IPSJ/SIGSE **パターンワーキンググループ第2回勉強会資料** (2004.1.16)
(**改訂版** 2004.1.20)

ツールを使ったパターンとの付き合い方

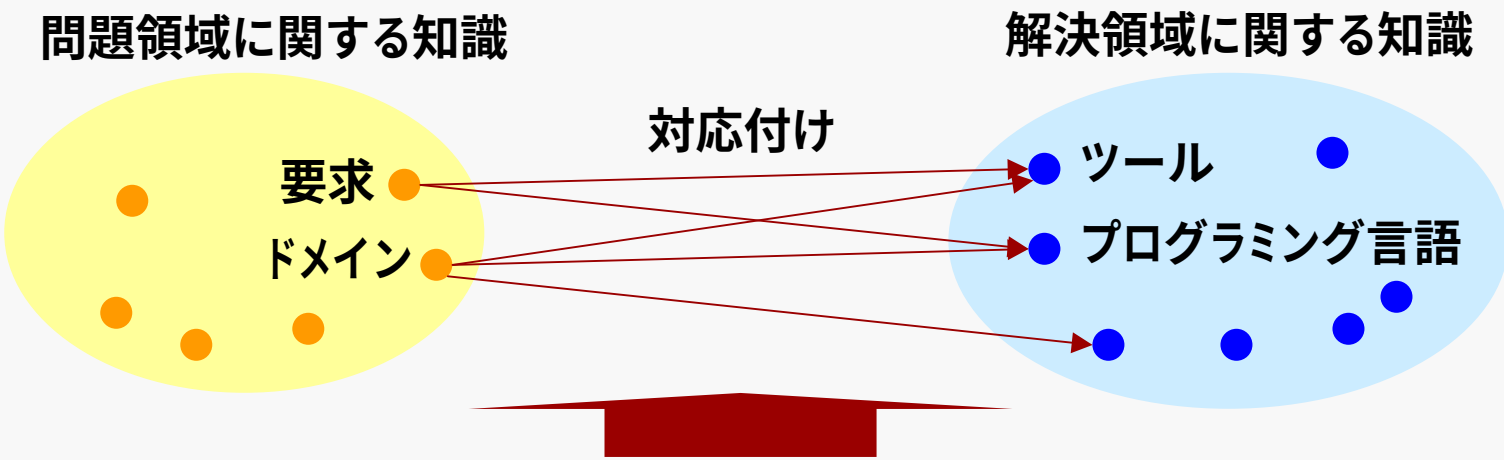
鷲崎 弘宜　久保 淳人

早稲田大学理工学部

http://www.fuka.info.waseda.ac.jp/

ソフトウェア開発と知識

- ソフトウェア開発:
  - 様々な知識を用いて、問題領域に関する知識から解決領域に関する知識への対応付けをおこなう知的作業
- 知識: 経験に基づいて理解し認識した事柄



開発方法論・プロセスに関する知識

知識そのものに関する知識

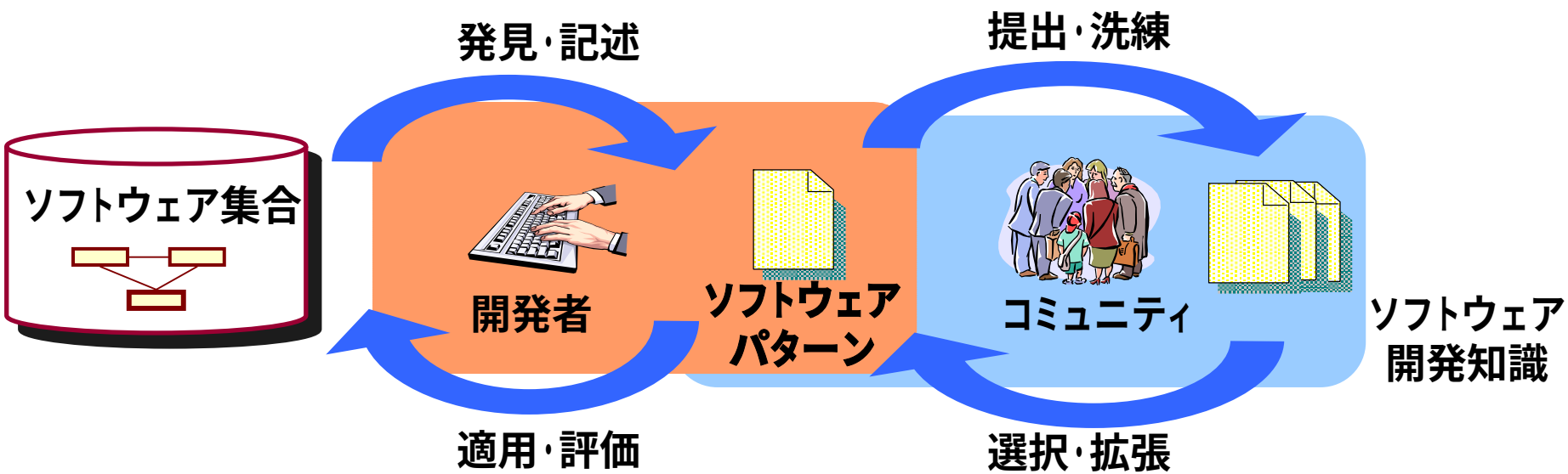
組織形成に関する知識

対応付けに関する知識

[藤野02] 藤野: ソフトウェア開発とパターンランゲージ, ソフトウェアシンポジウム (2002)

ソフトウェアパターン活動と知識

- ソフトウェアパターンとは？
  - ソフトウェア開発経験における特定の文脈上で、繰り返し発生する一定の出来事から得られる知識に名前を付けたもの
  - 利用実績上の基準: Rule of Three
  - 表現上の基準: 3つ組 (文脈・フォース体系・ソフトウェア構成)
- ソフトウェアパターン活動とは？
  - ソフトウェアパターンをソフトウェア開発に活用する活動
  - 抽出: 発見 → 記述 → 提出 → 洗練
  - 利用: 選択 → 拡張 → 適用 → 評価

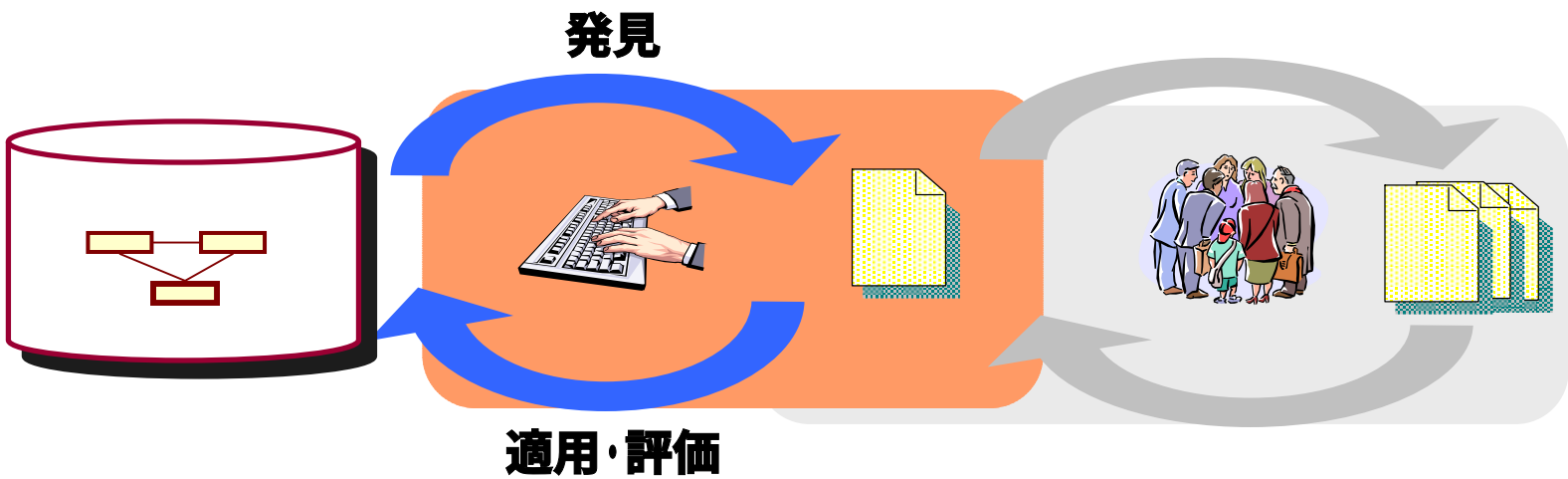


ソフトウェアパターンと道具

- パターン活動を円滑に進めるには？
  - ソフトウェアパターン (ランゲージ) の枠組みを理解する
  - 道具を用意し、理解し、利用する
- どのような道具が使えるだろうか？
  - 汎用的な道具: モデリングツール、コミュニケーションツール (Wiki, ML) 、構成管理ツール、エディタ、紙とペン
  - 特別な道具: ソフトウェアパターンツール
- ソフトウェアパターンツールとは
  - パターン活動を構成する何らかの作業を支援する特別なツール
  - ソフトウェアパターンの活用知識をソフトウェア化したもの
  - 必ずしもソフトウェアパターンの理解には役立たない

ソフトウェアパターンツールの現状

- 扱えるパターン
  - 特定の組織・開発に依存しない汎用的なもの
  - 最終成果物としてのプログラムに「近い」もの
  - 例: イディオム、GoFデザインパターン、POSAデザインパターン
- 支援可能な活動:
  - 適用: 適用支援ツール ＊ どのように適用するか
  - 発見: 検出ツール＊既にどのようなパターンが適用されているか
  - 評価: 検出ツール＊パターンを正しく適用できたか



利用可能な適用支援ツール

- **パラメータ入力に基づいた、既存のモデル・プログラムへのパターン適用の半自動化**

A **デザインパターンの雛型となるモデル・プログラムソースコードを新規生成**

- UML Studio, ModelMaker など

B **クラス・メソッドに対して役割を決めてデザインパターンの雛型となるモデル・コードを新規生成**

- **統合環境系**: TogetherControlCenter, Rational Rose, Konessa, Describe Developer
- **モデリングツール系**: PatternWeaver
- **プラグイン系**: Pattern Support for Eclipse, CodePro Studio

C **既存モデル・コード中の特定個所を、リファクタリングによりデザインパターンの雛型へ変換**

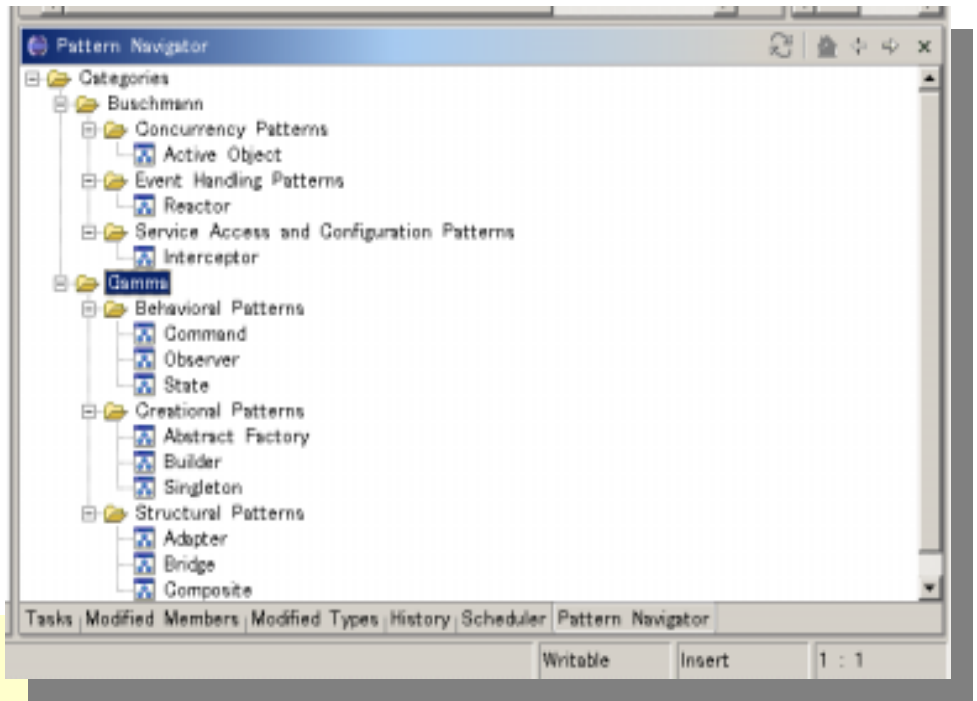
- PTIDEJ

D **特殊言語を用いたデザインパターン実装記述/適用**

- AspectJ, OpenJava, MixJuice など

Cの例: Pattern Support for Eclipse

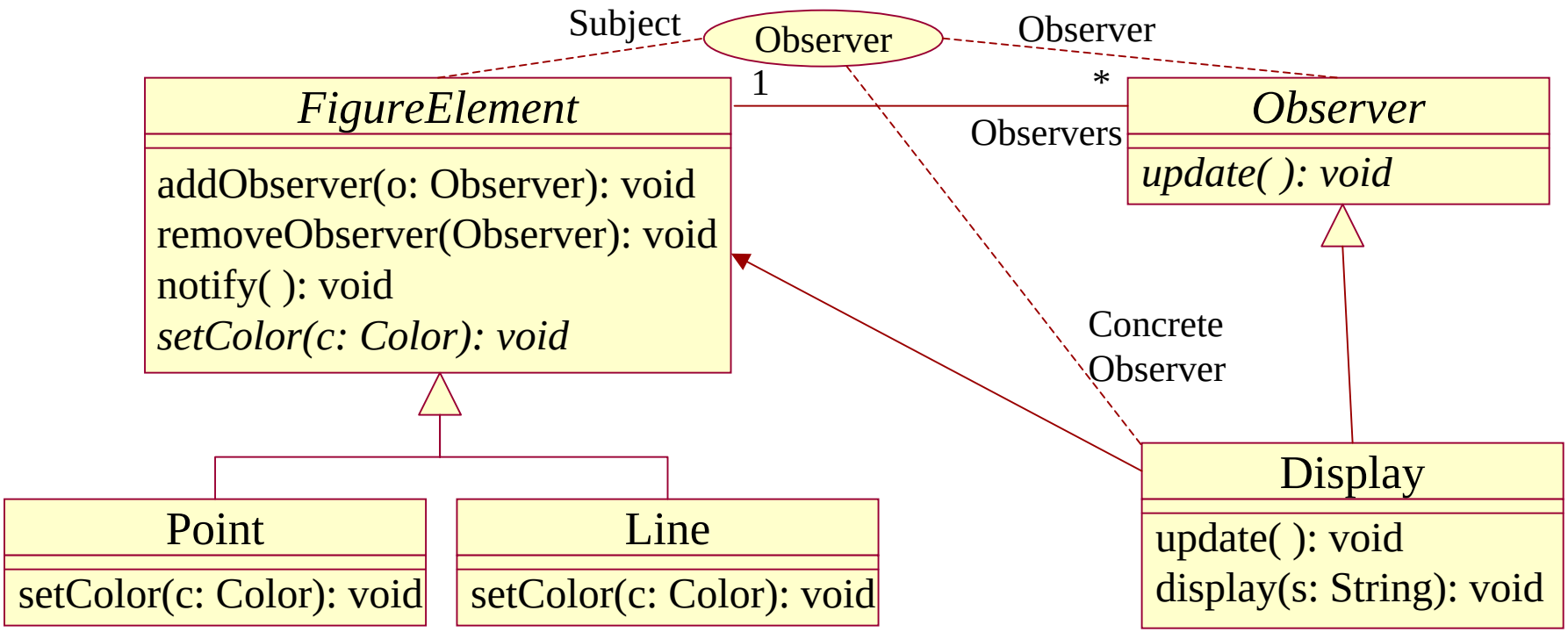
- **デザインパターン適用を自動化する**Eclipse Plugin
  - **扱うパターン**: GoF/POSA**デザインパターン**
  - **既存のソースコードにデザインパターンを適用可能**
  - **メソッド実装も生成可能**



- **対応環境: Eclipseの動作環境**
- **開発者: Irmgard Mayr-Kern, Andreas Wöckl (Johannes Kepler Universitat Linz)**
- **入手先: http://www.swe.uni-linz.ac.at/people/sametinger/research/pse.html**
- **ライセンス: フリー**

Dの例: AspectJ

- AspectJ: Javaのためのアスペクト指向言語処理系
- デザインパターン適用支援
  - デザインパターンの関心 ⟺ 適用前のプログラムの関心
  - デザインパターンを独立して記述し、コンパイル時に適用
- 例: 図形描画プログラムへのObserverパターン適用

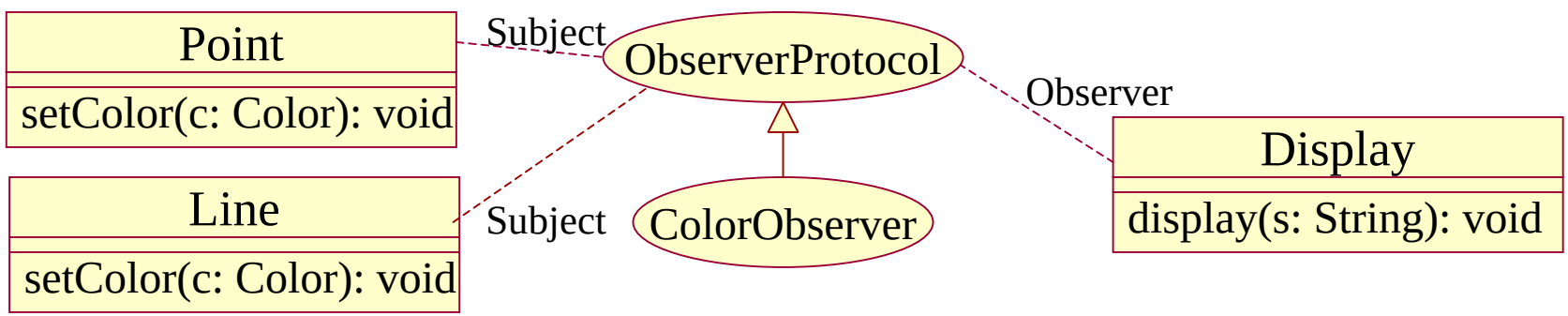


Dの例: AspectJ（つづき）

- Observerパターンの雛型を抽象アスペクトとして定義
- 対象プログラム用の具象アスペクトの定義

```
public abstract aspect ObserverProtocol {
  protected interface Subject { }
  protected interface Observer { }

  abstract protected pointcut subjectChange(
    Subject s);
  abstract protected void updateObserver(
    Subject s, Observer o);

  after(Subject s): subjectChange(s) {
    Itretor iter = …;
    while(iter.hasNext()) updateObserver(…);
  }  … }
```

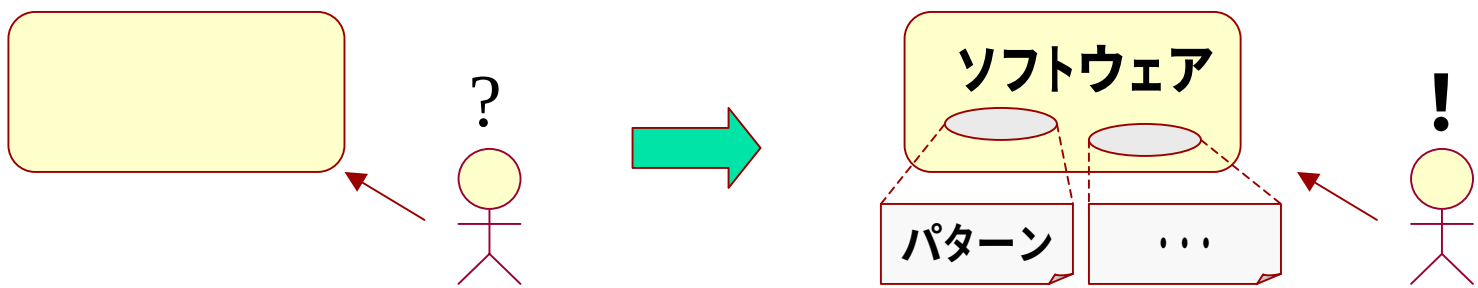
```
public aspect ColorObserver extends ObserverProtocol{
  declare parents: Point implements Subject;
  declare parents: Line implements Subject;
  declare parents: Display implements Observer;
  protected pointcut subjectChange(
    Subject s):
    (call (void Point.setColor(Color))
   || call (void Line.setColor(Color)) && target(s);

  protected void updateObserver(Subject s,
    Observer o) {
    ((Display) o).display("Color change.");
  }  }
```



利用可能な検出ツール: PTIDEJ

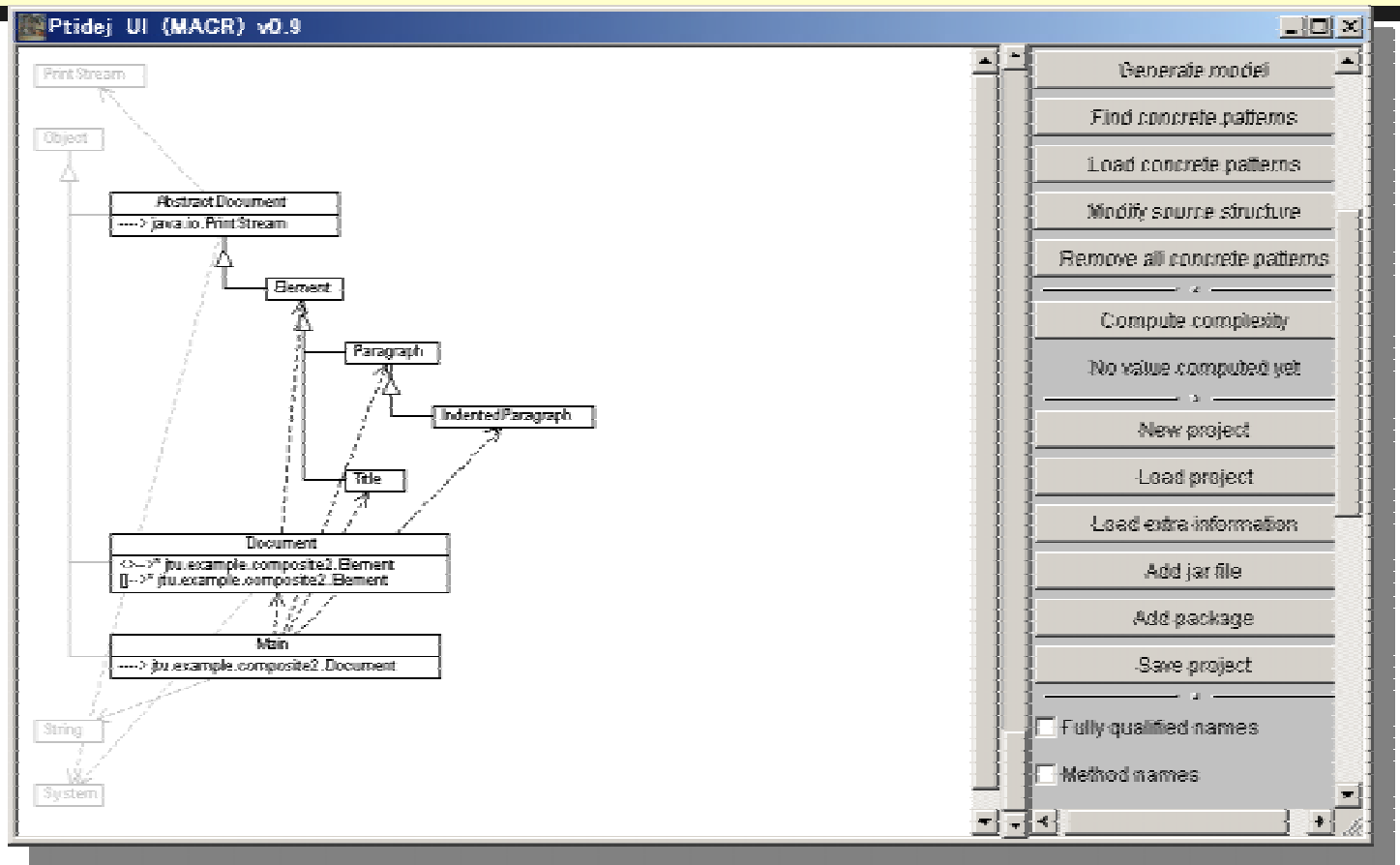
- 既存のソフトウェアについて、既知のパターンが適用されているかどうかを検出する
- PTIDEJ: Javaクラスファイルからデザインパターンを検出可能な変更支援ツール
  - 扱うパターン: GoFデザインパターン
  - デザインパターンの検出、部分的に合致するコードの自動リファクタリング → 定義済みの正しいコードへ変換
  - デザインパターンのモデル定義

```
anInterface = new Interface("Observer");
abMethod = new Method("Update");
anInterface.addElement(abMethod);
ContainerAggregation anAssoc = new
      ContainerAggregation("observers", anInterface, 2);
```



利用可能な検出ツール:PTIDEJ (つづき)

- 対応環境: Java2 SDK
- 開発者: Yann-Gael Gueheneuc (University of Montreal)
- 入手先: http://www.yann-gael.gueheneuc.net/Work/Research/Ptidej/
- ライセンス: フリー

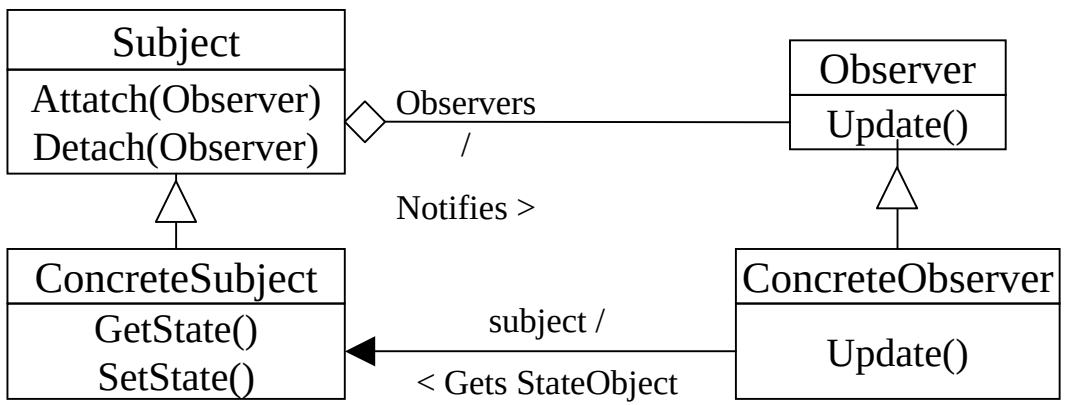


ツールデモ: シナリオ

- 時刻表示プログラムの開発
  - Java言語
  - GUI: アナログとデジタルの両方
- 状況:
  - GUIについて可変性が求まれる
  - 将来、GUIが変更・追加される可能性がある
- そこで、時刻管理部分とGUI部分を分離した設計
  → Observer デザインパターンの適用

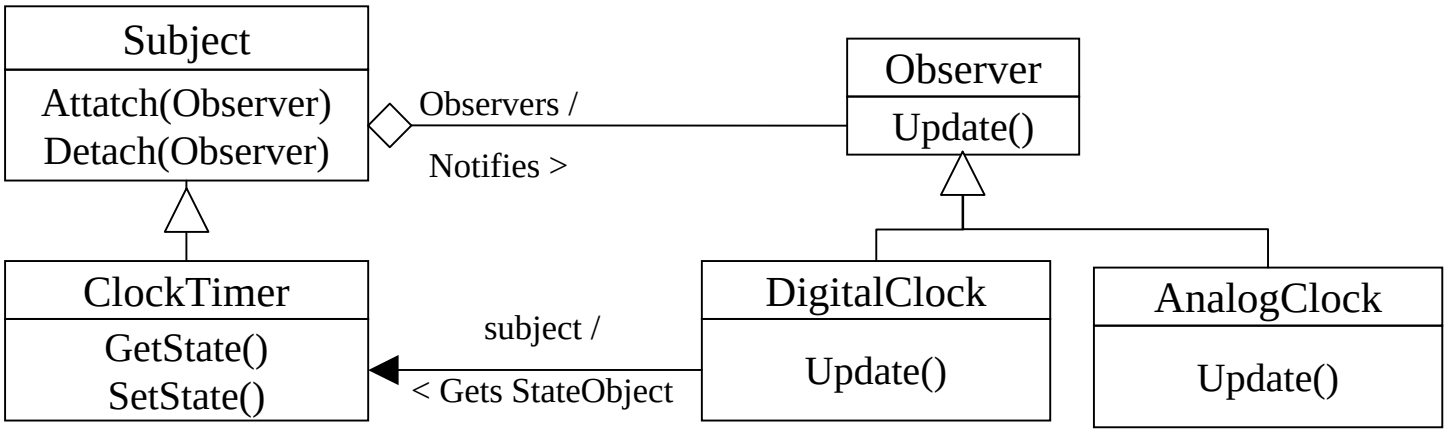
ツールデモ: Observerパターン

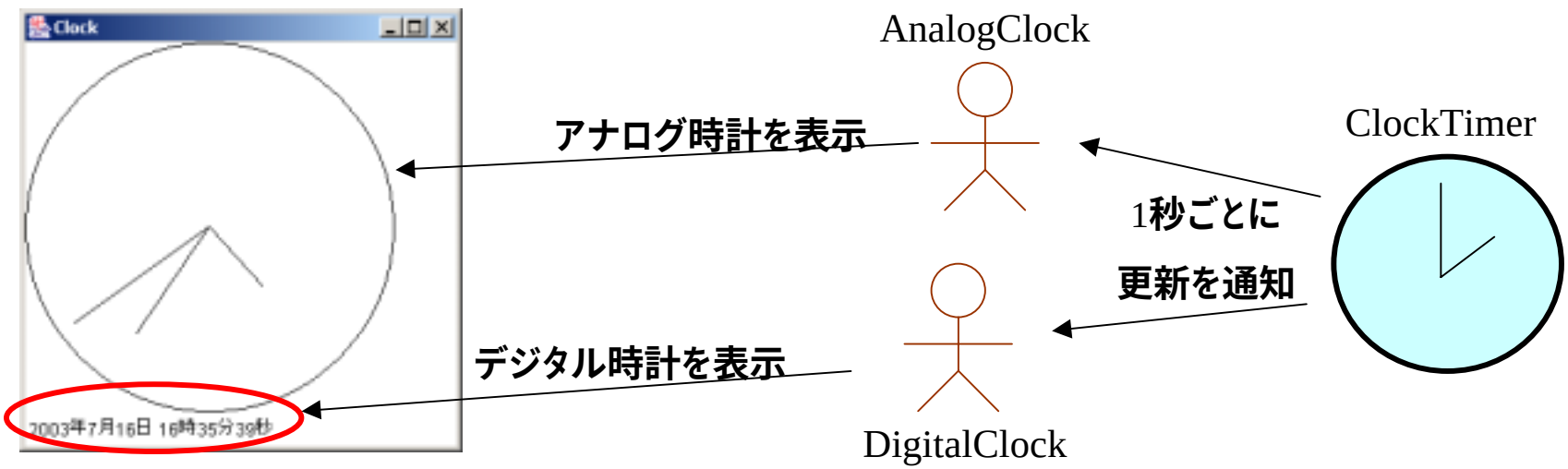
- **目的**:
  - 一対多の依存関係を定義し、オブジェクトの変化を依存する他のオブジェクトに通知・状態を自動更新する
- **適用可能性**:
  - 対多の依存関係があり、依存元・先を個別に再利用したい場合
  - オブジェクトの変化を依存する他のオブジェクトに通知し、依存先の状態を更新したい場合
  - 依存元と先の結合を疎にしたい場合
- **構造**:



- **出典**: GoFデザインパターン
  - オブジェクト指向フレームワークの設計経験上、繰り返し出現した目的・動機・解決等に名前をつけた 23 個のパターンカタログ

ツールデモ: デザインパターンの適用





パターンツールとの付き合い方

- ツール至上主義は誤り
  - 「こういうツールを用意すれば、ソフトウェアパターンの活用が上手くいく」というのは間違い
  - ツールを用いてソフトウェアパターンの活用作業を再利用することはできても、ソフトウェアパターンを理解することにはならない
- 身近な（慣れ親しんだ）道具
  - パターンツールというと何か特別なものをイメージするが、実際には、我々の身の回りにある利用可能な様々な道具をパターン活動に応用できるだろう
- 正しい付き合い方の三か条
  - その1: ソフトウェアパターンを正しく理解する
  - その2: 直面する問題と状況を正しく把握する
  - その3: 用いる道具の特性と役割を正しく理解する