



価格設定について

GUPTA™



価格設定について

Team Developerは、他の開発ツールと比べて製品価格が高く設定されています。

その分、製品購入時の投資費用は、開発者の数が多くなればなるほど、かなりの金額になります。しかしながら、Team Developerの最大の特徴は、アプリケーション開発において、他の開発ツールを圧倒する高い生産性を、すべての開発者に提供できる事です。ですから、製品購入時の投資費用は、驚くほど簡単に、しかも短期間で回収する事ができます。さらにその後は、加速度的に生産性が向上します。

高い生産性について

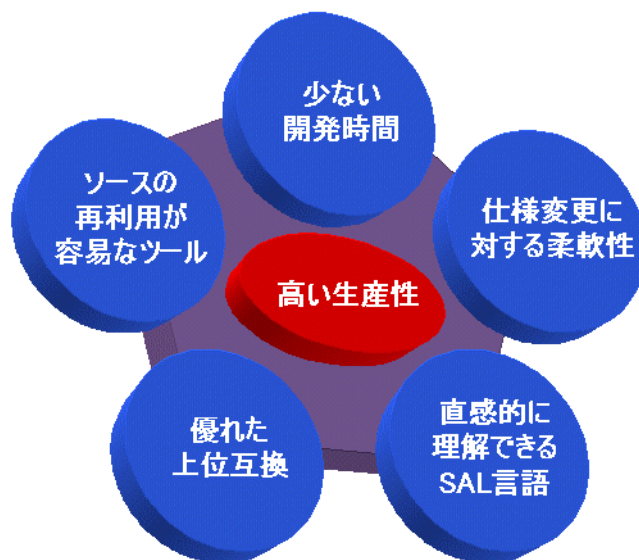
では、何を基準に生産性の高さ低さを判断しているのでしょうか？一連のアプリケーション開発工程と開発したアプリケーションのライフサイクルから次の5つの基準を定義しました。

1. アプリケーション開発に必要な時間

Team Developerは、JavaやVisual Basicなどの他の開発言語と比べると、平均して約40%の開発工数の削減が可能と言われています。

皆様は、JavaやVisual Basicで、データベースに接続するプログラムを何ステップで書きますか？帳票に出力するプログラムは何ステップで書けますか？ファイルを操作する場合はどうですか？アプリケーションにWebサービスを実装する場合はどうですか？Team Developerならわずか数行のプログラムを書くだけです。これは、Team Developerの持っている独自の開発言語のSALが、非常に良く体系立てて設計されているからです。

さらに、SALは直感的で覚えやすく、話し言葉に近いので、Team Developerを習得する為に必要な時間、機能拡張やメンテナンスの為に他のエンジニアが書いたプログラムを理解する為の時間など、今まで煩わしく感じられた時間を大幅に削減する事ができます。



2. 仕様変更が発生した時の対応

アプリケーション開発の工程で、必ずと言っていいほど仕様変更は発生します。仕様変更が発生した場合、変更点をどのように吸収するかが大きな問題です。一般的に、仕様変更が発生した場合、そのインパクトを最小限に抑える為の様々な工夫をしてアプリケーションを設計します。

このような設計をする為には多くの知識と経験が必要になります。

これに対してTeam Developerでは、SAL言語の特性を上手く利用する事によって、仕様変更が発生しても手戻りが少ないプログラミングが可能となり、柔軟に対応することが出来ます。





3. メンテナンス性

前述したように開発言語のSALは、直感的で覚えやすく話し言葉に近いので可読性に優れています。これは、デバッグ作業や機能の拡張作業など、一度書いたプログラムコードを、後日読み返す(しかも注意深く読み返す)時に大きなアドバンテージになります。また、プログラムの技術レベルに関係なく、プログラムコードの品質を一定以上に保つ事が容易に可能となります。

4. 新旧のバージョン間の互換性

Team Developerは、ほぼ100%上位互換です。つまり、旧バージョンのプログラムコードは、新バージョンでもそのまま使う事ができます。OSやPCの入れ替えに伴うアプリケーションのアップグレード作業を、驚くほど簡単、かつ迅速に完了する事が可能です。前例では、1997年にTeam Developer1.0で開発したプログラムコードを、2005年にTeam Developer2005にアップグレードしたところ、コンパイルエラーはありませんでした。約8年前(リリースアップで7世代前)の製品でも互換が保たれています。今までにTeam Developerで開発したソフトウェア資産を無駄にしません。

5. プログラムコードの再利用

アプリケーション開発のフレームワークから、方法論や手法論、精神論にいたるまで、様々な考え方がありますが、それらに共通して言える事は、どれもアプリケーション開発の生産性を高めるための方法や考え方が含まれています。生産性を高めるもっとも簡単な方法は、前回使ったプログラムコードを再利用する事です。Team Developerでは、画面やプッシュボタン、入力フィールドなどの目で見ることのできるコンポーネント、DLLやユーザー定義の関数など目で見ることのできないコンポーネントを、それぞれ部品として再利用する事が容易にできますので、2回目以降のアプリケーション開発は加速度的に速くなります。

もっとも重要な事は何でしょうか？

アプリケーション開発の現場でもっとも重要な事は何でしょうか？

それはコミュニケーションです。そのプロジェクトが成功したか失敗したかを判断する基準の1つは、スケジュール通りにプロジェクトが進行しているか否かです。失敗するプロジェクトは、例外なくスタッフのコミュニケーションが悪いです(最悪です)。そこにはチームワークとか助け合うという概念は既になく、逆に相手にスキがあれば足を引っ張って意地悪をしてやろうという雰囲気さえ感じられる場合があります。もうこうなると收拾がつきません。意地悪には意地悪で返しをしたりして、プロジェクトは悪い方へ悪い方へ進んで行きます。これを『マイナスの相乗効果』と言います。

(次ページへつづく)



GUPTA

Cross-Platform Development & Deployment



マイナスの相乗効果に陥ったプロジェクトチームは、毎日辛い事の繰り返しです。

人間関係はギクシャクして一触即発のピリピリモードがプロジェクトルームを支配しています。

スケジュールは遅れていますから、当然のように毎日残業で、運が悪ければ終電に乗り遅れる事もしばしばあります。朝もゆっくり寝ていられません。朝食もそこそこで済ませて入社しなければいけません。奥様やお子様たちはまだ寝てる時間です。通勤ラッシュの満員電車に乗ってへろへろになりながらオフィスに着いて、休む間もなくすぐに仕事を始めます。昼食もコンビニの弁当を食べながら仕事を続けています。スケジュールが遅れているから休みたくても休めません。

こんなプロジェクトを見た事があるという方もいらっしゃるのではないのでしょうか？

誰もが望んで参加したくはないプロジェクトです。

一方、成功するプロジェクトは、例外なくスタッフのコミュニケーションが良いです。

チームワーク、助け合いのお手本のような光景が繰り返されています。スタッフの1人がある問題に直面すると、頼まれてもいないのに「手伝おうか？」と別のスタッフが声をかけて、一緒に問題解決に取り組みます。ほとんどの問題が難なく解決するのですが、手伝ってもらったスタッフは、「ありがとう」とお礼を言います。手伝ったスタッフは、お礼を言われて嫌な気持ちはしないので、また誰かが困っていたら手伝ってあげようという気持ちになります。手伝ってもらったスタッフは、感謝の気持ちから、今度は自分が誰かを手伝ってあげようという気持ちになります。これを「プラスの相乗効果」と言います。もちろん、各スタッフは、自分の担当する仕事があるわけですが、スケジュールが遅れるスタッフはいません。

このプロジェクトのスタッフは、スケジュールは予定通りなので焦って出社する必要はありません。

お子様たちが学校へ行くのを見送ってから、コーヒーを飲みながら新聞を読んでから自宅を出ます。

この時間になると通勤ラッシュも一段落し、快適な通勤ができます。オフィスに着いたら、メールをチェックしてお気に入りのウェブサイトをチェックして、定例のブリーフィングが終わる頃には、そろそろランチタイムになります。ランチタイムもしっかり1時間あります。プロジェクトのスタッフと一緒にランチに行く事で、より一層コミュニケーションが良くなります。夕方まで仕事をして定時には退社する事が可能です。プロジェクトのスケジュールは予定通りなので、残業の必要がありません。この時間でも夏はまだ明るいので、帰宅してからお子様とキャッチボールやサッカーをして遊ぶ事もできます。

こんなプロジェクトが存在するはずはないと思われる方もいらっしゃるのではないのでしょうか？

しかし、この2つの例は、実在したプロジェクトです。誰もが遅らせようと思ってスケジュールを遅らせるのではないし、仕様変更が発生して、スケジュールが遅れてしまうのは仕方がない事ではありますが、この2つのプロジェクトで決定的に異なる点が1つだけあります。それは、コミュニケーションをとる為の時間が有るか無いかです。コミュニケーションの基本は会話です。スケジュールに追い掛け回されて忙しくなると、会話をする時間もなくなります。

(次ページへつづく)



GUPTA

Cross-Platform Development & Deployment



では、時間を作るにはどうしたら良いでしょうか？それは、仕事を速く終わらせる事です。
では、仕事を速く終わらせるにはどうしたら良いでしょうか？
それは、生産性の良い開発ツールを使う事です。

抜群に生産性の良い開発ツールのTeam Developerを是非お使いください。
そして、あなたが理想的だと考える『プラスの相乗効果』をイメージして下さい。
イメージできたらあとは実行あるのみ。
あなたのプロジェクトにもTeam Developerが『プラスの相乗効果』をもたらします。

GUPTA事業部
大野元嗣



GUPTA

Cross-Platform Development & Deployment