Migaro.
Technical
Report
セレクション４

第9回 最優秀賞受賞

日本調理機が取り組む システム改修方法の改善

IBM i資産の見える化と影響範囲の特定を容易にする アジャイル・ツールを自社開発

『Migaro.Technical Report No.9』（2016年秋）で最優秀賞を受賞した日本調理機の吉岡延泰氏の論文「IBM iの見える化で実現するアジャイル開発」は、システム改修の効率化・スピード化を実現する自社開発ツールのポイントを解説したもの。山積する改修案件に悩むIBM iユーザーにとって示唆に富む論文だ。今回は「特別編」として、吉岡氏に、ツール開発の背景とそのポイントをうかがった。

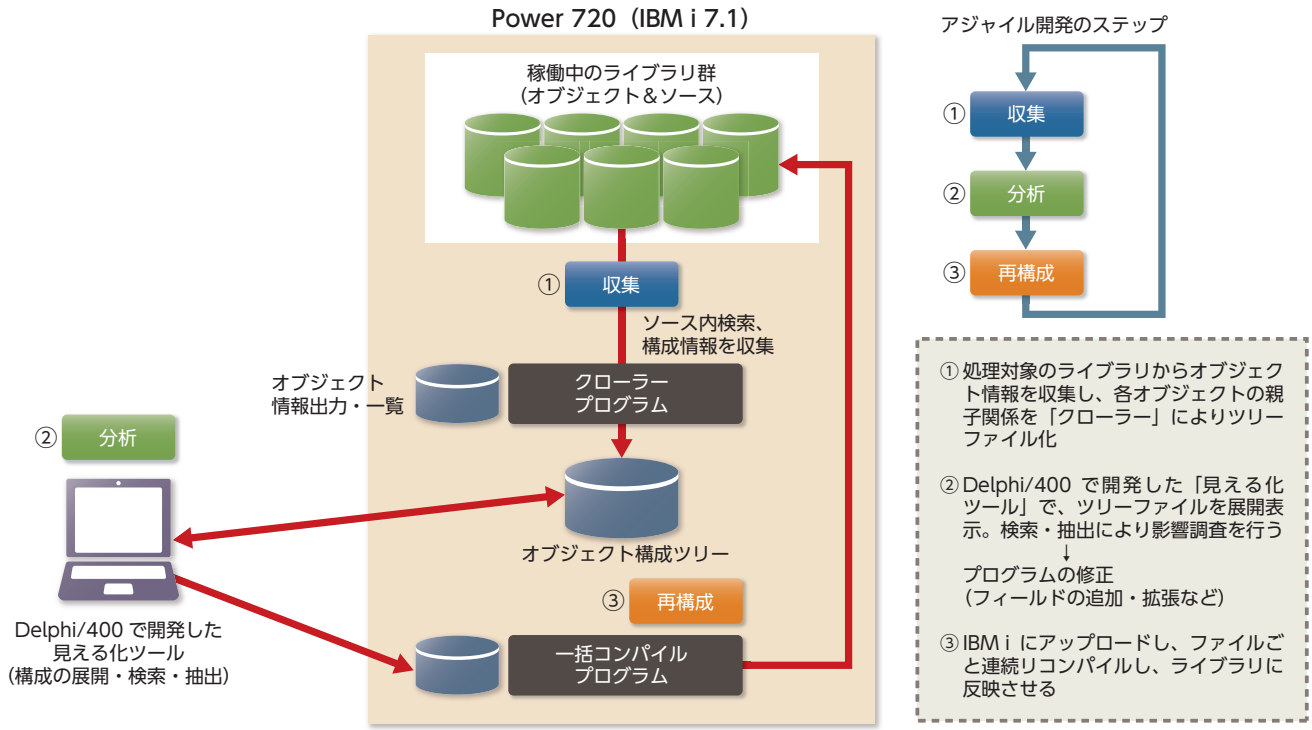
システムが肥大化・複雑化しスムーズな改修が困難に

戦後まもない1947年に設立された日本調理機は、半世紀以上にわたって、学校の給食施設や企業の社員食堂、病院・老後施設の食堂、ホテル・レストランなどに業務用厨房機器を提供してきた総合メーカーである。現在は、機器類だけでなく厨房施設の設計・施工なども手がけ、特注の大型設備・施設の開発に強みをもつことで知られる。

基幹システムは、1993年にAS/400を導入して以来、IBMミッドレンジ機上で構築・運用してきた。2012年に会計システムをPCサーバーへ移行したほかは、ほぼすべてのシステムがIBM i上での運用である。

基幹システム上のオブジェクト数は、現在利用されているものだけで約1万4000件。最近、プログラムの棚卸を行い不要なオブジェクトを廃棄したが、それでもこれだけの数がある。

図表１ アジャイル開発のステップと仕組み



システムの開発・運用・保守は「自社開発・自社運用」を方針としてきた。経営・業務ニーズに即した開発・改修をスピーディに柔軟に行えるのと、ユーザーの声を反映させた細かい作り込みができるのが、その理由である。業務ロジックの開発には、主としてRPG ⅢとRPG Ⅳを使用してきた。

しかしながら、長年の度重なる改修によってシステムが肥大化し、さらにプログラムの構造化を進めてきた結果、システムが複雑になり、改修が年々困難になるという事態が生じていた。これに加えて最近は、経営層から戦略的な改修の要請が増え、事態がますます逼迫する状況になっていた。

従来の影響調査は、対象となるプログラムをFNDSTRPDMコマンドを使って検索し、リスト化された一覧から関係のないプログラムを除外して、関係あるものについては1つずつソースを呼び出して影響範囲を確認するという方法を取っていた。そのためシステム全体の把握と影響範囲の特定に時間がかかり、改修の要請にタイムリーに応えられなくなっていた。

さらに、物理ファイルにフィールドを追加する場合、影響を受けるすべてのプログラムのリコンパイルに時間がかかるので、同じキーフィールドをもつ物理ファイルをもう1つ別に作成するようなことも起きていた。「結果的にデータベースの正規化を正しく行えず、システムがより複雑になるという悪循環を招いていました」と、情報システム部主任の吉岡延泰氏は振り返る。

IBM i資産を見える化し一括コンパイルまでカバーするツール

そこで開発したのが、プログラム資産を簡単に見える化し、その改修範囲の特定を容易にするシステムである。「その発想自体はかなり前からもっていましたが、RPGとCLと5250画面だけで見える化のためのシステムを開発するのは制約が大きすぎると考え、断念した経緯があります。2011年にDelphi/400を導入し、グラフィカルなインターフェースを簡単に作れるようになったので取り組んだのが、IBM i資産の見える化とともに、アジャイル開発も可能な今回のシステムです」（吉岡氏）

吉岡 延泰氏
日本調理機株式会社
情報システム部
主任

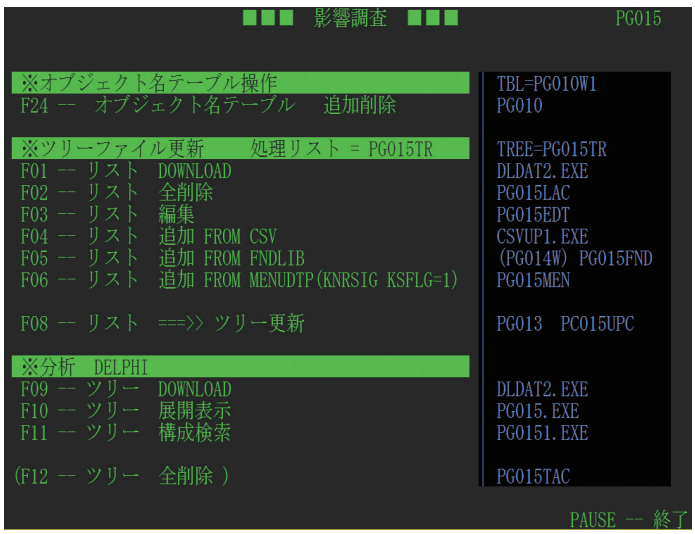
COMPANY PROFILE

本　　社：東京都大田区
設　　立：1947（昭和22）年
資 本 金：5億9760万円
売 上 高：165億3766万円
従業員数：509名
事業内容：厨房用機器、食品加工用機器などの製作・販売・輸出入、厨房施設の設計・施工・監理など
http://www.nitcho.co.jp/

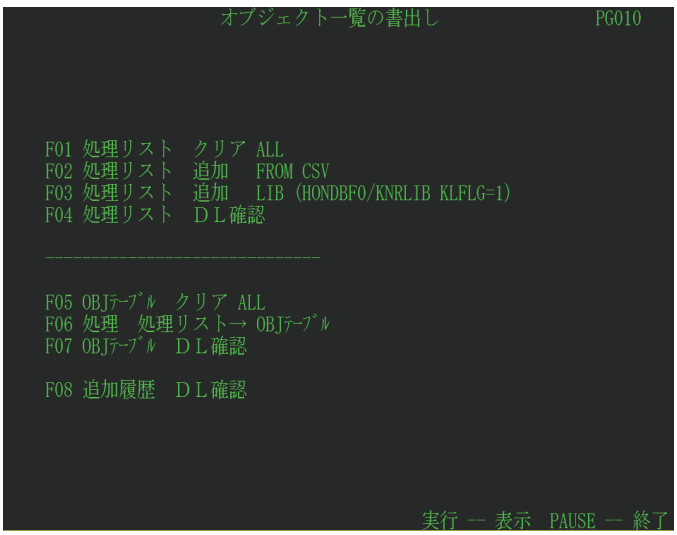
システムは、オブジェクトの収集から影響調査、修正後の一括コンパイルまでを行う一連のツール類と、プログラムの検索と結果のダウンロード、従属論理ファイルの調査、ソースとオブジェクトの整合性チェックなどの機能をもつ補助ツール類で構成されている。

改修作業はこれらのツールを使い、「収集」→「分析」→「再構成」の3つのステップで進められる（図表1）。

図表2　トップ画面（総合メニュー）。ここから作業を開始する



図表3　オブジェクト一覧の書き出しを行う画面



最初に、処理対象のライブラリからすべてのオブジェクトをテーブルファイルへ抽出する（オブジェクト情報の収集）。次に、収集した各オブジェクトの親子関係を、「クローラー」と呼ぶRPGとCLで開発したツールを使ってツリー構造のファイルに書き出す（以上、「収集」（図表2・図表3））。そして、そのツリーファイルをDelphi/400で開発した「見える化ツール」で表示・展開して、抽出を行うのが「分析」である（図表4）。

そして最後の「再構成」は、分析結果から得られたデータをCSV形式に変換してIBM iにアップロードし、「一括コンパイルプログラム」を使って連続リコンパイルする工程である（図表5）。その結果を稼働中のライブラリにリリースして作業は終了する（物理ファイルの設計の見直しは、従来どおり手作業で行う必要がある）。

「分析」ステップの見える化ツールを使うと、オブジェクトの親子関係を「親→子」や「子→親」などの表示形式で展開でき、全ノードの展開・格納もボタン1つで行える。精査したいプログラムは、クローラーの処理リストやメニューリストから選択したり、ツリー表示中の項目の選択で表示できる。

Delphi/400のコンポーネントを使い、短期にツール開発

見える化ツールの開発では、Delphi/400が備えるQueryコンポーネントとTreeViewコンポーネントを主に使用した。コンポーネントはソフトウェアの部品で、Delphi/400の開発画面上にコンポーネントを貼り付けて属性を設定するだけでGUI画面を作成できる。

Queryコンポーネントはデータベースや複数のテーブルにSQL文でアクセスするときのコンポーネント、TreeViewコンポーネントは、階層化されたデータを視覚的に表示するためのコンポーネントである。

同社では、この見える化ツールの開発以前に、これらのコンポーネントを使って、製品と組立品および部品との関係を表示する「部品構成図」プログラムを開発した経緯がある。吉岡氏は、「それと同様の仕組みを構造化された情報の展開に適用したのが、見える化ツールです」と語る。

プログラムの影響調査では、ファイルを使用している親プログラムや、子プログラムと連携する親プログラムを調べることが多い。そのようなとき、見える化ツールの検索機能によって関連ファイルやプログラムをすぐにリスト化しダウンロードもできる。プログラムの修正は、この影響調査の結果をもとにスピーディに行うことが可能だ。

システムの開発は2014年にスタートし、約3カ月半で終了した。「システム部員が利用するだけなので見た目やわかりやすさにはこだわらず、短期開発だけを考えて作業しました」（吉岡氏）。実際、プログラムの実装に要したのは1カ月たらずで、残りの2カ月半は事前調査に費やしたという。

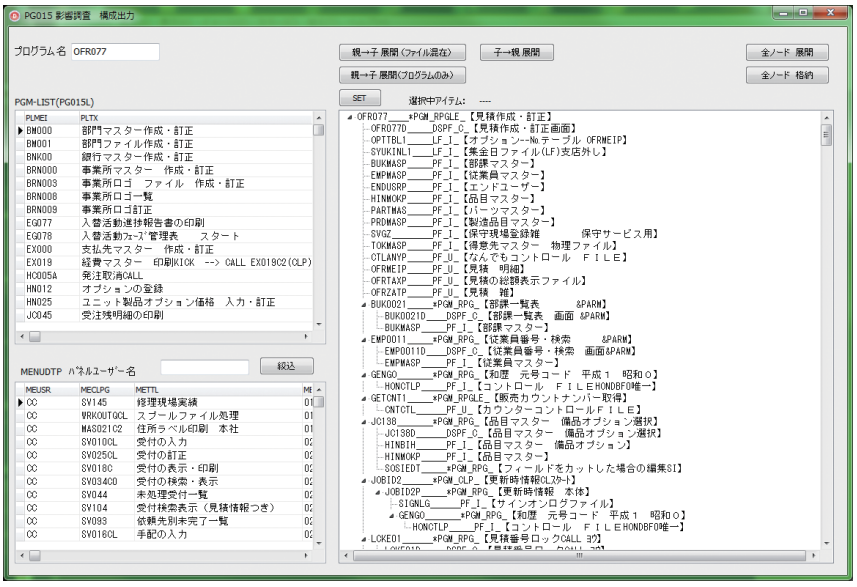
事前調査に時間を要した理由について吉岡氏は、「ツールを使ってソースの構造を解析し良好な結果を得るには、システムがルールに則って構成されていることが必須です。当社では従来から、“論理ファイルは物理ファイルと同じライブラリに作成する”“ソースとオブジェクトは同じライブラリに配置する”などの運用ルールを設けてきましたが、長年の度重なる改修で例外も数多く発生していました。その例外の修正や、影響調査の範囲を絞るための現行データと過去データとの区分けに、予想外に時間がかかりました」と説明する。

改修をアジャイル方式で行えるようにする

今回のシステムによって、同社では改修のやり方が大きく変わった。

大規模な改修の場合、それまでは計画と影響調査に長時間をかけ、設計、開発、テストへと移るウォーター

図表4　Delphi/400で開発した見える化ツールを使い、RPGソースをツール展開表示



フォール型で行っていたが、導入したシステムでは影響調査が短時間で済み、修正、リコンパイル、リリースもスピーディに進められるので、小さな単位の改修が可能になった。つまり、「修正→リリース」を小刻みに繰り返すアジャイルな改修が可能になったのである(図表6)。

その変化の1つとして、たとえば仕様書がある。従来、大規模な改修では設計前に時間をかけて仕様書を作成してきたが、現在は、改修のポイントを記したメモとラフな画面イメージをメンバー間で共有するだけで作業を進める。「仕様書の作成・管理は廃止してしまいました」と、吉岡氏は言う。

また、修正したプログラムはすみやかにリリースするので、最新のソースの影響調査でほかのメンバーの進捗状況が確認できるようになった。

吉岡氏は、「当社のような開発要員が少人数で、それぞれのスキルマップがわかっているような体制では、アジャイル型の開発・改修は非常に合っていると感じています。小さな単位で開発・改修作業を割り当てられ、修正を終えたところからリリースできます」と言い、今回のシステムの意義を次のように総括する。

「企業は、新しいイノベーションを創出し、その改善の繰り返しで成長していきますが、それを支えるシステムは、開発・改修をスピーディに行える要件を備えていることが重要です。今回のシステムは、開発・改修の効率化を目指したものですが、それと同時に、変化に強いシステム基盤への第一歩だと考えています」

見える化ツールについては今後、各ツールの機能の充実化(重複排除など)や、スケジュール機能による収集・メンテナンスの自動化、Delphiのソースファイルへの対応などをを予定しているという。

図表5 新規開発した一括コンパイルツールを使い、ファイルごとに連続コンパイルを実行

```
                    連続コンパイル                          PG020

------------------------------------------------------------------------

【 STEP1 】   リストエントリー     対象 DSPF.CL.RPG.RPGLE
              PG020WL … 1:ﾗｲﾌﾞﾗﾘ   2:ｿｰｽﾒﾝﾊﾞｰ( プログラム名 )

 F1 -- リストを空にする。
 F2 -- CSVﾌｧｲﾙ からリストへ追加
 F3 -- リストをＤＬして確認
------------------------------------------------------------------------
【 STEP2 】   リストのチェック   (ｿｰｽﾒﾝﾊﾞｰﾀｲﾌﾟ.ｻｰﾋﾞｽﾌﾟﾛｸﾞﾗﾑ)
              (PG015 で影響調査済みであること )

 F4 -- チェック＆リストＤＬ確認 --> 除外する場合は STEP1 へ
------------------------------------------------------------------------
【 STEP3 】   連続コンパイル      ﾗｲﾌﾞﾗﾘﾘｽﾄ = 全ての現行ライブラリ

 F5 --  連続コンパイル実行
------------------------------------------------------------------------
【 STEP4 】   コンパイル結果リストチェック

 F6 --  結果リストのＤＬ
------------------------------------------------------------------------
                                                    PAUSE -- 終了
```

図表6 見える化ツールを使ったアジャイル手法による改修の効果