GeneXus™

THE FIRST INTELLIGENT TOOL

GeneXus概要

最終更新：2007年8月

ARTech™

www.genexus.com

MONTEVIDEO – URUGUAY	Av. 18 de Julio 1645 P.4	+598 2 402 2082
CHICAGO – USA	400 N. Michigan Ave. Suite 1600	(312) 836 9152
MEXICO CITY – MEXICO	Calle Leibnitz N°20, desp. 801	+52 55 5255-4733
SAO PAULO – BRAZIL	Rua Samuel Morse 120 Conj. 141	+55 11 5502 6722

目　　次

紹介

GeneXusはARTech社が開発したインテリジェントツールであり、アナリストとユーザをアプリケーション開発のライフサイクル全体を通して支援することを目標としています。

設計とプロトタイプ・モデルの開発とテストは、Windows、WindowsNT/2000/XP/Vista環境のPC上で行われます。システムのプロトタイプが一旦ユーザから全面的に承認されると、データベースとアプリケーション・プログラムが自動的に生成され、以後のメンテナンスは完全に自動化されます。このようにしてシステム開発環境をもたらすツールです。

GeneXusの基本的な考え方は、自動化できるものはすべて自動化する、ということです。データの正規化と設計、データベース及びアプリケーション・プログラムの生成とメンテナンス、これらが自動化の対象です。この方法でアナリストたちを時間のかかる退屈な作業から解放し、プログラミングツールには絶対不可能な作業、つまり**「ユーザの抱える問題点を理解すること」**に専念できるようにします。

設計工程用には副次的機能が備わっており、これがシステム設計のドキュメントを自動生成し、しかも永久的に更新してくれます。

この資料の目的は、GeneXusの特質とその問題解決能力について、読者の皆様にお知らせすることにあります。

この資料の構成は以下の通りです。

- 理論上の問題…この節ではシステム開発における伝統的な方法論とインクレメンタル開発手法について、両者を比較しながら解説しています。
- インクレメンタル開発手法の実現：GeneXus
- GeneXusの他に類を見ない特質
- 世界各地に拡がるGeneXusユーザ

理論上の問題

伝統的な開発方法論と関連する問題

アプリケーション開発の伝統的な手法が依拠しているのは、**「会社の安定したデータモデルを造ることは可能」**という考え方です。 この仮定の下、最初に行う作業は会社の現実の姿を抽象化するための調査とデータ分析であり、最終的な目標は会社のデータモデルを得ることです。一旦会社の情報がデータベースに投入されると、データモデルはデータベースによって簡単に影響を受けてしまいます。次の作業は、データモデルに表現された知識を最もよく反映できる形でデータベースを設計することです。

会社の本質的な情報を反映したデータモデルが得られれば、次のステップは機能的な面から会社を分析することになります。会社の現実の機能の分析により、会社の現実を記述した機能仕様書を得られれば、望ましいでしょう。しかしながら、大半の開発方法論はデータベース・ファイルによって機能的な仕様書を得ています（または、より正確には、初期の段階ではデータモデルのエンティティと同等です）。

データベースと機能仕様書を入手した後は引き続き、最後のステップとして機能を実装することになります。この作業では第3または第4世代言語、言語ジェネレータ、インタープリタといった伝統的な選択肢を使うことになります。

しかしながら、これらの実装時の選択肢は共通の問題を抱えています。つまり**「会社の安定したデータモデルを造るのは可能」**という**誤った前提**に基づいているということです。

実際のところ、会社の詳細なデータモデルを抽象化された形で描き上げることは**不可能**ですし、要求されたレベルの対象を詳細な形で目にすることもできません。その理由は、社内の全知識を所有する権限を持つ人など存在しないからです。

従って会社の業務を完全に視覚化しようとすると、一般従業員から経営者（通常彼らをユーザとみなすとして）まで、多数の関係者にインタビューし、質問する必要があるでしょう。ユーザ側の必要性に対応する結果として、アプリケーションのライフサイクル全体を通して、モデルの変更も一緒に行わなければならないことは明白です。

たとえ担当アナリストが実現すべき機能を正確に把握し、かつ最適なデータベースを作成できたという理想的な状況を仮定したとしても、モデルは企業の発展に対応していく必要がありますから、静的な状態に留まるわけにはいきません。

機能仕様とデータベース要件とが完全に独立しているならば、これはさしたる問題ではありません。しかしながら、機能仕様はデータベース構造に完全に依存しています。ですからデータベースへの変更は、対応する機能仕様（手作り/手作業で作った）に対して必然的に影響を及すことになるのです。

これが意味するのは、変更が頻繁に発生するとメンテナンス費用が非常に高額になるということです。「理論的には」開発用に割り当てられた資源の80%が、現実は実装されたアプリケーションへのメンテナンス費用に使われているというのは周知の事実です。

現在、もし読者の方が大規模なアプリケーション開発に携わっているとするならば、状況はもっと悪いでしょう。つまりメンテナンス作業は、新しいコンポーネントを実装するはるか前に始まっています。従って実際の開発プロジェクトの規模と比較してみると、開発費用は指数関数的に上昇していきます。

データベース上の変更を決定し反映させる際に、その変更によって影響を被るすべての処理がしかるべく対応するように仕向けるのは非常に難しいので、通常は余分なファイルを追加することになります。このような解決策を採用すると、結果としてシステムの品質は大幅に低下し、メンテナンス費用は激増します。

インクレメンタル方法論

この問題を解決するには、別の手法が必要です。この手法は前述の基本前提を、**「会社の中に安定したデータモデルなど存在しない」**という前提と取り替え、現在使われている開発手法から一歩踏出して**「インクレメンタルな開発哲学」**を提案し、成功への真の解決手法を提案します。インクレメンタルな開発哲学というのは、大きな問題を直接解決しようとするのではなく、より小さな問題に分割し一つずつ解決していく、という意味で非常に自然な解決手法のように見受けられます。

この哲学はシステムのメンテナンス費用にどんな影響を与えるのでしょうか？

従来型の開発方法論を採用すれば、変更に伴う影響は必ず無視できないものとなります。その理由は、このアプローチには必然的に、データモデルへの定期的な変更及び当初の想定をはるかに超えるメンテナンス費用が伴うからです。

しかしながら、以下の点にお気付きになるでしょう。

会社全体のデータベース構造は分らなくても、ユーザは自分が日々取り扱っているデータビューを明確に把握しています。言い換えれば、各ユーザは会社の現実に対する独自の視点を持っているのです。

データビューには多くの形式があり、画面や帳票等、アプリケーションと外部との接点を通した形で認識されます。ユーザが明確に認識できるものはすべてデータビューなのです。

これらのビューから得られた情報を、どのようにデータモデルの構築に利用できるでしょうか？

この課題をいかにして数学上の問題と捉えることができるでしょうか？もしシステム開発作業の広い範囲を自動的に解決する数学上の表現形式を得ることができ、この課題を数式で厳格に仕様として表現できるならば、結果として、アナリストの作業を簡素化することができるでしょう。これは興味深い考え方です。つまりデータベースに関する完全な知識が得られれば、必要なビューをすべてのユーザに対して提供することができます。言い換えれば、データベースはすべてのユーザビューを満足させる義務があります。実例を見てみると明白ですが、ユーザの視点をまとめたデータがあれば例外なく、最小限正規化され、独自の形式も持ちうるデータベースを生成することができます。当該の課題が数学上の性質を帯びていることが明らかになった時点で、まずこれを解決し、その上でデータベースを決定することになります。これを実現するために、ARTech社は独自の調査を経て、自動化のプロシージャを開発しました。

しかし、どうやったらこの理論を実装することができるのでしょうか？

ユーザビューの中に存在する知識を収集し、知識ベースに移すことによって実装できます。知識ベースは収集した知識に基づきデータモデル、データベース、およびアプリケーション・プログラムを設計し、自動的に生成します。知識ベースと伝統的なデータディクショナリとの最たる違いは、知識ベースには推論する能力が備わっているという点です。ある要素を取込むことが、他の値を決定するための推論トリガーとしての機能を果たすことになります。

知識ベースを達成できれば、データベースとアプリケーション・プログラムは決定論的な存在となります。

- データベースとアプリケーション・プログラムが自動的に生成されます。
- ユーザビューにおける変更は自然の成り行きですから、
  データやプロセス上の変更がもたらす影響を完全に測定し、確定できます。
  変更への対処として、以下の二つのものが生成されます。
  - 必要なデータ変換プログラム
  - 変更によって影響を受けたアプリケーション・プログラムの新しいバージョン

GeneXus:インクレメンタル開発の実現

GeneXusはこの理論を実現可能にします。

GeneXusは「ユーザのビジョン」を起点とするツールです。つまりユーザの知識を収集し、知識ベース内部で体系化します。その知識ベースを使って、データベース構造やアプリケーション・プログラム（ユーザが自身のビジョンに沿って働くために必要なプログラム）を100%自動的に設計、生成、メンテナンスする能力を備えています。

GeneXus は非常に強固な数学的基盤の上に構築されています。

GeneXusに潜む最たる強みは何でしょうか？ それは業務システムの知識を非常に有効に自動管理することです。

GeneXusは純粋業務知識を使って稼働しています。この純粋業務知識を使ってプログラム（伝統的なソフトウェア）を生成し、知識を人間に理解可能なもの（作成されたら二度と更新されないような追加のドキュメントは不要）にできます。さらに知識を使って自動的に作動します（他のデータソースから取り出した別の知識と統合し、拡大し、他者が各々のアプリケーションの中でそれを統合できるように承認する）。従ってGeneXusを利用することで、「ソフトウェアビジネス」よりもワンランク上の「知識ビジネス」が実現可能になります。

この純粋業務知識を用いて作業する利点は、もう一つあります。複数のプラットフォームやアーキテクチャ上で稼働するアプリケーションを生成できる可能性が生じること、さらに特筆すべきことに、技術の諸変化に対するある種の保険にもなりうる点です。例を挙げると、GeneXusユーザは8年ないしは10年前にはテキストスクリーンやかなり原始的な技術を用いてiシリーズ用のアプリケーションを開発していました。ところが今や.NETアプリケーションを容易に開発する目的で、これまでにGeneXusで蓄積したそれらのアプリケーション開発に関する知識を活用することが可能になったのです。もちろん開発された時点では、当時の稼働環境と全く異なった環境など、誰も考えつくことすらできませんでしたが。

GeneXusを用いてアプリケーションを開発する際の最初のステップとしては、まずユーザの視点を起点にしてアプリケーションを**設計**します（ユーザの視点からシステムは知識を収集し、体系化していきます）。

第2ステップは、アプリケーションの**プロトタイプ**作成です。GeneXusはプロトタイプ環境に応じたデータベースとプログラムを生成します。生成後にアナリストとエンドユーザがプロトタイプを実行、テストします。

プロトタイプの目的は、生きたアプリケーションを介してユーザと対話する機会を提供することにあります。そうしてこそ、改良点を提案したりエラーを検出したりすることが可能なのです。設計者はこのフィードバックに基づいて設計工程に戻り、必要な修正を加えて新たに操作可能なプロトタイプを提案します。 このサイクルを通常、設計/プロトタイプサイクルと呼んでいます。

プロトタイプが全面的に承認されると、次の段階は実稼働環境への**実装**へと移ります。この段階で

GeneXusは実環境で動くデータベースとプログラムを生成します。

一言でまとめると、アプリケーションは最初に設計され、次にプロトタイプ化され、最後に実装されます。設計者はユーザからの定期的なフィードバックに基づいて、どの開発段階においても極めて自由に微調整を行うことができます。

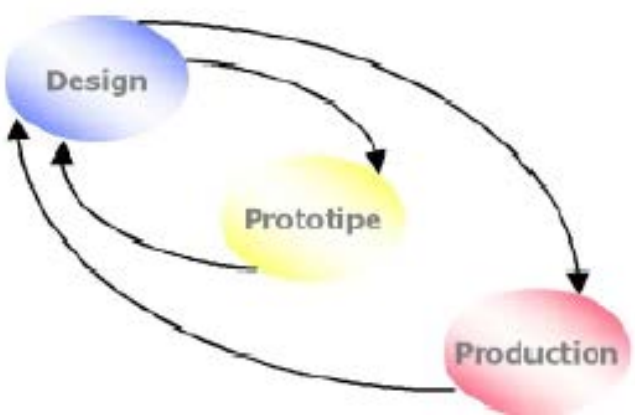

図1. 設計/プロトタイプサイクルと設計/実稼働サイクル

それでは各開発段階について詳細に説明しましょう。

設計

この工程ではアナリストとユーザが共同で作業します。ユーザ・オブジェクトと関連データビューを特定し、記述する作業です。

実際の作業はユーザのオンサイト、オフサイトのいずれかで進行します。この段階では、エンドユーザが熟知している用語やコンセプトを用いて低レベルの抽象化を実現します。

このユーザ参画型の開発は、ユーザに対して非常に意味のある肯定的な効果をもたらします。ユーザ（複数の場合もある）はより積極的に開発に参加するようになります。ユーザとの共同作業の場では、ユーザが進行状況を常時フォローしているため、通常のシステム開発を凌ぐ高品質のシステムを創り出すことが可能です。

GeneXusはユーザが現実のオブジェクトに対して抱いている視点に基づいて、知識を収集することができます。この知識はGeneXusのオブジェクトを通して得られます。オブジェクトとは**トランザクション、レポート、プロシージャ、ワークパネル、テーマ、ウェブパネル、メニュー、データビュー、**そして**データを蓄積するトランザクション（データウェアハウシング）**です。

設計と称される第一段階ではこのように、前述のオブジェクト類を特定し記述します。GeneXusはこれらの記述を前提として、収集した知識を自動的に体系化し、知識ベースを段階的に構築していきます。

この知識ベースは設計情報全体を格納した単一のリポジトリーであり、これをベースにGeneXusは物理データモデル（テーブル、属性、インデックス、冗長性、参照完全性規則、等）及びアプリケーション・プログラムを創り出します。

アプリケーションの分析と設計は、GeneXusオブジェクトの記述に基づいて実行されます。

さて次にGeneXusのオブジェクト類のうち、より重要性の高いものについて詳しく説明しましょう。

トランザクション

トランザクションとは対話型の処理であり、ユーザがデータベースのデータを作成、変更或いは削除する際に使用します。通常ユーザは「画面」(WinないしはWeb)として理解しています。
例:

- コーポレートデータベースにおける顧客エントリーを作成、変更或いは削除する時の画面
- 送り状画面…ユーザが送り状を作成、印刷する処理

この単一画面は高度な柔軟性をユーザに提供しているため、ユーザはメインメニューに戻らずとも情報を挿入、更新、削除、あるいは印刷といった様々な操作を実行することができます。

レポート

レポートはデータベースからデータを抽出する際に用いられます。レポート機能によって作成されたアウトプットはスクリーン、プリンタ(一般帳票)或いはファイルに送ることができます。
このオブジェクトは極めて単純なもの(顧客データ表)から複雑な帳票(複数の制御ブレイクとパラメータを持った多重データベースへのアクセスを必要とする)に至るまで、あらゆるレポートを作成することができます。
にもかかわらず、レポートはデータベースを更新することはできません(そのようなプロパティは割り当てられていないため)。さらにデータベース上でダイナミックなデータ抽出を実行するため、私たちはGXqueryというツールをご紹介しています。詳細についてはwww.genexus.com/gxqueryをご覧下さい。

プロシージャ

このオブジェクトは広範なレポート機能プラス、データベースの更新機能を用意しています。プロシージャは通常、以下の3種類の処理で用いられます。

- バッチ処理:例えば一定の日付に先立って発行され、かつ支払済みの送り状をすべて削除する、といった具合です。

- 汎用目的のサブルーチン:例えば指示された金額を文字で返すルーチンです。1010という数字が与えられると"千壱拾"という文字列を返します。

- アプリケーションサーバ処理:多階層アーキテクチャのための処理(一般にC/SQL或いはJavaで書かれています)。

ワークパネル

ワークパネルとは、ユーザが対話的にデータベースから情報を検索するための画面です。対話形式の処理は、ユーザが決定を下すために必要とする諸要素を提示しています。ですからユーザが仕事にコンピュータを使うケースが増えるにつれて、さらに洗練された対話形式の処理が必要となります。ワークパネルは、完全なユーザ・ダイアローグを定義する際に用いられます。

これは一例ですが、ワークパネルは顧客リストを表示し、ユーザが希望する場合はさらに、特定の顧客の登録済み送り状や債務額を表示することができます。

ウェブパネル

ワークパネルセットと類似のオブジェクトですが、インターネット/イントラネット/エクストラネット環境で動くブラウザ上のアプリケーションが必須です。

テーマ

このオブジェクトはテーマエディターを通して生成、更新されます。テーマエディターは字体やテーブル、ボタン等、アプリケーションのあらゆる視覚要素を定義するグラフィック・デザイナーです。またテーマはGeneXusのオブジェクト・フォームと関連づけられています。
これらのCSSベースのテーマ値は実行時に変更できることから、Webアプリケーションはよりダイナミックかつカスタマイズ可能なものとなっています。

メニュー

メニューとは固定項目の一覧であり、ユーザは実行したい項目を選択できます。

データビュー

データビューはコンポーネントもしくは外部ファイルと、GeneXusのテーブルの間の関連図を作成し、それらがすべてGeneXusのオブジェクトであるかのように処理します。

GeneXus は純粋業務知識に基づいて稼働します

物理データモデルはトランザクション・オブジェクトがすべて定義された後でのみ、データベース正規化基準に則って、冗長性の無い第３正規形を適用した関係データベース理論に基づき設計されます。GeneXusがこの正規化を自動的に実行し、メンテナンスします。

ですがアナリストが冗長性を定義することは可能ですし、定義後はGeneXusが自動的に管理してくれます。

GeneXusの情報リポジトリーには、設計仕様が抽象的な形式で格納されています。この設計仕様は環境には依存していません。ここで明確にしておきたいのは、情報リポジトリーが様々な環境用の機能的に同等なアプリケーションと対話し合える条件下では、単一のリポジトリーを生成できる、ということです。

複数プラットフォーム／多階層アーキテクチャ

結果として、例えば集中型iシリーズアプリケーション（GeneXusを100％用いて開発された）を開発したユーザは、元のオブジェクトを一切変更することなく、クライアント/サーバ環境上で動くアプリケーションを手に入れることができます。

ごく最近の例ですが、複数のプラットフォーム上で動くアプリケーションを生成する、言い換えれば、同じアプリケーションを複数の環境下で動かす必要性が生じました。例えば、ある銀行システムアプリケーションの場合、本行ではiシリーズ上で、各支店ではPCネットワーク上で、それぞれ動かすことも可能です。

現在では、クライアント/サーバとインターネット/イントラネット/エクストラネット環境の発展的な利用に伴って、新たなニーズが発生しています。これはどういうことかと言いますと、ひとつのアプリケーションが、

あるプラットフォーム上で動く部品と、それ以外のプラットフォーム上で動く部品の両方を備えていなければいけない、というわけです。そのような条件下では、双方の部品の間での適切なコミュニケーションを図る必要があります。

GeneXusを利用してアプリケーションを開発した場合、アプリケーションを分割して、各部品をそれぞれ別のプラットフォーム上で実行させることができます。またそれぞれのプラットフォームに対応したプログラムを生成する際には、最も適した言語が選択されますので、利用可能な資源の使用効率を向上させる多階層アーキテクチャの利用が求められます。

プロトタイプ

設計作業では人間同士のコミュニケーションの難しさが、以下のような形を取って露呈します。

- ユーザが一部の詳細を失念する。
- アナリストがいくつかの要素を見落す。
- ユーザが誤ったアプローチを伝える。
- アナリストがユーザの説明を部分的に誤って解釈する。

また他方では、システムの実装作業には通常多大な時間を要するため、

- 上述の現象に起因する不具合が発覚するのはシステムテストの終了時点であり、解決に要するコスト（時間とお金）は莫大なものとなります。

- 仕事の現実は常に変化しており、システムの実装作業中は仕様を凍結できる、などと考えるのは合理的ではありません。

- 仕様を凍結してしまった場合、設計チームは結果的に比較的満足度の低いソリュ―ションを実現することになってしまいます。

もし仕様を一つ一つその場でテストすることができれば、不具合の衝撃は格段に抑えられるでしょう。

システムの中には不具合の発生を事前に食止めるため、ユーザにメニューを通して画面形式、レポート等を見られる機能を提供しているものもあります。ただこの方法では、ユーザにシステムがどのような画面やレポートから成り立っているかについて、多少ましなアイデアを与えるだけで、実際にシステムが完成した後で不具合が発生するのは避けられないでしょう。

機能の細部に至るまでユーザの希望を満たしたアプリケーションをユーザがテストし実際に動かしてみることができれば、状況は180度転換します。

GeneXusは完成形のアプリケーションと同等な機能を備えたアプリケーションを実行してみせることにより、いち早くこの課題を解決したのです。GeneXusのプロトタイプは最終的に作られるアプリケーションと機能的に同じ、かつ「すぐにでも動かすことができる」アプリケーションです。

プロトタイプと製品モデルの違いは通常、プロトタイプはPC向けに作られ、製品モデルはユーザが選ぶ環境上（iシリーズ、LAN、クライアント/サーバ、等）で動くように作られているということです。故に最終的な生産工程に入る前に、このような特徴を持つGeneXusのプロトタイプを使って、十分にテストすることができるのです。エンドユーザはテスト工程の間、本物のデータを扱うことができます。つまりテスト画面形式やレポート等に加えて、規定類やビジネスルール、データ構造等を使って実際の日常業務と同じ環境でテストをすることができるのです。

GeneXus哲学の中核を成すのは、**インクレメンタル開発**と称されるコンセプトです。従来型の作業環境に拠る開発においては誰もが当然のように、実装工程中にプロジェクト変更が生じた場合は、実装終了後に検知される深刻な変更も含めて、非常に高くついてしまう、と考えています。すべての開発はインクレメンタルかつ連続した方法で行う必要があるという認識に基づき、GeneXusはこの問題を克服してくれる方法論に依拠して、アプリケーションを開発します。具体的には設計者が変更の必要性を発見した場合、即座に当初から使っている既存のプロトタイプを変更し、「すぐにテスト可能な状態」でエンドユーザに提案し、追加費用の負担なしでテストしてもらいます。

実装

GeneXusは自動的に、以下の作業に必要なコードを生成します。

- データベース図を作成しメンテナンスする。
- ユーザが記述したオブジェクトを扱うプログラムを生成し、メンテナンスする。

生成工程は、**「仕様作成」**と**「生成」**という2つの段階に分けられます。「仕様作成」はターゲットとする稼働環境からは完全に独立していますが、生成はそうではありません。つまり、同一のモデルは生成時にターゲットとしていたプラットフォーム以外のプラットフォーム上でも稼働できますし、さらに生成したバージョンを稼働する環境に応じてそれぞれ最適化することができます。

現在サポートされている環境と言語は以下の通りです（表紙の日付参照）。

プラットフォーム

実行プラットフォーム
JAVA、Microsoft .NET、Microsoft .NET Compact Framework

オペレーティングシステム
IBM OS/400、LINUX、UNIX、Windows NT/2000/2003 Servers、Windows NT/2000/XP/CE/Vista

インターネット
JAVA、ASP.NET、Visual Basic (ASP)、C/SQL、HTML、WebServices

データベース管理システム

IBM DB2 UDB、INFORMIX、Microsoft SQL Server、MySQL、Oracle、PostgreSQL

言語

Java、.NET、C#、C/SQL、COBOL、RPG、Visual Basic、Embedded Visual Basic、Visual FoxPro

Webサーバ

Microsoft IIS、Apache、WebSphere、Tomcat他

複数アーキテクチャ

多階層アーキテクチャ、Web-based、クライアント/サーバ、集中型（iシリーズ）

対応している技術の一覧表は、下記サイトにてご参照下さい。
http://www.genexus.com/technologies

上述の対応プラットフォームに加えて、GeneXusは特定の機能を強化する一連のコンプレメンタリ・ツールを提供しています。

- ワークフロー … GXflow (www.gxflow.com)
- クエリー … GXquery (www.gxquery.com)
- データウェアハウス … GXplorer (www.gxplorer.com)
- ポータルサイト開発 … GXportal (www.gxportal.com)

メンテナンス

GeneXusが誇る最大の特質とも言えるものであり、並み居る競合他社のどんな製品よりも先んじている点です。それはデータベース(構造と内容)とプログラム双方の、完全自動化メンテナンスです。

以下、GeneXusのあるオブジェクトの記述が変化した場合(ユーザビュー)、どのようにメンテナンス処理が働くかを説明します。

データベース変更の影響

影響分析

データベース上に変更が生じた場合は、GeneXusはその影響を自動的に分析します。続いてデータ変換の方法を説明するレポートを作成し、さらに該当する場合には、変換から派生し得る一連の不具合(新ルールの付与に伴って古いデータが引き起こす矛盾など)についても説明します。データベースの更新をそのまま進めるか否かは、開発者が決定することができます。

変換プログラムの生成

起こり得る一連の不具合が解消され、もしくは変換を進めるという意思決定が認められると、引き続いて古いデータベースを新しいものに変換するプログラムが自動的に生成されます。この工程をリファクタリングと呼んでいます。

変換プログラムの実行

次のステップでは該当する実行環境(プロトタイプ、iシリーズ・プロダクション、クライアント/サーバプロダクション、等)に移動し、変換プログラムを実行します。

プログラム変更の影響

影響分析

プログラム上で変更が発生した場合、GeneXusはその影響について分析を行います。さらにどのプログラムを生成あるいは生成し直す必要があるかについて、診断レポートを発行します。同時に新規プログラム一つ一つにつきデータベース・ナビゲーション図を作成しますが、ここでは2つの詳細レベルから選択できます。

新しいプログラムの生成

GeneXusは必要な各種プログラムを自動的に生成します。

ドキュメンテーション

アナリストが提供した情報あるいはGeneXusが推論した情報はすべて、アクティブなリポジトリー内に格納され、オンライン上の完成したドキュメントとして常に最新の状態に維持されます。

ドキュメンテーションは特定のオブジェクトの記述、完成した知識ベースおよび設計されたデータベースに関する情報を含んでいます。

GeneXusの知識ベースは、常時ユーザの希望に応じて、ベース内に蓄えている知識へのアクセスに加えて、格納しているあらゆる論理的な推論情報（参照整合性規則、データベース・ナビゲーション図、変更の影響分析、クロスリファレンス、格納された知識から推論されたE-Rダイヤグラム）へのアクセスも受け入れています。

複数のアプリケーションと知識再利用性の統合

GeneXusを使えば、複数の開発チームが同時に複数のアプリケーションを設計し、プロトタイプすることが可能です。各チームは**ナレッジ・マネジャー**と呼ばれるモジュールを介して、設計仕様を相互に交換できます。

このモジュールは自動的に以下の作業を実施します。

- ビジネス・オブジェクト、ソフトウェア・パターン、ドメイン、属性、および/またはパブリックドメインのスタイル（http://www.gxopen.com.uy参照）に基づいて、新しいアプリケーションの設計を開始する。
- コーポレート知識ベースから他のアプリケーション用の知識ベースへと知識を分配する。
- アプリケーションの知識ベースとコーポレート知識ベースとの間の一貫性を確認する。
- 2つのアプリケーションを統合する（特定のアプリケーションからコーポレート知識ベースへ知識を統合するのは、特に有益です）。

理想的な柔軟性がここにはあります。つまりアナリストはプロトタイピング環境において、小規模な知識ベースと対話しながら完全に自由に仕事をしています。開発中のアプリケーションがユーザの視点から見て使える状態になった時、ようやくアナリストはその小規模な知識ベースを大規模なコーポレート知識ベースへと統合することを検討し始めるのです。

この二つの知識ベースを統合する決定が下された時、強力な自動化ツールが利用可能となります。新規もしくは修正アプリケーションがコーポレートモデルに与える影響が分析され、一貫性を保証するために適当と認められれば、数々の変更が行われます。

このスキームを利用すれば、例えば第三者からライセンスを取得した知識ベースを再利用することができます。

前提として、統合の対象となる複数の知識ベース間で共通の用語体系を使用することが必要になります。しかし新機能である「Adapt From」を利用すれば、名前変換のマッピングを定義することによって、ターゲットの知識ベースに順応させることができます。

もう一点注目すべきことは、GeneXusの知識ベースを開発したソフトハウスは今や、その知識ベースを部品として販売するために、開発したソースコードを他者に開示することなく、独自開発部分をメンテナンスすることが可能になったのです。

これにプラスして、現在ではそれぞれのオブジェクトを定義する際に、公開するものかプライベートなものかを指定することができます。GeneXusではすべて自動的に使用可となりますが、プライベートなオブジェクトに関しては、オーナーのみがGeneXusのハイレベルなソースを参照し変更を加えることができます。

GeneXus 独自の特長

GeneXusは競合品とは明らかに異なる特性をいくつか備えており、それが差別化につながっています。以下に挙げるのは、代表的なごく数例に過ぎません。

- エンドユーザが日々対話するオブジェクト（TRANSACTIONS、REPORTS等）に関する十分な知識を提供してくれた後に、設計は始まります。その理由はエンドユーザこそが、日常業務がどのように機能しているか、またすべきかを把握している人達だからです。

- 各オブジェクトの記述は、他のオブジェクトの記述とは完全に独立しています。このため、たとえあるオブジェクトが修正されることになっても、他のオブジェクトに手作業で修正を加える必要はありません。これはGeneXusにしかない機能であり、こうしてアプリケーションの完全自動メンテナンスが可能になったのです。

- GeneXusの学習カーブは短いものです。

- データベースは自動的に設計、生成、メンテナンスされます。

- アプリケーション（データベースと各種プログラム）は変更が施されたか否かにかかわらず、最高の品質を保っています。

    - データベースは常に最適化された状態になっています。

    - プログラムは修正されません。現状にそぐわなくなった場合は、新しいプログラムを生成して取り替えます。

- GeneXusのファイルや外部データベースを簡単かつ効率的に利用できます。

- プロシージャ、ワークパネル、およびウェブオブジェクトを定義する目的で、強力かつ極めて高水準の諸言語が用意されています。オブジェクト・プロセスが生成される際にファイルが自動的に推論されるので、これらのオブジェクトはどれ一つとして、特定のファイル・リフェレンスを必要としません。それ故、オブジェクトが定義された時点で、すべてのデータが完全に独立を達成できるのです。その結果、データベース構造に修正が加えられた場合でも、GeneXusの高レベルのオブジェクト仕様には一切変更を加える必要はありません。

- メンテナンスの完全自動化…GeneXusは各種プログラムを100%自動的に生成、メンテナンスします。この利点は経済のあらゆるセクター（商業、管理、財務、工業等）で貢献できます。

- GeneXusはPC上で稼動しますから、実コンピュータ環境に設計時の負荷を与えないで済みます。
- コーポレート知識を分散し易くすることで、アプリケーション開発が容易になります。
- 簡素で強力なレポート・ソリューション及びデータウェアハウシング・ソリューション。
- 別々のモジュールとして開発されたアプリケーション間での、一貫性の自動検証と統合化。
- プラットフォームとアーキテクチャからの独立。
- 平易化された設計…GeneXusは人工知能の最先端リソースを採用しているため、システムアナリストとユーザにとっては使いやすいものとなっています。

世界に拡がる GeneXus ユーザ

世界各地の5,000社以上のお客さまがGeneXusの製品を利用して、ミッションクリティカルなアプリケーションを生成、統合し、容赦なく押し寄せるビジネス上の変化に余裕を持って対応しています。GeneXusの技術を採用することで、私たちのお客さまは主要な技術プラットフォーム上で、各社固有のノウハウを駆使することができるのです。

私たちの法人のお客さまは幅広い分野にわたる中、大規模な企業に及んでおり、今日では当社の収益の70%に達しています。

私たちのISV（独立系ソフトウェア会社）のお客さまは、実際にGeneXusを使って製品やソリューション開発を行っている、規模の様々なソフトウェアハウスです。この部門は収益の30%を占めていますが、現在急速な成長を遂げつつあります。

（訳責：大脇文雄，　番田康子）