ソフトウェア工学　46－20
（１９８６．２．７）

ユーザ・モデルに基づいたメカトロニクス開発環境の改善

石井　威望　　廣瀬　通孝　　小木　哲朗　　甘利　治雄
（東京大学　工学部）

１．序論

ソフトウェアの生産性が議論されるようになってから、かなりの時間が経過したが、決定的解決を得るには至っていないのが現状である。その理由の一つとして、多くの研究例が、実際の生産現場を対象としている点を挙げる事が出来る。この方法論は、現実性はあるものの、原理的に本質ではない要因を含むため、議論の一般性を欠く恐れがある。また、対象が余りにも大規模となるため、「仮説→実験→検証」といった自然科学の基本的方法論が取りにくくなるという点も無視出来ない。しかし逆に、研究が一般性を追求しすぎると、そこからの結論は実際の生産現場とかけ離れ、実用化出来ない恐れがある。

本研究では、この様な方法論による、長所、短所に留意し、生産の主体であるプログラマに注目した立場をとっている。更に、実験にあたっては、対象を制御可能な作業レベルまでスケールダウンしたが、そのためには、作業を出来るだけ簡単なモデル作業に置き換える必要がある。この場合、現実のソフトウェア作業と相似な点、異なる点を充分に認識しておく必要があることはもちろんである。この様な立場をとることによって、プログラマの心理的負荷等の重要な因子について、ある程度定量的な扱いを行なうことが出来る。

例えば、筆者らは人間の内面的状態を定量的に把握する手段として、生体情報の一つである心電図を用いている。心拍間隔（心電図におけるＲ波の間隔）の分散値を RRV値と呼ぶが、この値は人間の精神的覚醒状態を反映する量である事がわかっている。つまり、知的作業において作業負荷が大きくなるか、あるいは作業に対する覚醒度が大きくなると、RRV値 は小さくなる。この性質を利用すると、RRV値は、知的作業時における作業管理の指標として有望であると考えられる。

いくつかのモデル作業を通して、知的作業に対する人間の基本的特性を把握する研究を数年来進めてきたが、そこから得られるモデルは、かなり純粋化したユーザ・モデルとして考える事が出来る。本論文では、応答性、待ち時間、その他（例えば後述する思考の連続性の破壊など）主として人間の応答特性に関するユーザ・モデルを紹介する。更に、現実へのアプリケーション例として、本研究室で行なわれているメカトロニクス開発におけるプログラミング環境を例に取り、上記の視点に立った作業分析から、開発環境の改善を行ない、その評価を行なった。

２．応答特性に関するユーザ・モデル

プログラミング作業時において、ファイルの呼び出し、アセンブル等、計算機処理による待ち時間は、単純な時間の損失だけではなく、プログラマに与える影響を考えた場合、生産性を左右する重要な問題であると思われる。計算機からの応答時間が長いという事は、それによって作業時間が長くなる事はもちろんであるが、それ以上に思考の連続性が破壊され、知的作業の効率が著しく低下する事が懸念される。

２－１　応答時間による内面的影響

ここでは応答性、待ち時間等に注目した実験を行なったが、まず初めに単純な応答時間の遅れが作業者に与える影響を、心理的側面から明らかにする。モデル作業としては『数当て』という一種のゲームを採用した。これは、計算機からの応答を頼りに、３桁の数字を当てる作業である。例えば、"１２３"なる数字が答えであるとする（被験者はもちろんこの数字を知らない）。これに対し被験者が、"１３５"なる数字を入力すると、計算機から

の応答は、「正解した数：１個、数だけ合っている数：１個」という具合になる。

図１は、この作業において、作業者の入力に対する計算機の応答時間を０～６秒と変えていった時の RRV値を示したものである。図では、一定時間の作業中における、RRV値 のばらつきと平均値を示してあるが、応答時間が３秒と４秒の間では、作業者の受ける内面的影響に大きな違いがある事がわかる。速い応答時間では、RRV値は低く、一定の値を保つのに対し、応答時間が遅くなると、RRV値 は上昇し、値のばらつきも大きくなってくる。すなわち、この現象は、３秒以下では集中力が持続しているのに対し、４秒を越えると集中力は低下するばかりではなく、持続出来ない状態となり、思考の連続性に影響を及ぼしてくるものと考えられる。この限界値は個人により多少の差はあるが、およそ３～４秒という値を得た。

作業者が許容出来る応答時間は、作業の種類、規模、あるいは作業者の期待、達成感等により異なるが、集中力の持続という点から応答時間を捉えるならば、この３～４秒という値を一つの基準として考える事が出来る。

図２は、この時の RRV値の時間的平均を実際に観察した一例であり、前半は応答時間０秒、後半は応答時間１０秒の場合である。１０秒の応答時間では、０秒の時に比べて RRV値の変動が非常に大きい。すなわち、思考の連続性に注目した場合、その特徴は RRV値の大きさとともに、その挙動に現れてくる事がわかる。従って、思考の連続性についての議論をする場合、この変動量も一つの指標とする事が出来るが、ここでは RRV値の２階微分(DD-RRV)をとる事によってこの特徴を抽出する。

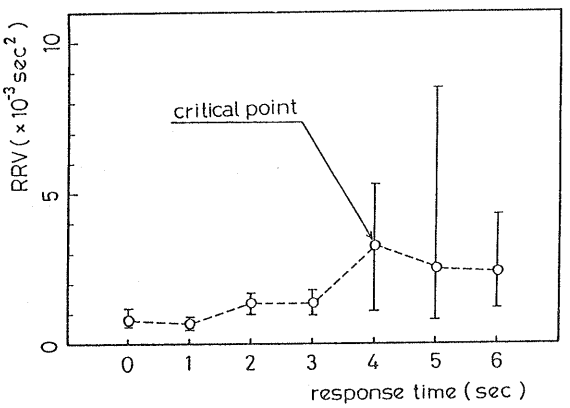


図１　応答時間と RRV値

２－２　応答時間による行動の変化

応答時間が、集中力の持続性等、作業者に与える心理的影響を示したが、それに加えて、作業者が自らの置かれた作業環境に合わせて行動パタンを変化させ、そのために作業能率に変化が現れる事にも注意しなければならない。例えば、待ち時間を要する作業等をなるべく避けたりする事は想像に難くない。ここでは、応答時間による行動パタンの変化に注目した実験例を示す。

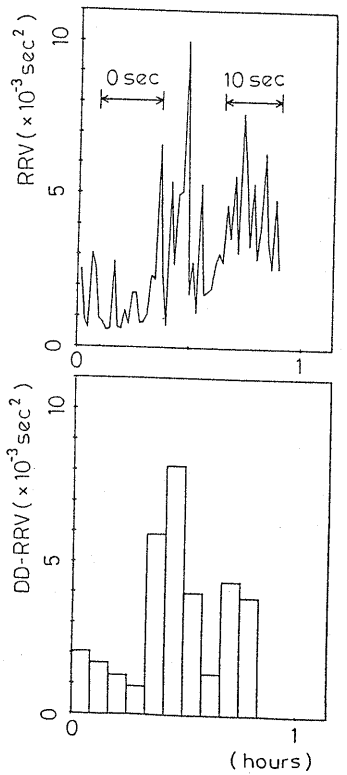


図２　応答時間による RRV値の挙動

1）実験例１（魔方陣の完成作業）

魔方陣と呼ばれる、縦、横の数の和が一定となる行列の中に、誤りを入れたものを訂正し、魔方陣を完成させる作業を第１のモデル作業とした。図３は作業の実例を示したものであり、(1)は正しい魔方陣であるが、(2)は○印で示した６箇所に誤まりを含んでいる。作業者は、helpキーを押す事によって、誤りの箇所を表示したヒント(3) を参照する事が出来るが、このhelpキーの応答時間により、作業者の行動がどう変わるかを観察した。図４は、この時の応答時間とヒントの参照頻度の関係を示したものである。応答時間が長くなると、ヒントの参照頻度は減少しているが、この行動の変化は５~１０秒の付近で現われている。

2）実験例２（座標合わせ作業）

グラフを表示する BASICプログラムの作成を第２のモデル作業とした。これは、画面上の任意の位置に表示された SIN曲線の上に、座標軸、目盛を付け加える作業である。ここでも、プログラムの実行時に待ち時間が設けてあり、この待ち時間によって作業者の行動が変わっていく様子を観察した。一般に、速い応答時間では、座標の計算をせずに少しずつ修正していくのに対し、遅い応答時間では、座標の計算を細かく行なった後にプログラム作業に入る様子が観察される。図５は、この行動をプログラムの実行頻度、あるいは実行間の思考時間として表示したものである。応答時間が長くなるに従って実行頻度は減少し、思考時間は増加しており、問題解決の方法が変化している事がわかる。

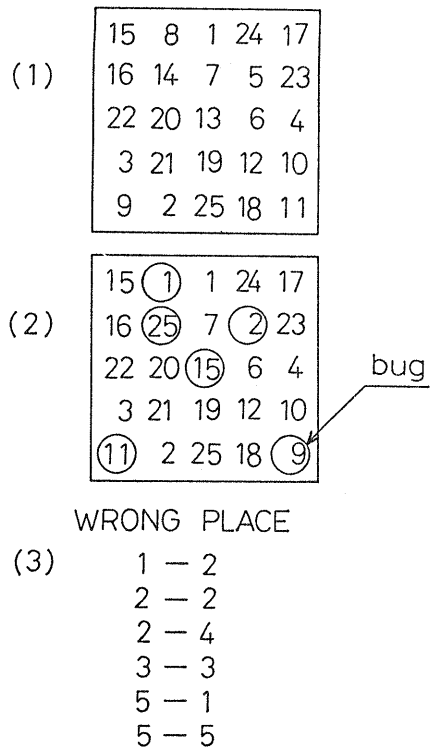


図３　『魔法陣の完成作業』

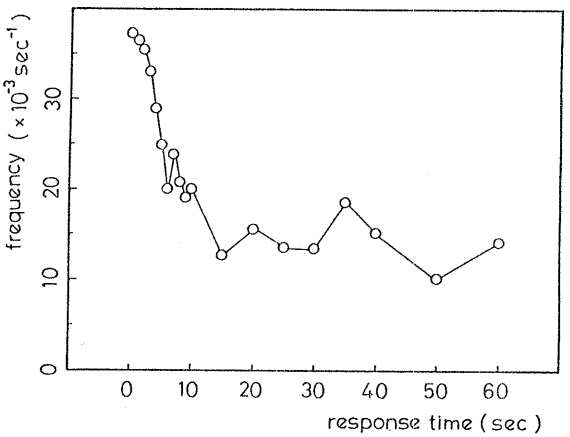


図４　応答時間による行動変化（実験例 1）

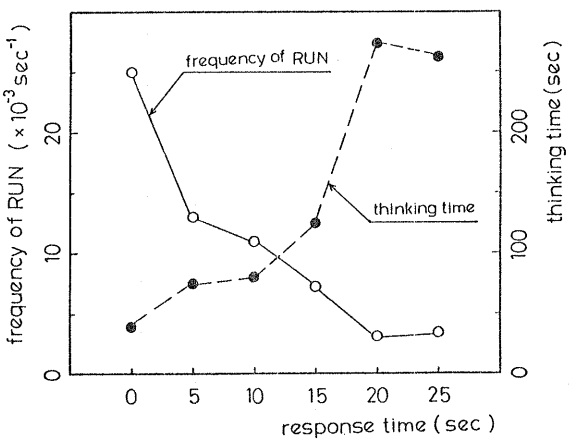


図５　応答時間による行動変化（実験例 2）

この作業者の適応のしくみを、人間の内面的状態から捉えてみる必要がある。図６は、『座標合わせ』の課題に対し、(A):応答時間０秒の場合、(B):15秒の応答時間に対し０秒と同様の方法を取らせた場合、(C):応答時間15秒の場合、の３通りについて、そのプログラム実行頻度と RRV値の関係を示したものである。15秒の応答時間に対し、行動を変えなかった場合(B)には、RRV値は高く、DD-RRVの大きさからも思考の連続性がかなり破壊されている事がわかる。しかし、行動を適応させていった場合（C）には、RRV値は低く、かつDD-RRVが低い事から、思考の連続性はかなり保たれているものと思われる。

この様に、待ち時間を避けるという作業者の行動は、思考の連続性を保つという適応に基づいた行動であると理解出来る。従って、環境の改善等を考える場合においても、そこから引き起こされる行動の変化を考慮に入れる必要がある。

２－３待ち時間の配分

応答性、待ち時間等の作業者に与える影響を考えてきたが、計算機の能力には限界があるため、応答時間等の縮小を実現しても、そのしわよせが、起動時間の増加等としてはねかえってくる恐れがある。また逆に、作業者の面から考えても、作業内容、あるいは作業規模等によって作業者の受ける心理的影響は異なり、同じ待ち時間であっても許容出来たり、出来なかったりする。

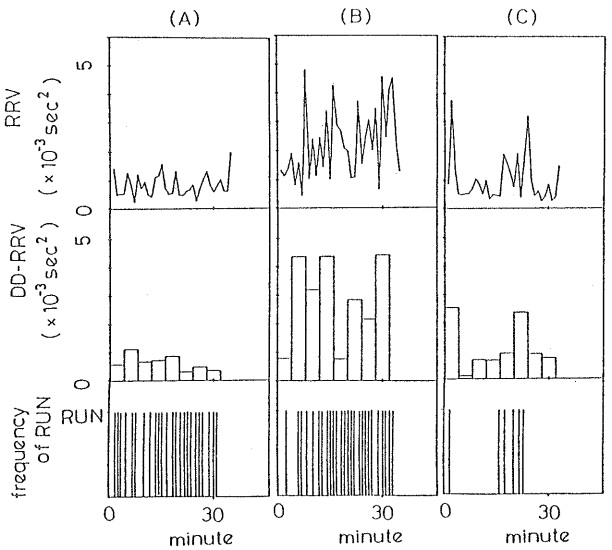


図6　作業者の適応と RRV値

従って、ある程度の待ち時間が避けられない場合、時間配分等、その与え方が問題となる。

ここでは時間配分に関する一つの例として、起動時間、応答時間に注目した実験例を示す。モデル作業としては、前述の『魔方陣の完成』作業を取り上げた。但し、作業開始時の起動時間、あるいはヒントの応答時間について、

A:起動時間１５秒，応答時間　０秒

B:起動時間　０秒，応答時間１０秒

という２種類の環境を用意した。作業は連続して行ない、１回毎に、A，Bどちらかの環境を選べるものとしてある。ここで、挿入された誤りの数に応じて作業者の選択する環境の変化を観察したものが図７である。なお、順序効果を排除するために、実験の順序はランダムとした。図より、誤りの数が多くなるに従って、選択される環境が(B)から(A)へと移り変わっていく事がわかる。この誤りの数は、作業の規模、あるいは作業の難易度に相当するものであると思われる。従って、作業規模が大きくなるか、作業が複雑になる程、応答性のすぐれたものが好まれ、作業規模が小さくなると、逆に作業者は応答性よりも起動時間のすぐれたものを好む事がわかる。

図８は、この両環境における現実に要した作業の完了時間を比較したものである。両環境での実際の完了時間の大小が逆転する条件下では、まだ作業者の選択には変化が現われておらず、作業者の主観的感覚にもとづく選択が、必ずしも完了時間の損得と厳密には合致していない事がわかる。この事実は、この種の作業においては、起動時間というものが、ユーザからみて過大に評価される事を意味している。これは、現実のツール設計においていわゆる、「システムの重さ」に対する示唆を与えるものであると考えられる。

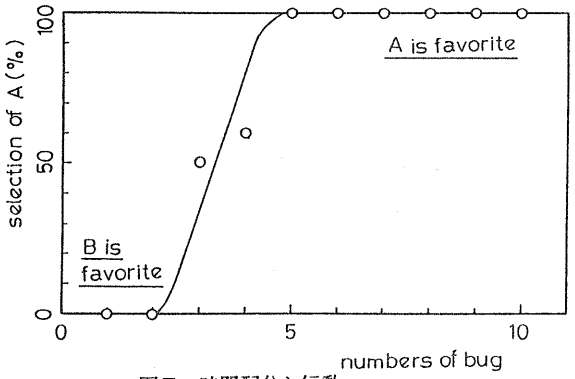


図7　時間配分と行動

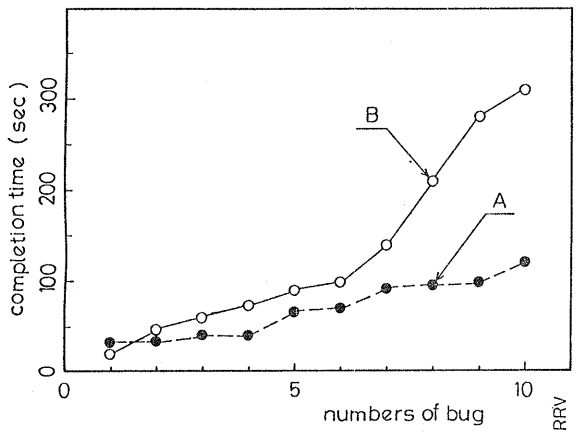


図8　時間配分による作業成績

2－4画面の切り換え等の影響

思考の連続性という視点から応答時間の問題を取り上げてきたが、実際の作業において思考が妨げられる要因は、単に時間の長さだけの問題ではない。時間の与え方等もそうであるが、ここでは画面の切り換えなど視覚的な面についても考えてみる。

モデル作業としては、再び『魔方陣の完成作業』を取り上げるが、ここではヒントを用意するのではなく、誤りの位置が異なるもう一つの魔方陣を用意して作業を行なわせた。作業者は、2つの魔方陣を交互に見比べながら作業を行なうことになるが、2つの魔方陣の与え方として、3通りの方法をとった。(1)は、別々の画面に用意し、画面の切り換えに10秒を要する。(2)は、別々の画面に用意するが、画面の切り換え時間は0秒とする。(3)は、同一の画面上に用意した場合である。図9は、この3通りの作業における行動パタンと、RRV値の挙動を示したものである。

DD-RRVから判断すると、(2)では待ち時間がないため、(1)よりは思考の連続性が保たれているが、(3)は更にすぐれていると思われる。この様に、応答時間としては0秒という理想的状況を実現しても、画面の切り換えなど視覚的な影響が思考の連続性に及ぼす影響は残される。

近年、高性能ワークステーション等において、マルチウィンドウの使用が普及してきたが、ここにおける(3)の作業行動は、マルチウィンドウにおける作業行動の一つのモデルとして考えることが出来る。従って、作業者の思考の連続性等の、心理的側面から捉えても、この様なマルチウィンドウにおける作業の有効性が示されたと考えられる。

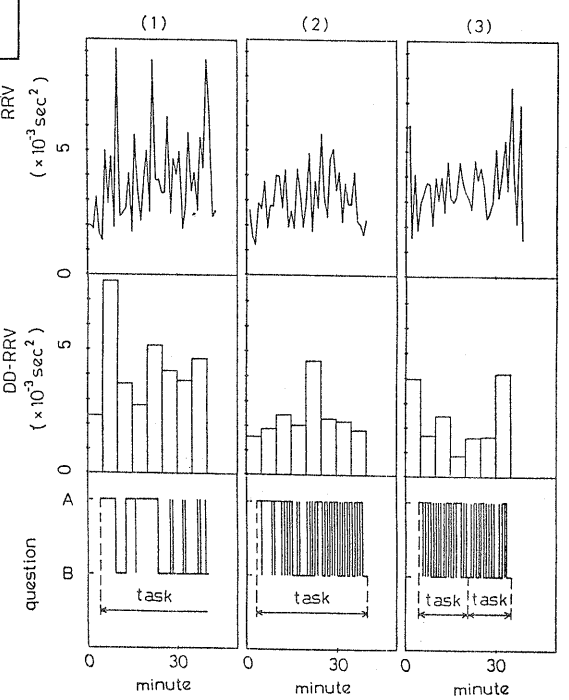


図9　画面切り換えの影響

３．　メカトロニクス開発環境の設計への応用

３－１　メカトロニクス開発環境

以上の知見を応用するための例題として、著者らが開発中であるメカトロニクス用ソフトウェア開発用ツールを取り上げる。

本ツールの設計にあたって留意せねばならない点は、メカトロニクス用ソフトウェアを対象としているため、開発用マシンとターゲットマシンが必ずしも同一ではない点、しかも、外部のハードウェアのデバッグとソフトウェアのデバッグが並行して行われなくてはならないという点である。すなわち、設計・製作・デバッグという一連の作業を、ソフトとハードの両面から考えながら進めて行かねばならない。

更に、メカトロニクスの分野においても複数のＣＰＵを使用した、分散型システムの設計が増加しつつあり、ソフトウェア開発において、ＣＰＵ間の通信・同期などを考えながら、複数のプログラムを同時に作成しなくてはならないという事が、開発上の大きなネックとなっている。

このような見地から考えると、従来の多くのシステムが採用しているソフトウェア構造は、必ずしも最適の物とは考えられない。

３－２　従来のツールによる開発作業

ソフトウェア開発におけるプログラマの挙動は、同じ作業をする場合でも、開発環境によって簡単にも複雑にもなる。ここではまずメカトロニクス製品の開発環境の効率化を進める上で、プログラマの挙動を分析し、それをもとに開発上の手順や無駄時間とその影響を考察した。

図１０に示すのが、従来使われてきた開発環境のプログラムの流れ、および制御の流れである。この環境においては、エディタ・アセンブラ・モニタという別々のプログラムがあり、それをＯＳ上で走らせるという構造である。

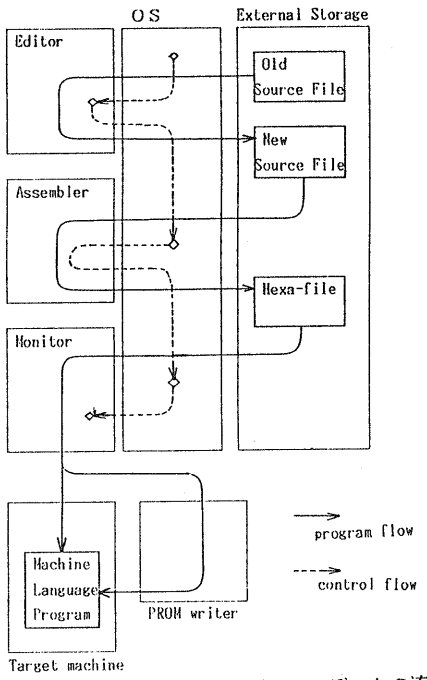


図１０　従来の開発環境とプログラム・データの流れ

まず、この環境下において、本研究室で開発中の１チップマイクロプロセサ i8086によるマニプレータ制御システムを例題として、実際のメカトロニクスシステムのソフトウェア開発作業時の、各プログラムのアクセス状況、所要時間、及び RRV値を測定したデータを図１１および表１に示す。

これより、以下のような改善すべき点を挙げる事が出来る。

（１）制御の流れは、ＯＳ上を中心としてエディタ、アセンブラ、モニタの間を行き来する。そのうえで、浪費される時間、すなわちコマンド投入やディスクアクセス等にかかる時間が、全体の６～９％に達している。図１１のＡの箇所などの RRV値を見ても、この行き来が多くなると、思考が中断している事がわかるように、実際に無駄時間が開発効率に与える影響は、純粋なロスタイムだけではなく、それ以上のものとなる。

（２）デバッグ時の手順が煩わしい。例えば、ソースコードをオブジェクトコードにアセンブルしてからバグを発見したとする

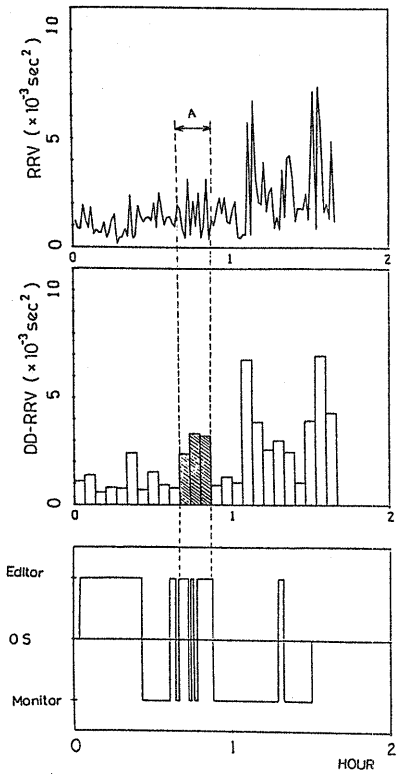


図１１　従来のツールによる作業の分析

表１　ソフトウェア開発作業の所要時間比　（単位　%）

	Editor	Assembler	Monitor	Others
first half	57.3	<0.1	20.4	8.6
second half	64.2	<0.1	29.4	6.4

と、再び最初のエディタによるソースプログラムのエディトから順にやり直さなければならず、プログラムの些細な変更にも、数々の手順と、それに伴う無駄時間がかかる。

（３）複数個のＣＰＵのプログラムを、並行して作成しなければならないので、前項までの問題点がさらに拡大、強調される。

３－３　メカトロニクスのソフトウェア開発ツールの設計

以上の結果を踏まえて、i8096 メカトロニクス製品のソフトウェア開発環境を設計し、XMAS/EVE (cross macro assembler system evaluation environment) と命名した。本ツールは現在開発中であるが、その設計において、先に述べたユーザ・モデルの考え方が、相当量反映されている。

XMAS/EVEの設計仕様および特徴的機能を説明する。

データの流れとプログラムの流れを、図１２に示す。本システムはＯＳレベルの機能を全て包含しており、一体としての環境を構成している。これにより、ターゲットマシン上のオブジェクトコードのデバッグをする場合においても、開発ツールからソースプログラムを通しての対話型の編集・実行が可能となった。

また、複数個のＣＰＵの並行プログラミングを効率化するために、複数個のソースファイルを同時に扱えるという機能を持たせた。これにより、１つのソースファイルのエディトの途中で、即座に別のファイルの参照やエディトに入る事や、複数のソースファイルにまたがる編集（複写、移動、ファイル分離、結合など）がコマンドモード上で簡単に出来る。

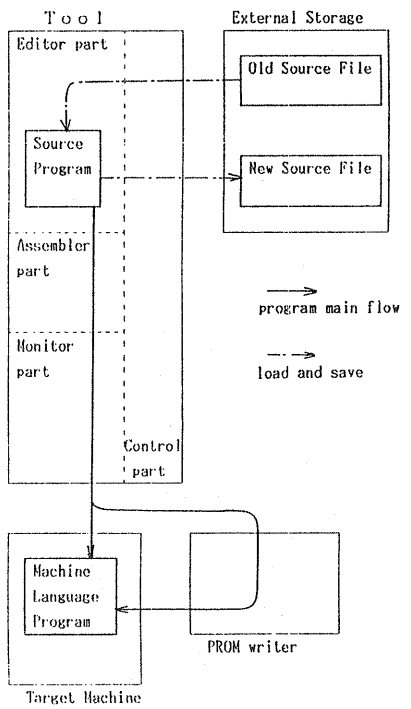


図１２　効率化された開発環境の設計

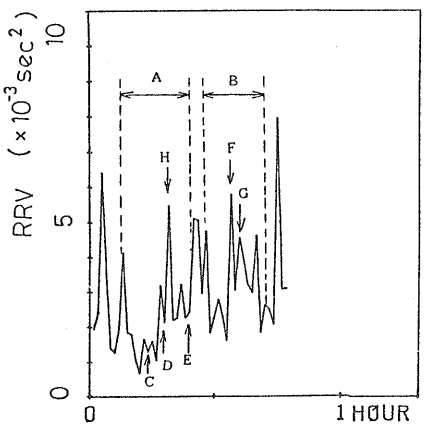
これらの実現にあたって採用した方式は、外部記憶からコマンドによりソースファイルを内部記憶に移す事によって、コマンドモードからのアクセスを可能とし、それらのうちの一つをカレントファイルと決め、現在のエディトの対象とする、というものである。

その他、ＣＰＵ間の通信・同期プログラムの開発を効率的に進めるため、計時機能をサポートした。この機能は、コマンドモードから始点と終点を指定する事によって、実現される。

以上のように、本システムは、そのままではかなり重装備な環境となるが、不要な機能の除去、立ち上げ時間の短縮など、2-4節の知見にもとづき、できるだけコンパクトな「軽い」ツールの実現を図った。

このツールによる開発作業の効率、あるいは作業者の状態を、従来のツールを使用した場合と比較対照するために、被験者に対し、簡単なプログラムをそれぞれのツールを使って製作させる実験を行なった。

結果を図１３に示す。RRV値を指標としてみると、従来のツールでは、プログラムからＯＳに抜けた時点で、RRV値が上昇しているが、このツールを使用した場合、トラブルで中断した点を除くと、低い値が続いている。



A　：新開発環境によるプログラミング
B　：従来の開発環境によるプログラミング
C,D,E：エディト終了、ターゲットマシンに実装
F,G　：エディタからＯＳ上に移る
H　：トラブルにより中断

図１３　新ツールと従来のツールの使用比較

すなわち、従来では制御が移るごとに緊張感が途切れ、思考の持続性に悪影響を及ぼしていたが、このツールにより、思考の連続性という点において、大いに改善された。

４．結論

幾つかのモデル作業から、応答性、待ち時間等に関する人間の応答特性というものを明らかにした。また、この作業者の基本的行動が、人間の心理的影響と深くかかわったものである事が、以前から報告している生体情報を利用する事によって示された。更に、実際の生産現場への応用を例として、メカトロニクス開発環境の基本設計に以上の知見を応用することを試みた。

参考文献

［１］廣瀬、石井：『ソフトウェアメトリクスにおけるミクロ分析の手法』、情報処理学会、ソフトウェア工学研究会 37-4

［２］石井、廣瀬、伊藤：『ソフトウェア生産工程におけるポインティングデバイスの性能評価』、情報処理学会第29回全国大会

［３］廣瀬、石井：『プログラマの状態モデル－生体情報の利用について－』、情報処理学会第31回全国大会

［４］石井、廣瀬、小木：『プログラマの状態モデル－実験的検証－』、情報処理学会第31回全国大会

［５］石井、廣瀬、池井：『自律分散型マニプレータ用制御計算機の設計と演算精度の検討』、日本ロボット学会第３回学術講演会 2103