TETDMの構成の特徴

- マイニング処理と可視化インタフェースを分離
  - 利用者：マイニング手法を固定，複数の結果を並べて分析可能
  - 利用者：複数の処理結果を同一の可視化手法で比較可能
  - 開発者：汎用性を意識したモジュール作りが可能
- 開発は独立に，動作は協調的に連動して
  - 仕様を満たすように作成したモジュールはすべて動作可能
  - 任意のモジュールが，モジュール間で共通の観点に基づいて協調的な再処理動作を行える
  - 可視化インタフェースを入力インタフェースとして利用できインタラクティブに操作を繰り返して分析作業に没入できる

TETDMの社会的意義

- 世の中の創造活動の効率が上がる
  - 電子テキスト，情報を分析する機会は多い
- 最新の研究が幅広く世の中に発信される
  - 共同研究，新技術の開発の効率が高まる
- 技術統合型プロジェクトの促進
  - テキストマイニング以外の分野での類似プロジェクトの促進につながる

利用者のスキルアップ につながる実践

- ツール選択支援インタフェースの開発（中垣内）
  - 複数のモジュールのセット，連動の表示により，
    テキストマイニング初心者のマイニングスキル獲得を支援

メール，ブログ，ツイッター，電子掲示板，レポートなど，
卑近なテキストデータに対する活用の可能性を広げ，
世の中のICT活用力の底上げを図っていきたい

- 学部3年次学生を対象にしたセミナー（梶並）
  - テキストマイニングに関する理論やアルゴリズムを，
    実際にプログラムを作成して確認

実践，実体験的にテキストマイニングを理解できる
枠組みになるよう拡張していきたい

利用者と開発者のスキルアップにつながる実践

病院の電子カルテの監査に向けたシステム構築

- TETDMの枠組みを前提として，システムの提案，構築に向けて共通のイメージを作りやすい

- 専門用語抽出支援ツールの開発（高間）
- 新人とベテランの電子カルテ比較ツールの開発[谷 13]
  - 個々のモジュールを利用者が利用，評価してマイニングについての理解を深めやすい
  - 開発者側は利用者のニーズを引き出しやすくニーズに応じたツールの開発を検討できる

効率よく，利用者と開発者がコミュニケーションをとれる枠組み（TETDMサイト，仕様）を提供していきたい

チャレンジの背景(利用者)

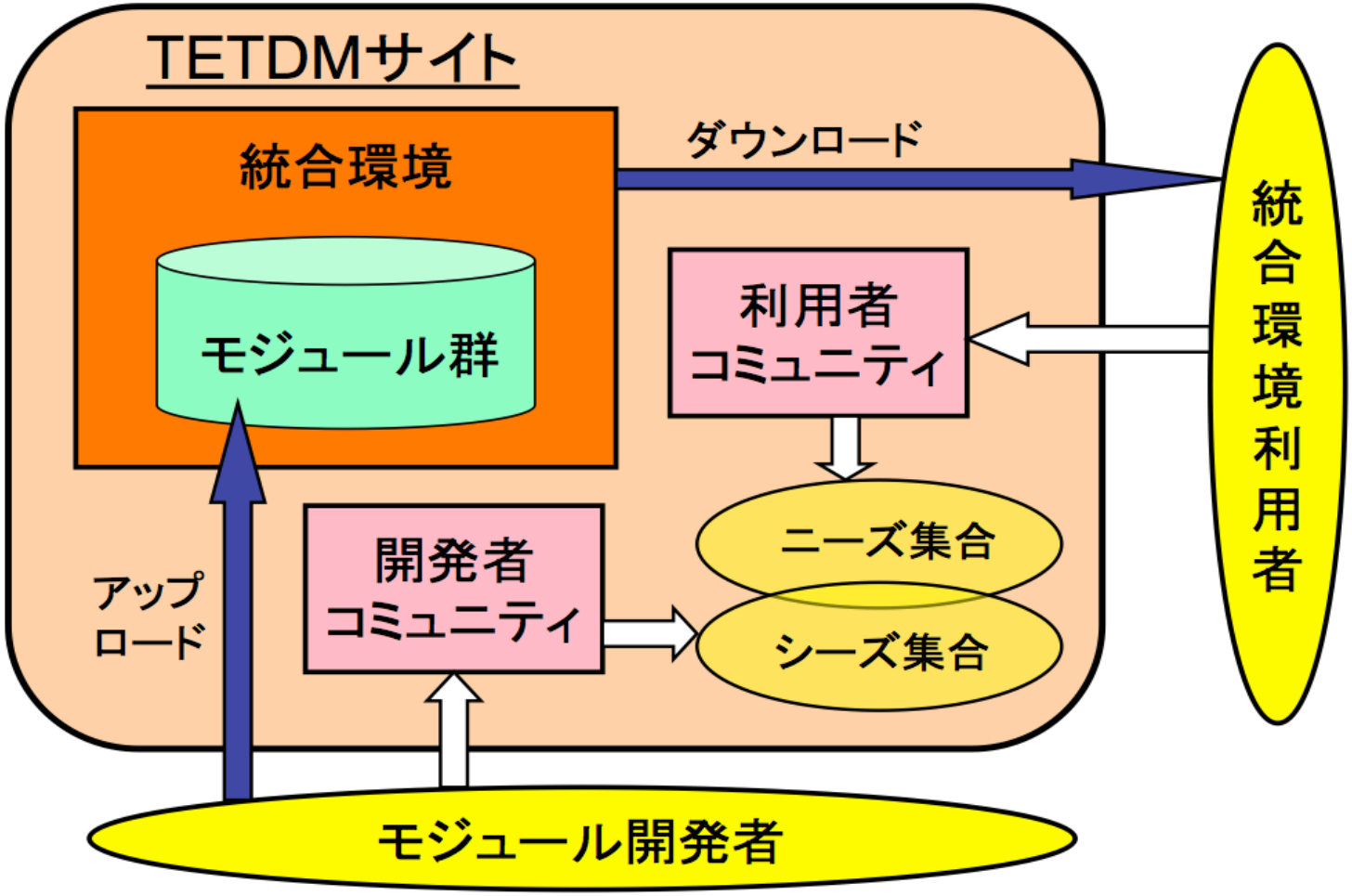
- 複数の技術を用いたい
  - システムの収集や連携の手間を省きたい
  - 集中して分析作業を行いたい
- 多様かつ最新の技術を使いたい
  - 確立した技術でなくとも幅広く使いたい
  - 技術の実用化までの時間を短縮してほしい

利用者と開発者の拡大に向けて

- 利用者:
  - TETDMの利用方法の整備
  - TETDMの活用事例の提示
  - TETDMの利用意欲の促進
- 開発者:
  - ツールの開発支援環境の整備
  - ツールの開発意欲の促進
  - ツールの開発対象領域の拡大

利用者，開発者，の参加のためのしきい値を，できる限り下げる

TETDMサイトの構成



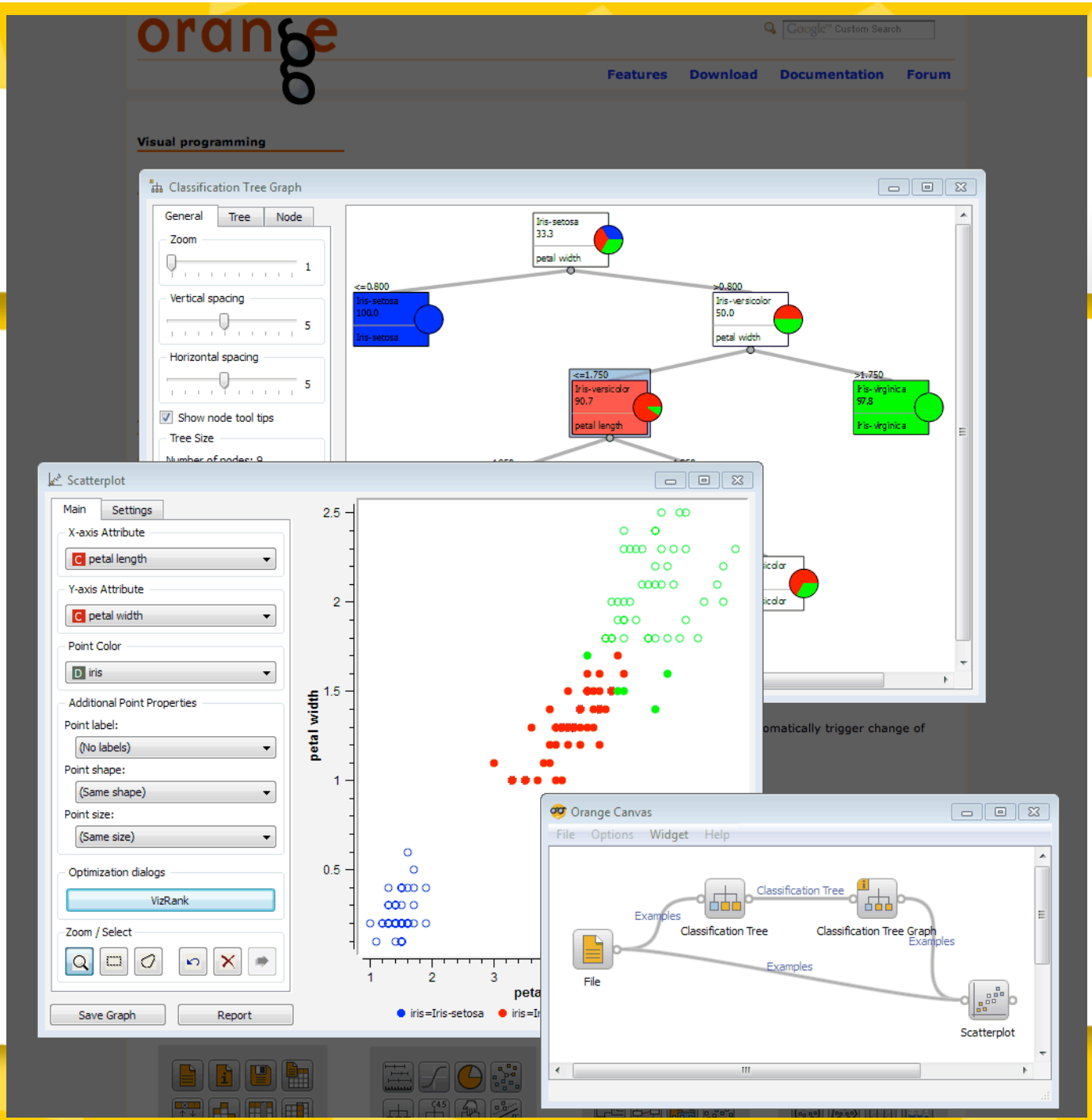
コミュニティの形成に向けて

- 利用者と開発者の多様なニーズとシーズのマッチングを実現したい
- 利用者主導の開発も可能な枠組みとする
- TETDMサイトの構築と活用
- 人工知能学会
「インタラクティブ情報アクセスと可視化マイニング」
研究会を発足(2012年4月から)

オンライン，オフラインの両面から
コミュニティの形成とサポートを行っていきたい

統合型データマイニング ソフトウェア

- フリーソフト
  - R, Weka, orange [R-Project, Weka, orange]
  - 統計，機械学習などの標準的な処理を実装
- 商用ソフト
  - DIAMining, Text Mining Studio, TRUE TELLER, Text Mining for Clementine
    [DIAMining, Mining Studio, TELLER, Clementine]
  - 主に確立した技術を収集, コスト面


orange
Features
Download
Documentation
Forum
Visual programming
Classification Tree Graph
General
Tree
Node
Zoom
Vertical spacing
Horizontal spacing
Show node tool tips
Tree Size
Iris-setosa
33.3
petal width
<=0.800
Iris-setosa
100.0
Iris-setosa
>0.800
Iris-versicolor
50.0
petal width
<=1.750
Iris-versicolor
90.7
petal length
>1.750
Iris-virginica
97.8
Iris-virginica
Scatterplot
Main
Settings
X-axis Attribute
petal length
Y-axis Attribute
petal width
Point Color
iris
Additional Point Properties
Point label:
(No labels)
Point shape:
(Same shape)
Point size:
(Same size)
Optimization dialogs
VizRank
Zoom / Select
Save Graph
Report
petal width
iris=Iris-setosa
Orange Canvas
File
Options
Widget
Help
Examples
Classification Tree
Classification Tree
Classification Tree Graph
Examples
Examples
File
Scatterplot

関連する統合環境プロジェクト

- VidaMine[Kinami 03]
  - データマイニングのための統合環境として開発
- U-Compare[狩野 08]
  - テキストマイニングのモジュールを収集

TETDMでは,
可視化インタフェースによるモジュール間の連動を実現
簡易な仕様でテキストマイニング以外のツールも集める

参考URL

[R-Project] http://www.r-project.org/

[Weka] http://www.cs.waikato.ac.nz/ml/weka/

[orange] http://www.ailab.si/orange/

[DIAMining] http://www.mdis.co.jp/products/diamining/

[Mining Studio] http://www.msi.co.jp/tmstudio/

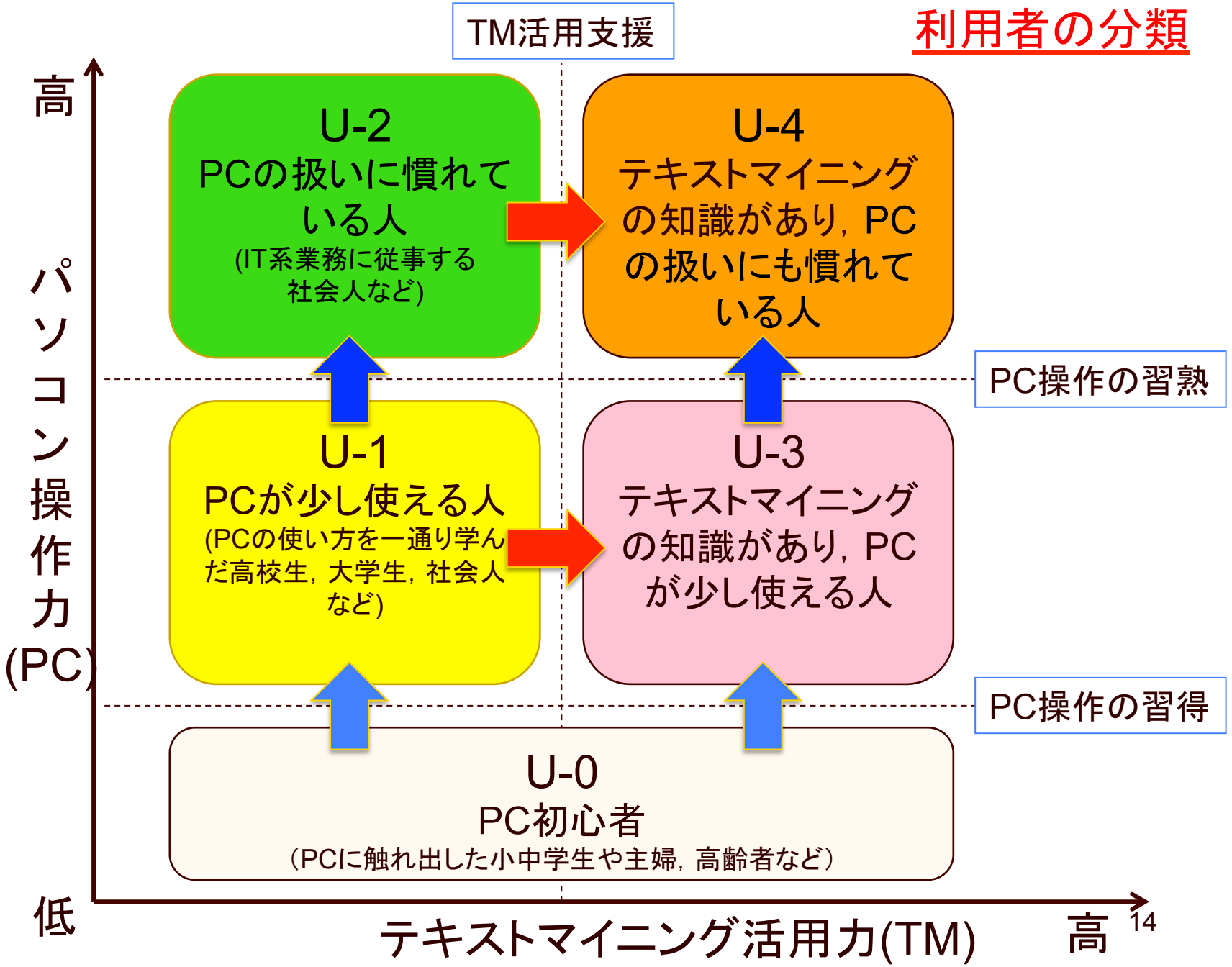
[TELLER] http://www.trueteller.net/

[Clementine] http://www.spss.co.jp/software/modeler_ta/

- ツール開発関連

- プログラミングの知識があり
  ツール（モジュール）開発にご興味がある方
  処理手順に興味のある方は
  以降のページをご覧下さい.

利用者の分類


TM活用支援
高
パソコン操作力(PC)
低
U-2
PCの扱いに慣れている人
(IT系業務に従事する社会人など)
U-4
テキストマイニングの知識があり，PCの扱いにも慣れている人
U-1
PCが少し使える人
(PCの使い方を一通り学んだ高校生，大学生，社会人など)
U-3
テキストマイニングの知識があり，PCが少し使える人
U-0
PC初心者
（PCに触れ出した小中学生や主婦，高齢者など）
PC操作の習熟
PC操作の習得
テキストマイニング活用力(TM)
高

チャレンジの背景(開発者)

- 関連技術の収集や比較を容易にしたい
  - 既存技術の拡張，融合
  - 開発したシステムの評価実験
- 研究成果と実用化が結びつけ
  モチベーションを高めたい
  - 開発したシステムを多くの人に使ってもらいたい

開発者のスキルアップ につながる実践

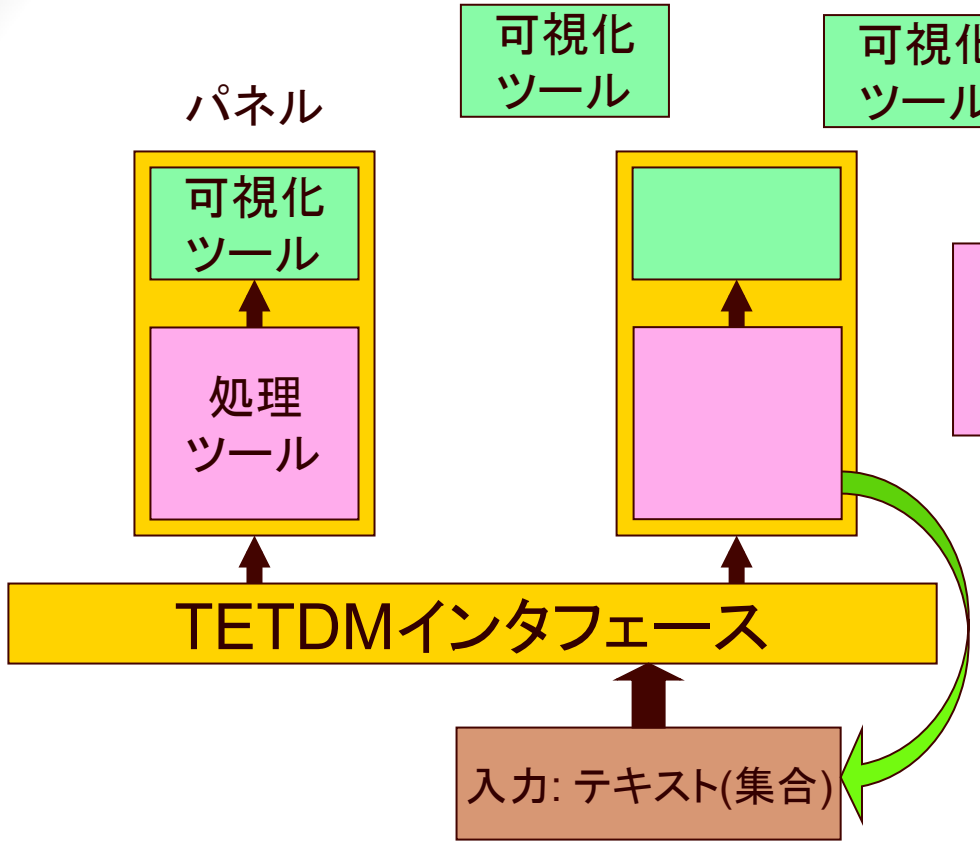
- 博士前期課程学生を対象とした演習（複数の大学で実施済）
  - マイニング処理に特化したモジュール作りにより，処理の内容，面白さ，アルゴリズムを比較可能

モジュール開発におけるプラットフォームとして，
開発者間の連携を図りやすくする支援を行っていきたい

- 特別研究における研究テーマ（徳永）
  - 複数の処理と処理結果を並列に動作，提示できる枠組みにより，既存システムの拡張や新しい組合せによるシステム作りが可能

再利用可能な，汎用性の高いモジュール作りを
積極的に支援できる枠組みに拡張していきたい

TETDMの拡張可能性



テキストマイニング以外の
処理でもOK

処理
ツール

処理ツールが取得，
生成するテキストの
再入力も可能

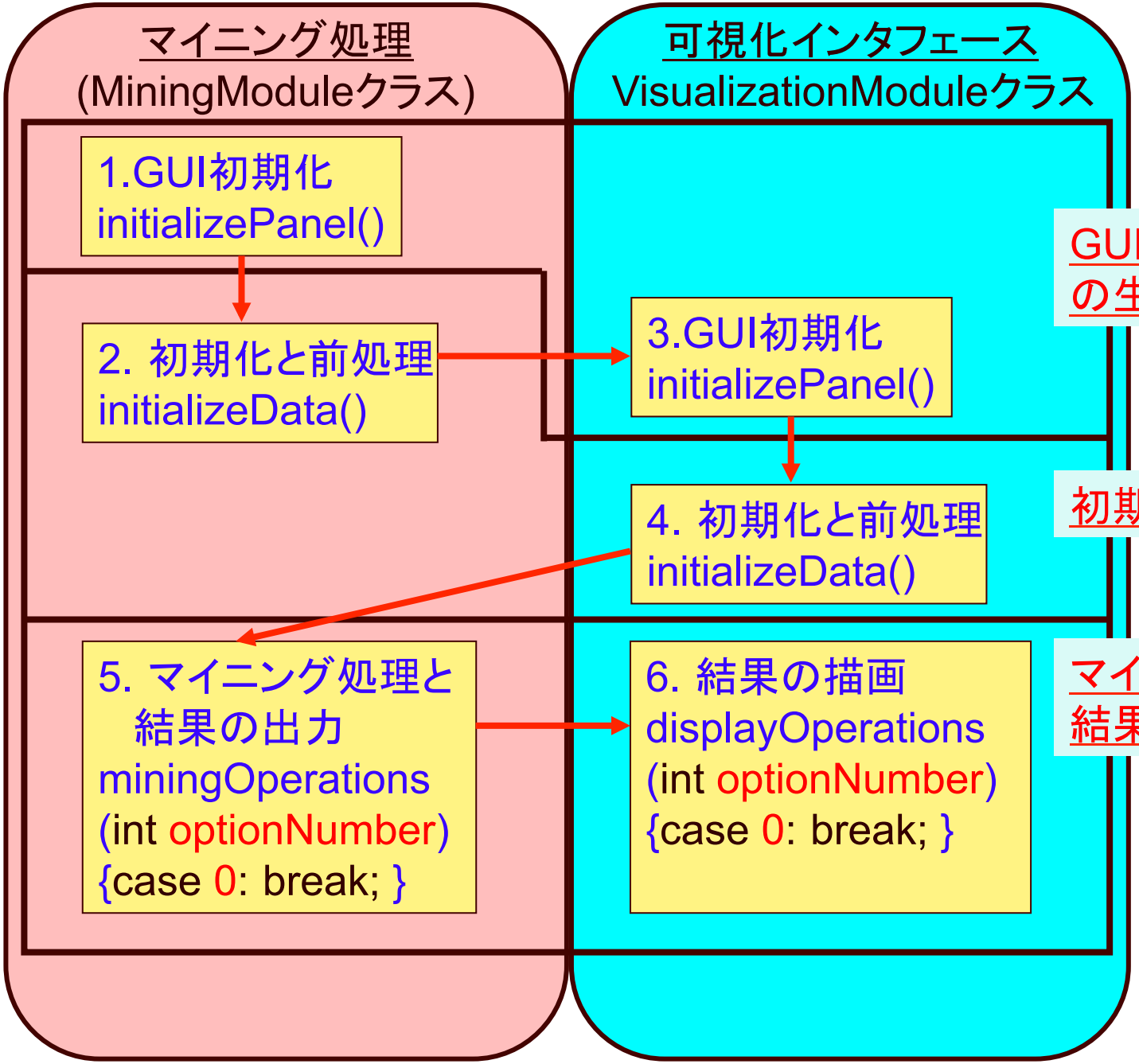
モジュール作成の概要

マイニング処理と結果の描画

作成したGUIで動作を制御

本質的な部分のみを作成して
既存のモジュールとの組合せを検討できる

自動実行処理フロー


マイニング処理
(MiningModuleクラス)
可視化インタフェース
VisualizationModuleクラス
1.GUI初期化
initializePanel()
2. 初期化と前処理
initializeData()
3.GUI初期化
initializePanel()
4. 初期化と前処理
initializeData()
5. マイニング処理と
結果の出力
miningOperations
(int optionNumber)
{case 0: break; }
6. 結果の描画
displayOperations
(int optionNumber)
{case 0: break; }
GUIコンポーネント
の生成
初期化と前処理
マイニング処理と
結果の描画

マイニング処理モジュールの基本構成

マイニング処理クラス (MiningModuleクラスを継承)

コンストラクタ: クラス名()

[自動実行メソッド1] GUI初期化: initializePanel()

[自動実行メソッド2] 初期化と前処理: initializeData()

[自動実行メソッド5] マイニング処理と結果の出力:

```
miningOperations(int optionNumber){
        switch(optionNumber){
                case 0: break;  //自動実行対象
                case ?: break;
        }
}
```

[自動実行] GUI操作時処理: actionPerformed ()

上記自動実行メソッドから呼び出されるメソッド

赤枠は
実装必須

マイニング処理モジュールの基本構成

マイニング処理クラス (MiningModuleクラスを継承)

コンストラクタ: クラス名()

[自動実行メソッド1] GUI初期化: initializePanel()

[自動実行メソッド2] 初期化と前処理: initializeData()

[自動実行メソッド5] マイニング処理と結果の出力:

```
miningOperations(int optionNumber){
        switch(optionNumber){
                case 0: break;  //自動実行対象
                case ?: break;
        }
}
```

[自動実行] GUI操作時処理: actionPerformed ()

上記自動実行メソッドから呼び出されるメソッド

赤枠は
実装必須

可視化インタフェースモジュールの基本構成

可視化インタフェースクラス(VisualizationModuleクラスを継承)

コンストラクタ: クラス名()

赤枠は
実装必須

[自動実行メソッド3] GUI初期化: initializePanel()

[自動実行メソッド4] 初期化と前処理: initializeData()

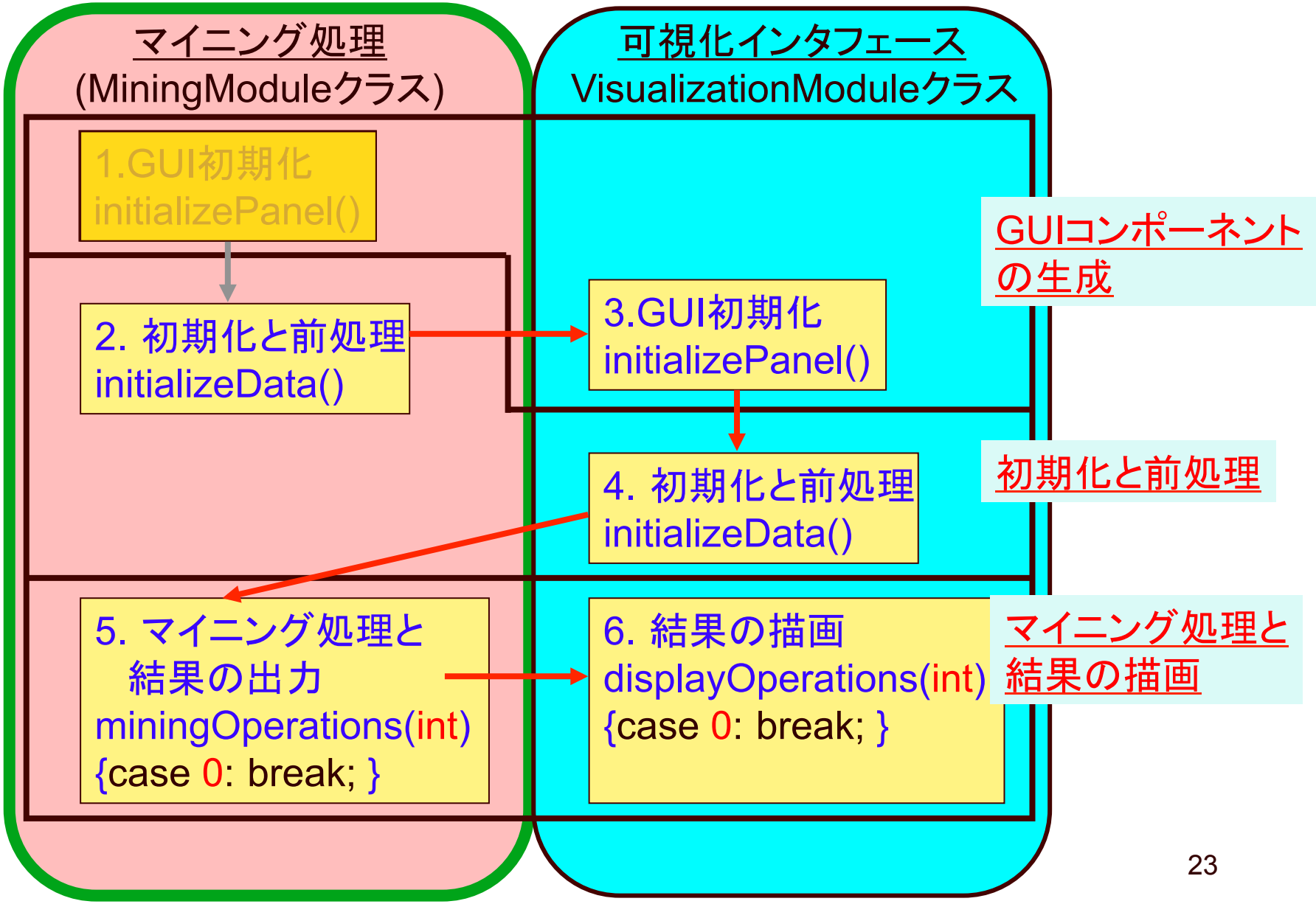
[自動実行メソッド6] マイニング処理結果の描画:

```
displayOperations(int optionNumber){
        switch(optionNumber){
                case 0: break;  //自動実行対象
                case ?: break;
        }
}
```

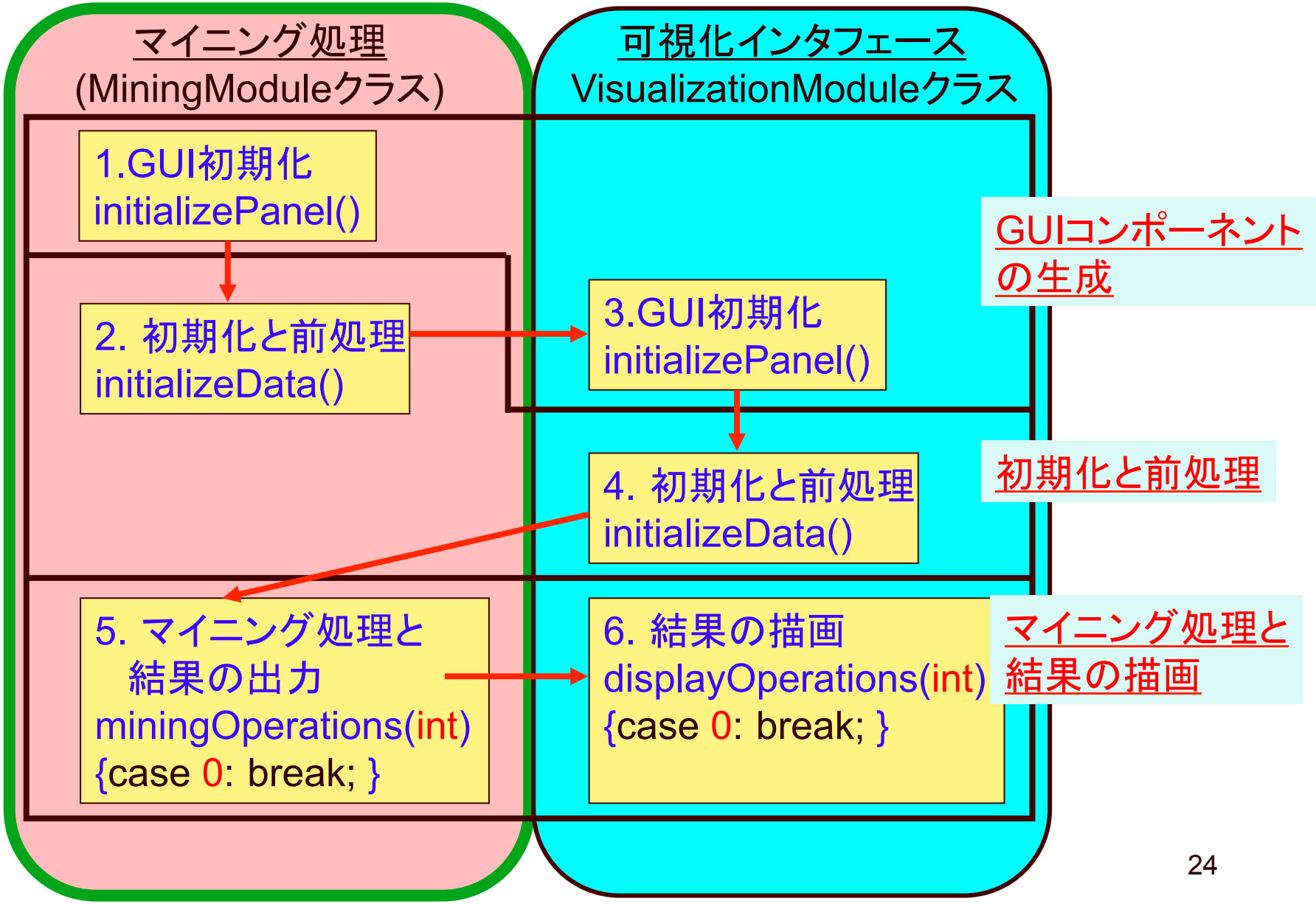
上記自動実行メソッドから呼び出されるメソッド

マイニング処理結果受け取り用メソッド setData(int dataID, **)

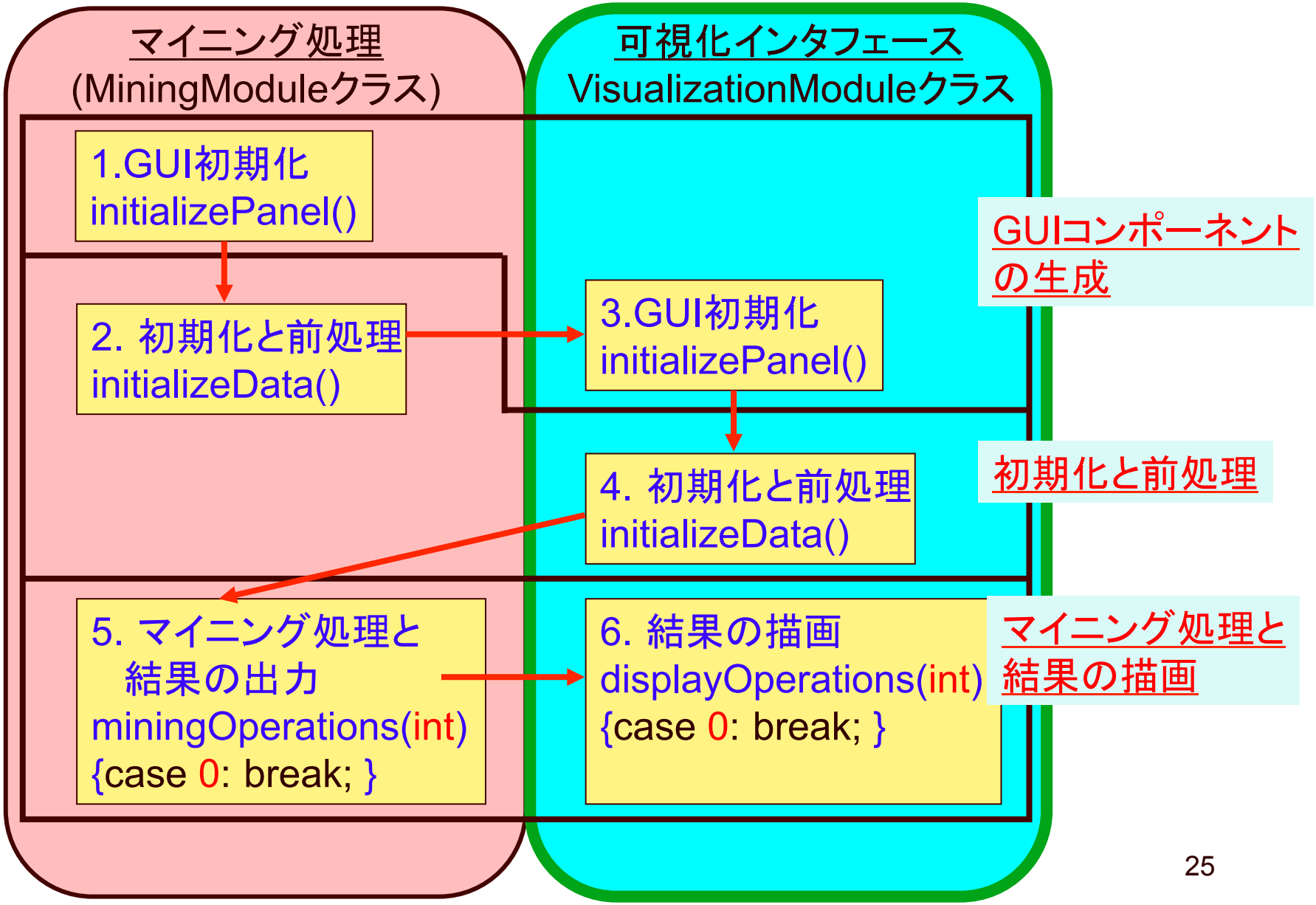
自動実行処理フロー


マイニング処理
(MiningModuleクラス)
可視化インタフェース
VisualizationModuleクラス
1.GUI初期化
initializePanel()
2. 初期化と前処理
initializeData()
3.GUI初期化
initializePanel()
4. 初期化と前処理
initializeData()
5. マイニング処理と
結果の出力
miningOperations(int)
{case 0: break; }
6. 結果の描画
displayOperations(int)
{case 0: break; }
GUIコンポーネントの生成
初期化と前処理
マイニング処理と結果の描画

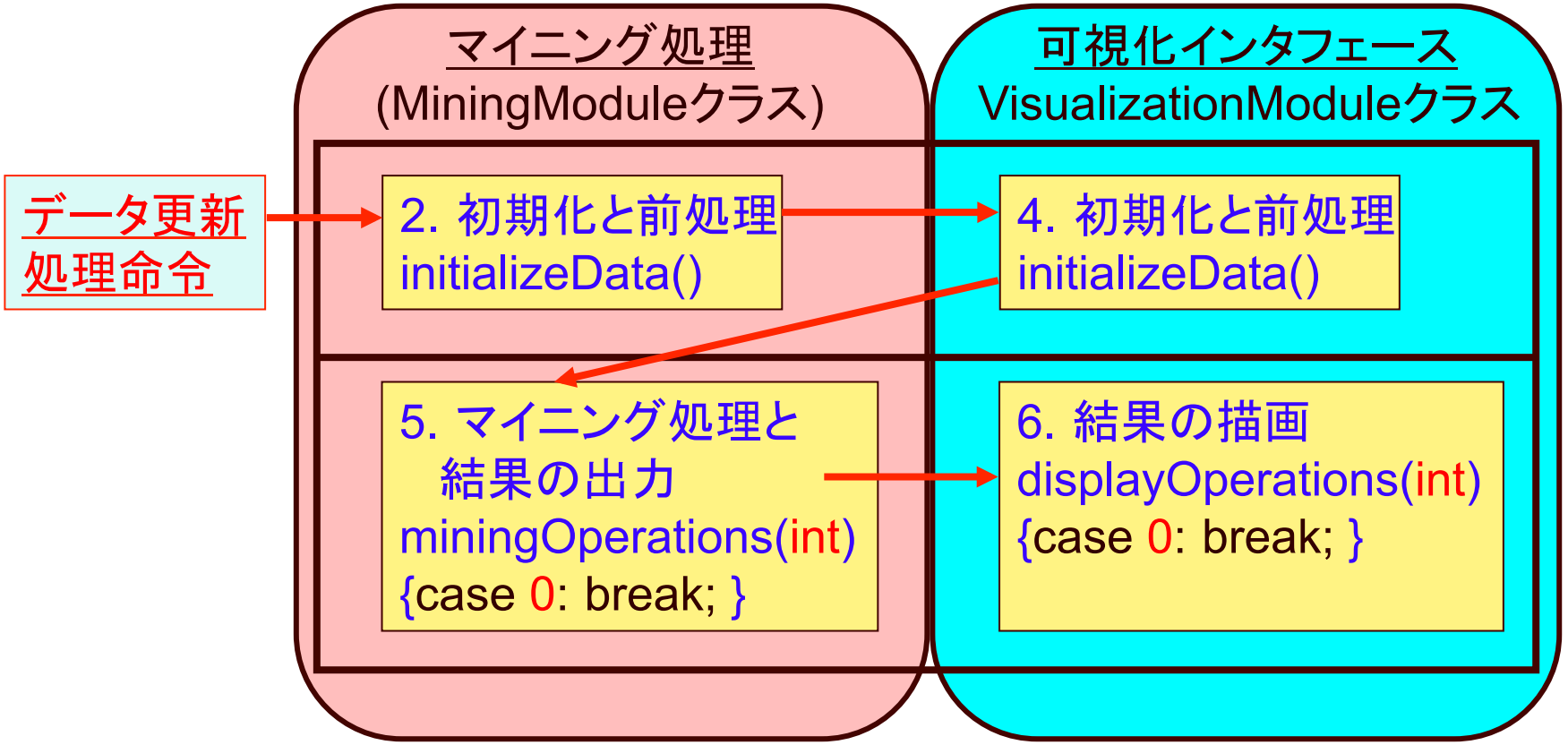
自動実行処理フロー


マイニング処理
(MiningModuleクラス)
可視化インタフェース
VisualizationModuleクラス
1.GUI初期化
initializePanel()
2. 初期化と前処理
initializeData()
3.GUI初期化
initializePanel()
4. 初期化と前処理
initializeData()
5. マイニング処理と
結果の出力
miningOperations(int)
{case 0: break; }
6. 結果の描画
displayOperations(int)
{case 0: break; }
GUIコンポーネントの生成
初期化と前処理
マイニング処理と結果の描画

自動実行処理フロー


マイニング処理
(MiningModuleクラス)
可視化インタフェース
VisualizationModuleクラス
1.GUI初期化
initializePanel()
2. 初期化と前処理
initializeData()
3.GUI初期化
initializePanel()
4. 初期化と前処理
initializeData()
5. マイニング処理と
結果の出力
miningOperations(int)
{case 0: break; }
6. 結果の描画
displayOperations(int)
{case 0: break; }
GUIコンポーネント
の生成
初期化と前処理
マイニング処理と
結果の描画

データ更新時処理フロー



連動処理

- 可視化連動:
  複数の可視化インタフェース上の表示を同時に変更
  - 複数のインタフェース上で対応するデータを明示

- 処理連動:
  複数の処理モジュールの処理を同時に実行
  - 複数のモジュールが同時に処理結果を変更する

可視化連動

- 1. 表示サイズの変更:
  複数の可視化インタフェース上の表示サイズを同時に変更

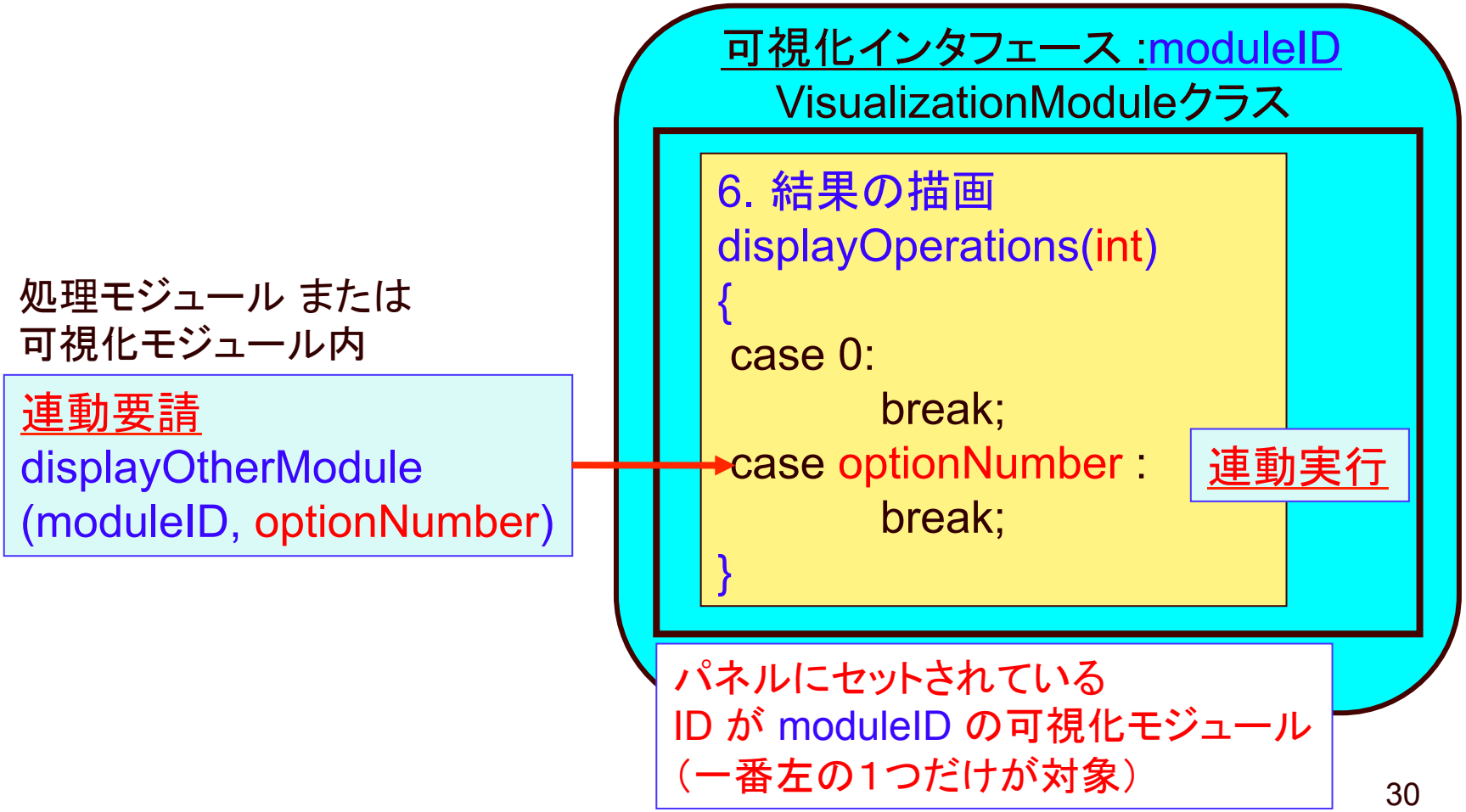
- 2. オプション番号による再表示:
  可視化インタフェースの表示オプションを
  他のモジュールから変更

- 3. フォーカス情報の強調表示:
  ある可視化インタフェース上でユーザが注目している
  データを他の可視化インタフェース上で強調表示

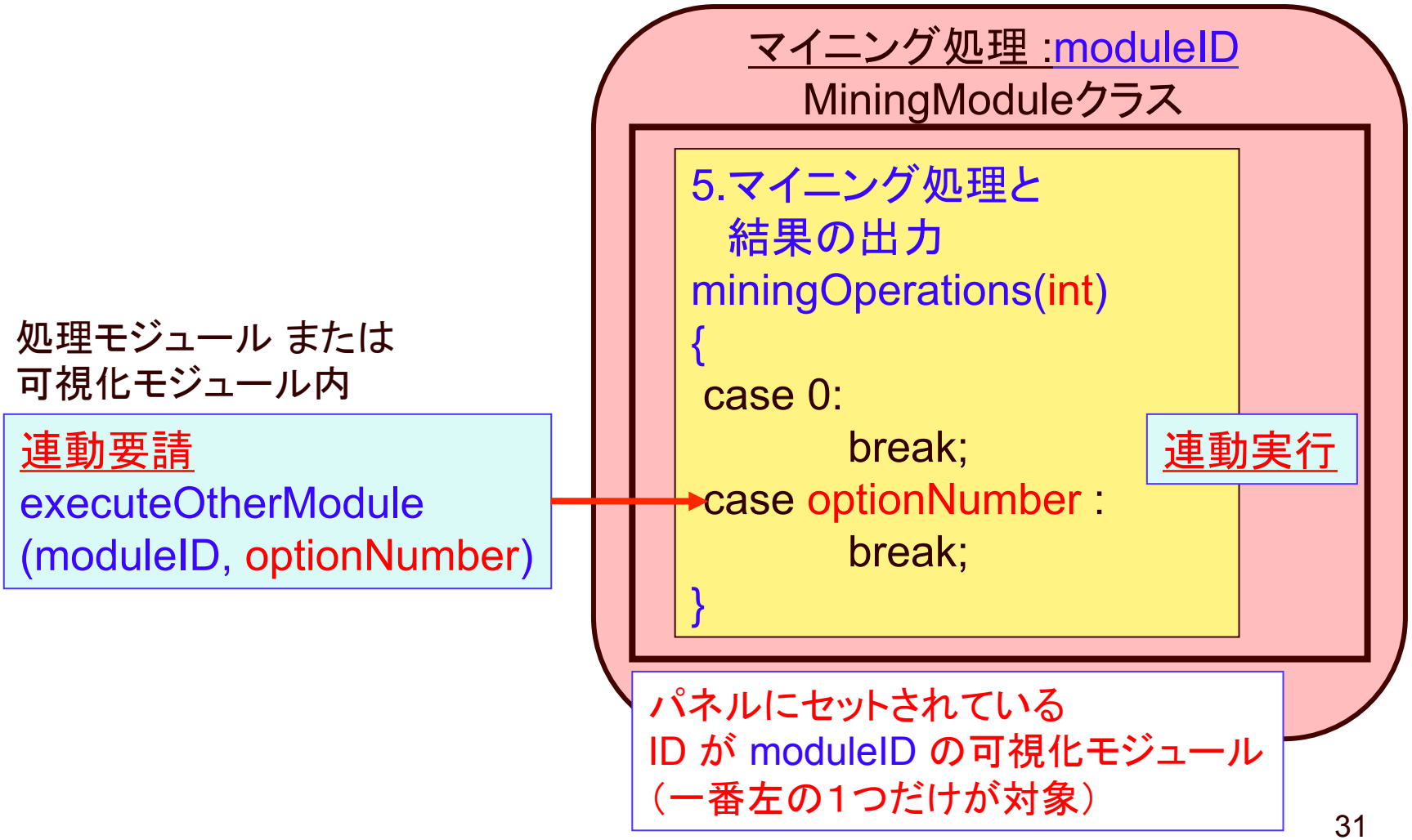
処理連動

- 1. 入力テキスト情報の更新による処理実行:
  入力テキストを編集，保存した後，
  すべての処理モジュールが再び処理を実行
- 2. オプション番号を利用した処理実行:
  処理モジュールの処理オプションを
  他のモジュールから変更
- 3. フォーカス情報を利用した処理実行:
  処理モジュールが，入力パラメータの1つに
  フォーカス情報を利用して処理を実行
- 4. データ取得による処理実行:
  処理モジュールが，他の処理モジュールの出力を利用

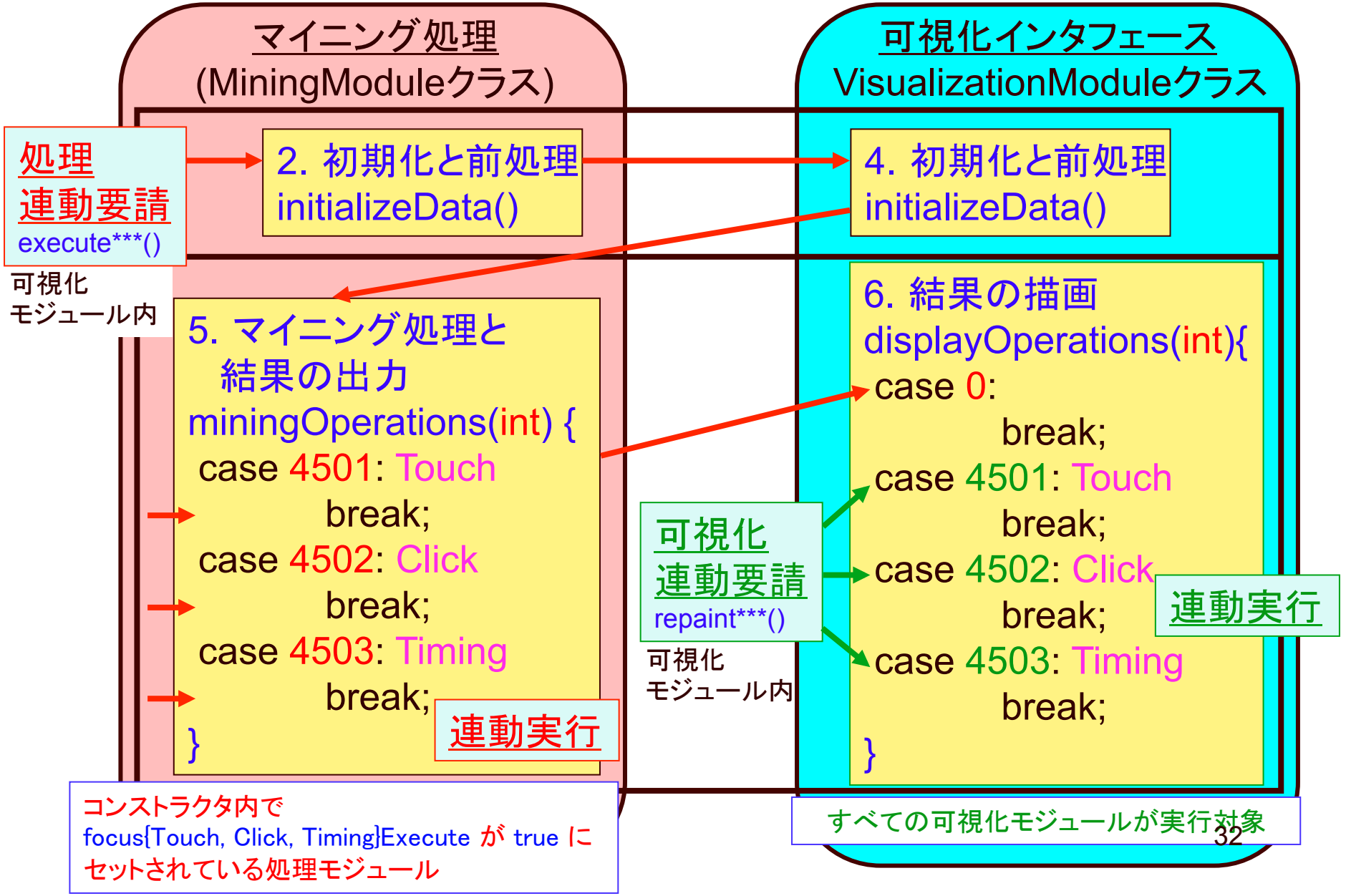
オプションによる可視化連動の処理フロー



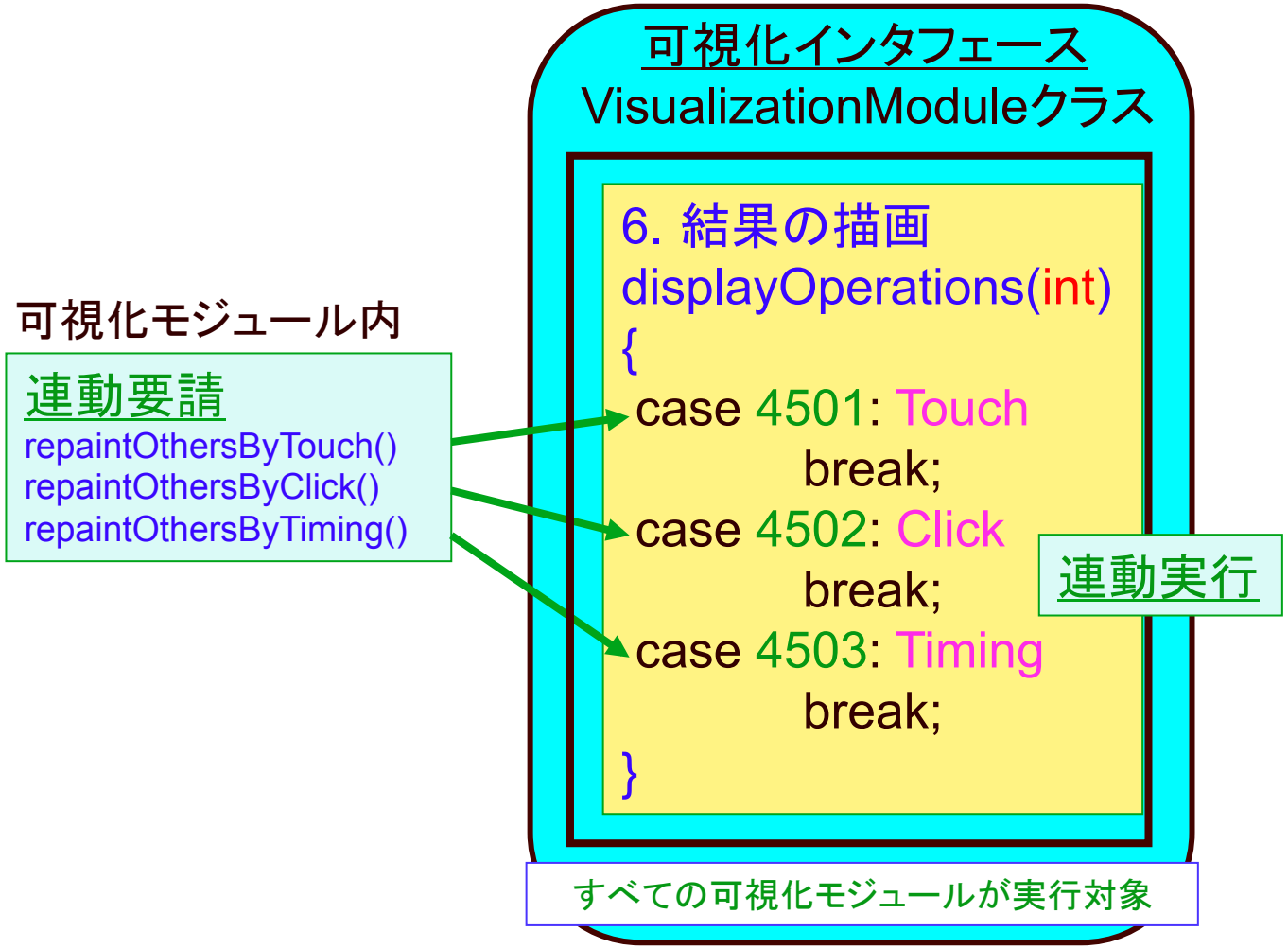
オプションによる処理連動の処理フロー



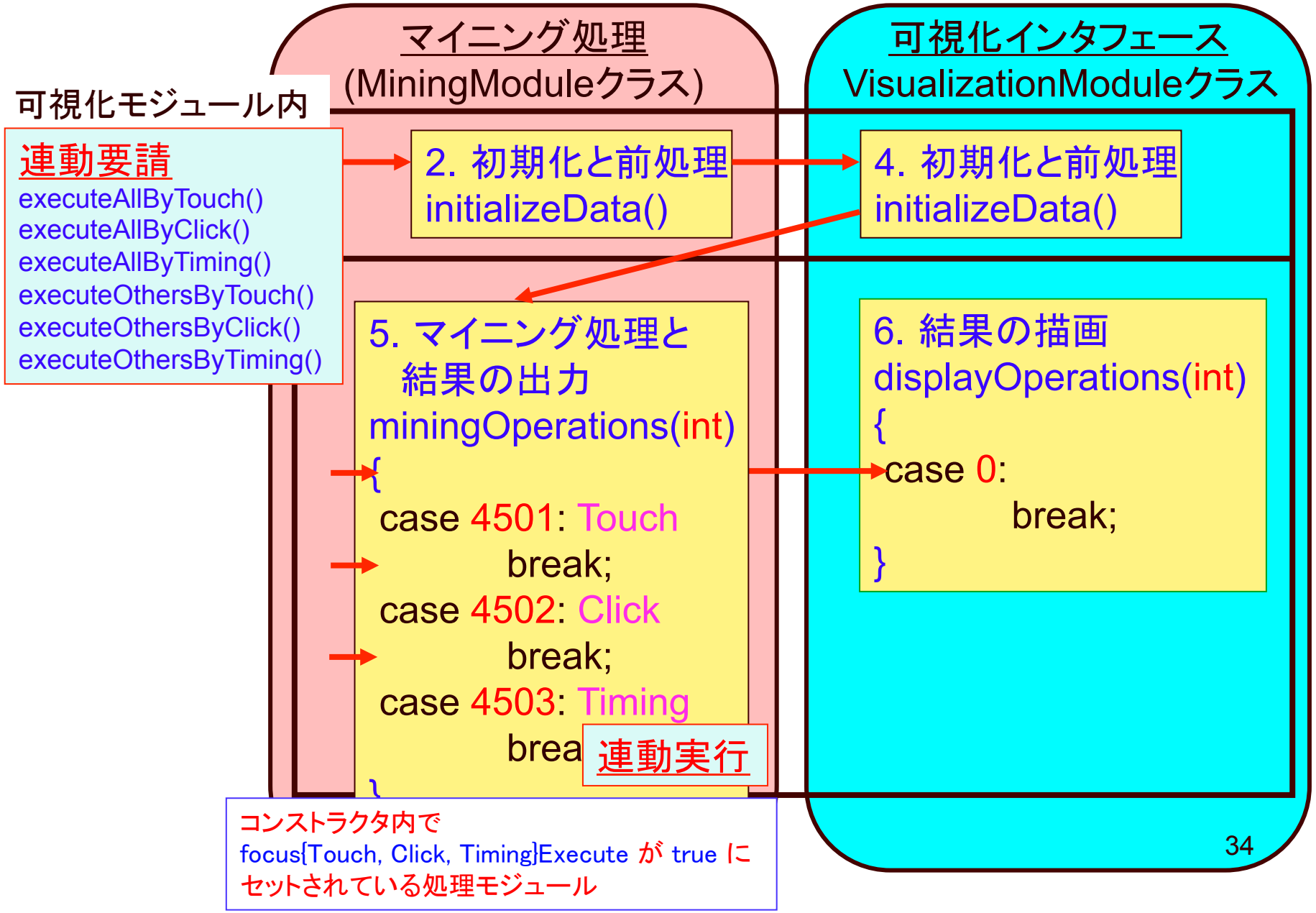
フォーカス連動の処理フロー



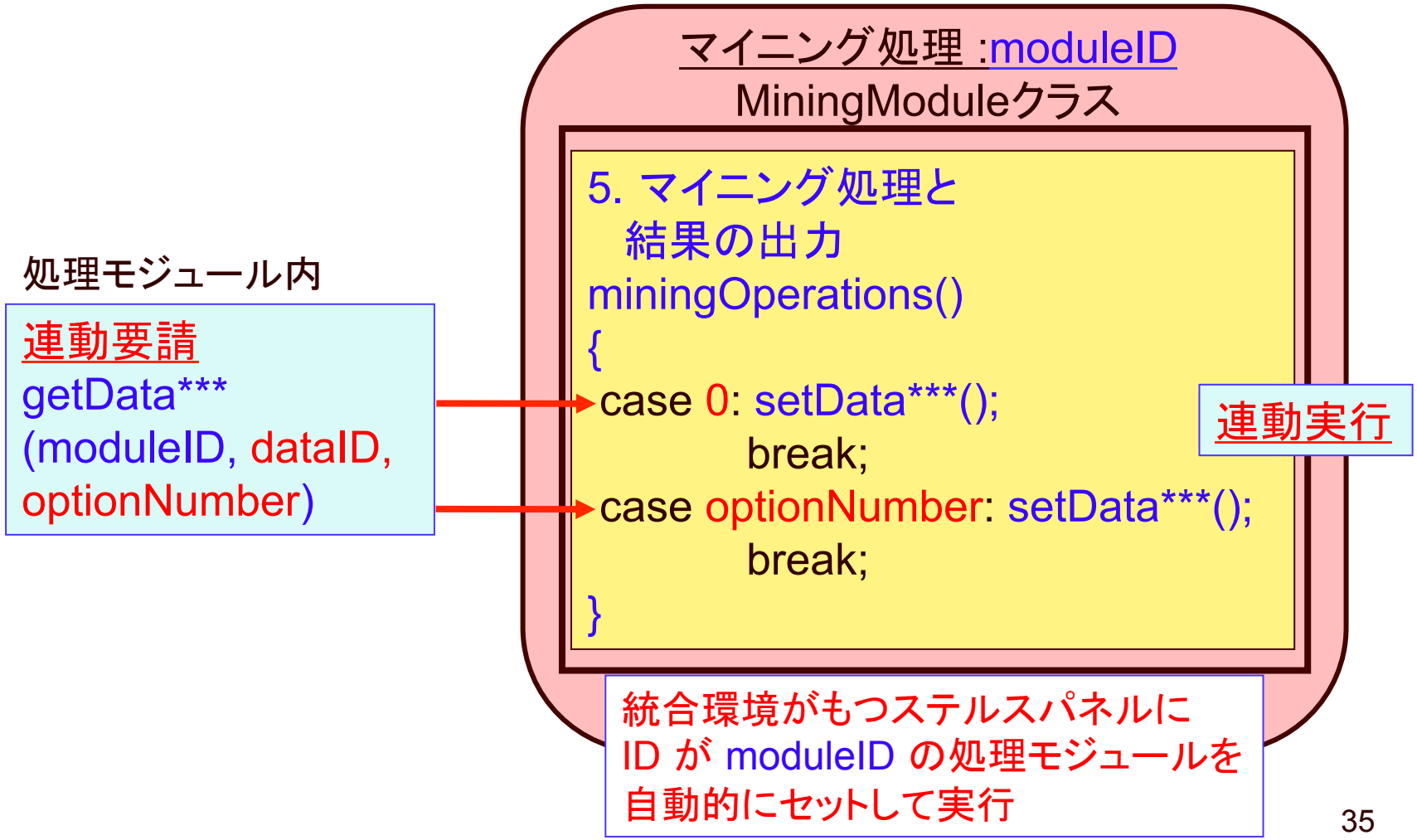
フォーカス連動の処理フロー（可視化）



フォーカス連動の処理フロー（処理）



データ取得による処理連動の処理フロー



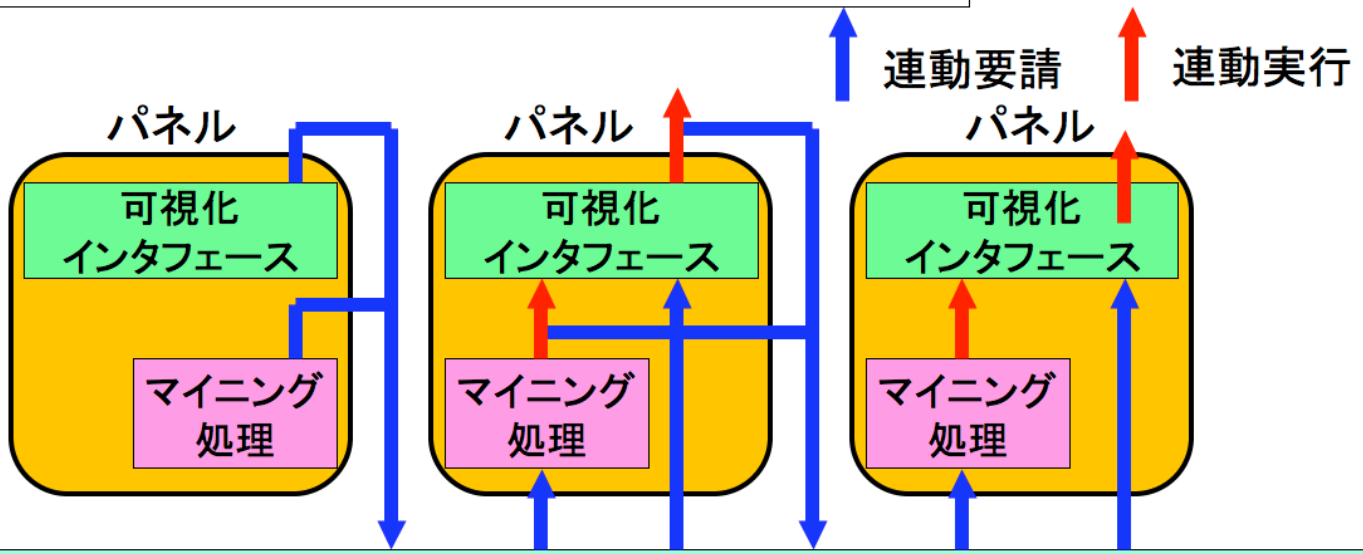
オプション番号を利用した連動

連動要請メソッド:

再描画: displayOtherModule(int displayModuleID, int optionNumber)
処理実行:executeOtherModule(int executeModuleID, int optionNumber)

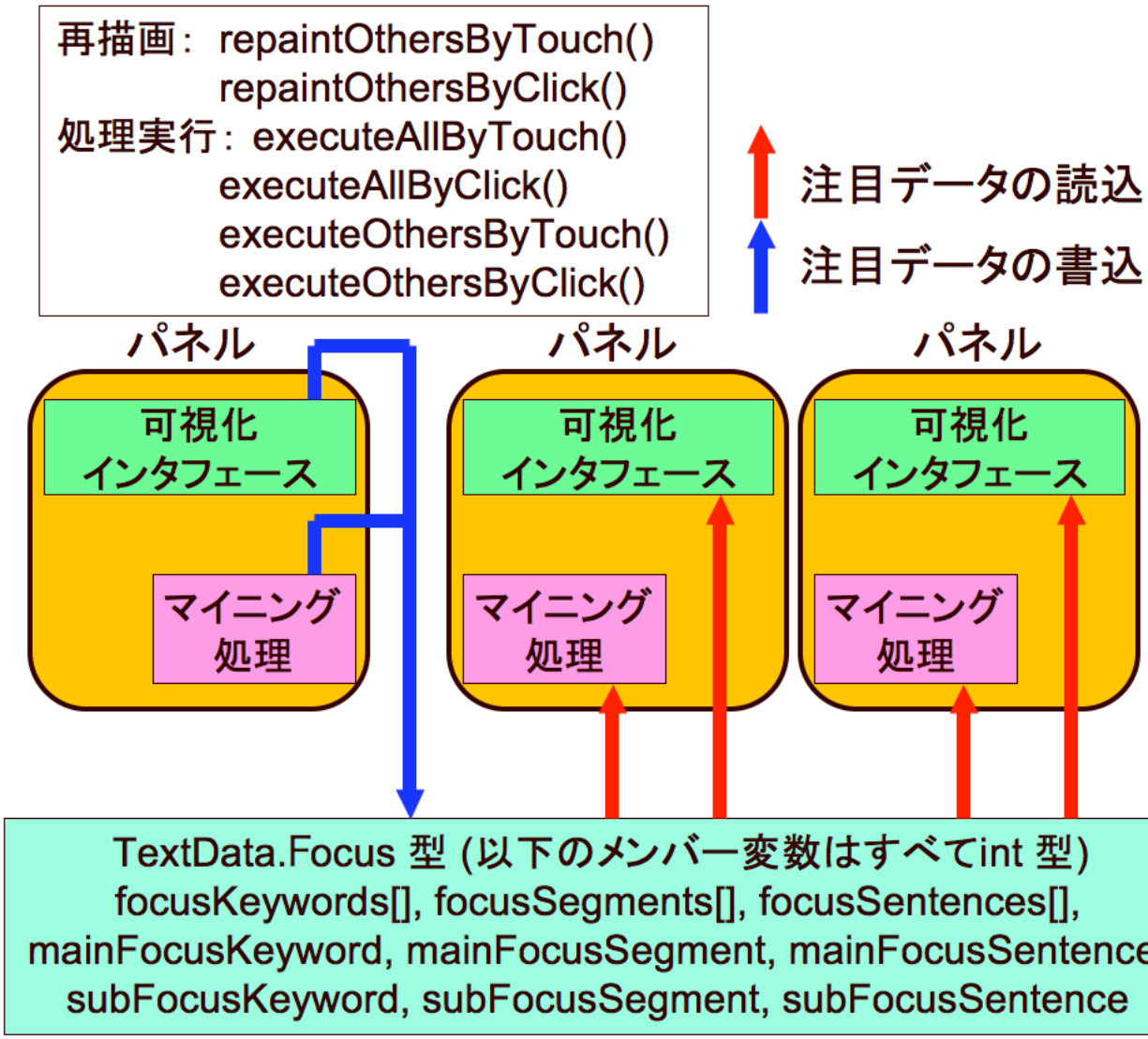
連動実行メソッド:

可視化モジュール:displayOperations(int optionNumber)
処理モジュール: miningOperations(int optionNumber)



Connection 型(TextData型でインスタンス化) 内部で処理を管理
1. パネルの中から，連動実行の対象モジュールがセットされているパネルを探す
（同一モジュールが複数ある場合，最初に見つかったパネル（一番左）が連動の対象）
2. 連動実行先が連動要請を出すことも可能
（最初の連動要請を起点として，各パネルは1回のみ連動して実行される）

フォーカス情報を利用した連動



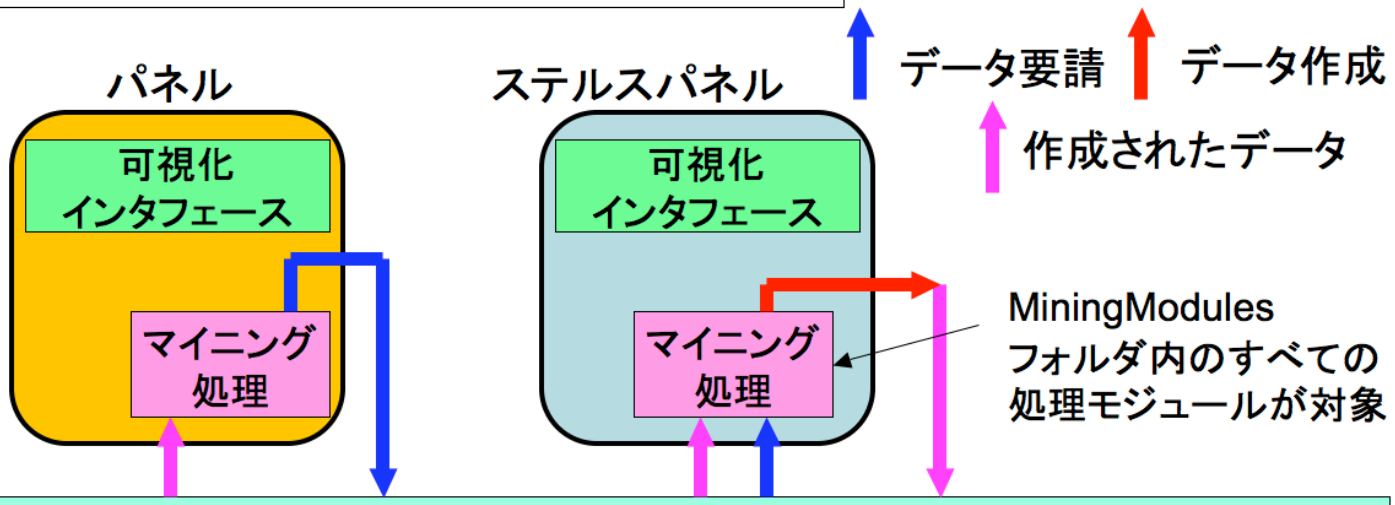
他のモジュールの出力を利用した連動

データ要請メソッド： *** の部分は，Integer(Array), Double(Array), String(Array), の6種類

処理モジュール：** getData*** (int getModuleID, int dataID)

データ作成メソッド： ** の部分は，int([]), double([]), String([]), の6種類

処理モジュール：void setData(int dataID, ** data)



リスト構造のデータベース：moduleIDとdataID で区別して データを蓄積

Connection 型(TextData型でインスタンス化) 内部で処理を管理

1. データベースに要請するデータがあれば，それを返す
2. データがなければ，ステルスパネルに対象モジュールをセット
   （miningOperations(0) を実行）してデータを作成
3. データが作成されない場合，miningOperations(dataID) を実行してデータを作成
   （データ作成のために，新たにデータ要請を行う場合もある）