

ソフトウェア・デリバリーのコミュニティ、モジュラー性、エンパワーメント
ホワイト・ペーパー
2007 年 2 月

Rational. software



ソフトウェア・デリバリーの将来像

Danny Sabbah, PhD

Rational ソフトウェア担当ゼネラル・マネージャー

目次

目次

2 概要

4 従来のソフトウェア開発

8 今日の重要な動向

11 コミュニティー・ベースのソフトウェア開発

14 サービス指向アーキテクチャー

17 ガバナンスとエンパワーメント

19 将来のソフトウェア・デリバリー

20 知的財産のサプライ・チェーン

22 コミュニティーの設立と参加

23 Rational の方向性

25 結論

概要

「いつの世も人は古い流儀をあざ笑い、新しい流儀を熱心に追いかける。」

—ヘンリー・デイヴィッド・ソロー

今日のソフトウェア開発およびデプロイメントの専門家が無視することのできない近年の重要な動向があります。それは、サービス指向アーキテクチャー (SOA)、コミュニティ・ベース・ソフトウェア (またはオープン・ソース・ソフトウェア [OSS])、そしてグローバル分散開発 (GDD) です。¹

あるアナリストは、2006 年末までにはグローバル 2000 企業の 62% が SOA を実装済みとなるだろうと予測しています。別のアナリストの予測はこれよりさらに高く、「10 社のうち 9 社がサービス指向アーキテクチャーの導入を進めているか、または導入済みであり、2006 年末までには SOA の計画、設計、プログラミングを経験しているだろう」³ としています。OSS は、今後 10 年間に国際的大企業の 4 分の 3 以上でミッション・クリティカルなソフトウェア・ポートフォリオに採用されると予測されています。また、今日すでに大半の企業が実稼働環境に OSS を使用していることを、多くの調査結果が示しています。⁴

SOA、OSS、GDD はこれまでにない現象だと思われるかもしれませんが。しかし、これらは決して何もないところから生まれたものではないということを忘れてはなりません。新しいアーキテクチャー、新しいソフトウェア・モデル、新しいソフトウェア開発および管理手法に着手する前に、SOA、OSS、GDD とは何か、それらは何を意味し、何に由来するのかを慎重に検討することが賢明です。そうすることで、やがて行うことになる大規模な導入に向けて、より周到な準備をしておくことが可能になります。

ハイライト

プロセスこそが、**重要な推進要素**です。

SOA は決して目新しいものではありません。Enterprise JavaBeans Java™ テクノロジーが開発された背景には、同種サービスの裏付けがあり、さらにその前には CORBA もありました。優れた SOA には、オープン、標準ベース、コミュニティ駆動型、統制可能、モジュラー形式、入手が容易などの特長があります。しかしこのような概念の一部は、以前から存在していました。OSS に関して同様です。実際、ソフトウェア開発という点では、OSS の基礎となるような革新は特にないのです。

オープン・ソース・ソフトウェアは 25 年にわたって利用されてきました。その多くは学術分野で開発されたものですが、わずかこの 2 年から 5 年の間で主要勢力へと進化しました。その主な理由となったのは、最も重要な動向、つまりオープン・ソース・ソフトウェアがコミュニティ・ベースのソフトウェアであるということです。OSS の最も顕著な特性、すなわち広汎性、イノベーション、低コストが、ここでも発揮されています。ソフトウェア・デリバリーの真の価値は、次の 3 段階を経て付加されていきます。

1. 優れたソフトウェア・モジュールが、OSS 開発へのコミュニティの参加を促します。
2. オープンな開発環境で、イノベーションとコンポーネントの共有化が起こります。
3. 標準化により、業界での導入が始まります。

つまり、成果物が OSS であっても、それを生み出す重要な推進要素となっているのはプロセスです。これが、コミュニティ・ベースのソフトウェアなのです。

従って、本当に重要なのは SOA、OSS、GDD などの 3 文字略語ではなく、それらの基礎となっている規範とプロセス、すなわち共通の興味を持つコミュニティ、モジュラー・システム、およびエンパワーメントです。流行の専門用語そのものではなく、それらの動向がもたらす「ilities」(qualities: 品質、abilities: 能力) に注目することで、そのような利点を今すぐ自分たちのソフトウェアに取り入れることが可能になります (まだ取り入れていなければ、の話です)。

ハイライト

IBM は、現場の人々のより広範なコミュニティの育成に努めています。

本書では、今日の急速に拡大する要件と手法を自社のソフトウェア・デリバリーに取り入れること、すなわち長い年月を経てその有効性が実証されているベスト・プラクティスと今日のイノベーションを引き出して、自社のニーズに的確に合わせることを中心に解説します。IBM Rational® グループはすでにこれを習得し、既存製品およびソリューションの開発と管理に反映させています。また、IBM ではこれらの原理を今後の Rational 製品リリースにも拡大して適用し、現場の人々のより広範なコミュニティを育成することに力を入れています。

本書ではこの後、次の 4 部に分けて解説を進めます。現在の環境に役立っている過去のソフトウェア開発の課題の概要、今日の IT 業界を推進する主要な動向の特定、Rational グループの将来のソフトウェア・デリバリー戦略の概観、および全体の総括です。

従来のソフトウェア開発

今日のソフトウェアおよびシステム開発では、新しいエンタープライズ・リソース・プランニング (ERP) システムをデプロイするよりも新しい製造工場を建築するほうが早いことすらある、というのが現状です。部品のサプライヤーを IT 上のサプライ・チェーンに統合するより、物理的なサプライ・チェーンに統合するほうが早いように思われます。

例えば、トヨタは 1986 年にケンタッキー州ジョージタウンで同社の新しい北米組立工場の建設に着工しました。その後 2 年足らず (正確には 23 カ月) で、第 1 号車はその組立ラインから出荷されました。SAP ERP システムの平均インストール所要期間は、約 3 年 (33.6 カ月) です。⁵ 電子的な成果物、アイデア、プロセスをデリバリーする仕組みを IT システム上に構築することは、組立工場をゼロから建設し、自動車製造用部品を世界中から調達する経路を決めて、第 1 号車を世に送り出すことよりも時間がかかるということです。

ハイライト

ソフトウェア産業は、まだ青年期
にあります。

ソフトウェア産業が出現してからすでに 50 年以上経ちますが、物理的な実装とソフトウェアの実装には、いまだに大きな隔りがあります。問題は、ソフトウェア産業とソフトウェア・プロセスがまだ完全には形成されていない、つまり未成熟だということです。ソフトウェア産業はまだ青年期にあり、急成長している最中です。そうした不安定さの最大の原因は、まさに私たちがソフトウェアを実行しているその基盤にあります。つまり、現在の IT インフラストラクチャーの基盤が、これまでになく急速に進化しているのです。また、CPU の高速化も、ソフトウェアの開発とデプロイの方法を非効率的なものにしています。

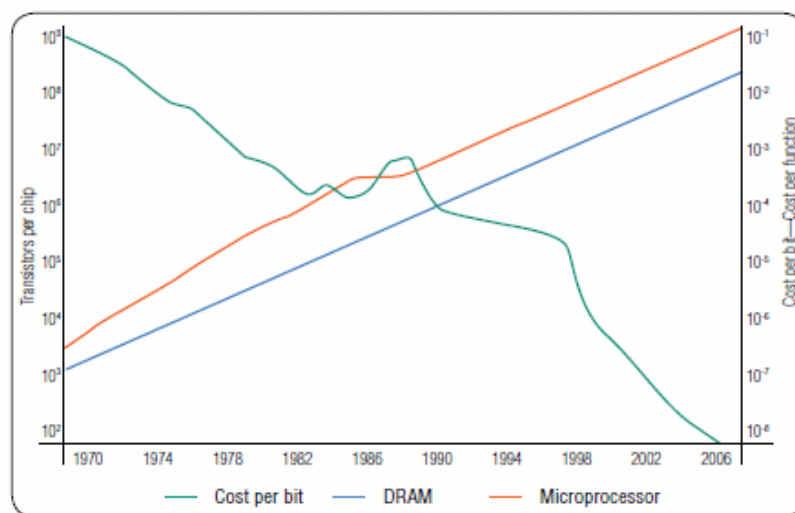


図 1. メモリー容量とプロセッサの処理能力の向上に従い、コストが低下

図 1 は、プロセッサの処理速度が「ムーアの法則」⁶ に従って向上し、同時にコンピューティング能力のコストは急速に低下することを示しています。ハードウェアとソフトウェアの急速な進化によって新しいテクノロジーの応用が可能になり、それにより、数年前には不可能だったソフトウェアも現在では構築やデプロイが可能になっています。

ハイライト

すべての開発者が、低価格のプロセッサやメモリーを利用できるようになったことで、それらの先進技術を生かした新しいソフトウェア・フレームワークが無数に登場することになりました。これらの新しいフレームワークに統合できないレガシー・アプリケーションは、はかりしれないほど不利となり、それ自体が変革の阻害要因となります。

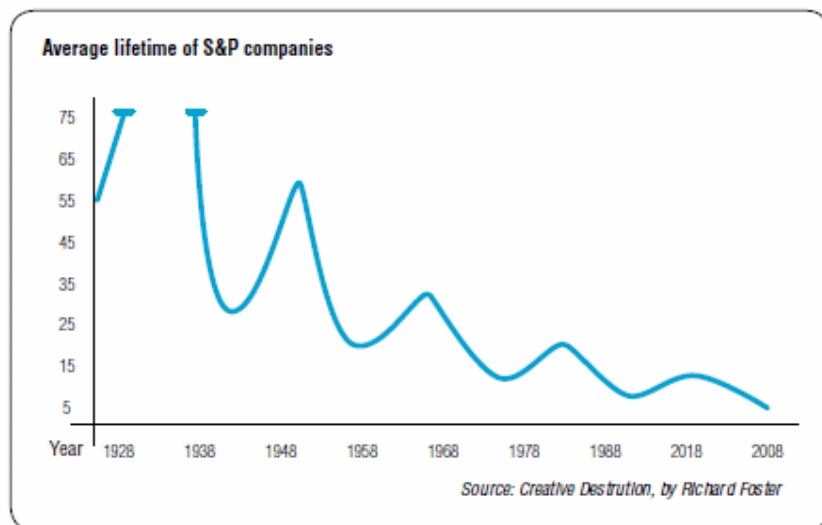


図 2 S & P 企業の平均存続期間は、1930 年以降、大幅に短縮

今日の市場は、わずか 5 年か 10 年前と比べても大きく変化しています。

コンピューティングおよび通信テクノロジーの加速に伴い、S&P（スタンダード・アンド・プアーズ）500 企業の平均存続期間も急速に短縮されています。1930 年には 75 年だった平均存続期間が、現在では 15 年となっています。グローバル市場の急速な進化と、市場における資本化の原理には、考察に値する貴重な教訓が含まれています。今日の市場は、わずか 5 年か 10 年前と比較しても大きく変化しており、市場のルールはすっかり変わってしまったと考えがちです。しかし、そうではありません。

ハイライト

ソフトウェア・エンジニアリングの手法、プロジェクト管理のパラダイム、プログラミング言語とスクリプティング言語、そしてエンタープライズ・フレームワークはいずれも、まだ確立されてません。

1990年代後半の「ニュー・エコノミー」と2000年代の株価収益率のことを思い出してみましょう。これらは企業評価における新たな実態の先駆けとなりました。ところが2001年を振り返ってみると、この年にこれは徹底的に反証されました。企業はこれまでにないスピードで設立されては解体されています。しかし少なくとも今のところ、旧来のルールがまだ市場の評価を支配しているように思われます。

この教訓は、今起きている技術革新の波にどう対応するかを理解する上でも役立ちます。これは非常に的を得た表現ですが、私たちは今、コンピューター・テクノロジーという竜巻の渦の中にいるため、その基礎となっている商業、建設、ソフトウェア開発、システム・インテグレーションなどの各分野で経験してきたことが見えなくなっているのです。少しの間その竜巻の外に出てみれば、後に残されたものがすぐに分かります。

ソフトウェア・アーキテクチャーのレベルは、まだ従来の建築アーキテクチャーの成熟度には達していません。また、ソフトウェアのサプライ・チェーン・テクノロジーの機能や堅牢性も、製造業の物理的なサプライ・チェーンと同等のレベルには達していません。ソフトウェア・エンジニアリングの手法、プロジェクト管理のパラダイム、プログラミング言語とスクリプティング言語、そしてエンタープライズ・フレームワークはいずれも確立されておらず不安定です。しかしながら、過去10年間の株式市場の教訓が物語っているように、真の価値評価につながるプロセスを支配する原理は、変わらない傾向にあるようです。

既存のプロセスや方法論の真の価値を認識すること、注力すべき分野を把握すること、プロセスが正しい方向に進んでいることを認識することは、今日入手可能な多数の選択肢の中からどれを選んでどれを捨てるべきかを知るために、きわめて貴重な洞察力となります。IBM Rational グループでは、戦略的ビジョンを提供する多数の動向を特定しました。それらは検証に値するものです。

ハイライト

低価格のブロードバンドによるデータ・アクセスとソーシャル・ネットワーキングが、大きな影響を及ぼしています。

今日の重要な動向

将来 21 世紀初頭を振り返ってみたとき、ソフトウェア・コミュニティーのみならず世界全体に影響を及ぼした最も重要な現象は、インターネットや携帯電話およびワイヤレスのテクノロジーがもたらした低価格のブロードバンドによるデータ・アクセスとソーシャル・ネットワーキングの到来であることは明らかです。例えば、新興諸国で携帯電話の利用が可能になったことは、多くの新興国の経済にすでに大きな影響をもたらしています。この進歩は、もっぱら話し言葉によってもたらされています。これらの社会が商業をインターネット上に拡大し、知的財産の開発を始め、時間、空間、資本資源という物理的境界から解放されたときにはどうなるのか、想像してみてください。

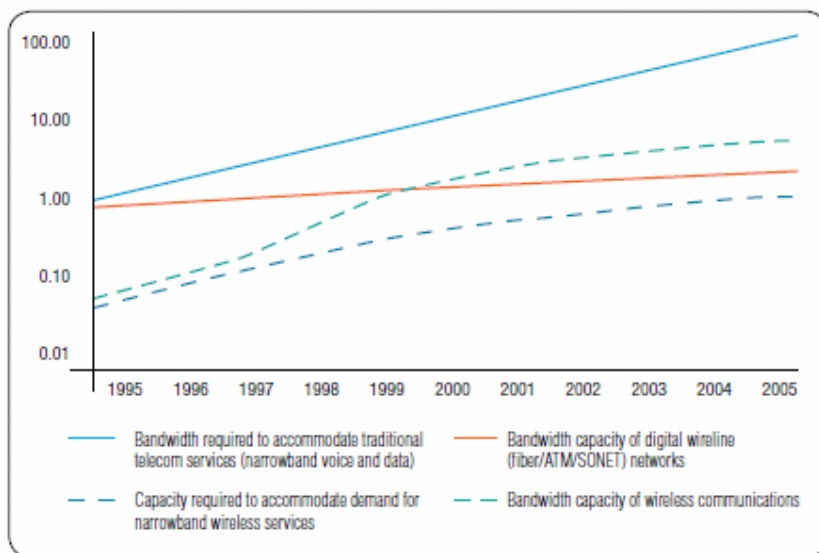


図 3. 帯域幅は拡大し続けています。

ハイライト

ブロードバンドの容量は年々増大し続けており、私たちが目にしているのは、この現象の始まりにすぎません。異種の複数メディアにわたって帯域幅が拡大した結果の 1 つとして、デリバリーメカニズムよりもデリバリーされたデータまたはデータの内容のほうが重要であると認識されるようになった点が挙げられます。テレコム業界では、データ、情報双方の流れを統合させる動きがすでに見られます。これはもはやテレビ、インターネット、電話間の対立という問題ではありません。データ転送の速度、信頼性、可用性の問題であり、最終的にデリバリーされるコンテンツの個々の所在場所、フォーマット、デバイスをエンド・ユーザーが決められるようになるということです。

広く浸透するデータ・アクセスが、私たちの生活を変化させつつあります。メトカフの法則⁷では、ネットワークの価値はそのシステムのユーザー数の二乗に比例するとされています。これは次のような式になります。

$$\frac{n(n-1)}{2}$$

つまり、ネットワークに接続する人が増えれば増えるほど、その接続によってより多くの価値が生み出されるということです。これについては、すでにソーシャル・ネットワーキングがその正しさを証明しています。

興味深い、革新的なアイデアをめぐって新しいコミュニティやサブグループが形成されています。

ソーシャル・ネットワーキングのコミュニティにそれぞれ固有の議題があるように、ソフトウェア・コミュニティにも同じように透過的（トランスペアレント）で共有の目標があります。例えば、Linux® のコントリビューター（貢献者）はライブラリーに関心があり、Eclipse のコントリビューターはそれぞれのプロジェクトに興味を持っています。今日のソフトウェア開発者には、既存のコミュニティに参加したり自らでコミュニティを立ち上げたりできる機会が多くあります。正式なガバナンスがないため、これらのコミュニティが非情な能力主義によって生み出され、支配されることもしばしば見受けられます。興味深い、革新的なアイデアをめぐって新しいコミュニティやサブグループが形成されています。これらのディスカッション・グループを観察したことがある人は誰でも、不正なアイデアや質の低いコードの提供者に対する辛らつな批判を耳にした経験があります。

テクノロジーのコミュニティは大きな影響力を持っています。Rational ソフトウェア・コミュニティ、Microsoft® ユーザー、システム管理者などです。

ハイライト

今日の状況が異なっている点は、
このようなコミュニティが形
成されるスピードにあります。

事実、これらのコミュニティでは参加者が開発にかかわるわけではありませんが、コミュニティ全体としての規模の大きさをゆえに、実際にデファクト・スタンダードを決定する最大の力となります。C 開発者、Java 開発者、personal home page (PHP) ユーザー、Web サイト開発者などのコミュニティが、コミュニティとしての活動領域を形成しただけではなく、ソフトウェアの作成方法の方向付けも行ったことを考えてみてください。今日の状況が異なっている点は、このようなコミュニティが形成されるスピードにあります。C テクノロジーが普及するまでには 10 年を要しましたが、Java テクノロジーはわずか 5 年で普及しました。さらに、比較的新しい Subversion テクノロジーは出現からわずか 2 年で、それまで 20 年間にわたって利用されてきた Concurrent Version System (CVS) テクノロジーに取って代わりました。

しかし、スピードは今日のコミュニティの特質の 1 つにすぎません。大きな転換点が訪れるのは、ある一定数のリーダーや利用者たちが、ある特定のテクノロジーについてそれが自分たちの求めているものだと判断したときであり、これは従来と変わりません。⁸急速に進化する今日のソフトウェア分野でまさに刺激的と言えるのは、ごくわずかな一部の人々が結果的に大きな転換をもたらすことになるという点です。個人のニーズが、変化をもたらすコミュニティの形成につながるのです。

ではここで、帯域幅やネットワーク接続が容易に利用できるようになったことによって推進されてきたさまざまな動向を見てみましょう。すなわち、サービス指向アーキテクチャー、コミュニティ・ベースのソフトウェア、およびそれらの成功をもたらした環境です。これらの動向の中には、長い年月の試練を経てその効果が実証されている多数の原理を見出すことができます。その原理には、複雑なテクノロジーからその本質を導き出す重要なプロセスなども含まれています。いわば、オッカムのかみそりの指針をソフトウェアの課題に当てはめるようなものです。例えば、Linus Torvalds はオペレーティング・システムを発明したのではなく、既存の UNIX® モジュールを単純化したのだということを覚えておくことが重要です。SOA は複雑なテクノロジーを多数網羅していますが、その中核となっているのはきわめてシンプルな概念、すなわち極小の同質サービスです。

ハイライト

コミュニティーによって推進される、複雑さから明確さへという革新的な流れが、本書で論じようとするすべての動向の原動力となっています。Rational グループでは以前から、成功するソフトウェアには一様に 3 つの基本的な特性があると考えています。つまり、コミュニティー、モジュール性、およびエンパワーメントです。各特性について、以下に詳しく解説していきます。

コミュニティー・ベースのソフトウェア開発

- 大規模プログラミングの進化⁹

大規模プログラミングは進化しています。

針の穴を潜り抜けられる天使が何人いるかについての議論と同種の哲学的論争で証明したいと考えることがもしあれば、ブロゴスフィア (blogosphere: ブログ界) にはそのような事例があふれています。グーグルで『『真の』オープン・ソース・ソフトウェアの構成要素は何か』の文言を含むスレッドを検索してみてください。数々の辛らつな非難のコメントを通り過ぎるとすぐに気付くのは、誰もが「純粋な」OSS であると同意できるフレームワークはごくわずかだということです。この地球上で最も成功したオープン・ソース・ソフトウェアだと言われている Linux テクノロジーですら、全部で¹⁰ 350 以上のディストリビューションに分かれており、それぞれの「オープン・ソース依存度」にはばらつきがあります。Ruby はほとんどすべてを 1 人の日本人が管理していますが、あらゆる人々が財政的支援を期待しています。現在の商魂たくましいオープン・ソース・コンリビューターたちは、慈善的であった Free Software Foundation (FSF) からは様変わりしています。ここで秘訣となるのが、細部にこだわった議論に陥るのではなく、ソフトウェアの各フレームワークや製品をその属性によって判断するということです。それではまず、OSS の最も基本的な定義の説明から始めましょう。

オープン

オープン・ソース・ソフトウェアの最適な定義は、その名前に内在しています。OSS では、開発者がソース・コードを見ることができます。しかし、以下に説明するように、オリジナル・コードにアクセスできることの利点は暗黙的であり、保証されてはいません。

ハイライト

変更の容易さ、修正の容易さ

この場合の修正とは、すべての OSS フレームワークで保証されているものではありません。OSS のライセンス供与体系は 30 数種類以上もあり、コードの変更を手を着けた場合に開発者に課される義務は各体系によって若干異なっているからです。また、コードの変更によってサポート契約に不利な影響が生じる場合もあります。また、「ハッカー」行為が自由であることも諸刃の剣です。ソフトウェアの目的を変更できることは開発者にとっては重宝なことですが、開発者がいなくなった後にそのソフトウェアを管理しなければならない IT マネージャーにとっては、頭痛の種です。

入手の容易性

商用ソフトウェア・ベンダーは過去数十年にわたり、自社のソース・コードを非公開にしてきました。コードの内部の仕組みにまで誰もがアクセス可能になれば、自社の競合上の優位性が失われてしまうと考えたからです（現在でもほぼ同じように考えられています）。こうした秘密主義はソフトウェアの適用にマイナスの影響を及ぼし、販売部隊および各種マーケティング資料はそれらを公にせざるを得ませんでした。ソース・コードはひとたび公開されると、誰もがそれ入手でき、販売価格は無償になり、そのソフトウェアで実際にどのようなことができるのかを誰もが評価可能になります。オープン・ソース・ソフトウェアは決して市販ソフトウェアではありません。

OSS の真のコストが発生するのは、インストールした後です。

無償

実際的な経験から、私たちはオープン・ソース・ソフトウェアというものは飼い犬の子犬のように無償であると考えています。しかし OSS の真のコストはインストールの後に、開発、デプロイメント、およびメンテナンス・コストとして発生します。これは決して目新しいことではありません。すべてのソフトウェアについて同様のことが言えます。しかも変更された OSS は他に類のない独自のソリューションであるため、それらのコストがさらにかさむこととなります。成功している OSS 企業の基本的なビジネス・モデルでは、サポート、サービス、トレーニングの各サービスを提供することで利益を上げており、それは道理にかなっています。つまり、子犬を飼うためには、世話をしたり餌を与えたりするコストが必要だということです。子犬の場合と同様、OSS もその選択を誤ると、高コストが継続的に発生することになりかねません。

ハイライト

これらはすべて、各レベルで適切なガバナンスが行われた結果として実現されるものです。

モジュラー性

モジュラー性は OSS に内在する特性ではなく、優れたソフトウェア開発全般に見られる特長です。これには、明確な境界線、合理的かつ周到なアプリケーション・プログラミング・インターフェース (API)、プラグイン可能なコンポーネントなどの拡張可能なフレームワークが含まれます。優れた商用ソフトウェア・パッケージと同様、優れた OSS フレームワークもモジュラー性を備えています。

コミュニティー・ベース

これは成功する OSS の核心であり、成功するすべてのソフトウェアの設計と実行の核心でもあります。コミュニティーとオープン・スタンダードは併存しており、主要コミュニティーもデファクト・スタンダードに影響を与えます。そこで問題となるのがベスト・スタンダードの確保であり、その鍵となるのが複数レベルでのガバナンスです。健全なバグ・トラッキング・システムの保持、活発な議論、そして実際的な標準の開発—これらはすべて、各レベルで適切なガバナンスが行われた結果として実現されるものです。

つまり、OSS は高品質なソフトウェアの既知の属性を数多く備えている可能性はあるものの、必須ではないということです。それらの属性とは、モジュラー性、可用性、合理的なコスト（取得コストと保守コスト）、オープン・スタンダード、堅牢性、パフォーマンス、および統制の取れた活発なユーザー・コミュニティーです。重要なのは、これらがオープン・ソース・ソフトウェア特有のものではなく、すべてのソフトウェアに当てはまる理想的な特質であるということです。その点では、OSS 開発者たちの前述の議論の精神にうなずけるものがあります。これらの望ましい特質を備えたソフトウェアを誰もが望んでいる一方で、これらの特質を実現する厳密な方法について、業界ではまだ合意がなされていません。

ハイライト

サービス指向アーキテクチャー (SOA)

SOA については、ここまで論じてきた OSS と同様の問題が多数ありますが、SOA の適用範囲には個々のフレームワークだけではなく、企業全体が含まれます。SOA の中核となっているのは、疎結合の相互運用可能なサービスによって構築されるアプリケーションの作成を可能にする、情報システム・アーキテクチャーのスタイルです。これらのサービスは、その基礎をなすプラットフォームやプログラミング言語に依存しない公式の定義（または規約）に基づいて相互に連携します。

SOA サービスは、適切に定義（カプセル化）され、発見可能な疎結合である必要があります。そうすることで、自律型で、抽象的で（再利用可能）、前述の属性を備えた複雑なサービスへの組み込みが可能になります。

SOA は特殊なテクノロジーであるという考えに陥りがちです。

現在 SOA 標準は定義が進められている状況であるため、SOA は特殊なテクノロジーであるという考えに陥りがちです。数年前には多くの人々が $SOA = Web$ サービスであると考えていました。偶然にも Web サービスが SOA サービスの主要要件を満たしたからです。今日、この $SOA \neq Web$ サービスは自明の理ですが、今度は $SOA = ESB$ (エンタープライズ・サービス・バス) という誤った考えを持つ人々が現れました。SOA と ESB は異なるテクノロジーですが、どちらの等式も同様に誤りです。実際には、Web サービスと ESB は、適切なサービス指向アーキテクチャーにおいて有益なツールです。適切な SOA がない場合に、ESB を通じて悪質な Web サービスや不十分なパターンが実行されれば、すぐに混乱が生じます。

ハイライト

SOA とは、1% のサービスと 99% のガバナンスです。

間違っているのは、単一のテクノロジー、あるいは標準を組み合わせることで SOA 構築が可能になるということです。真実はまったく異なります。実際には、トーマス・エジソンのあの有名な格言「天才とは 1% のひらめきと 99% の努力である」という言葉に近いものです。つまり、SOA とは 1% のサービスと 99% のガバナンスです。

SOA 構築を成功させることは**困難な作業**であり、多くの分散する部門や組織が個々の利害よりも全体としての利益を優先させることが求められます。このような目的の統一は、個人が集まった小さなグループでも達成が困難ですが、ましてや SOA 達成に向けて企業内のすべてのチーム間および組織間で協力し合う必要がある場合、変曲点の幾何学的累積を考えると、さらに困難なことです。

しかし困難であるとはいえ、不可能なことではありません。オープン・スタンダード、モジュラー性、および適切に管理されたコミュニティを活用すれば、SOA が成功するチャンスははるかに大きくなります。OSS の場合と同様、他のすべてのものが成功するために必要な特質はガバナンスです。ガバナンスが、標準規格への準拠とモジュラー・コンポーネントの規約を制御し、最終的には SOA コミュニティが有意義に協力し合うことができる能力を左右します。

ハイライト

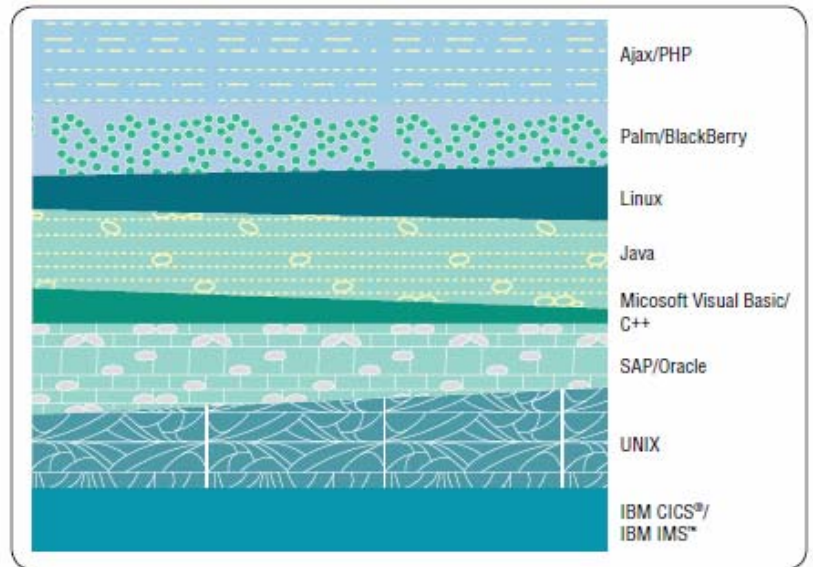


図 4. データ、サービス、およびプロセスのレイヤー

銀行や金融業界が SOA 導入後に実現したことを考えてみましょう。

数十年にわたってソフトウェア・レイヤーの下に覆い隠されていたサービス、データ、プロセスを開放するためには、企業にとって価値のある成果物を協力して発掘し、再利用することが必要です。銀行や金融業界が SOA 導入後に実現したことを考えてみましょう。口座、投資、クレジット・カード、抵当、ローンが共通のユーザー・インターフェースで管理できるようになり、これらのサービスは変更可能な返済プラン、自動引き落としおよび自動預金、分割繰り延べ、財務分析、税額計算などさまざまな機能と相互接続されています。これらのサービスは、掘り起こされた成果物が安全に世界中に転送され、失敗する確率はほとんどゼロに近いことを示しています。これが今日の金融業界で起きていることであり、状況はますます進展しています。

ハイライト

ガバナンスとは、監視フレームワーク以上のもの、すなわち一連の標準を指します。ガバナンスが実施されると、ソフトウェア・チームは妨害されず、むしろ強化されます。

後述しますが、Rational グループでは企業の SOA 開発プロセスを容易にするための取り組みを順調に進めています。IBM は、コミュニティによる開発、情報共有、アプリケーション・ライフサイクル全体にわたる適切なガバナンスが重要であると理解しています。

ガバナンスとエンパワーメント

多くの人々は、ガバナンスの話になると、サーベンス・オクスリー法や医療保険の相互運用性と責任に関する法律 (HIPAA) の順守について考えます。また、不順守や違反の場合の厳しい罰則についても考えます。ガバナンスという包括的な表現をソフトウェア・デリバリーに使用する場合、これは既存の法律的理解の実施よりももっと広い意味を持ちます。ガバナンスとは、監視フレームワーク以上のもの、すなわち一連の標準を指します。ガバナンスが実施されると、ソフトウェア・チームは妨害されず、むしろ強化されます。例えば、Eclipse プロジェクトの仕組みに見られるプロジェクト・ガバナンスのフレームワークにより、最も独創的で急速に成長しているオープン・ソース・コミュニティが強化されました。¹¹

つまり、成功するコンプライアンスとは、ソフトウェア・デリバリーへの注力ではなく、成功するガバナンスの結果として実現されるものです。C、Java、インターネット、XML、Linux、オープン・ソース・ソフトウェア、SOA - これらはいずれも、ガバナンスなくしては存在し得ません。これらのテクノロジー、アーキテクチャー、フレームワークはいずれも、標準が整備され、行動、議論、実装の限界を規定する規範が存在するからこそ、発展したのです。その結果、これまで以上に多くのユーザーがこれらのテクノロジーの安定性を信頼し、効率的に利用するようになりました。C 言語は、米国規格協会 (ANSI) と国際標準化機構 (ISO) のガバナンスに準拠していなければ、PC 世界の共通言語となることはなかったでしょう。インターネットは、ワールドワイド・ウェブ・コンソーシアム (W3C) と Internet Corporation for Assigned Names and Numbers (ICANN) に準拠しなければ普及しなかったでしょうし、Linux ソリューションも C 言語とインターネットがなければ普及しなかったでしょう。

ハイライト

OSS と SOA はこのような確固とした基盤の上に成り立っており、適切なガバナンスによって統制されるという点では、これら先駆のテクノロジーに劣りません。ガバナンスなくしては、多くの機会を逃すこととなります。

クルーズ・コントロールを使用すると、制限速度の上限を守りつつ車を運転することができます。これは、単に速度の上限を設定する「調速機(ガバナー)」とは異なります。クルーズ・コントロールは効果的なガバナンスによく似ています。どちらも、法定範囲内で最高速度を維持し、移動時間や細々としたアクティビティーを最小限に抑えつつ、最大限の効率性を発揮します。次に、この車が時限交通信号を利用する場合を考えてみましょう。これには電子化されたガバナンスが必要です。これは、ガバナンスを自分たちに有利になるように利用して、最大限のスピードと効率で目標を達成する方法を示しています。このシナリオに比べ、コンプライアンスを恐れている状態ならどうなるのでしょうか。あちこちの木陰に交通違反取締りの警官が待ち構えています。つまり、これと同じことが言えるのです。

自分たちにとって有利な方法でガバナンスを利用するということは、不要な作業を最小限に抑える標準、フレームワーク、製品に同意し、暗黙制御を利用してプロジェクトを正しく方向付けることを意味しています。ガバナンスは、すべてのソフトウェア・デリバリー、OSS 統合、SOA 実装を成功させるために不可欠であり、事実、ソフトウェアの開発、実装、メンテナンスのあらゆる側面で決定的な役割を担っています。このような一連の制御を賢く利用できるか否かは、それぞれの個人または組織次第です。

ガバナンスはあらゆるコミュニティーを結合する隠れた接着剤の役目を果たします。

ガバナンスは、すべてのコミュニティーを結合する隠れた「接着剤」の役目を果たします。コミュニティーには、共通の目標、共通の信念、共通の資産といった決定的な存在理由が必要です。そこには常に、暗黙の契約が存在します。多くのコミュニティーが設立されますが、長期にわたって成功を収めるコミュニティーはごく少ないものです。

ハイライト

ガバナンスとエンパワーメントは、健全なコミュニティの主要要素です。

コミュニティが、長期にわたって存続し、より広範な市場に影響を及ぼすようになるには、コミュニティのメンバーが知的バイタリティーを発揮できる健全性が備わっていなければなりません。健全なコミュニティの主要要素となるのが、ガバナンスとエンパワーメントであり、選択のモジュラー性による個々の活用が、今後のソフトウェア・デリバリーの成功の鍵となります。

将来のソフトウェア・デリバリー

「T 型フォードはどんな色でもお求めになれます。黒であれば。」
—ヘンリー・フォード

20 世紀初頭に発売された T 型フォードは、手頃な価格設定で大量生産され、宣伝にたがわぬ優れた性能を備えていました。適切なタイミングで発売された適切な製品でしたが、オプションがまったくありませんでした。サイズも 1 つ、色も 1 色のみで、選択の余地はありませんでした。

話を 2007 年に進めると、自動車業界がいかに変化したかが分かります。ユーザーは好みの仕様を工場に直接伝えて、オンラインで新車をカスタマイズすることも可能です。オプションとして、GPS 衛星測位システム、衛星ラジオ、地上波 CD または MP3、温度調節器、レイン・センサー、燃費調整、ハイブリッドおよび代替燃料制御、ナビゲーション・システム、駐車支援、スペース・ビジュアライゼーションなど、ボンネットの下に備わったコンピューターの能力を証明する多くのオプションが用意されています。もちろん、車体の色も選ぶことができます。自動車業界は、シャシーと製造サプライ・チェーン全体の双方でソフトウェアを統合し、オンデマンドでの製品提供を実現しました。

ハイライト

私たちは、自分たちのテクノロジーを自分たちのニーズに合わせて構成することが可能になると考えています。

Amazon.com や Netflix など、ジャスト・イン・タイムの小売業者の成功例も見てみましょう。これらの企業は、今では実店舗によるビジネスを上回る、数千にも及ぶ在庫を用意し、特殊な需要への対応も可能です。私たちは、カスタマイゼーションの時代に生きています。自分たちのテクノロジーを自分たちのニーズに合わせて構成することが可能になると考えています。システム・インテグレーションやソフトウェア・デリバリーが行われる時代にあって、そのようなことができないと考える理由はないのです。カスタマイゼーションにはコストがかかることを理解し、そのコストを支払うことも厭いません。

知的財産のサプライ・チェーン

ロング・テールの時代へようこそ。クリス・アンダーソンの同名の著書によって広く知られるようになった「ロング・テール」理論（この著書は Web サイトで販売され、アンダーソンの理論の正しさを日々証明しています）によると、将来のビジネスは多数派の支持者向けの商品は少なくなり、ごく少数のグループまたは個人向けにカスタマイズされた商品が増えるということです¹²。これはソフトウェア・ベンダーにとっては、高額な製品の開発が減少し、カスタマイズ可能なコンポーネントの開発が増加するということを意味しています。つまり、カスタマイズ可能な多数のプラグインを備えた汎用フレームワークです。

ロング・テール論は、供給側に重点をおいた理論です。テクノロジーによって、供給側のコスト・モデルは小売店の店頭から配送センターへの転換が可能になり、その結果、より多彩な商品を在庫として持つことが可能になります。さらにそれにより、サプライヤーは多彩な選択肢の提供が可能になり、需要曲線のロング・テールを獲得することができるとしています。

ハイライト

この現象をソフトウェア・デリバリーに当てはめて考えてみると、需要側のさまざまな問題が、供給側のソフトウェア・ベンダーに影響を及ぼしています。ある意味では、SOA は選択肢を求める要望の現れであり、同時にソフトウェア・アーキテクチャーの堆積層に埋もれた柔軟性の欠如に対処するためのメカニズムでもあります。これはまさに、知的財産のサプライ・チェーン構築にあたって、情報、データ、成果物を明確にすることにほかなりません。

この課題をソフトウェア開発で考えてみると、大半のソフトウェア会社は、さまざまな開発上の役割について使用パターンのロング・テールへの効率的かつ効果的な取り組みができていません。開発およびテスト組織の役割を、自分たちが使用するツールの特徴や機能に画一的に合わせるといふ本末転倒の状況が、あまりに多く見られます。ソフトウェア・ツール・ベンダーは、テスターがアプリケーションにおけるビジネスの意図ではなくパフォーマンスを重視しすぎることに不満を抱いています。しかしソフトウェア・ベンダー側にも落ち度があります。彼らは機能を限定するツール群を提供しているため、それが機能に対する固定観念を助長しているのです。

このような製品を開発する最善の方法は、コミュニティの声に耳を傾けることです。

前進するための最良の方法は、ソフトウェア・プロバイダーがこの等式の需要側に立ち戻って取り組みを進め、それによって、供給側がこの課題に対処するための選択肢と柔軟性を提供できるようにするにはどうすればよいかを理解することです。このような製品を開発する最良の方法は、コミュニティの声に耳を傾けることです。

ハイライト

Eclipse テクノロジーは、200 以上の IBM 製品に組み込まれています。

コミュニティの設立と参加

2001 年に IBM がオープン・ソース・コミュニティ向けに Eclipse をリリースしたときに目指したのは、開発者に Java テクノロジー・ベースのよりオープンなミドルウェアを提供することであり、異機種ツールを組み合わせた顧客の開発環境の実現でした。この目標は、堅牢なシック・クライアントの構築、およびすべてのサービスをプラグインにアクセス可能にすることにより、達成されました。これらのプラグインは、既存の Eclipse プロジェクト、サードパーティー・ベンダーの製品、あるいは独立ベンダーの製品として開発することが可能でしたが、すべては共通のプラットフォーム上で、すなわちソフトウェア・ツールのエコシステムを包含して構築されました。

Eclipse は間もなく、世界で最も人気の高いエンタープライズ Java 開発プラットフォームとなり、現在では Java 開発者の 65% から 75% に利用されています¹³。これは、Eclipse がユーザーだけではなくベンダーにも人気が高いことを証明しています。Eclipse テクノロジーは 200 以上の IBM 製品に組み込まれており、単一の開発およびデプロイメント用プラットフォーム上で標準化されていることが、製品の相互運用性向上に役立っています。

Eclipse プロジェクトの目標として定められているトランスペアレンシー、予測可能性、継続的なフィードバックにより、プロジェクトの健全性はきわめて高い水準に維持されています。プロジェクトの健全性は、高品質なソフトウェアの開発にとって、最も重要な要素です。健全なコミュニティ、強い団結心、堅牢なビルド、安定したマイルストーン、将来性の高いベータ版、継続的なテストなどがすべて、Eclipse プロジェクトの健全性に役立っています。

ハイライト

ガバナンスが、これらの活動をつなぎ合わせる上で必要な相乗効果を生み出す鍵となります。

Rational グループの過去 6 年間にわたる Eclipse プロジェクトへの取り組みは、結局のところプロジェクトの健全性がより重要であり、これがソフトウェアの品質を補完するものであることを実証しました。プロジェクトの状況は常に変化するため、これを健全な状態に維持するのはチームの責任です。いわば村のようなものです。オープン、責任、包括性に価値を置く健全な開発者コミュニティは、常に進化し続けるソフトウェアへの取り組みで生じるさまざまなストレス、例えば予期しない変更や新しい要件、常に変化する目標などにも、より適切に対応できます。

Rational の方向性

IBM Rational グループでは、ソフトウェア開発の将来は、オープンで動的に定義されたコミュニティが設定とカスタマイズを最大限自由に行えるようになり、組織の俊敏性と即応性を支援するモジュラー型のソリューションを構築できるようになることで拓かれると考えています。分散したグループと異機種テクノロジーのエネルギーを発揮させる力となるガバナンスが、これらの活動をつなぎ合わせる上で必要な相乗効果を生み出す鍵となります。開発組織がビジネス・イニシアティブへの取り組みを進める速度は、単一の標準やテクノロジーで決まるものではありません。推進要因となるのは SOA や OSS ではありません。健全なコミュニティを実現し即応性を向上させるのは、複数の推進要因に対する効果的なガバナンスです。

この効果的なガバナンスは、プラットフォームにかかわらずアプリケーション間でリアルタイムの情報共有を可能にする共通のデータ構造という形で実現されます。これはコミュニケーション統合の方法であり、これにより、ビジネス・アナリストから実稼働の専門家にいたるまでプロジェクト・チームのあらゆるメンバーが、現在進めているプロジェクトや使用しているアプリケーションのコンテキストで互いにメッセージを伝達することが可能になります。効果的なガバナンスはまた、世界中に分散したチームによるソフトウェア開発を可能にし、これらの製品をシームレスに連動させる力となります。

ハイライト

ソフトウェア・デリバリーの将来像は、IBM Rational Jazz プロジェクトによく似ています。

効果的なガバナンスでは、フレームワークの中にコンプライアンス機能がすでに組み込まれているため、行政機関による監査に対しても追加の対応を必要とすることなく監査に合格することができます。これは、役割第一の考え方からチーム第一の考え方への根本的な変化です。その鍵となるのは、プロジェクトのあらゆるフェーズにおいてチームを再配置して、より包括的なチームとすることです。

ソフトウェア・デリバリーの将来像は、IBM Rational Jazz プロジェクトによく似ています。これは、ソフトウェアおよびシステムデリバリーライフサイクルの全体にわたってチームのタスクを統合するテクノロジーです。組織は Jazz フレームワークを使用することにより、要件変更の影響を評価してその変更がビルドにどのような影響を及ぼすかを判断することが可能になります。その結果、組織は何を変更する必要があるか、誰がそれを変更する必要があるかをより正確に判断できます。このライフサイクルの統合化にはツールの支援によるプロセス・ガイダンスが盛り込まれます。このガイダンスでは、チームがどの開発プロセスを使用することにしたのかをツールが理解します。また、このライフサイクル統合は自動化されるため、チーム・メンバーはプロセスを妨げることなくこれに従うことができます。

チーム第一のライフサイクル・ガバナンス・プラットフォームの価値を完全に実現するためには、組織はそのプラットフォーム上のオープンな標準に基づき、柔軟なモジュラー型コンポーネントも提供しなければなりません。これらは互いに単独では存在できません。既存の特質や機能の分解とコンポーネント化は平行して行われ、どちらも同じように重要な作業です。

これはコミュニティとして行われるべきものであり、誰も単独でできることではありません。その点では Eclipse によく似ています。IBM Rational グループがコミュニティ駆動型のソフトウェアを新たなレベルに進化させ、Linux、Eclipse、Geronimo などの OSS プロジェクトと、企業のソフトウェア関連組織内におけるコミュニティの双方に対応可能にした理由も、そこにあります。

ハイライト

IBM Rational グループのプロセス・ガイダンス機能は、俊敏なプラクティスから構造化されたアプローチにいたるまで、さまざまな複雑さに対応できるよう設計されています。また、ごく小規模なチームから大規模な分散組織まで、開発環境の規模に合わせて拡大できます。組織は、ソフトウェア・デリバリーのためのツールやテクノロジーをさまざまに組み合わせたり調整することが可能になり、開発環境変革のあらゆる段階において最適なソリューションを得ることができます。

結論

私たちはエキサイティングな時代、すなわち変化の激しい時代に生きています。そしてソフトウェアは拡大し続けるニーズに対応するため、常に進化を続けています。これまでもそうであったように、私たちはこの時代に適応しなければならず、適応できなければ一蹴され、数年といわず数カ月で時代遅れになってしまいかねないというのが厳しい現実です。しかしこれは、すべてが変わってしまったということの意味するわけではありません。まったくその反対です。ソフトウェア開発者はいまだ、同じ現実と直面しています。それは、ソフトウェアのモデル化とビジネス要件の統合はいまなお不可欠なものであり、高品質で実用的なソフトウェアは、そうではないソフトウェアより常に優れている、ということです。変わったのはもちろん、スピード、範囲、そしてカバーする領域です。

答えは、俊敏性にあります。

ソリューション自体はまったく新しいものというわけではありませんが、このソリューションには新しいアイデアが盛り込まれています。ソフトウェア・デリバリーをカスタマイズして、これまでになく細分化された顧客の需要に応えることは重要ですが、際限なく増え続ける特質や機能を提供することに囚われると、破綻してしまいます。そうならないための答えは、俊敏性にあります。これは、意味のあるデータとそれに関連するアセットおよび成果物を既存の知的財産レイヤー上に取り出す能力のことです。ソリューションには、必ずさまざまな種類、ブランド、レイヤーのソフトウェアが含まれるからです。

ハイライト

将来のソフトウェア・デリバリーの主要 3 原則として、コミュニティ、モジュラー性、エンパワーメントが挙げられます。

データを組織にとって実用的なものにするということは、企業全体にわたって既存の成果物を細分化して再利用することを意味します。つまり企業は、顧客が自社との関係を調整できるように、敏捷性と順応性を維持しなければなりません。成功するかどうかは、将来のソフトウェア・デリバリーの主要 3 原則であるコミュニティ、モジュラー性、エンパワーメントにいかに関心していくかにかかっています。

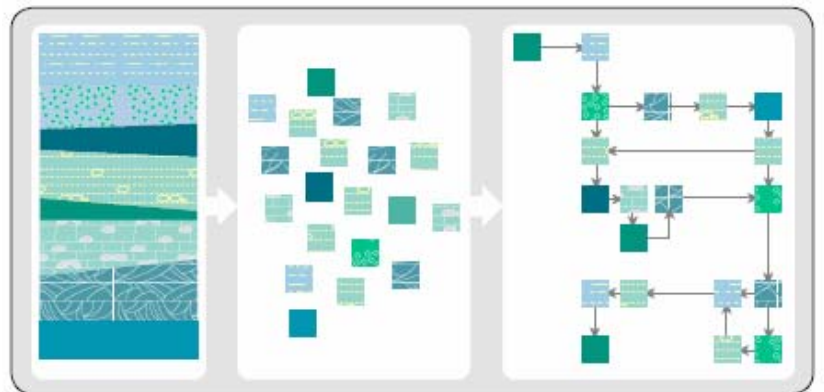


図 5. データ、サービス、プロセスのレイヤーを細分化して再利用

将来のソフトウェア・デリバリーは、サプライ・チェーンによく似たものになります。すなわち、疎結合で効果的に統制され、SOA および GDD を組み込むことが可能な水平な組織であり、必要なデータ、プロセス、サービスを合理的な形で明確化し、投資回収率 (ROI) の黒字化を達成します。今後は、個々のアプリケーション、オペレーティング・システム、ソフトウェアの選択肢に対する関心は小さくなり、代わってサービスのデリバリー、システムのモジュラー性、および重要な選択にあたって定義済みの柔軟なコミュニティが関与することに対し、より多くの関心が向けられるようになるでしょう。

ハイライト

IBM Rational グループは、さまざまな思い切った策を講じています。

IBM Rational グループでは、この将来の展望の実現のために、さまざまな改善策を講じています。コミュニティ、モジュラー性、およびエンパワーメントにより、Rational 現行製品、Jazz フレームワークをはじめ、将来のあらゆる Rational 製品オフリングの開発をより一層促進していきます。

ソフトウェア・デリバリーの将来は希望に満ちています。IBM Rational グループは今後も引き続き主導的な役割を果たしていくことをお約束します。



脚注

1. 「The World is Flat: A Brief History of the Twenty-first Century」 Thomas L. Friedman 著、Farrar, Straus and Giroux.、2005 年
2. 「Service Oriented Architecture: The Foundation for Digital Business」 Diego Lo Giudice 著、Forrester Research、2006 年
3. 「Enterprise Service Bus and SOA Middleware」 Aberdeen Group、2006 年 7 月
4. 「Open Source in Global Software: Market Impact, Disruption, and Business Models」 IDC、2006 年 7 月
5. Nucleus Research 発行の各種 ROI レポート、2003 年 (<http://nucleusResearch.com>).
6. http://en.wikipedia.org/wiki/Moore's_Law.
7. http://en.wikipedia.org/wiki/Metcalfes_law.
8. 「The Tipping Point」 Malcolm Gladwell 著、Little, Brown & Co.、2002 年
9. http://en.wikipedia.org/wiki/Programming_in_the_large
10. <http://distrowatch.com>
11. 「The Eclipse Way」 Erich Gamma、John Wiegand 共著、2006 年
12. ロング・テールのその他の条件については、http://www.longtail.com/the_long_tail を参照してください。
13. この数字には、商用 IDE に Eclipse コンポーネントを組み込んだ IBM Rational Application Developer ソフトウェアなどの製品も含まれます。

販売店、弊社営業担当員または、ダイヤルIBM (0120-04-1992) へ。受付時間：月～金9:00～

18:00 (祝日12/30～1/3 を除く) 携帯電話でおかけのお客様は下記の電話番号でご利用ください。
ダイヤルIBM 03-6220-8002 (この場合通話料金はお客様のご負担となります。)

Copyright IBM Japan, Ltd. 2007
日本アイ・ビー・エム株式会社
〒106-8711 東京都港区六本木3-2-12
Produced in Japan
Dec 2007
All Rights Reserved

このカタログの情報は2007年2月現在のものです。内容は事前の予告なしに変更する場合があります。表示画面および印刷帳票の出力例のうち、特に断わり書きのない出力例のデータ部分は全て架空のものです。画面ははめ込み合成で実際の表示とは異なります。製品、サービス等詳細については、弊社もしくはIBM ビジネスパートナーの営業担当員にご相談ください。

CICS, IBM, IBM ロゴ, IMS および Rational は、International Business Machines Corporation の米国およびその他の国における商標です。

JavaおよびすべてのJava関連の商標は Sun Microsystems, Inc. の米国およびその他の国における商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

RAW10983-USEN-00 (英文原典)

RAW10983-JAJP-00