ISSN 2186-5647

P-53

中小製造業における NC 工作機械のプログラム入力ミス軽減に関する研究

日大生産工（学部） ○佐藤 翔太 日大生産工 村田 康一

1. 研究の背景と目的

数値制御(Numerical Control: NC)工作機械のプログラム作成や、その確認の自動化は、コンピュータ支援設計(Computer Aided Design: CAD)や、コンピュータ支援製造(Computer Aided Manufacturing: CAM)の分野において多くの成果がある。しかし、経営資源に限りのある中小製造業においては、情報システムへの投資が、製品加工を行う工作機械・設備などのそれと比較し優先順位が低い。このためプログラム作成は、現場に頼らざるを得ないという企業も少なくない。また、NC 工作機械のプログラム作成は、国家資格である機械加工に関する技能検定の実技科目に取り入れられている。このような背景から、当該スキルをものづくりに必要なスキルと捉え、日ごろからその教育訓練や作業を推奨している会社も見られる。

一方で、製品不具合の発生は後を絶たない。特に、プログラムミスは、再加工など生産への影響が大きい。しかし、その原因追求は多くの場合、油断や疲労などといった人によるところに行きついてしまい、根本的な対策が難しい。

Shingo[1]は、統計的品質管理(Statistical Quality Control: SQC)のような製品群全体の品質を抜取り検査により推定する方法は、現場にある製品の状態や情報の一部わからなくしていると指摘し、全数検査による品質管理の方法論として源流検査とポカヨケ方式を導入し、不良ゼロの生産システムを構築した。源流検査は、ミスと不良を原因と結果の関係として明確に区別し、ミスを不良に転化しないように、ミスを検知し修正するための方法であり、「品質は現場で作りこむもの」という考え方に基づいている。ポカヨケ方式はミスを検知する仕組みであり、その情報により、機械の稼働や作業の進行を強制的に停止させる強制式と、情報提供のみの注意式の 2 種類がある。しかし、この仕組みは物理的なもの扱うオペレーションを対象にしており、プログラム作成など、電子的な情報を扱うオペレーションには例が少ない。

この分野に関しては、一定規模の情報システム開発において、障害を抱えながらもサービスを提供し続けること、すなわちフォールトトレラント性を高める構成が考えられている[2]。しかし、このような高度な知識を含むプログラムの習得やメンテナンスは、中小製造業には負荷が大きい。

本論文では、このような背景をふまえ、新規プログラム作成に焦点をあて、その作業を分解し、要素作業ごとに可能な限り高い頻度でミスに対するフィードバックをかけることができる方法を検討することを目的とする。また、この考え方をふまえ、簡単な入力ミス削減機能を付加したプログラム作成支援ツールを実装する。

2. プログラム入力におけるミスについて

本研究における企業 A との共同研究では、プログラム作成におけるミスについて、その事例、原因対策が次のように整理されている[3]。

＜プログラム作成におけるミスの例＞

例 1：加工対象と類似した製品加工のために過去に作成したプログラムに基づいて、そのプログラムを作成したが、更新が必要なパラ―メータの修正を忘れていた。

Study on Reduction of Programming Miss for NC Machine Tool in Small and Medium Manufactures

Shota SAITO and Koichi MURATA

例 2：図面に行われた修正の一部を見落とし、それをプログラムに反映しないまま加工を行った。
例 3：座標値を入力する際に、小数点の位置を間違えてしまった。
例 4：プログラム作成が間欠作業となり、中断前に作成したプログラムに反映すべきことを忘れてしまった。

＜プログラム作成ミスに至る原因＞
表 1 はミス発生時の状態とその人的要因をまとめたものである。

表 1. ミス発生時の状態とその要因

Ⅰ作成時の状態 / ⅡⅠの要因	忘れ・抜け	思い込み
スキル不足による（未熟さ）	○	○
慣れによる（油断）	○	－
忙しさによる（焦り）	○	○

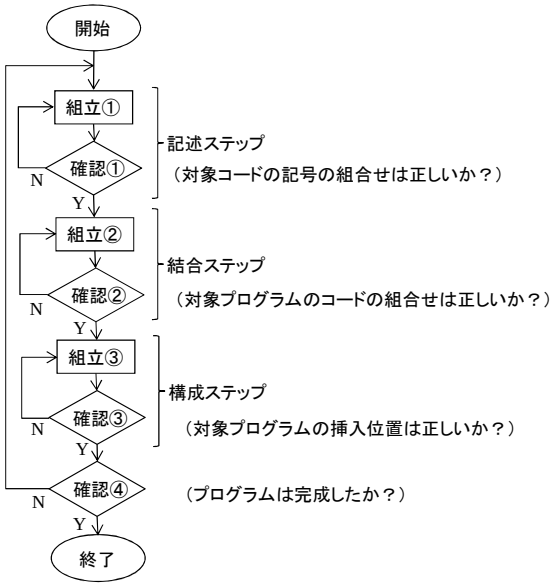
＜プログラム作成ミスに対する対策＞
・ミスが起こらない環境作り
 - 朝礼などを利用したポカミス情報・エピソードの共有化、熟練者から新人へのアドバイス
 - ポカミス発生者への面談
・作成方法の標準化とその徹底
 - プログラム作成法のルール化
 - ミスが起こりやすい作業の注意喚起表示
・作成プロセスでの確認作業への工夫
 - 図面へのマーキング
 - チェックシートによる確認作業の徹底
 - ダブルチェック（人別、時間別）
 - 加工前の空運転確認作業の徹底
 - 作業中断状態の見える化
・作成オペレーションの簡略化
 - プログラムモジュールの利用

3. プログラム作成プロセスの分析
完成されたプログラムは、①プログラムの行単位による分解、②1 行のプログラムのコード種類による分解、③コードの構成記号単位による分解、により体系的に分解することができる。なお、ここでいうコードには、G コード、M コード、座標（X 軸、Y 軸、Z 軸、A 軸、B 軸）、サブプログラムなどがあり、構成記号には、G や M といったコード名、コード番号、座標の＋やーといった符号や数値がある。また、分解と逆の流れ、すなわち、表 2 に示す組立①から組立③の順に行われる一連の作業がプログラム作成プロセスとなる。

表 2. プログラムの分解と組立

【プログラム】
↓↑
分解①↓↑組立③
↓↑
【1 行のプログラム】
↓↑
分解②↓↑組立②
↓↑
【コード】
↓↑
分解③↓↑組立①
↓↑
【構成記号】

上記の要素作業を順に進めた場合のフローチャートを図 1 に示す。各ステップは、各組立とその確認を行うサブステップにより構成される。
記述ステップにおいてはコードを作る。ここでの確認事項は、コード種類、コード番号、座標の符号、数値の桁数、小数点の位置、サブコードの種類などがある。結合ステップは、記述ステップで作成したコードを組合せ、1 行のプログラムにする。ここでは、コードの組合せが正しくなされているかということが確認事項となる。構成ステップでは、結合ステップで作成した 1 行単位のプログラムを体系化していく。ここでは、正しくプログラムが挿入されているかということが確認事項となる。また、全体として、プログラムに漏れがないかどうかについてもプログラム完成直前の確認事項となるであろう。



※ Y…Yes, N…No

図 1. プログラム作成フローチャート
（新規プログラム作成の場合）

4. プログラム作成支援ツールの実装

前節の考え方に基づくプログラム作成支援ツールを Microsoft Excel 及び Visual Basic for Applications (VBA)を用いて実装した。その概要を図 2 に示す。この中では、ステップごとに作業エリアがわかれており、それぞれを段階的に行うことができるようになっている。以下に各ステップの仕様について説明する。

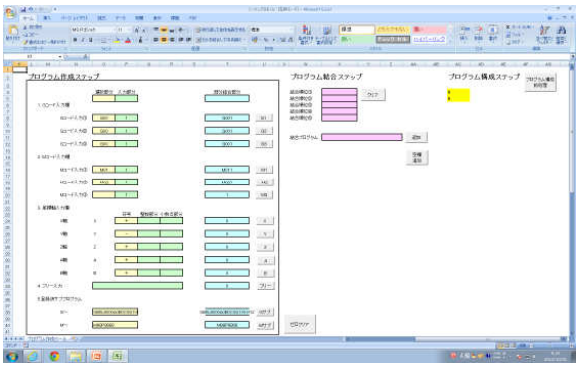


図 2. プログラム作成支援ツール画面

4.1 記述ステップの仕様

1) G コード、2) M コード、3) 座標軸、4) 登録済サブプログラム及び 5) フリー入力の 5 種類のコードカテゴリー別に作業エリアがわかれている。1) G コードと 2) M コードについては、コード番号を予めリスト化し、プルダウン方式によって選択するといった工夫がなされている。3) 座標軸については、軸の種類により入力エリアがわけられている。座標を表す記号は自動出力される。符号は選択式であり、数値は整数と小数をわけて入力し、小数点は自動出力される。これらにより軸の種類の入力ミスや小数点位置の間違いなどのミス軽減を狙っている。4) 登録済サブプログラムは、プログラムとしてモジュール化されたものを予めリスト化し、プルダウン方式によって選択する方式である。5) フリー入力は、1)から 4）で対応できない項目を入力するものである。

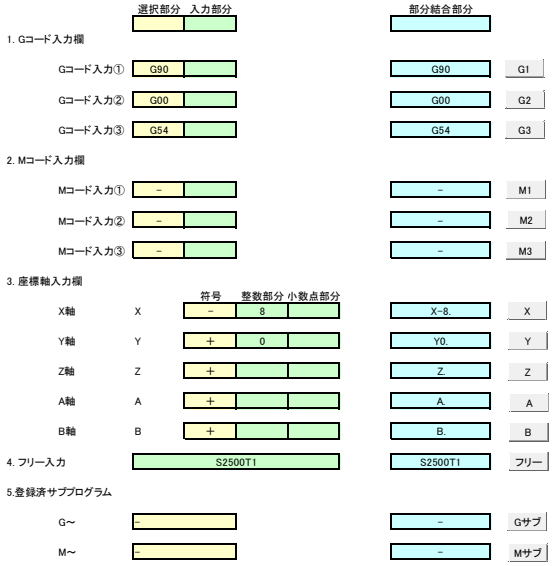


図 3. 記述ステップの作業エリア

4.2 結合ステップの仕様

1 行のプログラムに必要なコードを前ステップで作成した後、図 3 の右端にあるボタンを用いて結合順に作成済みコードを選択する。図 4 に示す結合ステップの作業エリアには、選択された順番にコードが表示され、それらのコードが自動的に結合される。

4.3 構成ステップの仕様

前ステップで作成されたプログラムは、図 4 に示す追加ボタンにより、図 5 のように挿入される。追加ボタンは 2 種類あり、プログラム挿入用と空欄挿入用のものがある。

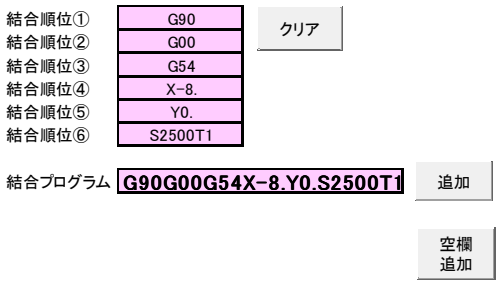


図 4. 結合ステップの作業エリア

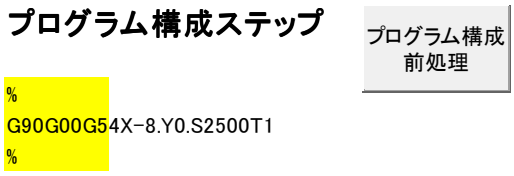


図 5. 構成ステップの作業エリア

4.4 確認プロセスと支援ツールの各機能との関係

図 1 に示した確認①から確認③と支援ツールの各機能との関係を表 3 に示す。後者については、エリア分割（ステップ別とコード別）、選択方式、自動出力機能、自働結合機能（結合順序表示、自働結合）、自働挿入機能の 7 種類がある。このうち、2 種類のエリア分割、結合順序表示、挿入機能により、段階的なオペレーションが可能になり、入力フローを意識しながらの作業が可能になり得る。また、選択方式、自動出力、自動結合により、入力作業負担の軽減につながるものと考える。

表 3. 各ステップと支援ツール機能との関係

ステップ名 / 機能	記述ステップ（確認①）	結合ステップ（確認②）	構成ステップ（確認③）
エリア分割①（ステップ別）	○	○	○
エリア分割②（コード別）	○		
選択方式	○		
自動出力	○		
自働結合機能 結合順序表示 自働結合		 ○ ○	
自働挿入機能（確認精度向上）			○

5. まとめと今後の課題

本論文では、新規プログラム作成の要素作業を体系的に分解し、可能な限り高い頻度でミスに対するフィードバックをかけることができる方法を検討した。また、この考え方をふまえ、簡単な入力ミス削減機能を付加したプログラム作成支援ツールを実装した。

現在、このツールの有効性について実験室内及び共同研究先において検証しており、以下のような課題がでている。今後これらに対応することにより、より実用性の高いものにしていきたい。

＜本支援ツールについて＞

- 記述ステップについて
  - 冒頭の 0 が表示できない。
- 結合ステップについて
  - 結合順位表示欄が不足している。
- 挿入ステップについて
  - 一旦作成した行を削除する機能が必要。
  - プログラムの転用機能が必要。

＜その他＞

- 過去のプログラムをベースにしたプログラム作成支援ツールがあると良い。

謝辞

企業 A の改善プロジェクトにおける議論から、本論文作成にあたり有意義な情報や知見を得ることができました。ここに感謝の意を表します。

参考文献

[1] Shingo, S., *Zero Quality Control: Source Inspection and the Poka-Yoke System*, Productivity Press, 1986.

[2] Parag, K. L.（著），当麻喜弘，玉本英夫，古屋清（訳），*フォールト・トレランス入門*，オーム社，1988.（原著：*Fault Tolerant and Fault Testable Hardware Design*, Prentice-Hall International, 1985.）