【セッションNo．1】

IBM i の見える化で実現するアジャイル開発

IBM i ユーザーにありがちな困り事を RPGとDelphiで解決する

日本調理機株式会社
情報システム部
吉岡　延泰　様

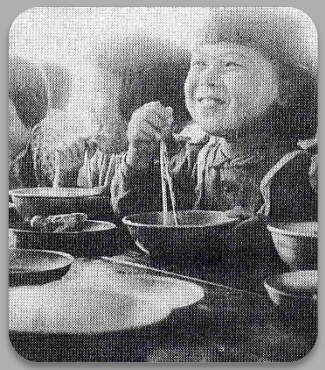
【アジェンダ】

1）　会社概要
2）　既存システムのあらまし

◆ アジャイル開発向けシステム
3）　開発の経緯
4）　解決したい既知の問題点
5）　システム構築の為の事前調査・整備
6）　構成と要素
7）　効果
8）　今後の展望
9）　まとめ

1）会社概要

会社概要

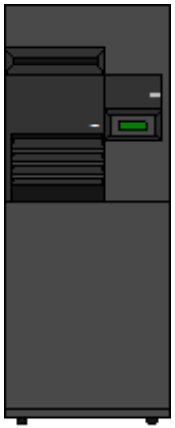


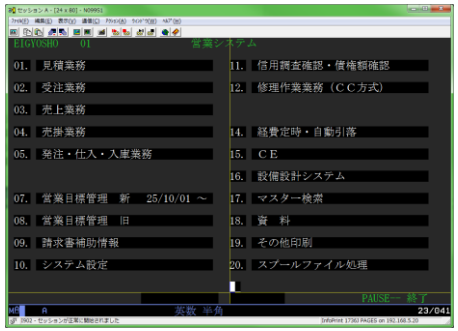

【事業所】	【設立】	【従業員】	【納入先】	【売上高】
40拠点 本社 東京都	1947年 昭和22年 設立69年	540名	学校 社員食堂 病院 ホテル	167億円

■ 会社概要

2）既存基幹システムのあらまし

■ 既存の基幹システムのあらまし

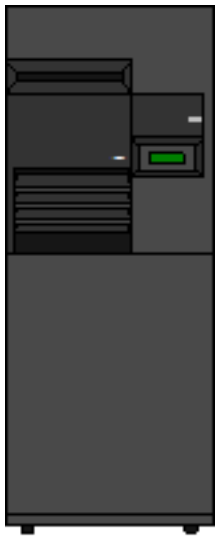




AS/400は1993年から現在まで利用。
2011年にDelphi/400 XEを導入。

データエントリーなどの入力系は5250画面を継続して使用。
表示・印刷などの出力系のプログラムでDelphiが増えてきた。

■ 既存の基幹システムのあらまし



オブジェクト総数

20,000 件超

アジャイル開発向けシステム

3）開発の経緯

■ 開発の経緯

◆ 事業範囲が広く、基幹システムのプログラムの総数が多い。

- 総合機器メーカーなので、業務の種類が多い。
- 【販売系】B to GもB to Bもある、備品から設備まで、幅広く対応。
- 【生産系】機器に特注品が多く、計画生産と受注生産の両方がありえる。
- 部門間をまたいだシームレスな管理システムが存在する。

→　**そこが企業としての強みでもある。**

◆ システムを内製している理由。

- 柔軟かつスピーディーに、業務に合わせた開発ができる。
- 部門間の連携を強化したシステム作りが可能。
- ユーザーの声をシステムに反映させやすく、細かい作り込みもできる。

→　**内製はスピーディーで業務改革にも強い！　はずだったが…**

■ 開発の経緯

長年の度重なるシステム改修による、システムの肥大化。

プログラムの構造化を進めたことによる、システムの複雑化。

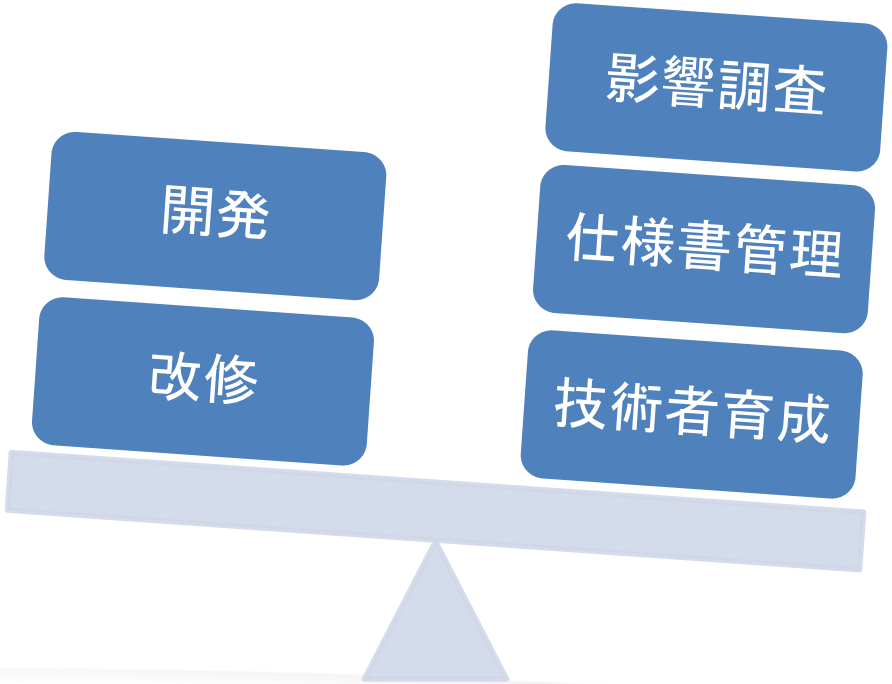
→ 影響調査に時間がかかり、開発時間が確保できない。
→ プログラムの品質の低下。
→ 棚卸しが出来ていない為、無駄な開発箇所が増えていく。
→ 大規模な改修が困難になる。
→ システムの構造を仕様書で管理する時間が確保できない。
→ システムを把握している技術者の育成には膨大な年数がかかる。

年々システム改修にかかる工数が増大していく悪循環に陥った。

■ 開発の経緯

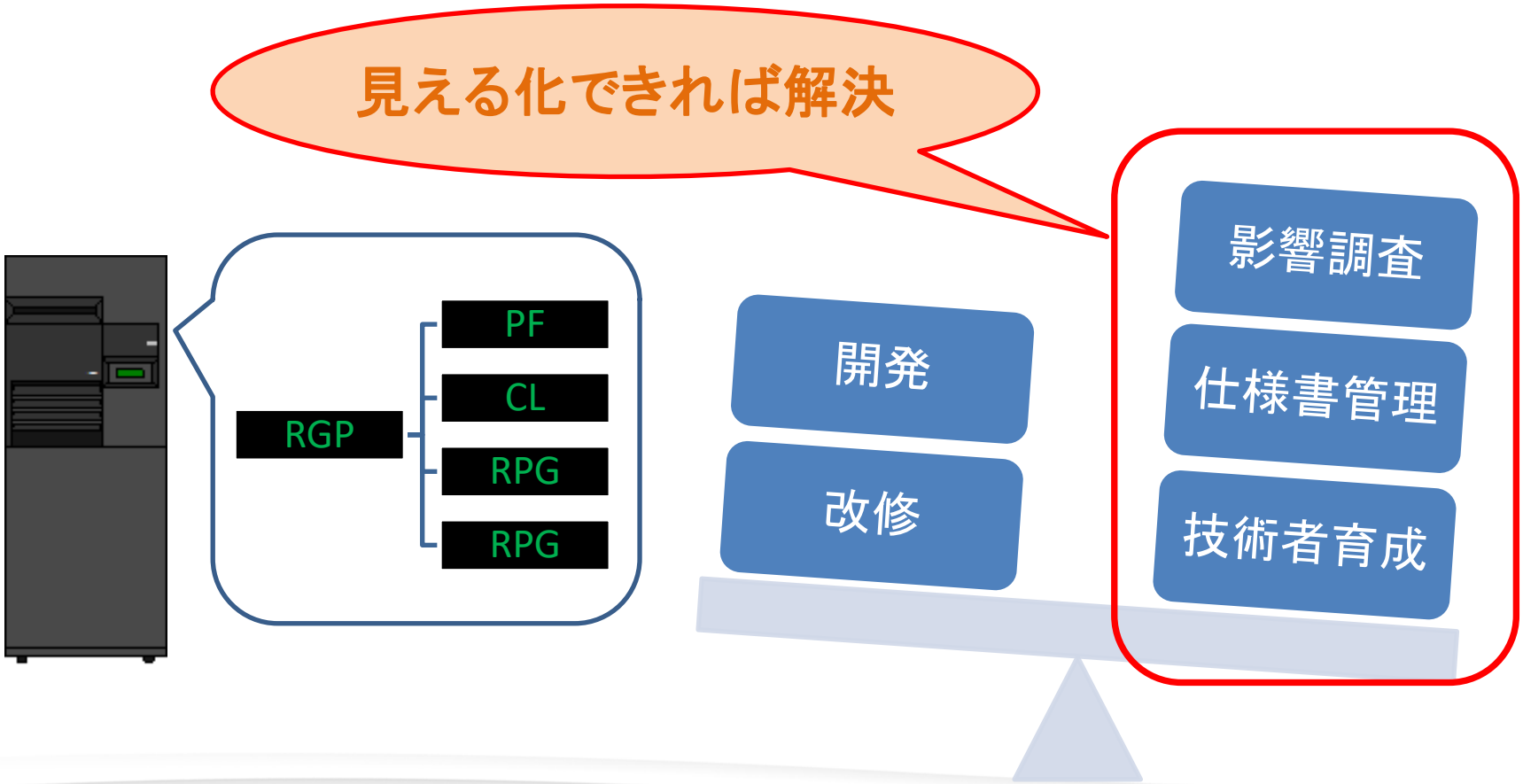
プログラムの構造化は必要不可欠。
（プログラムのカプセル化による品質の向上。開発の効率化。）

ただし、同時にシステム全体の把握が難しくなるジレンマが付き纏う。



■ 開発の経緯

RPG,CLとDelphi/400で、システムを見える化するツール類を作成し、
アジャイル開発手法にシフトする方針を打ち出した。



アジャイル開発向けシステム

4）解決したい既知の問題点

■ 解決したい既知の問題点

◆前項までの内容

- 影響調査の時間を短縮したい。
- 無駄な仕様書の作成・管理をやめたい。
- システムを知らない技術者でも構造が把握できるようにしたい。
- ソース・オブジェクトの棚卸しをしたい。

◆その他、既知の問題点

- システムに問題が発生した際にも調査に時間がかかっていた。

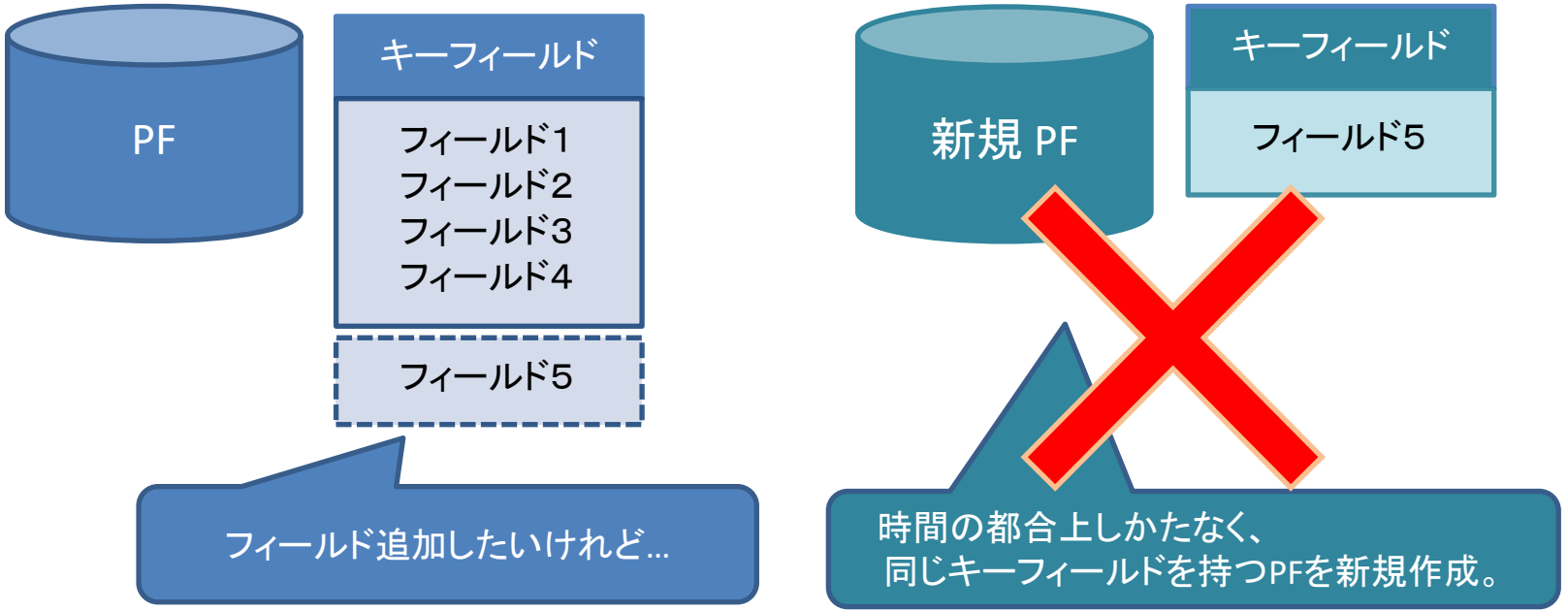
- ソースの検索(FNDSTRPDM)に余分な検索結果が多く、
  検索結果を元にした影響調査ではヒューマンエラーが起こりやすかった。

- **物理ファイルに新規フィールドを追加したいが時間がかかるため、
  新たに物理ファイルを作成するケースが多かった。**

■ 解決したい既知の問題点

新たに物理ファイルを作成してしまうケース

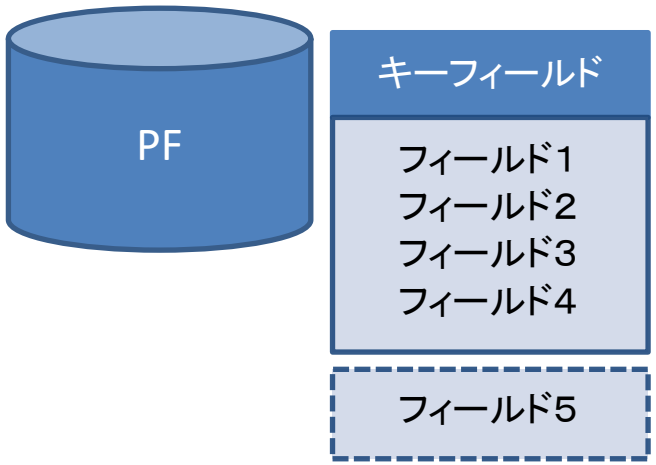
追加するには、関連プログラムを全てリコンパイルする必要がある。



本来１つであるべきものが複数に分かれる。（プログラムも同様）

システムがさらに複雑化してしまう要因。

■ 解決したい既知の問題点



システムを見える化する、
それだけでは解決しない。

- **時間がかかりすぎる。**
- **案件は期末前後に集中。**
  計画が立てられない。

- **時間がかからずに出来る仕組みがあれば良い。**
  （見える化で得たデータを使って、関連プログラムを一括コンパイル）

- **大規模な計画が必要なウォーターフォール型の開発をやめて、**
  アジャイル型の開発にシフトする。
  （先にフィールドを追加してから、プログラムを個別に順次リリース）

アジャイル開発向けシステム

5）システム構築の為の事前調査・整備

■ システム構築の為の事前調査・整備

◆事前調査でのポイント

① 自社の運用ルールが守られているかどうか。

想定外のプログラムや構造が無いかを洗い出す。

② 現行データと過去データをしっかり区分けする。

影響調査の範囲を確定させる。

③ 使用されている言語や呼び出し方のバリエーション。

CL、RPG３、ILE-RPG（一部 FF、バインド有り。）

■ システム構築の為の事前調査・整備

① 自社の運用ルールが守られているかどうか。

・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。

・別のライブラリに同名のファイルを作成する場合、
フィールドの構成などを全く同じ状態に保つ。

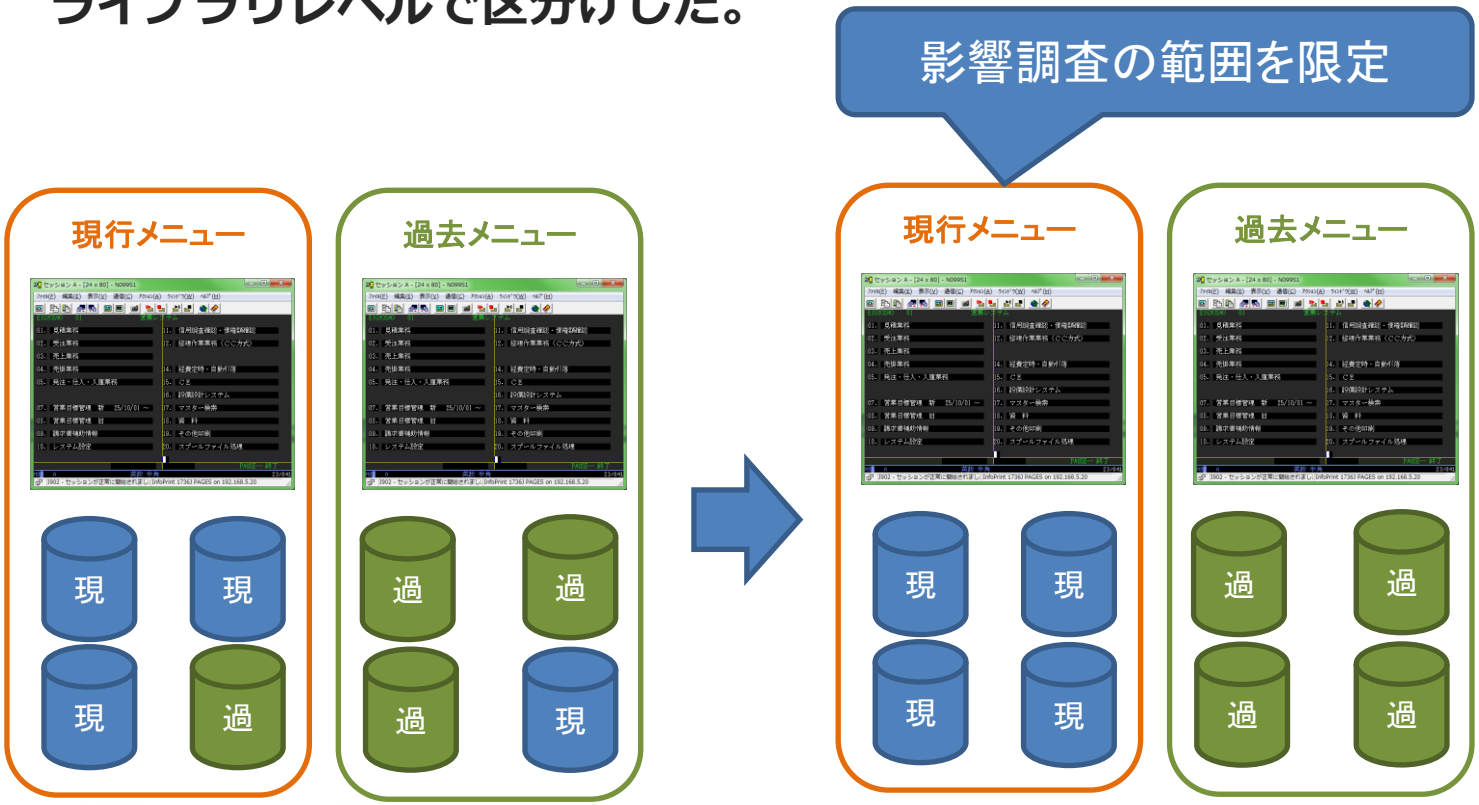
・ソースとオブジェクトは同じライブラリに配置。
（ソースとオブジェクトが対になっていること。）

・プログラムの命名は全ライブラリを含めて固有の名称にする。

■ システム構築の為の事前調査・整備

② 現行データと過去データをしっかり区分けする。

→ 現行メニューでは、現行データを扱うライブラリのみ参照。
ライブラリレベルで区分けした。



アジャイル開発向けシステム

6）構成と要素

■ 構成と要素

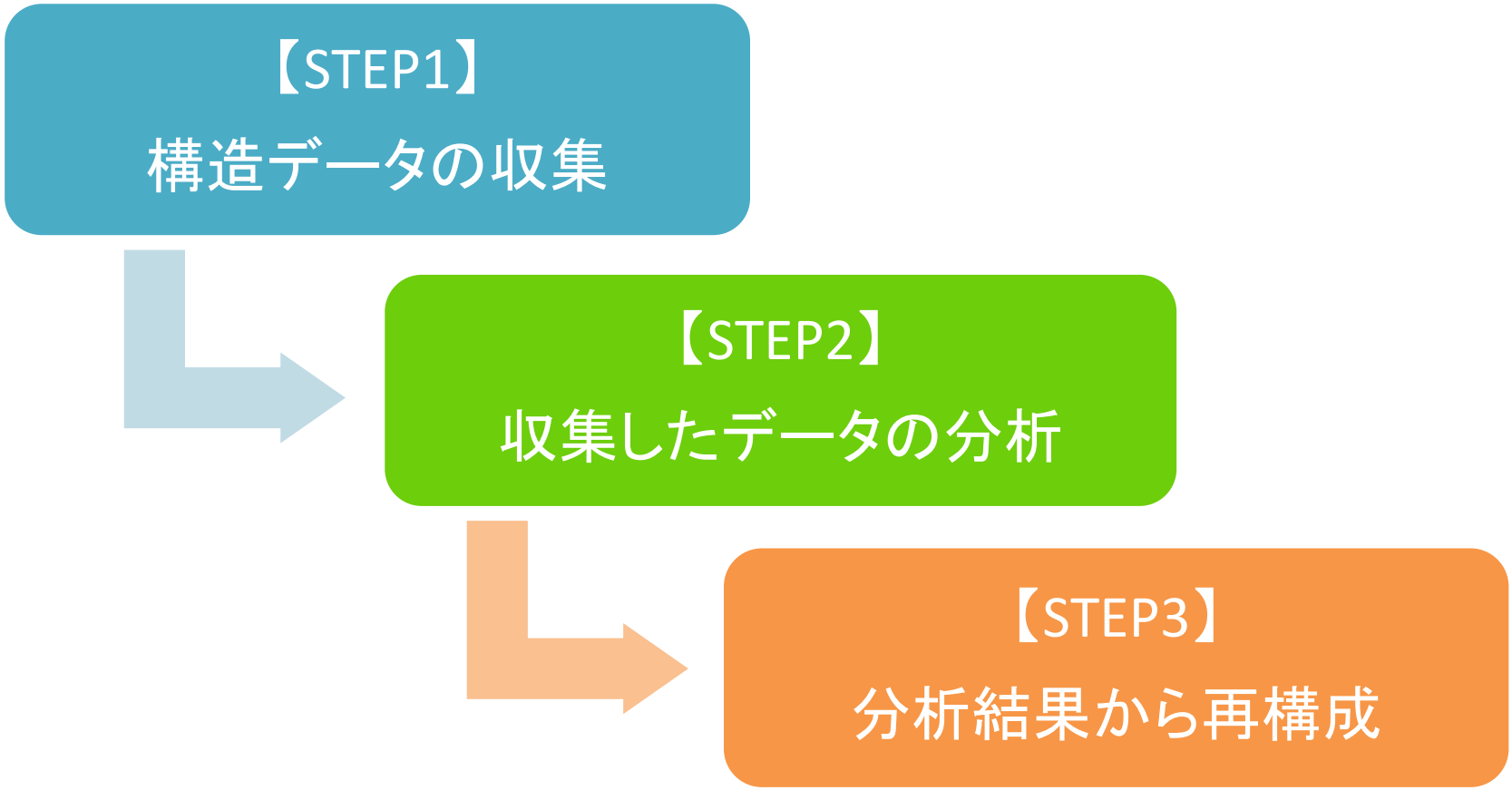
開発コンセプト

使用するのは RPG,CL と Delphi/400 のみ。

とにかく時間をかけずに使えるものを作る。

- 基本的な部分はRPGとCLで作成する。
- GUIが必要な部分はDelphi/400を使う。
- 各ツール間のデータのやり取りには、
  xlsやCSVファイルを使う。

■ 構成と要素



■ 構成と要素

各ツールの処理ファイルへ　xls・CSVファイルからの入出力。

メインツール

【STEP1】収集　【1】オブジェクト情報出力　【2】クローラー

【STEP2】分析　【3】構造の把握　【4】構造の検索・リスト化

【STEP3】再構成　【5】一括コンパイル

サブツール

(1) メニューの検索・展開

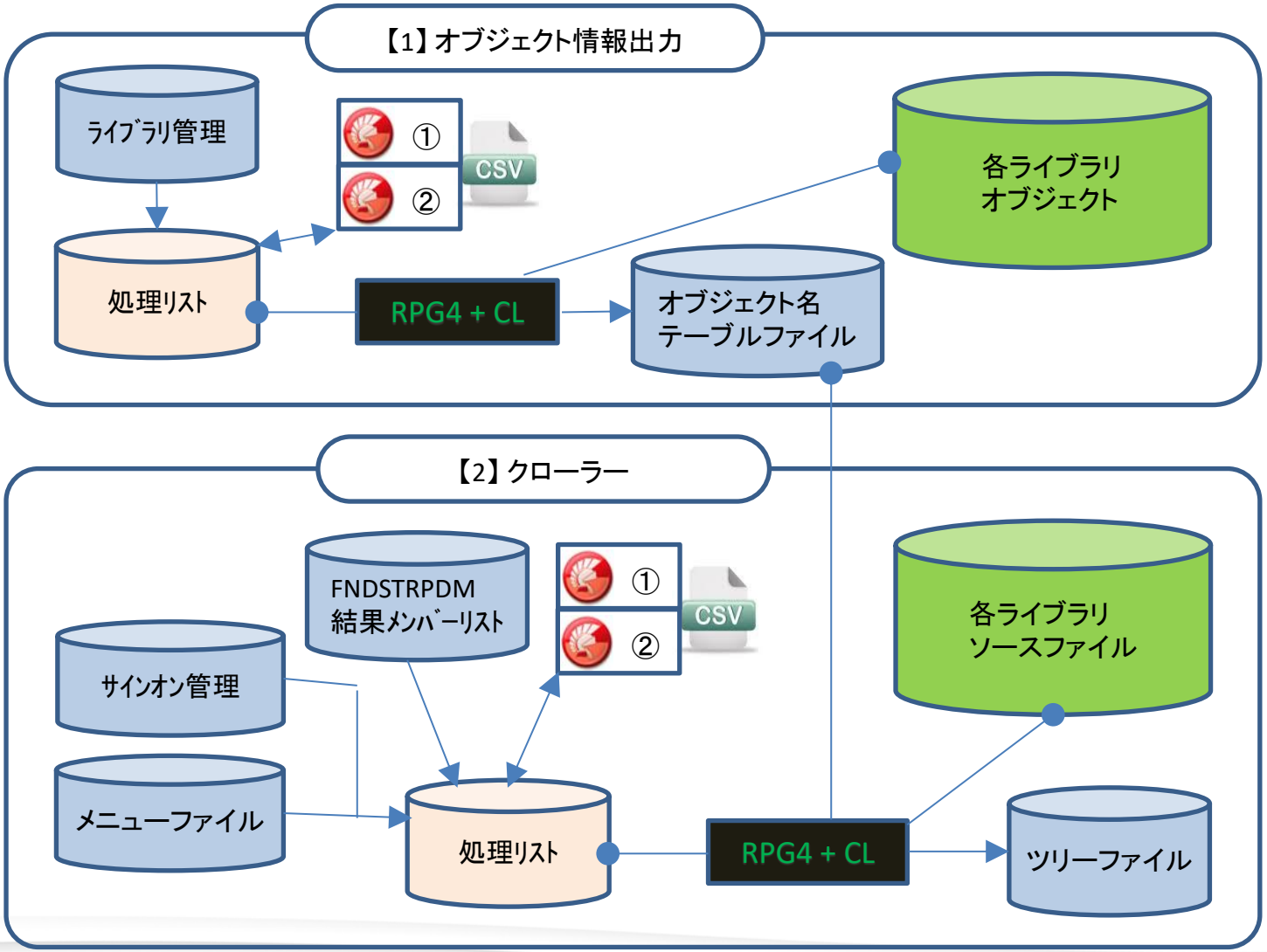
(2) FNDSTRPDM結果DL

(3) 従属論理ファイル調査

(4) 棚卸し用 調査・処理

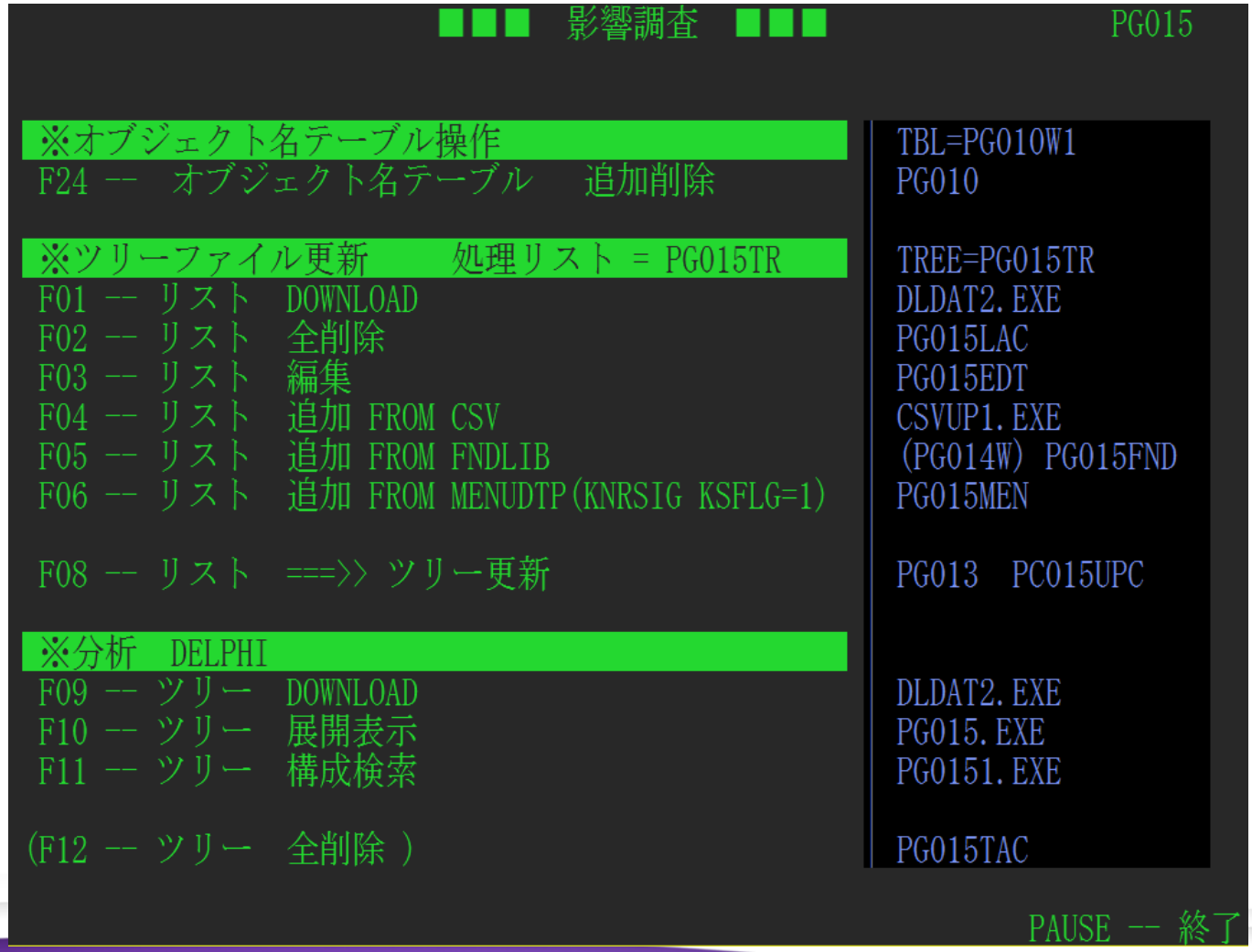
■ 構成と要素

【STEP1】収集



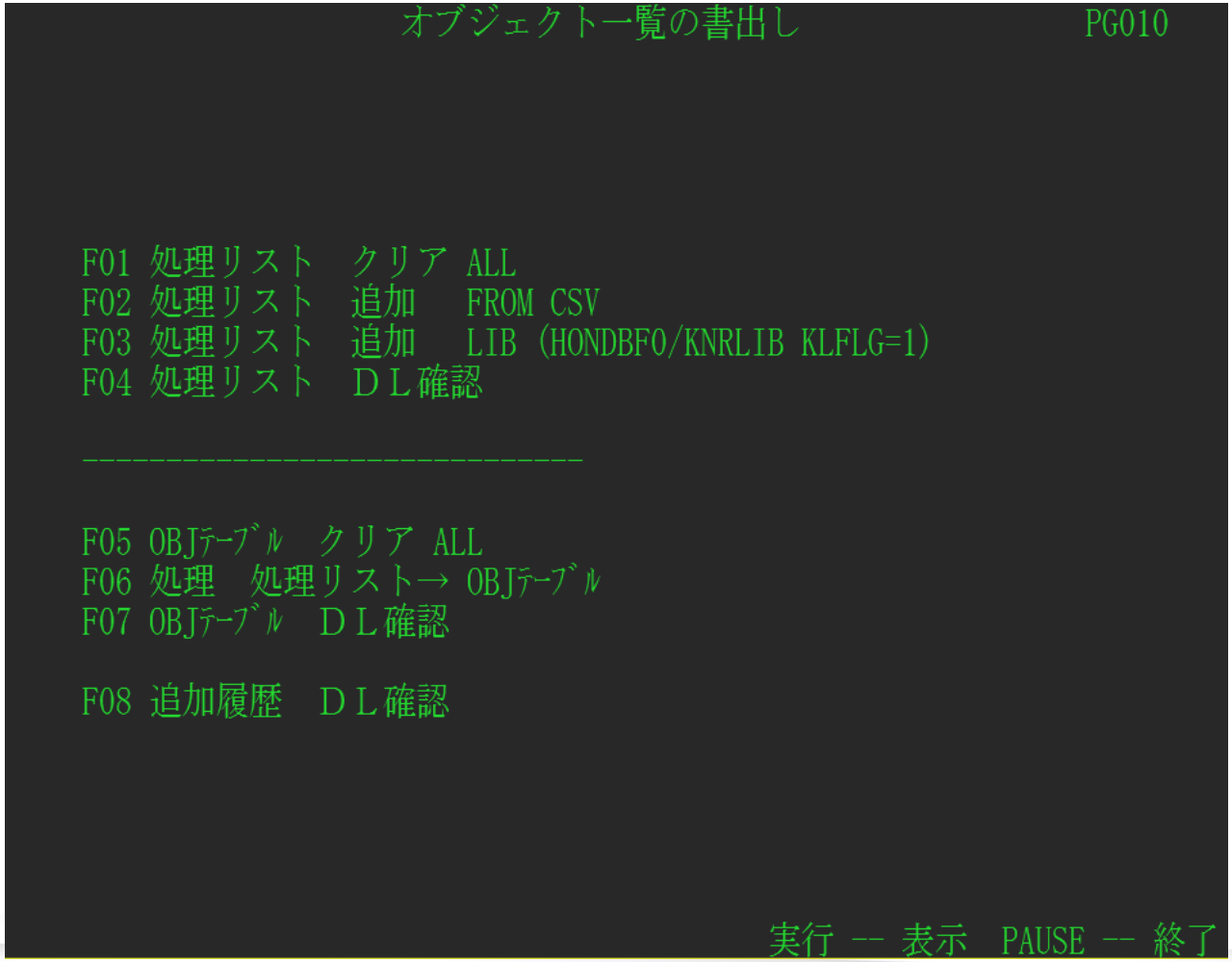
■ 画面例

- 影響調査（総合メニュー）

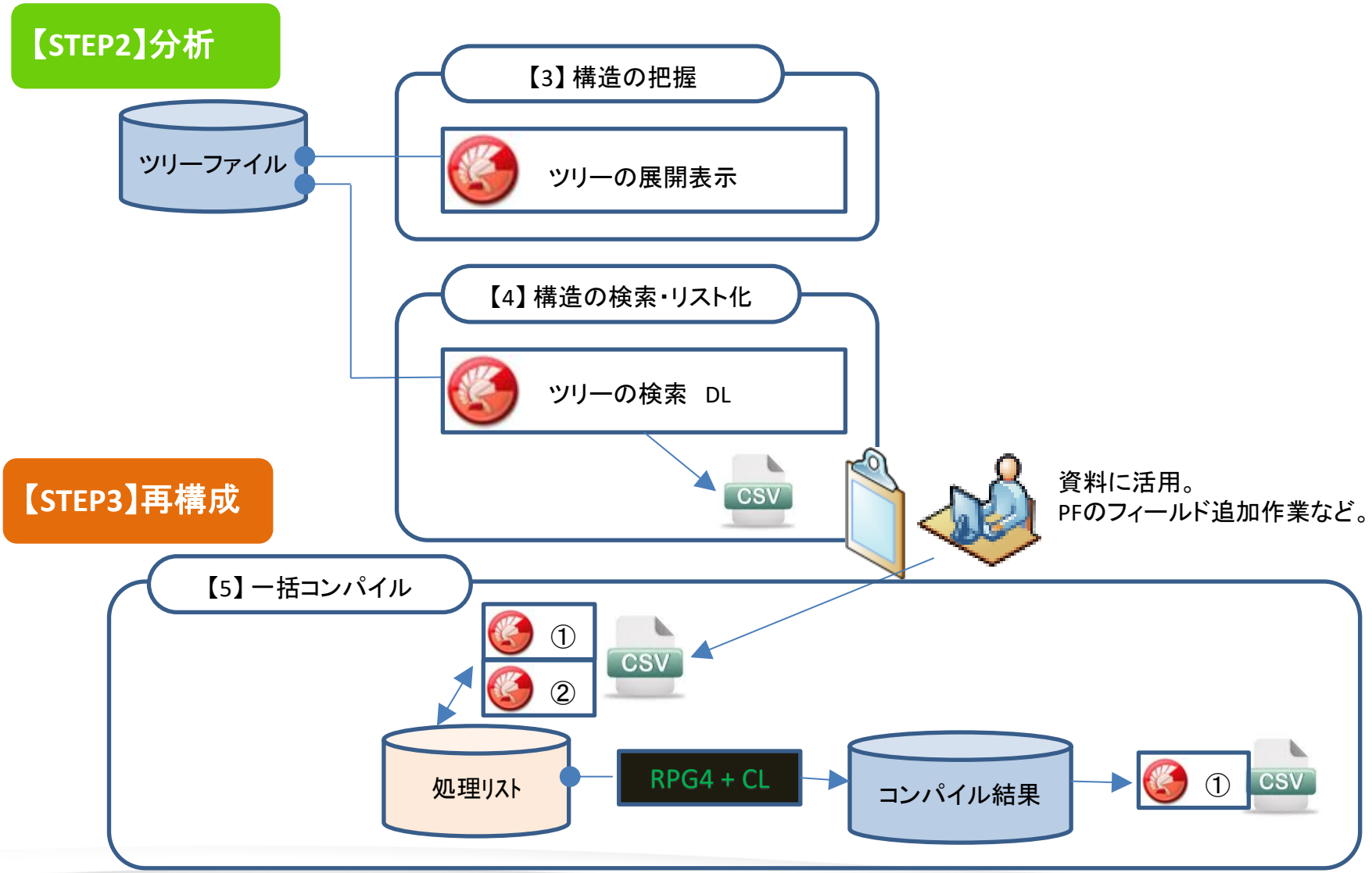


■ 画面例

- オブジェクト一覧の書出し

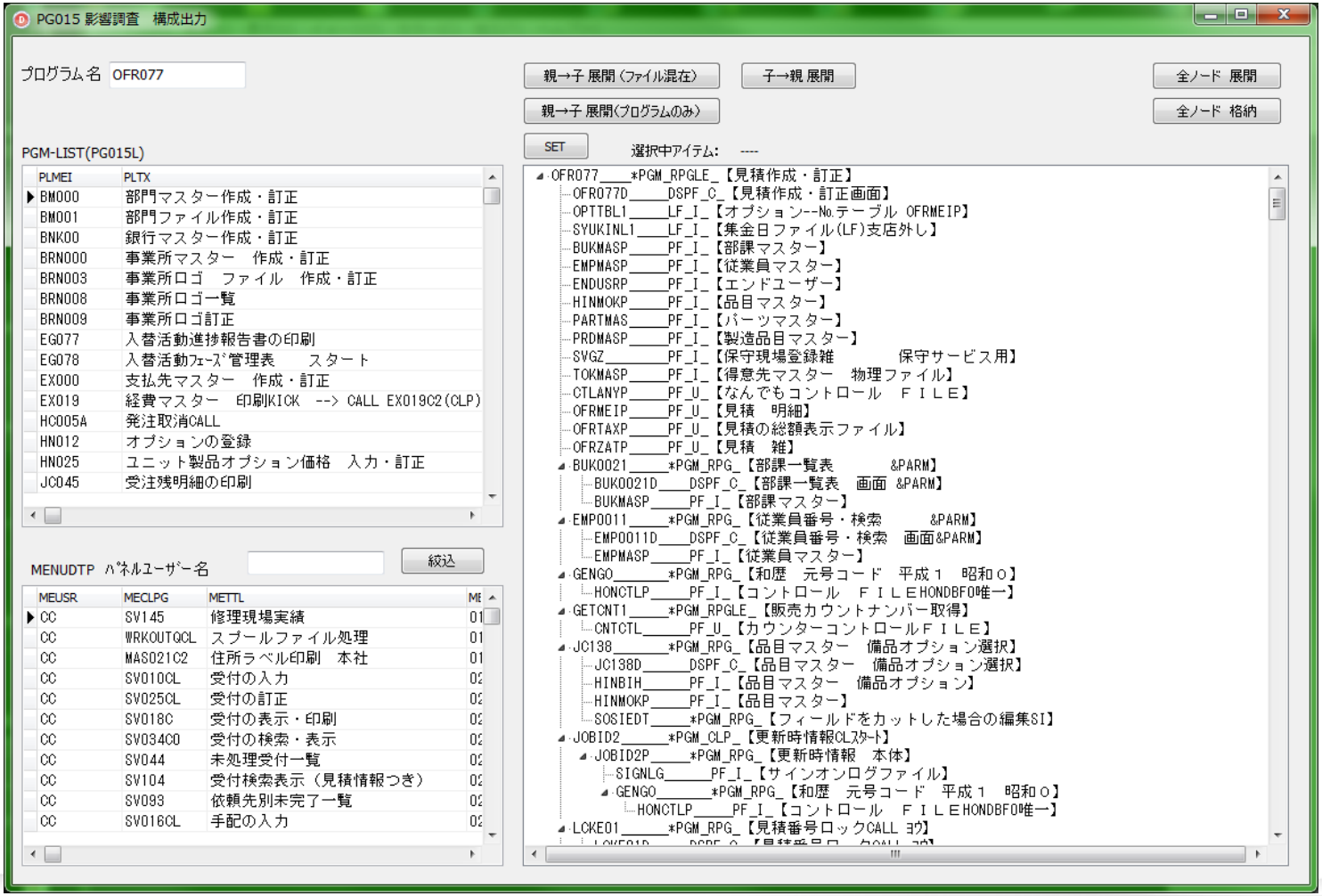


■ 構成と要素


【STEP2】分析
【3】構造の把握
ツリーファイル
ツリーの展開表示
【4】構造の検索・リスト化
ツリーの検索　DL
CSV
【STEP3】再構成
資料に活用。
PFのフィールド追加作業など。
【5】一括コンパイル
①
②
CSV
処理リスト
RPG4 + CL
コンパイル結果
①
CSV

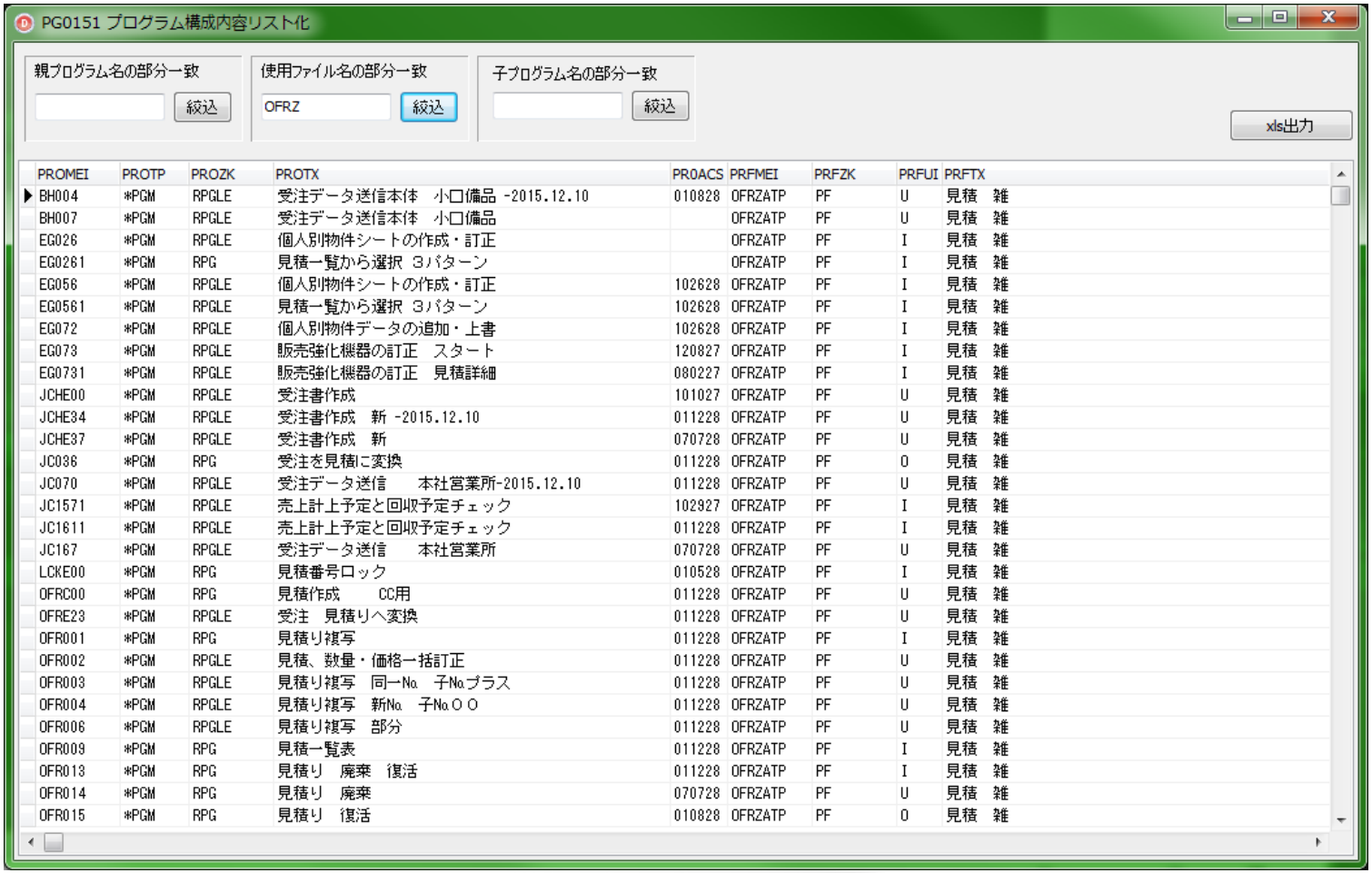
■ 画面例

- ツリーの展開表示



■ 画面例

- プログラム構成内容リスト化



PROMEI	PROTP	PROZK	PROTX	PROACS	PRFMEI	PRFZK	PRFUI	PRFTX
BH004	*PGM	RPGLE	受注データ送信本体　小口備品 -2015.12.10	010828	OFRZATP	PF	U	見積　雑
BH007	*PGM	RPGLE	受注データ送信本体　小口備品		OFRZATP	PF	U	見積　雑
EG026	*PGM	RPGLE	個人別物件シートの作成・訂正		OFRZATP	PF	I	見積　雑
EG0261	*PGM	RPG	見積一覧から選択 3パターン		OFRZATP	PF	I	見積　雑
EG056	*PGM	RPGLE	個人別物件シートの作成・訂正	102628	OFRZATP	PF	I	見積　雑
EG0561	*PGM	RPGLE	見積一覧から選択 3パターン	102628	OFRZATP	PF	I	見積　雑
EG072	*PGM	RPGLE	個人別物件データの追加・上書	102628	OFRZATP	PF	I	見積　雑
EG073	*PGM	RPGLE	販売強化機器の訂正　スタート	120827	OFRZATP	PF	I	見積　雑
EG0731	*PGM	RPGLE	販売強化機器の訂正　見積詳細	080227	OFRZATP	PF	I	見積　雑
JCHE00	*PGM	RPGLE	受注書作成	101027	OFRZATP	PF	U	見積　雑
JCHE34	*PGM	RPGLE	受注書作成　新 -2015.12.10	011228	OFRZATP	PF	U	見積　雑
JCHE37	*PGM	RPGLE	受注書作成　新	070728	OFRZATP	PF	U	見積　雑
JC036	*PGM	RPG	受注を見積に変換	011228	OFRZATP	PF	O	見積　雑
JC070	*PGM	RPGLE	受注データ送信　本社営業所-2015.12.10	011228	OFRZATP	PF	U	見積　雑
JC1571	*PGM	RPGLE	売上計上予定と回収予定チェック	102927	OFRZATP	PF	I	見積　雑
JC1611	*PGM	RPGLE	売上計上予定と回収予定チェック	011228	OFRZATP	PF	I	見積　雑
JC167	*PGM	RPGLE	受注データ送信　本社営業所	070728	OFRZATP	PF	U	見積　雑
LCKE00	*PGM	RPG	見積番号ロック	010528	OFRZATP	PF	I	見積　雑
OFRC00	*PGM	RPG	見積作成　CC用	011228	OFRZATP	PF	U	見積　雑
OFRE23	*PGM	RPGLE	受注　見積りへ変換	011228	OFRZATP	PF	U	見積　雑
OFR001	*PGM	RPG	見積り複写	011228	OFRZATP	PF	I	見積　雑
OFR002	*PGM	RPGLE	見積、数量・価格一括訂正	011228	OFRZATP	PF	U	見積　雑
OFR003	*PGM	RPGLE	見積り複写　同一№　子№プラス	011228	OFRZATP	PF	U	見積　雑
OFR004	*PGM	RPGLE	見積り複写　新№　子№００	011228	OFRZATP	PF	U	見積　雑
OFR006	*PGM	RPGLE	見積り複写　部分	011228	OFRZATP	PF	U	見積　雑
OFR009	*PGM	RPG	見積一覧表	011228	OFRZATP	PF	I	見積　雑
OFR013	*PGM	RPG	見積り　廃棄　復活	011228	OFRZATP	PF	I	見積　雑
OFR014	*PGM	RPG	見積り　廃棄	070728	OFRZATP	PF	U	見積　雑
OFR015	*PGM	RPG	見積り　復活	010828	OFRZATP	PF	O	見積　雑

■ 画面例

- 連続コンパイル実行

```
                        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
```

STEP1．収集　デモ

【STEP1】収集

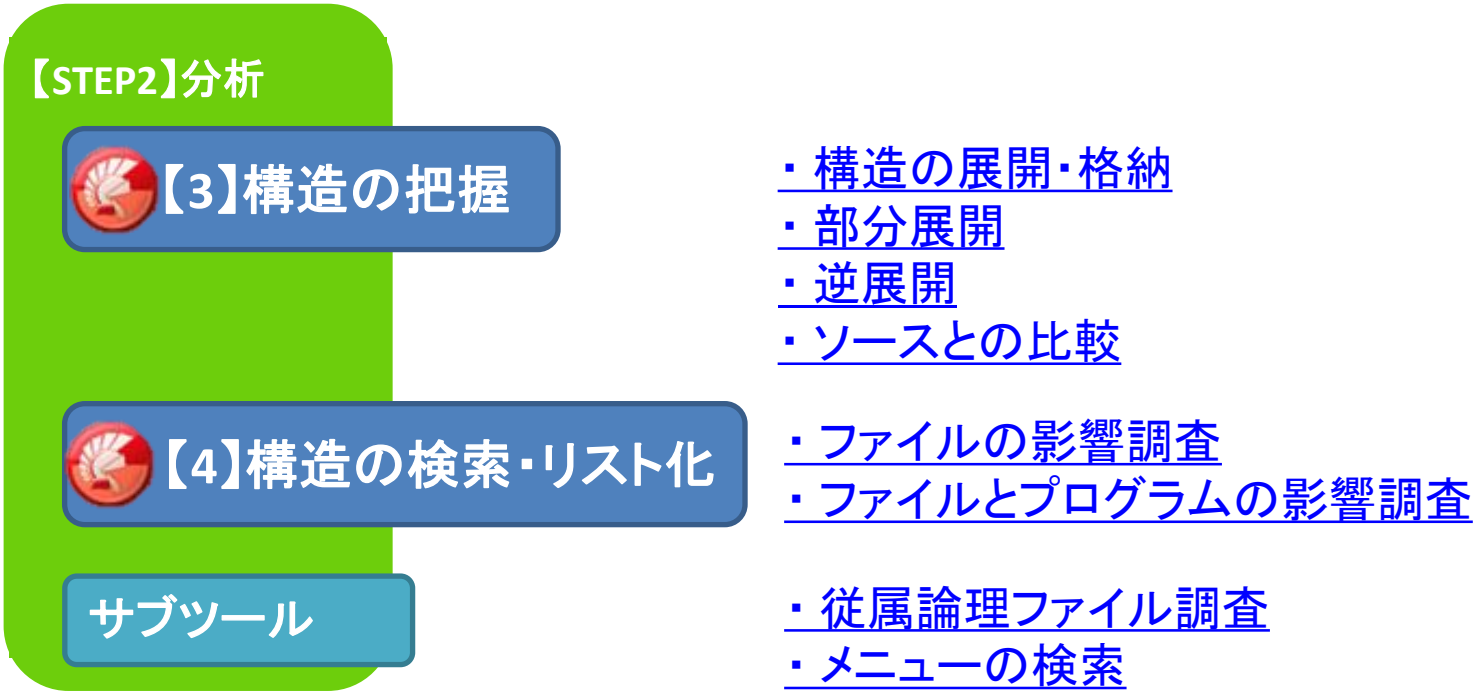
【1】オブジェクト情報出力

・処理リスト　追加・確認
・オブジェクト名テーブル更新

【2】クローラー

・処理リスト編集　ツリー更新
・全更新の場合

STEP2. 分析　デモ



・構造の展開・格納
・部分展開
・逆展開
・ソースとの比較

・ファイルの影響調査
・ファイルとプログラムの影響調査

・従属論理ファイル調査
・メニューの検索

STEP3．再構成　デモ

【STEP3】再構成

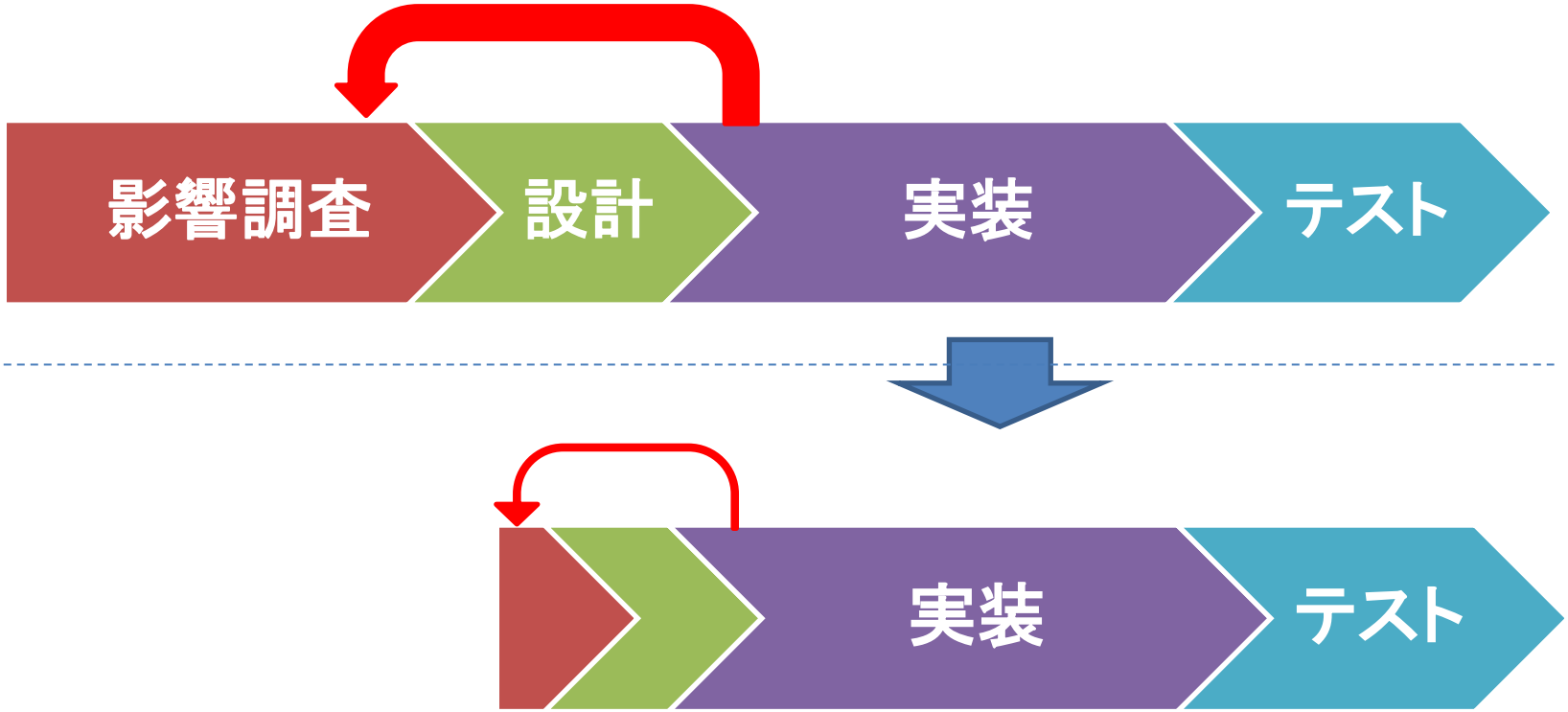
【5】一括コンパイル

・一括リコンパイル　結果確認

アジャイル開発向けシステム

7）効果

■ 効果



時間の短縮

手戻りによる時間ロスの減少

より良い実装とテスト

■ 効果

大規模改修の場合

本番環境 ファイル整備

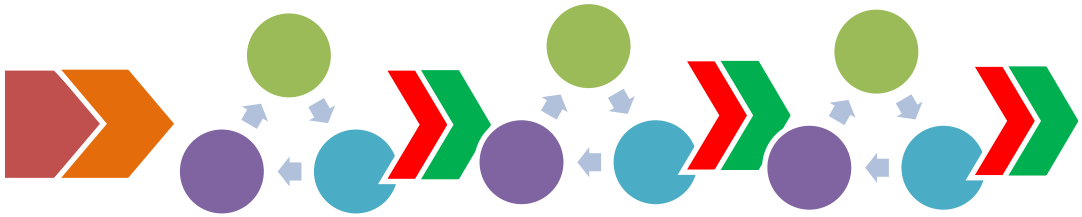
開発環境→本番環境 移行

本番環境テスト・リリース

影響調査 → 設計 → 開発環境 実装 → テスト

連休中に作業

アジャイル開発手法



システム停止時間が短い

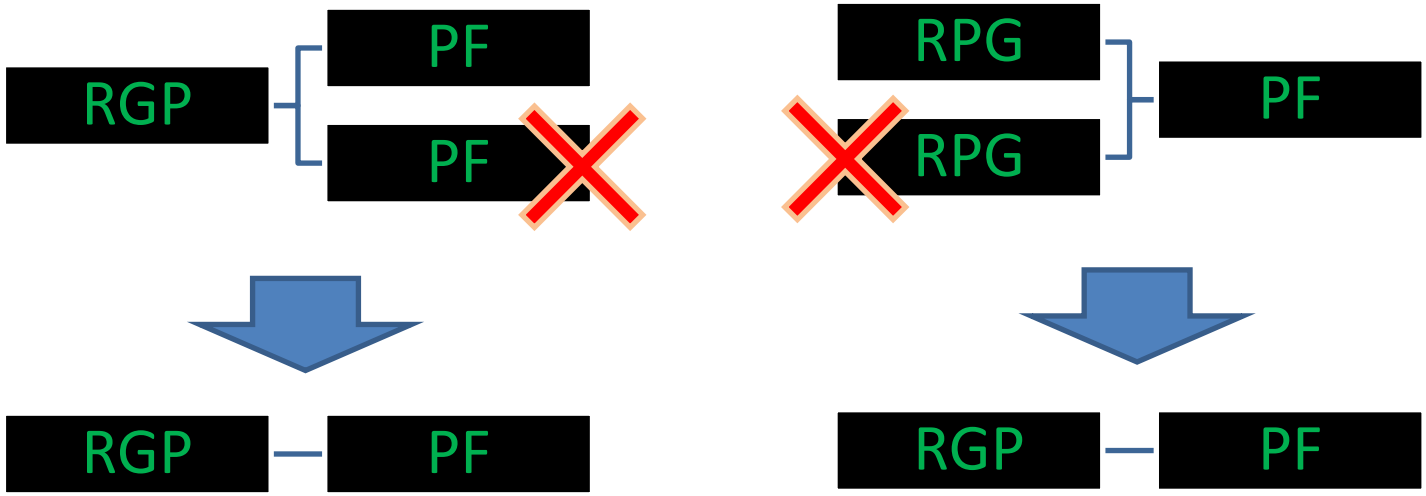
リリースのタイミングが早い。

計画を立てやすい

仕様書はユーザー目線のイメージ資料のみ。

■ 効果

1つにまとめた方が良いファイルやプログラム。



システム全体をシンプルに最適化しながら改修を進められる。

次の改修案件発生の際に、無駄な改修部分が無くなっている。

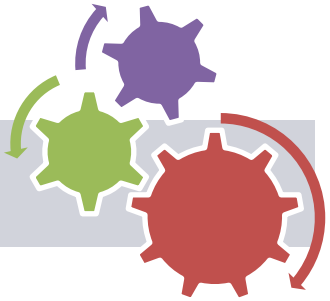
アジャイル開発向けシステム

8）今後の展望

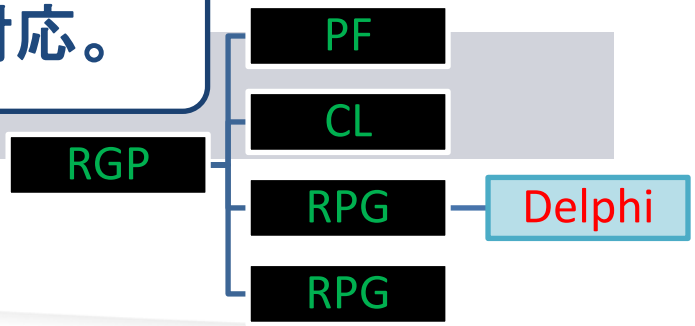
今後の展望

① 各ツールの機能の充実化。

② 構造の収集・メンテナンスの自動化。



③ Delphiのソースファイルへの対応。



アジャイル開発向けシステム

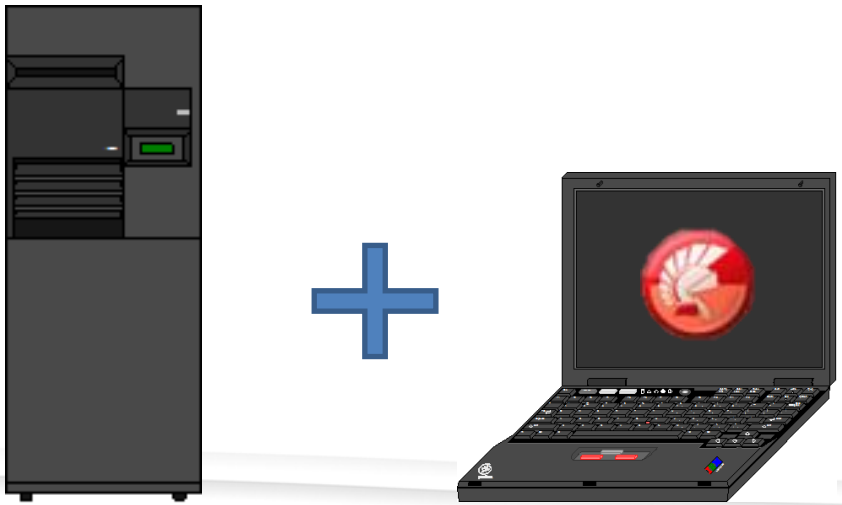
9）まとめ

■ まとめ

5250画面とRPG,CLだけでは難しいが、
Delphi/400を使えば短期間で十分使えるツールが作成可能。

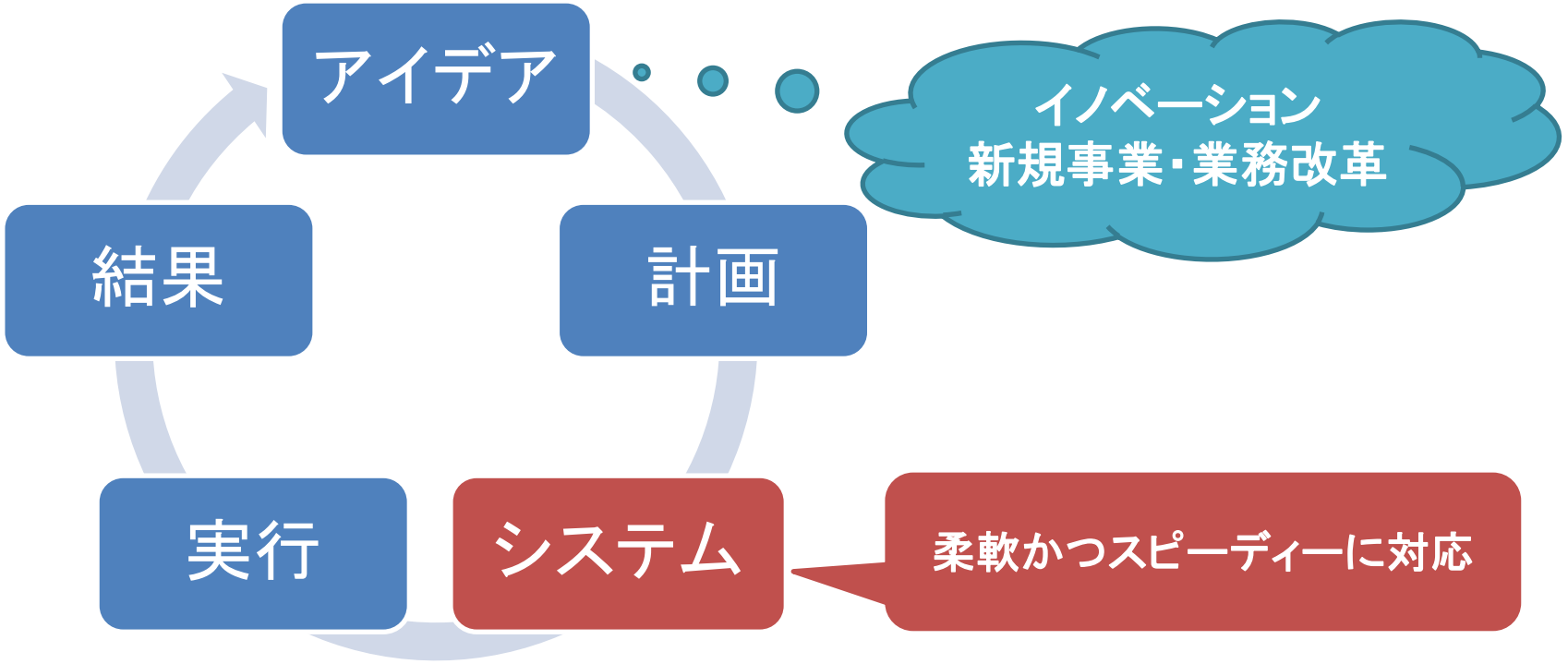
特にグラフィカルなインターフェースと、
高速で動作するSQLが扱えるQueryコンポーネントは有効。

汎用性の高いDelphiのプログラムは開発効率を高めてくれる。



■ まとめ

システム管理者・開発者が楽になるだけ？



価値あるイノベーションの 変化の流れを止めないこと。
変化に強い体制になれることが本当の価値。

ご清聴ありがとうございました。