2019.8 のブログ：「IT 事件史（1990 年）：Σ計画失敗が判明」を読んで
（→ http://www.1968start.com/M/blog/index.html#1908 ）の別紙

IT 事件史（1990 年）：Σ計画失敗が判明」を読んで

中所 武司

■本題の記事

日経ＢＰの記事（2019.8.21）
https://tech.nikkeibp.co.jp/atcl/nxt/column/18/00215/080700039/
「Σ計画」失敗が判明した 1990 年、DOS/V がひっそり誕生」

【記事概要】
・通商産業省が 1985 年から 5 年計画で進めていた国家プロジェクト
「ソフトウェア生産工業化システム構築計画」（250 億円の予算）が
主な成果を出せずに 1990 年に終わった
・このΣ計画は、「1990 年には大量のプログラマーが不足し、
深刻なソフトウェア危機が到来する」との 1984 年の予測を受けて、
ソフトウェアの設計からテストまでの開発工程の機械化が目標だった
・国内大手ベンダーやユーザ企業の約 200 社が参加

■ソフトウェア工学の視点でのコメント

・1980 年代は、増大するアプリケーション開発要求が大きな課題となり、
その解決策としての先進的なソフトウェア開発環境が必要とされた。

・1991 年の私の以下の情報処理学会の解説論文にも、関連する記述がある。
http://www.1968start.com/M/bio/ipsjpaper/chu91ipsj.pdf
「エンドユーザコンピューティング －ソフトウェア危機回避のシナリオー」
（情報処理，pp.950-960、1991.8）

「2 章：ソフトウェアの問題」の「2.1 節：ソフトウェア危機の歴史」の
「（２）1980 年代－量の問題」において、以下の記述あり：
『計算機システムの普及とともに、開発すべきソフトウェアの量が増大し、
情報処理技術者の不足が第二のソフトウェア危機をもたらした。』
『通産省では、1990 年には情報処理技術者が 60 万人不足（需要：160 万人、
供給：100 万人）という推定に基づき、1985 年にはΣプロジェクトを発足した。』

・当時のソフトウェア開発環境展において、主要メーカーが
Σワークステーションの特設コーナーを設置して、デモを実施していたが、
先進的な印象は受けなかった。

・当時、第２次ＡＩブームと相まって注目されていた、同じ通産省の
第五世代コンピュータプロジェクト（予算 570 億円）に比べて、
Σプロジェクトは地味な印象だった。

・1990 年以降のソフトウェア生産技術に対しては、オブジェクト指向技術と
オープンシステム化の影響が大きかったと思う。

■ソフトウェア開発環境の歴史

拙著「ソフトウェア工学　－オープンシステムとパラダイムシフト－」
（朝倉書店、1997.5) を引用して説明する。

●2章「ソフトウェア危機の歴史」の2.3節「量の問題」において、以下の記述あり：

『規模の問題の後に量の問題が続いた。1980年代にはコンピュータシステムの普及に
伴って開発すべきソフトウェアの量が増大した．その結果、情報処理技術者の
不足という事態が生じ，開発すべきソフトウェアのバックログの増大という
第２のソフトウェア危機をもたらした。』

『このような情報処理技術者の不足に伴う技術者数の急増は、必然的にその平均的能力の
低下という２次的な問題をも招いた。開発すべきソフトウェアがより複雑化し、
情報処理技術者により高い能力が要求されるにもかかわらず、その平均的能力が
低下している現象を筆者は図2.3に示すように「教育的ギャップの拡大」と呼んでいる。
そのため、ソフトウェア工学の分野では、開発者の作業効率向上や個人差の縮小を
目的とした統合的ソフトウェア開発支援環境や部品化・再利用を目的とした
パッケージ化技術が研究され始めた。』

『このような課題への対応としていくつかの政策もとられた．通産省では，1985年には
ソフトウェア生産工業化計画の一環として分散型開発環境構築を目的とした
シグマプロジェクトが発足し，1990年を目標に個人用高機能ワークステーションと
そのためのオペレーティングシステムやネットワークの開発が行われた.』

●10章「開発環境の概要」の10.2節「歴史」において、以下の記述あり：

a．50-60年代（抜粋）：
『この時期には，プログラムのソースファイル操作用のユーティリティや
コンパイラなどのツールが利用されていた.』

b．70年代前半（抜粋）：
『下流工程で利用するエディタやデバッガなどのプログラミングツールの
個別開発が中心に行なわれた.』

c．70年代後半ー80年代前半（抜粋）：
『当時の統合プログラミング環境は，使い勝手の良いユーザフレンドリな
ユーザインタフェースとプログラミング用データベースをツール間で
共通化することによってツールを統合する方式が基本であった.』

表 10.1　ソフトウェア開発環境の変遷

年　代	特　　徴
５０年代 ： ６０年代	●バッチ型プログラミングツール ・高級言語コンパイラ、ソースファイル操作ユーティリティ
７０年代 前半	●対話型プログラミングツール ～バッチ型処理から対話型（ＴＳＳ）処理へパラダイムシフト～ ・上流工程：ドキュメント作成支援 ・下流工程：ツールボックス（エディタ，デバッガなど）
７０年代 後半 ： ８０年代 前半	●統合プログラミング環境 ～開発手段（ツール）の高機能化から 開発工程（プロセス）の定式化へパラダイムシフト～ ・ユーザインタフェース、プログラミング用データベース共通化 ・特定プログラミング言語向き環境（構造エディタなど）
８０年代 後半	●統合開発環境 ～製造工程(how-to-make)中心から 計画・分析・設計工程(what-to-make)中心へパラダイムシフト～ ・ＣＡＳＥツール ・ユーザインタフェース統合、リポジトリによるデータ統合
９０年代 前半	●オープンシステム ～開発工程（プロセス）の定式化から 開発対象（プロダクト）の標準化へパラダイムシフト～ ・開発環境のプラットフォーム、ミドルウェアの共通化 ・分散環境化
９０年代 後半	●エンドユーザコンピューティング ～生産者中心の視点から利用者中心の視点へパラダイムシフト～ ・コンポーネントウェア、ビジュアルプログラミング

d．80年代後半（全文）：

『ソフトウェア開発の真の難しさは上流工程にあるという認識から，再び上流工程の
方法論や技法のツール化が積極的に試みられた．how-to-makeが主体の製造工程から
what-to-makeを重視する計画，分析，設計工程へと関心が移っていった．このような
パラダイムシフトのなかで，上流工程支援を含む統合開発環境は統合CASE
（ Computer-Aided Software Engineering ）と呼ばれるようになった．
従来の統合プログラム環境に比べた特徴として，ユーザインタフェースに関しては，
ワークステーションやビットマップディスプレイの進歩によりビジュアルで
使い勝手の良いグラフィカルユーザインタフェース（GUI）が一般化した．
データの管理に関しては，ソフトウェアエンジニアリング用のデータベースである
リポジトリにより全工程を管理するライフサイクル支援がある.』

e．90年代前半（全文）：

『情報化社会の進展と共にコンピュータシステムが複雑化していくにもかかわらず，
情報処理技術者の不足（または技術不足）などにより，アプリケーション・
ソフトウェアのタイムリーな開発が難しくなってきた．この時期には，従来のように
特定のメインフレーム（汎用大型コンピュータ）上でだけ稼働するアプリケーションを

個別に開発するメインフレーム文化に代わって，異機種間での移行性や接続性を
重視したオープンシステム実現へと急激な変化が生じた．このオープンシステムの
普及とともに，CASEツールのソフトウェア・アーキテクチャをできるだけ
標準化しておくことにより，ツール間の連携処理やツール自身の移植性が向上し，
分散環境下での共同開発などが容易になってきた．
80年代に従来の開発手段（ツール）の高機能化から開発工程（プロセス）の定式化へのパラダイムシフトが生じ，環境という表現が一般化したことに対応させると，
90年代には，CASEツール，ミドルウェア，プラットフォームなどと呼ばれる基本的なソフトウェアが先行する形で，開発工程（プロセス）の定式化や自動化から開発対象（ソフトウェア）の標準化へのパラダイムシフトが生じ，オープンシステムという表現が一般化したといえる.』

f．90年代後半（抜粋）：

『ソフトウェアに関してもエンドユーザが直接利用するアプリケーションが
より重要となり、生産者中心の視点から利用者中心の視点へパラダイムシフトが
生じている。』

以上