講演

Forum on Information Technology（FIT2002）基調講演
平成14年9月26日（木） 於：東京工業大学大岡山キャンパス

New Frontiers for Practical Computing

実際的なコンピューティングに向けての新たなフロンティア

Alan Kay
翻訳：小川 彩子

《コンピューティングの停滞》

数ヵ月ほど前に，米国の National Research Council で会議が開かれました．この会議中には，コンピュータの商業化が始まってから 20 年近くもの米国におけるコンピューティングの停滞について，活発な議論がありました．現在，コンピューティングは退屈で創造性のない分野になっています．その事実を明らかにし，コンピューティングを次の段階に進めるために，なにをすべきか，なににお金をだすか，私のような人を約 60 人よんで，議論が 3 日間続いたのです．

《初期のコンピュータ》

まず，コンピューティングの歴史における偉大な挑戦の数々を思い返してみましょう．1961 年，40 年前に，私は大学でプログラミングを学んでいました．当時初めて使ったコンピュータは IBM 1401 で，8 キロバイトの RAM を持ち，カードリーダとプリンタが付属していました．このマシンはオペレータがスイッチを入れると動き出すのですが，プログラムを入力してデバッグできる時間は 1 日に 3 分程度でした．クラッシュするとそれまでのプログラムがすべて無駄になってしまい，翌日またやりなおさなければなりませんでした．このシステムの OS（Operating System）は 1 キロバイトのメモリを使う，当時としては非常に高性能なものでした．コンピューティングの出発時点では，コンピューティングは，物理的には大きく，論理的には小さなものでした．

その次に使ったコンピュータはさらに巨大な B220 というマシンでした．Knuth も使いました．なんと 10 倍も大きく，0.1MIPS でした．

その 2 年後，1965 年ごろには最初期のスーパーコンピュータにかかわりました．Control Data 6600 というマシンには，0.5 メガバイトのメモリと 10 枚のディスクが格納されており，アーキテクチャのデザインには RISC を使っていました．

《ARPA での先端的研究》

1966 年に私は大学院に入学しました．当時のコンピューティングをめぐる状況は現在よりもずっと制約の多いものでした．コンピュータは大きく，プログラムは小さいことが当然でした．プログラムは単純なブロックで構成され，バグがあってもプログラム自体が小さいので追跡できました．当時のプログラムの大半はマシン語で書かれたものでした．

しかし，私はこの大学院時代に ARPA リサーチコミュニティでの研究に参加して，さらに先進的なコンピューティングに触れることになりました．ARPA では，

- コンピュータグラフィクス
- コラボレーション
- ハイーパーリンク

パーソナルコンピューティングの研究が始まっていました．また当時，チップのシリコンは 2 年で 2 倍に増えていました．

《Sketchpad》

それでは，当時の最先端のテクノロジを紹介していきましょう．まず，Sketchpad が実現されたのは，ちょうど 40 年前の 1962 年のことでした．このシステムは Ivan

Sutherland によって開発されました．最初のコンピュータグラフィクスの様子を知るために，当時の記録映像を見てみましょう．

Sketchpad に使われたコンピュータ[☆1]はこの会場よりも大きいもので，インタフェース部分は 2 つに分割されていました．右側でラフなスケッチや直接の操作を行い，左側で必要な処理を選択するようになっていたのです．このソフトウェアは非常に高性能で，直線はもちろん，曲線や相互に直角な線や平行な直線，破線なども描けました．また，プロッターはソフトウェアで制御されていました．

デモを見てみましょう．図形の拡大縮小から回転や複写など，できることはいろいろあります．

木ネジを描いてみましょう．1 つ木ネジができあがれば，もう 1 つ別の木ネジが欲しいときは，いまではクラスといわれるマスタを，インスタンスにコピーして大きさを調節します．さらに，当時としては驚くべきことに，ある図形を別の図形の画面にまとめることも可能でした．

この，最初のグラフィックシステムでは，現在流通しているグラフィックシステムとまったく同じことができるわけではありません．しかし，方向性は同じです．1962 年の Sketchpad は現在のコンピューティングと同じ流れの中にあります．そしてパーソナルコンピュータと巨大なメインフレームも同じといえます．1962 年当時からみた現在，だけで終わるのではなく，さらにこれからのコンピュータの未来へと続く流れがあるのです．2002 年の現在でも，コンピュータはいまだ発展途上にあるといえます．

現在のグラフィックシステムの性能は，Sketchpad で実現されていた以上に良くなっているとはいえません．私が思うに，1 つには，技術者がグラフィックスキルよりもむしろディスプレイの開発に力を注いできたのがその理由でしょう．計算機が高速になり，紙と鉛筆の時代からみると，処理能力はずっと高くなりました．でも，機能自体はほとんど変わりません．

私が ARPA に加わって最初の週に見たのが，この Sketchpad でした．私はデータ構造や数値の勉強にずいぶん時間をかけました．そして，今日私たちがオブジェクトと呼んでいる存在を理解しました．Sketchpad は数値やテキストを扱えるだけでなく，描画手続きも持ち，プログラムへのポインタやクラス階層もありました．

いったい，1962 年という年に，本格的なグラフィクスを発明し，最初のオブジェクト指向のソフトウェアシステムを発明し，最初のダイナミック問題解決システムを発明することは，どうして可能だったのでしょう．まったく驚くべきことです．当時の開発者たちは非常に努力していたのです．

今日と当時のコンピュータをめぐる状況の違いは，現在では苦労することもないようなさまざまな事柄について，当時は一生懸命習得しなければならなかったということなのではないでしょうか．今日の商業化されたコンピュータ業界では，多くのことが簡単な問題になっています．

《A Happy Thought：希望 1966》

その直後，1965 年頃，私は Simula を学ぶことになりました．Simula は，私がそれまで出会ったどんなプログラミング言語とも違っており，奇妙な英語で書いてありました．しばらく Simula に取り組んだ結果，Simula は Algol というプログラミング言語と同じ種類の構造を持っていることが分かりました．

この事実から，私はあることを思いつきました．私はもともと生物学を研究しており，虫を専門にしていました．そして，コンピュータでも生物を模倣してみたらどうだろう，と考えたのです．私は，この発想は新しい概念かもしれないと思いました．実際にコンピュータを昆虫のように動かせるのではないか，すなわち，あるコンピュータを他のソフトウェアコンピュータに接続して内容を受け渡し，そのソフトウェアコンピュータからソフトウェアコンピュータにまたその内容をコピーする…といったようにするのです．この概念は ARPANET，現在のインターネット上の動作に似ていました．この発見は，私の中で鍵となる重要な経験の 1 つでした．

[☆1] MIT の TX-2 計算機．

《Bob Barton と B5000（Burroughs）1960 頃》

大学院時代に，私はまた，さらに高いレベルの言語を実行するために設計されている専用のハードウェアの存在を知りました．現在私たちは，仮想マシン上で論理的なバイトコードを扱うのが普通です．しかし，1961 年頃の最初期のマシンは，何千もの実際のバイトコードで構築されていたのです．今日，私たちは JAVA や Smalltalk を高速で実行する CPU を入手できます．しかし，1961 年にはそれができたのです．これはよく考えてみるべき事実でしょう．

ハードウェアが論理的になるにつれ，性能の多くの部分が実際に動かすジョブ以外の部分に割かれるようになりました．そのため，現在ではハードウェアの速度は何百万倍も速くなったのですが，一方効率という点では昔のアーキテクチャの方が何万倍もよかったのです．

《JOSS（Johniac Open Shop System）1962-70》

最初期のオペレーティングシステムの中で，特に高性能だったのは Lampson らによる 1965 年の Genie でしょう．

そして最初のすばらしいエンドユーザシステムは，JOSS でした．JOSS は，本当によくできたユーザインタフェースを備えた最初のユーザシステムでした．1966 年の Cliff Show の言明は，ユーザインタフェースの本質を言い当てています．「Engelbart が作成したユーザインタフェースは，何百ものちょっとした作業のために設計されている．」これに対して，現在のユーザインタフェースの設計は，ほとんどがひどいものです．何百もの大仕事のためにできているのです．

《計算の Maxwell 方程式》

高レベル言語の基本である Lisp は，1960 年ごろから John McCarthy によって取り組まれてきました．

コンピューティングに関する私の最も思い入れが強い経験に，ある日曜の午後のできごとがあります．私はその午後中を，『LISP 1.5 Programmer's Manual』の 13 ページ目の下半分を学ぶことに費やしたのです．この半ページの中には，その後のすべての高レベル言語にある，真理と力と原理が基本的なかたちで含まれていました[☆2]．

プログラミング言語は 60 年代のたまものでした．今日の商業的なコンピュータ業界では，Lisp，Smalltalk，Algol のような言語では誰もプログラムを書きません．

《Wes Clark と LINC（Laboratory Instrument Computer）March 1962》

やはり 1962 年のことです．私は LINC という巨大なコンピュータを使うようになり，Sketchpad をこのコンピュータに移植しました．これが私の購入した最初の個人用コンピュータでした．LINC は個人用としては十分小さいマシンでしたが，使えるネットワークはそのサイズ以上に大きなものでした．グラフィクス・ディスプレイや仮想メモリを持ち，今日のパーソナルコンピュータに存在するすべての要素を持っていました．

《Engelbart の NLS（On-Line System）》

ちょうどそのころ，Engelbart がマウスを発明しました．マウスの使用は，いかに人がコンピュータを利用するかについての，また人々がコンピュータをどのように用いるべきかについての，まさに画期的な発明でした．1968 年のデモをお見せしましょう．このデモでは，マウスで線画を描いたり，マテリアルからのリンクを移動したり，などの動作が紹介されています．

Engelbart のマウスシステムの応答速度は 1 秒以下の

☆2 13 ページ下半分には evalquote の定義がある．

高速なものでした．現在のマウスのインタラクションの速度は当時より少し速くなっています．そうはいっても，Engelbart の使っていたシステムは，現在のものと比較してずっと性能が低く，さらにタイムシェアリングシステムだったことを考えてみましょう．

今日のマシンは 400MIPS で何ギガバイトものメモリを持つようになりました．そしてたいていのことには 1 秒以下で応答を得られます．しかし，いったい，なぜそのためにマシンパワーを使うのでしょう？

みなさんの答えは聞けませんが，私の答えは，Engelbart などの人々は 1 秒以下の応答を欲しがる人々だったから，というものです．高速な応答を欲する人々がいたからこそ，応答速度を重視したのです．私には，その理由がよく理解できます．しかし私自身は，どんなすばらしいマシンでも必ず 1 秒以下の応答を得たいと思っているわけではありません．他にもっとすばらしい使い道があるからです．

マウス以上にすばらしい Engelbart の発想は，グループで協同作業を行うシステムでした．パーソナルコンピューティングや単独の作業を超えた，グループによる知の増幅器を作り上げたのです．1968 年のデモをお見せしましょう．このデモでは，デモを行われたサンフランシスコから 40 マイルにあるパロアルトの Bill Paxton と共同作業実験を行っています．コンピュータとディスプレイにカメラを設置しているので，向こう側の人の顔とマウスカーソルが見え，音声も届きます．

《Flexible Personal Computer 1967-9》

1966 年に，Sutherland は Computer Environment の研究を始めました．それは人間対人間の環境を使って始まりました．インタラクションのために共同作業のできる環境が必要でした．ですから，グループコンピューティングの衝撃力はマウスよりも大きかったのです．共同作業のためのさまざまな努力を紹介しましょう．

《RAND Tablet 1964》

まず，RAND Tablet ペンベーストシステムを紹介しましょう．これは 1964 年に登場した最初のペンベーストシステムでした．デモの中では，ボックスを作図する，その中にコメントを描く，ボックスの大きさを変える，複数のボックスを接続する，ボックスの場所を変える，新しいボックスを作成する，などの動作が行われています．RAND Tablet ペンベーストシステムは今日から見ても優れています．

《フラットスクリーン登場 1966-8》

また，1966 年から 8 年ごろには，フラットスクリーンディスプレイもできました．

《Seymour Papert と LOGO Fall 1968》

1968 年の Seymour Papert による LOGO は非常に重要です．Seymour Papert が実証し，かつ実現したのは，計算機は大人だけのものではなく，子供のためのおもちゃとしても意味があるということでした．それも単なる一方向性のおもちゃでなく，子供が，微分幾何学を使って強力な概念を習得するための機器を作れるおもちゃだったのです．たとえば，小さな子供，6，7 歳の子供に円を描くプログラムを作らせてみましょう．

子供は何度も何度も向きを変えながら進むことを繰り返し，円を描くのです．実際に私がやってみると，完全な円ができます．子供に円が描けるのは，子供が微分幾何学を分かったから，子供のからだが知っているからです．数百年の歴史を持つ数学を子供も理解できます．

LOGO というすばらしいおもちゃは，大人のためのプログラムでもありますが，同時に子供が習得するための道具でもあるのです．

数学は，子供たちが科学という概念を理解するために非常に役に立ちます．なぜなら，それは科学を使い始めるための地図のようなものだからです．まったく数学ではない素材を使って子供たちをこの科学の地図にひきつけるのです．

この Papert のプロジェクトは完全に私の人生を変えました．私は，コンピュータは未来の生活の中では何のためのものになるのか，について考えるようになりました．

《A Happy Thought 1968》

私は，1968年のThe Dynabook: An Instrument Whose Music is Ideasというエッセイの中で未来の2人の子供のイメージを描写しました．子供たちは，自分たちの居場所を得て，私が今お見せしたようなもので学び，実行を通じて物理を学び，コンピュータを使って，ワイヤレスネットワークを通して通信しています．私は昔はこんなことが実現するとは考えていなかったが，しかしここで挙げた技術のすべてが実際に利用できるようになっています．ペンベーストシステムも存在するし，オブジェクト指向のOSも存在するし，もっともっと他にもマシンを構成するすばらしいものが存在しています．たとえばフラットスクリーンディスプレイもあります．これが実際にあり得る光景なのです．そして明らかに，最も難しい問題は，このこの空間全体（ディスプレイのスクリーン）が何であり得るかを理解することです．ここで描かれる世界は，ディスプレイの上だけではなく，ここにいる人間とコンピュータの向こう側の人間との間のインタラクションにあるのです．これは非常に難しい問題であり，実際のところ，ほとんどの人は十分分かっているとはいえません．学習という概念は理解しづらいものなのでしょう．

私が描いた考えを次の段階に進めるための材料は，Xerox Parcにありました．私たちが今日使っている技術の多くはXerox Parcの最初の5年間の成果なのです．現在の存在するような種類のコンピュータやスクリーン，インタフェース，アプリケーション，プリンタ，ローカルエリアネットワーク，インターネットは，すべてこの1971年から1976年の期間に生まれました．

私がそれ以来今日までずっと不満に思っていることは，このような60年代や70年代に生まれた概念は，いまだに重要ですばらしいものであり続けているというのに，実世界においてはこれらの概念は時代遅れだと思われて，次の段階のアイディア，そのまた次の段階のやり方があるはずだとされることです．そして実際そうなっています．つまり商業的コンピュータ業界です．

ユーザインタフェース設計に関して，もう1つここで付け加えておきたいのは，ユーザインタフェースにおいて，制御（control）のパネルから，習得（learning）のパネルに向かうことです．

実際に今日MS Windowsのシステムを使うユーザは，講習に行かなければならないほど複雑なユーザインタフェースを使うことになります．本来ユーザインタフェースは自分で習得できるように設計されているべきなのに，現在のユーザインタフェースは非常に退化している，と私は思います．

《Interim DynabookとAlto 1972-3》

ほとんどのユーザインタフェースが習得を考えて設計されるべき理由の1つは，子供のためです．将来のコンピューティングは電話産業主導で始まるように見えました．たしかに電話産業ではあらゆることが可能ですが，このようなマシンを使う人々は，コンピュータ産業をビジネスだとする考え方を強制されてしまいます．理想のマシンという概念に関して，あるマシンのデモを紹介しましょう．Interim Dynabookの1973年の記録です．このマシンのディスクはは200メガバイトあり，大きなディスクパックでしたがアニメーションはよいものでした．

このマシンでのプログラムの例としては，15歳の男の子による回路設計プログラムがあります．また，12歳の女の子が作ったメニューグラフィクスもあります．彼女は，四角をコピーしたり，さまざまな色を使ったり，いろいろな作業をしています．

このマシンのアプリケーションおよびディスプレイのプログラムSmalltalkは，すべて学齢期の子供のために作られていました．Smalltalkこそ，私がコンピュータの理想のために努力した結果でした．1970年代には何百ものマシンを用意して，学校などに設置しました．

《TechnologyとそのPowerful Mediaの間のギャップ》

しかし，私は心配するようになりました．西洋の印刷術では，印刷術は1450年ごろに発明されましたが最初に作られていたのは手書きの聖書を模倣したものでした．そして，本当の印刷術革命は150年後に起こり，ここではじめて人々は印刷の真の力を知り，模倣に使わなくなりました．

私はコンピューティングも同じ道をたどるのではないかと心配しています．最初のころ，私たちは巨大で高価なコンピュータを使っていました．すなわち印刷におけ

る聖書のようなものです．そしてラップトップが導入されてノートのように使われるようになりました．紙を模倣することは，完全にコンピュータの能力の損失です．紙の模倣の大きな例に，米国の何百もの企業において使われているスプレッドシートの 70% が 1 つの計算式も持たず，合計をとる以外の使い方をされていない事実があります．スプレッドシートを重要にしている機能はぜんぜん使われていません．

印刷術とコンピューティング，どちらの革命にも，この残念なタイムラグがあります．人間はどんな技術にせよ，その最も強力な部分に慣れるまでには時間がかかるのです．

《Grand Challenges Today》

現在におけるコンピューティングの重要な挑戦を考えてみましょう．実は，60 年代や 70 年代に始まり，この 20 年間に目的とされてきたことは，すべていまだに挑戦を必要としています．

- すべての人のためのパーソナル／グループコンピューティング
- 進んだエンドユーザオーサリング性能
- あらゆる活動のための実時間シェアリング
- 本よりも小さく，人間に読めるコード
- 60 年代の ARPA などによる研究・啓蒙のための財源の復活

これらはすべて 30 年前に発生した問題で，今もなくなっていません．

ここで私が注目したいのは，子供，です．なぜなら，私は多くの場面で，コンピュータはまったく大人のためのものとなっていると思うからです．

《6 歳児と数学》

コンピュータは商業的コンピューティング業界ではビジネスなので，啓発的ではなく，利潤追求型の使い方をされています．しかし，子供はコンピュータの利用に関しては大人よりずっと強力なので，もっと子供や子供のすることに注意を向けるべきです．

たとえば，コンピュータを使わずに，教師の力を借りて 6 歳の子供ができることを紹介しましょう．このデモ

は，10 から 15 年前のオープンスクールの様子です．この先生は，四角，菱形，台形のようなタイルを使って，授業をしています．まず子供たちに，タイルを寄せ集めて，さまざまな大きさの同じ形の図形を作らせて，次にそれぞれの大きさに必要なタイルの数を記録します．タイルの数の表を比較して，子供たちはタイルの個数と図形の大きさの関係に 1 つの規則があることを発見するのです．

6 歳の子供の多くは，教師が手助けしてやれば，最も強力な数学である二階微分方程式を理解できるのです．大きさ 1 のは 1 個のタイル，2 のは 3 個多くて，全部で 4 個のタイル，次は 5 個多くて，全部で 9 個のタイル．どの形もおなじ規則です．この先生は本当にこのことを理解しており，子供たちを愛していました．これによって私たちの，6 歳の子供に何ができるか，についてのイメージが完全に変わりました．

《Etoys》

子供がコンピュータを使って科学を学ぶための Etoys を紹介しましょう．まず，車のアニメーションです．まず車を書き，タイヤをつけます．それから適切なプログラムを選択すると車は前進します．円を描くプログラムを適用すると今度はぐるぐる回ります．別の円を描いて車のハンドルにすることもできます．スクリプトも作れます．道に沿って車を走らせてみましょう．車の前部に道の反対色を使って反発させることで，道に沿って進むようになります．この車を魚に変えて，対称形の図をかわりばんこにアニメーションさせると，魚が泳ぎ出します．

次は，Sam's "Face Ball" という，顔を描いたボールが弾むアニメーションです．アニメーションさせる絵を多くすれば動きがよりスムーズになり，速度も変えられます．

次は Sampling です．音の高低や速度を変えて，さま

ざまに変化させられます．シンセサイザーとしても使えます．

《Speed と Accelleration》

自動車のアニメを使って，速度と加速について学んでみましょう．車を走らせて，一定時間ごとに位置を点で記録します．速度が一定だと点は等間隔に並びます．速度を徐々に速くすると点が徐々に離れていきます．点の間の距離を比較すると，距離の増加に一定のパターンが発見できます．

《10 歳の児童に科学ができるか》

さて，10 歳や 11 歳の子供は本当に科学ができるのでしょうか．ボールを下に落とす実験のデモをご紹介しましょう．実際にボールを下に落とす実験を見て，落としたときの加速に規則性を発見できる子もいます．しかし，実験の様子をビデオに撮って，一定時間ごとの画像を並べたものを用意すれば，時間ごとの落下距離を簡単に計測できます．四角を使ってボールの間を測って，それからすべての四角を重ねてみましょう．すると四角の大きさの差がほぼ一定であるパターンが見えます．ボールの落下の様子が車の加速と同じことが分かります．

U.S. の子供は物理は得意ではありませんが，計算機でのシミュレーションを使うと非常によく理解できるようになります．

《WaterBaloon The Alien》

Etoys の 1 つ，WaterBaloon The Alien では，放物線をとぶ水風船とエイリアンは常にぶつかることが分かります．月着陸船の原理が分かるようになります．

その他にも，並列計算や 3 次元計算を使って，月面着陸ゲーム，ありのシミュレーション，木の成長モデル，シャークなど，さまざまな Etoy が作られています．

3 次元アニメでは，3D ドメイン，3D ワールドを作れます．デモをしてみましょう．画面に霧を入れると，もっとリアリスティックになります．ここは海の中です．海蛇もいます．ここに自分で魚を描いてみると，3D フォームになります．これは日本の東大または慶應の教授である五十嵐先生の作ったプログラムです．

《日本における Etoys －京都 CAMP Center》

2001 年の春の日曜日の午後に，京都の科学技術館でも Etoys を使ったワークショップを行いました．ワークショップの中で，子供たちはお互いに教え合い，コピーし合ってプログラムを作成しました．ここで大切なのは子供が子供を手伝ったことです．教育を変えるものは，子供たちが助け合うことなのです．

Engelbert が夢見たグループコンピューティングは，数千万人単位の大きなグループでも共同作業できるものでした．助け合うことがこの問題を解決することになるかもしれません．

最後に，実験的なエンドユーザオーサリングのデモをお見せしましょう．2 台のディスプレイで同時に操作します．両方のディスプレイはそれぞれ自分の視点になります．私のディスプレイには黄色いマフラーのペンギンが見えます．これはとなりの彼女を見ているわけです．私自身はもう一方のディスプレイの中の青いマフラーのペンギンです．3D 空間の中を見渡すと看板が並んでいます．看板の中に入ってみると，その中も別の 3D 空間になっています．これがダイナミックなシミュレーションの 1 つの方法です．子供にもぜひ使ってほしいものです．

ここでは，さまざまな 3D 空間を扱えます．数学的なオブジェクトの操作も可能です．たとえば三角錐を回転させることもできます．また，この空間の中で 2D のプログラムも動かせます．京都の学校で使ったソフトウェアはフリーで使用できます．

コンピューティングの未来に向けて，コミュニケーションを大切にしていきましょう．

（平成 14 年 12 月 10 日受付）