Impact and Feature Analysis (SB5-MAI)

Jan Corfixen Sørensen

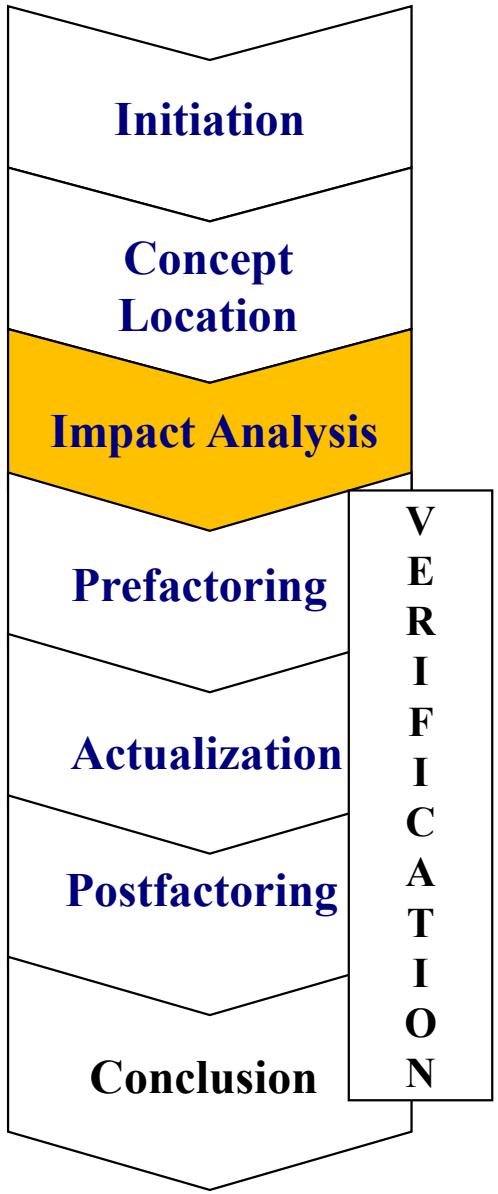
University of Southern Denmark

October 2, 2017

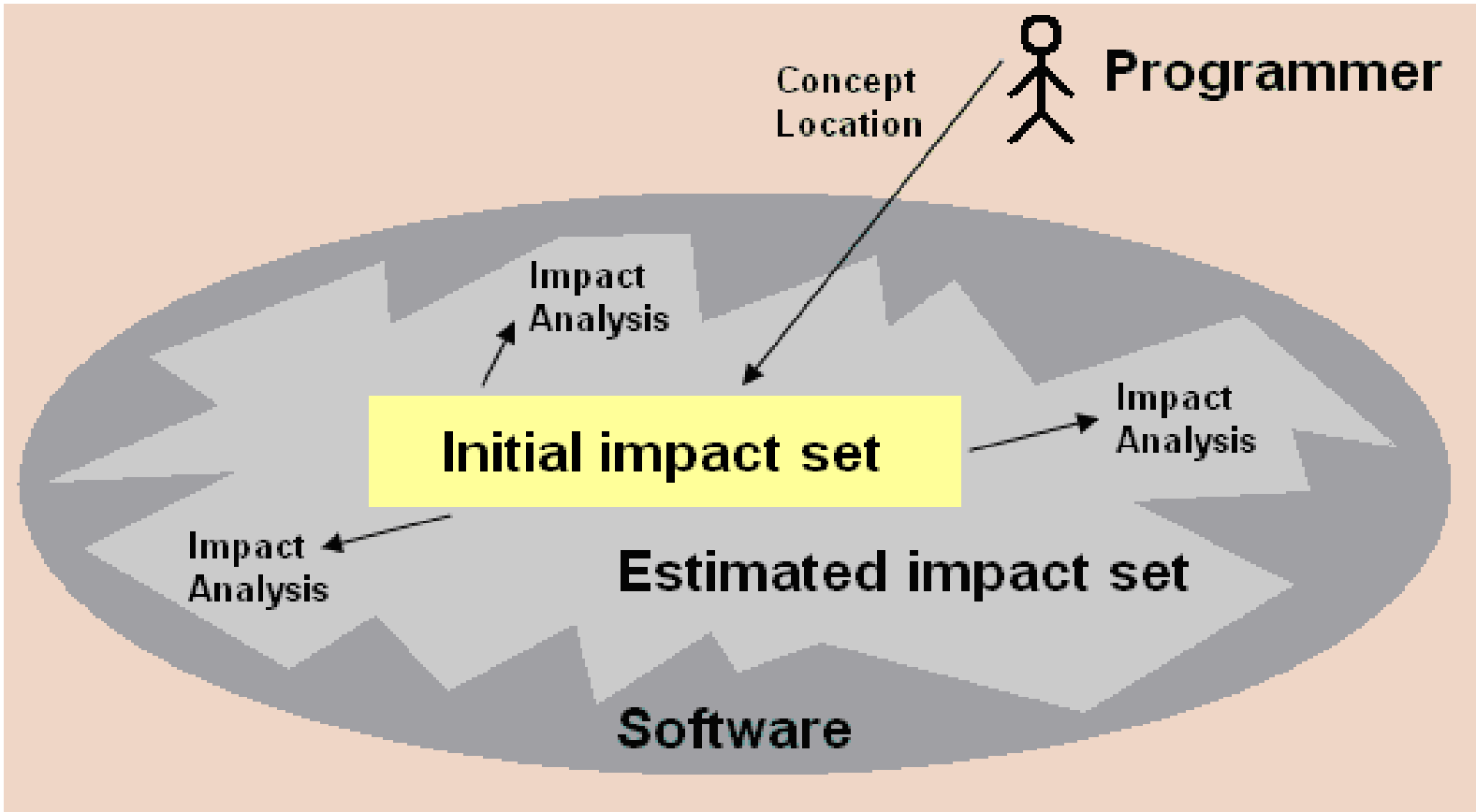
Impact and Feature Analysis

Impact analysis (IA)

- Determines the strategy and impact of change
- Classes identified in concept location are the initial impact set
- Class dependencies are analyzed, and impacted classes are added to the impact set
- Produces estimated impact set



Initial and estimated impact set



Class Interactions

- Two classes interact if the have something in common
  - One depends on the other. There is a contract between them
  - They coordinate. They share the same coding, schedule, etc.

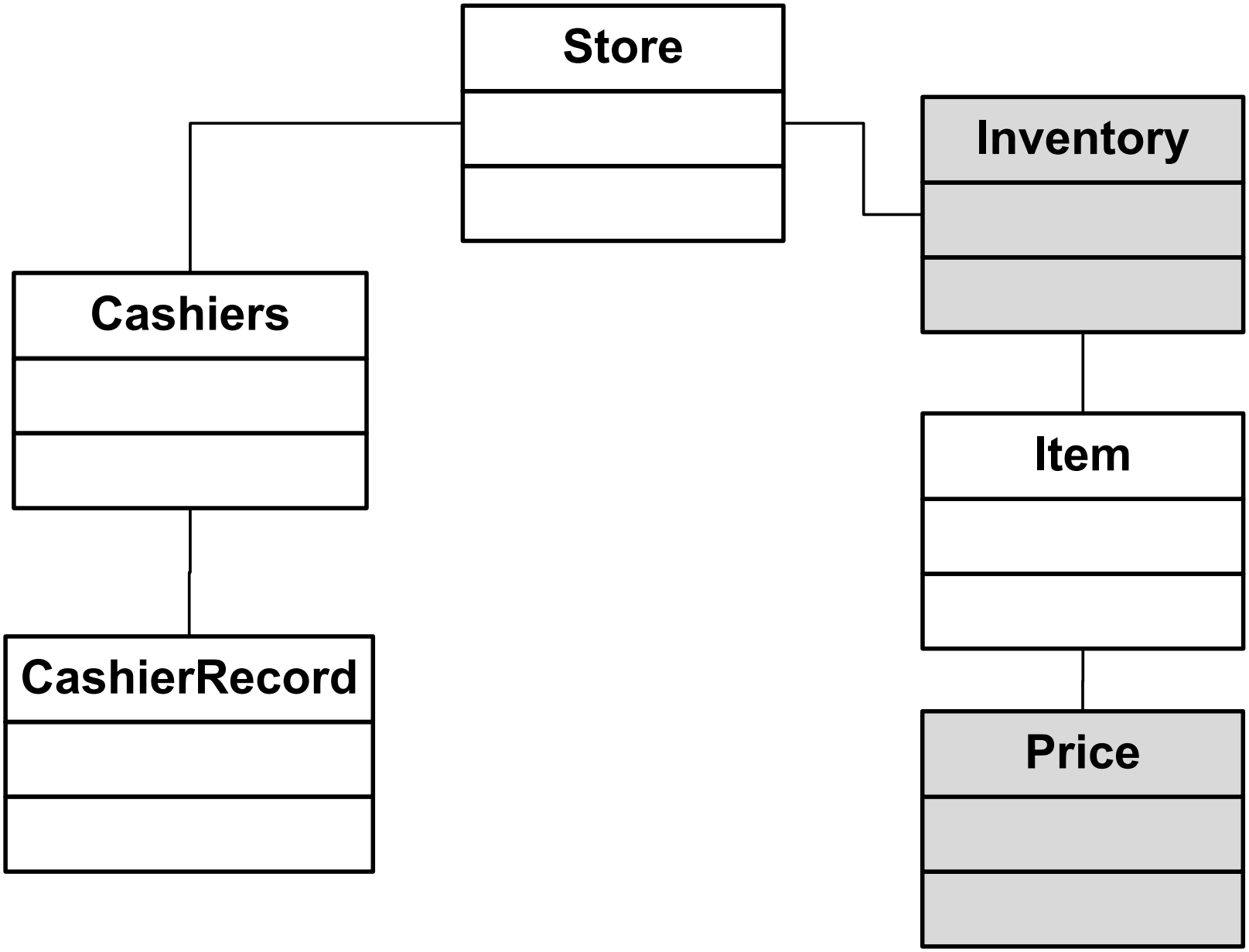
- Interactions propagate change in both directions
  - From A to B or from B to A

Class Interaction Graph

- $G = (X, I)$
  - X ... set of classes
  - I ... set of interactions

- Neighborhood of class A
  - $N(A) = \{B \mid (A, B) \in I\}$

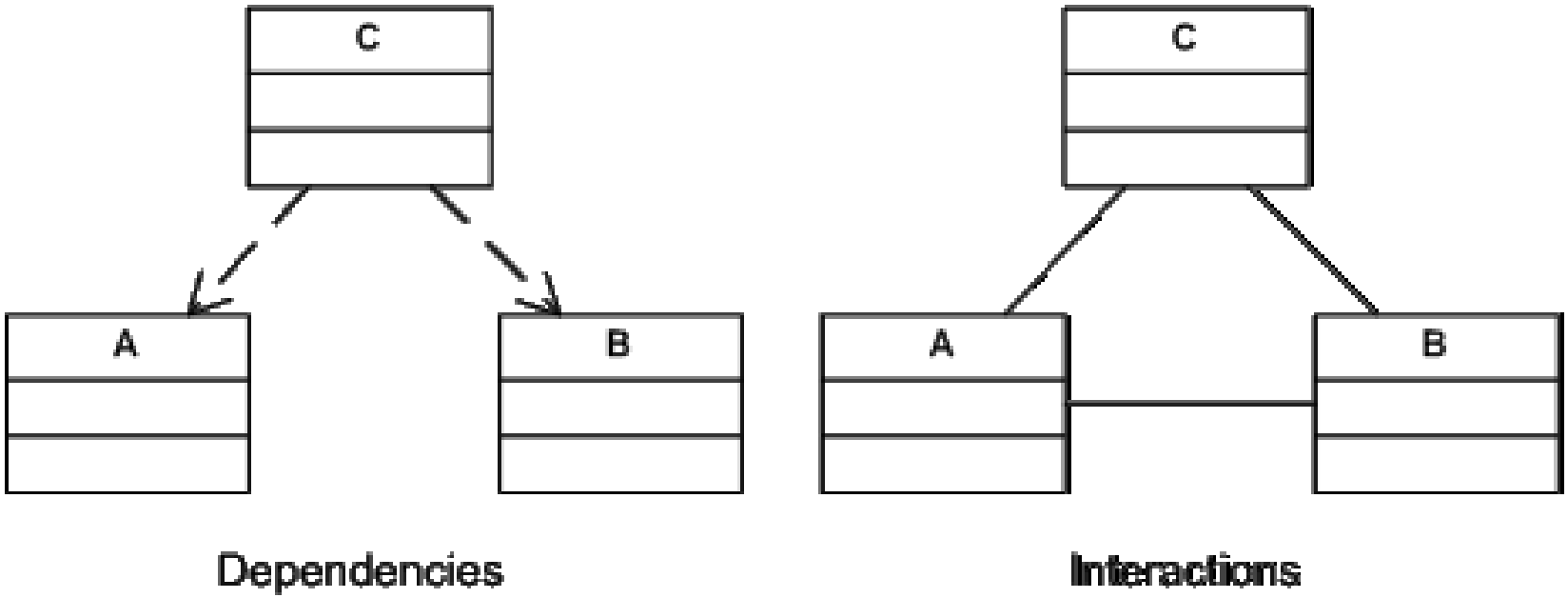
Neighborhood of Item



Coordination

```
class C
  {
    A   a;          //gets the color code
    B   b;          //paints the screen
    void foo()
      {
        b.paint(a.get()); // dataflow between a and b
      }
  };
```

Dependency diagram, Interaction diagram



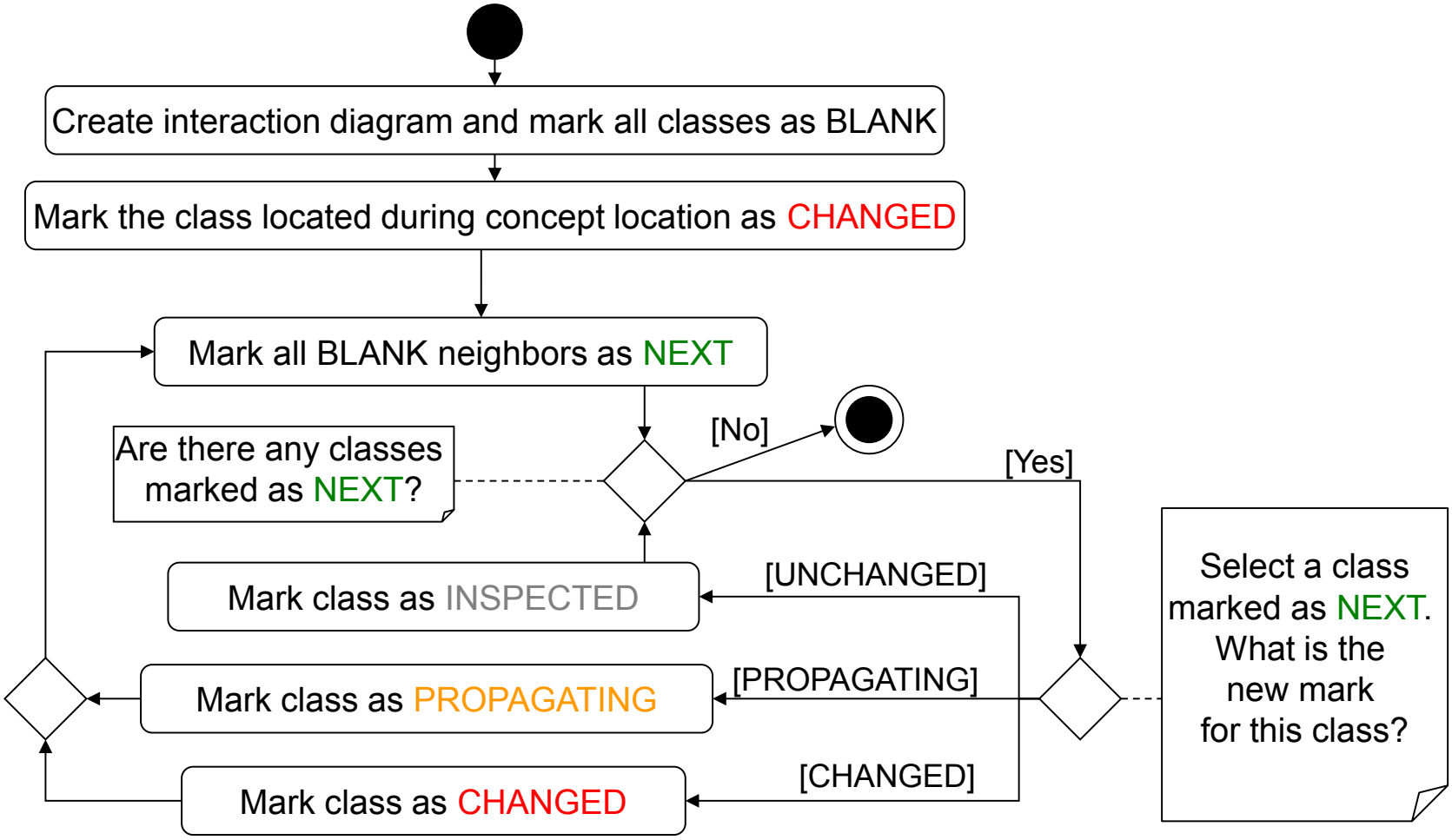
Propagating class: Mailman

- John loaned money to Paul:
  - Needs the money back
  - His situation changed
- John writes a letter to Paul:
  - Mailman takes the letter from John to Paul
  - Paul must take a part-time job
  - A big change that propagated from John to Paul
- John interacts with the mailman
- The mailman interact with Paul
- The change originated with John and propagates through the mailman to Paul

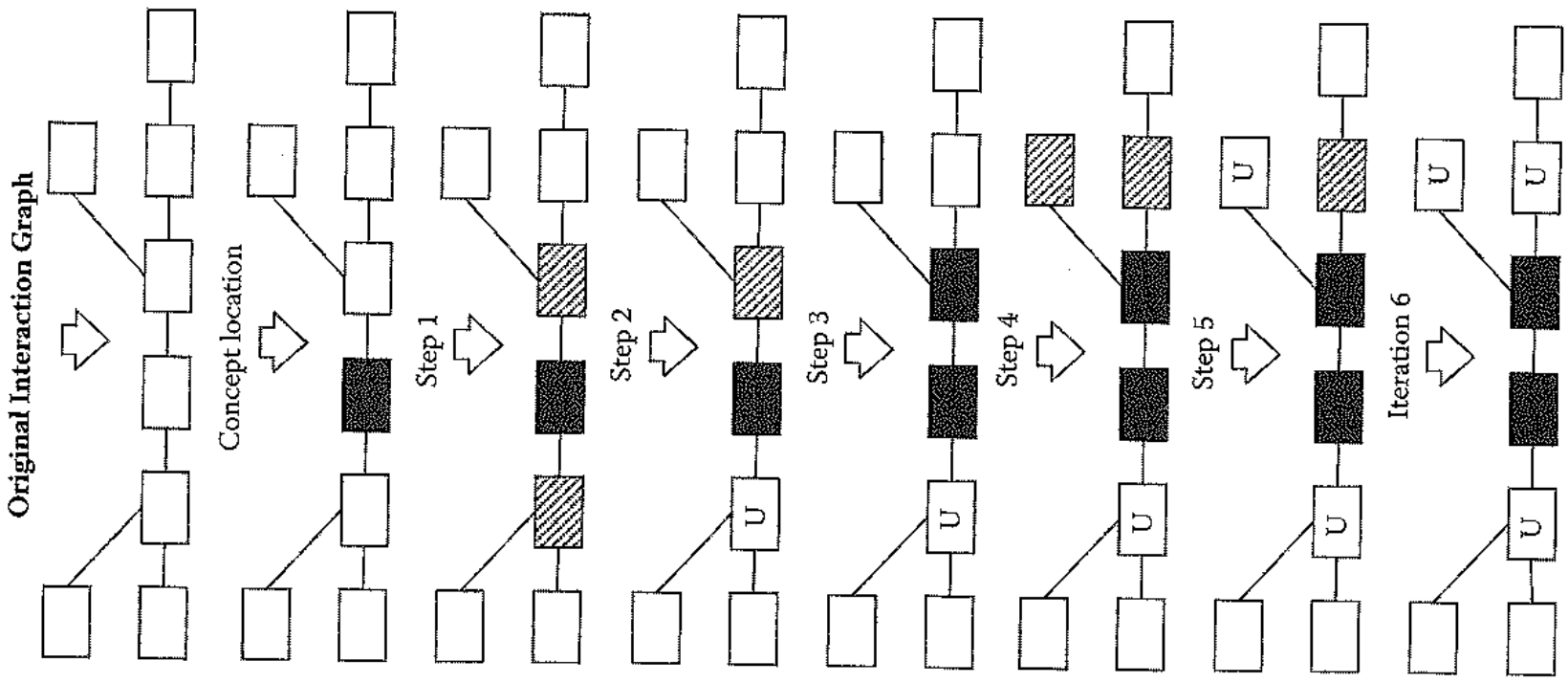
Process of Impact Analysis

Blank	The class was never inspected and is not scheduled for an inspection.
Changed	The programmers inspected the class and found that it is impacted by the change
Unchanged	The programmers inspected the class and found that it is not impacted by the change.
Next	The class is scheduled for inspection

IA including propagating classes



IA Example


Original Interaction Graph
Concept location
Step 1
Step 2
Step 3
Step 4
Step 5
Iteration 6
U

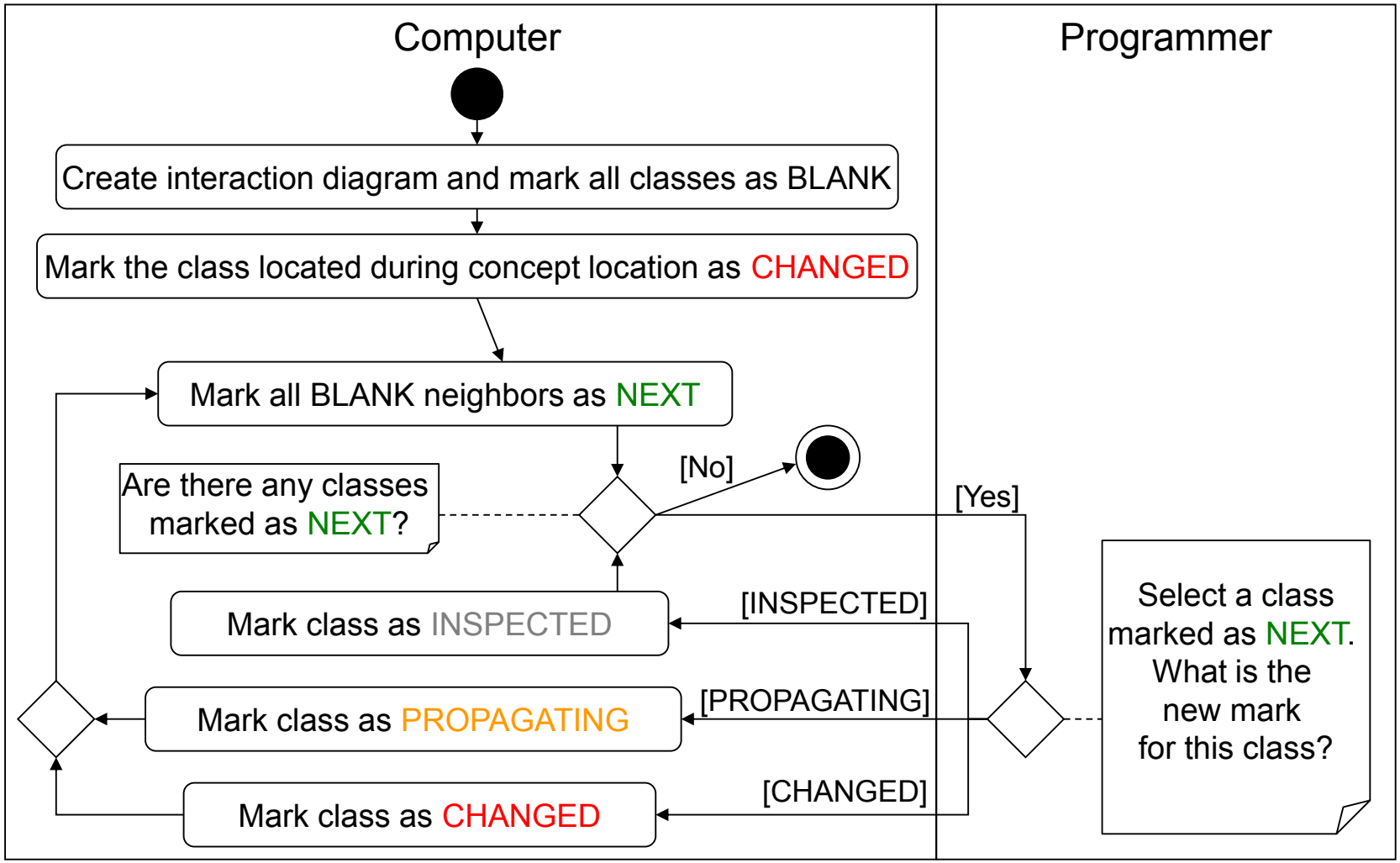
Alternatives in Software Change

- Program displaying a temperature in Fahrenheit
  - change request: display it in Celsius
- Two separate locations deal with temperature
  - sensor data converted to the temperature
  - temperature displayed to the user
- The change can be done in either place
  - impact analysis weights these alternatives

The Criteria

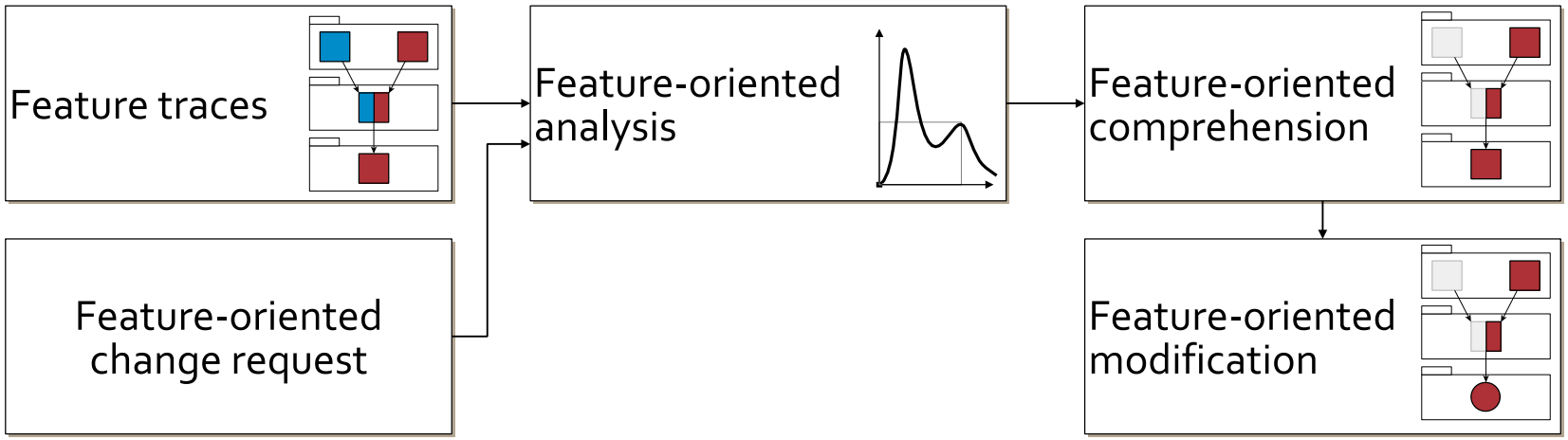
- Required effort of the change
- Clarity of the resulting code
- Often, these two criteria contradict each other:
  - it is easier to adjust the user interface
  - it is better to have all calculations of the temperature in one place
- Conflict between short-term and long-term goals

Interactive IA



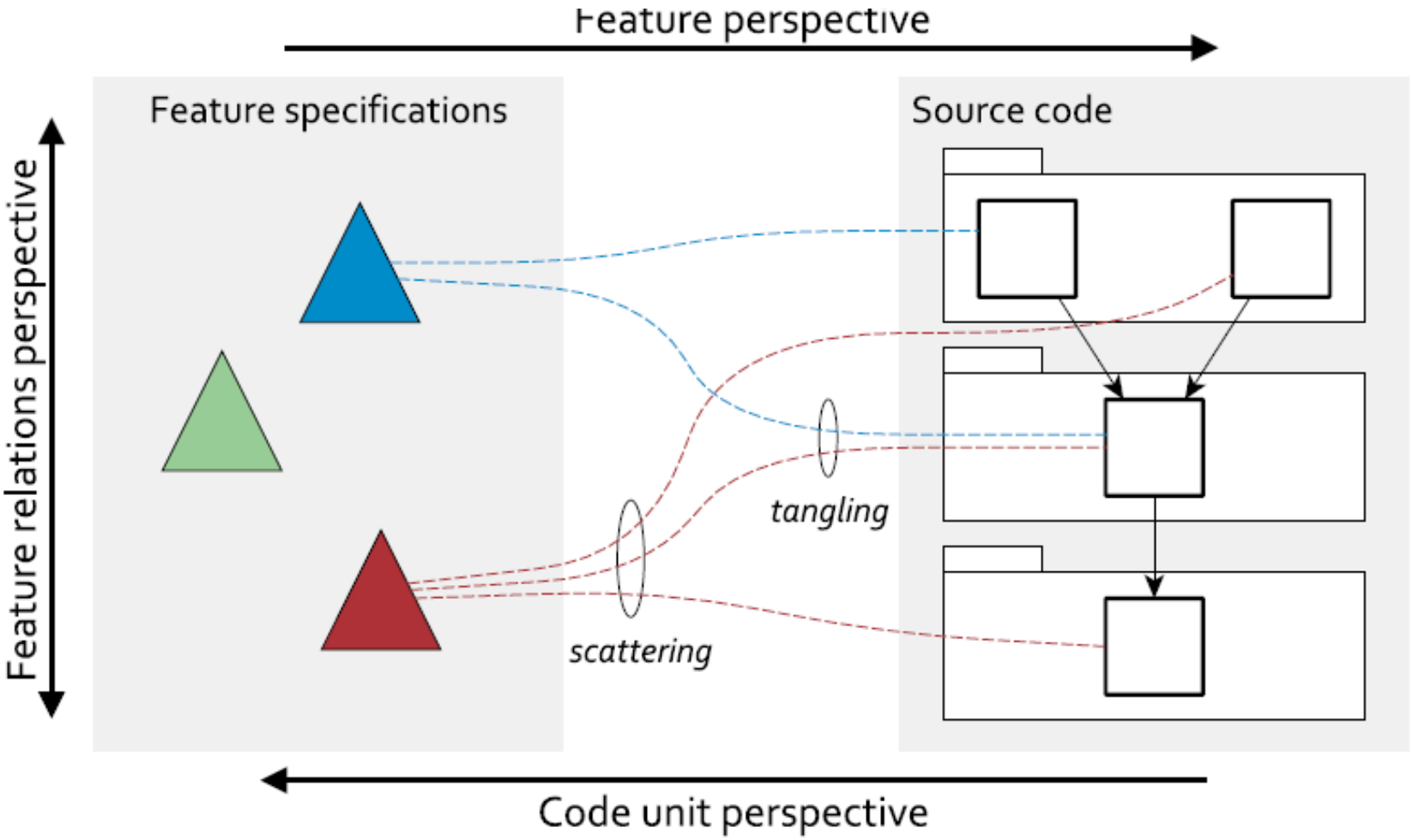
Featureous Analysis

Features as Units of Code Analysis



- Feature-oriented analysis treats features as first-class code investigation entities
- Several metrics and visualizations exist
  - Not always compatible with one another
  - Need firm evolutionary grounding
  - Lack of usable implementations

Perspectives



Feature Scattering

$$\boldsymbol{FSCA}(F) = \sum_{f \in F} fsca(f)\,, \;\; where: \boldsymbol{fsca}(f) = \frac{|\{p \in P_F : f \rightsquigarrow p\}| - 1}{|F| \cdot |P_F|}$$

FSCA measures the average number of packages P_F that contribute to implementing application features F (i.e. packages that fulfill the $\rightsquigarrow$ "implemented by" relation with features). This value is normalized according to the number of non-insulated packages P_F found in the application.

Feature Tangling

$$\boldsymbol{FTANG}(P_F) = \sum_{p \in P} ftang(p)\,, \;\; where: \boldsymbol{ftang}(p) = \frac{|\{f \in F : f \rightsquigarrow p\}| - 1}{|F| \cdot |P_F|}$$

FTANG is a measure complementary to *FSCA*, as it quantifies the average number of features F that use packages P.

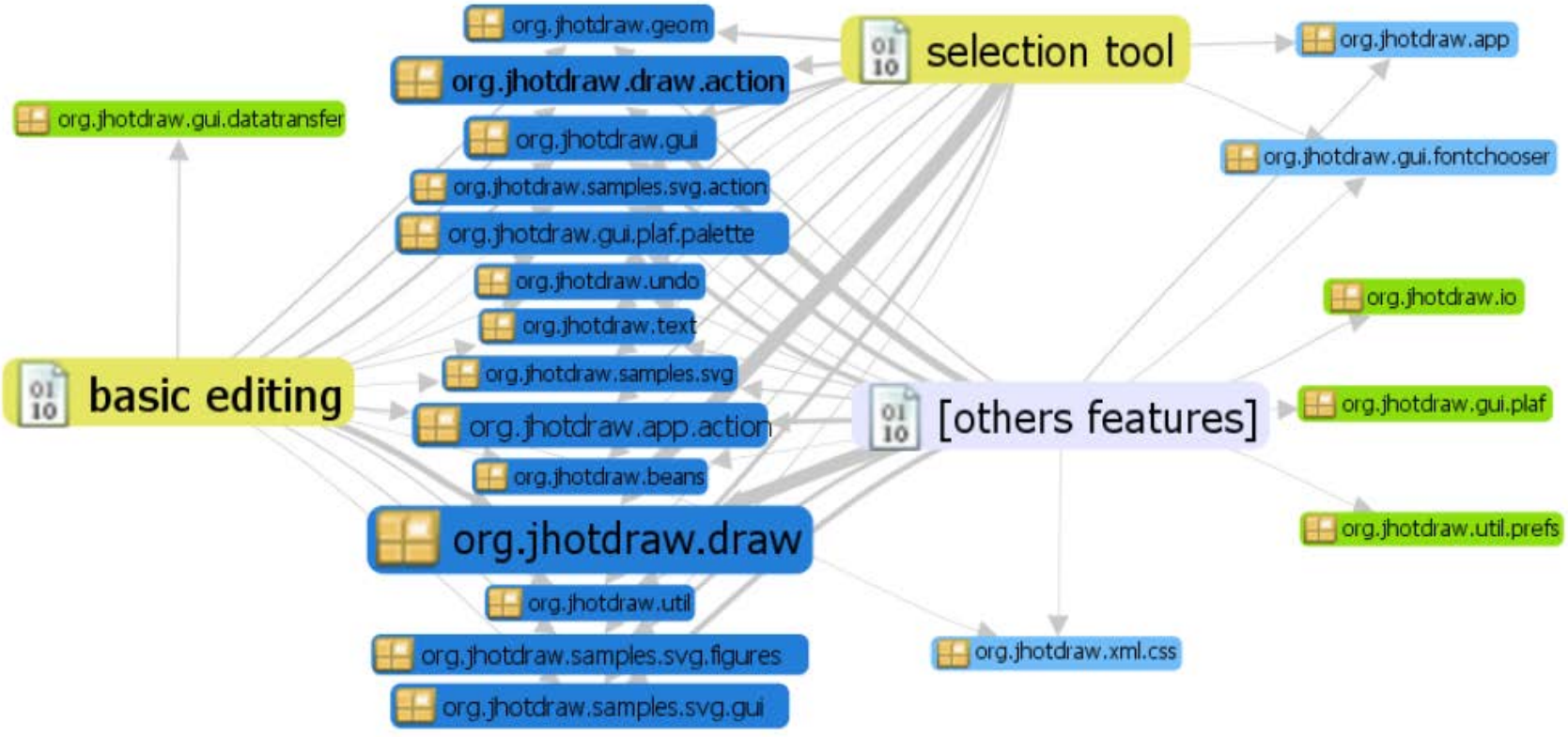
Feature-code Correlation Grid

	parameters	save image	import data
persistence.SavePngImageAction		1	
dataimportcsv.ImportCSVAction			
dataimportcsv.DataImportCSV			
persistence.DimStackImageFileF...		*4	
parameters.api.ParametersUtils	5		
parameters.ParametersController	*6, *7		
app.NDVis	8	8	
io.FileUtils			
image.ImagePanel	9, 10	10	
parameters.api.Parameters	11, 12		
model.DataInfo	13		
dbutils.DatabaseUtilities			

Feature-code Correlation Grid

- Correlates features with packages or classes using a **matrix-based representation**.
- In the case of classes, the matrix displays the concrete **identifiers of the objects** of a given class used by features.
- Sorting of the matrix's rows and columns is done according to the **scattering** of features and **tangling** of code units.

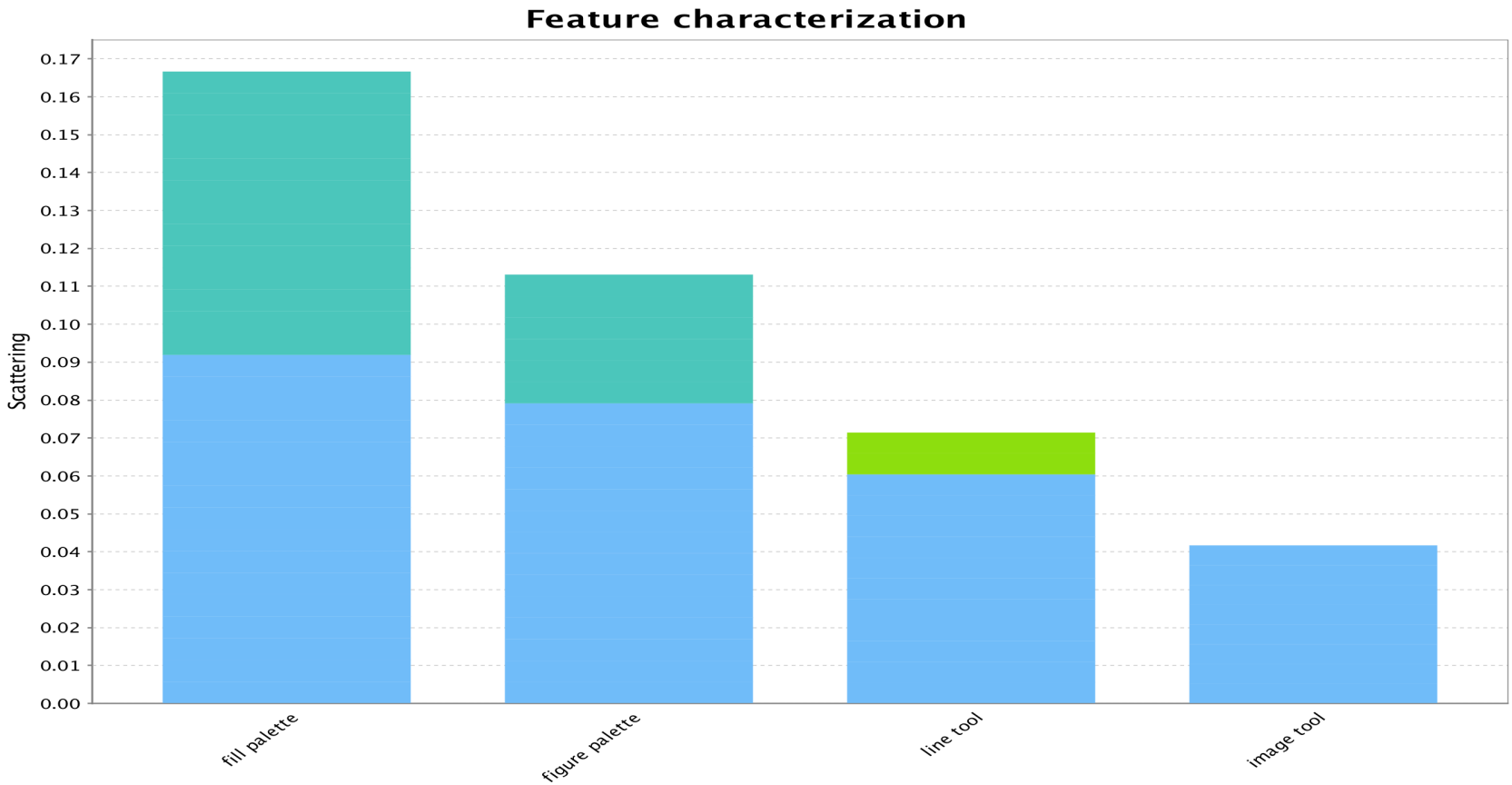
Feature-Package Correlation Graph



Feature-Code Correlation Graph

- Depicts the correspondences between **source code units** and features.
- Graph-based visualization of **traceability links** (edges) between fetures (nodes) and classes or packages (nodes).
- **Edge-thickness-based** indication of the number of contained source code units that participate in the traceability relations with individual features.
- Heights of the source code unit nodes reflect the **numbers of their contained finer-grained units** (e.g. the height of a package node reflects the number of the classes that it contains).

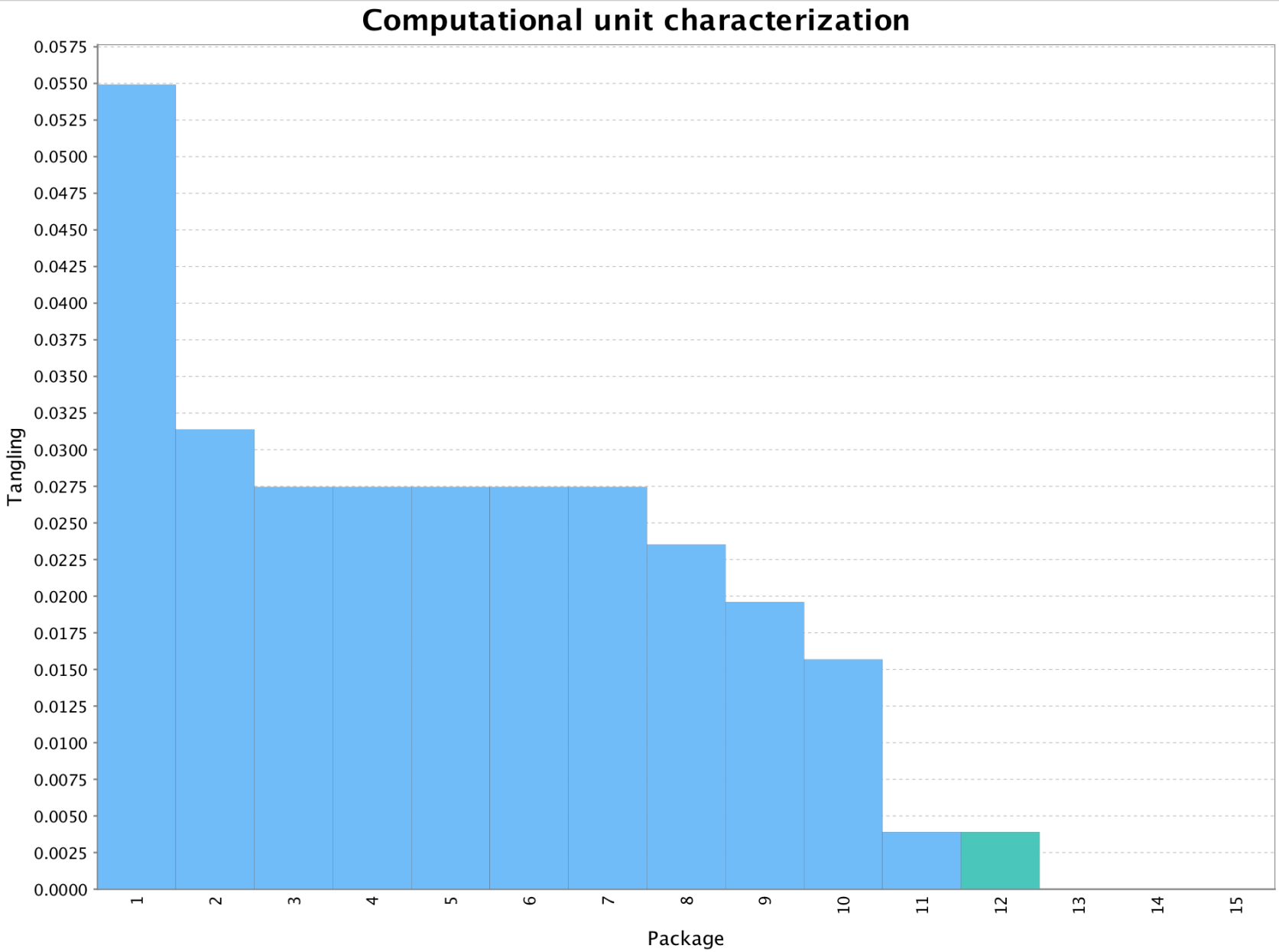
Feature-Code Characterization



Feature Scattering

- In turn, each of the bars represents a single **package** or **class** used by a feature.
- The names of the source code units that the bars represent can be displayed **on-demand as tooltips**.
- The height of the individual bars represents **Scattering of their corresponding code units**, measured by the Scattering metric.

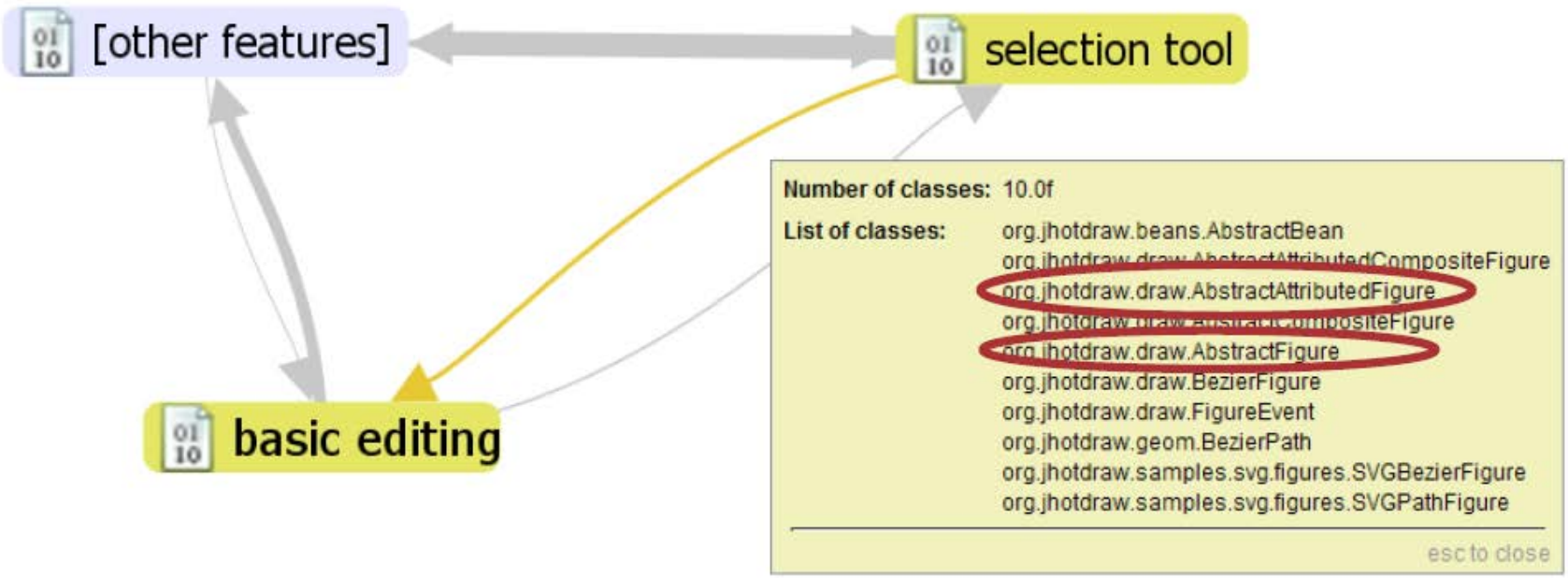
Feature-Code Characterization



Feature Tangling

- In turn, each of the bars represents a single **package** or **class** used by a feature.
- The names of the source code units that the bars represent can be displayed **on-demand as tooltips**.
- The height of the individual bars represents **tangling of their corresponding code units**, measured by the tangling metric.
- Tangling indicates the potential **inter-feature change propagation**.

Feature Relations Characterization



Feature Relations Characterization

- Relates features with respect to how they depend on the **objects created by other features** and how they **exchange data by sharing these objects** with one another.
- Features, denoted as **vertices**, are connected by dashed **edges** in case of common usage of objects at runtime.
- A directed, **solid edge** is drawn if a pair of features is in a producer-consumer relation.
- **Edge thickness** reflect the number of classes whose objects are being shared between features.