テーマ4

# 定着するローコード開発ツールによる生産管理システム開発

ほんま　本間 峰一

## ERPパッケージの導入トラブルが増えている

生産管理システムの構築作業は、市販のERP（もしくは生産管理）パッケージの生産管理モジュールを使って行うものだと思っている人が多い。ところがパッケージを用いた生産管理システム構築はトラブルを起こしやすいことがわかってきた。昨年末にはSAPの生産管理モジュールを導入しようとした某大手メーカー工場の本番開始が1年延期されたが、こうしたトラブルが相次いでいる。

トラブル発生の原因はさまざまである（**表1**）。その1つに、日本の製造業者は業務運営をパッケージの仕様に合わせるのが難しいことがある。パッケージの仕様に業務を合わせることが業務改革だといった話をまことしやかに流すIT関係者がいるが、実際にそんなことができる企業や工場はきわめて限られる。日本には受注生産型の工場が多く、取引先の意向に沿った業務運営が要求されるため、業務処理を変えるのは簡単ではない。たとえば海外では一般的ではない内示手配への対応が必要とされるが、海外製パッケージにはそんなあいまいな受注処理機能は搭載されていない。パッケージに大規模なカスタマイズを施して無理やり使ったり、在庫を積み上げたり、Excelを利用したりなどで何とか乗り切っている工場が多い。

多くの生産管理パッケージが採用しているMRPという生産管理ロジックが実用的でないという問題もある。MRPは、計画変更がほとんどない在庫補充型最終製品の生産工場のために生み出された生産管理ロジックである。当該市場を支配しているような大手企業が使うには問題ないが、取引先都合による計画変動への対応を余儀なくされる日本の工場の生産管理業務には向いていない。

日本の多くの工場は、計画変動対応力に優れた製番管理や製造ロット番号管理を使って生産管理を行ってきた。こうした工場が生産管理ロジックをMRPに変えたことで柔軟な変動対応ができずに製造現場が混乱を引き起こしている。現場の努力で混乱を回避してきた工場も多いが、最近は計画変動に加えて手配通りに部材を入手できないケースも増えており、現場の努力だけでは十分な納期管理ができなくなった。

**表1　主なパッケージトラブルの発生要因**

| トラブルの要因 | どんなトラブルが生じるのか |
|---|---|
| これからはパッケージに業務を合わせるのが基本だ | 自社内はともかく、取引先の業務要求までを合わせることはできない。受注生産企業では死活問題になることがあり、注意が必要 |
| 日常業務で忙しいので仕様検討する時間がとれない。現状業務に合わせてくれればよい | 利用者が十分に検討しない状態で新システムを開発し、後から「こんなはずではなかった」という不満が続出する |
| 最初にフィット・ギャップ分析するから大丈夫 | ギャップがあれば使わないのが基本なのに、気づいたらカスタマイズとアドオンの山になる（ベンダーはたとえギャップがあっても基本的に辞退しない？） |
| このERPはMRPというスタンダードな生産管理ロジックを採用している | MRPロジックは大手の最終製品メーカー対応で、日本企業に多い受注生産型企業には向いていない |
| 実績豊富なSI（システムインテグレーション）業者に委託するから安心だ | SIは宣伝文句にすぎず、パッケージの活用指導まではしないベンダーが多い（特に大手ITベンダーに注意が必要） |
| ベンダーのSEのスキルが心配だ | パッケージの機能は知っていても、生産管理理論や生産管理実務を知らないベンダーSEが増えている |

## ローコード開発ツールの活用が広がりつつある

生産管理パッケージに内在する問題を回避する手段として期

図1 ローコード開発ツールによるシステム生成イメージ

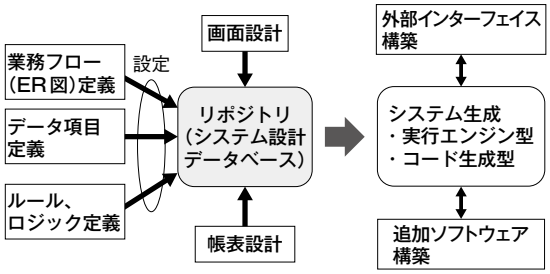



図2 ローコード開発ツールによる開発工数例

| 画面数 | 56 | テーブル数 | 35 | 帳票数 | 9 | CSV数 | 10 | バッチ数 | 4 |
|---|---|---|---|---|---|---|---|---|---|

開発工数自動積算

| 工数種別 | 計算根拠 | 工数(人月) |
|---|---|---|
| 設計工数 | 想定 | 3.0 |
| プロジェクト管理工数 | 想定 | 1.0 |
| 初期開発工数 | 工数積算計算式 | 6.9 |
| プロトによる修整工数 | 工数積算計算式 | 2.1 |
| 合計 | | 13.0月 |

待を集めているのが、パッケージではなく自工場の特性に合わせた柔軟な生産管理システムを独自開発しようという動きである。MRPが使えないので製造ロット番号管理による工程進捗管理システムを新たに構築するといった話だ。

ただし、一から個別にプログラムを開発していたのでは開発費用(開発工数)がいくらかかるかわからない。そこで注目を集めているのが「ローコード開発ツール」を使って安く開発するアプローチだ。特に中堅クラスの工場でローコード開発ツールを使って自工場に合った生産管理システムを個別開発する企業が増えている。筆者のクライアントでも複数の工場がローコード開発ツールを使ってオリジナルな生産管理システムを構築している。今後、業務システム構築はパッケージ利用ではなく、ローコード開発ツールを使って構築するという流れが定着していくと考えられる。

ローコード開発ツールとは、従来「超高速開発ツール」と呼ばれていたものとほぼ同じだ。業務システムをほぼ自動的に作成するツールを指す。ツール上に機能仕様を定義することでシステムは自動生成される。プログラムではなくシステムを自動生成するというところが、自動プログラミングツールとの違いだ。「MS Access」や「ファイルメーカー」などのDBシステムの作成ツールも広義のローコード開発ツールといえる。

ただし、本来のローコード開発ツールはシステムを作成するだけのツールではない。「リポジトリ」というシステム設計データベースで機能仕様を一元管理することを特徴にしている(**図1**)。ローコード開発ツールではリポジトリの設定内容を変えることでシステム生成する。これが単なるシステム作成ツールとの違いとなっている。

リポジトリで機能仕様を一元管理することにどういったメリットがあるか。最も大きなメリットは、システムのメンテナンス性が向上し、システムの改変が生じた際に開発した人間でなくてもシステムを修正できることにある。さらに、リポジトリを見るだけで最新の設計情報を取り出せるので、仕様変更が起こるたびに設計書を書き換えるといったこともしなくて済む。

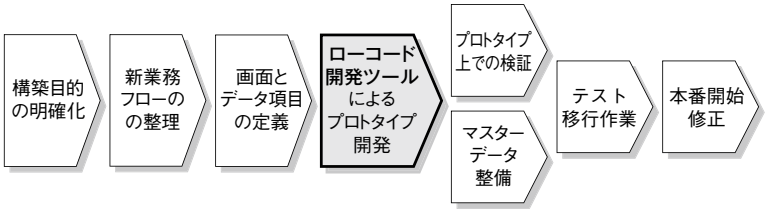
ローコード開発ツールはリポジトリにER図とデータ定義をベースに計算式や計算ルールなどの情報を登録することでシステムを自動生成する。開発方法はプログラミングの素人でも1カ月程度でマスターできる内容だ。

システム生成はプログラムコードを生成するものと、システムそのものを生成するタイプがある。プログラムコードが生成されるタイプでもコードをいじってしまうとリポジトリとの関連性が途切れるので原則はコードをいじってはならない。ローコード開発ツールではプログラム開発をなくせるので、システムの開発生産性は非常に高い。ただし、パッケージではないのでシステムの基本設計作業は自分たち、もしくはコンサルタントに手伝ってもらって行う必要がある。

## システムのプロトタイプ開発も容易に

前述のようにローコード開発ツールは開発生産性が高いので、システムを安く早くつくれる(**図2**)。各種調査でも従来の個別開発やパッケージのカスタマイズ(改造)開発に比べると半分程度の工数でシステム開発が可能といわれている。ただし、ローコード開発ツールの意義はそれだけではない。

**図３　パッケージ利用からプロトタイプ開発の時代へ**




**ローコード開発ツールの登場でプロトタイプ開発ができるようになった**

- ◆プロトタイプをそのまま修正していくことで、本番システムに仕立て上げていく
- ◆上流工程では基本となる業務フローを整理するだけで、細かな仕様検討はプロトタイプを実際に操作・検証しながら実施する。詳細な要件定義書はつくらない
- ◆プロトタイプをベースにした検証を行うので、画面確認だけでなく動作確認も容易にできる
- ◆修正可能な試作品により開発途中や本番開始後のシステム変更も可能となる
- ◆プログラム言語を使わないので、専門家がいなくてもシステム開発・保守作業ができる

**図４　プロトタイプで何を確認するのか**

●プロトタイプで確認したい事項

- ◆当該業務をシステム化する意味があるかどうかを確認し、ムダな機能や運用で改善すべき内容はシステム対象から外すようにする
- →　同時にシステム設計者の業務理解に対する勘違いがないかの確認も行う
- ◆新システムの導入目的と導入効果を見える化して、関係者で共有する
- ◆プロトタイプの業務フローで実際に業務が遂行できるかを検証する
- ◆プロトタイプを実際に操作して現行よりも手間や工数が増えてしまう業務がないかを洗い出す
- →　エンドユーザーにシステムの実物を見てもらうことで、追加したほうが効果的な内容を顕在化させる
- ◆システムを動かすために必要なマスター整備の重要性と必要性を理解してもらう

企業にとってはローコード開発ツールのメンテナンス性を活かしたプロトタイプ開発が容易にできることのほうがより重要だ。

プロトタイプ(試作品)を使って仕様や製造方法を検討するのは、工場が新製品を生み出す時に当たり前に行われている開発手法だ。ところが、システム開発現場でのプロトタイプ開発はほとんど行われてこなかった。システムベンダーからは設計段階で仕様を固めるのが一般的だと説明がされてきた。しかし、それは現実的ではない。開発が進むにつれて、追加要件による予期せぬ開発費用増加や稼働遅延といったトラブルが発生しやすい。

システム開発の世界でプロトタイプ開発が一般化しなかったのはプロトタイプ開発に適したツールがなかったためである。ベンダーによってはプロトタイプと称しながらExcelなどで画面イメージをつくってみせる程度の確認でごまかしてきた。それだけではプロトタイプ確認としては十分とはいえない。

ローコード開発ツールはシステム変更が容易にできるという特性を持っている。その特性を活かすことでシステム構築現場でもプロトタイプ開発が簡単にできる。これがローコード開発ツール活用の最大のメリットである。実際にローコード開発ツールを使ってシステム開発したプロジェクトの大半が、プロトタイプを使ったシステム開発作業を行っている。ローコード開発を単なる開発工数削減ツールと侮ってはいけない。

## プロトタイプの仕様は利用者が決定する

ローコード開発ツールを使った開発は**図3**の流れに沿ってプロトタイプ開発方式で行う。プロトタイプが何回も繰り返されるのを防ぐために、あらかじめプロトタイプ検証の回数上限を決めて行う方法もある。このアプローチをRAD(Rapid Approach Design)という。

ローコード開発ツールは業務パッケージソフトではないので、プロトタイプシステムの仕様は前もって利用者が決める必要がある。基本設計内容を仕様にしてプロトタイプをつくり、システムを利用するメンバーがそれを実際に触って考え違いや仕様の漏れがないかを検証する。検証結果をベースにプロトタイプの修正を繰り返すことでシステムに仕上げていく。初期段階のプロトタイプの設計がいい加減であったり、修正検討が不十分だったりするといつまでたっても実用的なシステムはでき上がらない(**図4**)。

プロトタイプづくりをすべてベンダーに任せるという考え方もあるが、ベンダーSEの業務知識が不足していると検証作業に耐えられるシステムができずに時間ばかりが消費されるといったことにもなりかねない。ERPパッケージの普及が広まった2000年頃から、ベンダーのSEやITコンサル

タントの生産管理知識は著しく低下しているので注意が必要だ。

## 自社の業務特性に合ったシステムがつくれる

ローコード開発ツールを使ったシステム開発の最大のメリットは、自社の業務特性に合ったシステムを簡単につくれることにある。さらにプロトタイプを使った検証を通じて、企業特有の処理を新システムに取り込むか、取り込まないかを検討することもできる。

参考までに、筆者が実際に支援した企業で出てきた、パッケージでは対応しにくい生産管理処理を紹介する。

- 計画変更や納期変動が激しく、生産指示に対する変更処理を頻繁に行う必要がある
- 生産ロットを工程途中で分割して、一部を外注生産などに変更することがある
- 取引先単位でオプション品やオプション工程が発生する
- 重要部品に関して、どの製品に使われたのかを1個単位でトレースする必要がある
- 毎回製品の加工仕様が変わるので、製品マスターやBOMをつくることができない

## ローコード開発ツールの普及が遅れる日本

国内工場でのローコード開発ツールの普及は遅れている。ベンダーがローコード開発利用に消極的なことがある。今まで開発工数商売で儲けてきた日本のシステムベンダーはローコード開発ツールによって開発工数が減ってしまうと自社の売上や利益が減ってしまう恐れがある。そのため、今まで工数商売で儲けてきたベンダーほどローコード開発ツールの利用を避けようとする。

エンジニアに十分な業務知識がないと、検証作業に使えるプロトタイプを設計・開発することが難しいという面もある。業務知識がない場合は業務仕様が固まらないリスクを負わねばならなくなる。本来はパッケージでも同じなのだが、パッケージの場合は使い物にならなくても、無理やり納品してしまうベンダーがいる。使いこなせないのはユーザーのせいというわけだ。ローコード開発ツールの場合はそういうわけにはいかないので、リスク発生を嫌がるベンダー経営者は敬遠しがちだ。

大手ベンダーに頼らずに自社開発やコンサルタントとの共同開発を指向する中堅クラスの企業では、ローコード開発ツールに踏み込むケースが増えている。**図5**は筆者が関与した銘板メーカーの工程管理システムの管理画面だ。パッケージではここまで細かい製造仕様管理を行うことは難しい。

日本企業のITベンダー依存は世界的にも異例の状態にある。ローコード開発は企業のベンダー依存から脱皮する大きなチャンスでもある。

**図5　ローコード開発ツール（Sapiens）で開発した工程管理システムの例**


筆　者：ほんま　みねかず
代表取締役
所在地：〒181-0011
東京都三鷹市井口5-1-15
E-Mail：m.homma@mbf.nifty.com
ＵＲＬ：https://homma-consulting.jp