平成１２年度調査報告書
ＮＥＤＯ-ＩＴ-００１６

「産業技術戦略策定基盤調査
＜高度知的情報技術基盤(Hyper-Intellectual-IT Infrastructure)調査＞」

平成１３年３月

新エネルギー・産業技術総合開発機構
（株）日本総合研究所

『産業技術戦略策定基盤調査＜高度知的情報技術基盤(Hyper-Intellectual-IT Infrastructure)調査＞』

（株）日本総合研究所

平成１３年３月　　３２７ページ

ブロードバンドネットワークの普及にともない効率的な設計・製造・研究開発を行う、仮想的な知的協調が必要となってきている。そのようなインフラストラクチャーにあたる環境（Hyper IT）がどのようなものであるべきかを調査することを目的とした。

平成１２年度調査報告書

NEDO-IT-0016

「産業技術戦略策定基盤調査
＜高度知的情報技術基盤(Hyper-Intellectual-IT Infrastructure)調査＞」

平成１３年３月

新エネルギー・産業技術総合開発機構
委託先　(株) 日本総合研究所

要約

(Summary)

要約

インターネットの急速な普及とネットワークのブロードバンド化に伴い、ネットワーク上に分散した多くのシミュレータやデータベースが協調して、設計・製造・研究を行う時代が目の前に迫ってきている。

しかし、このようなネットワークを介した仮想的な協調型のシミュレーションやデータベース利用については多くの未解決の課題が存在し、それらが解決されなければ、ネットワークを介した仮想的な協調型研究・設計・製造などもままならない。

そこで、本調査では以下の項目について調査した。

（1）コンピュータ利用法の現状調査

（2）データの表現法現状調査

（3）ネットワーク分散環境現状調査

（4）シミュレータの現状調査

（5）ソフトウエア計算サービスの現状調査

さらに、上記調査をもとに、仮想的な協調型研究・設計・製造環境の持つべき機能と現状での問題点を洗い出し、以下の点に関して提言をまとめた。

（1）Hyper-IT イメージの明確な定義と意義

（2）Hyper-IT イメージと現状の比較

（3）課題の洗い出し

（4）課題の重要度・難易度の考察

（5）国の研究促進政策とロードマップ提案

Summary

In recently years, we have broad band network and The Internet environment , using these infrastructure, we will develop knowledge co-operate manufacturing support system, which can be use various kinds of simulators and databases.

But knowledge co-operate manufacturing support system has a lot of problems, such as data format, legal problems, software support system and so no.

So, we investigate following items,

(1) Computer using Status on research, development, manufacturing fields.

(2) Present status of Data expression i.e XML,XSIL

(3) Present status of Network distributed softwares

(4) Present status of scientific calculation softwares

(5) Present status of software commercial services

Based upon above research works, we discuss about fundamental functions of knowledge co-operate manufacturing support system and present problems of knowledge co-operate manufacturing support system.

our discuss items as follows.

(1) Clear definition of Hyper-IT and its significance

(2) Comparison with Hyper-IT & present status

(3) Dig up present problems

(4) Classification of present problems

(5) suggestion for Government prolicy

産業技術戦略策定基盤調査＜高度知的情報技術基盤(Hyper-Intellectual-IT Infrastructure)調査＞

目次　　ページ

１．はじめに

インターネットの急速な普及に伴って、企業でのネットワークを利用した事務処理はごく一般的なものになりつつあり、それと呼応するように技術系業務分野（設計、製造、研究開発）でも、これまでのような CAD/CAM/CAE の相互に関連のあまりない利用形態から、CAD/CAM/CAE の統合利用環境を構築し、より迅速でミスの少ない設計・製造・研究開発プロセスの作成に躍起になっている。

さらに、最近のネットワークのブロードバンド化（高速・大容量化）および製品開発拠点の海外・国内分散に伴い、遠隔地に分散した設計・製造・研究開発部隊が仮想的に協調し、１製品を作り上げる必要が出てきている。

最近の市場競争および価格・サービス競争の激化に伴い、そう遠くない将来、現在の１企業内連携から企業間連携さらには業界間連携へと発展していくことは自明のことである。

このようなすべての業界を横断した仮想的な知的協調設計・製造・研究開発環境はどのようなもので、どのような条件を整えるべきなのであろうかと言うことを本調査で明らかにした。

すべての業界を横断した仮想的な知的協調設計・製造・研究開発環境とは非常に高速に、簡易に誰もが垣根を越えて、技術情報をやり取りできる環境であるので、これは現在の電気・ガス・水道が生活を支えるインフラという意味で、ライフラインと呼ばれているのからして、「デザインインフォメーションライフライン」とでも呼ぶべきものであるし、さらに物理的な製品製造・流通・販売を支えている高速道路や鉄道などのインフラストラクチャと比較すれば、「設計情報のインフラストラクチャ」とも呼ぶべきものとなることは容易に推測でき、これらの整備なくしてはより効率的で競争力のある未来の日本の製造業はありえないことも明白である。

上記のすべての業界を横断した仮想的な知的協調設計・製造・研究開発環境あるいは「デザインインフォメーションライフライン」、「設計情報のインフラストラクチャ」を総称して、本調査では高度知的情報技術基盤(Hyper- Intellectual- IT Infrastructure)（以後、略して Hyper-IT と呼ぶ）と呼ぶことにする。

本報告書では Hyper-IT の明確なイメージを得るために、多くの技術側面での現状調査を行い、さらにそれらをベースに、あるべき Hyer-IT とその実現へ向けた提言を行った。

２．コンピュータ利用法の現状調査

Hyper-IT で実現できることやしなければならないことを明らかにするために、２章では現時点での情報技術の最新動向特に、設計・製造・研究開発過程での利用状況およびその近未来像を把握するために、インターネット情報や国内大学での研究状況さらには各種業界での取り組みなどに調査した。(詳細は付録１コンピュータ利用法の現状参照)

なお、以下に記載する文書には多くの情報関連語句が含まれており、それらを本文中で説明していくと、論旨がわかりづらくなる可能性があるため、一般的でない語句やその定義内容を明確にしたほうが良い語句などを付録５：用語解説で別に解説しておいたので、適宜参照されたい。

２．１　最新設計・製造・研究開発向け IT 技術

２．１．１　設計・製造・研究開発過程での文書管理 (PDM)

PDM(Product Data Management)とは「製品 データ 管理」を意味する概念で、製造業における、「企画→設計→製造→アフタサービス」といった、「製品ライフサイクル」を通した、あらゆるﾃﾞｰﾀの一元管理を目指したコンピュータデータ管理システムであった。

しかし、設計を中心に生産、資材調達などにかかわる機能をサポートする電子システムであった PDM はこれまでのようなデータの管理から、それらを如何に視覚的に表現するかという所に重点を置きつつあり、日本では世界的な部品調達のためだけに PDM を導入している状況であることと考え合わせると USA と日本ではかなりの温度差ができつつある。

最近富に激しさを増す企業間の競争により、特に自動車業界では新車計画をすべてコンピュータ上でシミュレーションしようという機運が高まりつつあり、デジタルモックアップに対する野心的な試みがなされつつある。

そして、その場合最も重要な要素として各部品間の関連データと部品のデータを用いて設計変更時に関連する部品の形状・物性等の変化を如何に視覚的に表現するかという事があげられる。

そこで、最近の PDM では以下の過程を視覚的に表現するツール群が整備されつつある。

・プロセスとデータの関連図

・製品構成図

・図面、関連文書構成図など

また、これらを支えるデータベースはオブジェクト指向型（オブジェクト指向データベースおよびリレショナルデータベースにオブジェクトを埋め込んだものを含むものをいう）となっている。

(1)　SDRC メタフェーズの現状

現時時点でのSDRC メタフェーズ関連情報を付録１：（１）PDMメタフェーズの現状に示すとともに、現在最も多く使用されいているCAE支援ツール「I-DEAS」との関連情報を以下に示す。

米SDRCが1999年10月28日に発表した「I-DEAS Master Series」次期版の製品計画の概要は以下のとおりである。

2000 年の製品開発におけるキーワードは二つある。 一つはコラボレーション環境の実現、もう一つは自動車の製品開発に向けた機能強化である。 コラボレーションでは、3 次元の設計情報を配布する機能の強化、 PDM ツール「Metaphase」との統合環境の実現を予定している。 設計情報の配布については(1)寸法線の整理機能の追加や日本語を含む英語以外の言語への対応など3次元注記機能の強化(2)3次元モデルからの図面化作業の効率化などを予定している。3次元の注記情報は主要な3次元ビューワから取り出せるように API（アプリケーション・プログラミング・インタフェース）を公開していく。 Metaphaseとは、インタフェースを介さずにチェックインやチェックアウトができるなど、I-DEAS上で製品情報の作成と管理を可能にする仕組みを提供する。

自動車向けには形状作成機能の強化と複雑な製造プロセスへの対応に重点を置く。点群からのサーフェス作成、複雑な形状についてのシェル化、コーナー作成といったジオメトリの作成機能やアセンブリ機能の強化を予定している。「VGX」（拡張バリエーショナル機能）を利用して金型のキャビティとコアを効率よく生成したり、CAM 機能を強化して複数の加工対象を治具に配置したようなアセンブリモデルにカッタパスを生成する、など複雑なプロセスへの対応が進む。

解析機能では、パフォーマンスの向上や樹脂流動解析などで利用する中立面の作成機能の追加などを予定している。

２．１．２　動的な計算機能を持ったホームページ（Web 技術の現状）

ここではまず初めに、Web 技術の現状について概観し、それらの特徴について言及する。

最近のホームページ（Web 技術）を用いると、利用者のアクション（データ入力など）に応じて、動的に計算するなどの機能を持っており、最近ではそれらは以下に記述した Java ベースのアプリケーションサーバ上で、稼動しているものが多い。

（１）サーバ拡張 API（Application Program Interface）

①CGI（Common Gateway Interface）

CGI（Common Gateway Interface）は文書の表示機能のみであった Web サーバの機能を拡張するものとして最も古くから存在するインターフェースである。

その機構は以下のとおりである。

(i) Web サーバはまず、ブラウザから送られた HTTP のリクエストに記述された URL に、CGI プログラムが指定されているかをチェックする。

(ii) URL に CGI プログラムの指定されていると、Web サーバは自分自身とは別のプロセスとして指定されたプログラムの起動を行う。その際に、ブラウザから URL の一部として渡された変数や、POST メソッドによって送信されたデータを、環境変数や標準入出力からプログラムに引き渡す。プログラムは、これらの情報を基に処理を行い、標準出力に処理結果を出力する。その後、Web サーバは出力データをブラウザに送信する。

・拡張プログラムがWebサーバの一部として動作するため、拡張プログラムが異常終了すればWebサーバも異常終了する可能性があるため、Webサーバの安定性が拡張プログラムの質に依存してしまう。
・APIを使用できる言語が限定されている。
・サーバ拡張APIを使用する場合は利用者の要求が個々のスレッドになるため、スレッドの独立性を考えたプログラムを書く必要がある。

（2） Java環境

オブジェクト指向言語であり、ネットワークとの相性もよいJavaはインターネットを使った社外に広がるシステム構築に使用するのは効果的であり、以下の特徴を持っている。

(i) クライアント・プログラムの配布が不要である。

クライアントにWWWブラウザがあれば、インターネットを使ってシステムを外部に提供できる。

(ii) マルチプラットフォームが実現可能である。

正確にはJava環境は現在次の2つのタイプに大別され、以下のようなマルチプラットフォーム特性を持っている。

・Client Side Java

Javaのプログラムは JavaVM上で稼動するため、Java VMを搭載したマシンであれば理論的にはどれでも動くはずであるが、実際にはJava VMごとに動きが違うため、特定のJava VM上で動作しない場合がある。

Javaプログラムはアプレットとして、ローカルマシンにダウンロードし、利用せねばならないため、起動までに時間がかかるのが一般的で複雑で大規模なプログラムとすることは避けるべきである。

・Sever Side Java

サーバ側で起動する Java プログラムでローカルマシンにはその結果作成されたページ情報を送付するため、かなりの部分までマルチプラットフォームが実現できている。

(iii) オープンな言語で将来普及する可能性が高い。

厳密には、Javaはオープンではなく仕様を決めているのは米Sun Microsystemsを中

心としたベンダーである。しかしその仕様は広く公開されており、利用者の要望にもかなり迅速に対応しており、多くの利用者はJavaをオープンな言語だと感じている。

(iv) ガベージコレクションなど開発者を支援する機能が充実している。

Javaはプログラムのメモリ管理をJava VMで行うガベージコレクション機能を持っており、アプリケーション・プログラマが記述したプログラムが、メモリー・リークやメモリーの不正使用を起こす可能性はない。しかし、メモリー管理を行っている Java VMが、メモリーの問題を起こす可能性がある。

① Javaアプレット

Javaアプレットは Client Side Javaに分類され、利用者がブラウザを使ってWebサーバにアクセスすると、WebサーバからHTMLファイルと同時にJavaアプレットもダウンロードされ、ブラウザの中で実行される。

Javaアプレットはデータベースが Webサーバと同じサーバにある場合は直接アクセスできるが、データベースがWebサーバにない場合は、Webサーバに「中継プログラム」を置く必要がある。すなわち、アプレットはダウンロード元のサーバと通信をするのみであり、中継プログラムなしにWebサーバ以外のデータベースにはアクセスできない。

・JDK と JavaVM とのバージョンの組み合わせにより、動作しない場合がある。

② サーブレット・JSP (Java Servlet Pages)

サーブレットとは、Web サーバからのリクエストを処理するための拡張プログラムを Java で記述することができるように、Java のクラス・インターフェース「サーブレット API」を利用して開発されたサーバ拡張 Java プログラムである。

サーブレットではプログラムの中に処理ロジックと、HTML ファイルを作成するための記述をする必要があるが、JSP は Web ページを作成する HTML の中に JSP 固有のタグを埋め込んで利用するもので、Java Beans を呼び出したり、サーブレットから値を受け取ったりできる。

サーブレットと JSP は Server side Java に分類され、サーバで処理を行うためクライアントの負荷が少ないという特徴を持つ。

サーブレットは Java プログラムであり、ロジックを記述することが容易である。一方 JSP は HTML ファイルを拡張して作れられているので、HTML ファイルを作り出す部分に利用するのに適しているが、複雑なロジックを記述することには向かない。

サーバ側アプリケーションを作る際には、このサーブレットと JSP を連携させることでメンテナンス性の高いシステムを作ることができる。

長所

・スクリプトや拡張タグを利用した開発スタイルが理解しやすい。

・ソースをコンパイルする必要がない身軽な実行環境を持つ。

・データベースへのログイン、ログアウトのオーバーヘッドを抑えるデータベースコネクションプーリング機能を持つ。

・HTTP セッションレベルでの負荷分散機能を持つ。

短所

・システムの拡張性や信頼性を確保することが難しい。

・標準的な技術よりもベンダー独自技術の比率が高い。

・複雑な業務ロジックを実行するには向かない。

② Java サーバ側技術対応製品

EJB に代表される Java2 Platform, Enterprise Edition(J2EE)の仕様を全面的に取り込み、サーバサイドで Java を活用することが特徴である。

長所

・EJB の規約に従ってアプリケーションを開発することで、コンポーネント部品の再利用性や移植性を高められる。

・CORBA や DCOM などの分散オブジェクトとの連携が充実している。

・アプリケーションレベルでの負荷分散機能を持つ。

短所

・Java の高度な知識が必要となる。

3．データの表現法の現状調査

現在利用させているもしくわ近未来に利用させるであろうデータ交換法の現状について調査した結果を以下にまとめた。(詳細は付録2データ表現法の現状参照)

3．1　各材料データベースの現状

以下に示す各材料関連データベースの現状を訪問調査したので、以下にその概要を示した。

3．1．1　ファインセラミックスデータベース

（1）現状

付録「各材料データベースの現状」に示したように、構造材料・電磁気材料・材料組織・企業カタログデータから約1万点あまりのファインセラミックス関連材料データを収集しており、それらは以下のように大別されている。

・学術情報・データベース

・生データ・データベース

・カタログ・データベース

これらの内、学術情報・データベースはすでにインターネットでの利用を開始している。

材料分類方法はVAMAS分類を基にしたISOセラミックス分類規格（案）準拠したものとなっており、非常に整備されたものとなっている。

またCMC(セラミックス・マトリックス・セラミックス）のデータベースはCDROMによる配布が検討されている。

（2）問題点その他

上述のように精力的にデータ収集と整理は行われているものの以下の問題点が見受けられる。

・データ収集に６０名もの技術者が関与し、非常に膨大な経費が必要

・材料特性データは収集されているが、利用者の高度なニーズに応えられる検索システムやプログラムは未整備

・基本的な検索を複合的に行うため、Java Appletによる検索システムを構築したが、応答に時間がかかり不評である。

・生データの関数近似および異常データの検知などの高度利用や材料特性予測のため

の主成因分析などの初歩的な統計解析機能などがないと利用者が増えないと思われる。

・データの蓄積・システムの運用などの経費を利用者への課金によってまかなうことも考えられるが、国の知的基盤でもあり、国家での全額保証による無償サービスが望ましい。

・カタログデータについては企業の公開データに差があり、XML などの新しい情報技術を用いなければ、データが非常に冗長になると思われる。

・データの著作権などをどうするかについても問題がある。

・試験データなどはグラフを多用した視覚に訴えるものにする必要がある。

3．1．2　ニューガラスデータベース

（1）現状

付録「各材料データベースの現状」に示したように、国際ガラスデータベース(INTERGLAD)として、有償サービスを開始しており、形状・用途・出典・組成・ガラス汎用名・機械的特性・物理的特性・熱的特性・光学的特性・電気的特性・磁気的特性・化学的特性など広範囲に収録されたデータを Internet および CDROM によりサービスされ、それらは以下資料から収集されている。

・データブック

・学術論文・学会予稿集

・特許

・カタログ

98 年から 99 年の平均検索回数はインターネット上で 2000 件／月でこの種のデータベースとしては上々の成績であるといえる。

（2）問題点その他

・ガラスの透明度などは成分組成に依存しているがこれらを統計的に捉えるなどの高度利用を行わなければ、新材料の予測などの新しい利用法を模索しなければ、これ以上の利用が望めない。

・最近の環境アセスメントなどの流れに沿うために、LCA などの高度利用を模索する必要があり、現在の CGI ベースのシステムをアプリケーションサーバを用いた新システムに作り変える必要がある。

・ガラスの設計支援システムなどの高度利用システムを順次追加していく必要がある。

・試験データなどはグラフを多用した視覚に訴えるものにする必要がある。

3．1．3　アルミニウムデータベース

（1）現状

付録「各材料データベースの現状」に示したように、テスト的に伸銅協会と同様のシステムを用いて、組成・物理的特性・化学的特性・機械的性質・工学的特性・環境負荷特性などについてデータを保存し、インターネットでサービスするシステムを構築している。

（2）問題点その他

・システム全体がVisual basicとAccessで構築されており、このデータベースに高度な機能を付加するより、材料統合データベースに機能を付加し、このデータベースの負荷を低減するほうが良い。

・アルミニウムは現在建材として利用することが検討されているが、これを推し進めるには簡易な構造解析機能などを整備し、鉄鋼に見られるような型鋼表などから最適な鋼材を選択できるような機能を持つ必要がある。

・熱特性に優れたアルミニウムの特性を利用者により分りやすく示すためには、熱伝導解析の基本機能を用いた検索などを整備する必要がある。

3．1．4　伸銅データベース

（1）現状

付録「各材料データベースの現状」に示したように、以下の3種類のデータベースが構築されている。

・文献データベース（伸銅技術研究会誌全文収録）

・開発合金データベース

・会社情報データベース

この内材料データに関するものは開発合金データベースで、組成・物理的特性・機械的特性・用途・形状・製造会社アドレスなどが保存されている。

（2）問題点その他

・システム全体がVisual basicとAccessで構築されており、このデータベースに高度な機能を付加するより、材料統合データベースに機能を付加し、このデータベースの負荷を低減するほうが良い。

・基本物性値で、材料加工・成形過程に依存したものも存在するため、試験データなどを付加していきたいとの意向であり、またこの際には試験データなどはグラフを多用した視覚に訴えるものにする必要がある。

3．1．5　チタンデータベース

（1）現状

付録「各材料データベースの現状」に示したように、以下の8つに大別される特性を保存したデータベースを構築するためのデータ収集試験を行っている。

・基本特性（熱・電気・電磁・その他の特性）
・加工特性（切断性・溶接性など）
・環境負荷（LCA データ）
・化学特性（耐食性など）
・機械特性（ヤング率・クリープ強さなど）
・試験法・評価法
・規格。基準情報
・標準物質（試料）情報

（2）問題点その他

・データ収集模索段階であるので、材料統合データベースとの関連での問題は特に言及できないが、将来を見据えると他のデータベースと同様の問題が生じてくることは自明であるといえる。

3．1．6　複合材料データベース

（1）現状

付録「各材料データベースの現状」に示したように、繊維教化複合材と粒子分散強化複合材の2種類について各種試験情報データベースが構築されている。

格納されている試験情報は以下のとおりである。

a) 繊維強化複合材

・引張試験
・低サイクル疲労試験
・熱機械疲労試験

・クリープ試験

・端部効果評価試験

・残留応力測定試験

・線膨張係数測定試験

・比熱・熱伝導測定試験

・ＦＥＭ用データ取得試験

b) 粒子分散強化複合材

・引張試験

・硬さ試験

・疲労試験

・クリープ試験

・破壊靱性試験

・衝撃試験

・疲労亀裂伝播試験

・耐酸化性試験

・線膨張係数測定試験

・比熱・熱伝導測定試験

・弾性率測定試験

（２）問題点その他

・シーズ指向からニーズ指向への転換を課題に取り組まれた意欲的な研究であるが、さらに利用者の高度利用を促すため、構造解析や熱解析の基本機能を付加することが望ましい。

・個々の物性値のみでなく、その試験の全データを収録したデータベースで、データベースエンジンもフリーの PostGre を使用するなど野心的な試みが見られるが、試験から弾性率などの物性を求める部分などはどの材料にも共通であるので、材料統合データベースの機能とし、これから開発されるデータベースでも利用できるほうが望ましい。

３．１．７　ニューマテリアルデータベース

（１）現状

付録「各材料データベースの現状」に示したように、以下の３つに大別されるデータが

収集され、エクセルベースでCDROMにより供給される予定である。

・文献データ（超塑性材・形状記憶合金・金属触媒）

・試験データ

・試験法情報（高速変形・金属触媒・水素貯蔵）

（2）問題点その他

・データ収集模索段階であるので、材料統合データベースとの関連での問題は特に言及できないが、将来を見据えると他のデータベースと同様の問題が生じてくることは自明であるといえる。

3．1．8　現状調査総括

いずれのデータベースも材料の物性値を保存したものとなっており、それらを利用した検索を行う際に、材料の基本物性をかなり知っている事が前提となっており、一般利用者にとっては利用しづらいものとなっている。

各データベースの問題点で述べたように、より一般的な利用者のニーズに立った高度な検索機能を付加することが望まれるが、一方で著作権などの問題もあり、各協会でデータベースを個々に保有せざるを得ない面もあり、材料統合データベースは高度利用機能部に特化し、データベースは各協会のものを用いる分散型データベースシステムとすることが望ましい。

3．2　技術データの国際標準記述法策定への取り組み(STEP)

STEP (STandard for the Exchange of Product model data)は異なるCADシステム間で製品データを交換するための国際標準規で、IGESに代わるCADデータ交換用の標準フォーマットとして標準化が進められている。また、ソリッドモデルのデータ交換も考慮されている。(詳細は付録2データ表現法の現状参照)

ステップの特徴は、IGESが図形データの規定に限定されているのとは異なり、プロダクトモデルという概念を導入して、製品構成、材料データ、加工用のNC（数値制御）データなど、製品のライフサイクル全体にまたがる広範囲なデータを対象としている点である。これによって、製品の設計から生産プロセスまでの情報を共有化することができる。また、自動車、造船など産業ごとに製品データの種類や構造を記述したアプリケーションプロトコル（AP、データ交換フォーマット）が用意されるなど、多様な分野、

目的に合った標準化を目指している。ステップによるデータ交換機能を持つ CAD システムも登場しており、今後はIGES に代わってステップによるデータ交換が主流になるものと思われる。ただし、現状では CAD ベンダーの対応は統一されておらず、市販の CAD システムがどのような AP をサポートしていくかなど、データ交換の標準規格として共通に利用するための課題もある。

コンカレントエンジニアリングでは、アプリケーション間で共通に利用できるデータ形式の標準化が強く望まれており、製品モデル全体のデータ交換を対象としたステップの標準化は、今後の製造業の製品開発プロセスの枠組みを大きく変える可能性があるものとして注目されている。

3．3　科学技術計算データの構造記述言語その 1 (XSIL)

XSIL (Extensible Scientific Interchange Language)は現在 Web 上での商取引データ記述言語として、その標準化が薦められている XML (Extensible Markup Language) の科学技術計算データへの拡張であり、現在 Caltech Center for Advanced Computing Research, Projects and Collaborations (CACR).の中で、その仕様や機能が検討されており、Java3D を用いた Viewer など徐々にその全貌が明らかになってきている。(詳細は付録 2 データ表現法の現状参照)

3．4　科学技術計算データの構造記述言語その 2 (UDF)

UDF (User Definable Language)は「高機能材料設計プラットフォーム」(通称土井プロジェクト) の中で、その機能と文法が日々改良されており、各種解析プログラム間でのデータ交換用共通記述言語として期待されているもので、その特徴は UDF で記述されたデータはすべて同一のデータ編集とデータ View が可能な環境 (プラットフォーム) を OS に依存しないで得ることができるというものである。

さらに、プラットフォームでは思考過程を簡易スクリプト言語(Python)で記述し、保存しておくことが可能で、これにより研究者や設計者が何をどのように取り扱いたいかを明確な形で、蓄積していくことが可能になっている。(詳細は付録 2 データ表現法の現状参照)

4．ネットワーク分散環境の現状

ここでは HyperIT の最も重要な機能であるネットワーク分散環境下での解析エンジン（シミュレータ）・データベース・教育システムの協調についての現状調査の結果を以下にまとめる。（詳細は付録３：ネットワーク分散環境＆シミュレータの現状参照）

4．1　利用者端末利用の現状

HyperIT の最も重要な機能であるネットワーク分散環境下での解析エンジン（シミュレータ）・データベース・教育システムなどの利用法として、最新のソフトなどをすべて利用者側の PC などにダウンロードし、その環境下で利用するという方法が考えられる。

この場合利用者にとって、以下のメリット・デメリットがあると考えられる。

利用者端末上でのソフト等利用時のメリット・デメリット

・既存の一般販売ソフトと利用者側の操作などがほとんど変わることがない。

・購入ソフトではないので、常に最新のソフトを利用できる。

・データベースの情報なども常に最新のものが利用できる。

・利用時にはソフト等をダウンロードしなければならない。

一方で、ソフトサービス側から見たメリット・デメリット

ソフトサービス側から見たメリット・デメリット

・利用者へのソフトの配送などのコストを省ける。

・利用者は常に最新版のみを利用しており、古いソフトをサポートしなく良い。

・不正コピーされる危険性が高い。

・時間貸しや期間貸しなどの新たなサービスの可能性がある。

・新たな価格体系を構築しなければならない。

現在上記のメリット・デメリットをいろいろな形で、取り除こうとした野心的なソフトサービスが行われつつある。（詳細は付録３：ネットワーク分散環境＆シミュレータの現状参照）

5．シミュレータの現状

5．1　ネットワーク利用を前提としたシミュレータの登場

1章でも触れたように、ブロードバンド化されたネットワーク環境の中では、「付録4：Hyper-ITと同様の取り組み」に示したように、ネットワーク上に分散配置された各種解析エンジン（シミュレータ）がネットワーク上で利用されることを前提に整備されつつある。

また、進んだ商用サービスとして、最も有名な構造解析ソフト「MSC/NASTRAN」は米国内でのサービスではあるが、すでに昨年2月から、ネットワークを通じたソフトの時間貸しサービスをスタートさせている。

（詳細は付録3：ネットワーク分散環境＆シミュレータの現状参照）

5．2　複数シミュレータの協調解析環境の試み

5．2．1　名大　土井研究室の取り組み

名大　土井研究室で行われている「高機能材料プラットフォーム」プロジェクト（通称：土井プロジェクト）はメソスケールでの材料設計解析エンジンの開発とその協調環境の開発を目指して、プロジェクトが進められており、そこでは以下のような特徴を持つエンジン協調システムの開発が進められている。

名大　土井プロジェクトプラットフォームの特徴

（1）ＵＤＦ構文解析機能(UDF Grammar Analysis)

汎用性のある科学技術データ記述構文UDFの開発とその利用関数の開発

（2）I/Oインターフェース機能(Engine I/O Interface)

解析エンジンとの入出力関数群の開発と既存書式との入出力機能の開発

（3）データチェック機能(Data Check)

理工系データの整合性チェック関数の開発

（4）グラフィックス機能(Graphical Drawing)

汎用描画基本関数の開発

（5）データ抽出・変換機能(Data Select/Modify)

解析結果をいろいろな形で考察するための基本関数の開発

（6）データ作成支援機能(New data Making Assist)

初期入力データなどの作成支援関数の開発

（7）データベース機能(database Interface)

データベース利用関数の開発

（8）マルチプラットフォーム（ＯＳ）対応(Run on Multi-OS)

Java & C++を基本言語としたＯＳ非依存関数として開発

（9）簡易計算機能(Simple Calculation facility)

利用者の試行錯誤過程に対しては簡易スクリプト言語 python で対応

（１０）エンジン管理機能(Engine Management)

解析エンジンの計算途中の情報やエンジンの制御を行う関数の開発

５．２．１　米国MSC社のオンライン計算サービスの取り組み

付録３：ネットワーク分散環境＆シミュレータの現状に示したように、ブロードバンドが一般化している米国では、汎用構造解析ソフト MSC/NSTRAN の時間貸し計算サービスが実際に商用として、始められている。

このサービスでは利用者は専用のネットワークツールを用いて、利用者個人のＰＣから MSC/NASTRAN の入力データを自分専用の計算センターと呼ばれる領域に転送し、計算の実行はセンター背後に存在するスーパーコンピュータで行い、計算結果は自分専用の計算センターに格納され、ネットワークを介して、計算結果のグラッフィック表示がほぼリアルタイムに行える機構になっている。

これにより、利用者は高額でめったに利用しないソフトを購入する必要がなくなり、設計・研究時の固定経費の節約ができるとともに、自分専用センター領域に安全に自分のデータを保管できるというメリットがある。

勿論、こうしたサービスを行う上では、セキュリティ上の問題などで、データの流出などが生じた際の法的な問題などが日本では未整備で、米国 MSC も日本での上記サービスを行っていないのが現状である。

6．ソフトウエア計算サービスの現状

6．1　新しいソフトサービスの普及を阻害するの要因

１９９９年頃から、ASP (Application Service Provider)すなわち、サーバ上にしかないソフトウエアを利用者に時間貸しするという新しいソフトウエアのサービスが事務関連ソフト（給与計算、ERP (Enterprise Resource Planning)など）を中心に、急速に普及してきている。

しかし、このようなネットワーク中心のサービスは新しいサービスであるだけに、依然多くの問題が未解決で、それがこの新しいサービスの普及を拒んでいる要因でもある。そこで、ここでは「日経 BizIT:記者の眼　ASP の普及を阻害するこれだけの要因」に記載された各種の課題を以下にまとめた。

（1）自社システムや重要な情報を明け渡す不安

ASP の利点は，自社にシステム導入資金や技術スキルを持たなくても，エンタープライズ・アプリケーション（企業の発展、製品の製造・設計などの根幹のソフト）が利用できる点だ。しかし、アウトソーシングと同じように「自社のシステムや重要なデータを外部に委託しまって構わないのか」という心配は常について回る。また，ASP のコントロール下に置かれることにつながりかねないとの危惧もある。

（2）セキュリティの不安

自社の重要な情報をインターネット上でやり取りすることに企業は不安を覚える。比較的小規模な ERP 業務をアウトソーシングしているある自動車部品メーカの幹部は，「当社よりも低価格をつけるために，ライバル企業が部品番号や発注情報，入札価格を盗もうとするのではないか」と懸念する。このため同社はインターネットではなく専用線を利用している。

（3）ASP が期待するほどの能力を備えていないのではないかという不安

企業は ASP 事業者の能力を心配する。そこで重要になるのがサービス・レベル契約（SLA）だ。SLA には，システム連続運転に関する可用性やヘルプデスク対応，問題発生時の対応などの保証が含まれる。ところが日本では，この SLA 環境が整備されていない。アウトソーシングの例でも，外資系サービス企業は障害時の損害賠償金額を記入するのが通常だが，日本企業は「両社協議する」が大勢。力関係でユーザ企業が泣き寝入りするケースもある。ASP は中小・中堅企業向けに効果を発揮すると見られるサー

ビスだけに，SLA を明確にしなければならない。またソフトを常に最新に保つためのアップグレードの頻度や，使えるようになるまで（インプリメンテーション）の期間に関する方針も SLA に加えるべきだ。

(4) アプリケーションのカスタマイズが制限され，使い勝手が悪くなる不安

既存のアウトソーシングでは通常，アプリケーションのカスタマイズを行う業者のサービスが含まれている。しかし ASP では集中利用のコスト効果をあげるため，顧客に対するアプリケーションのカスタマイズは最低限に制限される。例えば日本企業は ERP 導入時に，使い勝手を考え ERP ソフト価格の平均 2～4 倍のカスタマイズ・コストを費やすと言われる。ASP 利用ではカスタマイズが制限されるので，ASP 導入企業はカスタマイズ（使い勝手）と，コスト，導入スピードを秤（はかり）にかけることになる。ある中小企業の社長に ASP 利用の可否を聞いたら「自社システムだとディーラが節税システムを入れてくれる。おおっぴらな ASP ではそうはいかない」と話す。

(5) インターネットの帯域幅とスピード

ASP が普及するためには，インターネットがもっと広帯域で信頼性が高く，コストが安くなる必要がある。わが国では，このインターネット回線そのものがネックとなって，ASP 普及を阻害する可能性が他国より大きい。

これらの障害や懸念は技術発展の側面や，初期の顧客に対する適用作業を通じた「ASP 習熟曲線」の中で学び，解決できるかもしれない。しかし，ERP やアウトソーシングが踊り場に来たと言われるように，今のような ASP の過大な約束は市場を後退させかねない。

6．2　ネットワーク上での学習サービス

米国では経済学修士号(MBA)を中心に、ネットワークを介した学習サービスが開始されており、全米の主要大学ではネットワークだけで単位を取得したり、学位を取得したりができるようになっている。日本でも放送大学などで、徐々にテレビなどのメディア利用が進んできているが、問題となるのは利用者の理解度を正確に把握できるかということにつきる。

すなわち、ネットワークの向こうにいる利用者がいかなる不正を行っているか？

（例えば、参考書をカンニングしたり、一度偽名で受験し、答えを不正に入手するなど）を把握しづらいという問題にどう対処するかである。

これについては、アクセス毎に、出題問題を変えるなどの取り組みが米国ではすでに行われており、TOFEL のように現地で、ＰＣによる出題しか行わないという資格試験も出てきており、日本でも情報処理技術者試験で似たような試みを予定している。

これにより、出題者は受験者の能力に応じたきめ細かな出題が可能で、さらに迅速で間違いのない採点が可能になり、受験者は自分の能力に応じたきめ細かな解答が可能となっていくものと思われる。

しかし、現状ではこれらはいずれも独自のシステムで運用されており、相互に互換性などがない状態で、これでは科学技術系の知識や知恵を評価するまでに、多くの無用な書式群が出来上がるのではないかと危惧される。

（詳細は http://www.ymd.com/forum/toefl/0009.htm など参照）

7．HyerIT イメージの明確な定義と意義

これまで述べてきたHyperITのイメージを図にすると、次ページのようになる。

7．1　HyperITの明確な定義

HyperITは次ページに示したように、以下の特性を持った仮想協調分散型の設計・製造・研究用インフラである。

（1）産業毎の設計・製造・研究ノウハウを横断的に学習できる機構である。

（2）多くの企業・大学・研究機関などが保有する技術計算用シミュレータが相互に連携して、シームレスにある目的の計算が可能な環境である。

（3）解析シミュレータの入出力書式を相互に自動的に変換する機能を有しており、利用者はそれらを意識せずに、利用可能である。

（4）企業などの保有しているシミュレータなどでは利用者から計算時間や利用した資源などに応じて、料金を徴収できる機構をもっている。

（5）基本となる物性などのデータについては登録させている世界中に広がるデータベースから解析対象に必要なデータとして検索･抽出できる機能を有している。

（6）物理量の物理的な意味付けなどを持っており、業界ごとの語句の違いなどを自動的に判断し、翻訳する機能を有している。

（7）データは企業や個人の秘密事項である場合が多いので、HyperIT上を流れるデータはハッキングなどに対して十分に安全であることが保証されるべきである。

（8）解析に用いるデータは人の知恵によってモデル化などを介して、作り出す必要があるが、その際の知識の学習のためのネットワーク上のオンライン学習機能を有しており、利用者の知識やスキルレベルに応じて、学習後解析ソフトが利用できるようになっている。

7．2　HyperITの意義

前節で述べたような機能を持ったHyperITインフラが整備されると、企業は新しい製品の概念やモデルを発想することに専念でき、これまでのように同じようなソフトを各企業ごとで開発するという無駄な投資を極力抑えることができ。

また、HyperITに備わった教育機構により、社員研修を行うことが可能で、熟練技術者の時間を裂くなどの無駄も省かれることになる。

8．HyperIT イメージと現状の比較

8．1　HyperIT と同様の取り組み

7章示したようなネットワーク上での解析ソフト・データベースなどが協調して、何かを行うという HyperIT と同様の取り組みは現在どのようなものが存在しているのか？また、それらはどのようなものであるのか？について以下にまとめた。(詳細については付録4：Hyper-IT と同様の取り組み参照)

8．1．1　ITBL

付録4：Hyper-IT と同様の取り組みに示したように、日本原子力研究所では ITBL(IT Based Lab.)という高速大容量ネットワーク上仮想解析システムの構築に着手しており、ここでは大型のスーパーコンピュータ上に配置された各種大規模解析ソフトがネットワークを介して、協調して解析が行えるよう研究が進められている。

本報告で調査しようとしている HyperIT はこの構想をさらに、個人のＰＣや企業の小規模な計算機などにも広げたときを想定したもので、ITBL は HyperIT の一部をなすものと言える。

8．1．2　e-Science

ヨーロッパや英国では ITBL と同様の GRID という大規模分散解析サービスシステムが計画されており、そこではさらに進んで、HyperIT と同様に、委員会形式で知識の共有化などの多くの議論されることになっている。

(詳細については付録4：Hyper-IT と同様の取り組み参照)

8．1．3　学術振興会の取り組み

付録4：Hyper-IT と同様の取り組みに示したように、未来開拓学術研究推進事業を平成8年度から行っており、そこでは「知能情報・高度情報処理」、「シンセシスの科学」、「計算科学」、「知的で動的なインターネットワーキング」など多くの HyperIT で利用可能な基礎研究が進められている。

本報告書で特に取り上げた東京大学　米澤研究室の AMO や JavaGO もこの事業の一部である。

8．1．4　造船業界の取り組み

造船業界では昔から SR という産業（造船大手7社：三菱重工、石川島播磨重工、日

本鋼管、川崎重工、住友重機械工業、日立造船、三井造船)、大学（造船５大学、東京大学、大阪大学、横浜国立大学、広島大学、九州大学）と国土交通省（旧運輸省）などが共同で進める勉強会を行っており、そこで厚板の溶接時の施工精度向上を目指した取り組みが進められている。これは HyperIT における熟練工の知恵の知識化に対応し、これも HyperIT の一部をなす試みと言える。

（詳細情報については機密扱いで不明な点が多い。）

８．１．５　航空機業界の取り組み

川崎重工、石川島播磨重工を中心に、航空機のリペット打ち、アルミ溶接なのでの熟練工の知恵の集積と CAD データと図面の連携などによるデジタル仮想組み立てなどの取り組みがなされつつある。(詳細情報については機密扱いで不明な点が多い。)

８．１．６　デジタルマイスタプロジェクト

経済産業省（旧通産省）素形材産業局では産業界の効率化と再生を目指して、自動車関連プレスや樹脂成形の分野での施工時の知恵の情報化をデジタルマイスタプロジェクトとして、立ち上げることを計画中でこれは上の２件と同様に、HyperIT における熟練工の知恵の知識化に対応し、これも HyperIT の一部をなす試みと言える。

9．課題の洗い出し

9．1　ネットワーク環境での協調利用という点での一般的な課題

6章の ASP サービスで述べたように、一般的なネットワーク上でのソフトウエアサービスには以下のような問題点が存在している。

・自社システムや重要な情報を明け渡す不安

・セキュリティの不安

・ASP が期待するほどの能力を備えていないのではないかという不安

・アプリケーションのカスタマイズが制限され，使い勝手が悪くなる不安

・インターネットの帯域幅とスピード

これらに対して、4章で取り上げたＡＭＯを用いたより高度・高信頼性のある新しい分散環境である程度の対策が可能であると考えている。すなわち、

・自社システムや重要な情報を明け渡す不安

ー＞電子認証を用いたネットワーク内のＡＭＯシステムの中でしかも社外に出したくないデータの処理は社内ＡＭＯで処理することができる。

何らかの意味で統合や比較をするために、サーバや社外環境に処理結果やその一部を出す場合にのみ、そのデータが社外ＡＭＯによって参照されるように全体の環境を構築することが可能であるからだ。

・セキュリティの不安

ー＞ＡＭＯには認証機構を組み込むことが可能で、しかも上記のように必要最小限のデータのみがやりとりされるため、安全性は格段に高くなっており、どのデータを社外とやり取りするかというルールが可変であれば、多くのデータのやり取りでのセキュリティ不安のほとんどが取り除かれうると考えている。

・ASP が期待するほどの能力を備えていないのではないかという不安

ー＞これまでのＡＳＰがすべてセンター中心処理であり、クライアントはデータをすべて預けて、処理サービスを依頼するだけだったのに対して、ＡＭＯシステムではデータとその処理を一体にしたオブジェクトであるＡＭＯが高度な情報処理を行う機構である。よって、クライアントはより高度な処理をＡＭＯとして組み込むことが可能で、処理上の能力問題を解決する大きな可能性も持っていると考えている。

・アプリケーションのカスタマイズが制限され，使い勝手が悪くなる不安

ー＞ＡＭＯはオブジェクト指向技術を用いたマルチプラットフォーム設計なので以前のシステムに比べると、カスタマイズはやりやすいものである。

・インターネットの帯域幅とスピード

ー＞これはインフラの整備の問題であり、最近の状況を見ると、ネットワークのブロードバンド化は非常に急速に進むと考えられ、この問題は主要な問題とはなりえないと考えられる。

９．２　データ交換上の一般的な課題

３章で示したように、これまでにも科学技術計算データの書式の共通化という取り組みはSTEP,CALSと言った研究でなされてきたが、それらはいずれも構文（文法）だけでなく、書式を規定しようとして、挫折に終わっている。

そこで、最近では簡単なプログラムとの連携機構を持つXML構文の一部として、科学技術計算用共通構文を規定しようと言うXSILや解析シミュレータの行き来を念頭においた名大土井プロジェクトプラットフォームなどの試みがなされえているが、これらをある程度規格として、オーソライズしていかなければ、微妙に相違する種々の規格によって、現在のＣＡＤデータのように、変換が不可能となる可能性がある。

９．３　商用サービスの一般的な課題

商用解析計算サービスでは、データの保護と課金の確実性など、いわゆる事務系ＥＣサービスで問題となっている課題が技術計算系でも同様に、重要な課題となる。

９．４　国の施策上の課題

国の施策は付録４：Hyper-IT と同様の取り組みで述べたように、国の各省庁で同様の取り組みを重複して、行っている面が見受けられ、これらは非効率であるばかりでなく、多くの別の規格のために、共倒れする可能性をも秘めており、早急にこれらを整理統合することが望ましい。(今回は調査できなかったが総務省（旧郵政省）でもＩＴ関連政策で、HyperITに似たプロジェクトを模索しているようである。)

１０．課題の重要度・難易度の考察

９章で述べた各種課題の重要度・難易度解決法などについて考察する。

これまでに述べてきた課題の重要度・難易度を以下に示す。

最重要課題

・データ交換共通プラットフォームとその機能の洗い出し

・課金・セキュリティ機構の洗い出し

・商用サービス時の知的所有権とその保護の問題

重要課題

・国の施策の統一・統合

・基本ハード環境の整備とソフト構築指針の策定

・ソフトやＯＳおよびデータ構造についての広い意見聴取

これらのために、情報処理企業・学会、製造業、大学、官庁などから広く意見を聞く委員会の設置が急務であると言える。

１１．国の研究促進政策とロードマップ提案

１０章で示した各種課題の重要度・難易度解決法などについて考察結果から、今後国として展開すべき施策とそのロードマップについて考察し、国への提言をまとめる。

施策のロードマップ

平成１３年度

「科学技術データ標準化検討委員会」の開催

・各種ＸＭＬ標準化委員会とのすり合わせ

・データ交換法のすり合わせ

・シミュレータやデータベースのインターフェースのすり合わせ

などを行う、「科学技術データ標準化検討委員会」を開催する。

この委員会では上記のほかに、米国・ヨーロッパの諸事情についても調査を行い、国際間でも問題の少ない形での規格の策定を目指すべきである。

平成１４年度

１３年度の標準データ構文などの提言を受けて、インフラとしての

「HyperIT プラットフォーム検討委員会」の開催

・学習機構や翻訳機構の検討

・基本プログラムインターフェースの検討

・ベースとなるハードウエア構成とネットワークの構成検討

などを行い、平成１４年度終了時に、国の基本指針として、広く各企業や世界に指し示す。

平成１５年度以降は

平成１３年、１４年の検討をベースに個々の問題の基本計画や基本指針を作成していく。

なお、HyperIT は実証インフラとして、５年度には稼動していることがのぞましい。

付録１　コンピュータ利用法の現状

収録資料

付録１．１　PDM～ＣＰＣ

Web Site e-ものづくり　より

付録１．２　ＰＤＭソリューション

製造業における製品データ管理ソリューション

（日本総研　プレゼンテーション資料）

付録１．１　PDM～ＣＰＣ

Web Site e-ものづくり　より

CPCは，従来のアプリケーション・アーキテクチャをWeb技術と融合させることだけで実現できるわけではない。CPCモデルで最も重要なコンセプトは，標準的で広範に利用可能なWeb技術を利用し，絶え間なく変化するビジネス・ニーズに応じて，迅速に変更できる敏捷性の高いアーキテクチャを構築することである。

図3】CPCと従来のPDMアーキテクチャの比較　出典：ガートナー

	従来のPDM C/Sモデル	CPCモデル
クライアント・インターフェース	独自仕様およびWeb（HTML）	Web（HTML, DHTML, java）
ツール	独自仕様	多数のjavaおよびWebツール
アプリケーション統合	独自仕様API	EAIアダプタ
データ・モデリング	厳格なデータ・モデルが必要	データ・モデリングはわずか、またはなし
データ統合	独自仕様API	XML
データの所有権	単一のモノリシック・アーキテクチャ内	複数のフェデレーション（連合体）にまたがる
データのリンク	独自仕様のデータ・モデリングによる	ハイパーリンクによる
インストール期間	月単位	週単位
アプリケーション開発時間	月単位	日単位
アプリケーションの主な用途	企業内の製品開発	コラボレーティブなプロダクト・ライフサイクル・アプリケーション

このアーキテクチャは，次の3つの要件を満たしている必要がある。

1）製品情報をWebアクセスが可能なデータ・ストア内に収集する（Webデータストア）。
2）Webベースのミドルウェアと「インテグレーションウェア」を使用して，エンジニアリング・システムとビジネス・システムを迅速に接続する（CPCアプリケーション）。
3）Webベースのアクセスを，プロダクト・ライフサイクルのさまざまなプロセス全体に亘って，すべてのユーザーに提供する（Webアクセス）。

Webデータストア：
製品データを変換して，Webサイトのデータ・ストアに保存する。CPCポータルは，ユーザーがプロダクト・コマースを実行するために必要とする情報，アプリケーション，およびプロセスへの単一の入口を提供する。これを達成するためには，すべての製品情報資産（例：CAD/CAMモデル，コンポーネント・データ，パーツ・データ）を，どこからでもアクセス可能な，セキュリティの高いリポジトリ（すなわち，Webサイト）に保存しなければならない。ただし，データの物理的な位置は，別の場所でもよい。ユーザーは，更に別の形式の製品のイノベーションに関する資本（例：設計の知識，生産工程の知識，一連のディスカッション・スレッド，その他の「非定型的表現」を持つ高水準なイノベーションに関する知的資本）を取り込むことで，製品情報を拡充するよう努めるべきである。そうした資本を活用することが，より革新的な製品の設計/製造に結びつくことになる。また，外部のコンテンツ（例：コンポーネント・ライブラリや，他のWebインフォメディアリから引き出された情報）を最大限に活用することを念頭において設計される必要がある。

CPCアプリケーション：
CPCプロダクト・アプリケーションには，エンジニアリング情報と強い関連性があるアプリケーション（例：CAD, CSM, PDM, およびVPDM）や，製品情報のサブセットを必要とするアプリケーション（例：ERP, CRM, およびレガシー・システム）が含まれる。ここで重要なことは，アプリケーションが製品情報と自由に対話できるようにするフレームワーク，また，プロダクト・アプリケーション資産の組み合わせの迅速な変更によって，動的な対応能力を備えた新しいアプリケーション（例：CAD, PDM, およびERPアプリケーションからのデータ/機能を使用して，設計変更のライフサイクル効果を理解できるようにするWebページ）を作成できるようにするフレームワークを構築することである。このためには，APIを使用して特定用途の専用アプリケーションをモノリシックなアプリケーション・スイートに結び付けるというアプローチよりも，XMLやEAIアダプタを通じた疎結合というアプローチの方が有利であると考えられる。

Webアクセス：
CPC環境へのアクセスは，「パーソナライズ」されたポータル・レンズを通じて行われるようになるであろう。これは，CPCを利用しようとするユーザの役割に応じて情報の利用を可能にし，必要なナビゲーションを行うことを可能にするものである。役割ベースのインタフェースは，CPCポータル内に含まれている資産の表示，アクセス，お

付録１．２　ＰＤＭソリューション

製造業における製品データ管理ソリューション

（日本総研　プレゼンテーション資料）

付録２　データ表現法の現状

収録資料

付録２．１　対談 STEP を語る

Web Site http://www.jstep.jipdec.or.jp/ja/step/talk1.htm　より

付録２．２　XSIL:Java/XML for Scientific Data

Web Site http://www.cacr.caltech.edu/projects/xsil/xsil_spec.pd より

付録２．３　UDF 改訂仕様書 Version 2.0

名大　土井プロジェクト　資料より

付録２．１　対談STEPを語る

Web Site http://www.jstep.jipdec.or.jp/ja/step/talk1.htm　より

実務レベルでの実用化展開を期待

木村　今後どうするかということについての感想です。

この２年間で非常に大きな進歩がありました。これは国が支援したプロジェクトとしてはかなり良かったと思います。国のプロジェクトは、特に実証と名前のついたものがそうですが、それだけで終わったら何にもならない訳です。勉強したという成果だけを残すのでは投資が生きない。この後、各業界がどういう取り組みをするかが非常に大事です。

表１に自動車業界が今後実用化を前提として具体的なビジネスパートナーとの間で交換実験を行っていくとあるが、非常に健全なやり方です。いつまでも実証実験とか国のプロジェクトに頼るのではなく、自分の実務レベルでの実用化展開をはかるために２年間の成果を生かす、という姿勢で、国の投資を正当化できることを期待したいと思います。

工業規格をどう活用すべきか

司会　先生が平成８年度調査研究報告書（STEP推進センター発刊）で、「規格開発に対する産業界の役割」についてお書きになっていますが、産業界はこの規格をどう活用すべきかという視点で、産業界の規格に対するあり方あるいは規格開発の役割についてうかがいます。

呼び水を超えている――企業は自分のツールとして位置づけを

木村　国内対策委員会などで何回か話しています。

規格には製造業、産業界がビジネスをうまくやるために必要だというものと、その範疇に入らない国とか公共組織が生活や社会を維持するのに必要なものがあります。国が主導してやるのは後者でなければならないでしょう。民間企業体が活動するための規格は民間がやるべきです。

典型的なものがアメリカのネットワークで、政府が提唱した規格に民間は乗らないで自分たちでやるというケースで、非常に健全な話です。

自分たちのビジネスは企業が一番よく知っているわけだから、自分たちのニーズに従って自分たちで推進すべきものです。その一部を国家戦略として産業競争力を維持するために国が支援するのはある意味では必要かもしれないが、やりすぎると現在のグローバル化の流れに反するわけで、国は控えめであるべきです。環境を整えて、その中で主体となる産業が自分のニーズで先頭に立たない限り健全な規格はできません。

このような意味で、STEPは明らかに今のところは産業規格です。実は、私は最初から期待していたのですが、将来STEPの利用が広がってくると一般の消費者が企業情報をとったり製品を買ったりするのにSTEPを使えるようになるかもしれません。広い意味で自動車はそういう製品になりつつあると考えることができますが、消費者が製造会社に近づいて自分たちのニーズを入れた製品を買うときに情報を使うということです。しかし、そのような形で、STEPが消費者に近くなるまで、まだまだ遠いでしょう。そこまで行かない間は生産システムに関する規格なので、産業規格であるといえるでしょう。

そういうことをやるときに、国家戦略だからと国が戦略を立てて資金を出し、企業にやりなさいというのは後進国のスタイルでしょう。日本はそのような後進国ではないし、先端産業があるわけで世界戦略でやらなければならないのです。国際企業としては日本の国益を考えるとともに、グローバルマーケットで成功する戦略をとるべきであり、その観点から必要な情報システムを考えていくことが大事です。

そうするとSTEPは当然、重要なはずですね。それを情報システムの根幹として位置づけられないような企業戦略をとっている企業は先がないと言ってもよいでしょう。このようなことを背景として、最初は呼び水として国の支援があるかも知れないが、いつまでもそれを期待するのは後進国のスタイルであり、先進国として自分たちのビジネスとして推進できるようにしなければならないでしょう。

日本はそういう時期に来ていて、この２年間のプロジェクトは呼び水を超えているとも思われます。今後はそういう位置づけで、企業は自分のツールとしてこの規格(STEP)を位置づけない限り先は難しいと思います。

司会　STEPは産業規格であり、先ほどお話の自動車CALSにおける位置づけは呼び水である。

これからが後進国であるか、先進国の仲間入りするかの分かれ目になる。これに対する予見あるいは予想をお聞かせください

自動車業界は1999年までにはSTEPの実用化を開始

長坂　いま先生のおっしゃったSTEPは産業規格である、産業界が主体でなければならないというのは全くその通りです。ただ現実は、国の資金を貰わなかったら日本のSTEPはどうなっていたことやら。２年前でもSTEPに関わっている実務者レベルでは何とかしたいという気持ちは大いにありましたが、STEPを業界あるいは各企業の経営マターにまで至らしめることはできていなかった。

そこで、私は業界としてのSTEP取組み姿勢を各業界のマネージャーの方に伺いました。その折感じましたことは、各業界あるいは各企業がSTEPに対しそれほど主体的に取組んでこなかったのは、一般論で言われている、「日本人は標準化に関しては消極的である」という単純な話ではなく、それぞれの業界のビジネスの構図がその時点でSTEPなどの標準化をさほど強く必要としてこなかったからだ、とのお答えが多かった。

つまり、国際競争に曝されていないとか、製品開発生産プロセスが電子化データの流通を前提とするような形態になっていないからなどが理由として挙げられていました。STEP推進のインパクトとして、外圧でもなければダメとの話もありました。このことは、STEPなり情報の標準化を推進するには小手先ではなく、ビジネスの構図の改革自体が本質的な課題であるということを意味します。

このような実態を見ますと、日本の産業界でのSTEP取組みの必然性は今一との感もありますが、ただ、今日現在では、どの業界ともグローバル化が進展する中でビジネスプロセス改革の必要性を既に十分認識しており、STEPのような標準化の気運

要求のかなりの部分をSTEPが将来的に担う

木村　本質的には変わっていません。当時、1983～84年ごろ主にアメリカの航空機・防衛産業からSTEPの種が出ています。そこでは、情報の共有化はもともと基本的に重要で、実務として皆が理解していました。それから見ると日本の認識はずっと遅れていたように思います。

ライフサイクルサポートは、82年ボーイング社のキャル・ブラウナー氏が最初のPDES文書にすべて書いており、中身は当たり前のことですが私にとって新鮮でした。こういうことを実務で考えていて、日本企業の情報システムに対する認識と比べて、例えて10年くらいギャップがあったと言えるでしょう。STEPとしては、あまり変わっていないが、ただ日本の企業はこの10年で変わった、ということでしょう。

企業がビジネスをするのに何が必要か、特に情報システムはどうあるべきかという大きな企業戦略、情報システム戦略があって、その中で情報共有が基本的に重要である、という認識がある程度できてきたのではないでしょうか。実現する技術、ツールにはいろいろなものがあり、その時その時に会社の規模、業種・業態などに応じて、いろんなものから最適なものを選んで決まってきます。

その中でSTEPは１つの基盤ツールであり、規格として重要なものでしょう。それがSTEPの価値だと思います。
　逆にいうとSTEPありきで、STEPを使ったことで情報共有が達成されるというのは前後が逆で、この辺りの認識が曖昧なことが最近のSTEPがポピュラーになってきたことに対する非常な不安でもあります。何にでも情報共有とか情報環境、ネットワークとかがあって、そこにSTEPがあるよと一言書いてあります。書いている人がSTEPを分かっているのかどうか分からないのです。

考え方として逆で、否定的なことを言っているわけではなくこれは大事なことですが、STEPを使わなければいけないという言い方は意味がないのです。先ほどお話したように、大きな企業戦略の中から情報共有があって、情報共有の要求があったときにSTEPがきちんと使えるようになります。本来はこのような戦略を持ったところがSTEPの規格化を推進して、できるだけ広い戦略に適用できるようにしていくのがスジでしょう。

要約すると、何から何まですべてSTEPでやろうということはほとんど不可能である、いろんなやり方で違ってくる、その中で情報共有しようと思う要求のかなりの部分をいまのSTEPが将来的には担うだろうという意味において重要である、ということです。

もう１つは、STEPはビジョンである、ということでしょう。先ほどのライフサイクルサポートという言葉は今でもビジョンであって、リアリティーではない、これを実現するためには技術的課題があって、いくつかの技術的ギャップがある、ということです。それがわれわれアカデミアにとっては研究テーマとなります。これは企業の今日の実務としての情報共有ツールとは違います。

従来、STEPの規格化作業はビジョンを現実化するための研究課題と、現実になったものを共有化ツールとするための標準開発課題とがありますが、この２つが混同されています。それがSTEPを困難にした原因です。分かってやっていたというよりも、その辺の技術が未成熟だった、というべきでしょう。難しいことを難しいと認識できなくて今日にきたのが実態であると思われます。

これからだんだん良くなると思いますが、過去十数年間の歴史は難しいことをはっきり難しいと認識できなかったことが大きな問題と思われます。われわれが研究課題として考えたときは十分認識していたと思いますが、実用化に近いほうでスタンダードをやる観点の人が認識できていなかったのではないでしょうか。

司会　貴重なご意見でした。難しいものはやはり難しかったというバックがあって今後どう進めるかが大事です。STEPありきではなく、One of themという形で有用であるなら使っていくあるいは普及させていくことが次のフェーズで出てくると思います。
　日本の環境を考えると使うユーザーのことが一番大事で、それをサポートするベンダー、そしてSTEP推進センターがある。ユーザーが使うには効果がないといけませんが、いかがでしょうか。

ユーザーとベンダーの役割

STEP活用にシフトする素地は十分

長坂　企業は世界レベルで激しい競争の中に曝されています。自動車業界の例でいえば、自動車会社本体だけでなく部品メーカーなどサプライヤーの多くが、国内だけなく国際的な取り引きをしている。その中でサプライヤーは相手が要求する内容、納期で電子化データを提供できないと取り引きが成立しないという局面に立たされている。そういう世界なので企業は現実的な方法をとります。

例えば、CADデータに関しては、相手と同じシステムを導入して使うとか、直接変換とかIGESを使う。この今の今日的対応方法はそれぞれコストや運用面で問題を抱えており、これが未来永劫にいいとは誰も思っていない。STEPがこの問題への現実的ソリューションを与えてくれるなら皆さんは必ずSTEPを使うでしょう。

今後共、各企業は常に現実的な方策を選択しますがSTEPの実用化に明るい兆しが見えた現在、STEPの活用にシフトしていく素地は十分にあるといって良いでしょう。

STEP推進センター（JSTEP）の役割

司会　企業を支えるベンダーの役割が重要です。STEP推進センター（JSTEP）がある。海外では米国にPDES、欧州にProSTEPがある。PDESは国の資金が入っているかもしれませんが、それぞれがSTEPを非常に良く理解してアクティブな活動をしているように見られます。JSTEPの役割等についてどのように考えられますか。

本来の役割を認識して進める

木村　一般的に言えば、仮に今のJSTEPがないと考えると、やはりなくては困る訳です。例えばSC4の審議をしようと思うと、工業技術院から団体に委託されて審議するが年間予算が80～90万円では委員会もできません。いわんや国際会議に人を派遣するなど論外である、という状況です。いずれにしても国際規格の審議をする体制が整っていないので、何らかの団体を作って支援する必要があり、JSTEPのような組織は絶対に必要です。国際規格を審議して、産業が使っていくうえから必要という観点です。

もう１つは規格を実践する観点です。規格は決めたらドキュメント化して終わりでは何の意味もありません。

規格は本来作って使うものではなく、ニーズがあってそのために作るものでしょう。ニーズを持っているところで規格を作ろうという活動が当然あるべきでしょう。そういうものによって初めて本来の規格化活動が推進されるのでしょう。

STEPが必要な規格であれば、当然STEP推進センターのようなものがなければ活動はできません。そこでやるべきことは、規格化が必要だということの意見を集約して、作らなければならない姿を明らかにしていくことと、それに対応して国際規格活動があれば支援していくことの２つです。そのときの国家戦略、その他を絡めて国からの支援が受けられれば申し分ない訳です。

STEP推進センターがやるべきことは、会員が本来どういうニーズを持っていて何をやりたいかを吸い上げて、それに対し個々のユーザー、ベンダーがやりにくい仕事をすることでしょう。それは共通的に必要なことなので費用はユーザー、ベンダーが負担して、さらに公共度の高いものであれば、国の費用でやるのもよいでしょう。広い意味での共通的作業、例えばどこの業界とか応用にも属さない長期的な技術課題への対応などがこれに当たります

STEPに何らかの技術的欠陥があるとすれば、STEPそのもののフレームワークとか規格化の基礎に関わるような研究開発などをやることもJSTEPには必要かもしれません。本来の役割を認識して進めないと、いままで述べたことも実際的には進めにくいですね。

司会　JSTEPの活動は、先ほどから出ているユーザーや企業サイドに立ち、いかにそこで実用化の支援をしていくかが一番の骨格になるということですね。

乗り越えなければならないハードルがあるとすれば、実用上の問題点としてどのようなものがあるでしょうか。

コストのカベは必ず乗り越えられる

長坂　STEP規格の実用化に向けてのハードルは大きく２つあります。

その第１のハードルは、イニシアルリリース（AP203、AP202）の実用化の範囲に限定して注目するにしても、関係する多くの企業がSTEPの利用技術を持つことです。今や２０人以下の企業でもCAD/CAMシステムを駆使してビジネスを行っている企業が多くあります。そのような企業がSTEPを実業務の中にどのように取り込むのでしょうか。結局、ユーザーが頼りにしているのは、取引先のCAD/CAMベンダーさんである。

では、国内に多く存在するそのCAD/CAMベンダーさんはユーザーをキチンとサポートできるポテンシャルをSTEPに関してお持ちだろうか。外国製のCAD/CAMシステムの場合、外国本社にはその実力があっても、そのシステムを扱う日本の代理店自身はそのポテンシャルをお持ちでしょうか。

要は、STEPが各業界の中で広く普及するには、そのユーザー企業群をサポートするCAD/CAMなどのシステムベンダーさん自身がSTEPの実装や利用に関しての技術、知識を備える必要があるということです。多くのベンダーさんの今日の実態は、まだこれからだという状況であると思います。そこで、私はJSTEPがシステムベンダーさんのSTEPに関する技術力が備わるような施策に力を入れて欲しいと思っています。

第２のハードルは、一番本質的かつ現実的な事ですが、STEPを利用する際のコストです。例えば、STEPが経験と実績の豊富なIGESの利用にとって替わるには、STEPが扱う情報がIGESの世界を越えるということに加えて、コスト的にもユーザー企業にとって許容できるものでなければなりません。

このハードルはSTEPの有用性に確信が持てれば、ユーザー業界団体およびシステムベンダー双方の努力と工夫で必ず乗り越えることができると思います。

STEPの抱える問題点

司会　産業界がSTEPを使っていけば、STEPの恩恵に預かるところがありうるでしょう。中小企業の皆さんにも恩恵が早くもたらされるようにすべきですが、JSTEPがやるのは、直接的には難しい。この点はベンダーが中心になり進めることと考えます。

次にSTEPが抱える開発の問題点について木村先生からお願いします。

研究開発と標準化を混同せず、的確なマネジメントを

木村　STEP開発の問題点は沢山ありますが、十分な技術検討がされていない、技術が未成熟であり、やさしい問題と難しいものをごちゃごちゃに議論している、適切な人材が入っていない、など繰り返し議論されています。

今でもSTEPの規格のフレームワークとか全体構成の再編成など多くの技術検討がされていますが、そこに上がっている技術的課題は、技術情報をどうデータベース化するかというような繰り返し議論されている古典的な問題で、すぐ解けないことは分かりきっているような問題も多いのです。研究開発的にみると、技術の検討が極めて甘いという点があります。

それが研究開発と標準化を混同しているということです。標準化するときは3年～5年というスケジュールで規格化できることをやる一方、過去10年くらい議論されたけれど全然解が見つからない難問題については、それをまともに持ち込んできても何も出てこないので、そういう点が大きな問題であることを認識して、研究開発をやる、ということです。

STEPがだめというのでなく、本来のSTEPの目的をはっきりさせ、それに対する必要なリソースを組織的に開発する、という極めて当たり前のプロジェクトマネジメントをしなければならないのでしょう。

一方、先行的に将来を見るような研究開発をやることも必要で、それは何年という期限を切ることはできないでしょう。これが私の一般的な感想です。

司会　私もSTEPに十数年関わってきましたが、国際標準の常なのかもしれないですが、出来上がったIS（International Standard）規格も、一度も現場で使いこんだ上での評価をしていない状態です。これが今のSTEP国際規格の大きな問題と考えます。

本来の先取り標準の精神に立ち返り検討

木村　われわれは一時STEPを先取り標準といっていました。意図していたところは、出来上った後で規格化しようとすると、バラバラになったものをまとめようとすることになり大変だから先回りして規格化しよう、ということでした。

ある意味では当然ですが、それが誤解されている感じがありますね。先取りなのだから、ということで技術や業務の現実を軽視していることはないか、規格化における実証を無視していないか、というようなことですね。

良い実例でいうと、いろいろなハードウエアの規格、IEEEの計算機間の高速インタフェース、DVD規格などがあります。使い込んだ技術を規格競争に持ち込んで競争しています。どれをとっても製品化できるような成熟したものでないと競争できません。製品化については後でどうこうしようでなく、製品化してないが製品化できるための技術の集積をする厳しい議論で決めています。

これが正しい先取り標準であり、STEPはこれらとはちょっと違うようです。現状がダメなのだからビジョンに基づいてあるものを作り出して、上から下へおろしていこう、という考えは研究開発では良いかもしれませんが、実務的な支援がしっかりしていない限り、国際規格をそのようにして作っていくことは難しいでしょう。

長坂　自動車用の応用規格（AP214）などでは、まだ、業務プロセスとして十分検討ができていない領域を含めて規格化しようとするような「先取り」になっている感じがあります。

木村　われわれ自身の反省でもあるのだけれど、そういう点をもう一度本来の先取り標準の精神に立ち返って真面目に検討する必要があります。

司会　大変難しい問題だと思います。

産業界の現場で着実に使っていく――更なる発展へ

長坂　これまで、自動車業界としてAP214の規格制定に取り組んできたのは、企業間ビジネスで実用化するという観点の他に、これから各企業が情報化の過程でいろいろなシステムを開発する際、STEP規格が情報モデルの指針として参考になるであろうという意識はありました。だからやったわけです。真面目に使うところと、将来のために一度整理しておこうというようなレベルを含めてやってきました。

最後に、改めて強調しておきたいと思います。STEPは、その理念は高くフォーカスは非常に広いのですが、理由はどうであれ、規格制定および実用化に時間がかかり過ぎているという大きな問題を抱えています。幸いイニシアルリリースされた規格が既に存在しています。従って、今後のSTEP活動の重点は、これらの規格を産業界の現場で着実に使っていくことを軸に据えて取り組むべきと考えます。

この実用化の活動の結果から、STEPは更なる発展があるのではないかと思います。

JSTEPの今後の活動もこのような考えの展開に沿う形になってほしいと願っています。

付録２．２　XSIL:Java/XML for Scientific Data

Web Site http://www.cacr.caltech.edu/projects/xsil/xsil_spec.pd より

XSIL: Java/XML for Scientific Data

Roy Williams

California Institute of Technology
6/27/00

Abstract: This paper describes the XSIL (Extensible Scientific Interchange Language) system that connects scientific XML to a Java object model and powerful visualization. There is an XML format for scientific data objects, and a corresponding Java object model, so that XML files can be read, transformed, and visualized. We describe the Xlook object browser, which reads XSIL files and allows them to be visualized in tree-like way, with viewers including a chart widget and a sortable table. We describe how to customize the XML format, and build custom viewing components that are dynamically loaded by Xlook. All code is open-source, obtainable from `http://www.cacr.caltech.edu/XSIL`.

1. Introduction

A. What is this?

1. XML is a "Language to make languages", a simple, yet effective way to build structured documents that is revolutionizing business practices on the Internet. An XML document is a hierarchy of "elements", each of which has a textual "tag", some "attributes", some text, and may also contain other elements.

2. XSIL is some basic syntactic structure for scientific data (Table, Array, Param, Time, etc), together with a mechanism to extend both the XML side and the Java object side. XSIL also offers a modular, extensible viewing platform called Xlook.

3. An XML document may refer to a DTD (document type definition) that expresses its structure. The DTD is analogous to the Class of object-oriented software, and the XML document is an instance of that DTD in the same way that the Object is an instance of a Class, or an Apple is an instance of Fruit.

4. For example:

```
<Book ISBN="0439139597">
    <Author>D. Scarlatti</Author>
    <Title>The Big Fight: Harpsichord vs. Violin</Title>
    <Author>A. Stradivarius
        <Email>strad@violin.org</Email>
    </Author>
</Book>
```

5. This is a record that expresses structured information about a book. The DTD for such an XML scheme defines syntactic information, such as the idea that a book has one or more authors, but exactly one title; that an author may have zero or more email addresses, but a title cannot have an email address. An example of an *attribute* is the ISBN attribute associated with the book element. When an XML file is parsed (by any XML application, not just XSIL), the syntax is compared against the DTD, and non-compliant files detected. An XML editor is powerful because it 'knows' the document structure through the DTD: once a title has been entered for the book record, the editor will not allow another title.

6. The XSIL DTD defines a few basic elements that form the core of scientific data. The Table is modelled on the relational table and expresses a collection of records, all of similar structure; the Array is for low-dimensional data cubes such a images, spectra, time-series, voxels, and so on. The Stream element provides a link to external and encoded data through files, URL's, Big/Little-endian and Base64 encodings, as well as delimited text.

7. The idea is that XSIL can be fashioned to meet the needs of a particular scientific discipline by building combinations of these base elements that have semantic meaning in that discipline, then extending XSIL appropriately to handle these. Dynamic loading ensures flexibility without relinking code or struggling with makefiles. The easiest way to extend XSIL does not entail DTD changes, but use of a special XML attribute `Type="MyStuff.MyObject"`.

8. Such extensions can also be used to build viewing extensions for custom objects, extensions that can use all the

power that comes with Java, including the Swing GUI components, advanced imaging, Java3D, the LiveTable component with direct database access, and the JChart component for graphs.

9. XSIL is a pure-Java application, so that portability and programming efficiency have been emphasized over CPU speed. However, it can deal effectively with very large quantities of data, using the following three strategies.

• First, the use of external files separates metadata (XML) from data (binary). Very large XML files are not efficient, and most XML software cannot handle more than a few megabytes. However, if a user wishes to create very large XML files, there is a utility with the XSIL distribution (Splitter) to break up such a file into a small metadata file, in XML, and many external data files.

• Second, data is not fetched from external files until it is needed, so that an XSIL file can contain hundreds of links to external data, but the parsing can still be fast.

• Third, XSIL stores data efficiently, so each 4-byte integer takes up only 4 bytes asymptotically. When data is fetched from a link, efficient calls are available for bulk transfer.

B. More information on XML and XSIL can be found at:

• XSIL web site: `http://www.cacr.caltech.edu/XSIL`

• The World Wide Web Consortium standardization effort for XML: `http://www.w3c.org/XML`

• Seybold publishing XML site: `http://www.xml.com/`

C. Reasons to use XSIL

1. XSIL provides an XML model for data transfer which is being adopted in the gravitational-wave and astronomical communities. Furthermore, it is XML, with an increasing range of transformation filters, making it a very plastic language. There is a **named, typed hierarchy** through the collection object, and base objects are **Table, Array, Param, Time,** and others, and the arrays and tables may be in these primitive types: boolean, byte, short, int, long, float, double, floatComplex, doubleComplex, and String. There is a Stream object to extract byte streams and primitive streams from files, URLs, databases, and delimited text.

2. An **object model** with Java API. Users can provide a file name or URL to the XSIL library, which returns an object which is the root of a tree. This generic container (class XSIL) may have been extended to form one of the objects described below (eg. Time, Table, Array, Param, etc). The structure of an XSIL file is defined by its DTD, saying, for example, that each `Array` must have at least one `Dim` element to give its dimensions. This model may be extended by adding domain-specific tags and the corresponding code to XSIL. New entries would be added to the DTD to define the syntax of the new structure.

3. The object model is extensible in a different, looser, sense: if the XML file has a tag such as `<XSIL Type="Banana">`, then the XSIL reader will seach for a class called `Banana` in the place where user extensions are built, for `class Banana extends XSIL`. In this way we establish a highly flexible connection between the XML file and the code that handles it. If such code is not found, then the XML element is assumed to be a generic collection object. This method of extension does not entail changing the DTD. Several extensions are in the XSIL distribution, an example being `TimeSeries`. This element consists of a one-dimensional array and a pair of parameters that define the start time and the delta-time between samples. Because of the dynamic linking, object handlers (not just objects) may be emailed to colleagues.

4. One major code written to the XSIL library is the **object browser (Xlook)**, that is intended as a way to view XSIL files and their contents in the same way as a web browser can be used to view collections of files. There is a tree-representation of the elements of the file, shown alongside the XML itself. When an object is "Viewed", then its specific viewer is displayed.

5. The object browser **can be extended with user-written code.** An example of this is the Java3D extension to XSIL: certain arrays are defined in XSIL, and an extension defined (as in (2) above). This extension treats these arrays as coordinates, colors, etc, and can display them in a Java3D window. Similarly the TimeSeries can be graphed.

D. Uses for XSIL

1. As a flexible and general transport format between disparate applications in a distributed archiving and

computing system; a text-based object serialization that can be handled by common tools, or

2. As a documentation mechanism for collections of data resulting from experiments or simulations; with all the parameters, structure, filenames and other information needed to keep a complete scientific record.

3. As an "ultra-light" data format: a user can, if he wants, simply read the markup, then delete all except the actual data, or all except for the filenames where the data may be found.

4. All XML files including XSIL, can be created and edited with a large range of high-quality, customizable tools.

5. XML files can be stored in an XML database, such as eXcelon (www.exceloncorp.com). Such products provide rich tools for storing, querying, presenting, and connecting to XML data.

E. Future Extensions

1. Data access: In addition to local files, URL's, and XML-encoded data, we would like to work with databases directly, using JDBC to view and edit database tables.

2. Output formats: In addition to the Matlab output mechanism, we would like to consider data output as IDL, multichannel Photoshop, FITS, Ligo Frame files, and Microsoft Excel.

3. Viewers:a viewer for Ligo Frame files will be connected, and for astronomical applications, a FITS viewer also .

4. Service definitions: objects that tell a client application how to format and send requests to a remote service, and the kinds of response that the service will provide. Such a definition could contain an XHTML form both to provide a human input, but also to define the syntax of the request that the service is expecting.

2. XML Basics

A. Syntax

1. An XML file is a hierarchical structure of elements that may contain other elements. An *element* generally consists of a *start tag*, a *body*, and an *end tag*, for example: `<Fruit>Banana</Fruit>`, where start and end tags are distinguished by the presence of a slash.

a) An element may be ***empty***, meaning that there is only a single tag, with no body, for example `<EmptyElement/>`; note the position of the slash.

b) Elements may contain ***attributes***, for example: `<Fruit color="yellow">`. There can only be one instance of a given attribute name in any tag.

2. XML is case-sensitive, so that `<Apple>`, `<apple>` and `<APPLE>` are all different tags.

3. Comments in XML are delimited like this:

```
<!-- This is a comment -->
```

4. An XML document can be handled by both human and computer. If a human sees the document, then the XML is *presented*, usually by means of a style language file, or through a filter. If a computer sees the document, there is an API to control a parser of the document.

3. XSIL files

A. Header

1. All XML files, which includes all XSIL files, have the following as their first line:

```
<?xml version="1.0"?>
```

2. The second line of an XML file may make a reference to an externally defined Document Type Definition (DTD), which defines the syntax of XSIL files. Thus every XSIL file has the following as its second line one of these:

```
<!DOCTYPE XSIL SYSTEM "http://www.cacr.caltech.edu/projects/xsil/xsil.dtd">
<!DOCTYPE XSIL SYSTEM "XSIL.dtd">
```

The DTD can be referenced from a remote location via the URL syntax, or locally with the file name. This file is presented in section 8.

3. There may follow other XML elements, but the XSIL parser will ignore these until it finds a XSIL element. Only the first XSIL element is then considered, so there should never be more than one in a file. Thus, in general, the third line is:

```
<XSIL>
```

and the last line of the file closes this element:

```
</XSIL>
```

4. When contained in this first, plain container element, containers may be typed, leading to user-side code being called. Currently,the Type of the outermost element is not handled properly.

B. Object Name and Type

1. Any of the XSIL elements may have a Name attribute, for example:

```
<Table Name="Bake_Out_Temperature">
```

While the names may be useful in presentation of the XSIL, the major intended use is in navigating the object model, through methods that find an object of a given name.

2. Any of the XSIL elements may have a Type attribute, which may also be used for navigating the object model. However, we are most interested in this attribute to link this XML element to a Java object, thus extending the container object:

```
<XSIL Type="MyStuff.MyObject" Name="Jack">
```

causes the linker to look for the file $XSIL_HOME/extensions/MyStuff/MyObject.class to act as a handler for the object called `Jack`.

C. The Container Object

1. XSIL

There is a generic `<XSIL>` element which is a container for other elements, including other `<XSIL>` elements, thus inducing a hierarchy. Each of these may have a `Name` attribute, to provide hierarchical naming that is visible from the API. When the file is parsed, there is a representation of the first XSIL element that is found in the document. Here is a XSIL fragment that consists of a container hierarchy with an array at the second level and a parameter at the third level:

```
<XSIL Name="Fruit">
    <XSIL Name="YellowFruit">
        <Array><Dim>7</Dim></Array>
        <XSIL Name="Banana">
            <Param Name="Inductance">1.34</Param>
        </XSIL>
    </XSIL>
</XSIL>
```

2. The XSIL object is the only one that can contain other XSIL objects. All the others (below) can only contain only:

a) The elements explicitly defined in their description, or

b) Stream object from which data may be requested by the code handling the object.

D. XSIL Base Objects

1. Comment

A comment object in XSIL is not meant to be interpreted or parsed. It appears only in the presentation of the document. For example:

```
<Comment>The data that follows is probably wrong</Comment>
```

2. Param

A parameter in XSIL is an association between a name and a value. In addition, it may have a Unit attribute. For example:

```
<Param Name="Fruit_Mass" Unit="kg">0.387</Param>
```

The meaning here is "Fruit_mass = 0.387 kg", usually found in "parameter files" or "header files" in scientific computing. If the element is empty, then the value is read from the corresponding stream. Note that the value of the parameter is not typed as float, int, etc, but rather it is string-valued.

3. Time

In the LIGO observatory, as with many other experiments in physical science, it is critical that timing information be not only accurate, but also easy to understand. The `<Time>` element in XSIL can represent either "natural" time (ISO-8601 standard, YYYY-MM-DD HH:MM:SS.mmmuuunnn), or GPS time, or "Unix time" (seconds since 1/1/1970). The different formats are differentiated by the `Type` attribute in the tag:

```
<Time Type="ISO-8601">1998-11-08 17:40:00.032</Time>
<Time Type="GPS">594582000.032</Time>
```

```
<Time Type="Unix">910546800.032</Time>
```

The default for the type is ISO-8601.

4. **Table**

A table is an unordered set of *records*, each of the same format, where a record is an ordered list of values. The contents of a record are defined by column headings, each of which may have a unit and a type. This definition of a table should be thought of as similar to the table object that is found in a relational database; we should point out that this is *not* the complex and exotic typographical beast of TeX or HTML.

The only tag specific to the Table object is `<Column>`, which specifies the name, type, and possibly units associated with one of the columns of the table. It can be thought of as the heading of a column in a table. Shown below is an example table that includes a stream from which the table can be populated.

```
<Table>
    <Column Name="ChannelName" Type="string"/>
    <Column Name="Site"        Type="float"    Unit="meter"/>
    <Column Name="Clock"       Type="float"    Unit="hour"/>
    <Column Name="Description" Type="string"/>
    <Stream Delimiter=",">
        "History Channel", 405.0, 3.7, "Another Channel"
        "Math Channel", 307.0, 2.1, "Channel about Math"
    </Stream>
</Table>
```

5. **Array**

An array is a collection of numbers (or other primitive type) referenced by subscripts, which is a list of integers whose maximum values are given by the list of dimensions of the array. This definition is very close conceptually to a Fortran or C array, with the `Type` attribute of the `<Array>` tag specifying which primitive type is contained in the array (float, int, etc.). The Dim element specifies the dimensions of the array, but it does not specify the subscript ranges. For a dimension of 5, a Fortran binding of the API would label subscripts from 1 to 5, but a Java or C++ binding would have subscripts from 0 to 4.

As with other XSIL objects, the Array tag may have `Name` and `Type` attributes. The Type attribute is interpreted as the data-type for the data in the aray. The only element specific to this class is `<Dim>`, which may have a Name attribute to specify the name of that dimension of the data for presentation purposes. For example:

```
<Array Type="int">
    <Dim Name="X-axis">5</Dim>
    <Dim Name="Y-axis">3</Dim>
</Array>
```

which specifies a 5x3 array of integers, with the last dimension changing fastest. The presumption is that 15 integers may be read from the Stream associated with this Array.

6. **Url**

This tag is quite similar to an HTML link:

```
<Url href="http://www.cacr.caltech.edu/XSIL">XSIL Home Page</Url>
<Url href="http://www.caltech.edu/">Caltech Home Page</Url>
```

There is an `href` attribute, containing the URL, and the text part is a message associated with the link.

4. Streams

A. Connecting Streams and XSIL Objects

1. Some of the objects from the previous section are self-contained, for example Param and Comment. Others (XSIL, Table, Array) define the structure of a data object, without necessarily making available the data itself. The Stream object provides the input that allows these structures to be filled.

2. A XSIL document may define many streams and many XSIL objects (Table, Array etc.). In general a stream may contain the data for many different objects. We use the collection mechanism induced by the XSIL tag to provide this association.

3. For a stream to be available to a XSIL object, either:

- The object contains the stream explicitly between its start and end tags, or
- the collection in which the object resides must contain exactly one Stream. The objects in the collection will then be read sequentially from the Stream.

4. Delimited Text

```
<Stream Type="Local" Delimiter=",">
    4.76,5.77,8.99,3.44,2.11,0.93
</Stream>
```

The delimiter string are additions to the default delimiter string, consisting of the newline character. Thus a string read in this way cannot contain a newline, and the String primitive in XSIL cannot contain a newline.

5. Remote data

```
<Stream Type="Remote">
    datafile1.dat
</Stream>
```

The stream may have Type `Local` — the data is contained in the XSIL file itself — or it may be `Remote`, as in this case. Local is the default type for a stream. Data is read from the local file system. The name is assumed to be relative to the XSIL filename, not relative to the current working directory. Thus XSIL files and their data can be packaged and moved without renaming files.

If the file name is fully qualified, it can be either in Unix (`/blah/blah`) or Windows style (`C:\blah\blah`).

6. Endian encoding

```
<Stream Type="Remote" Encoding="Littleendian">
    datafile1.dat
</Stream>
```

Java reads and writes bigendian binary, as this is the same byte-ordering as Sun computers have. Windows machines, though have the opposite ordering, so binary files made by a PC will need to be converted from little-endian format as above.

7. URL streams

```
<Stream Type="Remote">
    http://www.cacr.caltech.edu/blahblah/datafile1.dat
</Stream>
```

If the stream has type `Remote`, and the text in this element contains the string "`://`", then it is interpreted as a URL-type locator for the data that this stream represents. The file:// and the ftp:// and http:// protocols are supported.

8. Base64 streams

```
<Stream Encoding="Base64">
    AAAAAAAAAAEAAAACAAAAAwAAAAQAAAAFAAAABgAAAAcAAAAIAAAACQ==
</Stream>
```

The first ten integers have been encoded in Base64 and put directly into the XML document.

B. Stream Content

1. The XSIL data stream can be considered to be an arbitrary sequence of bytes. In the ImageList demo (section 7C), streams get the image data, to be delivered as gif. Some of the images are external files, others are base64 encoded as part of the XSIL file.

2. In the ImageList code, a Java `InputStream` is obtained from the `<Stream>` tag in the XML. The stream of bytes is read in and converted to `java.awt.image` and displayed with a chooser. The metadata for each image is also shown.

3. In other cases, however, for example the Array and Table implementations, the Stream contains something higher level than raw bytes: a sequence of primitive types.

4. The types that are available for Tables and Arrays are listed in the following (with the corresponding number of bits), and some alternate spellings:

- `boolean` (1)
- `byte` (8)
- `short` (16) (int_2s)
- `int` (32) (int_4s)
- `long` (64) (int_8s)
- `float` (32) (real_4)
- `double` (64) (real_8)
- `floatComplex` (64) (complex_8)
- `doubleComplex` (128) (complex_16)

• `String` (arbitrary length) (lstring, char, character)

5. To read such a stream, there may be some filtering process. A base64 encoded stream is converted to binary byte stream, available as a java.io.InputStream. If the stream is to be read as a sequence of primitives, it is converted to java.io.DataInputStream, and the byte swapping necessary for little-endian conversion may then be applied.

6. If the stream is text-based, either local or remote, the newline character is always a delimiter, plus any other characters nominated in the Delimiter attribute of the Stream element. The text is read line by line from the source, and primitives generated. If there is data that cannot be converted to the relevant type (eg. the number 3.56A7464), then an exception is **not** thrown, but rather a default value is put in place.

7. The array of primitive objects contains strings, it cannot be read from a binary stream. Strings must be handled as delimited text.

8. Any primitive type can be converted to any other: to convert from complex we take the real part, to convert to boolean we ask if it is nonzero, to convert to String we use the toString() method.

C. Stream Encoding

1. The `Encoding` attribute specifies how the data in a `Stream` is encoded. It is a comma-separated list chosen from the values:

• `Text, Binary, base64, BigEndian, LittleEndian, Delimiter`

2. The Text attribute is assumed by default for Local streams, the Binary attribute is the default for Remote streams.

3. When the stream is Base64, it is decoded by the XSIL library. When the stream specifies an Endian order, it is converted to the Endian order appropriate to the current machine before being delivered to the application code.

4. The other attribute defined in this section is `Delimiter`: It is only relevant in the case of a Text stream. The characters in the attribute are appended to a delimiter string that already contains newline.

5. Extending XSIL

A. Object Model of XSIL

1. When a XML file is read by an application program through the XSIL API, the hierarchical structure of the file is parsed to a hierarchical *base object*, which is then made available to the application. The XSIL software layer then extracts from the base object the first `<XSIL>` element, which is returned to the application. Thus XSIL can be mixed in a straightforward way with other kinds of XML or HTML, such as Math Markup Language, Chemistry Markup, or other XML languages.

2. As well as the parser, the XSIL API provides a rich set of methods to extract objects of given element-type, objects which have given attributes, given Name or Type, and so on.

3. Information can then be extracted from objects. For example once a Param object of given name has been found, the string which is its value can then be extracted. In this way we have a dictionary of name-value pairs. Similarly the rows and colums of a Table, or the start and end time of a TimeSeries object are available to the application.

4. We can extend XSIL in an informal, perhaps personal way, simply by creating a collection object with "well-known" parameter names. In this case, we expect the application that is reading the file to understand the special significance of the word `Type="TimeSeries"`, and to know the names of the parameters that it expects to find.

B. Example: TimeSeries

1. The following example illustrates the collection of parameter names that would be appropriate for a time-series object. To make a valid TimeSeries there must be (in addition to data), Start time (t0), and delta time between samples (dt). Given these, and the number of samples (from the dimensionality of the array), the third parameter can be computed.

```
<XSIL Type="TimeSeries.TimeSeries" Name="My Time Series">
    <Comment>A sample time series</Comment>
    <Param Name="t0">6.0</Param>
    <Param Name="dt">0.001</Param>
    <Array>
        <Dim>1000</Dim>
        <Stream Type="Remote" Encoding="LittleEndian">
            mydata.dat
```

```
        </Stream>
    </Array>
</XSIL>
```

The XSIL parser looks from its current classpath to find code to handle this object. In this case, the code is in a classfile called TimeSeries in a directory called TimeSeries.

2. The code to handle the extension type could look something like this:

```
package extensions.TimeSeries;
import org.escience.XSIL.*;
import java.util.*;

public class TimeSeries extends XSIL {
    double t0 = 0.0;
    double dt = 0.01;
    int ndata = 0;
    Array a;

    public void construct(){
        for(int ichild=0; ichild < getChildCount(); ichild++){
            XSIL x = getChild(ichild);
            if(x instanceof Param){
                Param p = (Param)x;
                if(p.getName().equals("t0")){
                    t0 = new Double(p.getText()).doubleValue();
                }
                if(p.getName().equals("dt")){
                    dt = new Double(p.getText()).doubleValue();
                }
            }
            if(x instanceof Array && ((Array)x).getNdim() == 1){
                a = (Array)x;
                ndata = a.getNdata();
            }
        }
        if(ndata == 0){
            System.out.println("ERROR: improper TimeSeries");
        }
    }

    public double gett0() {return t0;}

    public double getdt() {return dt;}

    public int getNdata() {return ndata;}

    public double getData(int i) {
        return a.getPrimArray().getDouble(i);
    }
```

3. The constructor for the new object is handled by XSIL, and initalization of the new object is handled by overloading the `construct(XSIL x)` method. The vector of child objects is examined, looking for the data we need to make the TimeSeries, being some parameters, and an array. If the parameters have the right names, it is assumed that numbers can be read from the corresponding text, and if the array is one-dimensional, it is assumed that it is the TimeSeries data. Note that the TimeSeries may contain other XSIL objects, which are ignored by this piece of code.

4. The other methods provide access to the TimeSeries itself, the last being the data. Associated with each Array or Table object is a PrimArray object, which holds a sequence of any of the ten primitive types (boolean, byte, short, int, etc.) from which any desired type can be extracted.

5. The data making up the time series is not read at the initialization of the object, but rather at the first call to the getData method. Files are read or URL's resolved, or delimited text is parsed. If the number of objects in that data stream is not sufficient, default values are substituted instead.

6. The `getData` call in the TimeSeries implementation fetches one number at a time. Another call is available from XSIL to get large quantities of data, it is of the form `double[] getDoubleArray(int start, int end)`, which creates an array of doubles with `end-start` elements (subscripts from `start` to `end-1`). Othe calls are available for all ten primitive types.

7. Parent objects are constructed after their children. If a user-created object is a child of another user-created object, then the `construct()` method of the child has been called before the `construct()` of the parent.

```
        <Param Name="Java3DGeometryType">TriangleArray</Param>
        <Array Name="Coordinates" Type="float">
            <Dim>60</Dim>
            <Dim>3</Dim>
            <Stream Encoding="Text" Type="Remote" Delimiter=", ">icocoord.dat</Stream>
        </Array>
        <Array Name="Colors" Type="float">
            <Dim>60</Dim>
            <Dim>3</Dim>
            <Stream Encoding="Text" Type="Remote" Delimiter=",">icocolor.dat</Stream>
        </Array>
    </XSIL>

    <XSIL Type="Java3DGeometry">
        <Param Name="Java3DGeometryType">LineStripArray</Param>
        <Array Name="Coordinates" Type="float"><Dim>5</Dim><Dim>3</Dim>
            <Stream Encoding="Text" Type="Local" Delimiter=", ">
                -1.0,   0.0,  -1.61803399,
                -1.0,   0.0,   1.61803399,
                 1.0,   0.0,   1.61803399,
                 1.0,   0.0,  -1.61803399,
                -1.0,   0.0,  -1.61803399,
            </Stream>
        </Array>
    </XSIL>
    .... (two more rectangles like the one above) ....
</XSIL>
```

There is an outer enclosing element of type Java3D.Java3D, which contains a number of objects of type Java3D.Java3DGeometry, which are the Triangle Arrays, Line Arrays, and so on which can be used to create 3D objects in the view window. Of course, other types of data can be encoded in the XML, then rendered in a different way by modifying the rendering code (in the extensions/Java3D directory).

7. Extending Xlook

A. Object Model

1. An Xlook view component is an extension of a class XSILView, which extends javax.swing.JComponent, which is a generic graphics object in Swing.

2. The class implementing the viewing must have the same name as the object which it views, but with the suffix "View" added. In the code below, for example, the Label object is viewed by a LabelView object.

3. When an Xlook user requests a view component, an outer frame is made by Xlook, with the Name and Type of the object, and a scrollable panel (javax.swing.JScrollPane) is created to hold the client-supplied viewer.

4. The instantiate() method of the viewing object is called, with the argument guaranteed to be castable to the expected object, and the completed frame rendered. This code may create buttons and other widgets that can receive events, as in the TimeSeries viewer shown above, where a checkbox can cause a power-spectrum to be computed.

B. Complete Example

1. In this section we present a very simple, but complete example of the XSIL system, with the XML, the Object code, and the Viewer code. The Label is defined in XML like this:

```
<?xml version="1.0"?>
<!DOCTYPE XSIL SYSTEM "xsil.dtd">
<XSIL>
<XSIL Type="Simple.Label" Name="Example">
    <Param Name="Message">Hello Auntie Joan</Param>
    <Param Name="FontSize">96</Param>
</XSIL>
</XSIL>
```

So the label object is just a piece of text (Message) and an integer (FontSize).

2. Code to read this object looks like this. The children are examined to find parameters that can supply the message and fontsize fields.

```
package extensions.Simple;
import org.escience.XSIL.*;

public class Label extends XSIL {
    public String message = "No message";
    public int fontsize = 12;
```

8. The DTD for XSIL

1. An XML file may be associated with a Document Type Definition (DTD) which defines the allowed tag names in the document, and how these fit together: which elements may contain which other elements, and how many of each element there may be.

```
<!ELEMENT XSIL ((XSIL|Comment|Url|Param|Table|Array|Stream)*)>
<!ATTLIST XSIL Name CDATA "" Type CDATA "">

<!ELEMENT Comment (#PCDATA)>

<!ELEMENT Param (#PCDATA)>
<!ATTLIST Param Name CDATA "" Type CDATA "" Unit    CDATA "" >

<!ELEMENT Url (#PCDATA)>
<!ATTLIST Url Name CDATA "" Type CDATA "" href    CDATA "" >

<!ELEMENT Array (Dim* , Stream?)>
<!ATTLIST Array Name CDATA "" Type CDATA "" Unit CDATA "">
<!ELEMENT Dim (#PCDATA)>
<!ATTLIST Dim Name CDATA "" Type "">

<!ELEMENT Table (Column* , Stream?)>
<!ATTLIST Table Name CDATA "" Type CDATA "">
<!ELEMENT Column EMPTY>
<!ATTLIST Column Name CDATA "" Type CDATA "" Unit CDATA "">

<!ELEMENT Stream(#PCDATA)>
<!ATTLIST Stream Name CDATA "" Type CDATA "" >
<!ATTLIST Stream Content CDATA ""Encoding CDATA "" Delimiter CDATA "">
```

9. Installation and Use

1. The installation is available as a zip file from

- `http://www.cacr.caltech.edu/XSIL/`

by following the link "available software".

2. Javadoc documentation comes with the distribution, or else from

- `http://www.cacr.caltech.edu/XSIL/javadoc/`

3. The XSIL environment has been tested on Solaris and Windows 98 and NT.

4. To run the XSIL environment, you will need:

• A computer with Java Development Environment (JDK) 1.2 or later. You will need to know the directory where it is installed, meaning the directory which has the /bin and /lib subdirectories. For more information and free download, see

`http://java.sun.com/products/jdk/1.2/`

• One of the demonstration codes utilizes the Java3D package. If you wish to run this demo, you will need to install Java3D, which is available free from:

`http://java.sun.com/products/java-media/3D/index.html`

The Java3D is implemented with OpenGL, so it will probably find your fast graphics card. ***NOTE:*** Installation is much more difficult if you choose to put Java3D in a different directory from that suggested by the install wizard.

• You must also have an unzip or untar facility.

5. Unpack the distribution, and go to the XSIL root directory, which is the one that contains com, org, doc, and extensions directories. There should also be some scripts here, labelled `.bat` for Windows, `.sh` and `.source` for Unix.

6. Make sure the `JAVA_HOME` and `WEB_BROWSER` locations are correct in the setup.bat (Windows) or setup.source (Unix) script.

• The web browser is not necessary unless you intend to use the <URL> tag, which spawns a browser for viewing.

7. Start a command window and run the setup script. Type `setup` (for Windows) or `source setup.source` (for Unix). Make sure the Java interpreter is in your path: type `java`, it should give a usage message rather than command not found.

8. For Unix users,

```
% source setup.source
% xlook.sh samples/all.xml
```

9. For Windows users,

```
c: setup
c: xlook samples\all.xml
```

10. The browser should now come up, and there are a number of sample in the samples directory.

11. ***NOTE to DEVELOPERS.*** The key to the Java package system, how it finds classes, how it compiles, making life easy, requires you to follow one rule: always run your compile (`javac`) and applications (`java`) from the XSIL home directory, the one with demo scripts.

12. You can also examine and modify the code in the `extensions` directory. To make a custom viewer, a good start might be a copy of the `extensions/Simple` directory.

付録２．３　UDF改訂仕様書 Version 2.0

名大　土井プロジェクト　資料より

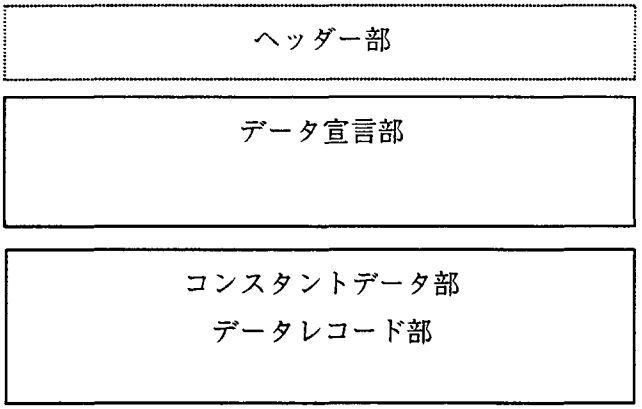
１．はじめに

本書は汎用技術計算データ書式(UDF)の基本文法および規約を以下のように規定するものである。UDF は科学技術計算で用いられるすべてのデータの記述に用いられ、この書式で記述されたデータは高機能材料プラットフォーム上ではいかなるデータも読み書きできるように考えられている。

本書の草案は UDF 仕様（1999.05.18）をもとに、これをより汎用的に記述するために、部分的に改訂したものである。

２．UDF の構造

UDF の構造は下図に示すようになっている。



２．１　ヘッダー部

UDF インポート時に生成される UDF 登録名データを記述する。

書式は以下の通り。

```
¥begin{header}
¥begin{def}
        EngineType:string;
        EngineVersion:string;
        IOType:string;
        ProjectName:string;
¥end{def}
¥begin{data}
        EngineType:"EngineTypeName"
        EngineVersion:"EngineVersionName"
        IOType:"IN_OUT"
        ProjectName:"ProjectNameString"
¥end{data}
```

¥end{header}

このデータにより、UDF 登録名は以下のようになる。*file_name* は UDF インポートファイル名である。

EngineTypeName.EngineVersionName.IN_OUT.ProjectNameString.file_name

ヘッダー部が無い場合は、*file_name* が UDF 登録名となる。

2．2　データ宣言部

データ定義を記述する。データ定義部とコンスタントデータ部（次節参照）は混在していても良いが，１つのデータについてはデータ定義部がデータ実体部に必ず先行していなければならない.

データ宣言部にはコンスタントデータ部およびデータレコード部で使用するすべてのデータの宣言を記述しなければならない。

2．3　コンスタントデータ部

コンスタントデータ部を記述する。データ定義部とコンスタントデータ部は混在していても良いが，１つのデータについてはデータ定義部がデータ実体部に必ず先行していなければならない.

ここにコンスタントデータ部とは解析計算中は解析ステップが異なっても変化しない変数（すなわちコンスタント）のデータ実体部をいう。

2．4　データレコード部

レコード部にはデータ実体部のタイムステップ毎のデータを記述する。

レコード部にはデータ定義部を置くことはできない。

エンジンインターフェイスライブラリからデータレコード部を操作する時は、１レコード目をゼロ番目ステップとしてアクセスする。

¥begin{record} {"*Record_Label*"}
¥begin{data}

　　DATA ...

¥end{data}
¥end{record}

3．　インクルード機能（データ宣言部およびコンスタントデータ部に限定）

データ宣言部およびコンスタントデータ部については他の UDF ファイルのデータ宣言部およびコンスタントデータ部を埋め込むことができる。

但し埋め込む UDF ファイルにデータレコード部があってはならない。

インクルード書式

¥include{"include_UDF_name"}

・"¥"は１桁目にあること。
・"include"は小文字のみ。
・"¥","include","{","include_UDF_name","}"の各トークンの間には空白が入っても良い。
・呼び出されるファイル名は最初の指定ファイルからの相対パスあるいは絶対パスで指定する。

4．ユーザー定義型とコンポーネント機構

4．1　ユーザー定義型

UDF のデータ宣言部において、構造体の中に定義されるデータが非常に多い場合や深い構造体になっている場合、データ定義が複雑になり読みづらくなる。それに対応するために、構造体の宣言をユーザー定義型として使用できるようにした。

ユーザー定義型として宣言された変数は、ユーザー定義型が持つ構成要素（変数）を下位に持つ構造体変数となる。

制限事項としては、

（1）構造体内の１変数としてユーザー定義型を使える。
（2）配列型のユーザー定義型は使えない。

```
¥begin{def}
Atom:{
        molID:int,              // 分子内の ID
        typeName:string,        // AtomType の名前
        coord:{                 // 座標
                x:double,
                y:double,
                z:double
        }
        vel:{                   // 速度
                x:double,
                y:double,
                z:double
```

```
        }
        force:{                 // 力
                x:double,
                y:double,
                z:double
        }
        totalID:int,            // システムトータルの ID
};
Atom.show["point"]:{"point(coord,1)"};
¥end{def}

・・・省略・・・

¥begin{component}
Molecule:{
        name:string,
        atom[]:Atom,
        bond[]:Bond,
        angle[]:Angle,
        torsion[]:Torsion,
        interactionSite[]:InteractionSite
};
¥end{component}
```

4．2　コンポーネント機構

ユーザー定義型変数に対して show・test・unit・help の UDF 属性が設定されている場合、その属性を上位構造体に継承させるのがコンポーネント機構である。

上記の

Atom.show["point"]:{"point(coord,1)"};

では、自動的に

Molecule.atom[].show["point"]:{"point(coord,1)"};

の show 属性が設定される。

5．UDF 文法

UDF はデータ定義部とデータ実体部からなる．

データ定義部

データ定義部は名前付けを行うためのデータ名定義部とデータ属性定義部からなる．

データ名定義部はデータに一意的な名前付けを行うために必要な部分である．

データ属性定義部は表示や、ヘルプの要請に応えるために必要な部分である．

データ名定義部は必ず必要であるが、データ属性定義部はなくとも良い．

データ実体部

データ実体部には数値データまたは文字列データをスペース，空白，改行，";","}"などの

区切り記号で区切ってかく．文字列データは必ず引用符(" または ') で括る．
データ定義部とデータ実体部は一つのストリーム中に混在していても良いが，ある名前のデータについてはデータ定義部がデータ実体部に必ず先行していなければならない．

5．1 予約語

(1)定義文

データ定義に以下の予約語を用いる。

¥def

１行でデータの定義が完了する場合のキーワード

¥begin {def}

構造体などの複合データを定義開始キーワード

¥end {def}

構造体などの複合データを定義終了ワード

(2)データ型

データの型を宣言するために以下のキーワードを用いる。

int

データが長整数であることを宣言する場合に用いる。(long と同じ)

long

データが長整数であることを宣言する場合に用いる。(int と同じ)

short

データが短整数であることを宣言する場合に用いる。

float

データが短精度実数であることを宣言する場合に用いる。(single と同じ)

double

データが短精度実数であることを宣言する場合に用いる。

single

データが短精度実数であることを宣言する場合に用いる。(float と同じ)

string

データが文字型であることを宣言する場合に用いる。

(3)区切り記号

構造体・配列などの定義区切りを示すために以下のキーワードを用いる。

{}

データが構造体であることを宣言する場合および
構造体のスコープ（定義範囲）を明示する場合に用いる。

[]

データが配列である事を宣言する場合および
データ配列のスコープ（定義範囲）を明示する場合に用いる。

<>

データ定義部ではポインターである事を宣言する場合に用いる。
データ実体部ではどの変数のポインターであるかを明示する場合に用いる。

;

データ定義の終了を明示する場合に用いる。

:

データ定義部ではデータ定義区切り子として用いる。
データ実体部ではデータ値区切り子として用いる。

""

データ実体部で文字データの設定に用いる。

(4)Platform に依存した予約語

platform では Python による簡易計算をサポートしているため以下のキーワードは使用できない。

and	del	for	is	raise
assert	elif	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	def
finally	in	print		

UDF fileとPythonのinterfaceとして以下のキーワードが予約されている。

UDFPythonParse	UDF	val

また、UDF変数の配列[]内ではPythonのリストに使用できるスライス演算子「：」を使用することはできない。UDFデータ変数の「配列」は「Pythonのリスト」とは別物である。

UDF 変数については、以下に示すような多重代入操作によりデータを変更することには対応しないこととする。

a, b, c= x, y, z= y, x, x +y

以下のように空白を含むUDF変数はない（UDFの定義より、2変数となる）ものとする。

atm position ->atom?positionなどとする。

c, c++をエンジンの開発言語としている場合は c,c++の予約語である以下のキーワー

ドは使用できない。

asm	auto	break	case	catch
char	class	const	continue	default
delete	do	double	else	enum
extern	float	for	friend	goto
if	inline	int	long	new
operator	private	protected	public	register
return	short	signed	sizeof	static
struct	switch	template	this	throw
try	typedef	union	unsigned	virtual
void	volatile	while		

FORTRAN をエンジンの開発言語としている場合は FORTRAN の予約語であるキーワードは使用できない。(FORTRAN 文法書参照)

5．2 データ定義部規約

(1) 単純データのデータ定義

1つの変数や単純な構造体のような1ラインでその定義の記述ができる場合は以下のように¥def 文を用いて、データの定義を行うことができる。

¥def　*Variable Name*: *Type*;

ここに、*Variable Name* は a-Z で始まる任意の変数名で、*Type* はデータの型である。

¥def の最後には必ず;をつける。

(例)

```
¥def   temperature: double;
¥def   vec:| x:int, y:int, z:int |;
```

(2) 一般データのデータ定義

一般データのデータ定義は以下のように¥begin|def| 文を用いて、データの定義を行うこととする。

¥begin|def|

Variable Name: *Type*

Variable Name: *Type*;

¥end|def|

ここに、*Variable Name* は a-Z で始まる任意の変数名で、*Type* はデータの型である。

¥begin|def| の最後は必ず end|def| で終了する。

(例)

```
¥begin{def}
   atom:{ pos: { x:int, y:int, z:int },
          vel: { x:int, y:int, z:int },
          id: int
          kind:string
        };
end{def}
```

(2) 配列データのデータ定義

配列データのデータ定義は以下のように[]を用いて、データの定義を行うこととする。

Variable Name[]: Type

(例)

```
¥begin{def}
 config:{
   time:double,
   atom[]:{pos[]:{int},
           vel:{x:int,y:int,z:int},
           id:int
           }
   kinetic_energy:int,
   pressure:int
   };
¥end{def}
```

(3) ポインタ変数のデータ定義

ポインタ変数のデータ定義は以下のように<>を用いて、データの定義を行うこととする。

Variable Name<>;*

```
¥begin{def}
   p1:<*>;
   p2:<*>;
   p3:<*>;
   x:double;
   vec:{x:int,y:int,z:int};
   atom:{
           vec:{x:int,y:int,z:int},
```

```
        id:int
    |;
¥end|def|
```

(4) データ属性のデータ定義

データ属性のデータ定義は以下のように.help, .test, .unit, .showを用いて、データの定義を行うこととする。

Variable Name.help: "Expressions";

Variable Name.test: "Expressions";

Variable Name.unit: "Expressions";

Variable Name.show: "Expressions";

これらはデータの情報表示(help)、範囲チェック(test)、単位指定(unit)、図示指定(show)などを行うもので、*Expressions*中には任意の文字列を記述することができる。

また、showでは*Expressions*中に属性IDを用いることができ、これにより、データの表示法を簡単に変更することが可能である。

5．3 データ実体部規約

(1) 単純データのデータ実体部

1つの変数や単純な構造体のような1ラインでその定義の記述ができる場合は以下のように¥data文を用いて、データの定義を行うことができる。

¥data *Variable Name: Value;*

(例)

```
¥def   temperature: double;
¥data   temperature: 1.0;
```

(2) 一般データのデータ実体部

一般データのデータ定義は以下のように¥begin|data|文を用いて、データの定義を行うこととする。

```
¥begin|data|
    Variable Name: Value
    Variable Name: Value;
¥end|data|
```

(例)

```
¥begin|def|
 config:|
```

```
    time:double,
    atom[]:{pos[]:{int},
            vel:{x:int,y:int,z:int},
            id:int
            }
    kinetic_energy:int,
    pressure:int
    };
¥end{def}

¥begin{data}
    config:{
    time    10.0
    atom    [
                {[10,11,12],{-10,-11,-12},15},
                {[20,21,22],{-20,-21,-22},25},
                {[30,31,32],{-30,-31,-32},35}
            ]
            40
            5
    };
¥end{data}
```

（注）

ここで、配列 pos の実体は[]で区切り、構造体 vel の実体は{}で区切っている。

オリジナル仕様ではすべて{}でデータを区切っていたが、それでは変数が構造体か配列かの区別がデータ実体部で不可能であるため、UDF 定義部と実体部がファイル内で大きく離れている場合などにデータの定義があいまいになったり、エラー処理があいまいになることを避けるためである。

(3) ポインタ変数の利用法

ポインタ変数の利用法は以下の通りである。

Variable Name< Variable Name 2>;

```
¥begin{def}
    p1:<*>;
    p2:<*>;
    p3:<*>;
```

```
    x:double;
    vec:{x:int,y:int,z:int};
    atom:{
            vec:{x:int,y:int,z:int},
            id:int
};
¥begin{data}
    p1:<x>;
    p2:<vec>;
    p3:<atom.vec>;
    x:1.0;
    vec:{1,4,5};
    atom:{{3,4,5},7};
¥end{data}
```

この場合

```
p1 = 1.0
p2.x = 1
p3.x = 3
```

（注）

ここではポインタ指定に変数名を用いている。

オリジナル仕様では番号を用いることになっていたが、その場合あるデータの中で注意深く番号付けをしなければ、意図した変数を指し示すことができないからである。

また、配列の場合は以下のような使用法が可能である。

・配列の扱い

```
¥begin{def}
    p4:<*>;
    p5:<*>;
    array[]:{int};
¥end{def}

¥begin{data}
    p4:<array[2]>;
    p5:<array>;
    array[]:[0,1,2,3,4,5];
¥end{data}
```

上記の例の p4 は変数 array の 3 番目の要素 array[2]のポインタを指し、

p5 は変数 array のポインタを指しているので、実体は以下のようになる。

```
p4 = 2
p5[1] = 1
```

(4) データ属性のデータ実体

データ属性のデータ実体は show 属性以外は現在検討中で確定していない。この理由は Python とのインターフェースと関連して、この仕様を決定する必要があるからである。よってここでは show に限って解説する。

a) show 属性の書式

a.1 show 属性の書式

Variable Name.show["label"]:{"Expressions"}

UDF データの図示指定(show)を行うもので、*Expressions* 中には以下に示す Python 描画関数を含む Python 文、または他の show 属性式を記述することができる。

また、*Expressions* 中の変数で、*Variable Name* 構造体の成分として宣言された変数や show 属性は、先頭にピリオド (.) を付けることにより、*Variable Name* の記述を省略できる。

(簡単な show 属性の例)

```
Atom:{
        molID:int,              // 分子内の ID
        typeName:string,// AtomType の名前
        coord:Vector3d          // 座標
        vel:Vector3d            // 速度
        force:Vector3d          // 力
        totalID:int,            // システムトータルの ID
};
Atom.show["ball"]:{"sphere(.coord,2)"}

Bond:{
        typeName:string,// bondType の名前
        atom1:<*>,              // 両端の Atom のポインタ
        atom2:<*>               // (当然 Atom.id を持ってもよい。
};
Bond.show["stick"]:{"cylinder(<.atom1>.coord,<.atom2>.coord,6)"};

¥begin{component}
TotalMolecules:{
        molecule[]:{
                name:string,
                atom[]:Atom,
                bond[]:Bond,
```

```
                angle[]:Angle,
                torsion[]:Torsion,
                interactionSite[]:InteractionSite
        }
};
TotalMolecules.molecule[].show["ball-stick_c"]:{
        '.bond[].show["stick"]'
        '.atom[].show["ball"]'
}
¥end{component}
```

a.2 Python 描画関数

上記の１．show 属性の *"Expressions"* 部や Java クライアント３D描画画面の Python 入力エリアに記述できる描画関数を以下に示す。引数の coordinate*N* は、[x, y, z]値を持つ Python リスト、または (x, y, z) を成分としてもつ UDF 構造体変数である。
coordinate_list は、３次元座標[x, y, z]の Python リストをさらにリストに集めたものである。attribute_id は属性指定値または、下記の４．描画属性に示す描画属性値を直接 Python リストにまとめたものである。

・線

```
line( coordinate1, coordinate2, attribute_id )
```

・点

```
point( coordinate1, attribute_id )
```

・四面体

```
tetra( coordinate1, coordinate2, coordinate3, coordinate4, attribute_id )
```

・多面体

```
polygon( coordinate_list, attribute_id )
```

・ポリライン

```
polyline( coordinate_list, attribute_id )
```

・円

```
disk( coordinate1, attribute_id )
```

・楕円

```
ellipse1( coordinate1, attribute_id )
ellipse2( coordinate1, coordinate2, attribute_id )
```

・円柱

```
cylinder( coordinate1, coordinate2, attribute_id )
```

・球

```
sphere( coordinate1, attribute_id )
```

・楕円体

ellipsoid1(coordinate1, attribute_id)

ellipsoid2(coordinate1, coordinate2, attribute_id)

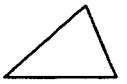
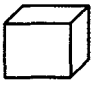
・矢印　(矢は coordinate2 の方に付く)

arrow(coordinate1, coordinate2, attribute_id)

・コンター (断面、等値面)

contour(element_type, node_number, node_coordinate_list, node_value_list)

コンター図を描く対象の 1 要素形状を設定する。

要素	element_type		node_number
三角形	"triangle"		3
四角形	"quad"		4
四面体	"tetra"		4
六面体	"hexa"		8

断面の指定

cplane(point_on_plane, normal_vector, attribute_id)

contour で指定した形状の断面等高図を描く。

等値面レベルの指定

clevel(min_value, max_value, attribute_id)

contour で指定した形状に min_value と max_value の間にある値を持つ点をつないだ等値面を描く。

a.3 補助ツール関数

・配列サイズ取得

sizeof(UDF_data_name)

UDF_data_name の最右端配列にある要素数を返す。
下記例では hexa[10]の下にある gid[]の要素数を返す。
hexa[10]の 10 が指定されていないときは、サイズが特定できないので None を返す。

(例)
elesize=sizeof(hexa[10].gid[])

・インデックスマップ生成
indexmap(UDF_data_name)

UDF_data_name の値と配列インデックスのマップを生成し、マップオブジェクトを Python オブジェクトとして返す。
値からインデックスを取得するには、find メソッドを使用する。
find メソッドが返す値は UDF_data_name の配列ｲﾝﾃﾞｯｸｽの Python リストを検索数だけリスト化したものである。
存在しない値を find で指定したときは、None を返す。

(例)
GridData[2].grid[3].gid 及び GridData[3].grid[4].gid の値が 6 である時、
imap=indexmap(GridData[].grid[].gid)
idlist=imap.find(6)
とすると、idlist[0][0]=2、idlist[0][1]=3 及び idlist[1][0]=3、idlist[1][1]=3 である。
検索数は、len(idlist)で得られる。

a.4 描画属性

a.4.1 描画属性ファイル

上記の描画関数に、データの表示法を描画属性 id で渡す。これにより、描画方法を簡単に変更することが可能である。
描画属性のデフォルト設定ファイルは Java サーバーの作業ディレクトリに置く。描画属性ファイルの名称は"*ShapeDrawingAttr.txt*"である。
デフォルト描画属性ファイルの書式例を下記に示す。

```
### Shape Drawing Attribute ###
LineAttr    <- Shape Key
```

```
# ID   R   G   B  TRANS
   1  1.0 0.0 0.0 1.0
   2  0.0 1.0 0.0 1.0
   3  0.0 0.0 1.0 1.0
   4  1.0 1.0 0.0 1.0
   5  1.0 0.0 1.0 1.0
   6  0.0 1.0 1.0 1.0
   7  0.0 0.0 0.0 1.0
######################
PointAttr
# ID   R   G   B  TRANS
   1  1.0 0.0 0.0 1.0
   2  0.0 1.0 0.0 1.0
   3  0.0 0.0 1.0 1.0
   4  1.0 1.0 0.0 1.0
   5  1.0 0.0 1.0 1.0
   6  0.0 1.0 1.0 1.0
   7  0.0 0.0 0.0 1.0
```

LineAttr 及び PointAttr は、それぞれ線分と点の描画属性指定キーワードである。

キーワードに続く行に、それぞれの描画属性を記述する。＃で始まる行はコメントとみなす。

描画属性指定キーワードには、下記の（1）に示すものがあり、それぞれの描画属性は（2）に示すとおりである。

（1）キーワード

- LineAttr　：線分
- PointAttr　：点
- PolygonAttr ：三角形、四角形、・・多角形
- DiskAttr　：円盤
- EllipselAttr：楕円盤 1（中心 1 点の座標指定）
- Ellipse2Attr：楕円盤 2（焦点 2 点の座標指定）
- CylinderAttr：円柱
- SphereAttr　：球
- EllipsoidlAttr　：楕円体 1（中心 1 点の座標指定）
- Ellipsoid2Attr　：楕円体 2（焦点 2 点の座標指定、長軸 2a）
- TetraAttr　：四面体
- ArrowAttr　：矢印
- CplaneAttr ：断面コンター
- ClevelAttr　：等値面

（2）描画属性

色（RGB、透明度）は、0.0～1.0で指定する。

- □ LineAttr　：ＩＤ、色（RGB、透明度）
- □ PointAttr　：ＩＤ、色（RGB、透明度）
- □ PolygonAttr：ＩＤ、色（RGB、透明度）
- □ DiskAttr　：ＩＤ、色（RGB、透明度）、半径、法線ベクトル(XYZ)
- □ EllipselAttr：ＩＤ、色（RGB、透明度）、径長(a,b)、径ａ方向ベクトル(XYZ)、法線ベクトル(XYZ)
- □ Ellipse2Attr：ＩＤ、色（RGB、透明度）、径長(a)、法線ベクトル(XYZ)
- □ CylinderAttr：ＩＤ、色（RGB、透明度）、半径
- □ SphereAttr　：ＩＤ、色（RGB、透明度）、半径
- □ EllipsoidlAttr　：ＩＤ、色(RGB、透明度)、径長(a,b,c)、径ａ方向ベクトル(XYZ)、径ｃ方向 ベクトル(XYZ)
- □ Ellipsoid2Attr　：ＩＤ、色（RGB、透明度）、径長(a)
- □ TetraAttr　：ＩＤ、色（RGB、透明度）
- □ ArrowAttr　：ＩＤ、色（RGB、透明度）、矢半径、矢高さ
- □ CplaneAttr：ＩＤ、最小値色（RGB、透明度）、最大値色（RGB、透明度）
- □ ClevelAttr　：ＩＤ、色（RGB、透明度）

（３）規約

①データ値の間は空白で区切る。

②１桁目が”＃”の行はコメント行。

③ファイルが無い時は、デフォルト値を使用する。

④同じＩＤ指定がある場合は、後の方に書かれた属性値が採用される。

a.4.2 UDF での描画属性データ指定

UDF のデータ宣言部に以下の書式で描画属性データを設定すると、その描画属性データが優先的に使用される。

```
¥begin{def}
LineAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
};
PointAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
};
PolygonAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
```

```
};
DiskAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
        radius:float,
        vertical:{ x:float, y:float, z:float }
};
Ellipse2Attr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
        radius_a:float,
        vertical:{ x:float, y:float, z:float }
};
CylinderAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
        radius:float
}
SphereAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
        radius:float
};
Ellipsoid2Attr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
        radius_a:float,
};
TetraAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
};
ArrowAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
        hat_radius:float,
        hat_height:float
};
CplaneAttr[]:{
        id:int,
        min_color:{ r:float, g:float, b:float, t:float }
        max_color:{ r:float, g:float, b:float, t:float }
};
ClevelAttr[]:{
        id:int,
        color:{ r:float, g:float, b:float, t:float }
};
¥end{def}
¥begin{data}
LineAttr[]:[
        { 1, { 1.0 1.0 0.0 1.0 } }
        { 2, { 1.0 0.0 1.0 1.0 } }
        { 1, { 0.0 1.0 1.0 1.0 } }
]
```

```
PointAttr[]:[
        | 1, | 0.0 1.0 0.0 1.0 | |
]
PolygonAttr[]:[
        | 1, | 1.0 0.0 0.0 1.0 | |
]
DiskAttr[]:[
        | 1, | 1.0 1.0 0.0 1.0 | 0.5 | 1 0 0 | |
]
Ellipse2Attr[]:[
        | 6, | 1.0 0.0 0.0 1.0 | 0.8 | 1 0 0 | |
];
CylinderAttr[]:[
        | 1, | 1.0 0.0 0.0 1.0 | 0.5 |
]
SphereAttr[]:[
        | 1, | 1.0 0.0 0.0 1.0 | 0.5 |
];
Ellipsoid2Attr[]:[
        | 1, | 1.0 0.0 0.0 1.0 | 0.5 |
        | 5, | 0.0 0.0 1.0 1.0 | 0.5 |
        | 5, | 1.0 0.0 0.0 1.0 | 0.5 |
];
TetraAttr[]:[
        | 1, | 1.0 0.0 0.0 1.0 | |
];
ArrowAttr[]:[
        | 1, | 1.0 0.0 0.0 1.0 | 0.5 0.5 |
];
CplaneAttr[]:[
        | 1, | 0.0 0.0 1.0 1.0 | | 1.0 0.0 1.0 1.0| |
];
ClevelAttr[]:[
        | 1, | 1.0 1.0 0.0 1.0 | |
];
¥end{data}
```

a.4.3 描画関数での直接指定

描画属性を直接、Python 描画関数に渡す方法として、上記 a.4.1（2）の描画属性のＩＤを除いた数値を Python リストにまとめて描画関数に渡すことができる。

（例） line([1,1,1],[2,2,2],[1,0,0,1])

a.5 描画属性の使用例

```
¥begin{header}
¥begin{def}
        EngineType:string;
```

```
        EngineVersion:string;
        IOType:string;
        ProjectName:string;
¥end{def}
¥begin{data}
        EngineType:"MSC_NASTRAN"
        EngineVersion:"V01"
        IOType:"IN"
        ProjectName:"RelationSample"
¥end{data}
¥end{header}

¥begin{def}
Coord: {
        x: double,
        y: double,
        z: double,
};
Grid: {
        gid: int,
        coord: Coord
        // User DATA.
        value1: double
        value2: double
        value3: double
};
Element: {
        eid: int,
        gid[]: int
};
Material: {
        mid: double,
        e: double,
        g: double,
        nu: double
};
¥end{def}

¥begin{component}
GridData: {
        grid[]: Grid
};

ElementsData: {
        hexa[]: Element
        mat1:Material
};

ElementsData.show["flame"]:}'
# Creating index map.
```

```
imap=indexmap(GridData.grid[].gid)
# Getting size of "ElementsData".
elesize=sizeof(ElementsData.hexa[])
for n in range(0,elesize):
        pt=[]
        for i in range(0,8):
                idlist=imap.find(.hexa[n].gid[i])
                if(idlist is None):
                        raise "no index"
                id=idlist[0][0]

        pp=[GridData.grid[id].coord.x,GridData.grid[id].coord.y,GridData.grid[id
].coord.z]
                pt.append(pp)
        line(pt[0],pt[1],1)
        line(pt[1],pt[2],1)
        line(pt[2],pt[3],1)
        line(pt[3],pt[0],1)
        line(pt[0],pt[4],1)
        line(pt[1],pt[5],1)
        line(pt[2],pt[6],1)
        line(pt[3],pt[7],1)
        line(pt[4],pt[5],1)
        line(pt[5],pt[6],1)
        line(pt[6],pt[7],1)
        line(pt[7],pt[4],1)
'}

ElementsData.show["surface"]:{'
# Creating index map.
imap=indexmap(GridData.grid[].gid)
# Getting size of "ElementsData".
elesize=sizeof(ElementsData.hexa[])
for n in range(0,elesize):
        pt=[]
        vd=[]
        for i in range(0,8):
                idlist=imap.find(.hexa[n].gid[i])
                if(idlist is None):
                        raise "no index"
                id=idlist[0][0]

        pp=[GridData.grid[id].coord.x,GridData.grid[id].coord.y,GridData.grid[id
].coord.z]
                pt.append(pp)
                vd.append(GridData.grid[id].value1)
        contour("hexa",8,pt,vd)
#contour level
clevel(7.5,8.5,1)
'}
```

```
ElementsData.show["contour"]:{'
# Creating index map.
imap=indexmap(GridData.grid[].gid)
# Getting size of "ElementsData".
elesize=sizeof(ElementsData.hexa[])
for n in range(0,elesize):
        pt=[]
        vd=[]
        for i in range(0,8):
                idlist=imap.find(.hexa[n].gid[i])
                if(idlist is None):
                        raise "no index"
                id=idlist[0][0]

        pp=[GridData.grid[id].coord.x,GridData.grid[id].coord.y,GridData.grid[id
].coord.z]
                pt.append(pp)
                vd.append(GridData.grid[id].value1)
        contour("hexa",8,pt,vd)
#contour level
cplane([1,1,1],[1,1,1],2)
'}

¥end{component}

¥begin{record} {"step 1"]
¥begin{data}
GridData :{
[
        {0, {0,0,0},0,0,0},
        {1, {1,0,0},1,0,0},
        {2, {2,0,0},2,0,0},
        {3, {0,1,0},3,0,0},
        {4, {1,1,0},4,0,0},
        {5, {2,1,0},5,0,0},
        {6, {0,2,0},6,0,0},
        {7, {1,2,0},7,0,0},
        {8, {2,2,0},8,0,0},
        {9, {0,0,1},8,0,0},
        {10, {1,0,1},8,0,0},
        {11, {2,0,1},8,0,0},
        {12, {0,1,1},8,0,0},
        {13, {1,1,1},8,0,0},
        {14, {2,1,1},8,0,0},
        {15, {0,2,1},9,0,0},
        {16, {1,2,1},9,0,0},
        {17, {2,2,1},9,0,0},
        {18, {0,0,2},9,0,0},
        {19, {1,0,2},9,0,0},
        {20, {2,0,2},9,0,0},
```

```
        |21, |0,1,2|,10,0,0|,
        |22, |1,1,2|,10,0,0|,
        |23, |2,1,2|,10,0,0|,
        |24, |0,2,2|,10,0,0|,
        |25, |1,2,2|,10,0,0|,
        |26, |2,2,2|,10,0,0|,
]
|;

ElementsData: |
        [
                | 0, [0,1,4,3,9,10,13,12] |
                | 1, [1,2,5,4,10,11,14,13] |
                | 2, [3,4,7,6,12,13,16,15] |
                | 3, [4,5,8,7,13,14,17,16] |
                | 4, [9,10,13,12,18,19,22,21] |
                | 5, [10,11,14,13,19,20,23,22] |
                | 6, [12,13,16,15,21,22,25,24] |
                | 7, [13,14,17,16,22,23,26,25] |
        ]
        | 0, 21000, 0.0 0.0 |
|;
¥end|data|
¥end|record|
```

(5) 配列の記述順序指定子(IndexOrder)

データが2次元以上の配列である場合どちらの次元を先に read/write するかを規定しなければ、UDF は自己完結とならない。そこで、UDF ファイル内にこの順序を規定する配列の記述順序指定子を設けることにした。

この配列の記述順序指定子はデータの実体部の状態を示すものであるので、データ実体部で何度でも記述できるものと考えた。

¥indexOrder | *Variable Name* [i][j]|

(例)

```
¥begin|def|
 array:|
   array2[][]:||int||,
   array3[][][]:|||int|||
   |;
¥end|def|
¥begin|data|
¥indexOrder |
            array.array3[1][2][3],
            array.array2[2][1]
```

```
    }
    array:{
            [
            [10,11,12,13,14],
            [20,21,22,23,24],
            [30,31,32,33,34],
            [40,41,42,43,44]
            ],
            [
            [
            [110,111,112,113,114],
            [120,121,122,123,124],
            [130,131,132,133,134],
            [140,141,142,143,144]
            ],
            [
            [210,211,212,213,214],
            [220,221,222,223,224],
            [230,231,232,233,234],
            [240,241,242,243,244]
            ],
            [
            [310,311,312,313,314],
            [320,321,322,323,324],
            [330,331,332,333,334],
            [340,341,342,343,344]
            ]
            ]
    };
¥end{data}
```

付録３　ネットワーク分散環境＆シミュレータの現状

収録資料

付録３．１　ＡＭＯ

Web Site http://www.yl.is.s.u-tokyo.ac.jp/amo/より

付録３．２　JavaGo

Web Site http://www.yl.is.s.u-tokyo.ac.jp/amo/JavaGo/doc/index.html
より

付録３．３　NASTRAN

Web Site http://www.engineering-e.com/software/index.cfmより

付録３．１　ＡＭＯ

Web Site http://www.yl.is.s.u-tokyo.ac.jp/amo/より

付録３．２　JavaGo

Web Site http://www.yl.is.s.u-tokyo.ac.jp/amo/JavaGo/doc/index.html
より

```
public class Contact implements java.io.Serializable {
    public static String MessageBoard = "";
    public static void gmain(String args[]) {
        try {
            undock {
                go ("//flute:12002/");
                MessageBoard = "mountain";
                go ("//ritsuko:12002/");
                MessageBoard = "river";
                go ("//flute:12002/");
                System.out.println ("MessageBoard = " + MessageBoard);
                go ("//ritsuko:12002/");
                System.out.println ("MessageBoard = " + MessageBoard);
            }
        } catch (Exception e) {
            System.out.println ("migration failed!");
            System.out.println (e.getMessage());
        }
    }
}
```

このプログラムは MessageBoard という変数に二回別の値を代入している。もしマイグレーションを行わないプログラムであれば意味のないことである。このプログラムは flute の MessageBoard に mountain と書き込み、その後 ritsuko に移動し、その場所の MessageBoard に river と書き込む。そしてそれぞれのホストで MessageBoard の内容を表示する。

プログラムのコンパイル

JavaGO の移動方式では、マイグレーションを行うプログラムは専用の translator (jgoc) で処理をする必要がある。jgoc は go や undock を含んだプログラムを入力として受け取り、標準的な Java のプログラムを出力する。出力された Java のプログラムは普通の Java のコンパイラを使ってコンパイルすることができる。

上述されているプログラムを Contact.jgo というファイルに保存をしておく。それを jgoc と javac でコンパイルする。

```
jgoc Contact.jgo
javac Contact.java
```

プログラムの起動

プログラムを起動するためには、このプログラムを構成しているすべてのクラスファイルを migration server に転送する必要がある。その処理は JavaGo パッケージに含まれている javago.Run というプログラムが行う。

```
java javago.Run //flute:12002/ Contact.class ST*.class
```

このように入力すると flute には mountain、ritsuko には riverと 表示されるはずである。

```
                int maxx = 260 + Size * 4 + (j - 1) * 170;
                int y = 192 - i * 8;
                offgraphics.fillRect( minx, y, maxx - minx, 6 );
            }
        }
    }
    g.drawImage(offscreen, 0, 0, null);
}
public void Move( int From, int To ) {
    int     i, j;
    for ( i = Size - 1; i >= 0; i-- )
        if ( Tower[From][i] != 0 )
            break;
    for ( j = 0; j < Size; j++ )
        if ( Tower[To][j] == 0 )
            break;
    Tower[To][j] = Tower[From][i];
    Tower[From][i] = 0;
    repaint();
    try {
        Thread.currentThread().sleep (30);
    }
    catch ( InterruptedException e ) {
    }
}
public int TheRestOne( int i, int j ) {
    if ( i == 0 ) {
        if ( j == 1 )
            return 2;
        else
            return 1;
    }
    else if ( i == 1 ) {
        if ( j == 0 )
            return 2;
        else
            return 0;
    }
    else {
        if ( j == 0 )
            return 1;
        else
            return 0;
    }
}
public void Solve( int k, int From, int To ) {
    if ( k == 1 ) {
        Move( From, To );
    }
    else {
        Solve( k - 1, From, TheRestOne( From, To ) );
        Move( From, To );
        Solve( k - 1, TheRestOne( From, To ), To );
    }
}
public static void main(String args[]) {
    Hanoi   Self = new Hanoi();
```

```
                offgraphics.setColor (Color.black);
                int minx = 260 - Size * 4 + (j - 1) * 170;
                int maxx = 260 + Size * 4 + (j - 1) * 170;
                offgraphics.fillRect( minx, y, x - minx, 6 );
                offgraphics.fillRect( x + width, y, maxx - (x + minx), 6 );
            }
            else {
                offgraphics.setColor (Color.black);
                int minx = 260 - Size * 4 + (j - 1) * 170;
                int maxx = 260 + Size * 4 + (j - 1) * 170;
                int y = 192 - i * 8;
                offgraphics.fillRect( minx, y, maxx - minx, 6 );
            }
        }
    }
    g.drawImage(offscreen, 0, 0, null);
}
public void Move( int From, int To ) {
    int     i, j;
    Counter++;
    for ( i = Size - 1; i >= 0; i-- )
        if ( Tower[From][i] != 0 )
            break;
    for ( j = 0; j < Size; j++ )
        if ( Tower[To][j] == 0 )
            break;
    Tower[To][j] = Tower[From][i];
    Tower[From][i] = 0;
    repaint();
    try {
        Thread.currentThread().sleep (30);
    }
    catch ( InterruptedException e ) {
    }
}
public int TheRestOne( int i, int j ) {
    if ( i == 0 ) {
        if ( j == 1 )
            return 2;
        else
            return 1;
    }
    else if ( i == 1 ) {
        if ( j == 0 )
            return 2;
        else
            return 0;
    }
    else {
        if ( j == 0 )
            return 1;
        else
            return 0;
    }
}
public migratory void Solve( int k, int From, int To ) throws NotifyGone {
    if ( Counter >= 100 ) {
```

```
        Self.setSize( 520, 200 );
        Self.setResizable (false);
        Self.setBackground (Color.black);
        Self.setForeground (Color.green);
        Self.show();
        Self.Solve( Self.Size, 0, 2 );
    }
}
```

マイグレーションを行うプログラム

このプログラムをマイグレーションを行うように書き換える。マイグレーションを行う基準としては「板を100枚移したら1回マイグレーションを行う」こととする。そのため板を移した回数を数えておくカウンターを付け加える。マイグレーションは二つのホストの間を交互に行き交うことにする。

```
import javago.*;
import java.awt.*;

public class Hanoi extends Frame implements java.io.Serializable {
    public int Size = 20;
    public int Counter = 0;
    public int Tower[][];
    public String Destination[];
    public int NextDestination;
    Image offscreen;
    Graphics offgraphics;
    public Hanoi() {
        super ("Tower of Hanoi");
        int     i;
        Tower = new int[3][20];
        for ( i = 0; i < Size; i++ ) {
            Tower[0][i] = Size - i;
            Tower[1][i] = 0;
            Tower[2][i] = 0;
        }
        Destination = new String[2];
        Destination[0] = "//ritsuko:12002/";
        Destination[1] = "//flute:12002/";
        NextDestination = 0;
    }
    public void update( Graphics g ) {
        int     i, j;
        if ( offscreen == null ) {
            Dimension d = size();
            offscreen = createImage(d.width, d.height);
            offgraphics = offscreen.getGraphics();
        }
        for ( j = 0; j < 3; j++ ) {
            for ( i = 0; i < Size; i++ ) {
                if ( Tower[j][i] != 0 ) {
                    int x = 260 - Tower[j][i] * 4 + (j - 1) * 170;
                    int y = 192 - i * 8;
                    int width = Tower[j][i] * 8;
                    offgraphics.setColor (Color.gray);
                    offgraphics.fillRect( x, y, width, 6 );
```

Append メソッドに引き渡す。

WWW ページ

ユーザの入力を受け取り、アプレットに渡し、その結果を 新しいページとして表示する WWW ページは次のようになる。

cgiemu.html:

```
<html>
<head>
<title>JavaGO CGI emulation</title>
<script>
function invoke() {
    var arg = document.forms[0].elements[0].value;
    var docj = document.applets["TestApplet"].Invoke(arg);
    var doc = docj + "";
    document.open("text/html");
    document.writeln("<html><head>");
    document.writeln("<title>The CGI Result</title>");
    document.writeln("</head>");
    document.writeln("<body>");
    document.writeln(doc);
    document.writeln("</body></html>");
    document.close();
}
</script>
</head>
<body>

<applet codebase="." name="TestApplet" code="TestApplet.class" width=10 height=10 MAYSCRIPT>
</applet>

<h1>JavaGO CGI emulation</h1>

<form>
<input type=text name="Message" value="default">
<input type=button name="Click" onClick="invoke()" value="Click">
</form>

</body>
</html>
```

このページには文字列を受け取るフォームとボタンが一つ定義されている。ユーザがそのフォームに文字列を入力し、ボタンを押すと invoke という JavaScript の 関数が呼び出される。 その関数はフォームの内容を取り出し、 その内容を引数としてアプレットの Invoke というメソッドを呼び出す。 最後に Invoke メソッドの結果を内容として、現在のページを置き換える。

JavaScript の invoke という関数の中で空文字列を加えている部分がある。 こうしている理由は JavaScript と Java の間でのデータの受渡しの規約と 関係がある。 変数 docj の内容は Java の関数の返り値である文字列である。 JavaScript はそれを扱うことができない。そのため Java の文字列を JavaScript の文字列に変換する必要がある。 これを行うのが空文字列の結合である。

プログラムのコンパイル

コンパイルは次のようにすればよい。

```
jgoc Moving.jgo
javac TestApplet.java
```

この時に生成された class ファイルと JavaGO のクラスファイル群がブラウザからロードできるようにそれらを cgiemu.html が置いてあるのと同じディレクトリにコピーする。

付録３．３　NASTRAN

Web Site http://www.engineering-e.com/software/index.cfm より

付録４　Hyper-IT と同様の取り組み

収録資料

付録４．１　日本学術振興会

未来開拓学術研究推進事業

Web Site http://www.jsps.go.jp/j-rftf/physics_i.htm より

付録４．２　日本原子力研究所　ITBL

Web Site http://www.jaeri.go.jp/genken/press/001116/より

付録４．３　英国の最新科学ニュース他

Web Site http://www.ph.qmw.ac.uk/escience.html 他より

付録４．１　日本学術振興会

未来開拓学術研究推進事業

Web Site http://www.jsps.go.jp/j-rftf/physics_i.htm より

付録４．２　日本原子力研究所　ＩＴＢＬ

Web Site http://www.jaeri.go.jp/genken/press/001116/より

付録４．３　英国の最新科学ニュース他

Web Site http://www.ph.qmw.ac.uk/escience.html 他より

Data Management in an International Data Grid Project

Wolfgang Hoschek[1,3], Javier Jaen-Martinez[1], Asad Samar[1,4], Heinz Stockinger[1,2], and Kurt Stockinger[1,2]

[1] CERN, European Organization for Nuclear Research, Geneva, Switzerland
[2] Inst. for Computer Science and Business Informatics, University of Vienna, Austria
[3] Inst. of Applied Computer Science, University of Linz, Austria
[4] California Institute of Technology, Pasadena, CA, USA

Abstract. In this paper we report on preliminary work and architectural design carried out in the "Data Management" work package in the International Data Grid project. Our aim within a time scale of three years is to provide Grid middleware services supporting the I/O-intensive world-wide distributed next generation experiments in High-Energy Physics, Earth Observation and Bioinformatics. The goal is to specify, develop, integrate and test tools and middleware infrastructure to coherently manage and share Petabyte-range information volumes in high-throughput production-quality Grid environments. The middleware will allow secure access to massive amounts of data in a universal namespace, to move and replicate data at high speed from one geographical site to another, and to manage synchronisation of remote copies. We put much attention on clearly specifying and categorising existing work on the Grid, especially in data management in Grid related projects. Challenging use cases are described and how they map to architectural decisions concerning data access, replication, meta data management, security and query optimisation. [1]

1 Introduction

In the year 2005 a new particle accelerator, the Large Hadron Collider (LHC), is scheduled to be in operation at CERN, the European Organization for Nuclear Research. Four High Energy Physics (HEP) experiments will start to produce several Petabytes of data per year over a life time of 15 to 20 years. Since this amount of data was never produced before, special efforts concerning data management and data storage are required.

One characteristic of these data is that most of it is read-only. In general, data are written by the experiment, stored at very high data rates (from 100 MB/sec to 1GB/sec) and are normally not changed any more afterwards. This is true for about 90% of the total amount of data. Furthermore, since CERN experiments are collaborations of over a thousand physicists from many different

[1] This paper has been submitted to the IEEE, ACM International Workshop on Grid Computing (Grid'2000), 17-20 Dec. 2000, Bangalore, India

universities and institutes, the experiment's data are not only stored locally at CERN but there is also an intention to store parts of the data at world-wide distributed sites in so-called Regional Centres (RCs) and also in some institutes and universities. The computing model of a typical LHC experiment is shown in Figure 1.

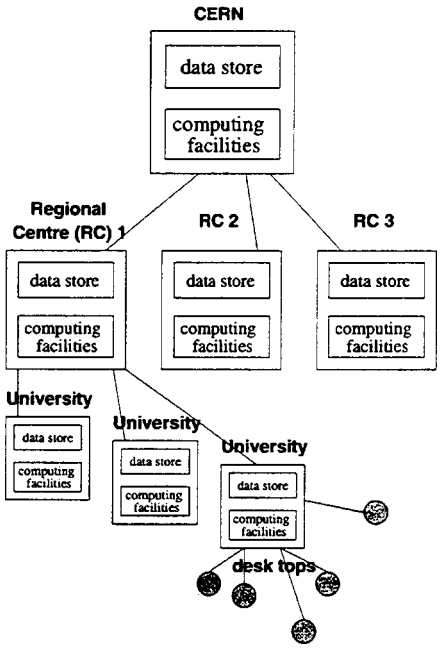


Fig. 1. Example of the network of one experiment's computing model

These RCs are part of the distributed computing model and should complement the functionality of the CERN Centre. The aim is to use computing power and data storage in these Regional Centres and allow physicists to do their analysis work outside of CERN with a reasonable response time rather than accessing all the data at CERN. This should also help the collaboration to have many scientists working spread around the world. Regional Centres will be set up for instance in Italy, France, Great Britain, USA and Japan.

By its nature, this is a typical Grid application which combines two aspects of today's Grid technology: Computational and Data Grids. In order to meet this challenge, the HEP community has established a project called "Research and Technological Development for an International Data Grid". The objectives of this project are the following. Firstly, establish a Research Network which will enable the development of the technology components essential for

the implementation of a new world-wide Data Grid on a scale not previously attempted. Secondly, demonstrate the effectiveness of this new technology through the large scale deployment of end-to-end application experiments involving real users. Finally, demonstrate the ability to build, connect and effectively manage large general-purpose, data intensive computer clusters constructed from low-cost commodity components. Furthermore, the project does not only cover HEP but also other scientific communities like Earth Observation and Bioinformatics.

The entire project consists of several work packages for middleware development, computing fabric and mass storage management, testbeds and applications. In this paper we present the data management aspects of the project. The objectives are to implement and compare different distributed data management approaches including caching, file replication and file migration. Such middleware is critical for the success of heterogeneous Data Grids, since they rely on efficient, uniform and transparent access methods. Issues to be tackled within three years include:

- the management of a universal namespace
- efficient data transfer between sites
- synchronisation of remote copies
- wide-area data access/caching
- interfacing to mass storage management systems.

A major aim of the project is to build on existing experience and available software systems. For the startup phase we have chosen the Globus toolkit as the starting point for our middleware research and development. Globus is a promising toolkit and has already proved several times that it is applicable for large Grid projects [12].

The paper is organised in the following way. The section on related work gives an overview of data management in current data Grid projects and discusses related issues of distributed database management systems and distributed file systems. Section 3 emphases the challenging requirements of data-intensive Grid applications. In sections 4 and 5 we present the overall architecture of the data management middleware components and give details on the individual components. Finally, conclusions and future work are presented.

2 Survey and Discussion of Related Work

Traditional distributed file systems like Network File System (NFS) [18] and Andrew File System (AFS) [15] provide a convenient interface for remote I/O with a uniform file name space. However, this approach does not support multisite replication issues and also cannot achieve good performance due to a lack of collective I/O functionalities, i.e. batch I/O and scheduled I/O. In contrast, parallel file systems like Vesta [5] and Galley [16], provide collective I/O but do not address complex configurations, unique performance trade-offs and security problems that arise in wide area environments. Finally, remote execution systems

enable location-independent execution of tasks scheduled to remote computers, but do not support parallel I/O interfaces or access to parallel file systems.

In distributed database research replication becomes more and more important. However, the research emphasis is on update synchronisation of single transactions on replicas [1] rather than considering the problem of transferring large amounts of data, which is an issue in our case.

None of these legacy systems are able to satisfy the stringent requirements, posed by both the scientific community and the industry, of having geographically distributed users and resources, accessing Petabyte-scale data and performing computationally intensive analysis of this data.

The notion of the "Grid" has been related to having access to distributed computational resources, resulting in being able to run computation intensive applications. The concept of having a Grid infrastructure which can support data intensive applications is new to the Grid community. There are a few projects, like Globus [4] and Legion [13], which were initially directed towards computational Grids but are now also adding support for distributed data management and integrating this with the computational Grid infrastructure. There is yet another class of on-going projects which have directed their efforts to support the distributed-data intensive applications from the very beginning. These mainly include Particle Physics Data Grid (PPDG)[17], Grid Physics Network (GriPhyN)[8], Storage Request Broker (SRB)[21] and the China Clipper[11] project.

The Global Access to Secondary Storage (GASS) API provided by Globus is the only component in the latest version of the toolkit which performs tasks related to data management. The scope of GASS API, however, is limited to providing remote file I/O operations, management of local file caches and file transfers in a client-server model with support for multiple protocols [3]. The Globus group is currently working on some of the data management issues including replica management and optimising file transfers over wide area networks [4]. The Globus philosophy is not to provide high level functionality, but to develop middleware which can be used as the base for developing a more complicated infrastructure on top.

The Legion project does not have any explicit modules working on data management issues. However, it does provide very basic data management functionality, implicitly, using the backing store "vault" mechanism [13]. High level issues like replica management, optimised file transfers and data load management are not addressed.

The Particle Physics Data Grid (PPDG) project is focussed on developing a Grid infrastructure which can support high speed data transfers and transparent access. This project addresses replica management, high performance networking and interfacing with different storage brokers [17]. This is a one year project and so the intentions are not to have very high level deliverables but to develop a basic infrastructure which can fulfill the needs of physicists.

The Grid Physics Network (GriPhyN) project is a new project whose proposal has been sent to NSF for approval in April 2000. The main goal of the project

is to pursue an aggressive programme of fundamental IT research focussed on realising the concept of "virtual data" [8].

Storage Request Broker (SRB) addresses issues related to providing a uniform interface to heterogenous storage systems and accessing replicated data over wide area. SRB also provides ways to access data sets based on their attributes rather than physical location, using the Metadata Catalog (MCAT) [21]. MCAT is a meta data repository system, which provides a mechanism for storing and querying system level and domain independant meta data using a uniform interface [14]. The China Clipper project has its high level goals to support high speed access to, and integrated views of, multiple data archives; resource discovery and automated brokering; comprehensive real-time monitoring of networks and flexible and distributed management of access control and policy enforcement for multi-administrative domain resources [11]. The project goals cover most aspects of a Grid infrastructure and also addresses the middleware development and not only the high level services.

These are the main initiatives which are looking at data management issues in a distributed environment. One of the main goals of our project is to work in collaboration with these on-going efforts, and use the middleware developed by them if it satisfies our requirements. Our final work aims at a system which would integrate or interact with these projects so that end-users can benefit from the efforts being put in, from all over the globe.

3 Use Cases

In our Data Grid initiative three different real-world application areas are included:

- High Energy Physics (HEP)
- Earth Observation
- Bioinformatics

Common to all these areas is the sharing of data in terms of information and databases, which are distributed across Europe and even further afield. The main aim is to improve the efficiency and the speed of the data analysis by integrating widely distributed processing power and data storage systems. However, the applications offer complementary data models, which allow us to assess how well a given solution can be applied to a general-purpose computing environment.

HEP is organised as large collaborations where some 2,000 researchers distributed all over the globe analyse the same data, which are generated by a single, local accelerator. The data access itself is characterised by the generalised dynamic distribution of data across the Grid including replica and cache management. As for Earth Observation, data are collected at distributed stations and are also maintained in geographically distributed databases. In molecular biology and genetics research a large number of independent databases are used, which need to be integrated in one logical system.

In order to get a better understanding of some of the requirements for the Data Grid, let us briefly outline the general characteristics of HEP computing. Experiments are designed to try to understand the physical laws of nature and to test the existing models by studying the results of the collisions of fundamental particles, which are produced after acceleration to very high energies in large particle accelerators. For example, beams of protons are accelerated in opposite directions and are forced to collide in all detectors along the accelerator. Each of these collisions is called an *event*. The detectors track the passage of produced particles. Moreover, the analysis of physical contraints of the produced particles implies computationally intensive reconstruction procedures.

Typical uses in HEP fall into two main categories, namely data production and end-user analysis:

1. Data production
   - central experimental data production at CERN (these data come directly from the on-line data acquisition system of the detector)
   - distributed event simulation
   - reconstruction of event data
   - partial re-reconstruction of event data
2. End-user analysis
   - interactive analysis
   - batch analysis on fully reconstructed data
   - analysis on full event data including "detector studies"

A typical interactive end-user analysis job starts with selecting a large initial collection of independent events. This means that the physics result obtained by processing the event collection is independent of the sequence of processing each single event. During the analysis jobs physicists apply some "cuts" on the data and thereby reduce the number of events in the event collection. In other words, a cut predicate is developed which is applied to the event collection in order to sieve out "interesting" events.

The process of constructing single cut predicates, i.e. optimisations of physics selections, can take several weeks or months, where the current version of the cut predicate is applied to the whole event collection or to subsets of it. One obvious optimisation for such an analysis job is to keep the most frequently used subset of events on the fastest storage (for example, in the disk cache).

An analysis job can possibly compute some very CPU intensive functions of all events, for example, a reconstruction algorithm could create a complex new event object which has to be stored for later analysis. This new object can be regarded as some additional information for this particular event.

Other jobs could apply multiple functions to every event. However, a considerable amount of time is spent on reading the objects, i.e. fetching the objects from the disk cache or from tape. Since all events are independent, a coarse grained parallelism based on the event level allows for a high degree of freedom in I/O scheduling and thus the events can be processed on different CPUs in parallel.

The Data Management tasks are to handle uniform and fast file transfer from one storage system to another. What is more, by studying the access patterns, meta data and file copies need to be managed in a distributed and hierarchical cache. In addition, security issues and access rights of the particular users must be considered.

4 Architecture

The Data Grid is a large and complex project involving many organisations, software engineers and scientists. Its decomposition must meet a number of challenges. The architecture must

- be easy to understand in order to be maintainable over time. Complex and fragile components are discouraged.
- be flexible so that different organisations can plug-in their own packages. A model based on a layered set of interfaces enables multiple implementations to coexist. Each implementation of an interface may focus on different characteristics such as performance or maintainability.
- allow for rapid prototyping. Thus it should leverage previous work as much as possible.
- be scalable to massive high throughput use cases. Careful design and layering is necessary to achieve this.
- respect the nature of distributed development. Effort is split between multiple teams, each working on a substantial component. Therefore components must be well defined and loosely coupled.

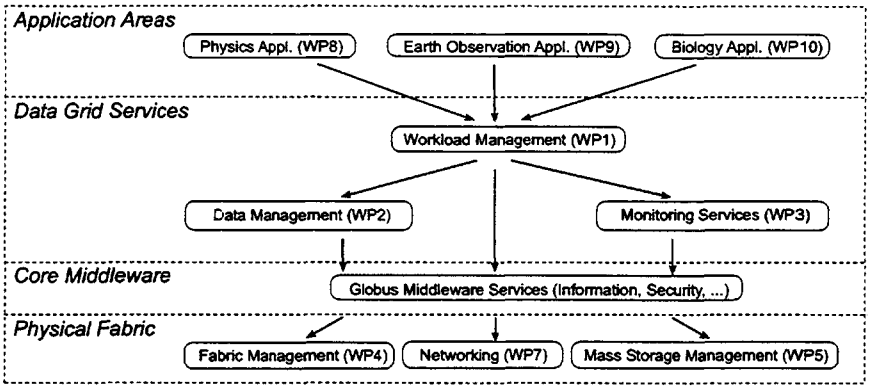


Fig. 2. Overall interaction of project work packages

We now sketch the overall architecture of the Data Grid as depicted in Figure 2. WP indicates the work package within the entire project. *High Energy Physics*, *Earth Observation*, and *Biology* exploit the developments of the project to offer transparent access to distributed data and high performance computing facilities to their respective geographically distributed community. *Workload Management* defines and implements components for distributed scheduling and resource management. *Data Management* develops and integrates tools and middle-ware infrastructure to coherently manage and share Petabyte-scale information volumes in high-throughput production-quality Grid environments. *Monitoring* provides infrastructure to enable end-user and administrator access to status and error information in a Grid environment. *Globus* services form the core middleware. *Fabric Management* delivers all the necessary tools to manage a computing centre providing Grid services on clusters of thousands of nodes. The management functions must uniformly encompass support for everything from the compute and network hardware up through the operating system, workload and application software. The *Networking* work package uses the European and national research network infrastructures to provide a virtual private network between the computational and data resources forming Data Grid testbeds. *Mass Storage Management* interfaces existing Mass Storage Management Systems (MSMS) to the wide area Grid data management system.

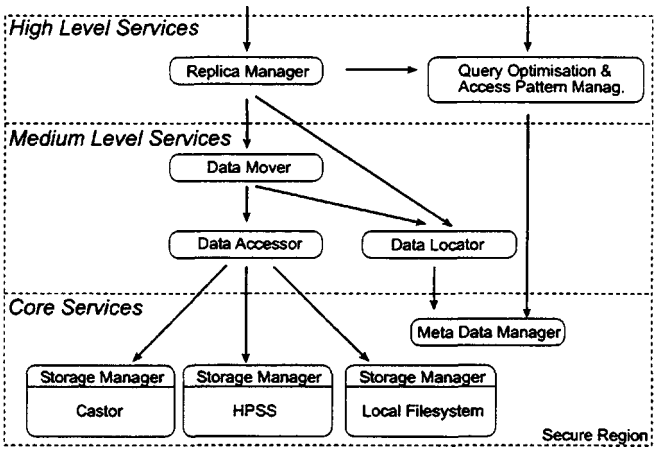


Fig. 3. Overall interaction of data management components

The Data Management work package which is our primary concern in this paper consists of a layered set of services as shown in Figure 3. Arrows indicate "use" relationships. Component A uses component B to accomplish its responsibilities. The *Replica Manager* manages file and meta data copies in a distributed

and hierarchical cache. It uses and is driven by plugg-able and customisable replication policies. It further uses the *Data Mover* to accomplish its tasks. The data mover transfers files from one storage system to another one. To implement its functionality, it uses the *Data Accessor* and the *Data Locator*, which maps location independent identifiers to location dependent identifiers. The Data Accessor is an interface encapsulating the details of the local file system and mass storage systems such as Castor [2], HPSS [10] and others. Several implementations of this generic interface may exist, the so-called *Storage Managers*. They typically delegate requests to a particular kind of storage system. Storage Managers are outside the scope of this work package. The Data Locator makes use of the generic *Meta Data Manager*, which is responsible for efficient publishing and management of a distributed and hierarchical set of objects. *Query Optimisation and Access Pattern Management* ensures that for a given query an optimal migration and replication execution plan is produced. Such plans are generated on the basis of published meta data including dynamic monitoring and configuration information. All components provide appropriate *Security* mechanisms that transparently span worldwide independent organisational institutions.

5 Data Management Components

5.1 Data Accessor

One of the core problems that any data management system has to address is the heterogeneity of repositories where data are stored. This is even more of a critical aspect when data management has to be targeted in a wide area network environment. The main problem to be solved is the variety of possible storage systems. These can be either mass storage management systems like HPSS, Castor, UniTree [22], and Enstore [7], multiple disk storage systems like DPSS [6], distributed file systems like AFS, NFS [18], or even databases. This diversity is made explicit in terms of how data sets are named and accessed in all these different systems. For instance, in some cases data are identified through a file name whereas other systems use catalogues where data are identified and selected by iterating over a collection of attributes or by using an object identifier.

To limit the scope of our initial work we will concentrate on data collections that are stored in either Hierarchical Storage Management (HSM) or local file sytems, leaving aside the extremely complex case of homogeneous access to data stored in different database systems. We are targeting specially HSMs because they provide an automatic and transparent way of managing and distributing data across a storage hierarchy that may consist of tapes, compressed disk and high-performance disk. This type of storage system is vital for HEP applications where the volume of data generated requires the use of tapes as a cost-effective media, and where data access requirements range from accessing it many times an hour during analysis to accessing it very infrequently ("cold" data).

Having these assumptions in mind, the problem to be solved within this component of our system is the definition of a single interface that can be used by higher level modules to access data located in different underlying repositories.

Thus, this module will have to make the appropriate conversions for Grid data access requests to be processed by the underlying storage system and to prepare the underlying storage system to be in the best condition to deliver data in a Grid environment. For HSMs, strategies like when data should be staged to a local disk cache before a Grid transfer is triggered, what requests are queued together to get the best performance in terms of tape mounts, and when files in the local cache are released to free space for new incoming requests will be performed by this subsystem in close coordination with the facilities provided by the storage system (existing internal catalogues, mechanisms for data transfer between tapes and local disks, etc).

In summary, this subsystem will hide from higher layers the complexities and specific mechanisms for data access which are particular to every storage system manipulating the performance factors which are proprietary for each system.

5.2 Replication

Replication can, on the one hand, be regarded as the process of managing copies of data. On the other hand, replication is a caching strategy where identical files are available at multiple places in a Grid environment. The main purpose of replication is to gain better response times for user applications by accessing data from locally "cached" data stores rather than to transfer each single requested file over the wide area network to the client application. Fault tolerance, and hence the availability of data, are key items of replication. Replication yields performance gains for read operations since a client application can read data from the closest copy of a file. Update and hence write operations need to be synchronised with other replicas and thus have a worse performance than updates on single copies. The performance loss of replication depends on the update protocols and network parameters of the Grid.

The problem of data replication not only involves the physical transfer of data among sites and the update synchronisation among the different available copies, but is also related to the more complex problem of deciding which are the policies or strategies that should trigger a replica creation. In a Grid environment replication policies are clearly not enforced by a single entity. As an example, system administrators can decide for production requirements to distribute data according to some specific layout, schedulers may require a particular data replication schema to speedup execution of jobs, and even space constraints or local disk allocation policies may force certain replicas to be purged. Therefore, the replication subsystem needs to provide adequate services for task schedulers, Grid administrators, and even local resource managers within clusters to be able to replicate, maintain consistency, and obtain information about replicas to be able to enforce any required policies.

The replication domain includes data and meta data to be replicated. These impose different requirements on the underlying communication system in the Grid. Since we are dealing with Petabytes of data that have to be transferred over the network to Regional Centres, there is an essential requirement for fast point-to-point file replication mechanisms for bulk data transfer. However, in

case of limited available bandwidth and limited efficiency (not all the theoretical bandwith is available) a solution that replicates everything everywhere may not be feasible.

The meta data replication requires a client-server communication mechanism at each Grid site. The Globus toolkit offers two possibilities: sockets and a more high level communication library called *Nexus*. The communication subsystem is required to implement different replication protocols like synchronous and asynchronous update methods. An important input factor for the decision of the underlying update mechanism is data consistency. A detailed survey of replica updates can be found in [19].

Once data are in place, the Data Locator is responsible for accessing physical files, mapping location independent to location dependent identifiers. This mapping is required in order to enable transparent access to files within a uniform namespace.

User requests are not directly handed to the Data Accessor, but are routed through the Replica Manager. The Replica Manager provides high level access services and optimises wide-area throughput via a *Grid cache*. It is an "intelligent" service that knows about the wide-area distribution of files. It analyses user access patterns in order to find out where and how files are to be accessed in optimal ways. As a consequence of these access pattern analysis, replicas are created and purged at remote sites.

The Data Accessor simply accesses files which are selected by the Replica Manager. With the Replica Manager taking care of wide-area caching, the mass storage system at each site is responsible for the *local caching* of files.

5.3 Meta Data

The glue for components takes the shape of a Meta Data Management Service. Particularly interesting types of meta data are:

- Catalogues comprising names and locations of unique and replicated files, as well as indexes.
- Monitoring information such as error status, current and historical throughput and query access patterns.
- Grid configuration information describing networks, switches, clusters, nodes and software.
- Policies enabling flexible and dynamic steering.

The key challenge of this service is to integrate diversity, decentralisation and heterogeneity. Meta data from distributed autonomous sites can turn into information only if straightforward mechanisms for using it are in place.

The service manages a large number of objects referred to by identifiers. Respecting the loosely coupled nature of the Grid, it must allow for maintainance of autonomous partitions *and* good performance, both over the LAN and WAN. Thus, the service is build on a fully distributed hierarchical model and a versatile and uniform protocol, such as Lightweight Directory Access Protocol (LDAP)

[23]. Multiple implementations of the protocol will be used as required, each focussing on different trade-offs in the space spanned by write/read/update/search-performance and consistency.

5.4 Security

Certain security aspects of a Grid infrastructure are tightly coupled to Data Management. Identifying these issues, and adapting the Data Management components accordingly, is of vital importance. Some of these issues are discussed here.

An important global security issue is to deal with the Grid cache. The site which owns the data has to make sure that the re mote sites hosting its data caches provide the same level of security as the owner requires for their data. This will be a serious issue when dealing with sensitive data where human or intellectual property rights exist. The fact that different sites will probably be usi ng different security infrastructures will result in more complications. I t is, therefore, required to evaluate strategies and develop tools which c an be used to ensure the same level of security with heterogenous underlyi ng security infrastructures.

Synchronous replication strategies, instead of using an on-demand or time scheduled approach, raise a lot of security concerns for the participating sites. A synchronous solution would involve giving time indefinite write permissions to other nodes in the Grid, so that whenever a replica is updated or deleted, the same operation can be propagated to all the remote replicas. An on-demand or time scheduled solution (asynchronous) is more secure and less consistent, though not as responsive.

The replica selection will depend on many factors, including the security policies of the nodes which contain these replicas. We may want to select a replica from a node which is more "friendly" as compared to one which forces more access restrictions.

The sensitivity levels associated with data and meta data might be different. The actual data might be more sensitive for some sites than their meta data or vice versa. This difference has to be incorporated in the overall design of the Data Management as well as the security system.

Several policy matters which are expected to vary from site to site include

- the usage of a synchronous replication strategy or something more secure,
- the importance of meta data as compared to real data in terms of security
- how much weight to be given to security when selecting a replica.

The intention is not to force all the sites in the Grid to agree on a common policy, but to design a system which is flexible enough to absorb heterogeneity of policies and present a consistent yet easy-to-adapt solution.

5.5 Query Optimisation

Queries are one way for an application user to access a data store. In a distributed and replicated data store a query is optimised by considering multiple copies of a file. A set of application queries is considered to be optimally executed if it minimises a cost model such as a mixture of response time and throughput.

The aim of a query execution plan is to determine which replicated files to access in order to have minimal access costs. We do not want to elaborate much further on a cost model here. However, the optimal query execution plan is based on static and dynamic influences like the following [9]:

- size of the file to be accessed
- load on the data server to serve the requested file
- method/protocol by which files are accessed and transferred
- network bandwidth, distance and traffic in the Grid
- policies governing remote access

The outcome of a query can be either the result of the query itself or a time estimate of how long it takes to satisfy a query. The Meta Data Management service will be used to keep track of what data sets are requested by users, so that the information can be made available for this task.

Query optimisation can be done at different levels of granularity. The method stated above is only based on files and also requires a set of files as an input parameter to the query execution plan. Often files have a certain schema and users query single objects of a file. This requires an additional instance that does the mapping between object identifiers (OID) and files by using a particular index [20] which satisfies the expected access patterns. This introduces another level of complexity for the query optimisation because objects can be available in multiple copies of files and the optimal set of files has to be determined to satisfy the query. Note that the OID file mapping is not an explicit task of the Data Management work package and needs additional information from Workload Management and Application Monitoring.

6 Conclusion and Future Work

In this paper we reported on preliminary work and architectural design which has been carried out in the work package "Data Management" in an International Data Grid project which has been proposed recently. We motivated our Data Grid approach by a detailed discussion and categorisation of existing work on the Grid, especially of data management in Grid related projects. The aim of our three year project is to provide Grid middleware services for the scientific community dealing with huge amounts of data in the Petabyte range, whereas the essential goal is to support world-wide distributed real-world applications for the next generation experiments in High Energy Physics, Earth Observation and Bioinformatics.

Basing the initial work on Globus, we have a Globus test bed running and preliminary promising prototypes are being implemented and tested. First results will be available by the end of the year. Furthermore, we are in close contact with the Globus developers concerning evolving Data Grid ideas and implementations.

Acknowledgements

We would like to thank Les Robertson for bootstrapping this interesting work, and Ben Segal for valuable feedback and contributions to the paper. Thank you to all Data Grid members for their interesting discussions.

References

1. T. Anderson, Y. Breitbart, H. Korth, A. Wool. Replication, Consistency, and Practicality: Are These Mutually Exclusive? Proc. SIGMOD International Conference on the Management of Data, pp. 484-495 1998.
2. O. Barring, J. Baud, J. Durand. CASTOR Project Status, Proc. of Computing in High Energy Physics 2000, Padova, Febr. 2000.
3. J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke. GASS: A Data Movement and Access Service for Wide Area Computing Systems. In Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems, May 1999.
4. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific DataSets. Network Storage Symposium, Seattle 1999.
5. P. Corbett and D. Feitelson. Design and Implementation of the Vesta Parallel File System. In Proceedings of the Scalable High-Performance Computing Conference, pages 63-70, 1994.
6. DPSS: Distributed Parallel Storage System, http://www-itg.lbl.gov/DPSS/
7. Enstore: http://www-isd.fnal.gov/enstore/design.html
8. GriPhyN: Grid Physics Network, http://www.phys.ufl.edu/ avery/mre/
9. K. Holtman, H. Stockinger. Building a Large Location Table to Find Replicas of Physics Objects. Proc. of Computing in High Energy Physics 2000, Padova, Febr. 2000.
10. HPSS: High Performance Storage System, http://hpcf.nersc.gov/storage/hpss/
11. W. Johnston, J. Lee, B. Tierney, C. Tull, D. Millsom. The China Clipper Project: A Data Intensive Grid Support for Dynamically Configured, Adaptive, Distributed, High-Performance Data and Computing Environments. Proc. of Computing in High Energy Physics 1998, Chicago 1998.
12. W. Johnston, D. Gannon, B. Nitzberg. Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. Eighth IEEE International Symposium on High Performance Distributed Computing, Redondo 1999.
13. LEGION: http://www.cs.virginia.edu/ legion/
14. MCAT: A Meta Information Catalog, http://www.npaci.edu/DICE/SRB/mcat.html
15. J. Morris, et al. Andrew: A Distributed Personal Computing Evironment. Comms. ACM, vol 29, no. 3, pp. 184-201, 1996.

16. N. Nieuwejaar, D. Kotz. The Galley Parallelle System. In Proceedings of the 10th ACM International Conference on Supercomputing, pages 374-381, Philadelphia, ACM Press, May 1996.
17. PPDG: Particle Physics Data Grid, http://www.cacr.caltech.edu/ppdg/
18. R. Sandberg. The Sun Network File System: Design, Implementation and Experience, Tech. Report, Mountain View CA: Sun Microsystems, 1987.
19. H. Stockinger, Data Replication in Distributed Database Systems, CMS Note 1999/046, Geneva, July 1999.
20. K. Stockinger, D. Duellmann, W. Hoschek, E. Schikuta. Improving the Performance of High Energy Physics Analysis through Bitmap Indices. To appear in 11th International Conference on Database and Expert Systems Applications, London - Greenwich, UK, Springer Verlag, Sept. 2000.
21. SRB: Storage Request Broker, http://www.npaci.edu/DICE/SRB/
22. UniTree: http://www.unitree.com/
23. W. Yeong, T. Howes, S. Kille. Lightweight Directory Access Protocol, RFC 1777. Performance Systems International, University of Michigan, ISODE Consortium, March 1995.

付録５　用語解説

参考用語集ホームページ

- 社団法人　日本電子工業振興協会(JEIDA)用語集
  (http://www.jeida.or.jp/document/geppou/yogo/index.html)
- ASCII Glossary Help
  (http://www.ascii.co.jp/ghelp/index.html)
- ネットワーク関連用語集
  (http://www.glasscom.com/Tone/Glossary/)
- 情報技術用語検索　(日経BPデジタル大事典 1998年度版)
  (http://bizit.nikkeibp.co.jp/it/hsia/keyword.html)
- 日経コミュニケション用語・略語検索
  (http://www4.nikkeibp.co.jp/NCC/f_yougo.html)

Active Server Page

WWW サーバ側で，JavaScript や VisualBasic Script などのスクリプト言語や，各種 AcviteX コンポーネントを動作させるためのフレームワーク。

ASP として記述された Web ページは，まずサーバ側で解釈・実行されるため，参照時の状態によってダイナミックなコンテンツをクライアントに提供することが可能である。

ActiveX

静的なものだったWWW（ワールド・ワイド・ウエブ）ページを動的なものに変えていくための技術。クライアント側の技術にはサーバーからプログラムをダウンロードして実行するActiveXコントロール、WWWページを記述するActiveスクリプティング、WWWブラウザー上で他のアプリケーションのデータをシームレスに表示するActiveDocument、各種フォーマットのビデオを再生するActiveMovieなどがある。

サーバー側の技術としてはWWWサーバー用APIのISAPI、サーバー・プログラムを記述するActiveXサーバー・スクリプティング、ActiveXコントロールをサーバー側で実行するActiveXサーバー・コントロールなどである。ActiveXコントロール対応のWWWブラウザーでは、WWWサーバーからWWWブラウザーにActiveXコントロールをダウンロードして実行することができる。ActiveXコントロールはx86ネイティブ・コードなのでJavaアプレットに比べて実行速度が速く、Windowsクライアントの機能をフルに利用できる。半面、セキュリティーはJavaよりも弱い。

アプリケーションサーバー（Application Server）

3階層システムにおいて，業務処理ロジック部分に相当するアプリケーションプログラムを開発したり，そのプログラムを実行する上で必要は通信機能や負荷分散機能を提供するソフトウェアのカテゴリを指す用語。3階層システムでは，サーバ部分がデータベースサーバと業務処理ロジック部分を実行するサーバに分かれるが，後者を指してアプリケーションサーバという場合もある。APサーバと省略することもある。

ASP (Application Service Provider)

アプリケーションに関するサービスを企業との契約を元にして提供する業態。具体的なサービス内容は，アプリケーションの配布，ホスティング，管理，ライセンス管理などだが，センタ側でアプリケーションを一元管理し，ネットワーク経由で利用する点が特徴。米国の調査会社 IDC が使い始めた用語で，同社はこの業態の急激な成長を予測している。

ASP は PC の普及に伴って、問題となってきた TCO の増大を解決するために、PC への負荷を押さえようと提唱された Thin Client や Network Computer すなわち Server Based Computing の思想や BPR を通じた業務標準化された ERP パッケージおよび最近の Internet 特に WWW の急速な普及によってもたらされた当然の帰結ということができ、決して全く新しい考えではない。

また、このことよりすべてのシステムがサービス可能なアプリケーションとなるわけではなく、ASP に適したアプリケーションとはどのようなものかを十分考察すべき段階にあるといえる。

B-Rep(Boundary Representation)

３次元形状を、その表面を構成するすべての面・稜線・頂点の境界データで定義する。

CSGに比べて自由曲面を含む物体の形状を正確に表現することができると言う長所がある。半面、形状の入力や修正が複雑ですべての境界データの記憶が必要なためデータ量が大きくなる。B-Rep形式からCSG形式へのデータ変換は困難である。

BPR (Business Process Reengineering)

ビジネスを行っている現状すなわち、製品の研究開発、製造工程、原価、品質、サービス、製品の提供行程など、業務全体を対象として効率や生産性を劇的に改善するために業務全体を根本的に見直し、再構築すること。企業がポスト工業化時代に生き残るため、業務を首尾一貫したビジネスのプロセスに再統合するという考え方である。

以前からあるTQC（全社的品質管理）が部門内の日常的な業務改善であるのに対して、BPRは企業活動全体を対象とした総括的な改革である。米国でいくつもの企業がBPRで成功を収め、90年代前半に日本でも大きな話題を呼んだ。BPRを行うと、業務や情報の流れが大きく変わる。BPR実施後の企業における、ビジネスプロセスは次のようになる。

(1)複数の仕事を一つにまとめる、(2)現場の従業員も意思決定を行う、(3)プロセス内のステップを既存の組織にとらわれず自然な順序で行う、(4)プロセスには複数のパターンを用意する、(5)仕事はもっとも適当と思われる場所で行う、(6)チェックと管理を減らす、(7)調整は最小限に抑えられるー。これらを実現するために、情報システムが大きな役割を担う。

CALS

製造業を中心に関心が高まっているのがCALSである。CALSは、インターネットの爆発的な拡大と合わせて進展する企業行動の総合的なネットワーク化である。

まず、CALSとは何かから触れていこう。CALSは、現在Continuous Acquisition and Life-cycle Supportという言葉またはCommerce At Light Speedの略とされている。しかし、CALSはもともと米国防総省で使われた言葉であり、初めはComputer Aided Logistic Supportとしてスタートした。その後、Computer-aided Acquisition and Logistic Supportになり、調達の意味が加わった。

CALSという言葉には以上のような変遷があるが、現在では国防総省の軍事用語的色彩を払拭している。現在のCALSの目的は、設計や製造、受発注など、企業間の取引に関するすべての行動を電子ネットワーク上で電子データとして交換し、商取引のスピードを劇的に向上させることにある。

CALSはそれ自体がまったく新しい技術革新をめざしているのではない。CALSの要素として重要な電子取引は、EDI（エレクトロニック・データ・インタチェンジ）としてすでに欧米では実行され、日本でも普及し始めている。EDIは、電子データ交換と訳されるが、その基本は電子ネットワークによる受発注データの交換である。

こうしたネットワーク活用の重要性が増してきた背景には、米国を中心とした企業の行動がグローバル化したことが挙げられる。設計は米国で、部品生産は東南アジアで、また製品組み立ては米国でというように、企業は世界的なネットワーク構造の中で行動している。電子ネットワークによる設計・製造・受発注データの交換が必要になるのは当然である。

こうした企業行動のネットワーク化の進展の中で重要なのは標準化である。企業間でデータ交換を行うには、プロトコル（手順）やフォーマット（形式）を標準化する必要がある。米国では、民間の任意団体である「CALSISG」がCALSの普及に向けた推進活動を行っている。

日本では、日本電子工業振興協会がCALSの研究を進めてきたが、その流れの上で95年5月に「CALS推進協議会（CIF）」し、国際的なCALS推進活動と連携するとともに、CALSを生産性向上に結び付けるための基礎研究に取り組んでいる。

また、現在通産省と各業界ではCALSプロジェクトを推進している。たとえば、自動車産業におけるCALS実用化研究事業、宇宙産業におけるCALSの運用、鉄鋼設備CALS実用化研究、建設全ライフサイクル策にわたるCADデータ交換、航空機の設計・生産・運用システムの開発と実証実験、電子機器・部品CALS実証実験、設計・製造データの交換および共用化の基盤技術（STEP）の開発、プロセスプラントのCALSの研究開発および実証、ソフトウェアCALS環境構築用のソフトウェア開発および実証、造船における技術情報の電子的交換に関する実証実験などがCALSプロジェクトであり、各業界では実用化に向けて研究を進めている。

このように、日本でもCALSが注目を集めている。それは米国製造業の復活がネットワークの活用にあるという認識が深まってきたことの現れだろう。インターネットではショッピング・モ

ールが形成され、電子空間上でビジネスが行われるようになってきている。セキュリティ面でインターネットで決済するのは危険だといわれてきたが、そうした問題も解決されるだろう。いわば、電子ネットワークなしに企業行動は考えられない時代に入りつつある。日本企業は世界市場のなかで競争しているのであり、CALSもそうした視点で考えなければならないだろう。

クライアントサーバシステム（Client Server System）

情報システムがダウンサイジングの傾向を強めるなかにあって、システム構築の考え方にも新しい波が押し寄せてきている。それがクライアント／サーバ型システムである。

従来のメインフレームを中心としたシステムはメインフレームをホストとし、端末はそれ自体では処理能力を持たないダム端末という構成だった。いわば、計算処理はホスト・コンピュータが全面的に受け持っていた。

しかし、パソコンの登場にみられるように端末もインテリジェント化が進み、垂直的な分散処理が70年代に始まった。そして、80年代後半からはパソコンやワークステーションをLANで結んだネットワーク・コンピューティングの時代にさしかかっている。

ダウンサイジングは、単にパソコンやワークステーションといった小型コンピュータを利用するといった大きさの問題ではなく、利用スタイルが異なってきていることに注目すべきである。

従来のホスト集中処理では、計算からメモリ、さらには通信処理などすべての機能をホスト・コンピュータが担っていた。端末はホストに依存する関係にある。しかし、クライアント／サーバ型システムでは、ネットワークにつながれた利用者が操作するクライアントと、種々の機能をクライアントに提供するサーバ（どちらもワークステーションやパソコン）は基本的に対等な関係にある。

ネットワークにつながれたコンピュータはそれ自体で処理能力を持ち、メモリを持っている。そして、LANを通じて他のコンピュータのCPUやメモリを利用することができる。また、データベースや通信処理などの機能をサーバとして設定したコンピュータに持たせ、システム全体の負荷を軽減するシステム構成をとることが柔軟にできる。

クライアント／サーバ型システムは、ワークステーションの発展のなかで生まれてきた。もともとワークステーションは、メインフレームが持っているCPUとメモリを複数のコンピュータに分割し、なおかつネットワークで接続した環境で利用しようという発想で生まれたコンピュータである。

米国では、企業の基幹システムをワークステーションをベースとしたクライアント／サーバ型システムに置き換えるケースがかなり報告されている。日本でも金融機関のトレーディング・システムをはじめ、ワークステーション・ベースのクライアント／サーバ型システムが構築されはじめている。

クライアント／サーバ型システムの特徴は、システム規模の拡大や機能拡張が簡単に行える点にある。ホスト中心型システムでは複雑なシステム変更が必要になるのに対し、クライアントを増設したり、機能ごとにサーバを追加することによって、スピーディかつ柔軟に規模の拡大や機能拡張に対応できることである。

クライアント／サーバ型システムの考え方は、最近のメインフレームも取り入れてきている。メインフレームは従来のように1台ですべての機能を持つのではなく、大規模システムのなかで役割ごとに各種のサーバとして機能を提供していくという考え方である。

クライアント／サーバ型システムによって情報システムはより柔軟になるとみられる。

参考：3-Tier Client/Server Environment Goals and Specifications
(http://http://sandbox.aiss.uiuc.edu/3-tier/goals.htm)

CORBA (Common Object Request Broker Architecture)

イントラネットが企業の情報システムとして定着し、また企業間や企業－消費者間をつなぐエレクトロニック・コマースが出現しようとしている。その基盤となっているのはネットワークであり、とりわけ、インターネット技術である。インターネットが、そしてWWWが人々や企業の行動を変えていこうとしている。そんな中で、コンピューティングのアーキテクチャにも変化が生じている。80年代後半から90年代前半にかけてメインフレームの時代からクライアント／サーバの時代へと移行し、そして、現在はネットワーク・コンピューティングという言葉が注目されている。

IBMはネットワーク・コンピューティング・フレーム（NCF）という表現で、オラクルはネットワーク・コンピューティング・アーキテクチャ（NCA）という表現でネットワーク・コンピューティングを戦略的なターゲットに据えている。また、サン・マイクロシステムズはJavaコンピューティングと名付けている。名称は異なるが、これからはサーバ中心であり、しかもさまざまなサーバの資源をネットワークで接続されたシン･クライアントで利用するという形態を目指している点では同じである。国内コンピュータ・メーカもそれぞれネットワーク・コンピューティングへの対応を進めている。

こうしたネットワーク・コンピューティングを実現するための中核的な規格がCORBA（コルバ）と呼ばれるものである。CORBAは、Common Object Request Broker architectureの略で、異なるコンピュータ上のオブジェクト同士のメッセージを通信するオブジェクト･リクエストブローカー（ORB）の標準的な仕様である。

CORBAは、1991年8月に米国のオブジェクト指向技術に関する標準化団体であるOMG（オブジェクト･マネジメント･グループ）がバージョン1.1をまとめ、1994年12月に2.0が策定された。600社に及ぶ会社がCORBAをサポートしているといわれ、事実上の業界標準となっている。

CORBAは、(1)ユーザインタフェース、(2)情報管理、(3)システム管理、(4)タスク管理－の4つの機能に分類される。また、OMGでは、CORBAサービスについて、(1)並列コントロール・サービス、(2)イベント・サービス、(3)外部サービス、(4)ライフ・サイクル・サービス、(5)ネーミング・サービス、(6)永続オブジェクト・サービス、(7)クエリー・サービス、(8)関係サービス、(9)トランザクション・サービスーについてまとめている。

CORBAの特徴は、メインフレームやUNIXなどに搭載されているアプリケーションをオブジェクトと位置づけ、基幹システムに必要なトランザクション処理などの機能をを持つソフト部品と組み合わせる方式を採用することにより、アプリケーションをプラットフォームに依存することなく拡張したり変更することが容易にできることである。

オブジェクト指向技術は70年代に登場し、ソフトウェア開発の生産性を向上させる技術として注目されてきたものの、普及には至らなかった。しかし、インターネットやイントラネットが普及し、分散コンピューティングの時代が到来してきたことにより、分散コンピューティングある

いは分散オブジェクトを実現する基盤とも言えるORBへの関心が急速に高まっている。CORBAはその標準である。

そのCORBAに対して、マイクロソフト社が提唱している分散オブジェクト技術がDCOM（distributed component object model）である。DCOMはマイクロソフトが開発したインターネット技術であるActiveXの中核技術の1つで、DCOMによって、クライアントのプログラムとサーバ上のソフト部品を連携させることができる。日本のコンピュータ・メーカはCORBAへの対応を中心としながらも、DCOMのサポートも表明している企業が多い。DCOMは、WindowsNT4.0に取り入れられており、サーバをWindowsNTとするケースでは、DCOMへの対応が必要となる。

ネットワーク・コンピューティングでは、オラクルなどが推進するシン・クライアントとしてのネットワーク・コンピュータとマイクロソフトのNetPCやWindows-based Terminal（WBT）との対抗に注目が集まっているが、新しいアーキテクチャとしての分散コンピューティングを支える分散オブジェクト技術をめぐって大きなうねりが起きている。ネットワーク・コンピューティングの世界は、インターネットと同様に急激に変化・成長していきそうである。

CAD, CAM, CAE (Computer Aided Design, Computer Aided Manufacturing, Computer Aided Engineering)

CAD Computer Aided Design

コンピューターによる設計支援システム。2次元処理と3次元処理用のシステムがある。3次元CADは、コンピューター上の仮想空間に3次元形状を作成しながら設計を進めていく設計支援システム。2次元CADシステムは主に図面作成に利用される。

3次元CADシステムを利用すれば、CAEや金型設計、試作、製品ドキュメントなど製品開発に関わる多くのプロセスで設計データを再利用できるなどの利点があり、コンカレントエンジニアリングには欠かせない。また、3次元の実体形状で表示されるので形状認識が容易になり、デザインレビューの効率化にもつながる。これまでは、金型設計やCAEなど特定の分野で利用されてきたが、ソリッドモデリングの実用化、パソコン上で稼動する3次元CADシステムの販売などにより急激に設計部門への広がりを見せている。

CAM Computer Aided Manufacturing

生産活動のための工程設計や作業設計、NC（数値制御）プログラミングなど生産準備を支援するシステム。加工や組み立てなどの生産工程に対する制御までを含める場合もある。一般には、NC制御装置や組み立て装置の制御プログラミングを支援するシステムをいうことが多い。

以前は、設計プロセスで作成された図面データをもとに制御プログラミングを行うものが中心であったが、最近では、立体形状データを受け取り、工具経路を自動計算して制御データを生成するものが主流になってきた。3次元CADの普及に伴い、ソリッドモデルの3次元CADシステムとの連携も進んでいる。

CAE Computer Aided Engineering

製品開発にかかわるエンジニアリングをコンピューターで支援するシステム。一般には、有限要素法などの解析手法による、強度解析、熱解析や振動解析などをいう。CAEは1980年に米SDRCが提唱した概念であり、設計案の性能や機能の検討に際して、コンピューター上に設計案を作成し、試作、実験を行うのと同様の検討をコンピューターシミュレーションで行う。実際の試作品を減らす、または試作せずに検討するアプローチである。

設計の初期段階からCAEを使いこなすことによって、設計変更などのむだを省くことができるので、コンカレントエンジニアリングに不可欠なツールになってきた。CAEシステムは、これまでは解析専門家向けのツールとして発展し、解析モデルも解析専門家が作成してシミュレーションすることが多かった。しかし、3次元CADの普及に伴って、3次元CADで作成されたモデルをそのまま利用して、設計者がCAEを実行できる環境が整ってきた。

ほとんどの3次元CADシステムが、CAE機能を統合した機能をもっており、設計環境下で手軽にCAEを呼び出せるようになっている。CAEは、これまで試作品での不具合の予測や対策に利用さ

れてきたが、今後は、設計の上流工程における製品の品質作りこみのための手法として利用されると予想される。感度解析や最適設計など設計の品質向上に不可欠な機能が実装されてきたことで、今後の設計環境に不可欠なツールとなるだろう。

CSG (Constructive Solid Geometry)

プリミティブと呼ばれる基本図形要素（立方体、直方体、円筒など）の集合演算（ブーリアン・オペレーション：和・差・積）により３次元形状を定義し、この形成過程を２進ツリー構造で記憶する手法。

B-Rep に比較すると形状の入力や修正が容易であり、データ構造も単純かつコンパクトである。半面、自由曲面を正確に扱うことは苦手である。CSG 形式から B-Rep 形式へのデータ変換は比較的容易である。

CRM (Customer Relationship management)

顧客との関係を緊密にすることによって、販売の拡大を目指すカスタマ・リレーションシップ・マネジメント（CRM）という手法が注目されている。CRM を効果的に実現するためには、インターネットへの対応や顧客データベース、受発注から出荷に至る情報システムを統合的に整備することが必要となる。こうした顧客対応に必要なシステム環境を統合的に提供するのが CRM システムである。

CRM は、顧客ひとりひとりの好みやニーズに対応した商品・サービスを提供することによって、顧客を離さない仕組みを作ることを目指している。そのために、業務プロセスやデータベースを顧客中心に組み直すことが求められる。

これまでの企業組織は、マーケティングや営業、製造、サポートなどの機能が企業の業務の流れに沿って作られてきたが、これを顧客中心に組み替えることが大きな課題となる。それを実現する手段が、顧客データベース（データウェアハウス/ビジネスインテリジェンス）やコールセンターなどの情報技術である。

顧客データベースは CRM システムの要となる。顧客データベースには、顧客の属性、過去の取引内容が納められている。また、コールセンターは、電話による注文はもちろん、電子メールやファクスなど、顧客からの多様なアクセス手段に応じ、顧客データベースと連携しながら CRM システムのフロントエンドとして効果的なセールス機能を果たす。

CRM システムはすでに通販業界や金融機関などで導入が進んでいる。また、製造業でもクレームや相談を受け付ける相談窓口をコールセンターへ衣替えする動きがでてきている。
CRM システムによる業務の流れは次のようなものである。

まず、顧客からコールセンターに電話によるアクセスがあるとする。コールセンターには CTI（コンピュータ・テレフォニ・インテグレーション）という技術によって、電話をかけてきた顧客のデータがオペレータのパソコンの画面に表示される仕組みだ。オペレータは過去の取引履歴などが掲載されている画面を見ながら顧客と会話をする。また、商品データベースを参照しながら、注文に応じて受注データを入力することもできる。商品在庫があれば、そのデータは商品管理部門や物流部門にネットワークで共有され、迅速な受注処理と配送処理が行われる。

仮に商品在庫がなければ、パソコン画面を見ながら別の商品を提案することも可能である。顧客の側がインターネットに対応していれば、顧客とオペレータが同じ画面を共有して、検討と推奨を行うことができる。代替商品に気にいったものがあれば、そこで商談が成立する。このよう

に、商機を逃がさない仕組みを作るのがCRMシステムである。

以上のように、顧客からのアクセスを商談に結びつけることをインバウンドと呼ぶが、企業の側から顧客に働きかけることをアウトバウンドと呼ぶ。CRMシステムはそのアウトバウンドにも大きな効果をもたらす。

アウトバウンドで重要な役割を果たすのはデータウェアハウスである。データウェアハウスに蓄積した膨大な顧客データから顧客の購買パターンなどを分析して、DM などで顧客ごとの嗜好にマッチした商品案内を行うことがアウトバウンドの一例である。的確な分析を行うことによってDMのヒット率を向上させることが可能となる。

また、コールセンターから受けた注文処理を効率的に物流など他のセクションに流すためのソフトとしてワークフローも重要である。また、ワークフローは、受注から出荷に至る処理の流れを管理する機能を持つ。そのため、「注文した商品は今どのように処理されているのか」といった顧客からの問い合わせにもワークフローがあればすぐに確認することができるようになる。

このように、CRMシステムはさまざまなITが活用され、顧客の満足度向上と販売増の両面に効果を発揮するものと期待されている。

Data ware House

データ・ウェアハウスとは、データの「倉庫」を意味する。現在、企業では情報システムによって意思決定を支援し、さらには迅速なアクションにまで結び付けることが大きなテーマになっている。そうした意思決定に重要な役割を果たすのがデータ・ウェアハウスである。

企業の情報システムは集中処理から分散処理へ移行しはじめ、クライアント／サーバ・システムが大きな流れになろうとしている。クライアント／サーバ・システムはまたエンドユーザ・コンピューティングのプラットフォームでもある。エンドユーザはクライアントであるパソコンを操作しながら、サーバに蓄積されたデータベースにアクセスして情報を入手し、自分の目的に合わせて加工する。それをもとに意思決定を行い、適切で迅速なアクションをとる。これが情報活用のスタイルである。

こうした情報活用は米国で始まり、日本企業にも定着しようとしている。日本企業では部門や機能別にリレーショナル・データベースを導入してデータベースを構築している。エンドユーザが情報活用が行える環境の整備に取り組んでいる最中である。情報活用によって企業はビジネスのスピードをアップし、競争力を高めることができるからである。

しかし、一方で問題点も指摘されている。

一般的に、意思決定支援システムはある部門や機能をサポートするためにシステム構築がスタートする。そのために特定の業務のアプリケーションが複数存在してくる。また、データベースについても特定ユーザ向けに加工されたデータベースが散在するケースも生じてくる。それぞれの目的に合わせて加工されたデータをもとに意思決定を行えば、判断のもとになるデータがばらばらであるところから適切な意思決定が行えないという事態を招く。

こうした事態を避け、全社的なデータを蓄積し、共通データによる意思決定を行うために考えられたのがデータ・ウェアハウスというアーキテクチャである。

データ・ウェアハウスは、全社規模のデータを蓄積し、各業務グループ間で共有されるものである。データ・ウェアハウスによって異なる仕事を行っているグループがそれぞれにデータを入手し活用しても、基となるデータは同じであり、適切な意思決定を行うことができるわけである。

データ・ウェアハウスはいってみれば、データを蓄積した巨大な倉庫である。しかも、その倉庫に蓄積されているのは加工されたデータではなく、「生データ」である。生産材にたとえれば、原材料あるいは素材といえる。生データを蓄積するという点がデータ・ウェアハウスのポイントである。

生データを一元的に蓄積するためには巨大な入れ物が必要になる。また、膨大な倉庫から求める答（データ）を得るためには高速検索処理が必要になる。したがってこうしたデータ・ウェアハウスを実現するためには超並列コンピュータなどの高速データベース・マシンがハードウェアとしては有効である。

また、今後は膨大な生データを検索して、入手する仕組みだけでなく、入手したデータを意思決定につなげる機能やプロセスをシステムに組み込むことが求められる。いわば、知識のメカニ

ズムを情報システムに組み込むことである。それが生データと結びつくことによって、適切なビジネス・アクションのプロセスを作り出していくことになる。

このようなデータ・ウェアハウスをさらに発展させたコンセプトを打ち出したコンピュータ・メーカもすでに存在している。さらに情報活用のスピードと付加価値が向上していくことになるだろう。

現在の日本における企業の課題はホワイトカラーの生産性を劇的に向上させることである。また、ビジネス・スピードを格段に速めることにある。そのための武器として情報システムの活用は不可欠である。

データ・ウェアハウスをはじめとするインフォメーション・テクノロジーは企業の活動を支えるインフラといえるだろう。

データベース管理システム DBMS (DataBase Management System)

データベース管理システム（DBMS）は、データベース（DB）の維持・運用を行う専用のソフトウエア。企業情報システムでは、ある組織のメンバーが持っている知識や得た情報をデータに加工・編集して蓄積し、これ整列・統合して共有するツールがDBMSだといえる。データベースが特定グループのためだけに構築されても、データとして統合化している規則さえ開放されれば、他の人が利用できることが特徴である。

コンピューターシステムでDBMSを利用するメリットには、以下のようなものが挙げられる。

(1)データを物理的、論理的にアプリケーション（応用）ソフトから独立させることができ、各種アプリケーションの作成やシステム構築が容易になる。

(2)データの整合性（一貫性）を保て、また、無駄な重複が少なくなる。

(3)データ利用の標準化ができ、情報の有効活用やソフトウエアの生産性向上が可能になる。

(4)データ（情報）に対するセキュリティーが確保しやすくなる。

DBMS は、どのようなデータの構成にも対応するため、汎用的なデータの管理構造をあらかじめ持っており、このデータモデルに合わせてデータベースを構築していく。DBMS で管理しているデータを利用する場合、具体的なデータの問い合わせ（クエリー）方法と検索のための索引（インデックス）を使用する。簡単にいうと、データベースの内側のモデルは、リストや表の組み合わせとして実現していることになる。

データモデルには、(1)ネットワーク型、(2)階層型、(3)リレーショナル型、(4)オブジェクト指向型――がある。現在、もっとも広く使われているのはリレーショナル型である。また、リレーショナル型のDBMS製品がオブジェクト指向型を取り込み始めている。

DCOM (Distributed Component Object Model)

米マイクロソフトのオブジェクト間通信規約である COM をネットワーク上で使えるように拡張したもの。既存の COM オブジェクトをそのままネットワーク上で利用できるように、DCOM の機構は OS レベルでサポートされる。クライアント・オブジェクトは、ファイル名の代わりに URL を使ってサーバー・オブジェクトを指定できるので、インターネット上で分散処理を実現することも可能である。マイクロソフトは DCOM をインターネット上の分散処理の標準にするために、その仕様を IETF（インターネット・エンジニアリング・タスクフォース）に提出した。また同社は米ディジタル イクイップメント社と VMS および UNIX 版を、独ソフトウェア AG とも MVS および UNIX 版を開発している。Macintosh にもインプリメントする予定である。また、Java アプレットを COM/DCOM のオブジェクトとしたり、Java アプレットから COM/DCOM オブジェクトを呼ぶことができるようにする。

DXF (Data eXcahnge Format)

Autodesk がバージョンが異なる AutoCAD 同士で図面データをやりとりするために制定した、ベクトルデータのファイルフォーマットである。

ファイルのフォーマットが公開されていることと、AutoCAD の普及によって、他の多くの CAD や CG ソフトでもサポートされている。

しかし DXF は、IGES,STEP,VRML といった業界あるいは公的機関で制定されたフォーマットではなく、正確な DXF ファイルであることを認定する組織もない。

よって、特に他の CAD とのデータ交換のために使うときには、十分検証してから利用する必要がある。

DXF の書式についての注意事項

テキスト形式の DXF ファイルとバイナリ形式の DXF ファイルの２種類がある。

AutoCAD は両方の形式をサポートしているが、他社の CAD では、テキスト形式の DXF ファイルしかサポートしていないものも多い。

テキスト形式では、DXF ファイルとして出力するときに、丸め誤差が発生してするが、バイナリ形式では、丸め誤差はおきない。また、テキスト形式の DXF ファイルは可読性がある。

しかし、DXF ファイルは AutoCAD のすべてのデータを表現できるという仕様であるため、AutoCAD がバージョンアップされるたびに、データの種類が増えており、このことが完全に DXF をサポートすることをより困難なものとしている。

また、DXF ファイル自身はその一部のみを利用することが許されており、したがって完全な DXF ファイルと、部分的な DXF ファイルの２種類が存在することになる。すなわち、完全な DXF ファイルとは、すべてのセクションが存在する DXF ファイルであり、部分的な DXF ファイルとは、一部のセクションが欠落した DXF ファイルのことをいう。例えば図形を定義する ENTITIES セクションだけの DXF ファイルでも、R13 以前の AutoCAD なら自由に利用できていた。

DXF 最新情報

以下の URL に多くの AutoCAD,DXF に関する情報が掲載されているので参照されたい。

http://www.autodesk.co.jp/

ERP (Enterprise Resource Planning)

企業の情報システムで利用されるソフトウェアに注目される動きが出てきている。それはエンタープライズ・リソース・プランニング・パッケージ (Enterprise Resource Planning Package) に対する関心の高まりである。ERP パッケージの普及は日本企業の情報システム部門の考え方に新しいインパクトをもたらす可能性を秘めている。

ERP パッケージは、日本語では統合業務アプリケーション・パッケージと呼ばれる。ERP パッケージは、製造、販売、物流、会計といったさまざまな業務を統合的に処理できる機能を持つソフトウェアである。ERP 自体は広義には、「企業の業務環境の統合管理」を指し、具体的には、「企業における生産、物流、会計などの各業務機能の最適化とバランスの確保を企業横断的に行うための企業モデル」を指す。そして、狭義には統合業務パッケージを指す (ERP 研究推進フォーラムによる)。

ERP パッケージを提供している会社としては、SAP、バーン、QAD、J.D.Edwards、SSA などがある。日本でもすでにこうした企業の ERP パッケージを導入するケースが出始めている。また、国産の ERP パッケージ・ベンダも登場し始めている。

ERP パッケージ導入のメリットはいくつかある。第一にスピードである。パッケージを利用するために導入に至る期間が、ソフトウェアをゼロから開発するより短縮される。

第二にコスト面での効果。パッケージ自体は海外製のものはかなり高価だが、開発・導入期間の短縮を考えればコスト削減を期待できる。

第三に ERP パッケージは、ソフトウェアに業務ノウハウが蓄積されており、新しい業務の流れを構築しやすいという面がある。

実は、第一のスピードや第二のコスト削減などの効果より第三の効果のほうが ERP パッケージ利用の本質的な効果である、と見る向きが多い。その背景には、本格化する日本企業の BPR (ビジネス・プロセス・リエンジニアリング) への取り組みがある。

BPR の目的は、ビジネス・プロセスを根本的に見直し、変革することによって競争力を強化することにある。仮に、ERP パッケージ製品が、簡素化された業務プロセスに基づいて作成されていれば、それを導入した企業は簡素化された業務プロセスを手中にすることができるわけである。いわば、BPR という目的を効率よく支える道具が ERP パッケージだと言える。

さらに、競争が国際的になり、日本企業の海外進出が当たり前になっていることも ERP 普及へ

の追い風になっている。ERP パッケージは各国のさまざまな基準に対する対応を進めており、グローバル化を推進する企業にとっては格好の道具なのである。

このように、ERP パッケージはさまざまなメリットを持っている。だが、同時にいくつかの課題が浮かび上がっていることも事実である。

たとえば、開発コストの削減が裏切られることもある。カストマイズやユーザ教育に予想を超えたコストがかかれば、得られるはずのコスト・メリットが得られなくなる。導入期間の短縮も同様である。カストマイズに時間がかかれば、効果は期待以下になる可能性がある。

こうした課題が浮かび上がってきた背景としては日本における ERP パッケージ活用の経験不足が指摘されている。現在のところ、ERP パッケージの大半は海外製であり、パッケージ・ベンダの考え方は母国の業務の流れをベースにしている。日本企業が必要とする機能に対するサポートが十分ではない面がある。また、ERP パッケージを活用してシステム構築を行うシステム・インテグレータも ERP パッケージに関する経験・知識が不足しており、必ずしもユーザの期待通りにシステムを構築できないという声がある。

また、ユーザの側も同様に ERP パッケージに何を期待するのかが明確でないと、そのメリットを十分に生かせない。日本企業の多くははまだ ERP パッケージに関する調査をしている段階であり、パッケージ・ベンダ、システム・インテグレータ、ユーザがともに日本における ERP パッケージ普及のための基盤作りを進めていくことが望まれる。

そうした点では、96 年 9 月にパッケージ・ベンダ、システム・インテグレータ、ユーザが集まって ERP 研究推進フォーラムを結成しており、ERP パッケージ普及への弾みになることが期待される。

GIS先進国の一つの例として、米国では環境問題、森林管理、国土計画などの分野でGISが先進的に用いられているのに対し、日本においてはこれらの分野における総合的な適用は非常に少ないのが実状です。

ただ、例として総務庁統計局、国土庁においてはGISの導入および稼動が行われています。しかし、総務庁の導入は1990年（平成2年）とつい最近であり、国土庁においてはGISの範疇には現在では含まれない旧式のシステムに頼っているのが実状です。

日本においてGISの普及が少ない原因：

○計画の過程が実際の開発の過程に対して追いつかない場面も出ており、 GIS は問題提議、問題表示、プレゼンテーションの手段としてしか利用されていない場合がある。

○GISを導入するのが組織内の一部に留まり、組織全体でそれを活用する事が困難な状況である。

○日本の都市計画は国や地方自治体の計画担当者より、GISのような有効ではあるが高価な手法を導入できない民間企業によって立案されることが多い。

○GISが高価なものである、同時に入力するための情報の入手が困難である。

○情報自体の標準化がなされていない。

○GISについてその背景とともに教育すべき機関が少ない。

○国内で開発されたGISソフトウェアがほとんどなく、海外から日本市場に対して完全に調整されたGISソフトウェアが少ない。

IGES (Initial Graphic Exchange Specification)

異なるCADシステム間で図形などのデータを交換する場合に利用するファイルフォーマットの1つ。米国規格協会（ANSI）により、1981年に図形データの規格として制定された。現在は、IPOが規格を作成している。IGESは、サポートするCADシステムが多いので、CADデータ交換用の標準フォーマットとして最もよく利用されている。ワイヤフレーム、サーフェイス、ソリッド要素の規定があるが、サーフェイスデータの交換に使用されることが多く、ソリッドデータ交換にはほとんど利用されていない。IGESでは、規定の使い方を厳密に規定していないため、CADベンダーが独自に解釈してデータ変換機能を実装することが多く、異なるCADベンダー間で正確なデータ交換ができないことが多い。また、ソリッドモデリングなど3次元CADの最新技術に追いついていないなど、コンカレントエンジニアリングを支える標準としては不十分である。将来的には、データ交換の標準は、新国際基準として制定が進められているステップに移行していく。

これらの状況への具体的な対応策として、自動車業界など主にデータを交換し合うユーザー間でIGESの運用規約（サブセット）を決めたりしている。日本自動車工業会が中心となって1993年に制定したJAMAIS（ジャパン・オートモーティブ・マニュファクチャリング・アソシエーション・IGES・サブセット）では、仕様の解釈が統一され、使用できる図形要素や出力媒体を限定している。

インターオペラビリティ（Interoperability）

異なるシステム間の相互運用性。他のコンピュータが管理するデータをアクセスしたり、他のコンピュータにあるプログラムを起動して、そのサービスを受けるなど、ネットワークで接続されたすべてのシステム資源をユーザーが自由に利用できることが理想である。

Java

インターネットの普及に伴ってさまざまな企業が新しいビジネス･チャンスをつかもうとしている。ブラウザ・ソフトやWebサーバなどインターネット関連ソフトでリーダーシップを握るネットスケープ・コミュニケーションズ。そして、WWW（ワールド・ワイド・ウェブ）の検索ビジネスで世界の基本となる情報を提供するヤフー。こうした新興企業群とともに見逃せないのがサン・マイクロシステムズである。

サン・マイクロシステムズは、ワークステーション・メーカとして成長を続けてきているが、ここにきて注目されているのは同社が開発した新しい言語である「Java」の存在である。

Java は、ネットスケープやオラクル、ボーランド、シリコン・グラフィックス、ＩＢＭ、マイクロソフトなど、コンピュータ関連の主要企業がライセンス契約している。いわば、インターネット上のアプリケーション･ソフト開発用プログラム言語として標準の座をねらう地位にいるのである。

Java はサン・マイクロが開発したプログラム言語であるが、まったく新しい言語というわけではない。UNIX用言語であるC言語の系統に位置づけられるが、C言語よりもプログラミングが容易という特徴を持つ。そして、何よりも重要な特徴は、Java はネットワーク（インターネット）上のコンテンツを作成し、配布し、使うための言語だということである。つまり、Java はインターネットと切り離せない言語として登場してきたのである。

しかも、現在のWWWが「見る」ものであるのに対し、Javaで開発したコンテンツをインターネット上で入手すると、ユーザはそのコンテンツで遊んだり学んだりすることが可能になる。つまり、ユーザがインタラクティブに参加できるのである。この「対話性」がJavaの大きな特徴といえる。

また、Javaでインターネット上のアプリケーション・ソフト(Appletという）を書くソフト会社は無償でJavaを入手し、利用することができる。

さらに、Java はOSを選ばない。Windows、MacOS、UNIX などの上で動く。つまり、Java は、OS を選ばず、無償で利用できる新世代のインターネット用アプリケーション開発言語である。ここにJavaが注目される理由がある。

とりわけ、ソフト会社にとってJavaは大きな魅力を持っている。それはソフトの流通を変える可能性をJavaとインターネットが持っているからだ。

これまで、ソフト会社はソフトを開発するための投資だけでなく、開発後に販売していくためにCD-ROMやフロッピィディスク代、収納するパッケージ、さらには物流費などの投資を必要としてきている。Java とインターネットを利用すれば販売にかかわるコストは激減すると考えられる。ソフト会社は開発に専念すればいい。Java とインターネットという環境がそのまま流通機構になるからだ。

Javaとともに注目されるのがHotJavaである。HotJavaはJava言語で書かれたブラウザ・ソフトである。Java を見るには HotJava が必要になる。HotJava はサン・マイクロシステムズの

WWW からダウンロードすることができるようになっている。HotJava はいわば Java という新しい世界への窓口の役割を担っているといえるだろう。

サン・マイクロシステムズでは、９６年１月に Java の普及を促進させるために子会社として JavaSoft を設立している。この会社の主要事業としては、(1)Java 製品を開発・販売・サポートすること、(2)世界市場におけるカスタマ向け製品を提供すること、(3)ネットワーク端末用 OS を Java をベースに開発することーなどが挙げられている。JavaSoft はサン・マイクロシステムズにとって戦略的な子会社だといえる。

インターネットは、政府や国際的な標準機関の主導ではなく、インターネット・コミュニティともいうべき民間の人々の情熱によって発展してきた。それをオープンな土壌というなら、サン・マイクロシステムズの企業風土や企業戦略とも重なりあう。だが、インターネットの世界のスピードは速い。また、競争も激しい。Java がどのような位置を占めるか、予測することは難しいが、Java がインターネットにおけるきわめて重要な技術／言語であることは間違いない。

Linux

パソコンOSの世界に新しい動きが起きている。それは、Linuxという名前のOSが急速に台頭しているためである。これまで、パソコンOSはサーバならWindowsNT、クライアントならWindows95(98)というWindowsベースがデファクト・スタンダードの地位を占めていた。だが、Linuxの登場によって、もう1つの選択肢が出現しつつある。

Linuxは、フィンランドのリーヌス・トーヴァルズ氏が開発したフリーソフトのUNIX。インターネットを通じてだれでも無料で入手したり、配布することができる。インテル系MPU(386以上)でもRISC系MPUでも稼働するため、PCサーバのOSとして利用するケースが出始めている。

Linuxは、94年3月にバージョン1が登場した。LinuxはOSのカーネルのソース・コードを公開している。そのため、世界中のユーザからの提案や指摘を受けてバージョンアップして来ている。いわば、草の根OSである。Linuxのユーザは世界中で数百万人、日本でもユーザが数十万人と言われるほどに普及している。

ただし、Linuxと呼ばれるものはカーネルのみである。企業でLinuxを利用するためには各種のユーティリティやライブラリが付属していたほうが便利だ。そこで、LinuxにOS周辺のソフトやアプリケーション・ソフトをセットにしてCD-ROMの流通パッケージとして提供する有料の販売形態も登場してきている。日本でも、Linuxとワープロ・ソフトやWebブラウザを組み合わせたパッケージが販売されている。

Linuxの特徴は、OSとしての安定性と導入費用が安価なことである。安定性については、長く使い込まれてきたUNIXをベースにしていること、また世界中のユーザの指摘を反映して改善していることなどから、Linuxは商用UNIXより安定しているという声が聞かれるほどに定評がある。

また、ソース・コードが公開されているため、障害が起きても原因究明や障害対策が容易に行えるのも大きなメリットで、導入費用についてはフリー・ソフトであるから周辺ソフトを含めても圧倒的に安価だ。

こうしたLinuxのメリットが評価され、企業ユーザにLinuxが浸透しているわけだが、大手のソフト・ベンダもLinux対応版の出荷を表明し、Linuxの地位はさらに強固になっている。米国オラクルや米ネットスケープ・コミュニケーションズ、インフォミックスなどが相次いでLinuxのサポートを表明している。こうしたベンダの動きがLinuxの普及に拍車をかけていると言える。

フリー・ソフトのデメリットとして指摘されるのはサポートが不十分であるということ。商用ソフトであれば、メーカやベンダが責任を持ってバグ修正を行うなどサポート体制が確立しているが、フリーソフトの場合は個人が開発したケースが多く、サポートには不安があるといわれてきた。しかし、Linuxの場合はボランティアによるサポートの仕組みがインターネットを通じて確立され、システム・インテグレータもLinuxのサポートをビジネスとし始めており、サポート面での不安はないといっても過言ではない。

日本でもLinuxの企業利用が始まっている。特に、WWWサーバやメール・サーバなど、インターネット関連の用途はUNIXのツールが利用できるため容易に利用できる。また、前述のように、RDBもLinuxで稼働するようになっている。

ただ、日常的な運用を考えると、社内にUNIXに詳しい技術者がいたほうが望ましいようだ。やはり、Linuxの導入を行っている企業は従来からUNIXに取り組んできた企業が多いようである。

ともあれ、PCサーバのOSとして、WindowsNTしか選択肢がなかった世界に新たに選択肢が登場したことは意味のあることだろう。

ネットワーク・コンピュータ (Network Computer)

インターネットの急速な普及に示されるように、情報システムはネットワークなしには考えられなくなってきた。企業内の情報システムの流れも、LANをベースとするクライアント／サーバ・システムが定着してきた。クライアント／サーバ・システムは、各種のサーバに蓄積されたデータベースやソフトウェアをクライアント側のパソコンで利用する仕組みである。一方、インターネットのWWW（ワールド・ワイド・ウェブ）にはさまざまな情報のソースが膨大に蓄えられている。

こうしたネットワークの進歩に伴って、コンピュータも変化しようとしている。この代表的なものがネットワーク・コンピュータである。ネットワーク・コンピュータは、パソコンをPCと略すように、NCと呼ばれる。NCの提唱者はRDBMS (Relational Database Management System) のリーダー的存在であるオラクル社である。

ネットワーク・コンピュータとはどういうものか。その概略を説明しよう。

まず、仕様としては、RISCタイプのCPUを搭載し、メモリは数メガバイト。ネットワーク・インタフェース、I/Oインタフェースを備え、ネットワークから起動することができる。価格は５００ドル程度を想定している。

ネットワーク・コンピュータの最大の特徴はパソコンのようにアプリケーション・ソフトウェアを購入してハードディスクに保存するのではなく、ネットワーク上のサーバからアプリケーション・ソフトをダウンロードする方法を取ることである。アップグレードやバグの修正など、ソフトウェア・アプリケーションの変更はサーバ側で行われ、ユーザは自ら操作することなしに最新のソフトウェアを利用することができるとオラクル社では述べている。

オラクル社はネットワーク・コンピュータの市場としてビジネス・ユース、教育機関、ホーム・ユースを想定しているが、まず登場しそうなのはビジネス・ユース向けの製品と考えられる。企業内（あるいはWWW）サーバにアプリケーション・ソフトを蓄積しておき、ユーザは必要に応じてソフトをダウンロードしたり、WWWにアクセスする。パソコンに比べてネットワーク・コンピュータの価格は数分の１であり、企業は情報システム投資を抑制することができる。

ネットワーク・コンピュータは単一の製品ではなく、製品ファミリーとして考えられている。具体的には、(1)ツーウェイ・ページャ／双方向型ポケベル＝電子メールや株価情報などを送受信する、(2)PDA＝手のひらサイズで無線による双方向通信が可能。電話番号やスケジュールなど毎日使う情報を整理するパーソナル・デジタル・アシスタント、(3)ウェブ・ターミナル＝デスクトップ型で、ウェブを検索できる、(4)ISDNビデオ電話＝ビデオ電話を主要な目的とするデスクトップ型で、ビデオ・メッセージやビデオ・クリップの書き込み・読み出しができる、(5)セット・トップ・ボックス＝テレビにつなげて使用するタイプーなどが製品化される予定である。

オラクル社のネットワーク・コンピュータ構想は、WWWをベースに発想されているようである。また、既存のOSに縛られない製品として考えられている。オラクル社のローレンス・エリソン会長は次のように述べている。

「ネットワーク・コンピュータの場合、オペレーティング・システムの標準自体が、デスクトップ・パソコンの世界からバーチャル・インターネット・デスクトップ、すなわちWWWをベースとしたハイレベルのオペレーティング環境にシフトしています」

このようにネットワーク・コンピュータはインターネット時代の到来に対応したコンセプトで設計されているのである。

日本のメーカにも富士電機がNCを投入するなどの動きが出てきている。パソコンの大量導入により、パソコンの年間維持コストの削減が企業の関心を集めているが、ネットワーク・コンピュータはそれに対する1つの解答と位置づけられる。

オブジェクト指向データベース管理システム OODBMS (Object Oriented DataBase Management System)

オブジェクト指向データベース管理システムは、オブジェクト指向プログラミング言語で記述されたデータを永続的に保存するためのツールとして出てきた。オブジェクト指向のプログラムでデータベースを利用することにより、長期間保存する必要のあるトランザクションのデータなどを管理したり、複数の処理プログラムでデータ（オブジェクト）を共有することができるようになる。オブジェクト指向データベースは、種類が多く、お互いの関連付けが複雑な情報やサイズが可変なデータを扱うことに適しているといわれている。

製品としては、米オブジェクトデザインの「オブジェクトストア」、米オブジェクティビティの「オブジェクティビティ/DB」、米バーサントオブジェクトテクノロジーの「バーサント」などがある。

また、米ジェムストーンシステムズの「ジェムストーン」では、RDBMS を OODBMS として利用するメカニズムを提供しており、ORDBMS として見ることもできる。

オブジェクト指向プログラミング（Object oriented programing）

システム開発のウエートが、ハードウェアからソフトウェアに移行するにつれ、ソフトウェア開発や保守・改良などの生産性を上げることが大きな課題になりつつある。その中で、最近注目されているのがオブジェクト指向プログラミングである。

オブジェクトとは「対象」、つまり現実にあるモノのことであり、オブジェクト指向言語とは現実のモノを指向した言語である。オブジェクト指向の基本的な考え方は、データと処理の方法を一本化（カプセル化）することにある。

データと処理方法の一本化は現実の世界では普通のことである。たとえば、会社の上司と部下の仕事の流れを考えてみればよい。上司はあるデータを部下に集計し、分析してもらいたいと指示を出す。その場合、経験豊富な部下は自分の方法でデータを集計し、自分の方法で分析して結果を出す。つまり、部下には処理方法が内在されているわけである。これをプログラミングに当てはめれば、何をしたいかという指示を出せば「データと一体化した処理方法」（オブジェクト）がその指示を実行するということである。処理方法はすでにオブジェクトに一体化されているから、新たに処理の方法に関するプログラムを書く必要はない。そのためにソフトウェアの生産性は飛躍的に上がることが期待されるのである。

こうした考え方を取り入れたのがオブジェクト指向言語で、シミュレーション用言語のSimulaやアラン・ケイが開発したSmalltalkなどである。UNIXで利用されるC言語にもオブジェクト指向プログラミングの発想が取り入れられつつある。

また、最近ではプログラミングだけでなく、ソフトウェアのライフサイクル全体のオブジェクト指向を取り入れようとする動きが定着化しつつある。プログラミングはソフトウェア開発ライフサイクルの一部を占めるにすぎない。システム分析、システム設計からプログラミングまで、すべてのフェーズにオブジェクト指向を取り入れなければ有効なソフトウェア開発はできないからである。

こうした観点から、日本でもオブジェクト指向がプログラミングだけでなく、ソフトウェア開発の方法論として注目されるようになってきた。

オブジェクト指向方法論は欧米で誕生し、すでに20種類以上の方法論が提唱されている。主なものに、コード・ヨードン法やOMT、BOOCH法などがあり、それぞれCASEツールが用意されている。

こうした方法論は、システム分析やシステム設計のフェーズでシステム化の対象である現実の仕事をモデル化する方法とツールを提供するものである。いわば、最も手のかかる分析や設計の生産性を向上させることを目的にしているわけである。

オブジェクト指向方法論とオブジェクト指向プログラミングの両方を利用することによって、上流から下流までのソフトウェア開発工程全体の生産性を上げることができるようになるだろう。

オブジェクト指向技術はデータベースにも取り入れられ、オブジェクト・データベースも商品

化されてきた。オブジェクト指向技術は今後のソフトウェアのなかで重要な地位を占めることになると思われる。

アウトソーシング（Out Souring）

91 年あたりからコンピュータ業界および情報サービス産業界で注目されている言葉にアウトソーシングがある。アウトソーシングは、業務の外部委託を指すがコンピュータ・システムの運用を一括して専門企業（コンピュータ・メーカや情報サービス企業）に任せることを IT アウトソーシングという。その業務範囲はソフトウェア開発やメンテナンスまで含むことが多い。

顧客からみれば、アウトソーシングは情報の資源を外部に委託することになるが、システムの受託サイド（アウトソーサー）はシステムズ・オペレーションと呼ぶこともある。システムズ・オペレーションは、システム・インテグレーションと重複する内容を持っているが、システムズ・オペレーションが運用にポイントを置くのに対し、システム・インテグレーションは企画・開発に力点を置いていると考えていいだろう。

アウトソーシングが話題になり、また実際にアウトソーシングが行われ始めたのには、ユーザのシステム部門を取り巻く環境の変化がある。

まず、ネットワーク化、分散化を軸にコンピュータ・システムの複雑化が進み、ユーザ企業の IT 部門では運用に対応しきれなくなっていることがある。とくに、ネットワーク・システムによって全国のデータ・ギャザリングを行ったり、受発注を行っているケースでは、ビジネスの拡大に伴うトラフィックやデータ量の増大に対して、ホストの能力増大やシステム・ダウン対策に追われているという現状がある。企業の IT 部門では、障害が起きたときにネットワーク上のどこに原因があるのか把握することが困難。また、インターネットやエクストラネットのように、テクノロジの進展が急で、企業の IT 部門が対応しきれないという側面もある。そこで、システムの運用を専門家に任せようということになる。

また、コンピュータ技術がオープン化指向を強め、従来のように 1 社の技術をフォローしておけばいいという時代ではなくなり、数社の最新で最良の技術・製品を把握しなければならなくなった。いわゆるマルチベンダ対応の中でメーカの競争も激しさを増し、それだけ製品のライフサイクルも短くなっている。さらに、ユーザのシステム部門がウォッチすべき要素は多様化している。ここでも、インフォメーション・テクノロジー（IT）の専門家のノウハウが期待される。その IT をもとに開発・運用に反映させようということである。

以上のように、現行のシステムが複雑化し続ける環境、テクノロジーの急速な進歩がアウトソーシングの背景にある。

米国では、イーストマン・コダック社がアウトソーシングをしてシステム・コストを削減させ

たという事例が有名である。同社は、コンピュータだけでなく、システム部門の社員をIBMに移管した。それが経費削減に寄与したという。米国のアウトソーシングの目的はこのように経費削減にある。しかし、一時的には経費削減につながっても、将来のシステム企画・開発を考えた場合、企業にシステム部門が存在しなくなることがいいのかどうか、米国でも見直しの気運が出てきているともいう。

日本では、IT 部門の社員がアウトソーシング先に移動することはないといわれてきたが、最近ではアウトソーシング専門の合弁会社を設立するなど、米国型に近いアウトソーシング形態も出現している。システムの規模のわりにスタッフが足りない企業やを最新のテクノロジの導入を急ぐ企業を中心としてアウトソーシングが進展していくだろう。BPR（Business Process Reengineering）に代表されるように、情報システムがビジネスの効果的な実行と密接に結びついている今、システム部門は、従来以上に戦略を意識したシステム立案と実行能力が求められている。そのなかで、運用とテクノロジ・ウォッチングはITの専門家に、システムの企画・立案は自分たちで、という一種のコラボレーションが浸透している。

PDM（Product Data Management）

企業の競争力を向上させる手法としてサプライチェーン・マネジメントが注目されているが、製造業におけるサプライチェーン・マネジメント実現のかなめとして注目されているのが PDM（Product Data Management/製品情報管理）である。

PDM は、自動車や家電などの設計図や部品表などのデータをコンピュータで管理するソフトウェアのことを指す。サプライチェーン・マネジメントは、設計から生産、資材調達、販売といった企業の一連の活動の全体最適を図る手法である。その流れのなかで、PDM は設計を中心に生産、資材調達などにかかわる機能をサポートする。いわば、サプライチェーンにおける重要な要素の１つと言える。そのために近年、PDM に対する関心が高まっている。

PDM 普及の背景には CAD の普及がある。製造業では CAD が普及し、１人１台が普通のこととなっている。一方で、CAD が普及することによる弊害も生じている。それは進捗管理の問題である。以前のように紙ベースで設計を行っていたときには、設計者ごとにどこまで進んでいるかが容易に把握できたが、コンピュータで設計をするようになると進捗管理が困難になってきた。一般に CAD が 10 台を超えると、どのコンピュータにどんな図面が存在しているかを把握することが困難になると言われている。

市場の動向に応じて設計変更が度々行われることも珍らしくないが、設計変更を行うには、どこのコンピュータに最新バージョンの設計図があるかを管理しておくことが重要である。これはアドレス管理機能と呼ばれる。DM は、このような設計図の図面を管理することが基本機能である。

また、PDM には設計作業のプロセス管理機能も用意されている。設計の承認など設計業務に関するワークフロー機能が提供され、設計プロセスの標準化を実現することができる。

さらに、部品管理機能も PDM の重要な機能である。この部品管理機能を利用することによって部品の選定や代替部品の選択、ベンダの選定などを迅速に行うことが可能になる。

最近の PDM 機能として注目されるのは、他の機能や社外との連携機能である。例えば、部品管理機能では資材部門との連携がある。部品コストを削減することは製品コストの削減につながる。部品コストを削減するためには部品の共用化が有効である。そこで、資材部門が共用を推進する部品をリストアップした部品データベースと PDM を連動させ、設計者が部品を選択するときに共用部品リストを見て選択する仕組みを作れば、低コスト部品の選定が簡単に行える。PDM がない場合は分厚い部品表から部品を選択するという作業になるから、作業スピードの面でもメリット

は大きい。

また、部品メーカへの発注という外部との連携でも効果を発揮する。この機能は設計 EDI と呼ばれる。部品メーカに発注する際には、設計図とともに見積もり依頼書や仕様書のやりとりが必要になる。紙ベースの書類では伝達速度が遅い。インターネットなどのネットワークを使用して設計図や書類を伝達すれば伝達速度を飛躍的に向上させることが可能になる。また、設計変更が起きてもすぐに最新バージョンの設計図を迅速に伝えることができる。部品メーカへの発注では、こうした設計図や文書のやりとりに時間がかかっており、PDM を利用したドキュメントのネットワーク化によって作業時間を３分の１に短縮することが可能になると言われている。

PDM は従来、大型コンピュータ上で稼働するものが多かったが、最近ではクライアント/サーバ型のPDM製品が登場してきている。また、操作もWebブラウザで利用できるようになっており、インターネットを通じたEDI(Electronic Data Interchange/電子データ交換)に対応している。

リレーショナル型データベース管理システム RDBMS (Relational DataBase Management System)
データベースのデータの単位であるレコードを、別のレコードと関連づける(リレーションする)処理機能を持つデータベースソフト。現在のオフィスソフトに位置するデータベースはほとんどこのタイプのデータベースを作成することができる。例えば、名前と住所のテーブルに対して、都道府県と郵便番号のテーブルを組み合わせることにより、完全な住所と名前のデータを作成したり、顧客管理データベースと売上管理データベースを連携して月別の顧客別売上げ表を作成するといった処理が可能となる。

ライトサイジング（Right Sizing）

コンピュータの世界でダウンサイジングという流れが大きな潮流になっている。極端な言い方をすれば、「大型汎用機は、ワークステーションやパソコンのネットワーク・システムにとって替わられる」ということになる。

確かに、ワークステーションやパソコンの高性能化が進み、また、これらをネットワークで接続することによって汎用機が行っていた業務をダウンサイジングすることができる。米国の銀行などでもそうした事例が見受けられる。

しかし、ダウンサイジングとは異なる視点からライトサイジングという言葉も語られる。日本語で言うならば「適材適所のコンピュータ利用」ということであろう。

ダウンサイジングとライトサイジング。結果として、両者は重なり合うこともあるが、根本的にはその内容は異なっている。

ダウンサイジングは、より小型のコンピュータを利用しようということが基本的な発想である。それに対して、ライトサイジングは、ある情報システムを構築する際に最もふさわしいサイズのコンピュータを利用しようという立場をとる。

だから、ライトサイジングでは、そのシステムにとって必要であれば大型汎用機を入れ、ワークステーションを入れ、パソコンを入れるということになる。ライトサイジングを追求した結果、汎用機が不要になったというようにダウンサイジングになることもあり得る。

ダウンサイジングは、80 年代後半に起きたハードウェア技術の進展（おもにワークステーションやパソコンの高性能化）、それにネットワーク技術の進展が相まって小型コンピュータが汎用機の世界に入り始めた現象のなかで出てきた言葉である。

それに対して、ライトサイジングはダウンサイジングの急速な進行に危機感を持った汎用機メーカの立場を表していると言えないこともない。ダウンサイジングの流れそのものは変わらないだろう。しかし、汎用機の世界もなくなるものではない。これが、汎用機メーカの認識であり、また、多くの汎用機ユーザの意識でもあるだろう。

コンピュータは、従来の汎用機だけだった世界から、多様化が進み、ミニコンが登場し、ワークステーション、パソコンと小型化し、用途別には科学技術計算用のスーパーコンピュータ、金融機関などで必要なオンライン・トランザクション処理用の OLTP コンピュータ、事務処理用のオフィスコンピュータなど、種々のコンピュータが商品化されてきた。

こうした各種のコンピュータを用途と目的に合わせて適材適所に配置し、最適な情報システムを構築することが求められている。その意味で、ライトサイジングはコンピュータの多様化に対するひとつの解答だと言えるだろう。

見逃してならないことは、ライトサイジングの裏側には「最適な投資で最適な情報システムを構築・運用したい」というユーザのニーズが存在することである。ユーザにとってはかつてのようにハードウェアの選択はさほど重要ではなくなっている。ユーザにとって重要なのは、情報システムによって企業目的を解決あるいはサポートするシステムを実現することである。その意味

では、コンピュータ・メーカにとって今後のサイズの問題よりも問題解決能力（ソリューション）が問われることになるだろう。

SET (Secure Electronic Transaction)

エレクトロニック・コマース（電子商取引）に大きな関心が集まっている。インターネットの普及はさまざまな新しい事業機会をもたらしている。インターネット上に店や商店街を構築するサイバーショップやサイバーモールがその典型的な例である。

ネットワークに店を出すコストは現実の店を出すコストに比べて安価ですむ。そのため、中小規模や地方の企業・店舗が全国ベースで出店をすることが可能になる。こうした企業はいわば、インターネットによって新しい事業機会を得ているのである。また、消費者にとっても現実の商店街では知ることのできない商品に出会うチャンスがもたらされる。

電子商取引に欠かせないのは決済である。商品をサイバーショップで購入する場合は、画面上のを選択し、購入したい商品アイテムや数量などの情報をショップ側に送る。そのときに同時にクレジットカードで決済できれば購入する側にとっては便利である。

しかし、インターネットはオープン・ネットワークであり、それが長所でもあり、短所でもあるという側面を持っている。クレジットカード情報の不正入手などに対するセキュリティ面での不安が指摘されているのである。

サイバーショップやサイバーモールの発展にはこうした安全な決済の仕組みを確立する必要がある。その標準的な手順としてSETと呼ばれるプロトコルが96年6月にまとめられた。SETは、セキュア・エレクトロニック・トランザクションの略で、クレジットカードの情報（本人確認など）をインターネット上で安全にやり取りするための通信手順の統一的な規格である。

このSET規格は二大クレジットカード会社であるVISAインターショナルとマスター・インターナショナルおよび技術提携パートナー7社（ＧＴＥ、ＩＢＭ、マイクロソフト、ネットスケープ・コミュニケーションズ、ＳＡＩＣ、テリサ・システムズ、ベリサイン）がが合意してまとめたものである。

この合意に至る以前は、VISAはとマスターは対立関係にあった。VISAはマイクロソフトと開発した「STT」という規格を前面にたて、一方のマスターはネットスケープ、ＩＢＭとともに開発した「SEPP」という規格を主張していたが、96年2月に規格を統一することに合意し、6月にSETの最新バージョンを発表し、クレジットカードによるインターネット・ショッピングの決済の統一規格が成立したのである。

この最新バージョンには、ビジネス上の要件や技術仕様書およびプログラマーに対する手引きが含まれている。最新バージョンがまとまったSETは、次の段階としてSETの実用化試験に入ることになる。具体的には、ソフトウエア開発会社がSET対応のソフトウエアを開発し、VISAやマスターカードはそれぞれの技術提携先とともに、顧客、小売業者、金融機関によるインターネット上の電子決済を実現していく。

電子決済のポイントは本人確認をするための暗号化技術にあるといわれる。SETで採用された暗号化技術は米国ＲＳＡデータ・セキュリティ社が開発した技術である。

ＲＳＡ社の暗号化技術は事実上の世界標準としての位置を占めている。日本企業もＲＳＡ社の

暗号化技術に大きな関心を寄せている。

電子決済の普及にはまた「電子公証」制度が重要になってくる。ネットワーク上で取引や申請を行う場合、申請者が実在するかどうか、取引先の会社が実在するかどうかといった確認がネットワーク上で行えるようにすることが大切になる。また、取引を行った後でも、その取引内容がネットワーク上に保存されている必要がある。申請内容や取引内容をを官公庁や地方自治体などの公的機関が保存し、開示する仕組みが電子公証制度である。

通産省はエレクトロニック・コマース推進事業の中で、電子公証制度について実用化に向けたプロジェクトを進めている。具体的には、(1)公的文書の届出、(2)電子商取引、(3)電子出版、(4)電子情報を保管する電子金庫の4システムについて実証実験を行う。また、ネットワーク上で「競り」が行える「電子取引所」の実用化にも取り組む計画である。

このようにインターネット上の取引やショッピングに関するセキュリティ面での対応はさまざまなアプローチで取り組みがなされている。サイバーショップやサイバーモールの普及のカギは、消費者や取引先の企業の安心感を与えるシステムの実現によるところが大きいと言えるだろ。

SSL, VPN, SDN／VPN, SET, S/MIME (Secure Socket Layer, Virtual Private Network, Software Defined Network / Virtual Private Network, Secure Electronic Transaction, Security services for MIME)

SSL Secure Socket Layer

WWWブラウザーおよびWWWサーバー間でやり取りするデータのセキュリティーを守るための技術。ソケットレベルでの暗号化および認証機能を実現するプロトコル。米ネットスケープ・コミュニケーションズが開発、米マイクロソフトなど主要なWWWベンダーも採用している。HTTPだけでなく、Telnet、FTP、mailやNewsなどにも適用可能。相手のWWWサーバーが本物であることを認証したり、ユーザーがブラウザーでデータを流す前に暗号化を行うことで、盗聴される危険を小さくすることができる。

VPN（仮想私設網） Virtual Private Network

仮想専用網ともいう。専用線を利用して構築した企業などの専用通信網と同様に、接続する端末の範囲を限定した閉鎖的な通信網を、電気通信事業者の公衆通信網の設備を利用して実現したもの。初期のVPNとして、パケット交換サービスのCUG（closed user group：閉域接続）がある。電話網を利用するVPNはインテリジェント・ネットワークのアプリケーションの1つとして実現した。NTTが1994年からサービスを提供している「メンバーズネット」が一例である。各利用者のVPNは、それぞれの電話番号体系を決めることによって、論理的に閉じた電話網となる。最近はインターネットの設備を利用するVPN、つまり広域イントラネットが注目されている。VPNは電気通信事業者の設備を多数の利用者が共同利用するので、広域専用網を経済的に実現できる。

SDN／VPN Software Defined Network / Virtual Private Network

仮想私設網または仮想閉域網と訳され、公衆ネットワークを専用線のように利用できるサービス。米AT&Tが1985年11月に最初に行ったサービスであり、一般的にはVPNと呼ばれる。VPNでは企業などのユーザーグループが任意の電話番号体系を設定し、その番号を使ってVPN加入者間で自由に電話が利用できる。大口割引が適用されるため、専用線のシステムを導入するより安価なのが特徴。国内では、94年2月にNTTが「メンバーズネット」というサービスを開始、その後、新電電3社も追随した。国際通信でもKDDの「VIRNET」をはじめとして、各社がサービスを提供している。最近は、音声だけでなく、インターネットでもVPN技術が登場、セキュリティー技術を利用した仮想的なプライベートネットワークの構築が進んでいる。

SET Secure Electronic Transaction

オープンなネットワークであるインターネット上で、クレジットカードによる決済を安全に行うためのプロトコル仕様。

インターネットを使ったオンラインショッピングでは、クレジットカードの番号などをそのまま

電子メールで送ったりすると、盗聴や改ざんの危険がある。こうした危険を避けるため、電子メールそのものを暗号化して仮想店舗へ送るという方法が現在は取られている。しかし、この場合でも、暗号メールを受け取った仮想店舗が入手したカード番号を悪用する可能性はある。
SETを使えば、仮想店舗側でカード番号を直接見ることができなくなるため、悪用の可能性をなくすことができる。また、店舗にとっても、カード利用者を確実にチェックできるようになるため、他人のカードによる不正購入などの危険を回避することができる。
SETは米ビザ・インターナショナル、米マスターカード・インターナショナルの2大カード会社をはじめ、米マイクロソフト、米ネットスケープ・コミュニケーションズ、米IBM、米ベリサインなどが共同で標準化を進め、97年6月に第1版のSET1.0が公開された。SET1.0は公開鍵暗号によるRSAを採用したが、現在、仕様の検討が進んでいるSET2.0では楕円暗号を採用する動きがある。

S/MIME　Security services for MIME
電子メールのセキュリティーを守る伝送方式の1つ。インターネット環境はもともとセキュリティーが弱く、オンライン・ショッピングなど電子商取引が普及しにくい傾向にあるため、こうした技術の向上が急がれている。もう1つの有力な方式であるPEM(プライバシー・エンハンスト・メール)が電子認証局(CA)による運用を前提としているのに対し、S/MIMEはCAなしでも実現できる。ユーザーは1組の公開鍵と秘密鍵を事前に持っている必要がある。電子メールの標準仕様であるMIMEを拡張し、電子メール本体に対する暗号処理と電子メールに添付するデジタル署名を提供している。セキュリティーの必要性が認識されてきている中、次世代電子メール標準規格として期待が高まっている。

ソリッド・モデリング (Solid Modeling)

３次元物体の表現手法の１つ。物体表面の情報だけでなく、内部についても色や質量などのデータを持つ。ワイヤフレーム・モデルやサーフェース・モデルに比べて、物体の加工を行う際に高度な処理が可能。

データの表現方法としては、CSG や B-Rep などが存在する。

ソリッド・モデリング・カーネル【幾何エンジン】(Solid Modeling Kernel)

ソリッドモデラ等を構築する際に核となるソフトウェアライブラリで、ライセンス契約により自由に利用することが可能となるソフトウェア。基本形状の生成、曲面の生成、属性の付加、集合演算といったソリッドモデリングの基本機能を備えている。

ベンダーはこれに、ユーザーインターフェース、データ変換機能、レンダリング機能、パラメトリック変換機能、フィーチャー・ベース・モデリング機能など、必要と思われる機能を付加することで、目的のソフトウェアを完成させる。

現在、ACIS(米 Spatial Technology)、Parasolid(米 Unigraphics Solutions)、等が各種 CAD、CAE ソフトのモデリング・カーネルとして採用されている。

ステップ STEP (STandard for the Exchange of Product model data)

異なる CAD システム間で製品データを交換するための国際標準規格。IGES に代わる CAD データ交換用の標準フォーマットとして標準化が進められている。ソリッドモデルのデータ交換も考慮されている。ステップの特徴は、IGES が図形データの規定に限定されているのとは異なり、プロダクトモデルという概念を導入して、製品構成、材料データ、加工用の NC（数値制御）データなど、製品のライフサイクル全体にまたがる広範囲なデータを対象としている点である。これによって、製品の設計から生産プロセスまでの情報を共有化することができる。また、自動車、造船など産業ごとに製品データの種類や構造を記述したアプリケーションプロトコル（AP、データ交換フォーマット）が用意されるなど、多様な分野、目的に合った標準化を目指している。ステップによるデータ交換機能を持つ CAD システムも登場しており、今後は IGES に代わってステップによるデータ交換が主流になるものと思われる。ただし、現状では CAD ベンダーの対応は統一されておらず、市販の CAD システムがどのような AP をサポートしていくかなど、データ交換の標準規格として共通に利用するための課題もある。

コンカレントエンジニアリングでは、アプリケーション間で共通に利用できるデータ形式の標準化が強く望まれており、製品モデル全体のデータ交換を対象としたステップの標準化は、今後の製造業の製品開発プロセスの枠組みを大きく変える可能性があるものとして注目されている。

SGML, XML, HTML, ダイナミック HTML (Standard Generalized Markup Language, eXtensible Markup Language, HyperText Markup Language, Dynamic HyperText Markup Language)

SGML Standard Generalized Markup Language

電子文書を標準化するための汎用的な言語で、文書の構造を「マーク」を使用して記述する。1986年にISO（国際標準化機構）の国際規格となっている。

タイトルや製品名、人名など、文書の意味や構造からみて、重要な事項にマークをつけることで、検索が容易になる、抄録や索引を作成できるといったメリットが得られる。CALS を実現するための標準規格の1つだ。1986年10月に国際規格（ISO 8879-1986）になっている。

XML eXtensible Markup Language

次世代HTML。普通のテキスト形式のファイルで、HTML 文書と表面上はよく似ており、文字列をタグと呼ばれる予約語で挟み込む。異なるのは、文書の構造を DTD（＝文書型定義。ドキュメント・タイプ・デフィニション）ファイルにすることで、表現方法の指定や文章中の文字列に意味を付加するような独自のタグを拡張できる点だ。オブジェクト指向の階層構造、認証機能によるドキュメントのチェック機能、強力なハイパーリンク機能などが特徴。電子出版などで用いるSGML の不要な機能を省き、インターネット向けに最適化する形で開発された。WWW コンソーシアムが標準化の作業中。

HTML HyperText Markup Language

ホームページを作成するときに使われるページ記述言語。WWW の標準化団体 WWW コンソーシアムが仕様を策定している。タグと呼ばれる「<」と「>」で挟まれた予約語を使って、テキストの整形や画像ファイルの表示位置およびリンク先の指定、スクリプトの宣言などをする。これをWWW ブラウザーで開くと解釈・表示される仕組みになっている。本来はエディターなどで記述するのが一般的だが、ドロー系ソフト感覚で自動的に HTML 文書化できる開発ツールや、HTML 化機能を標準搭載したワープロ／表計算ソフトが登場し、ホームページ作成だけでなく一般公式文書の記述にも利用されている。SGML を基に開発された言語だが、特定のブラウザーでしか表示できない拡張タグが乱発され、問題になっている。

ダイナミック HTML Dynamic HyperText Markup Language

Netscape Communicator4.0 と Internet Explorer4.0 で扱える仕様を備えたプログラム言語。HTML の機能を拡張したもので、マルチメディアやエンタテインメント分野の表現法を高度化する技術として注目されている。最大の問題はブラウザー間の互換性がないことで、標準化に向けて WWW コンソーシアムが調整中。主な特徴としては、以下の各点が挙げられる。

(1)ダイナミックスタイル：スクリプトによる文書スタイルの動的な変化が可能になる。

(2)ダイナミックコンテンツ：スクリプトによるコンテンツの動的な変化が可能になる。

(3)視覚効果：スクリプトを使わず、フィルターやトランジション、アニメーションコントロールなどを使ってページの要素を切り替えられる。
(4)位置指定：CSS を使い各要素をページ内の意図した場所に置くことができる。オーバーラップも可能。
(5)データ連携：データベースのデータと HTML とを連結して扱える。一般のファイルでも組み込みが可能。

サプライチェーン・マネジメント（Supply chain management）

ここのところ、日本企業の関心を集めているソフトウエアとしてERP（エンタープライズ・リソース・プランニング）パッケージがあるが、それとともに話題となっているのがサプライチェーン・マネジメント・ソフトウエアである。

米国ではこのサプライチェーン・マネジメント・ソフトを活用して競争力を高めているケースが相次いでいる。サプライチェーン・マネジメントとは、企業の調達・生産・輸送・保管・販売という一連のビジネス・プロセスの最適化を図る考え方である。ERPが企業内の資源のリアルタイムな把握を目的にしているのに対して、サプライチェーン・マネジメントは利益増大を目的にしている。

最近では、ERPパッケージとサプライチェーン・マネジメント・ソフトを組み合わせたり、ERPにサプライチェーン・マネジメント機能を付け加えたりするなど、両者の融合が始まっているが、米国では必ずしも両者を統合的に利用しているケースが格別多いわけではない。

サプライチェーン・マネジメント・ソフト分野では、米国のi2テクノロジーズ社やマニュジステックス社が有名である。

サプライチェーン・マネジメント・ソフトはいくつかのモジュールで構成される。たとえば、スケジューリング支援ソフトでは、各種資源の制約を考慮して最適なスケジュールをダイナミックに作成することを支援する。また、生産計画支援ソフトも生産計画のシミュレーションを行うことにより最適な生産計画の作成を支援する。需要管理支援ソフトは、過去の実績や市場の需要動向を調べて生産計画に反映させるためのソフトで、精度の高い需要予測を可能にする。

サプライチェーン・マネジメント・ソフトを導入した企業ではさまざまな指標で効果を上げている。たとえば、あるパソコン・メーカーでは在庫回転率を30回から50回に改善するとともに、売上げを3倍に増加させながらプランナーの増員はしなくて済んだ。また、ある大手電機メーカーはプランニング・サイクルを50%以上短縮している。このように、サプライチェーン・マネジメント・ソフトを活用すると、計画のサイクル・タイムを短縮したり、納期や生産のリードタイムの短縮を実現できる。また、市場の変化に応じた計画作成・変更がスピードアップするので、市場機会を見逃したり、逆に過剰生産に陥るといったことを防ぐことができる。パソコン・メーカなどでは、注文仕様に基づく生産が主流になってきた。限られた製品を大量生産方式で製造・販売すると、市場ニーズが変化したときに大量の在庫を抱えることになってしまう。そこで、注文が入ってから組み立てを開始する方法がクローズアップされてきた。

この場合、適正な需要予測にもとづく資材や部品の手配が重要になる。資材や部品の手配が最適にできていなければ注文に応じきれなくなるからだ。また、注文を受けてから組み立て・生産を開始するリードタイムも短縮しなければ、顧客満足度が下がり競争力が低下する。さらに、需要動向が変動すれば、それに合わせて調達や生産をダイナミックに変更しなければならない。サプライチェーン・マネジメント・ソフトはこうした問題点を1つずつ解決するのではなく、全体のプロセスの最適化と同期化を図りながらシミュレーションして問題を解決するツールである。

こうしたアプローチは企業間でも取り組みが始まっている。ある飲料メーカは缶を製造する会

社とエクストラネットで情報を交換し、飲料の生産計画にタイミングを合わせた発注を行っている。これは企業内のサプライチェーンを超えて、企業間のサプライチェーンの最適化を図ろうとする動きである。このような企業間で協調しながら効率化を図る動きはインターネットの普及とともにますます広がっていくことになるだろう。

日本ではERPパッケージの導入が緒についた段階だが、サプライチェーン・マネジメント・ソフトに対する関心が高まってきており、その動向が注目される。

サーフェース・モデリング（Surface Modeling）

ワイヤフレームの線分で囲まれたところにサーフェース（面）を定義した数値モデル。隣り合う２つの面が連続的につながるような処理を施したり、逆にエッジとして強調したりして一つの曲面を表現する。加工用NCデータを作成したり、工具の干渉チェックなどが行える。しかし、面のどちら側が物体の内部であるかの情報を持ちあわせていないため、モデルの体積や重心といったマスプロパティの計算はできない。

シン・クライアント（Thin Client）

企業におけるパソコンの導入が進み、ホワイトカラーに対するパソコンの配備が１人１台体制になっている会社も珍しくなくなっている。今や、電子メールによる連絡は当然のこととなり、企画書やプレゼンテーション資料をネットワークでやりとりに活用するなど、ビジネスマンにとってパソコンは不可欠な道具となっている。

このようにパソコンの普及はビジネスの効率化に大きく寄与しているが、一方で情報システムの運用という観点からは問題も見えてきている。それは、まずエンドユーザ側の負荷の増大という点である。

パソコンは年々高機能化し、複雑さを増している。それだけ、パソコンを使用するエンドユーザが覚えることが増え、負荷が増しているのである。こうした問題を解決するために、エンドユーザに操作方法を教えるヘルプデスク・サービスが登場してきているほどである。

また、情報システム部門にとっても負荷が増えている。それは、アプリケーション・ソフトを更新する際に全国に散在しているクライアント・パソコンごとにソフトを入れ換える作業に手間暇がかかることである。エンドユーザが自分でソフトの入れ替えを行うことは難しいから、システム部門や外部のソフト会社などが行うことになるが、全国の支社や支店に足を運んでソフトを入れ替えるのは時間とコストがかかる。

こうした問題点を解決するべく登場してきたのがシン・クライアントと呼ばれるコンピュータである。

シンとは「薄い」あるいは「ほっそりした」という意味の英語。それに対して、従来のパソコンをファット（太った）PC と呼ぶことがある。シン・クライアント製品としては、オラクル社などが提唱しているネットワーク・コンピュータがある。また、マイクロソフトとインテルが打ち出している NetPC もシン・クライアント製品である。

シン・クライアントの考え方は、データの管理やアプリケーション・ソフトの管理などをサーバ側で行い、クライアント側の負荷を少なくすることにある。クライアント側で行っている機能をサーバ側に移すことにより、クライアントの負荷を減少するわけである。

それによって、問題になっているパソコンの TCO（トータル・コスト・オブ・オーナーシップ＝総所有コスト）を削減することが可能になる。TCO とは、パソコンの購入費とは別に、パソコンを維持・運用するためのコストを指している。その中には、エンドユーザがパソコンの操作を学ぶための時間や新たにソフトをインストールする時間などが含まれる。シン・クライアントを導入することによって、このようなエンドユーザの負荷を軽減し、ひいては TCO の削減を実現することが期待されるのである。

例えば、アプリケーション・ソフトを切り換える際には、個々のクライアントごとにソフトを入れ換えるのではなく、クライアントがネットワークを通じてサーバにアクセスした際に自動的に新しいアプリケーション・ソフトを読み込むスタイルなので、ソフトのバージョンアップが簡単にしかも速やかに行える。

このようなことから、シン・クライアントを導入すると、TCOを20～30%削減できるという見方もある。

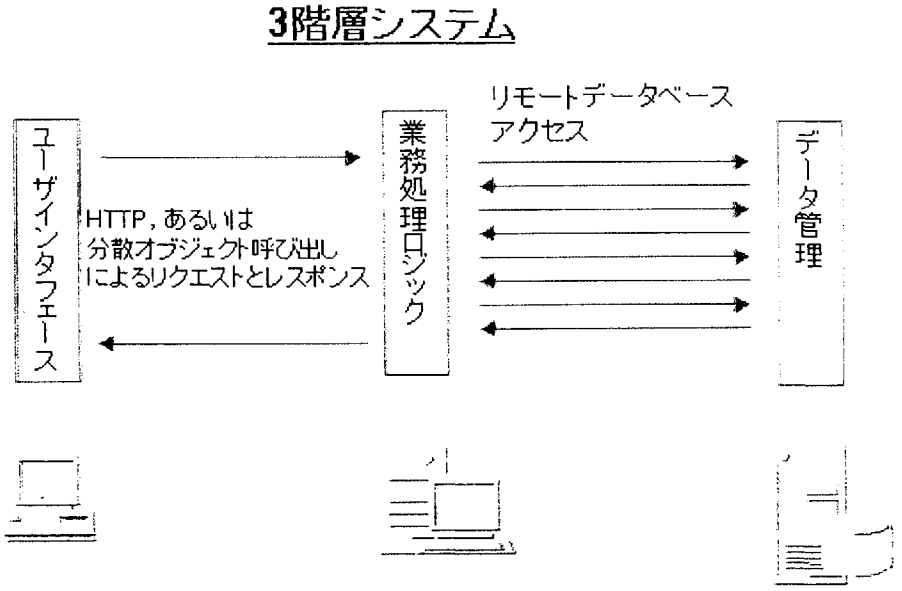
シン・クライアントは既存のパソコンに比べて機能をシンプルにしている分、価格が安いのも企業にとっては魅力の1つといえる。そこで、ホスト・コンピュータにつながっている何百台、何千台というダム端末の置き換えとして有望といわれる。ただし、昨今は、パソコンの価格が下がり、1,000ドル・パソコンという言葉も聞かれ始めている。価格面ではパソコンとシン・クライアントにそれほどの差はなくなり始めており、パソコンかシン・クライアントかはやはりTCO削減という視点から選択すべきだろうと考えられる。

また、シン・クライアントを導入するためにはサーバやネットワークの見直しが必要になる。シン・クライアントによってクライアントの負荷を軽減する一方で、サーバの負荷が増大する。さらに、ネットワークの伝送スピードも問題になる。シン・クライアントを導入するにはこうしたクライアント以外の部分にも考慮する必要があるだろう。

パソコンがビジネスの基本的な道具となっている今、エンドユーザの負荷軽減やTCOの削減は企業にとって重要なテーマである。シン・クライアント登場の意義は大きい。

3階層システム（Three Tiered System）

クライアントサーバ型システムの形態の一つ。ユーザインタフェースを受け持つクライアント，業務処理のロジックを実装したアプリケーションサーバ，データを管理するデータベースサーバの3階層でシステムを構成する。主に大規模システムで採用される。



TCO (Total Cost of Ownership)

企業ではビジネスの道具としてパソコンが1人に1台行き渡りはじめ、また、イントラネットによって情報の共有が進んでいる。しかし、それに伴って企業には対応すべき課題が浮上してきている。その大きなものがTCO（トータル・コスト・オブ・オーナーシップ）と呼ばれるものである。

TCOは、「総所有コスト」とか「所有維持コスト」などといわれる。TCOを初めに指摘したのは米国の調査会社であるガートナー・グループである。同社では96年9月に、ネットワークに接続されたパソコン1台の所有コストは1年間に約1万2,000ドルかかると発表している。日本円では120万～130万円というあたりである。その内訳は、aハードウェア/ソフトウェア投資、sエンドユーザ・オペレーション、d技術サポート、fシステム管理ーなどとなっている。

そのうちの、ハードウェア/ソフトウェア投資が占める割合は26%で、ほぼ4分の1程度である。それに対して最も多くの割合を占めるのがエンドユーザ・オペレーションで43%を占める。その他は、技術サポートが17%、システム管理が14%を占めている。

このTCOの指摘において重要なことは、ハードウェア/ソフトウェア投資という従来からの投資に加え、エンドユーザ・オペレーションという目に見えない投資が4割以上も占めているということである。また、ネットワーク運用などに関するシステム管理コストも問題になってきている。

TCOが指摘された背景には企業におけるパソコンの大量導入がある。そのため、「パソコンに触るのは初めて」というユーザが大量に発生した。そこで、ユーザがパソコンを操作するときに生じるコストが出現してきたわけである。

ガートナーグループは、TCOのうちのエンドユーザ・オペレーションに関する調査も発表している。それによると、エンドユーザ・オペレーションのなかでプログラム開発が39%を占め、第一の比率を持っている。このプログラム開発の高さは日本においては同等に論じられないかもしれないが、その他の項目は十分に納得がいく。エンドユーザ・オペレーションに要するその他のコストの内訳を比率の高いものから並べると、aソフトウェアの導入・バックアップが14%、s時間の浪費が14%、d自己学習が13%、f同僚のサポートが13%、g研修が7%といった具合である。時間の浪費というのは、凝った資料を作成したり、ゲームやネットサーフィンに使われる時間のことを指している。

新しいソフトをインストールしたりバックアップを取ること、あるいはパソコンを使いこなすために研究を受けたり自分で学習したり、さらには同僚からの求めに応じて操作を教えたり、というそれぞれの社員のビジネスとは関係のない部分に時間が費やされているのである。

こうしたTCOというテーマに対する有効策を前面に出して登場したのがオラクルなどが中心となっているネットワーク・コンピュータである。ネットワーク・コンピュータの効果はシステム管理の容易性やソフトウェアのバージョンアップにエンドユーザの手間がかからないという点である。ネットワーク・コンピュータはクライアント側にアプリケーション・ソフトを持たせ

ずに、サーバに配置する。サーバに新しいバージョンのアプリケーション・ソフトを搭載しておけば、クライアント側がサーバにアクセスしたときに自動的に新しいアプリケーションが利用できるようになる。ネットワーク・コンピュータは、いわゆるシン・クライアントの一種であり、マイクロソフト社の推進するNet-PCも同様の発想で開発されている。

また、システム管理面でのコストを削減するためにはネットワーク管理ツールや運用管理ツールを利用することも有効である。1人1台環境のイントラネットではクライアントの台数が増加し、ネットワーク資源を効率的に利用することや、ネットワークの負荷を軽くすることが求められる。そうした面でネットワーク管理ツールや運用管理ツールが脚光を浴びている。

さらに、エンドユーザがパソコンを操作していて問題に突き当たったときに、支援するヘルプデスクもTCO削減に効果をもたらす。そのため、ヘルプデスク・サービスを専門的に提供する会社が登場している。また、企業では社内にヘルプデスク部門を設けて社内ユーザを支援するケースも出てきている。

このように、TCOを削減するためには1つのアプローチではなく、さまざまなアプローチで取り組むことが必要なようである。

Web Application Server

インターネット技術の普及に伴い、企業内の情報システムにも新しい波が広がり始めている。音声網とデータ網を IP（インターネット・プロトコル）を用いて統合する動きがその 1 つである。これはネットワークの効率的な利用を目的としている。また、従来のクライアント/サーバ・システムにもインターネット技術が導入されてきた。それがWebアプリケーション・サーバと呼ばれるソフトウェアである。

Webアプリケーション・サーバは、Webシステム上でアプリケーションをブラウザを利用して操作するためのサーバ・ソフト。インターネットの普及により、企業内のパソコンの大半にブラウザが搭載されており、また受発注システムなどでインターネットを利用するシステムが増加している。Webアプリケーション・サーバはこうした動向を背景に登場してきた。

Webアプリケーション・サーバの特徴は、サーバの負荷を軽減できること。従来のWebシステムでは、Web サーバを通じてデータベース・サーバにアクセスする際に CGI（Common Gateway Interface）を利用している。CGI 方式では、クライアントからサーバに対する要求に応じて CGI アプリケーションが複数配置され、Webサーバの負荷が増大する。これに対して、Webアプリケーション・サーバは、アプリケーションを Web サーバと切り離して分散配置しているので Web サーバの負荷が軽減される。そのため、アプリケーションの実行速度も上がる。

Webアプリケーション・サーバは、ネットスケープコミュニケーションズや IBM、富士通などから製品化されている。

Webアプリケーション・サーバの用途は、まずエレクトロニック・コマースなど、Webを用いて情報をやり取りするケースからスタートしそうである。いわゆるインターネット・ショッピングのようなWebサイトにおける受発注業務への適用のほか、サプライチェーン・マネジメントのような企業間のデータのやり取りにも利用されるだろう。

また、卸と小売り間で従来のファックスによる受発注業務をインターネット経由に置き換えるというように、中小規模も企業や店舗でもパソコンとブラウザさえあれば受発注業務が行えるため、幅広い普及が期待できる。最近では、24 時間営業の店舗が増え、夜間に発注データを送信したいというケースが増えている。Webアプリケーション・サーバは多地点からのデータ配信やアクセスに向いている。

こうしたエクストラネット的な利用のほかにも、Webアプリケーション・サーバの利用メリットがある。それはクライアント・パソコンを管理するためのコストを削減する効果である。

クライアント/サーバ・システムではクライアント側にもアプリケーション・ソフトをインストールするため、アプリケーションの更改時には再インストールする必要がある。企業では、パソコンを大量導入しており、アプリケーションを更改するための費用が無視できない額になっている。Webアプリケーション・サーバを利用すればアプリケーション・ソフトをサーバ側で一元管理する仕組みなので、クライアントの維持・管理コストを削減することができる。いわゆるTCOの削減にも寄与すると期待される。

このように、Webアプリケーション・サーバはエクストラネット的な利用やクライアントのコスト削減といった部分的な適用からスタートしていくだろうが、企業では基幹系システムにもWeb技術を取り入れることを検討し始めており、Webシステムはいっそう広がっていきそうである。Webアプリケーション・サーバはそのカギを握るソフトとして導入が進んでいくと見られる。

Windows NT 4.0

インターネットの企業利用が注目されている。いわゆるイントラネットがそれである。

イントラネットが注目される理由は、WWW（ワールド・ワイド・ウェブ）が社内外の情報共有に優れた効果をもたらすことが挙げられる。

マイクロソフトが発表した WindowsNT の最新バージョンである「WindowsNT Workstation Versin4.0」および「WindowsNT Server Version4.0」（いずれも日本での発売は 96 年 12 月 10 日）はそうしたイントラネット/Web 利用を意識した新しい OS である。

WindowsNT4.0 は前バージョンの 3.51 に比べ、アプリケーションの実行パフォーマンスを大幅に向上させている。Server4.0 では、ファイル・サービス性能が 3.51 に比べて最大で 60%以上、Web サービス性能は 40%以上向上している。また、アプリケーション・サービスの性能は、マイクロソフト SQL サーバとの組み合わせでは約 50%性能が向上している。

こうした性能面での強化に加えて、ネットワーク機能の強化が一段と進んでいる点が WindowsNT4.0 の大きな特徴である。もちろん、その主要な強化はインターネット/イントラネットへの対応部分である。

まず、Workstation4.0 では、インターネット・ブラウザであるマイクロソフト Internet Explorer 日本語版や Peer Web Service などを標準装備し、インターネットやイントラネットを利用することができる。

Server4.0 では、Web サーバであるマイクロソフト Inetenet Information Server2.0 (IIS2.0) と新たに Domain Name System (DNS) を OS の機能の一部として標準で搭載しており、インターネット、イントラネットに対応している。

IIS2.0 が管理する Web ページに対する全文検索機能を提供するマイクロソフト Index Server 日本語版や、社内クライアントからインターネットへ透過的なアクセスを可能にするマイクロソフト Proxy Server 日本語版を利用することで、より柔軟なインターネット／イントラネットへの対応が可能になる。

さらに、新たに搭載された PPTP (Point-to Point Tunneling Protocol) により、高いセキュリティ機能を持つプライベート・ネットワークとしてインターネットを利用することができる。

以上のように、WindowsNT4.0 はインターネット/イントラネットへの対応を意識した OS である。

サーバOSでは、UNIXが高いシェアを持ってきた。とりわけ、米国ではWindowsNTのシェアは少ない。しかし、米国調査会社は、WindowsNTの急伸を予測している。ハードウェアも含めたPCサーバのコストパフォーマンスの向上がその背景にある。今回のWindowsNT4.0の登場によってサーバOS市場に大きな変化が起きる可能性がある。

また、コンピュータ・システムの普及という面でみてもインパクトは大きい。それはWindowsNT4.0のマーケティングに当たってマイクロソフトは中小企業やSOHO（スモールオフィス／ホームオフィス）をこれまで以上に意識しているからである。 WindowsNTは日本ではオフコンと競合する側面を持ち、オフコン・ユーザに対する展開を進めるためにマイクロソフトは日本におけるシステム・インテグレータと提携してきているが、今回の発表ではそうした策をさらに推し進め、量販店でもWindowsNTを販売していくことにしている。これは全国120店以上の店舗にWindowsNT4.0の展示コーナー「NTビジネス・ソリューション・スクエア」を設置して、ユーザがWindowsNT4.0を体験できる環境をつくるものである。また、店頭でのヘルプ・デスク・サービスも実施し、購入の際にアドバイスできるようにしていく。

1995年のWindows95発売はパソコン市場の飛躍的な拡大をもたらした。個人・ホーム市場の拡大が目につくが、企業のコンピュータ利用の普及も加速させた。WindowsNT4.0が、インターネット／イントラネット時代のサーバOSとして大きな地位を築くかどうかはさらに見守っていく必要があるだろうが、その考え方や市場への意欲はコンピュータ利用の流れを見定める上で大きなインパクトを持っていると言えるだろう。

ワイヤーフレーム・モデリング (Wireframe Modeling)

物体の形状を線分の集まりで表現したもの。

1次元要素の線分，および2次元および3次元要素の曲線によって構成される。線分の情報しかもたないので，面積や体積の概念がない。グラフィックスエンジンなどで高速に表示できるため，対話的にモデリングをしていく場合によく使われる。

WWW (World Wide Web)

インターネットに大きな関心が集まっている。インターネットの利用者は正確に把握することが難しいが、3,000万人とも4,000万人とも言われ、現在も爆発的に拡大している。インターネットは、世界中のさまざまなコンピュータに蓄積された情報を簡単に引き出せるだけでなく、地球レベルでコミュニケーションできることが大きな特徴である。

しかも、かなり手軽に情報を発信できる。いままで数多くの人々に情報を発信するにはマスメディアを使う、つまりコストがかかるという条件があったが、パソコンとソフトがあれば基本的には世界に向けた情報発信がインターネットによって可能になった。

しかも、インターネットで扱える情報は文字や数字というキャラクター・ベースだけではなく、写真やビデオ、サウンドまで含めた情報・マルチメディア情報なのである。インターネットはマルチメディア時代を先取りしていると言える。

そうしたインターネットの特徴を最もよく現しているのがWWWサーバという機能である。

WWWはWorld Wide Webの略で、「世界中に張り巡らされた情報の蜘蛛の巣」という語感を持っている。WWWサーバでさまざまな組織や企ニのさまざまな情報が提供されている。

WWWサーバの基本的な利用としては活動内容の紹介がある。例えば、米国ではホワイトハウスが運営するWWWサーバがある。これにアクセスすれば米国大統領の講演や公文書など、ホワイトハウスに関する情報がネットワークを通じて得られる仕組みだ。クリントン大統領の家族の紹介も見ることもできれば飼い猫の声を聞くこともできる。

また、企業では半導体メーカがWWWサーバで人材募集を行っているケースもある。学生は電子メールで自己紹介を企業に送るわけである。

日本でも米国と同様に会社案内や商品案内をWWWサーバで提供する試みがスタートしている。それを発展させれば宣伝になっていく。いわば、世界に顔を向けた独自の宣伝番組を持つことと同じになる。

ある会社では、海外進出に伴う現地パートナーの募集にインターネットを利用している。また、日本企業でも人材募集がインターネットで行われるケースが出てきている。

こうしたインターネットあるいはWWWサーバの利用を促進させた大きなきっかけがある。それがMosaicと呼ばれるソフトウェアである。

Mosaicは、WWWサーバを活用するための検索ソフトである。Mosaicを使うと画面上のアイコンをクリックするだけで情報を選択することができる。さらに、現在アクセスしているWWWサーバから関連する情報を持っている別のWWWサーバへマウスをクリックしながら簡単に移動して情報を見ることができる。

こうして次から次へとWWWサーバの情報を見ていくことを米国ではネットサーフィンと呼んでいる。

このMosaicはインターネット上で手に入れることができる。MosaicにはUNIX版、マッキントッシュ版、Windows版があるが、米国で大きなシェアを持つWindows版Mosaicが登場してか

らインターネットが爆発的に拡大したと言われる。

このように、さまざまな情報を提供するWWWサーバと、パソコンでWWWサーバの情報を簡単に手に入れることができるMosaicの登場はインターネットの「車の両輪」とも言えるだろう。

Mosaicは日本語版が登場している。また、最近ではMosaicではないがインターネット接続ソフトをあらかじめ内蔵しているパソコンも登場している。

95年度のパソコン市場は94年度をさらに上回ると予想されているが、日本におけるインターネットに対する関心の高まりに加え、インターネットにアクセスするソフトの充実もパソコン市場拡大の大きな要素になると言えるだろう。

X.509 に基づく厳密認証

国際標準である X.509 ではコンピューターネットワーク上の認証基盤を確立し、より確かなセキュリティサービスを実現するために、公開鍵暗号方式 を用いたフレームワークとその周辺のデータ構造を定めている。

(1) 公開鍵暗号

公開鍵暗号方式では、データを暗号化、複合するのに対になった二つの鍵を用いる。暗号化のための鍵は公開鍵といわれ、文字通り通信相手に公開する鍵である。複号のための鍵は個人鍵といわれ、他のユーザに知られないように所有者が保管、利用する。

秘密情報を送りたい場合は、相手の公開鍵を入手、情報をその鍵で暗号化して送る。暗号化した情報を複合できるのは、公開鍵に対応する個人鍵を持つ受信者だけである。公開鍵暗号アルゴリズムでは、公開鍵から個人鍵を求めるのは計算量的に不可能であり、攻撃者を含めて暗号鍵を公開して大丈夫である。

(2) ディジタル署名

X.509 では公開鍵暗号方式を認証技術に利用している。つまり、送信者は情報を個人鍵で暗号化して送り、受信者は送信者の公開鍵で情報を正しく複号できることで、情報が送信者から送られてきたことを確認するのである。正しく複合できるのは、公開鍵に対応する個人鍵で暗号した場合に限られるので、本人から送られたものであることが確認できる。

ここで問題となるのは、公開鍵暗号アルゴリズムは他のアルゴリズムよりも暗号化、複号の速度が三桁程度遅いことである。そのため、ハッシュ関数(一方向関数、指紋関数)と公開鍵暗号を組み合わせて、利用者の作成した情報を認証するディジタル署名を使用する。ハッシュ関数とは、任意のデータを一定長(128 ビットや 160 ビット)のハッシュ値(指紋)に変換する関数である。同じハッシュ値を持つ二つのデータを捜し出したり、ハッシュ値が同じになるようにデータを変更したり、あるハッシュ値になるデータを見つけだすのは計算量的に不可能だといわれている。以下にディジタル署名の概要を示す。

署名者はデータのハッシュ値を求める。
署名者はハッシュ値を個人鍵で暗号化して、データのディジタル署名を 生成する。
署名者はデータとディジタル署名を相手に送る。
受信者(署名検証者)はデータのハッシュ値を求める。
検証者はディジタル署名を署名者の公開鍵で復号し、ハッシュ値と比較する。
復号結果とハッシュ値が同じなら署名は正しく情報は署名者が作成、送信　　したものと判断す

る。
ディジタル署名によりデータの作成者が確認できるが、同時にデータが改竄されていないこともわかる。データやディジタル署名自体が第三者によって改変されていれば、署名を復号したものとデータのハッシュ値が異なるからである。

(3) 証明書
ディジタル署名で復号に用いられる公開鍵は、正しいものでなければ正しく相手の確認が行なえない。X.509 では公開鍵証明書発行局(Certification Authority、CA)を用いた正しい公開鍵の入手、配付方法の枠組みを定めている。つまり、CA という信用できる第三者が発行した証明書をディレクトリと言われる電子電話帳から入手し、その中から公開鍵を取り出して、利用するのである。証明書には、公開鍵とともにCAのディジタル署名が施されており、他者が偽造することはできない。CAの公開鍵さえあれば相手ユーザの正しい公開鍵を入手することが出来る。

ＪＤＫ（Java Developers Kit）

ＪＤＫとは、Java Developers Kit の略でして、直訳するとＪＡＶＡ開発者用キット、つまり、いわゆる開発環境です。

これさえあなたのマシンに入れておけば、すぐにでもアプレットが作れるというお得なもので、その上タダです(笑)

これは、雑誌の付録やダウンロードする事により入手でき、使用期限なども、一切ありません。

では、ＪＤＫには何ができるのか、詳しい使い方や説明は後の講義に譲る事にして、とりあえずざっとご説明しましょう。

■ＪＤＫにできる事

ＪＤＫにできる主要な事

javac（コンパイル）・・・あなたが書いたファイルを実行できる状態(class)にします

java（実行）・・・アプリケーションを実行する。

appletviewer（実行）・・・アプレットを実行する。

Applet と Application の違いは後述します。

その他の機能

javap（デコンパイル）・・・class ファイルを、ソースの状態に戻します。

javadoc・・・ドキュメントの自動生成

これ以外にも、いくつかありませが、通常使用するのは javac, java, appletviewer の３つのみです。

JNI (Java Native Interface)

JNI というのは Java Native Interface の略です。要するに、JAVA と C や C++で書かれたコードとの架け橋をしましょうということです。既に JAVA 以外で開発したコードがある場合や、高速で計算したい場合とかプラットホームに依存する処理が必要な場合に JNI を使います。

==========

(Java ネイティブインターフェイス仕様)

概要

JNI を使用すべきシチュエーションとしては、以下のようなものが考えられる。

・アプリケーションが、標準 Java クラスライブラリがサポートしないような機能を必要とする場合。
・既に異なる言語で記述されたライブラリが存在し、Java コードからアクセスしたい場合。
・実行時間を高速化すべき部分を、低レベルの言語により実装したい場合。

JNI を利用してプログラミングすることにより、以下のようなネイティブメソッドを使用することが出来る。

・Java オブジェクトを生成、検査、更新する。(配列や文字列を含む。)
・Java のメソッドを呼び出す。
・例外をキャッチ、スローする。
・クラスをロードし、クラスの情報を得る。
・実行時タイプチェックを行う。

ネイティブメソッドを Java へ実装するインターフェイスとして JNI の他にもいくつか存在する。特に、JDK1.0 ネイティブメソッドインターフェイスでは、 Java のコアパッケージでウインドウの描画や通信インターフェイス等の機種に依存する部分の実装が行われている。それ以外にも Java の機能の相当多くの部分がネイティブメソッドにより実装されているものと思われ、ネイティブメソッドのサポートは Java の機能の一部分であるだけではなく、核に近い部分を構成するものの 1 つであると考えられる。

XHTML (eXtensible HyperText Markup Language)
XHTML は、HTML 4.0 [HTML] を XML 1.0 [XML] のアプリケーションとして再定式化したものである。

XHTML 1.0 は、Strict, Transitional, Frameset という HTML 4.0 の 3 つの DTD に対応して、3 つの XML 名前空間を規定する。この 3 つの名前空間はそれぞれ独自の URI によって識別される。

XHTML 1.0 は、HTML を拡張およびサブセット化する将来の文書型のファミリーの基礎となるものである。この考え方は、将来的な方向性に関する節でさらに詳細に論じられる。

1.1 HTML 4.0 とは何か
HTML 4.0 [HTML] は、国際規格 ISO 8879 に準拠した SGML(標準的汎用マークアップ言語)アプリケーションであり、広くワールド=ワイド=ウェブの標準的なパブリッシング言語として認められている。

SGML は、マークアップ言語、とりわけ電子文書交換や文書管理、文書出版に使われる言語を記述するための言語である。 HTML は、SGML で定義された言語の一例である。

SGML は、1980 年代半ば以来広まっており、きわめて安定的なままでいる。この安定性の多くは、この言語が機能に富みつつ柔軟でもあるという事実によるものである。もっとも、この柔軟性は一定の対価によって実現されるものであり、その対価とは、ワールド=ワイド=ウェブを含め多様な環境で採用することの妨げとなるレベルの複雑さである。

HTML は、元々の認識では、文書の専門家でない者による利用に適した、科学やその他の技術的な文書の交換のための言語たらんとするものであった。 HTML は、比較的単純な文書を制作するのに適した構造的タグおよび意味論的タグの小集合を規定することにより、SGML の複雑さの問題に対処した。文書構造を単純化することに加えて、HTML は、ハイパーテキストのサポートを追加した。マルチメディア機能が後に追加された。

驚くほど短時間で HTML は広くポピュラーなものになり、急速に元の目的を越えて成長した。 HTML の起こり以来、(標準としての)HTML で利用したり、HTML を垂直的で高度に特化されたマーケットに適合させたりするための新しいエレメントが急速に開発され続けている。この大量の新しいエレメントが、異なるプラットフォームにわたる文書についての互換性問題をもたらしている。

ソフトウェアおよびプラットフォームの双方の異種混合性が急速に広がるに伴い、これらのプラットフォームで利用する「クラシックな」HTML 4.0 の安定性は幾分限定されることが明らかである。

1.2 XML とは何か

XML とは Extensible Markup Language™ の短縮表記であり、eXtensible Markup Language [XML] の頭字語である。

XML は、SGML の複雑さの大半を取り除きながら SGML の力と柔軟性とを取り戻す手段として考えられていた。 SGML の制限形態であるけれども、にもかかわらず XML は、SGML の力や豊富さのほとんどを保ち、なお SGML の一般的に使われる機能のすべてを残している。

有益な機能を維持しつつ、XML は、適したソフトウェアの制作や設計を困難かつ高コストなものとする SGML の複雑な機能の多数を除去するのである。

1.3 XHTML はなぜ必要か

コンテンツ開発者が XHTML を採用するべき主な理由は 2 つある。

第一に、XHTML は拡張可能であるよう設計されている。この拡張性は、文書が整形式でなくてはならないという XML 必須事項を当てにしている。 SGML の下では、新しいエレメントグループの追加が DTD 全体の変更を意味することがよくあった。 XML ベースの DTD では、既存の DTD に追加されるために要求されることは、新しいエレメントセットが内部的に一貫性があり、整形式であることだけである。これが新しいエレメント集合体の開発や統合を著しく容易にするのである。

第二に、XHTML はポータビリティをねらって設計されている。非デスクトップのユーザエージェントを利用してインターネット文書にアクセスすることは、ますます増えるであろう。ある推計が示すところによると、2002 年までにインターネット文書閲覧の 75%がこれらの代用プラットフォームで実行されるという。ほとんどの場合、これらのプラットフォームにはデスクトッププラットフォームの計算力をもっていないであろうし、不整な HTML も、現在のユーザエージェントならば調整する傾向があるが、これらのプラットフォームはこれを調整するように設計されないであろう。実際、これらのユーザエージェントは整形式でない XHTML を受け取っても、単にその文書を表示しなくてよいのである。