジェクトを継続するかどうか再検討します。

アーキテクチャー – アーキテクチャーとコンポーネントの設計

Rational Unified Process においては、ソフトウェア・システムの (ある時点での) アーキテクチャーは、システムの主要なコンポーネントの組織または構造です。 その主要なコンポーネントはインターフェースを通じてより小規模なコンポーネントやインターフェースから構成されるコンポーネントと相互に作用し合います。主要な部分は何でしょうか。これらをどのように組み合わせるのでしょうか。

RUP は、ソフトウェア・アーキテクチャーを設計、開発、検証するための組織的、系統的な方法を備えています。これは RUP における*分析/設計の作業分野* の「重要な要素」であり、アーキテクチャーの候補の定義、アーキテクチャーの洗練、振る舞いの分析、システムのコンポーネントの設計です。

ソフトウェア・アーキテクチャーについて説明するためには、アーキテクチャーの重要な側面を表現する手段であるアーキテクチャーの表現方法を最初に定義する必要があります。RUP では、この定義は、複数のビューでアーキテクチャーを表すソフトウェア・アーキテクチャー説明書に取り入れられます。

各アーキテクチャー・ビューでは、開発過程で利害関係者が関心を抱く、特定の一まとまりの事項を取り上げます。そのような利害関係者とは、エンド・ユーザー、設計者、管理者、システム・エンジニア、保守担当者などです。ソフトウェア・アーキテクチャー説明書は、プロジェクトで行われるアーキテクチャー上重要な決断に関して、アーキテクトとその他のチーム・メンバー間のコミュニケーションの手段として利用されます。

製品 – 製品の段階的なビルドとテスト

RUP での*実装とテストの作業分野* の「重要な要素」は、方向づけした後の各反復の終わりに実行可能なリリースを使用して、システムのコンポーネントをインクリメンタルにコーディング、ビルド、テストすることです。

推敲フェーズの終わりでは、評価のためにアーキテクチャー・プロトタイプを利用することができ、このフェーズには必要に応じてユーザー・インターフェース・プロトタイプを入れることもできます。作成フェーズの各反復全体にわたって、コンポーネントを実行可能なテスト済みのビルドに統合し、最終的な製品に発展していきます。

この重要な要素で重要なことは、進行中の構成管理とレビュー・アクティビティーだけでなく、製品のビルドに伴う統合された一連のテストを行うことです。

評価 - 結果の定期的な評価

反復評価書では、反復の結果、評価基準がどれくらい満たされているか、教訓と実装される変更を把握します。

反復評価書は反復アプローチには不可欠の成果物です。プロジェクトの開発範囲とリスク、反復の本質によって、反復評価書の内容は、シンプルなデモンストレーションと結果の記録から、すべての正式なテストのレビュー記録まで多岐に渡ります。

ここで重要な点は、製品の問題だけでなく処理上の問題に着目することです。「後手に回るほど、追いつくには時間がかかります」。

変更依頼 - 変更の管理と制御

構成と変更管理の作業分野 の「重要な要素」は、すべての利害関係者のニーズを考慮して対応するという目標を維持しながら、プロジェクトのライフ・サイクルを通じてどのような変更が発生した場合でも、それに応じてプロジェクトの開発範囲を管理することです。

ユーザーに最初のプロトタイプが提供されるとすぐに (また、その前であっても)、変更が要求されます (最も確実なものの 1 つです) 。これらの変更を管理し、プロジェクトの開発範囲と利害関係者の期待を効果的に管理するために、開発成果物へのすべての変更は変更依頼を通じて提案され、一貫したプロセスで管理される必要があります。

変更依頼は、障害、拡張依頼など、製品に対するあらゆるタイプの変更の依頼を記録し、追跡するために使用します。変更依頼の利点は、決定事項の記録が残ることです。 さらに、その評価プロセスによって、プロジェクト・チーム全体に、確実に変更による影響を周知できます。変更依頼は、プロジェクトの開発範囲の管理と提案された変更の影響の評価にとって重要です。

ユーザー・サポート – ユーザーへのサポートの提供

最低限、オンライン・ヘルプを介して実装されるユーザー・ガイドが必要で、インストール・ガイドとリリース・ノートが必要になる場合もあります。製品の複雑さに応じて、トレーニング教材、製品の梱包材料と共に部品構成表も必要です。

RUP では、*配置作業分野* の「重要な要素」は、エンド・ユーザーによる製品の学習、使用、運用、保守をサポートするために必要なすべての資料と共に製品を梱包して、納品することです。

要求について

この重要な要素のリストを見てその選択内容にまったく同意できない可能性もあります。例えば、次のような疑問が考えられます。「要求はこのリストの中でどれに当てはまるのか?」「開発構想書を使わずに、何か代わりのものを選んだほうがよいのではないか?」「要求は間違いなく重要なものなのか?」これらはもっともな疑問です。これに対する回答は、基本的に最も重要な要素は何かということであり、リストに「要求」を入れることが望ましいなら、その場合はリストに入れるとよいということです。(それぞれのプロジェクト・チームで独自のリストを決定する必要があります。ここで説明した 10 の重要な要素は、さらに検討を進めるための出発点としてのみ使用してください) 。

ただし、過去の経験では、プロジェクト・チームに (特に内部プロジェクトに) 「要求は何ですか」とたずねると、「要求は何もありません」という返事が返ってくることがありました。最初はこの返事に驚きました (私は軍事、宇宙航空分野の出身ですから) 。要求が何もないということがあるのでしょうか。そこで、チームのメンバーが「要求」という言葉を聞いて頭に浮かぶのは、外部から課せられ従うべきことであり、そうしなければプロジェクトが却下されてしまうという一連の「すべきだ」という事柄だということです。実際にはこうしたものは何もありません。彼らは、プロジェクト全体にわたって製品の要求が発生する、調査と開発に携わる場合があります。

今度は、「要求は何もありません」と返事された場合に、「わかりました。では製品の開発構想は何ですか」と質問してみます。すると彼らの目が輝きます。 前に述べた開発構想書の 7 つの質問それぞれについて問題がないことを容易に確認できると、その後に要求が出てきます。これは、契約上のプロジェクトよりも共同作業の環境にはよくあることで、この場合は与えられた指示よりも多くの要求が見つかります。

テストについて

RUP の「重要な要素」の 1 つに「テスト」を入れていないことにも気付かれたかもしれません。なぜ入っていないのでしょうか。効果的なソフトウェア開発プロセスに対してはオプションと考えているビジネス・モデリングとは異なり、テストは決してオプションではありません。テストは、進行中の構成管理やレビュー・アクティビティーとほぼ同じように、製品の設計とビ

ルドに伴う統合された一連のアクティビティーであると考えています。これは偶然にも、IEEE 1074 ソフトウェア・プロセス標準とまったく同じアクティビティーです。

また、実際にはテストが、製品のビルド (重要な要素の 7 番目) に含まれており、テストの結果が、検証と評価 (重要な要素の 8 番目) にとって非常に重要であるということに気付かれたかもしれません。ソフトウェア開発に対する Rational Unified Process の反復手法での重要な要素は、問題の特定とリスク、問題の解決を早期に行うために、頻繁に、実行可能な製品のバージョンのビルド、テスト、評価を行うことにあります。

まとめ :10 の重要な要素の適用

それでは、「RUP の 10 の重要な要素」を見つけると実際の場面でどのように生かすことができるのでしょうか。

非常に小規模なプロジェクトに対して

まず最初に、プロセスについて学習したばかりの友人のランディーが彼自身で、または非常に小規模なチームと共に、RUP を使用して簡単な製品をビルドする方法を聞いてきた場合、RUP と Rational Suite ツール群の膨大なすべての機能で彼らを圧倒する代わりに、独自の「10 の重要な要素」のリストを提供します。

実際、これら 10 の重要な要素は、特殊なツールを利用しなくても実現することができます。10 の重要な要素の 1 つごとに的を絞ったプロジェクト・ノートは、プロジェクトを管理するための最適な出発点となります。また、RUP の個人的なアプリケーションに対する初期の変更依頼の管理、優先付け、追跡を行う場合に、ポストイットが役に立つことがわかりました。

規模が拡大していくプロジェクトに対して

もちろん、10 の重要な要素を適用するこれらの簡単な方法では、プロジェクトの規模と複雑さが増すにつれすぐに管理できなくなってきます。 このため、Rational ツール群を購入する見込み顧客を探している場合は、「RUP の 10 の重要な要素」が当てはまる顧客に大きな見込みがあることがわかります。

もちろん、その顧客がそれぞれの分野で単に表面的ではなく詳細な部分まで実施することを決定すれば、Rational ツールのサポートに加えて、最終的に完全な RUP も必要になります (交渉がうまくいった場合) 。ただし、RUP ツールを使用してこれらの人達を指導するのではなく、「最善の実践原則」と「RUP の 10 の重要な要素」を使用して指導することをお勧めします。

熟練したプロジェクト・チームに対して

より完成度の高いプロジェクト・チームでは、「10 の重要な要素」を使用するとそのプロセスの重要な要素のバランスを素早く評価し、改善により最も効果の出る分野を特定することができます。

すべてのプロジェクトにおいて、これらの重要な要素のいずれかを無視した場合に何が発生し、重要な要素を含めなかった場合に何ができないかを検討することは重要です。例えば、次のようなものです。

- 開発構想書がない場合、目標を見失い、すぐに回り道して迷ってしまう可能性があります。
- 計画書がない場合、進行を追跡できません。
- リスク・リストがない場合、見当違いの問題に焦点を当てる可能性があり、思いもしなかった大きな問題が 5 カ月後に発生するようなことになります。
- 問題点リストがない場合、問題点に対する責任がないと、成功を妨げる障害になります。
- 開発企画書がない場合、プロジェクトでの時間と資金を失う危険性があり、プロジェクトはキャンセルされるか、破産する可能性があります。
- アーキテクチャーがない場合、通信、同期、データ・アクセスにおいて、ある程度の規模とパフォーマンスを伴う問題が発生したときに、処理できない可能性があります。

- 製品 (プロトタイプ) がない場合、コードを作成することから開始しなくてはならず、単に事務処理が増えていくだけなので進捗速度は遅くなり、顧客を前にしても作業を続けることになります。
- 評価書がない場合、問題から目をそらさないでください。真実と向き合うことが重要です。実際に締め切りはどれだけ迫っていますか。品質における目標に達していますか。あるいは予算はどうですか。
- 変更依頼がない場合、変更の依頼をどのようにして追跡しますか。
- ユーザー・サポートがない場合、ユーザーが質問したい場合や、製品の使用方法を理解できなかったりした場合、どうなりますか。サポートはどれくらい容易に受けられるでしょうか。

これですべてです。

なぜ「10」の重要な要素なのでしょうか。10 以上にはしたくない、という以外に特に理由はありません (そんなにたくさんは数えられません)。「9 の重要な要素」またはそれ以下にできれば幸いですが、それらのいずれかを除くことはできそうにありません。これが 10 が重要な要素として適切である理由となりませんか。これが正しい結論と言えそうです。これらをプロジェクト・グループの出発点として使用することをお勧めします。あなたの「10 の重要な要素」は何でしょうか。

記憶を引き出すちょっとした仕掛け、または頭字語を決めると「10 の重要な要素」を覚えるために役立ちそうです。例えば、4 音節にまとめて、V-PRI-BAPE-CU とします。もちろん重要な要素の数は 10 にしたので、PalmPilot に書いてあるリストを見なくても手の指でこれらを勘定することができます。(V-PRI-BAPE-CU は、Vision、Plan、Risks、Issues、Business Case、Architecture、Product、Evaluation、Change Request、User Support のそれぞれの頭文字です。(これが秘訣となります。)

Rational®
the software development company

Dual Headquarters:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Toll-free: (800) 728-1212
E-mail:info@rational.com
Web:www.rational.com
International Locations:www.rational.com/worldwide