トレーサビリティとインパクト分析

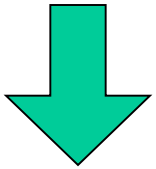
2022年12月12日

海谷 治彦

背景: ソフトウェア成果物

プロの開発では多様な成果物が作成される.

- 要求仕様書
- 設計仕様書
- ソースコード
- テストケース

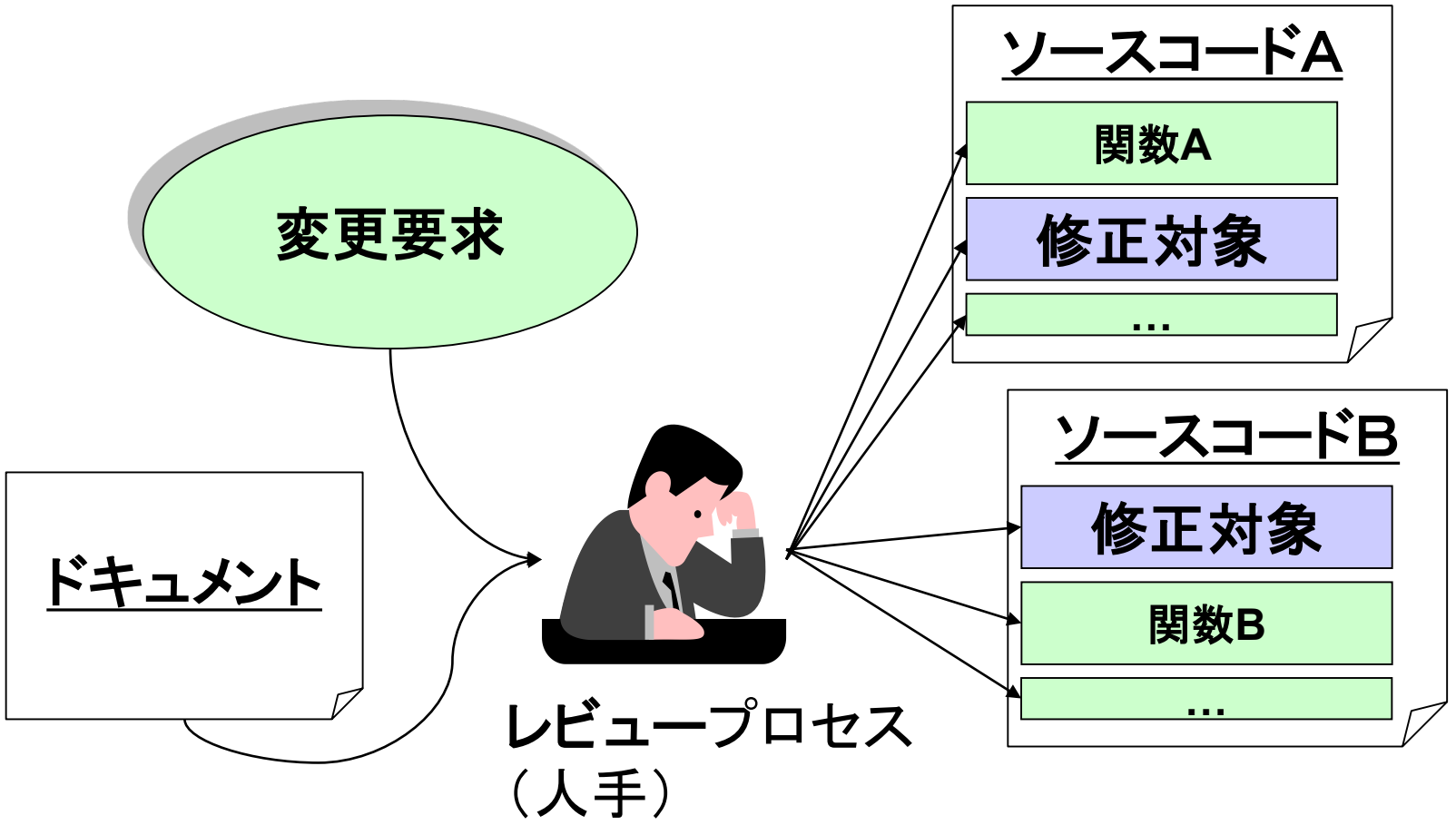


成果物内，成果物間の相互の関係(トレース)を知ることが重要.

トレースする理由

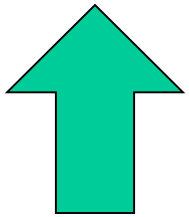
- 開発中に，どの機能が実現済か確認する.
  - 要求仕様書 ⇒ コード
- バグの原因を探す.
  - 仕様書 ⇒ コード
- 既存ソフトに新機能追加や性能アップのため，改造・改良する際に，どこを書き直せば良いか探す. (インパクト分析)
  - 仕様書 ⇒ コード
- 仕様が見当たらない実システムの仕様を明確にする. (リバースエンジニアリング)
  - コード ⇒ 仕様

人手が主の変更要求への対応



例: 過去の演習2,3の解答例

- 要求項目は14個
- ユースケースは6個
- (概念)クラスは5 or 6個
- ソースは6 or 7個
- 予想される要求変更が7個



- それぞれ単純に対応付いていないため，トレースをするための技術が必要.

トレーサビリティの分類

- 水平方向のトレーサビリティ
  - 関数から関数，クラスからクラス等.
  - 変更や依存性.
  - 呼び出し関係，共有関数の存在
- 垂直方向のトレーサビリティ
  - 要求項目から設計要素
  - 設計要素からコード片(関数やクラス)
  - 情報検索的なアプローチ，もしくはマニュアルでトレースを記録

トレーサビリティ作成戦略

- 手作業で対応を記録
  - 正確であるが手間はかかる.
  - 特定のツール等を使うのであれば，現実的に適用可能.
- 自動的に対応を記録
  - 成果物内の文書の内容の類似性や，作成時刻の近接性で対応を予測.
  - 予測なので，正確でない部分もある.

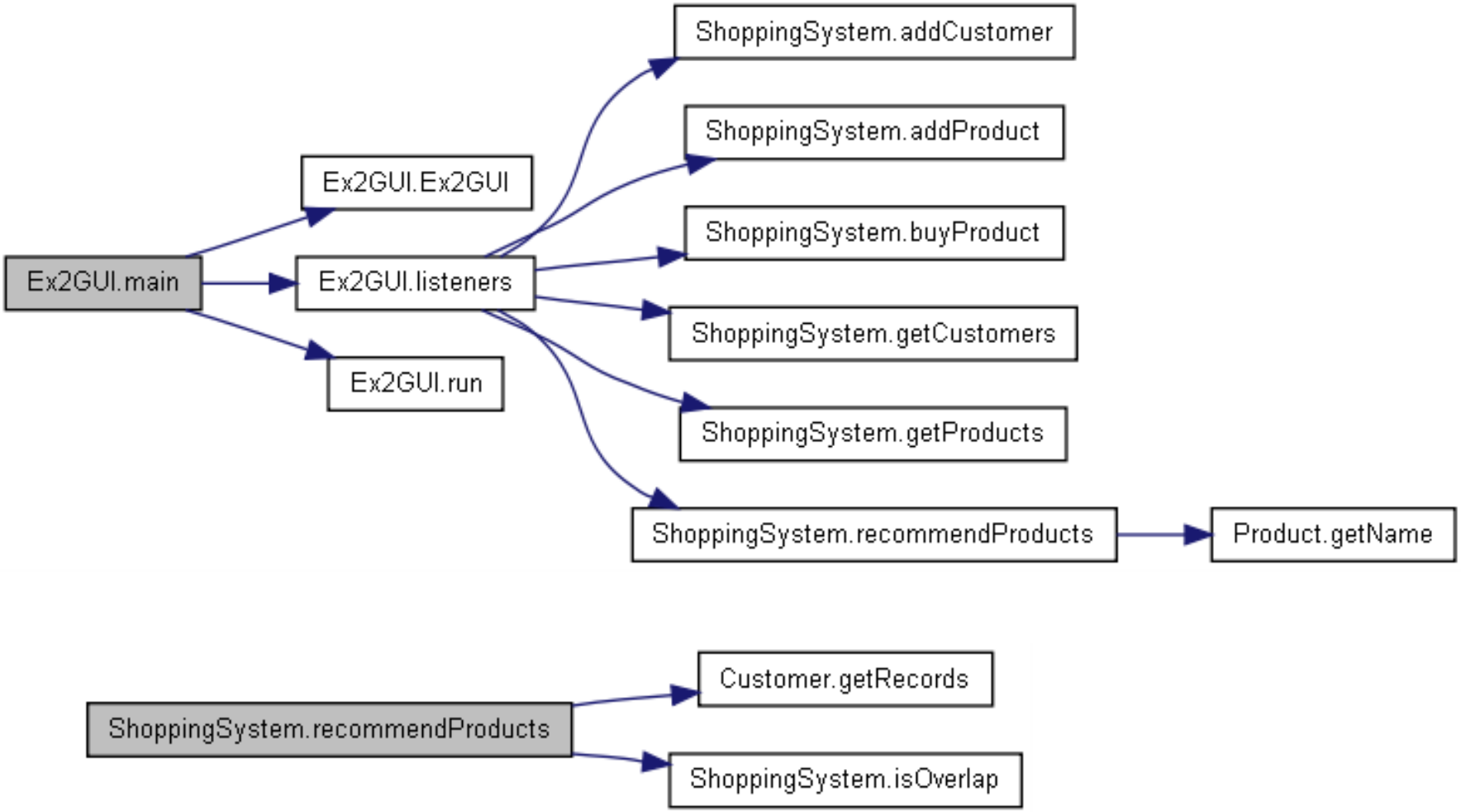
予測技術の分類

- 静的アプローチ
  - プログラムや文書の記述内容をもとにトレースをとる.
  - コールグラフ，データフロー解析.
  - 文書の場合は，章構成や段落構成，文内の係り受け等.
- 動的アプローチ
  - プログラムを動かしてみてトレースをとる.
  - プロファイリング.
  - デバッガ等を利用した解析.
  - 文書は当然，動かないので原則，適用できないが，ユースケース記述やシーケンス図を描いてみるのは，ある意味，動的なアプローチ.

コールグラフ

- 水平型トレースのための静的技術.
- 要は関数やメソッドの呼び出し関係を明確にする.
- 呼び出し関係がある関数間には，変更の波及がある場合が多いため.
- コールグラフを作成するツールもある.
  - doxygen
  - 描画には graphviz を用いる

例

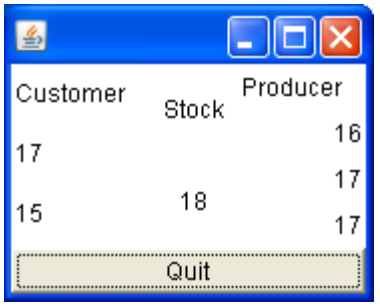

ShoppingSystem.addCustomer
ShoppingSystem.addProduct
Ex2GUI.Ex2GUI
ShoppingSystem.buyProduct
Ex2GUI.main
Ex2GUI.listeners
ShoppingSystem.getCustomers
Ex2GUI.run
ShoppingSystem.getProducts
ShoppingSystem.recommendProducts
Product.getName
Customer.getRecords
ShoppingSystem.recommendProducts
ShoppingSystem.isOverlap

プロファイラ

- プログラムの実行時の情報を収集するツール.
- 本来はパフォーマンスの向上のための情報等を集めるために使われる.
- ちょっとしたプログラムでも長大なデータになる.

- コレも水平方向のトレースに役立つ.

例 hprofの出力例



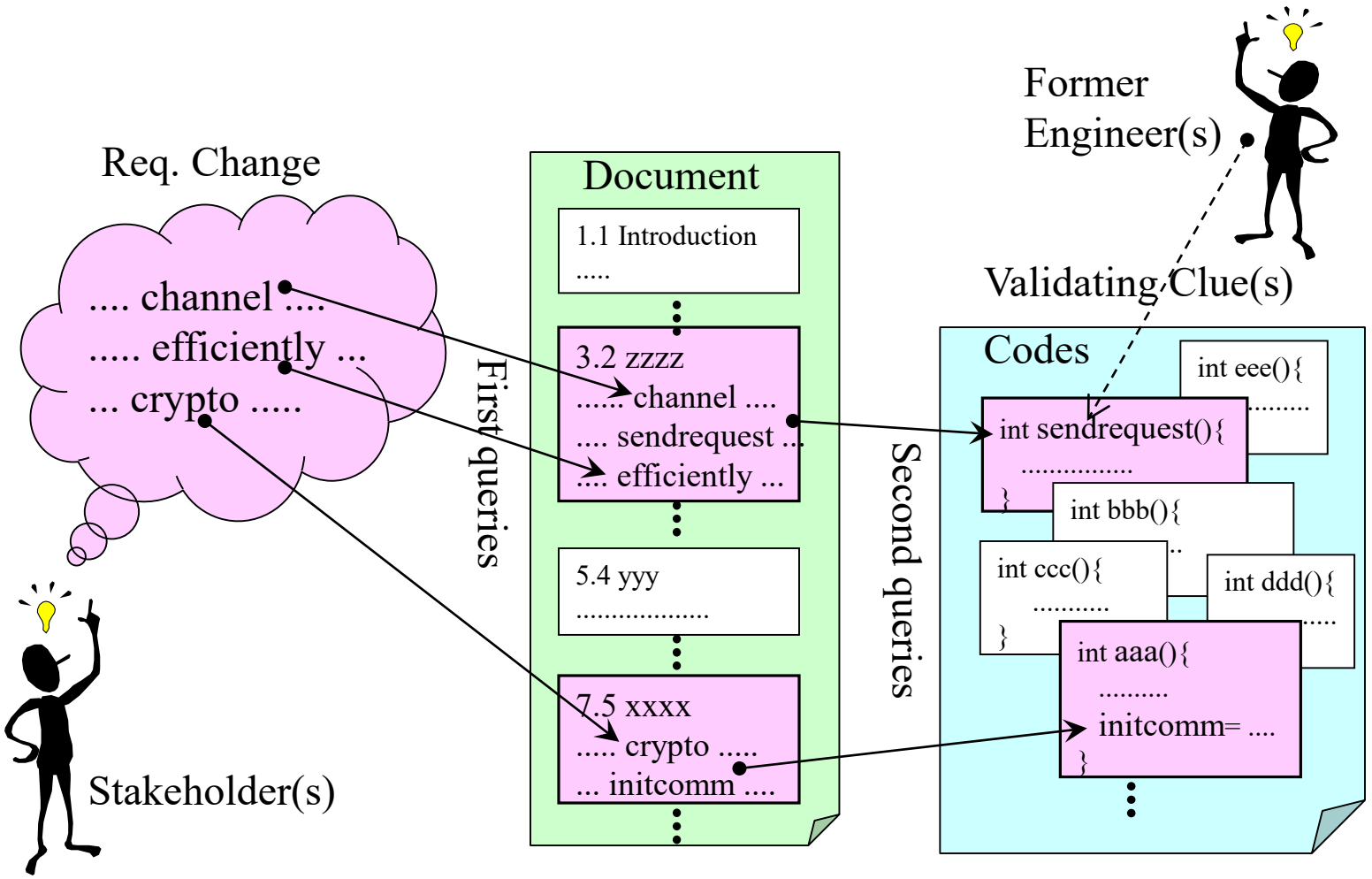
C:¥> java -agentlib:hprof -jar Run.jar
スレッドのところでやった生産者・消費者問題
数秒動かしただけで，7万行の情報を出力.

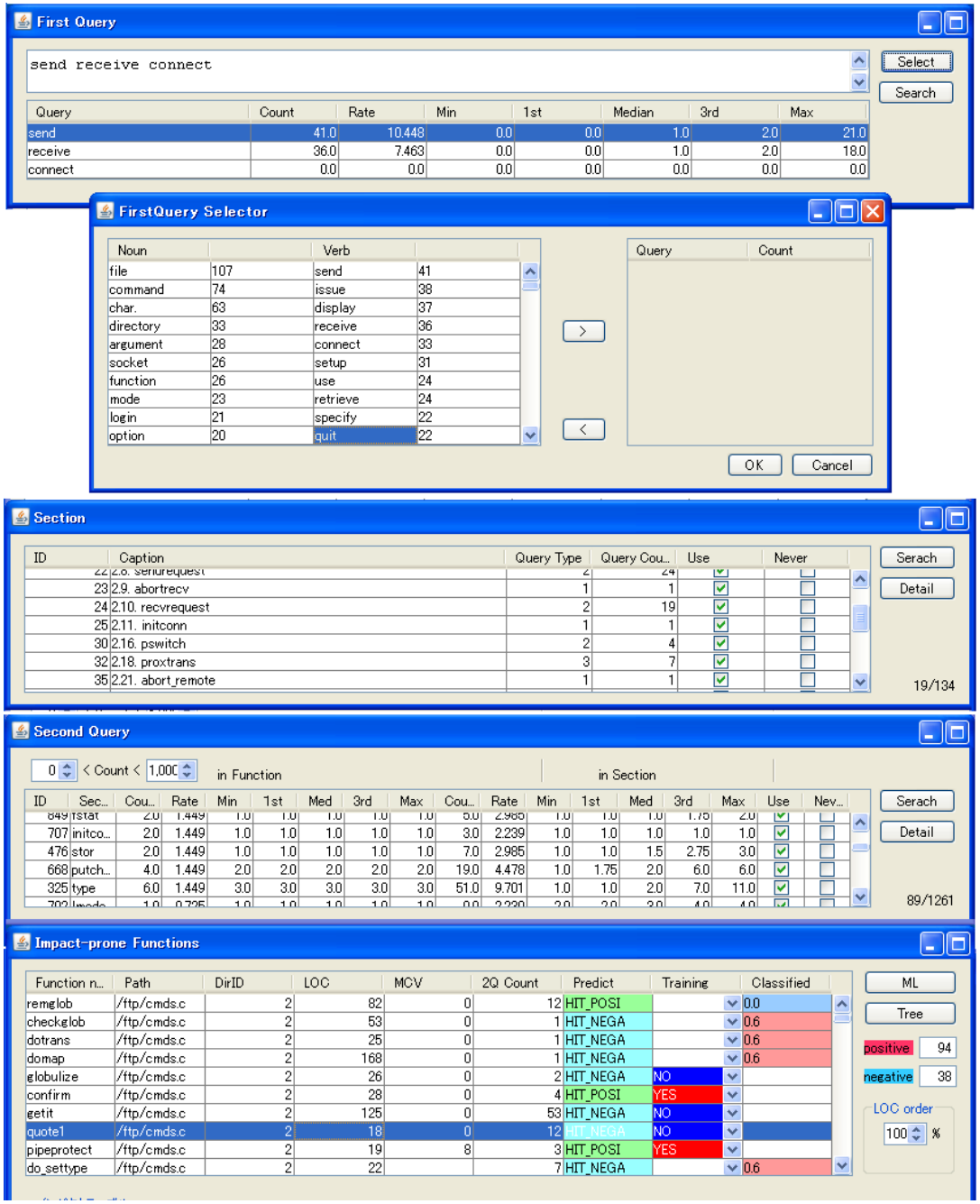
```
TRACE 302139:
        java.awt.Component.<init>(<Unknown Source>:Unknown line)
        java.awt.Label.<init>(<Unknown Source>:Unknown line)
        TabacoSale.init(TabacoSale.java:29)
        TabacoSale.main(TabacoSale.java:66)
TRACE 302140:
        IntLabel.<init>(IntLabel.java:5)
        TabacoSale.init(TabacoSale.java:32)
        TabacoSale.main(TabacoSale.java:66)
TRACE 302141:
        java.lang.Thread.<init>(<Unknown Source>:Unknown line)
        LabelUpdater.<init>(LabelUpdater.java:8)
        Customer.<init>(Customer.java:3)
        TabacoSale.init(TabacoSale.java:32)
```

垂直の例と技術

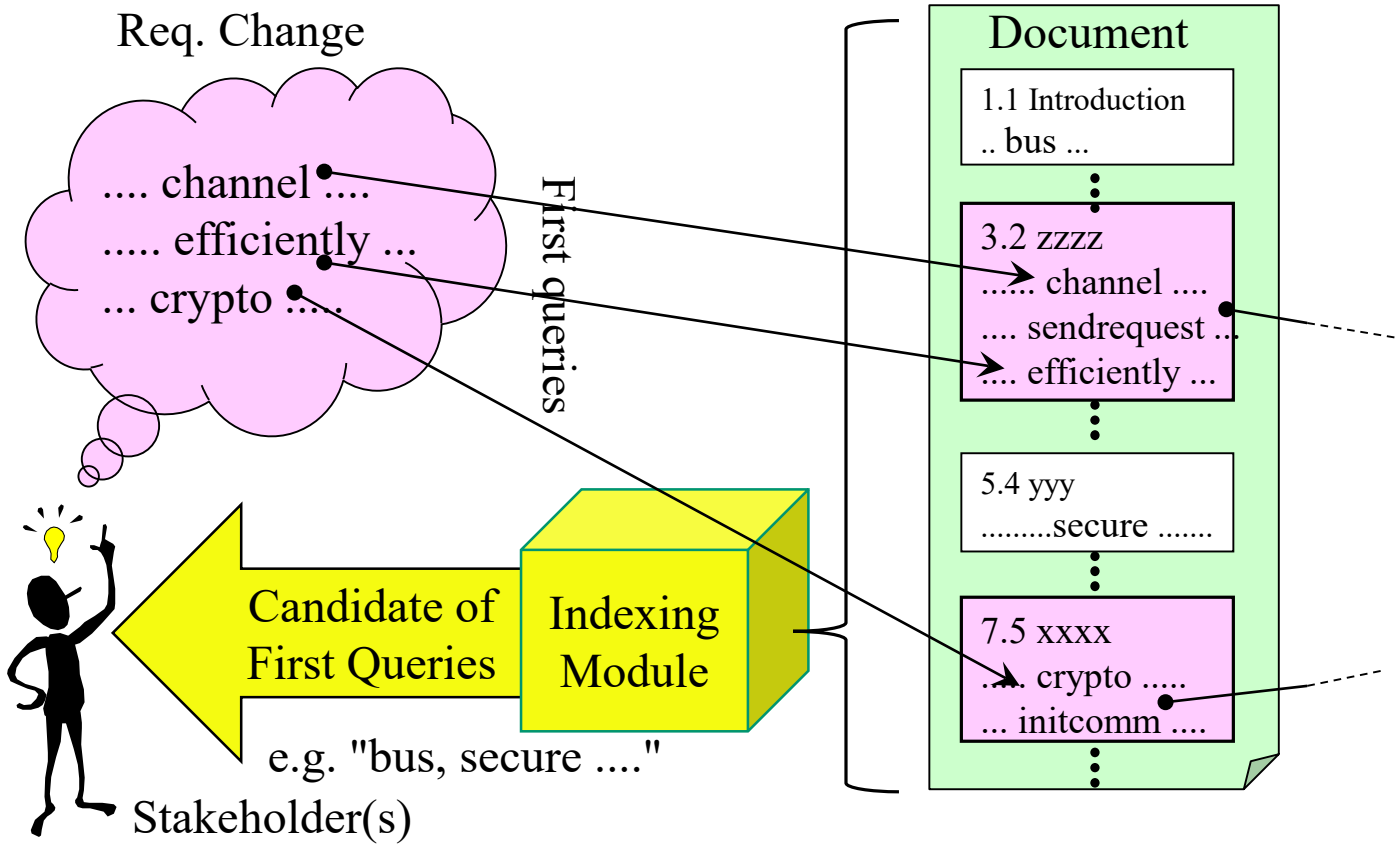
- まだ気軽に使えるツールはあまり無い.
  - ソースだけでなく，文書解析をしないといけないため.
- 情報検索
  - 文書の出現語句に基づき，トレースをとる.
- 同義語の解決
  - 「言い回し」の違いの吸収.
  - 辞書(オントロジー)の利用

要求変更箇所を探す手法の例



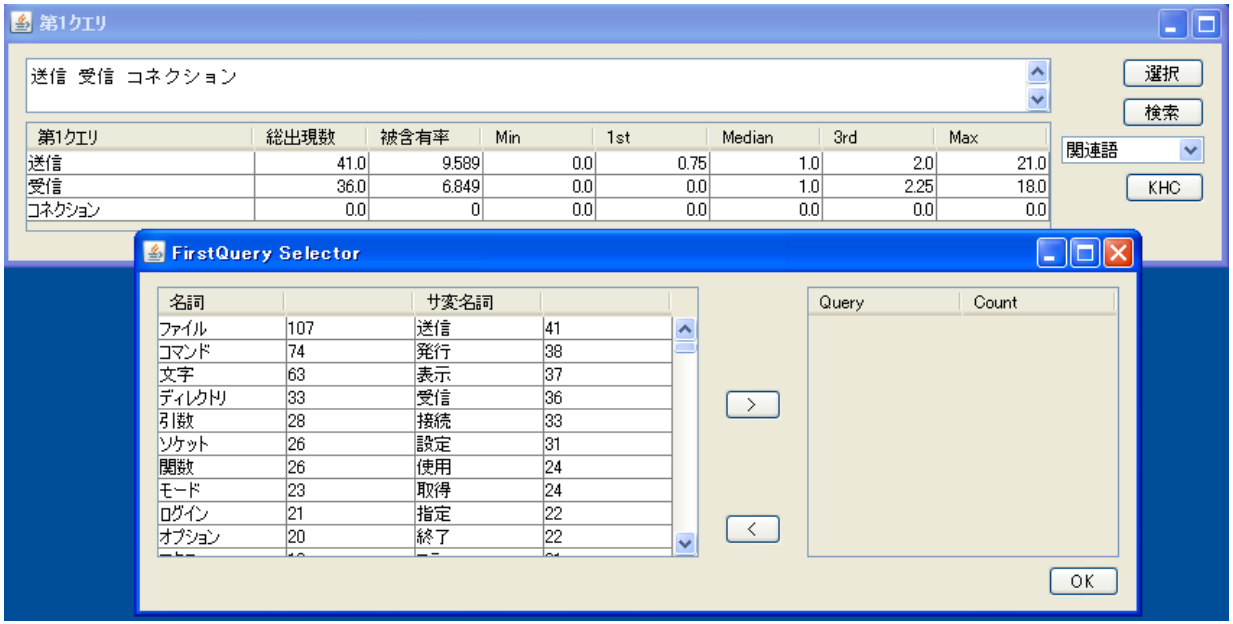

First Query
send receive connect
Select
Search
Query Count Rate Min 1st Median 3rd Max
send 41.0 10.448 0.0 0.0 1.0 2.0 21.0
receive 36.0 7.463 0.0 0.0 1.0 2.0 18.0
connect 0.0 0.0 0.0 0.0 0.0 0.0 0.0
FirstQuery Selector
Noun Verb
file 107 send 41
command 74 issue 38
char. 63 display 37
directory 33 receive 36
argument 28 connect 33
socket 26 setup 31
function 26 use 24
mode 23 retrieve 24
login 21 specify 22
option 20 quit 22
Query Count
OK
Cancel
Section
ID Caption Query Type Query Cou... Use Never
23 2.9. abortrecv 1 1
24 2.10. recvrequest 2 19
25 2.11. initconn 1 1
30 2.16. pswitch 2 4
32 2.18. proxtrans 3 7
35 2.21. abort_remote 1 1
Serach
Detail
19/134
Second Query
0 < Count < 1,000
in Function
in Section
ID Sec... Cou... Rate Min 1st Med 3rd Max Cou... Rate Min 1st Med 3rd Max Use Nev...
707 initco... 2.0 1.449 1.0 1.0 1.0 1.0 1.0 3.0 2.239 1.0 1.0 1.0 1.0 1.0
476 stor 2.0 1.449 1.0 1.0 1.0 1.0 1.0 7.0 2.985 1.0 1.0 1.5 2.75 3.0
668 putch... 4.0 1.449 2.0 2.0 2.0 2.0 2.0 19.0 4.478 1.0 1.75 2.0 6.0 6.0
325 type 6.0 1.449 3.0 3.0 3.0 3.0 3.0 51.0 9.701 1.0 1.0 2.0 7.0 11.0
Serach
Detail
89/1261
Impact-prone Functions
Function n... Path DirID LOC MCV 2Q Count Predict Training Classified
remglob /ftp/cmds.c 2 82 0 12 HIT_POSI 0.0
checkglob /ftp/cmds.c 2 53 0 1 HIT_NEGA 0.6
dotrans /ftp/cmds.c 2 25 0 1 HIT_NEGA 0.6
domap /ftp/cmds.c 2 168 0 1 HIT_NEGA 0.6
globulize /ftp/cmds.c 2 26 0 2 HIT_NEGA NO
confirm /ftp/cmds.c 2 28 0 4 HIT_POSI YES
getit /ftp/cmds.c 2 125 0 53 HIT_NEGA NO
quote1 /ftp/cmds.c 2 18 0 12 HIT_NEGA NO
pipeprotect /ftp/cmds.c 2 19 8 3 HIT_POSI YES
do_settype /ftp/cmds.c 2 22 7 HIT_NEGA 0.6
ML
Tree
positive 94
negative 38
LOC order
100 %

要求変更を特徴付けるには？
索引付けの利用

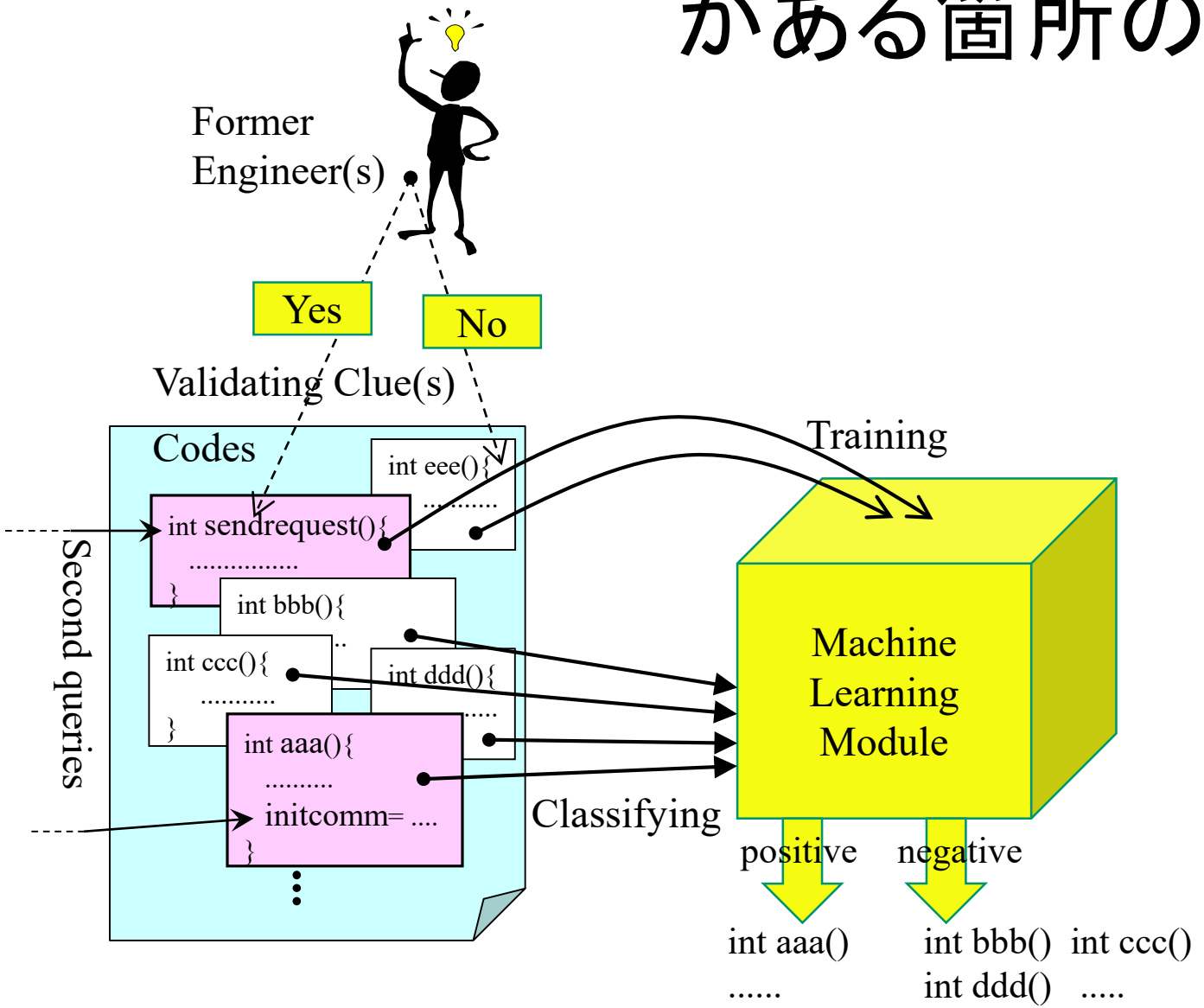


画面例

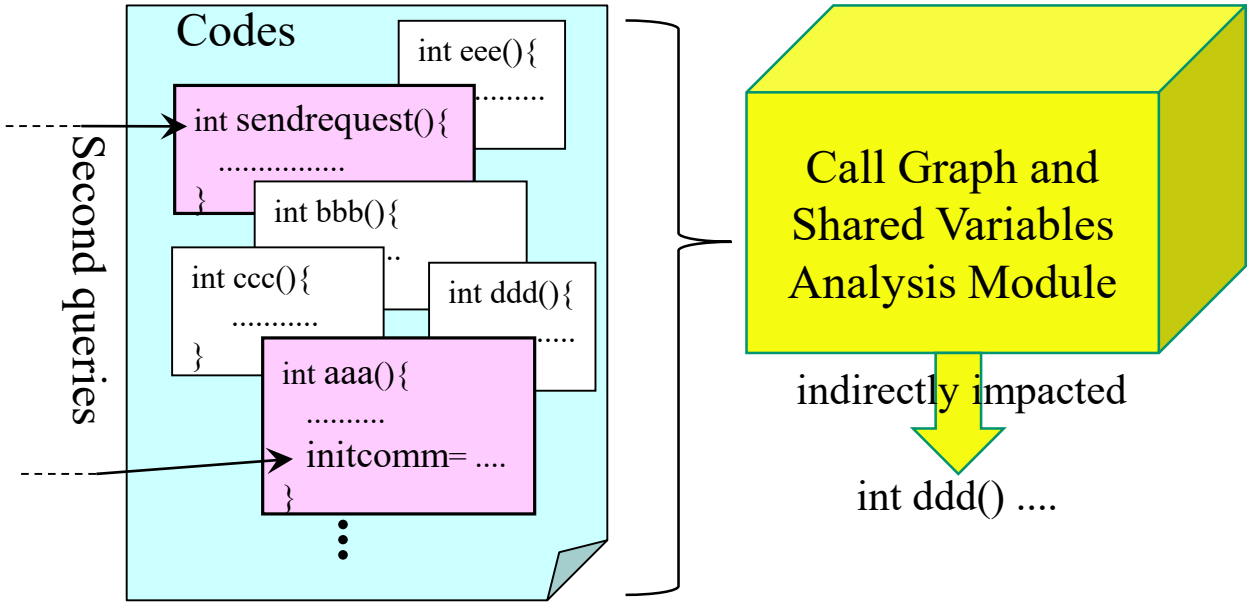
- 変更要求を特徴付けるキーワードを見つけるための支援.
- 候補となる索引単語を提示する方法.



機械学習の利用による変更の可能性がある箇所の予測

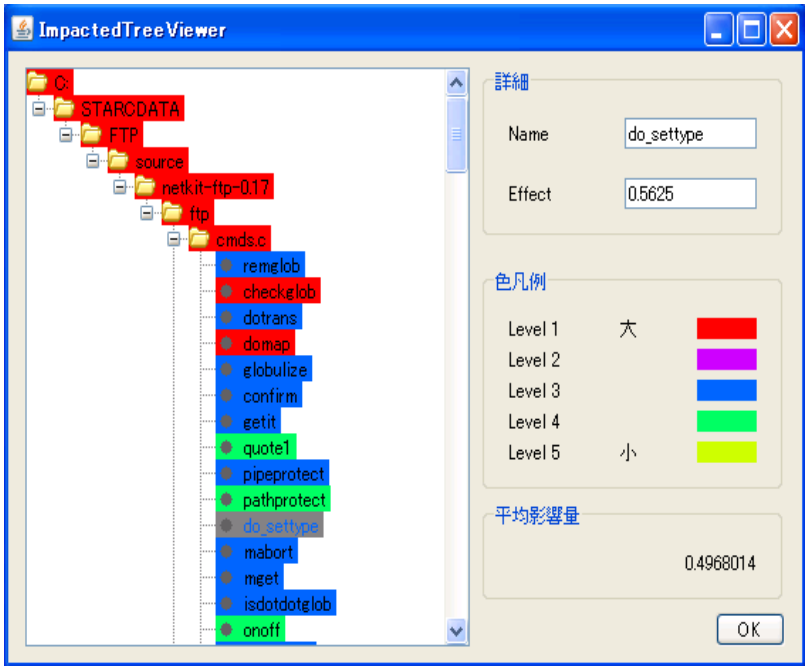


変更波及の可視化
コールグラフと共有変数の利用



画面例

- シード関数を起点
  - 「関数の呼び出し関係」+0.5
  - 「共有変数による関係」+0.75
- 影響値が大きい関数ほど赤に近い色で表現され，小さい関数ほど黄に近い色を示す．



インパクト分析ツール Jripples

- Eclipse上で変更波及の分析(インパクト分析)を行うツール.
- Wayne State Univ. (ミシガンの大学らしい)で開発された.
- 演習で行う小規模なプログラム開発では，あまりピンと来ないが，クラスの数が数百，数千のプログラムでの分析に有効.

- 基本的に対話的にインパクトを探すツール.

インストールと使い方

- 基本的に公式HP参照

http://jripples.sourceforge.net/

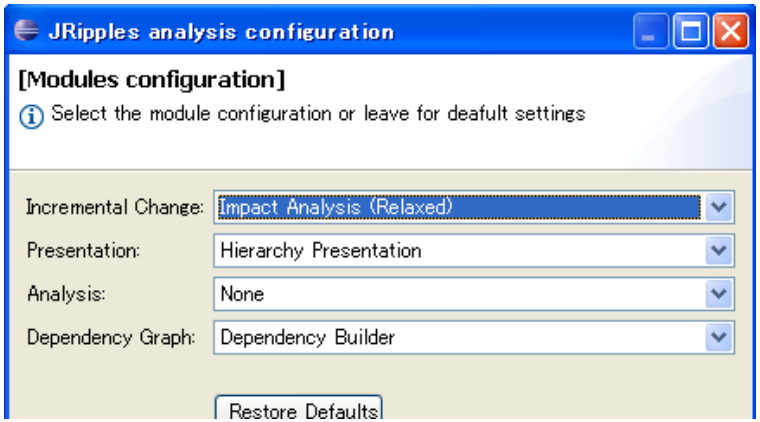
三つの主要概念

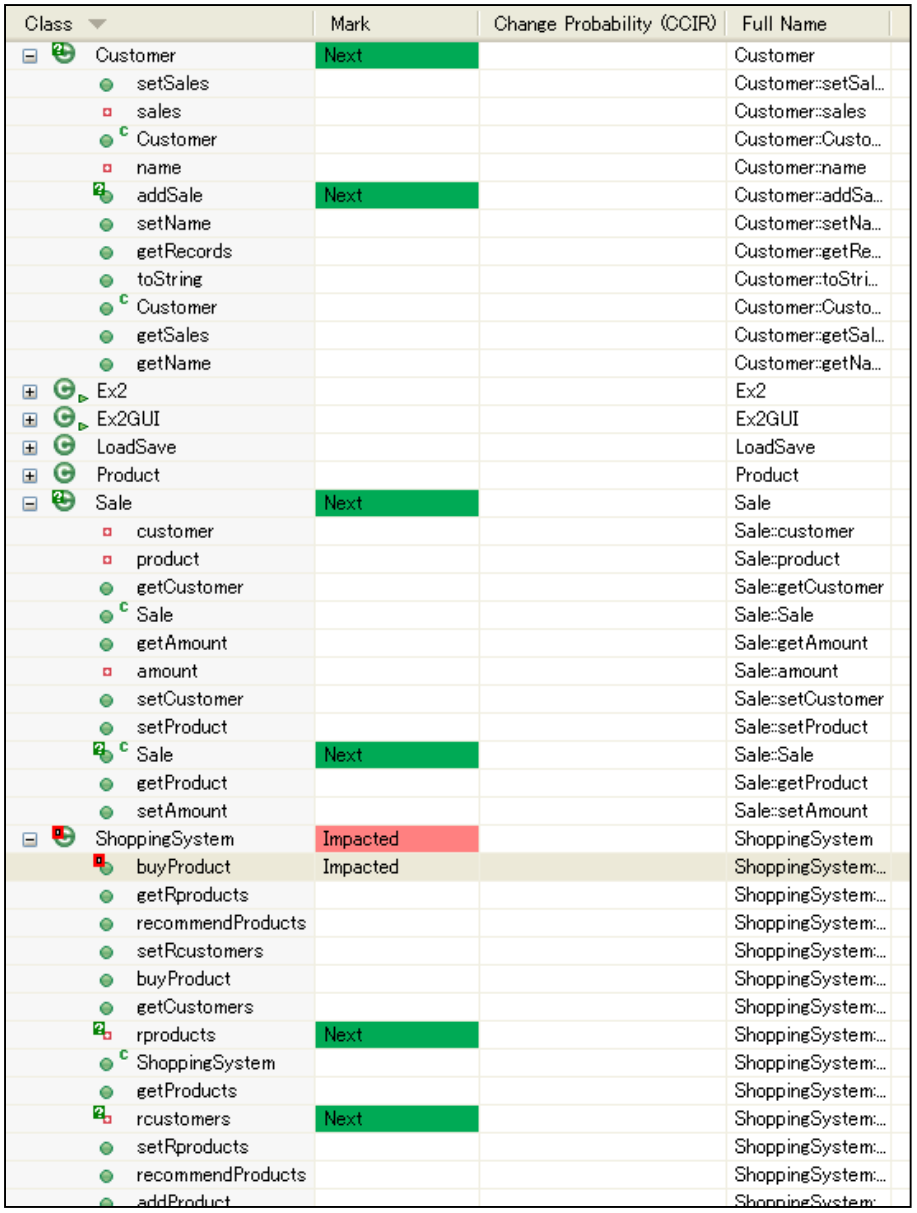
- Concept Location
  - ソース中で変更に最初にインパクトがある部分を見つける作業.
- Impact Analysis
  - 最初のインパクトから波及してインパクトを受ける部分を探す作業.
- Change Propagation
  - 変更波及の実現によって，整合性がとれなくなった部分を探す作業.

TIPS

- Configuration において，Relaxed を指定すること.
  - そうしないと，mainメソッドからしか辿れなくなる.
- Impactedをクラスやメソッドにマーキングして，変更の種を植え付け，その波及箇所を予測するのに使う.

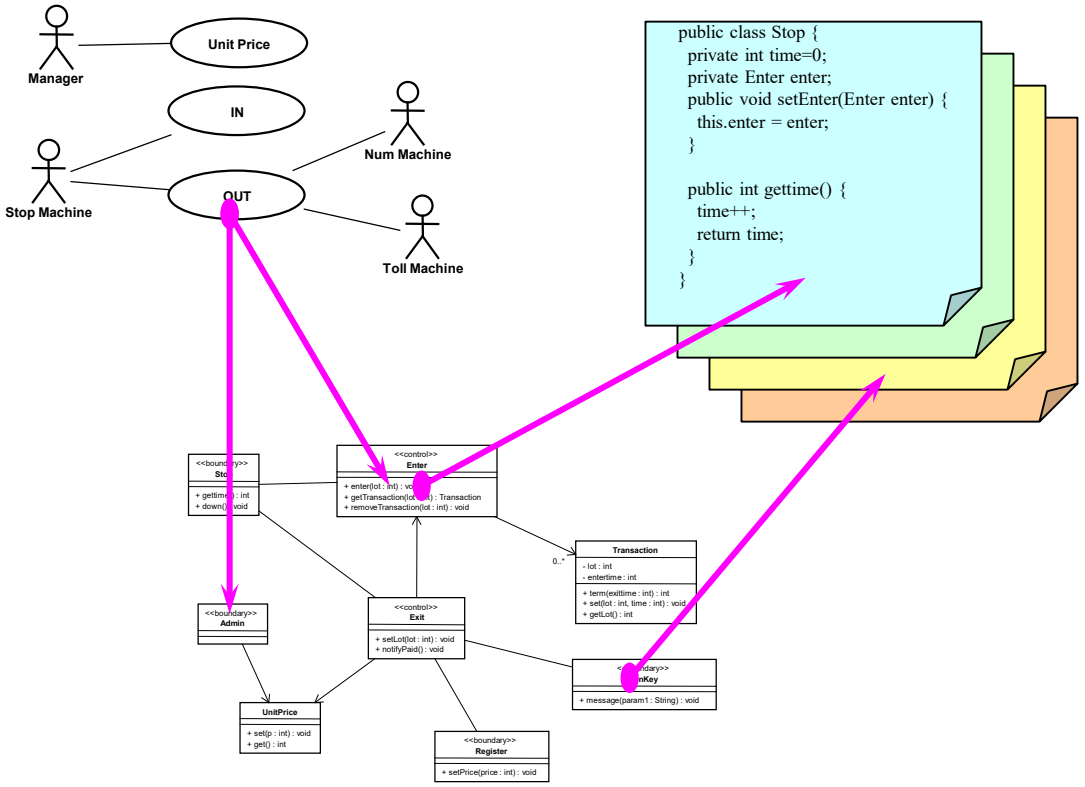
画面例


JRipples analysis configuration
[Modules configuration]
Select the module configuration or leave for deafult settings
Incremental Change:
Impact Analysis (Relaxed)
Presentation:
Hierarchy Presentation
Analysis:
None
Dependency Graph:
Dependency Builder
Restore Defaults


Class
Mark
Change Probability (CCIR)
Full Name
Customer
Next
Customer
setSales
Customer::setSal...
sales
Customer::sales
Customer
Customer::Custo...
name
Customer::name
addSale
Next
Customer::addSa...
setName
Customer::setNa...
getRecords
Customer::getRe...
toString
Customer::toStri...
Customer
Customer::Custo...
getSales
Customer::getSal...
getName
Customer::getNa...
Ex2
Ex2
Ex2GUI
Ex2GUI
LoadSave
LoadSave
Product
Product
Sale
Next
Sale
customer
Sale::customer
product
Sale::product
getCustomer
Sale::getCustomer
Sale
Sale::Sale
getAmount
Sale::getAmount
amount
Sale::amount
setCustomer
Sale::setCustomer
setProduct
Sale::setProduct
Sale
Next
Sale::Sale
getProduct
Sale::getProduct
setAmount
Sale::setAmount
ShoppingSystem
Impacted
ShoppingSystem
buyProduct
Impacted
ShoppingSystem:...
getRproducts
ShoppingSystem:...
recommendProducts
ShoppingSystem:...
setRcustomers
ShoppingSystem:...
buyProduct
ShoppingSystem:...
getCustomers
ShoppingSystem:...
rproducts
Next
ShoppingSystem:...
ShoppingSystem
ShoppingSystem:...
getProducts
ShoppingSystem:...
rcustomers
Next
ShoppingSystem:...
setRproducts
ShoppingSystem:...
recommendProducts
ShoppingSystem:...

手作業でのトレース管理

- 左図のような対応を，ツール等を使い，記録する．
- 開発ツールが特定のものに決まっていれば，わりとうまく管理できる．



手作業トレース管理の例

- UMLのモデルとモデルの間の要素のトレースを管理する.
- 皆さんに使ってもらってる astah に拡張機能をつけ，これを実現している.
- 異なるファイルに属するモデルの要素間でもリンクを作成し，管理することができる.

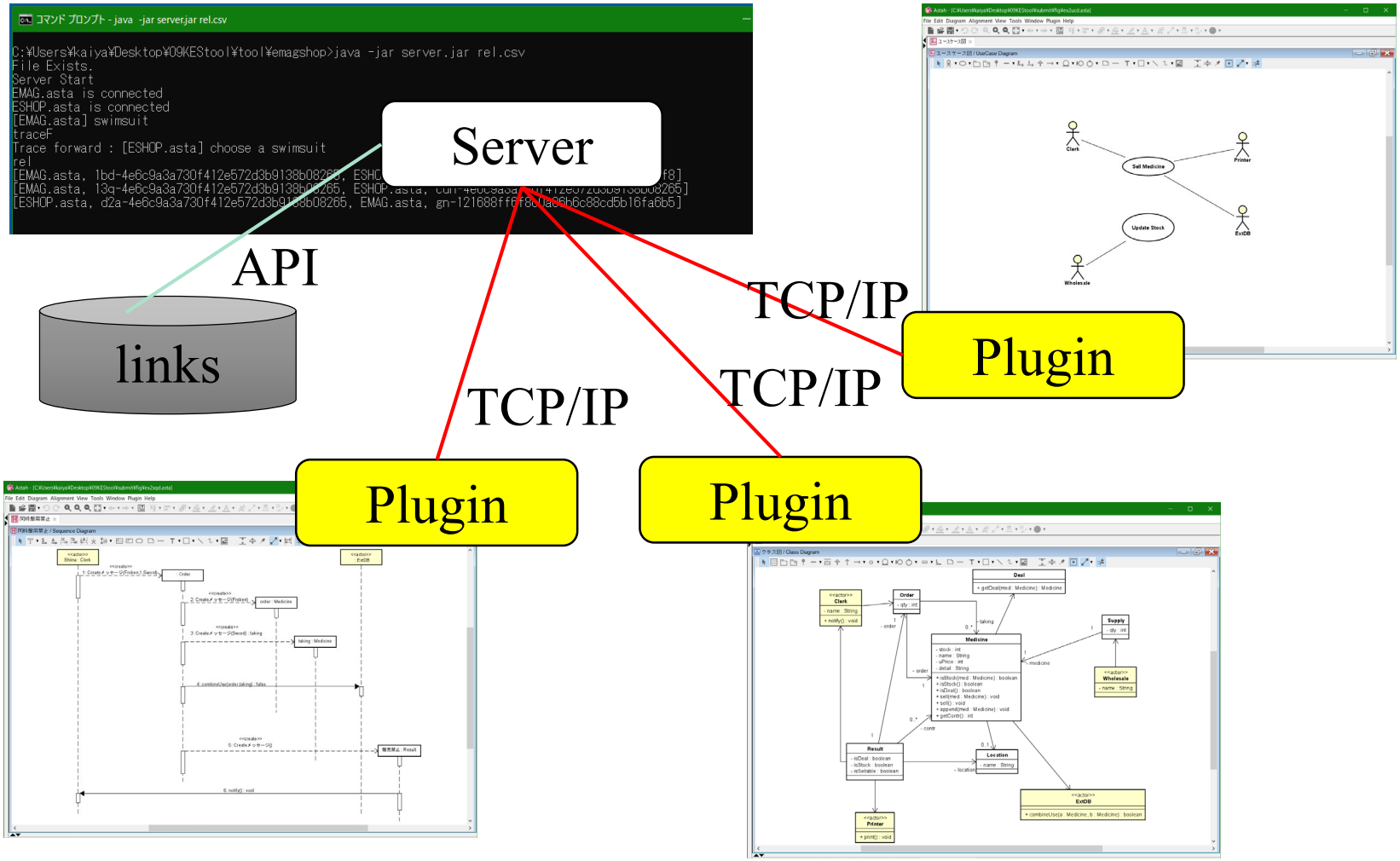
⇒ 複数の異なるシステム間の関係も扱える.

ツールのアーキテクチャ

- クライアント・サーバー方式
- クライアントは拡張された astah
- トレース情報はサーバーで管理
- クライアントである astah は同時に複数プロセスを利用可能.

- 現時点のサーバーはastahのみの対応だが，設計上，他のツールもクライアントして接続可能.
  - 例えば，テキストエディタ等.

アーキテクチャの図


コマンド プロンプト - java -jar server.jar rel.csv
C:¥Users¥kaiya¥Desktop¥09KEStool¥tool¥emagshop>java -jar server.jar rel.csv
File Exists.
Server Start
EMAG.asta is connected
ESHOP.asta is connected
[EMAG.asta] swimsuit
traceF
Trace forward : [ESHOP.asta] choose a swimsuit
rel
Server
API
links
TCP/IP
TCP/IP
TCP/IP
Plugin
Plugin
Plugin

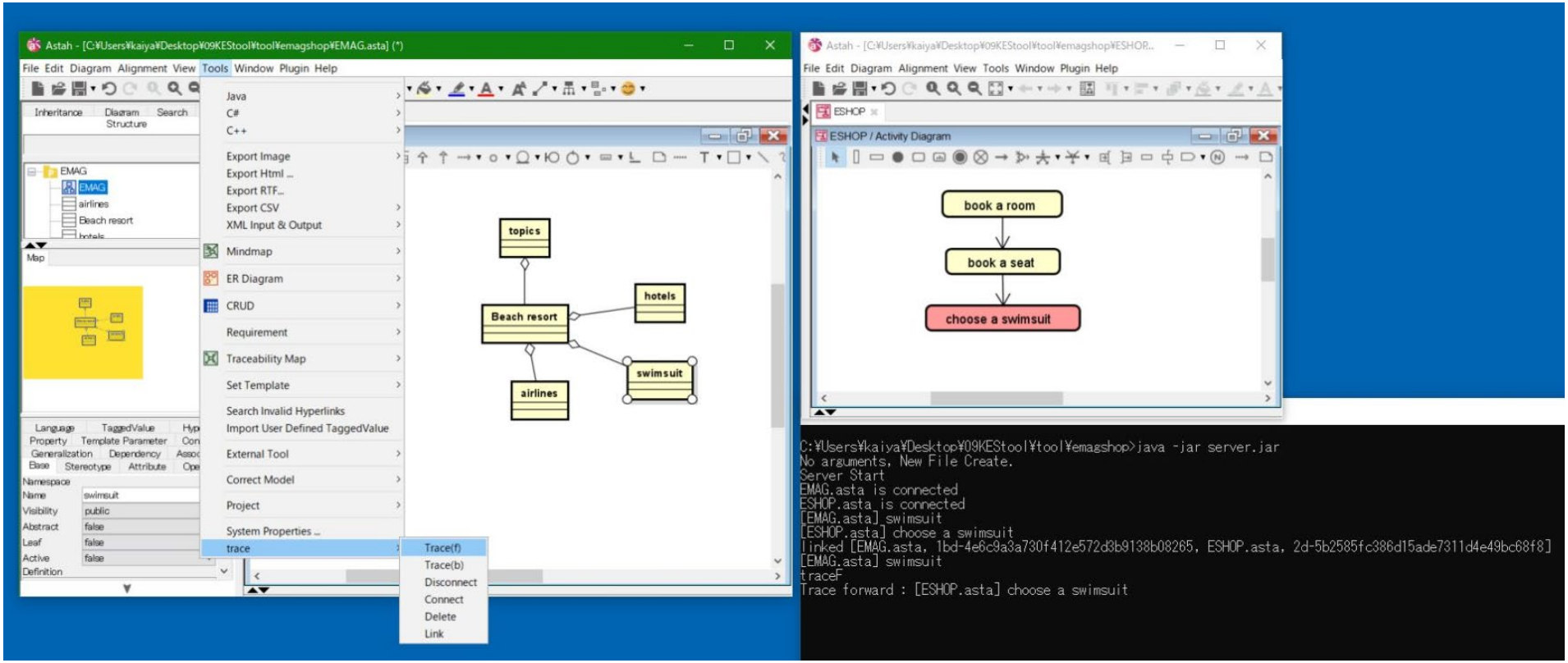
機能一覧

- サーバーは過去に記録したリンクを保持するファイルからリンクを読み出し起動.
  - 新規の場合は空のファイルを作成.
- クライアントの機能

1. サーバーへ接続，切断
2. 選択している図要素をサーバーに通知する.
   - サーバーは，その履歴を2個分，記憶する.
3. サーバーが記憶する図要素間にリンクを張る.
4. リンクを削除.
5. リンクを元から先に辿り，ハイライトする.
6. 同様に先から元に辿り，ハイライトする.

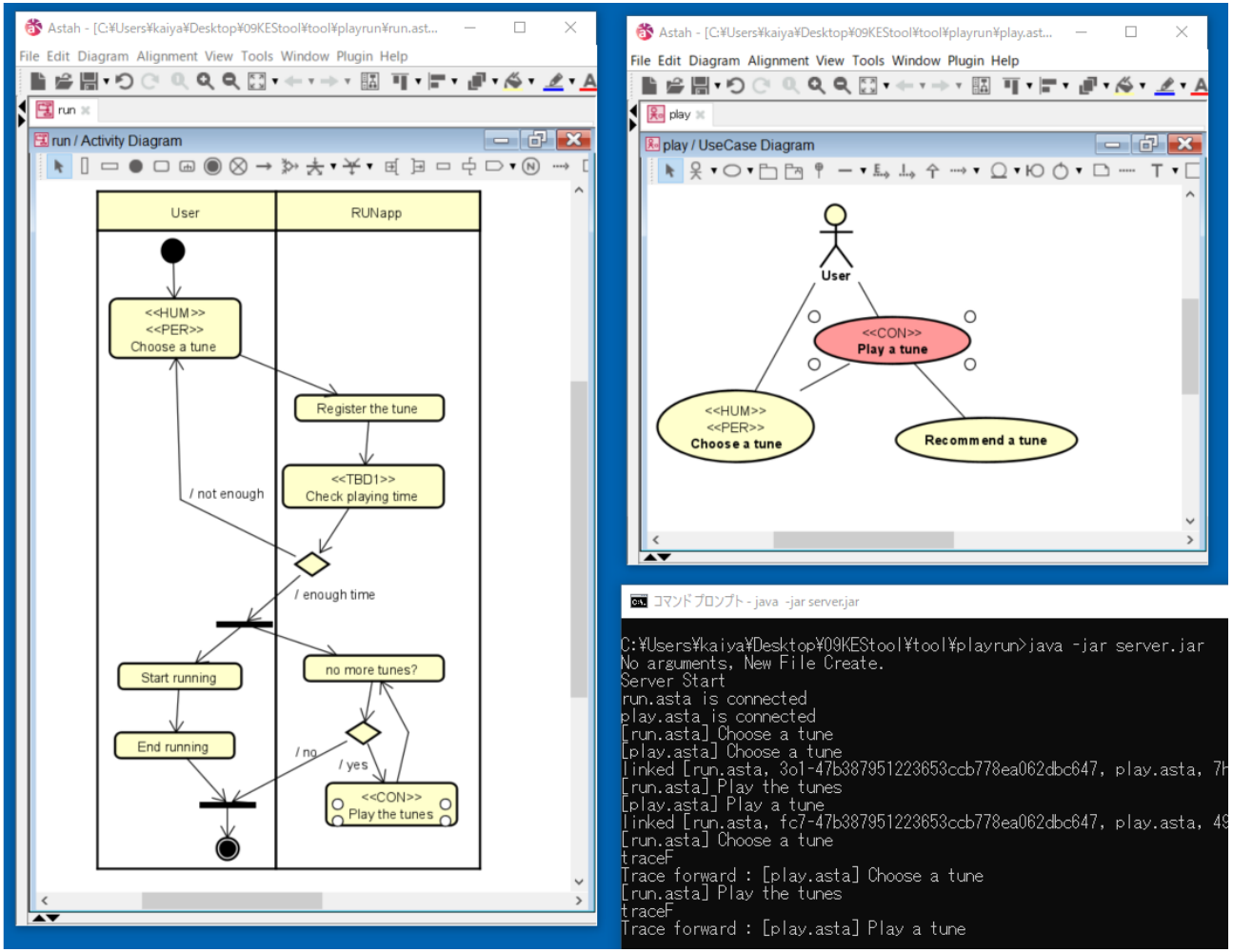
トレースを辿る画面例

- 単一システムの開発例

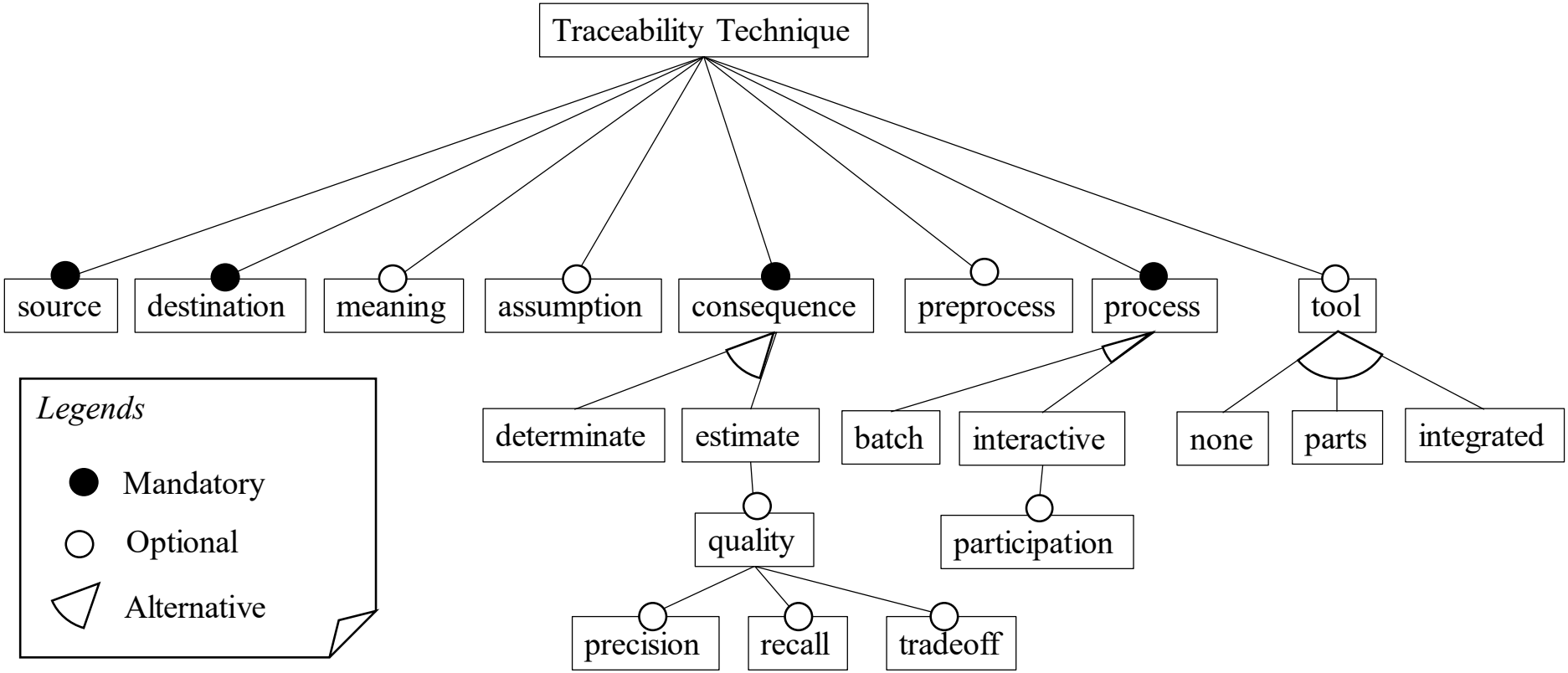


別のトレース例

- 異なるシステムの関連する機能を関係付け管理する.



まとめ トレース技術の分類



Source, Destination

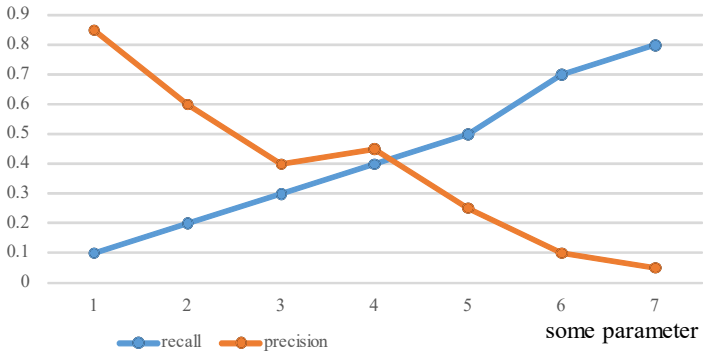
- トレース技術は，ソフトウェア開発成果物の中のある部分から，他の部分への管理を管理するもの．
- 部分の例: 文書の段落，クラス図のクラス，ユースケース図のアクター，テストケース，Javaのクラス，Cの関数 等

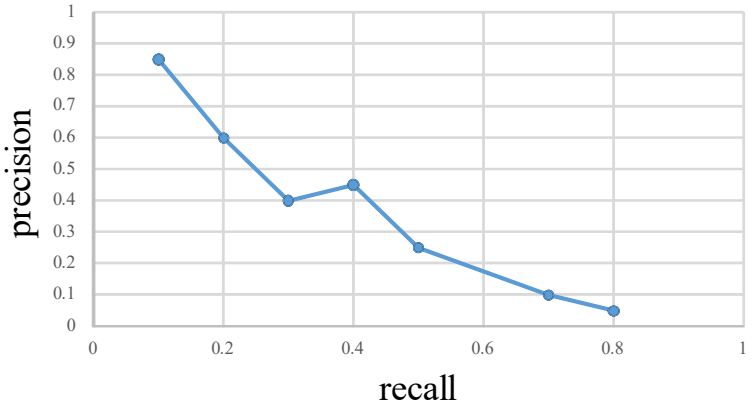
Consequence

- 決定論的に決まる (determinate)
  - 手作業や，ツール上の操作に連動して対応を付ける.
  - MDD等，自動変換の過程で対応をつける.
- 予測する (estimate)
  - 手法は出現単語の類似性等，様々.
  - 予測なので，正確さと漏れの無さのトレードオフがある.
    - 正確さトレースがあると予測されてものが正しいか？
    - 漏れの無さ 予測結果にトレースが全て含まれるか？

トレードオフ

- 正確さ(precision)と漏れの無さ(recall)は，大抵，トレードオフがある．
  - 正確にすれば，漏れが増える．
  - 漏れを減らすと，正確でなくなる．





処理方式

- 一括処理(batch)と対話的処理(interactive)がある.
- 後者の場合，人間が結果を見ながら，パラーメータの調整等を行う場合が多い.

意味，前処理

- 意味
  - トレースの意味は様々
  - 例 仕様と実装の関係，詳細化の関係，変更波及の関係，依存関係.
  - どの関係を扱うか明示してない手法も多い.
- 前処理
  - 特に文書の場合，前処理が必須.
  - 分割，語彙の統一，一般語の除去等.

ツール化

- 手法によっては，アイディアのみでツールが無いものもある.
- 独立したツールの場合もある.
- IDEに組み込むように作られたツールもある.
  - Eclipse, VSC のプラグインとして.

手法やツールの制限

- 色々な想定があることも多い，例は以下.

- 英語のみに対応.
- 関数やクラスの厳格な命名規則を適用
- 開発履歴やログが必須
- 開発対象分野の概念辞書（ドメインモデル）が存在する.
- MDDで開発している.

以上