Aspect Oriented Knowledge-Driven Evolution of Software Product Lines With Hierarchically-Expressed Variability Information Preserved in Code

Software Knowledge Comprehension and Reuse

Author: Jakub Perdek

Motivation: Studying the SPL evolution and variability

Less rigorous evaluations of variability management

-knowledge modeling,
-applying principles of variability modeling
-simulating feature interactions

…to handle variability

General handling of the variability is still not fully covered/supported by variability management

M. Galster, D. Weyns, D. Tofan, B. Michalik, and P. Avgeriou. Variability in software systems—a systematic literature review. IEEE Transactions on Software Engineering, 40(3):282–306, 2014.

- Data should be used further to ***detect defects*** and ***provide quality assurance*** between selected variants

L. Chen, M. Ali Babar, and N. Ali. Variability management in software product lines: A systematic review. pages 81–90, 01 2009.

various models and data representations are required for this purpose

Can our software product line be capable to support a high number of given requirements?

from possible "solution space" as the reaction on the previous slide

L. Chen, M. Ali Babar, and N. Ali. Variability management in software product lines: A systematic review. pages 81–90, 01 2009.

Modeling variability of software products as part of software product families under different settings

- possibility to evaluate supporting methods and tune them

- possibility to observe problems with automatic management of configuration expressions

Adaptation of evolutionary algorithms for SPL

Creating and managing catalogs of "correctly" annotated scripts

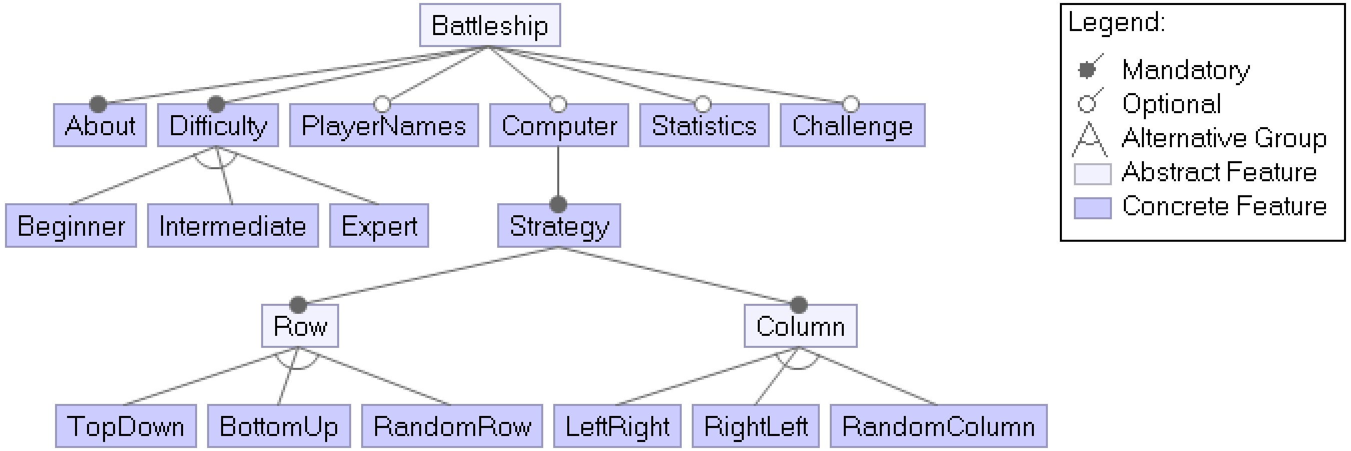
Example of Given solution

Text interface

Console game – can play using command line

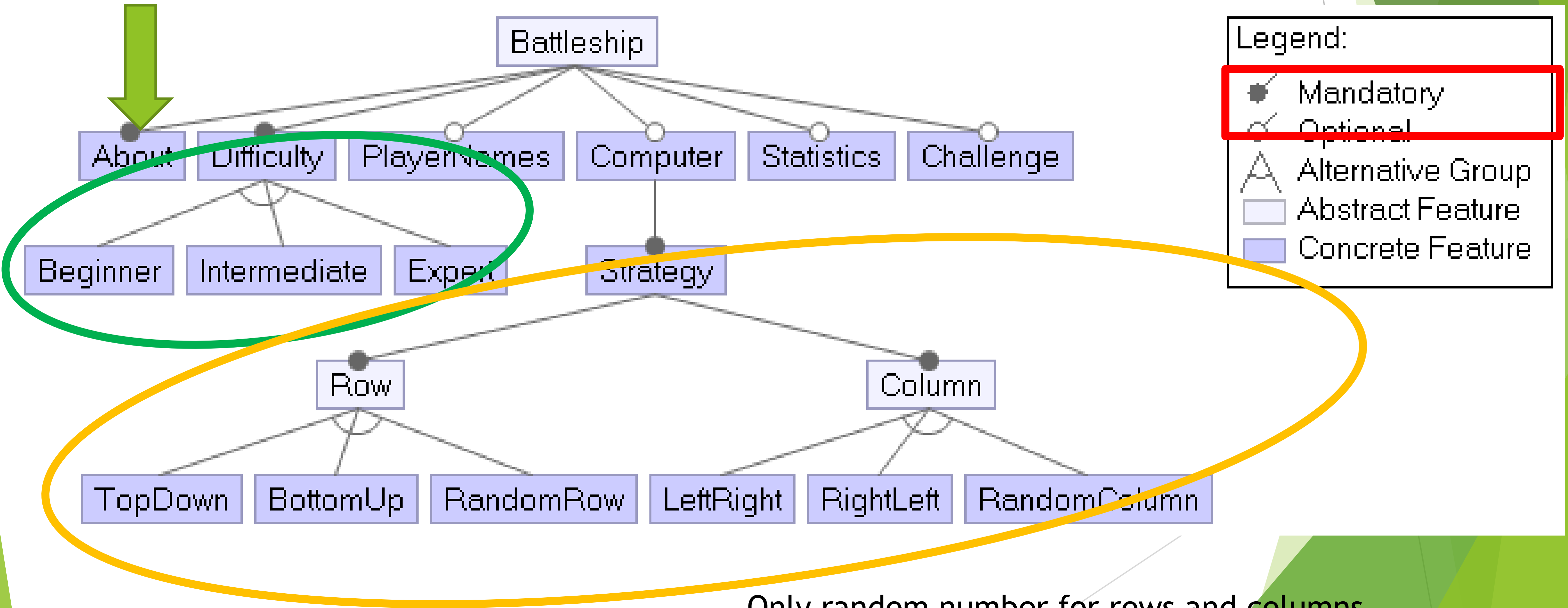
Real application

- **Feature analysis already created - domain already analyzed**
- Suitable for next implementation and improvements



Mandatory parts in the game

Prints info
at the beginnning



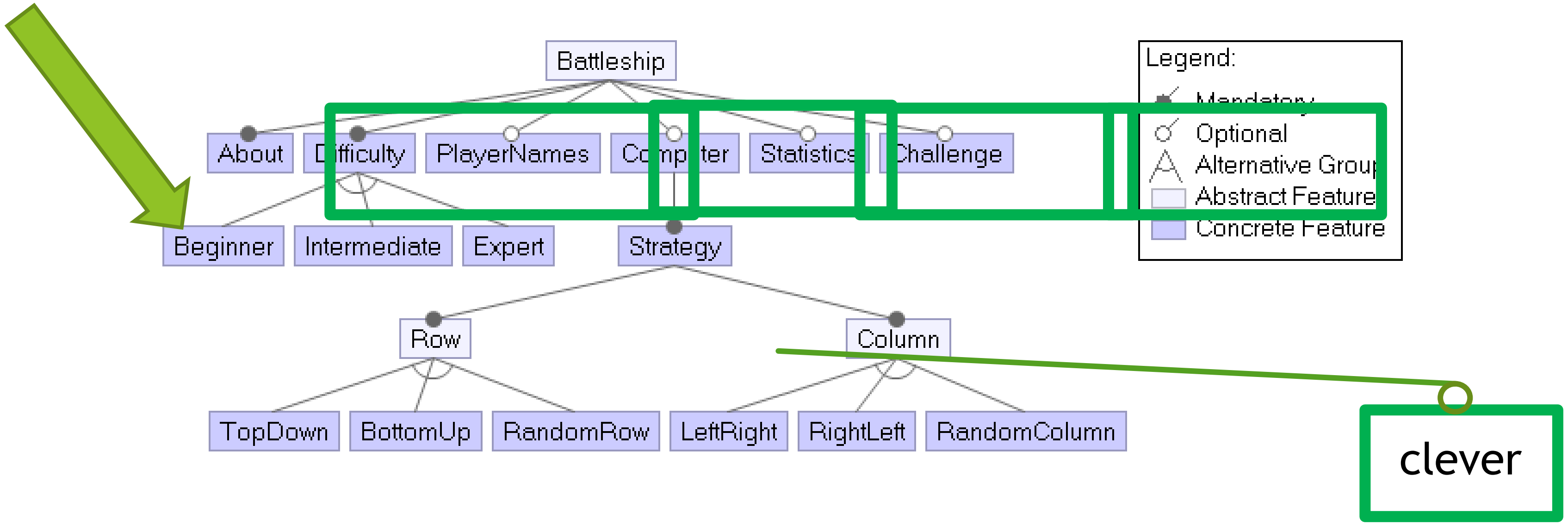
Only random number for rows and columns

Our focus

- focus on variable features

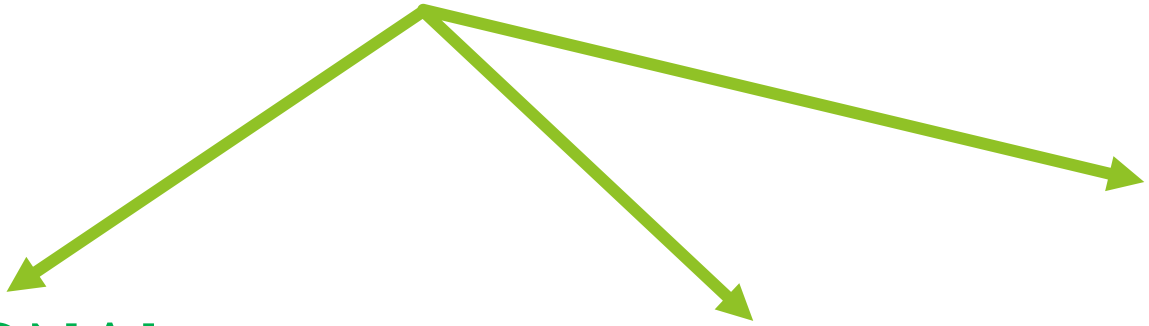
Chosing one feature
From set can be
Implemented
in aspect oriented way

+ maybe other potential improvements from observed domain



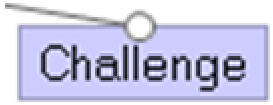
From the same
author in another
his scatch

Types of variations



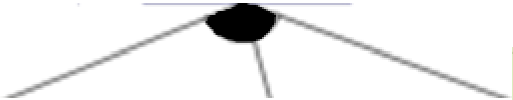
OPTIONAL VARIATION

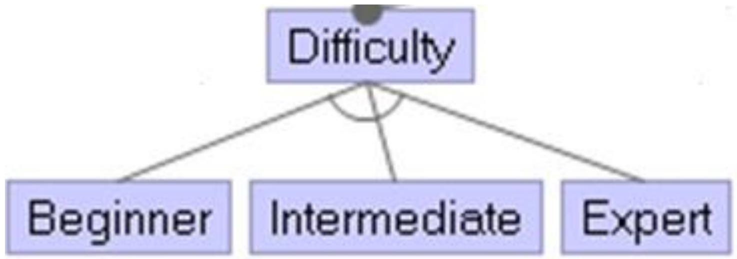
0 or 1 instance



INSTANCE OF SEVERAL INSTANCES

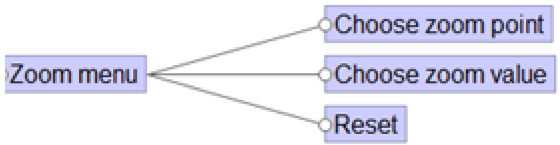
1 instance from
n instances





SET OF INSTANCES OUT OF SEVERAL ALTERNATIVES

k instances from
n instances



Felix Bachmann and Len Bass. 2001. Managing Variability in Software Architectures. (2001), 7.

Meta-model Concepts

DESCRIBED WITH VARIANT

How target models vary, based on

Selected

Unselected

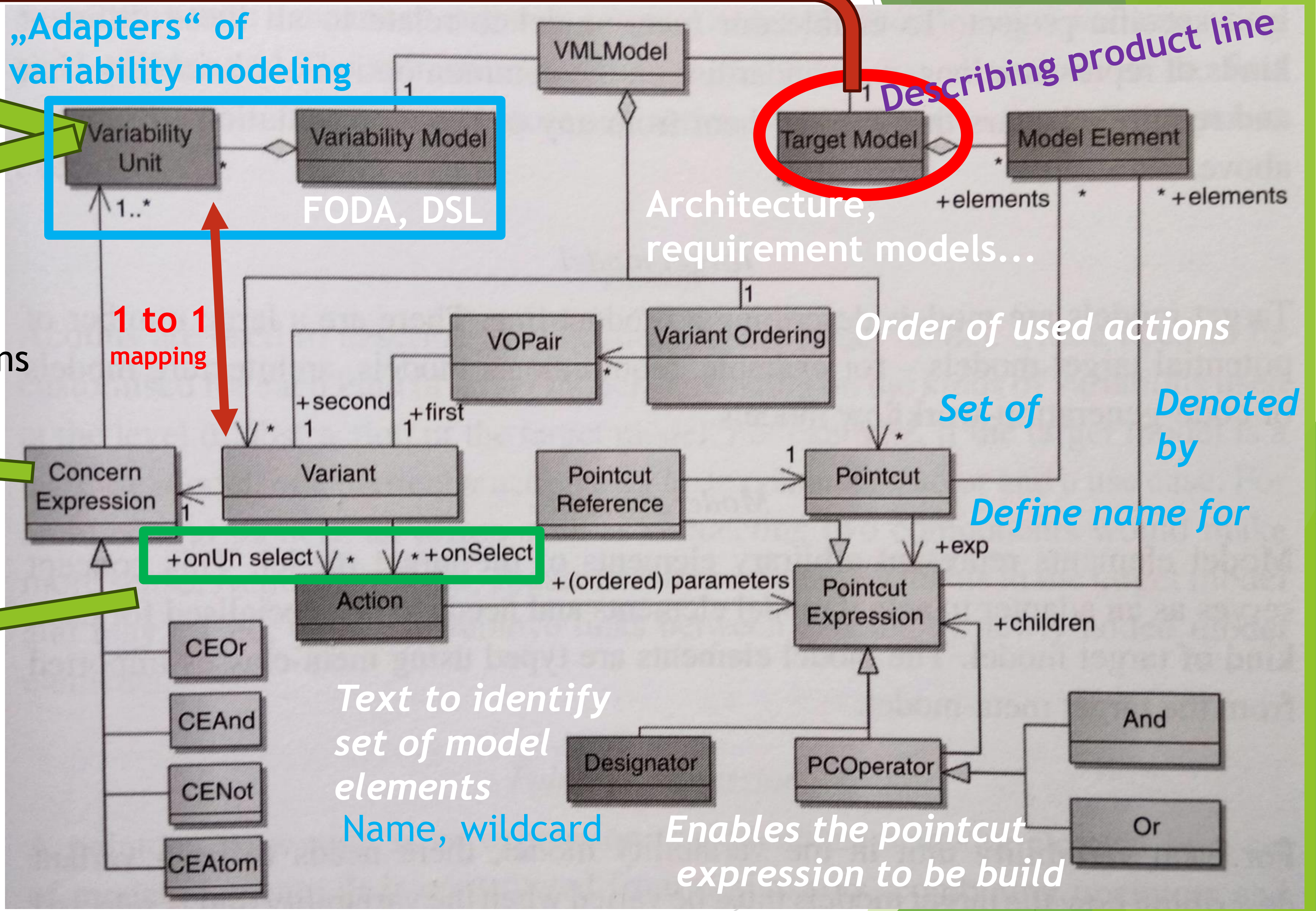
Prescribed by variability model:
- what they may have
- constraints governing selections

DEFINE CONCERNS to variability rather than target models

With AND, OR NOT operators

VARIABILITY OPERATIONS

To handle:
- positive variability
- negative variability



„Adapters“ of variability modeling

FODA, DSL

1 to 1 mapping

Describing product line

Architecture, requirement models...

Order of used actions

Set of

Denoted by

Define name for

Text to identify set of model elements

Name, wildcard

Enables the pointcut expression to be build

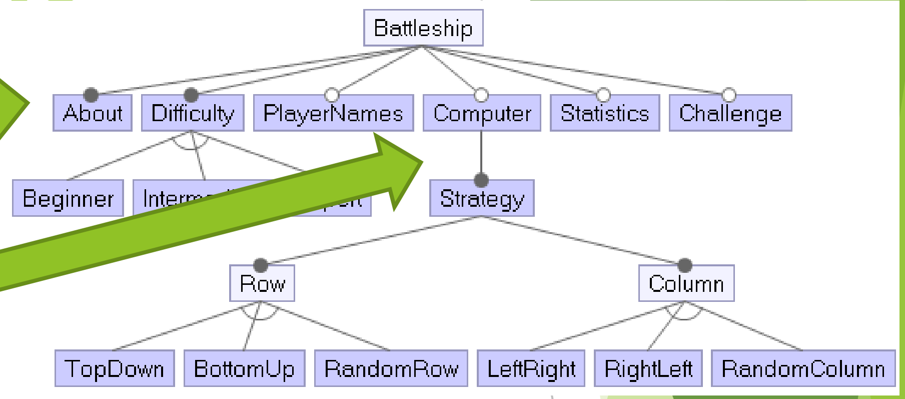
Rashid, A., Royer, J., & Rummler, A. (Eds.). (2011). *Aspect-Oriented, Model-Driven Software Product Lines: The AMPLE Way*. Cambridge: Cambridge University Press. doi:10.1017/CBO9781139003629

Hierarchic nature of configuration expressions

- {
  - “AND“: {
    - “Statistics”: true,
    - “Challenge”: false,
    - “AND”: {
      - “Computer”: true,
      - “Row”: “RandomRow”,
      - “Column”: “RandomColumn”
    - }
  - }
- }

Configuration of the first later

Configuration related to computer as player



Focus during their creation can be on:

- hierarchy levels
- feature groups
- certain hierarchies

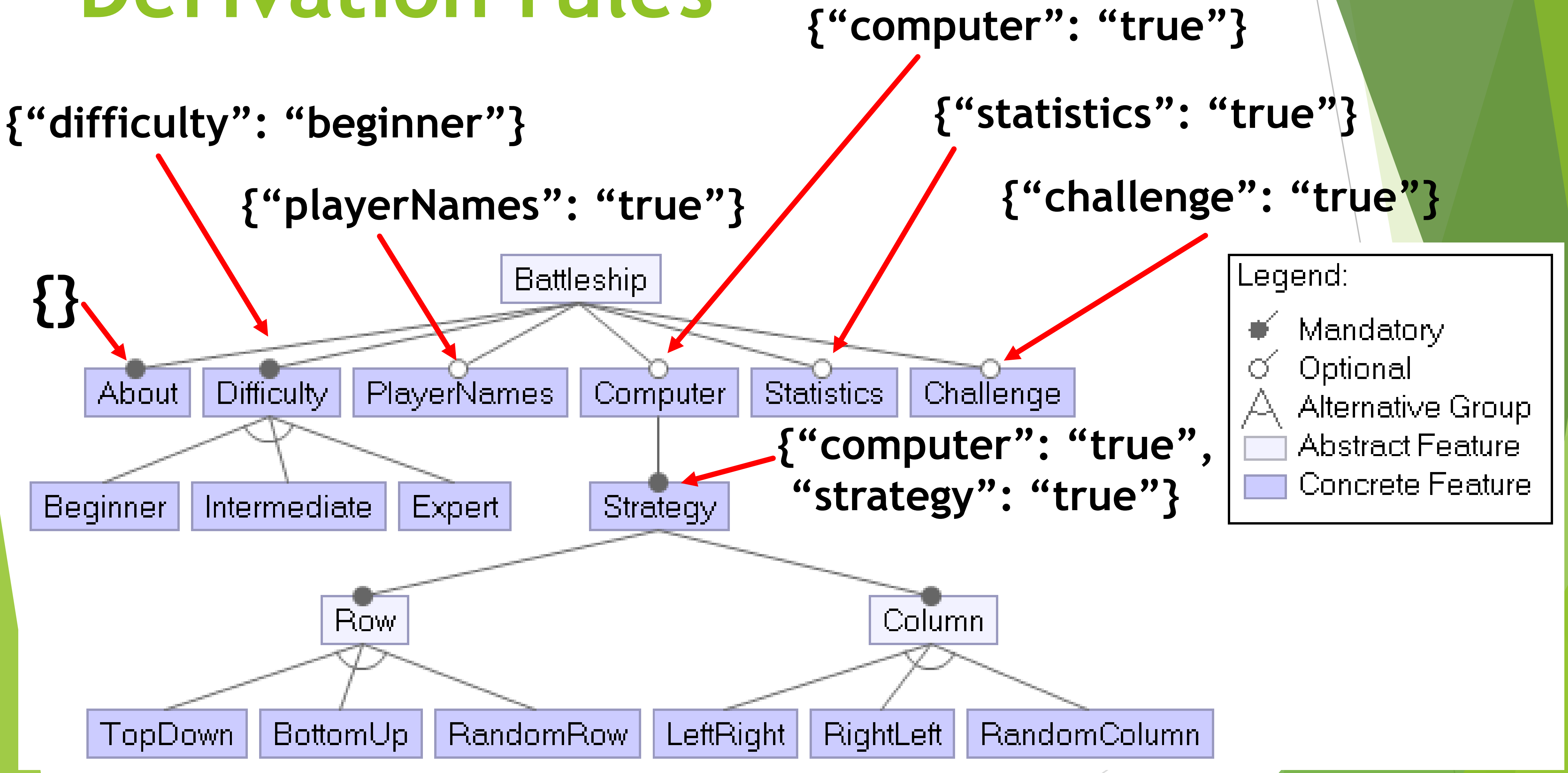
Product derivation

PROBLEM SPACE → SOLUTION SPACE

Product derivation

PROBLEM SPACE → SOLUTION SPACE

Derivation rules



{"computer": "true"}

{"statistics": "true"}

{"difficulty": "beginner"}

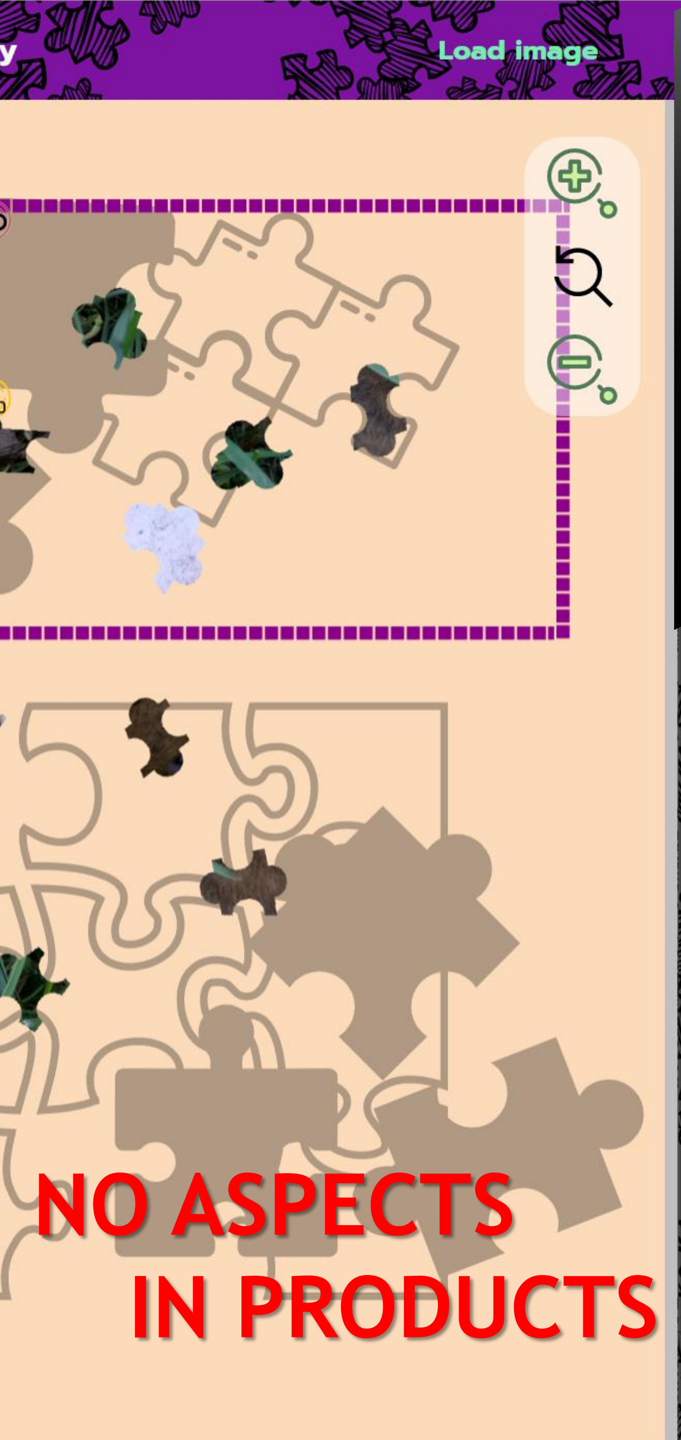
{"playerNames": "true"}

{"challenge": "true"}

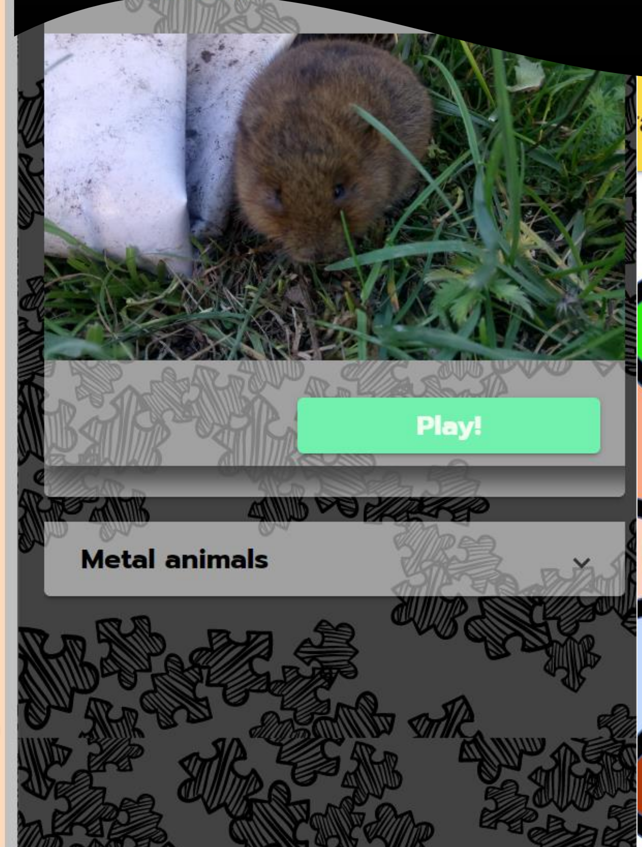
{}

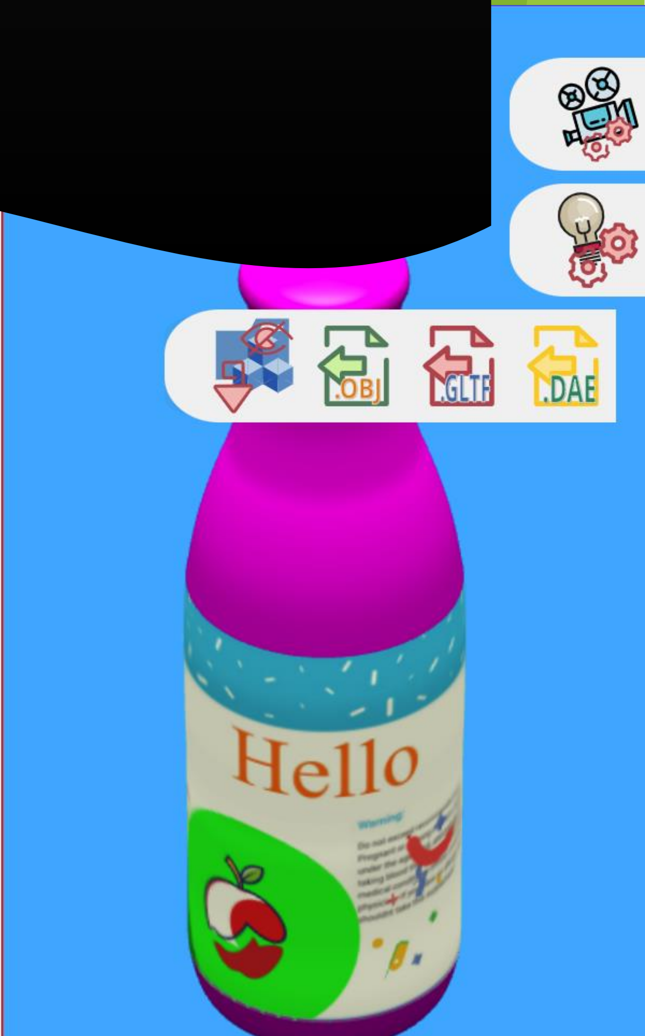
{"computer": "true", "strategy": "true"}

Application in TypeScript

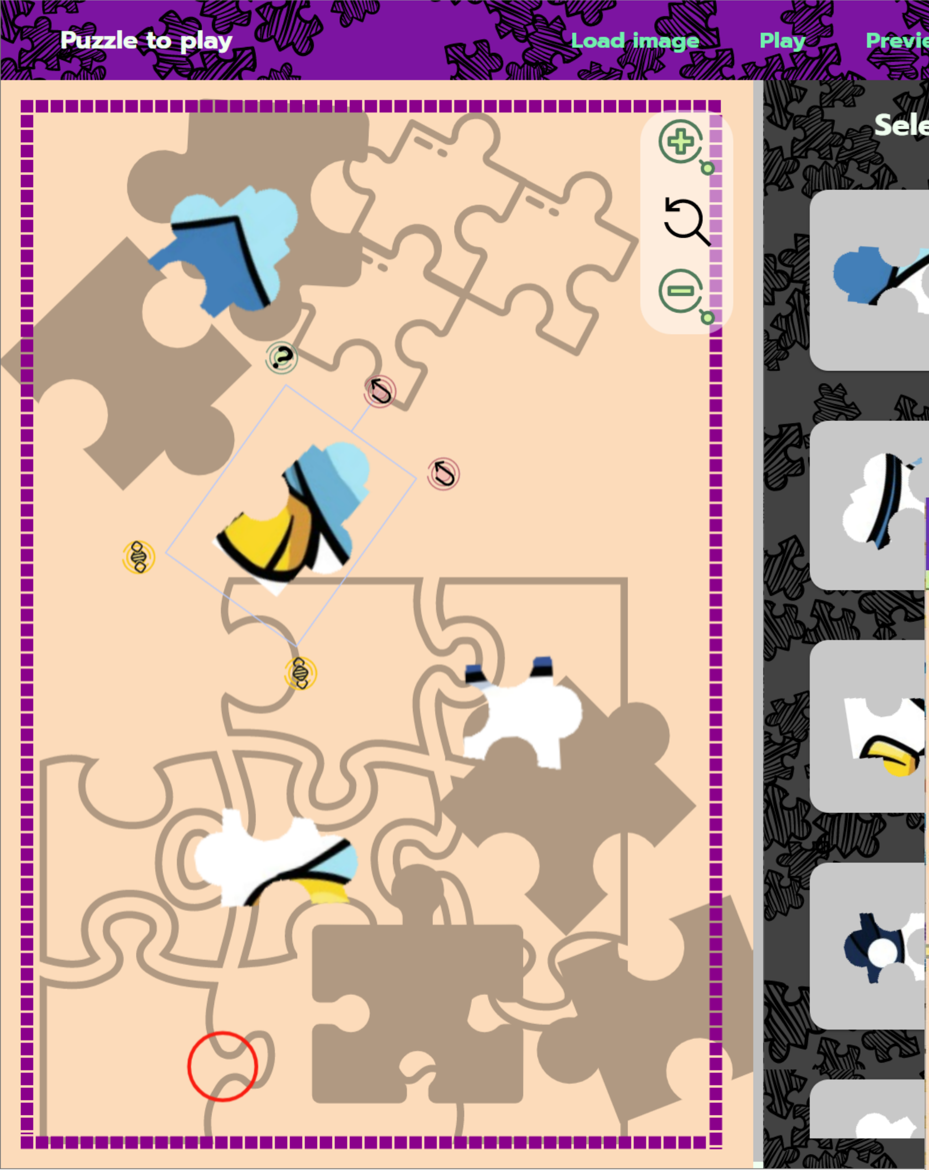


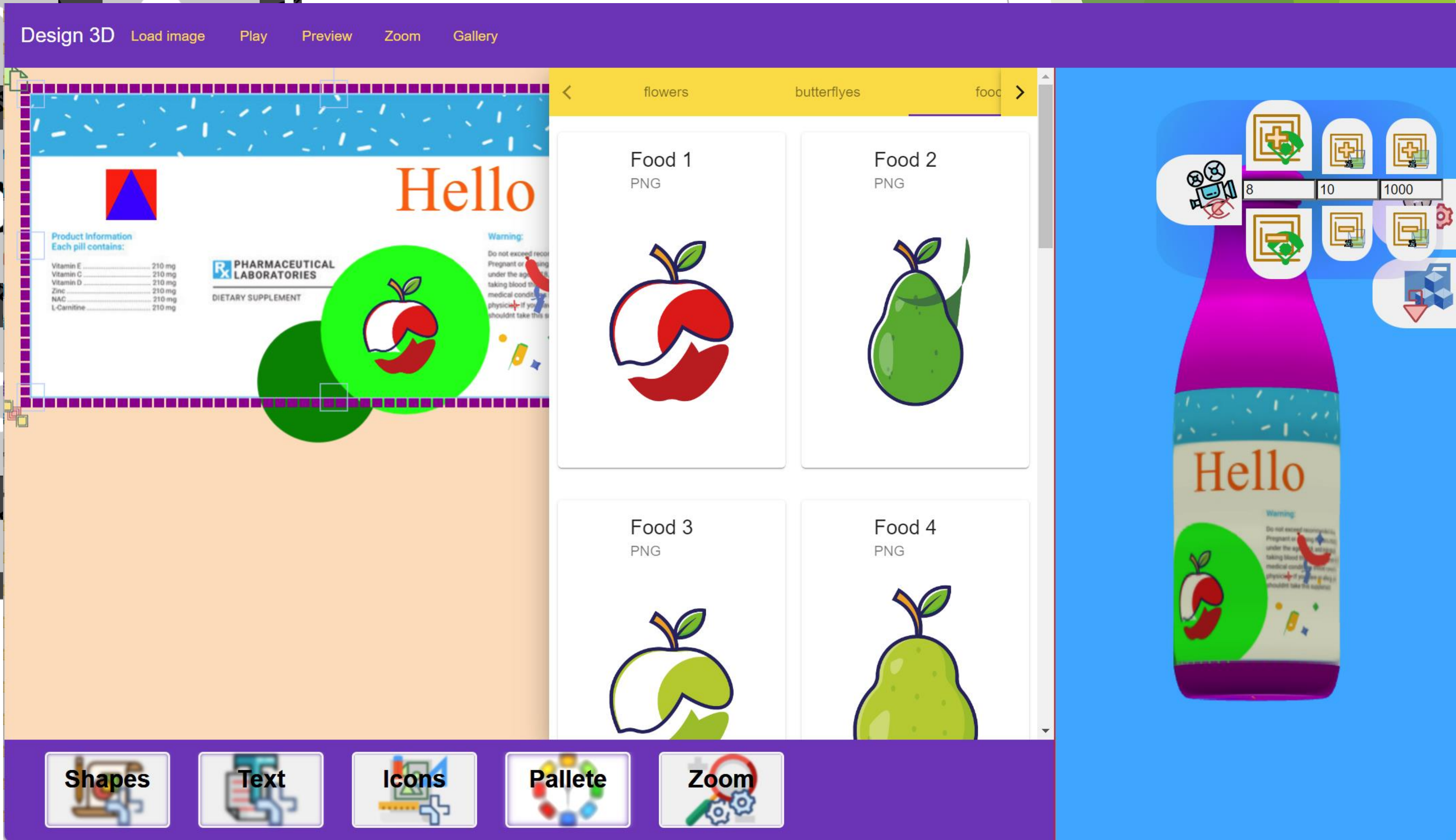
NO ASPECTS
IN PRODUCTS





Commonality vs. Variability

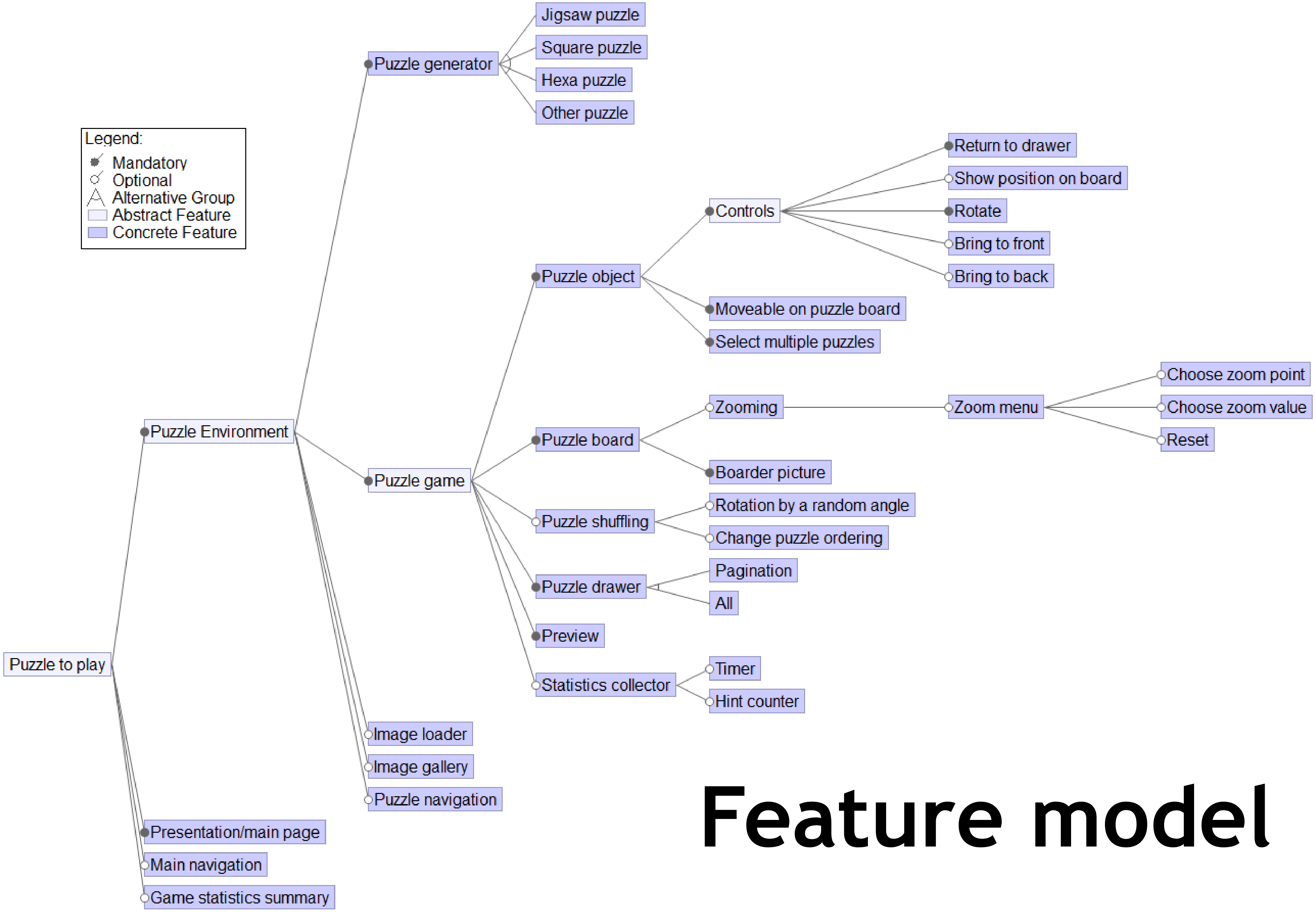




Puzzle app.

vs. Desing app.

Feature model



Legend:
- Mandatory
- Optional
- Alternative Group
- Abstract Feature
- Concrete Feature

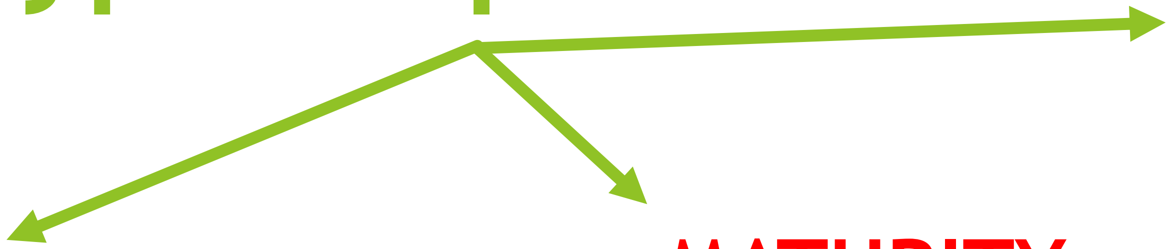
- Puzzle to play
  - Puzzle Environment
    - Puzzle generator
      - Jigsaw puzzle
      - Square puzzle
      - Hexa puzzle
      - Other puzzle
    - Puzzle game
      - Puzzle object
        - Controls
          - Return to drawer
          - Show position on board
          - Rotate
          - Bring to front
          - Bring to back
        - Moveable on puzzle board
        - Select multiple puzzles
      - Puzzle board
        - Zooming
          - Zoom menu
            - Choose zoom point
            - Choose zoom value
            - Reset
        - Boarder picture
      - Puzzle shuffling
        - Rotation by a random angle
        - Change puzzle ordering
      - Puzzle drawer
        - Pagination
        - All
      - Preview
      - Statistics collector
        - Timer
        - Hint counter
    - Image loader
    - Image gallery
    - Puzzle navigation
  - Presentation/main page
  - Main navigation
  - Game statistics summary

TypeScript product families

- In one application (without backend if necessary)
- Accessible from everywhere (from the browser)
- High UX possible (known elements, reactive forms, own routing,…)
- Possibility to easily evolve SPL
- Possibility to easily evolve product derivation (aspects are not dependent here)
- Reusing proven solutions (resizing canvas (board) during play, rendering algorithms,..)
- Customization of graphic libraries for each specific case
  - Managing small variability changes across many types of products and requirements

ASPECTS FOR SPL FEATURE MANAGEMENT ↕ **GETTING RID OF ASPECTS IN RESULTING PRODUCT DERIVATIONS**

Restrictions of using aspects in TypeScript



BRIEFNESS

How easy, how exactly, and without complications is possible to use a given tool

INVASIVENESS

How well aspects are separated from the rest of the code

MATURITY

All abilities and possibilities of the whole functionality provided by a given library

Komponent / Nástroj	AspectScript	AOJS	AspectJS
Invasiveness	-	+	-
Briefness	++	+	++
Maturity	++	-	-

The comparison of AOP tools (Huang et al. 2015)

Wenhao Huang, Chengwan He, and Zheng Li. 2015. A Comparison of Implementations for Aspect-Oriented JavaScript:. Zhengzhou, China. https://doi.org/10.2991/csic-15.2015.9

Ricardo Sá Loureiro Ferreira da Silva. 2019. Aspect-Oriented Programming for Javascript using the Lara Language. Dissertation thesis. Universidade do Porto, Porto.

SPL Process

- 1. Separating aspects from business logic
- 2. Adding business logic and annotating variable parts
- 3. Deriving requested products from SPL with NO ASPECTS

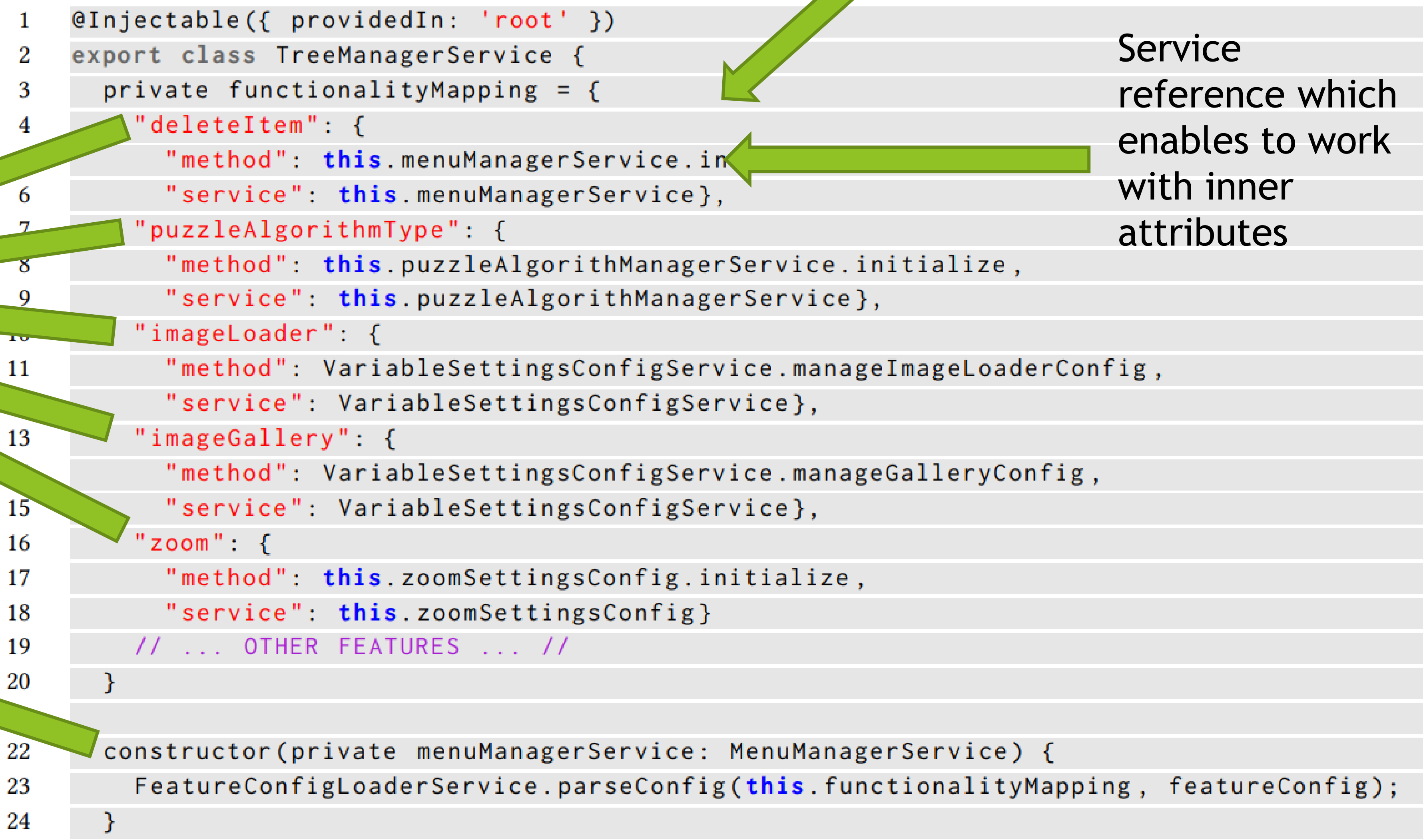
```
//${}|[path]|number_block
```

-proposed annotation to reduce code duplication

PROPOSED WHEN ASPECTS SHOULD NOT BE INCLUDED IN RESULTING DERIVATIONS

STEP 1: Separation of “aspects” from code

1. SEPARATION OF FEATURE MANAGEMENT



```
@Injectable({ providedIn: 'root' })
export class TreeManagerService {
  private functionalityMapping = {
    "deleteItem": {
      "method": this.menuManagerService.in
      "service": this.menuManagerService},
    "puzzleAlgorithmType": {
      "method": this.puzzleAlgorithManagerService.initialize,
      "service": this.puzzleAlgorithManagerService},
    "imageLoader": {
      "method": VariableSettingsConfigService.manageImageLoaderConfig,
      "service": VariableSettingsConfigService},
    "imageGallery": {
      "method": VariableSettingsConfigService.manageGalleryConfig,
      "service": VariableSettingsConfigService},
    "zoom": {
      "method": this.zoomSettingsConfig.initialize,
      "service": this.zoomSettingsConfig}
    // ... OTHER FEATURES ... //
  }

  constructor(private menuManagerService: MenuManagerService) {
    FeatureConfigLoaderService.parseConfig(this.functionalityMapping, featureConfig);
  }
```

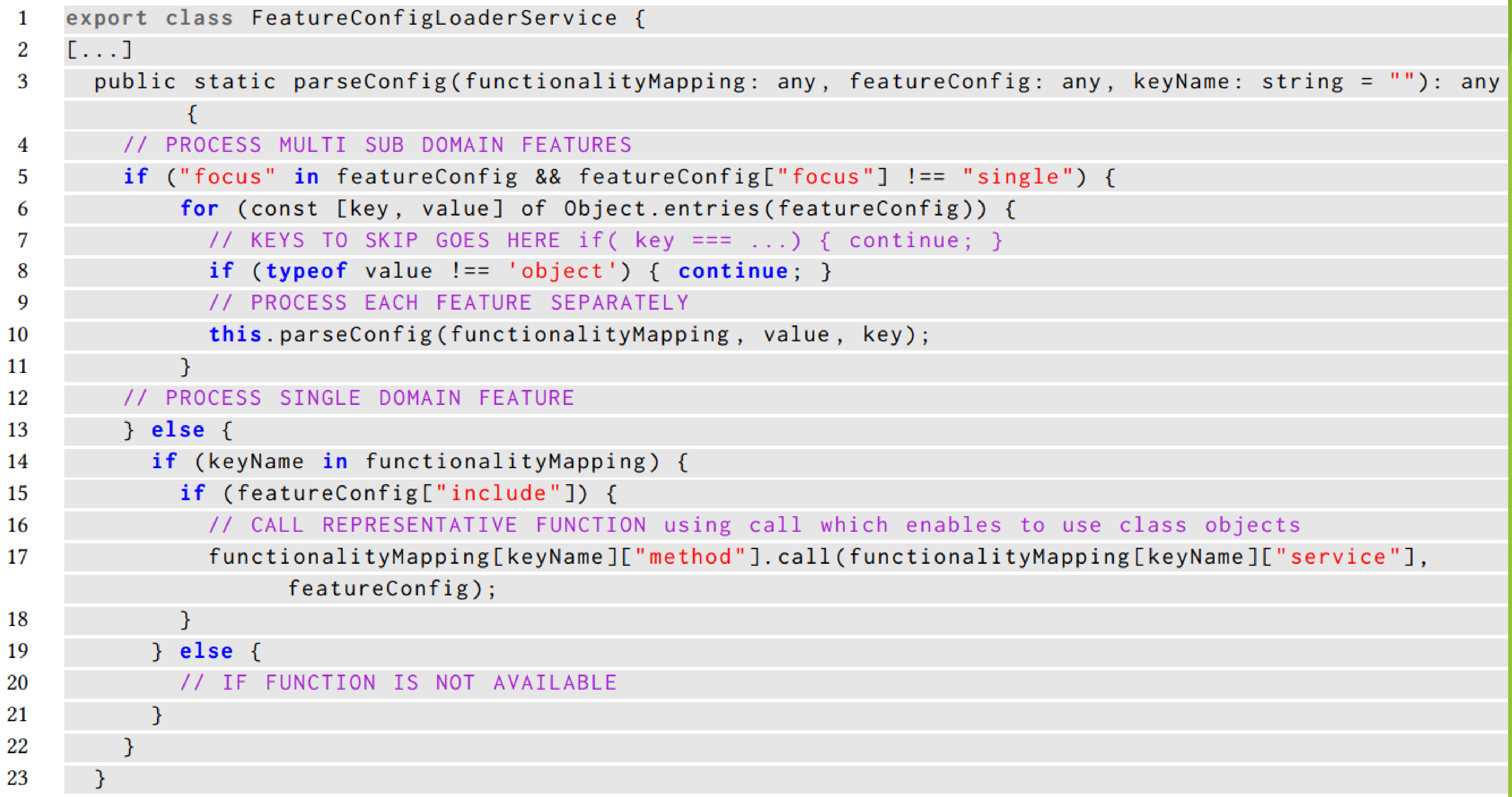
Initial method to apply aspect for given feature

Service reference which enables to work with inner attributes

FEATURES

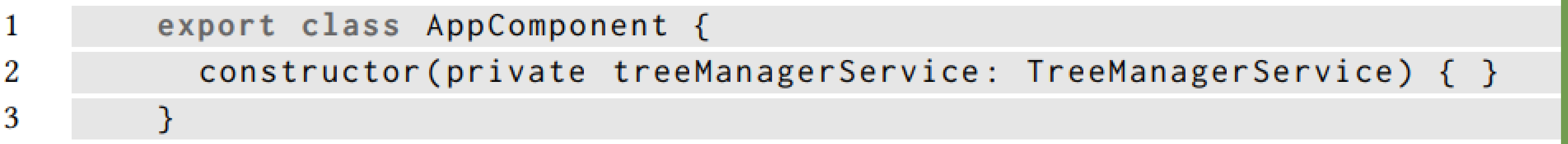
Loads values from configuration file

2. LOADING VARIABLES WHICH REPRESENT FEATURES FROM CONFIGURATION FILE



```
export class FeatureConfigLoaderService {
[...]
  public static parseConfig(functionalityMapping: any, featureConfig: any, keyName: string = ""): any
        {
    // PROCESS MULTI SUB DOMAIN FEATURES
    if ("focus" in featureConfig && featureConfig["focus"] !== "single") {
        for (const [key, value] of Object.entries(featureConfig)) {
          // KEYS TO SKIP GOES HERE if( key === ...) { continue; }
          if (typeof value !== 'object') { continue; }
          // PROCESS EACH FEATURE SEPARATELY
          this.parseConfig(functionalityMapping, value, key);
        }
    // PROCESS SINGLE DOMAIN FEATURE
    } else {
      if (keyName in functionalityMapping) {
        if (featureConfig["include"]) {
          // CALL REPRESENTATIVE FUNCTION using call which enables to use class objects
          functionalityMapping[keyName]["method"].call(functionalityMapping[keyName]["service"],
                featureConfig);
        }
      } else {
        // IF FUNCTION IS NOT AVAILABLE
      }
    }
  }
```

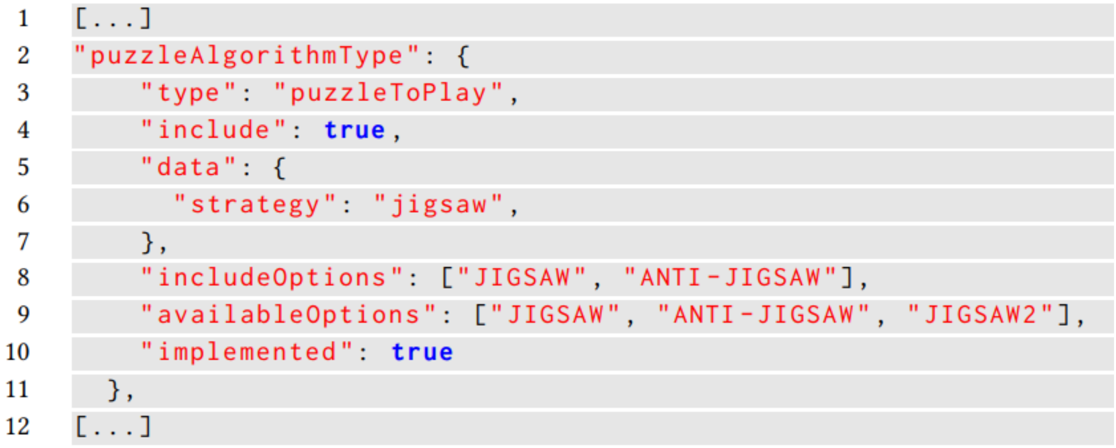
3. CONNECTS FEATURE MANAGEMENT WITH THE BUSINESS LOGIN ONLY IN ONE PLACE



```
export class AppComponent {
  constructor(private treeManagerService: TreeManagerService) { }
}
```

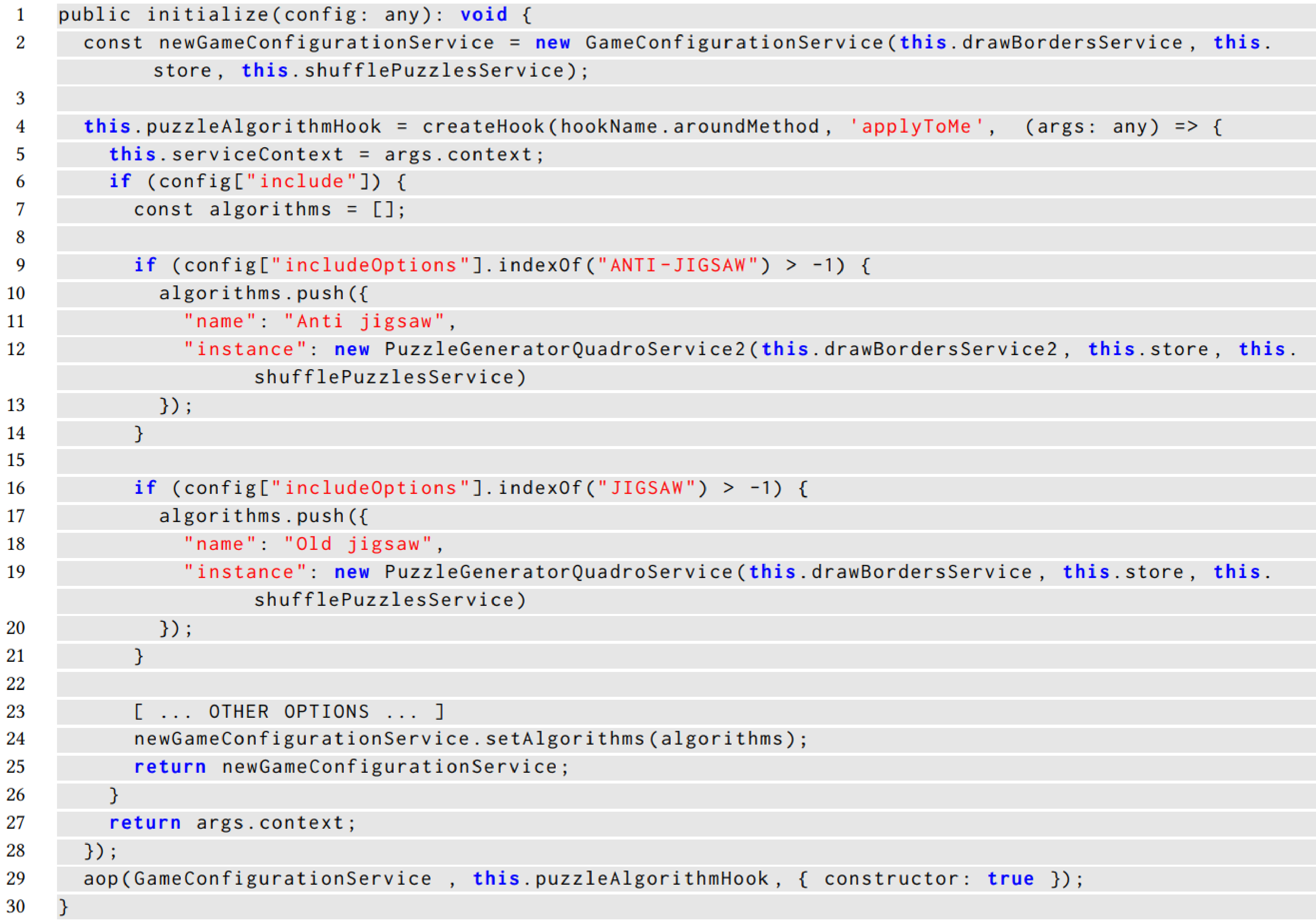
Aspect example - to-aop library

1. CONFIGURATION FILE



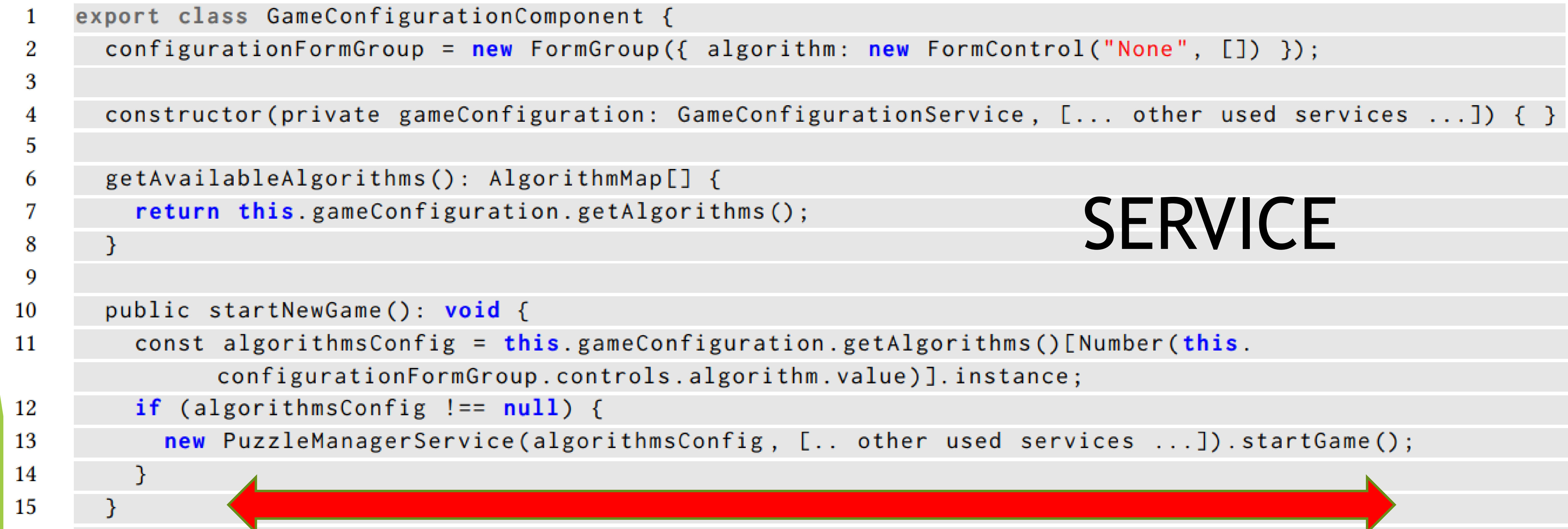
```
[...]
"puzzleAlgorithmType": {
    "type": "puzzleToPlay",
    "include": true,
    "data": {
      "strategy": "jigsaw",
    },
    "includeOptions": ["JIGSAW", "ANTI-JIGSAW"],
    "availableOptions": ["JIGSAW", "ANTI-JIGSAW", "JIGSAW2"],
    "implemented": true
  },
[...]
```

2. ASPECT DEFINITION



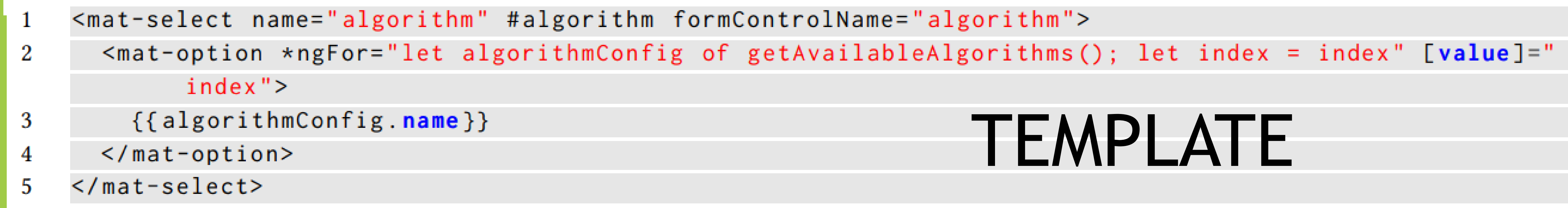
```
public initialize(config: any): void {
  const newGameConfigurationService = new GameConfigurationService(this.drawBordersService, this.
        store, this.shufflePuzzlesService);

  this.puzzleAlgorithmHook = createHook(hookName.aroundMethod, 'applyToMe', (args: any) => {
    this.serviceContext = args.context;
    if (config["include"]) {
      const algorithms = [];

      if (config["includeOptions"].indexOf("ANTI-JIGSAW") > -1) {
        algorithms.push({
          "name": "Anti jigsaw",
          "instance": new PuzzleGeneratorQuadroService2(this.drawBordersService2, this.store, this.
                shufflePuzzlesService)
        });
      }

      if (config["includeOptions"].indexOf("JIGSAW") > -1) {
        algorithms.push({
          "name": "Old jigsaw",
          "instance": new PuzzleGeneratorQuadroService(this.drawBordersService, this.store, this.
                shufflePuzzlesService)
        });
      }

      [ ... OTHER OPTIONS ... ]
      newGameConfigurationService.setAlgorithms(algorithms);
      return newGameConfigurationService;
    }
    return args.context;
  });
  aop(GameConfigurationService , this.puzzleAlgorithmHook, { constructor: true });
}
```

3. NATIVE SERVICE AND TEMPLATE



```
export class GameConfigurationComponent {
  configurationFormGroup = new FormGroup({ algorithm: new FormControl("None", []) });

  constructor(private gameConfiguration: GameConfigurationService, [... other used services ...]) { }

  getAvailableAlgorithms(): AlgorithmMap[] {
    return this.gameConfiguration.getAlgorithms();
  }

  public startNewGame(): void {
    const algorithmsConfig = this.gameConfiguration.getAlgorithms()[Number(this.
         configurationFormGroup.controls.algorithm.value)].instance;
    if (algorithmsConfig !== null) {
      new PuzzleManagerService(algorithmsConfig, [.. other used services ...]).startGame();
    }
  }
```

SERVICE

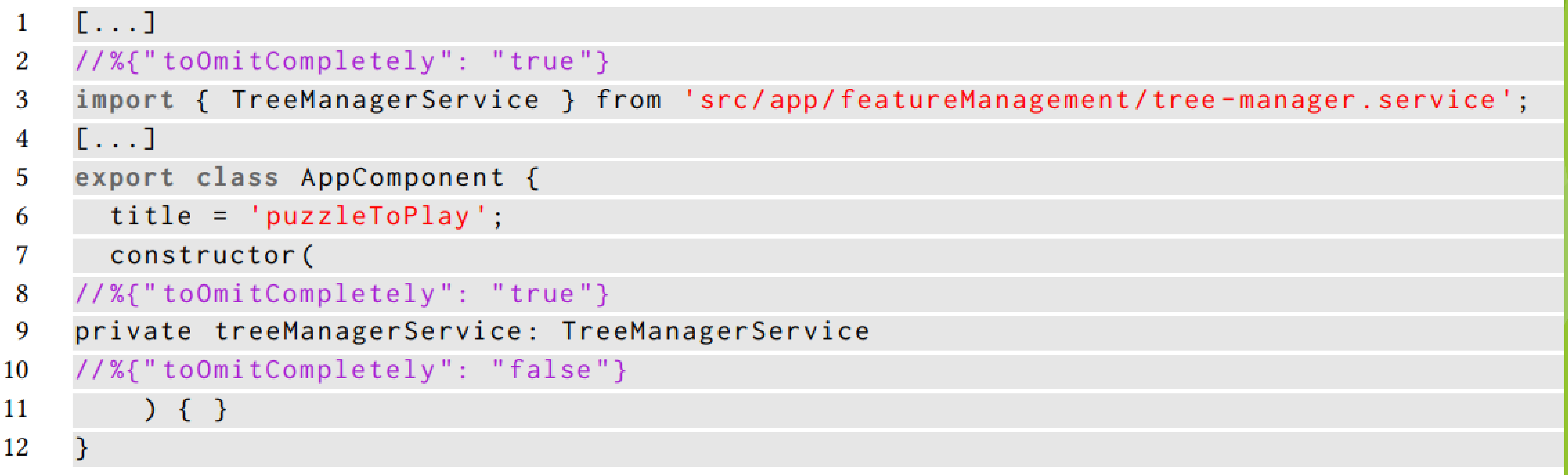


```
<mat-select name="algorithm" #algorithm formControlName="algorithm">
  <mat-option *ngFor="let algorithmConfig of getAvailableAlgorithms(); let index = index" [value]="
       index">
    {{algorithmConfig.name}}
  </mat-option>
</mat-select>
```

TEMPLATE

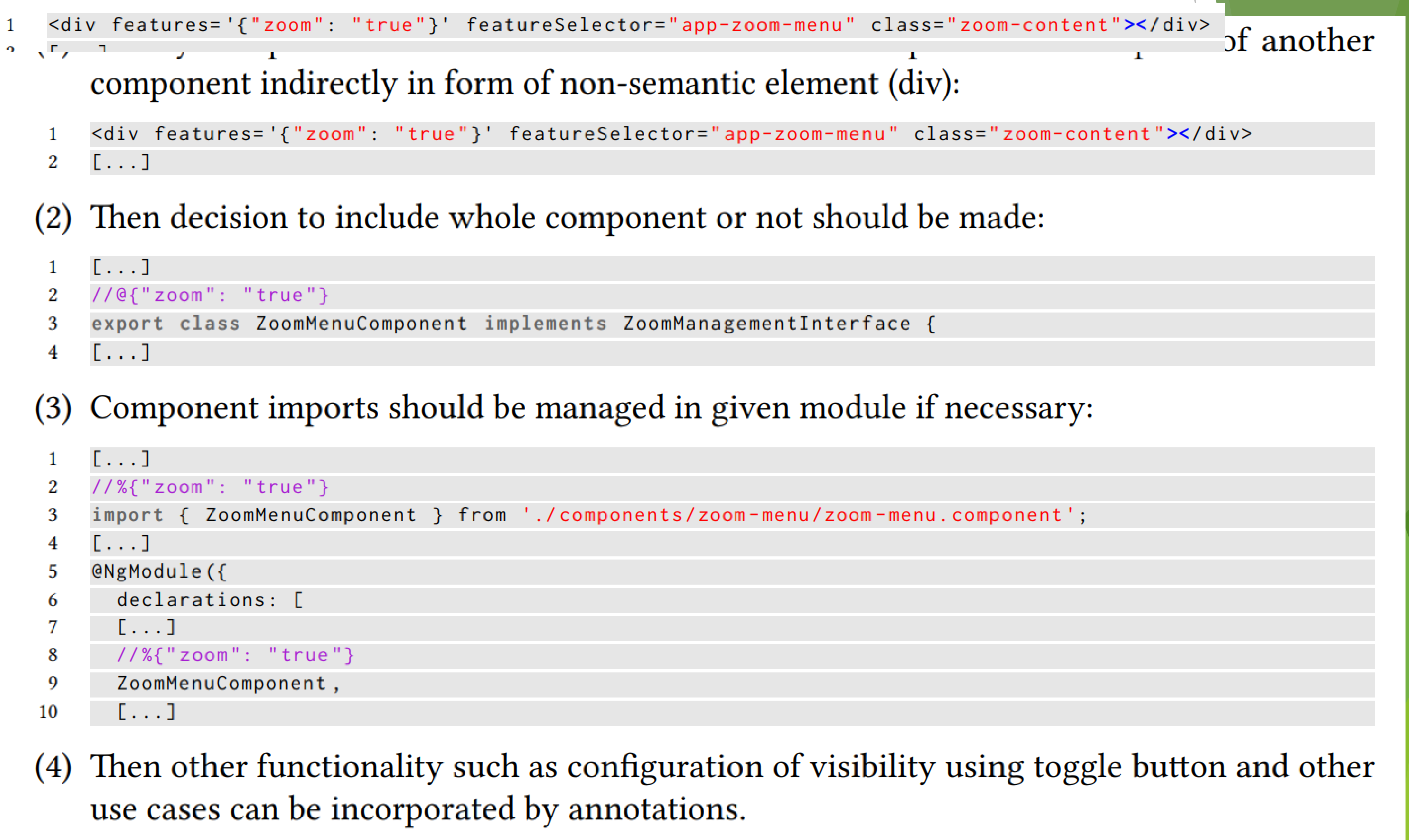
STEP 2: Creating and annotating functionality

REMOVING ONLY ONE DEPENDENCY ON ASPECTS FROM CONSTRUCTOR



```
[...]
//%{"toOmitCompletely": "true"}
import { TreeManagerService } from 'src/app/featureManagement/tree-manager.service';
[...]
export class AppComponent {
  title = 'puzzleToPlay';
  constructor(
//%{"toOmitCompletely": "true"}
private treeManagerService: TreeManagerService
//%{"toOmitCompletely": "false"}
    ) { }
}
```

Example: using expressions inside template



```
<div features='{"zoom": "true"}' featureSelector="app-zoom-menu" class="zoom-content"></div>
```

of another component indirectly in form of non-semantic element (div):

```
<div features='{"zoom": "true"}' featureSelector="app-zoom-menu" class="zoom-content"></div>
[...]
```

(2) Then decision to include whole component or not should be made:

```
[...]
//@{"zoom": "true"}
export class ZoomMenuComponent implements ZoomManagementInterface {
[...]
```

(3) Component imports should be managed in given module if necessary:

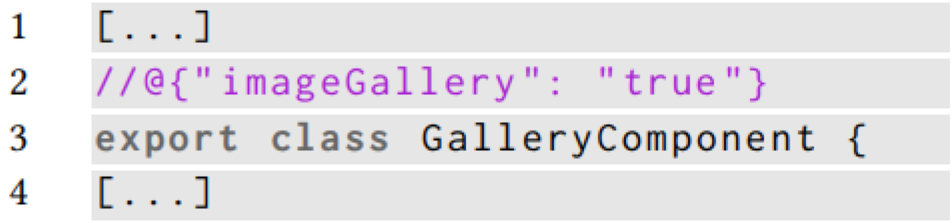
```
[...]
//%{"zoom": "true"}
import { ZoomMenuComponent } from './components/zoom-menu/zoom-menu.component';
[...]
@NgModule({
  declarations: [
  [...]
  //%{"zoom": "true"}
  ZoomMenuComponent,
  [...]
```

(4) Then other functionality such as configuration of visibility using toggle button and other use cases can be incorporated by annotations.

Example: Making gallery variable

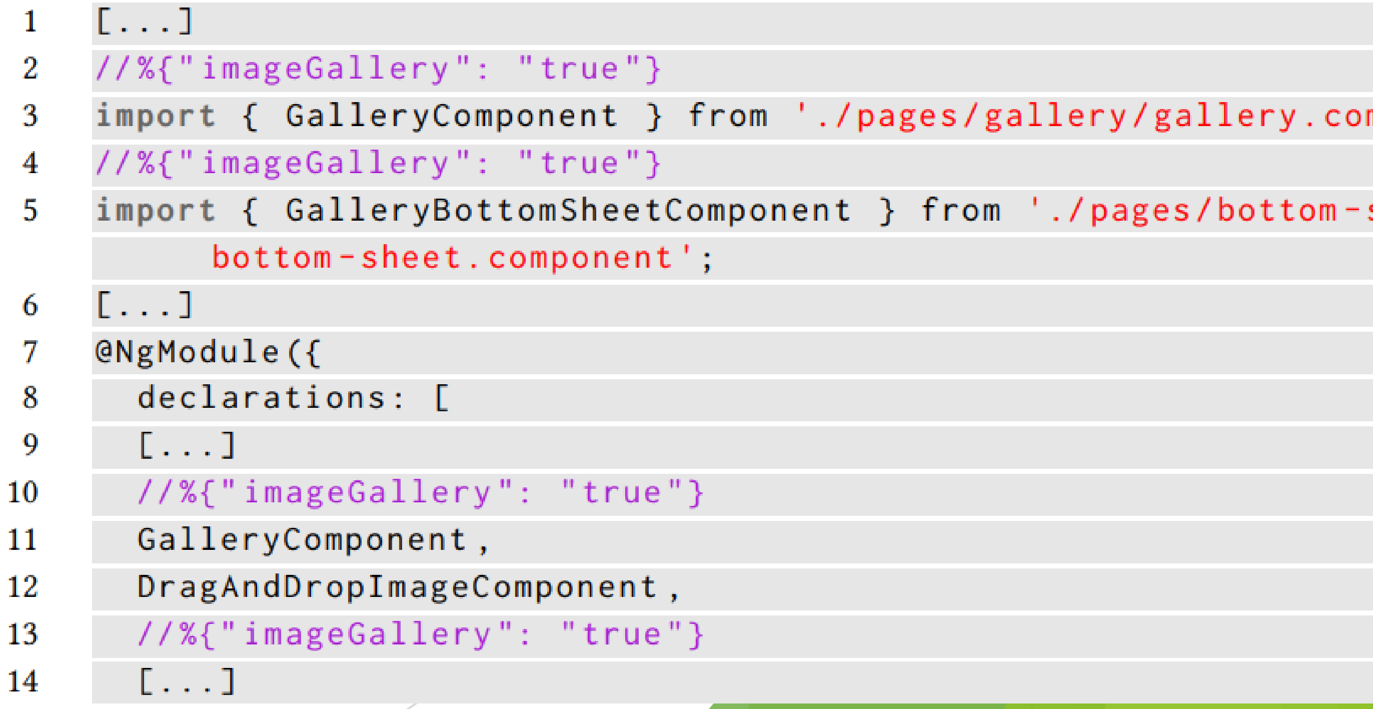
Example: Gallery should be variable (condition: natively is accessed by routing)

1. Annotate entire class for future exclusion



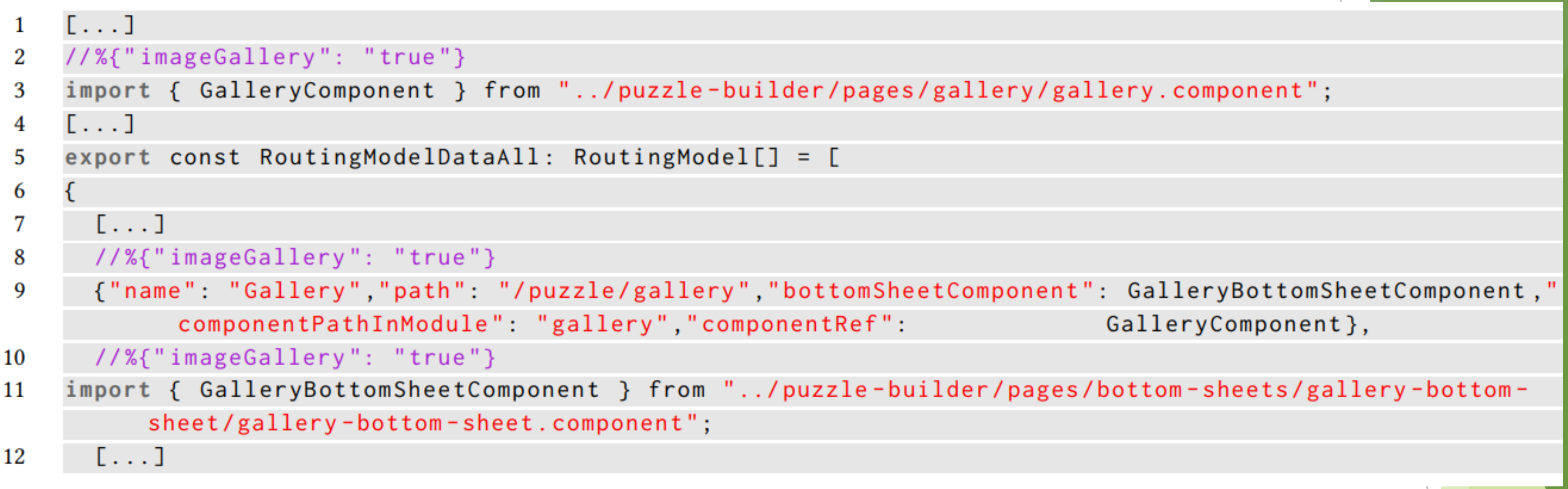
```
[...]
//@{"imageGallery": "true"}
export class GalleryComponent {
[...]
```

2. Annotate gallery imports for future exclusion



```
[...]
//%{"imageGallery": "true"}
import { GalleryComponent } from './pages/gallery/gallery.co
//%{"imageGallery": "true"}
import { GalleryBottomSheetComponent } from './pages/bottom-s
    bottom-sheet.component';
[...]
@NgModule({
  declarations: [
  [...]
  //%{"imageGallery": "true"}
  GalleryComponent,
  DragAndDropImageComponent,
  //%{"imageGallery": "true"}
  [...]
```

3. Annotate mock data, only those which belongs to the gallery



```
[...]
//%{"imageGallery": "true"}
import { GalleryComponent } from "../puzzle-builder/pages/gallery/gallery.component";
[...]
export const RoutingModelDataAll: RoutingModel[] = [
{
  [...]
  //%{"imageGallery": "true"}
  {"name": "Gallery","path": "/puzzle/gallery","bottomSheetComponent": GalleryBottomSheetComponent,"
        componentPathInModule": "gallery","componentRef":              GalleryComponent},
  //%{"imageGallery": "true"}
import { GalleryBottomSheetComponent } from "../puzzle-builder/pages/bottom-sheets/gallery-bottom-
      sheet/gallery-bottom-sheet.component";
  [...]
```

STEP 3: Product derivation

-starting derivator

Evaluating variability and commonality

our adaptation to make *assumptions* on partial components

$$SSC = \frac{|Cc|}{|Cc|+|Cv|} = \frac{3}{3+6} = 0{,}3333$$

Structure similarity coefficient

$$SVC = \frac{|Cv|}{|Cv|+|Cc|} = \frac{6}{6+3} = 0{,}6666$$

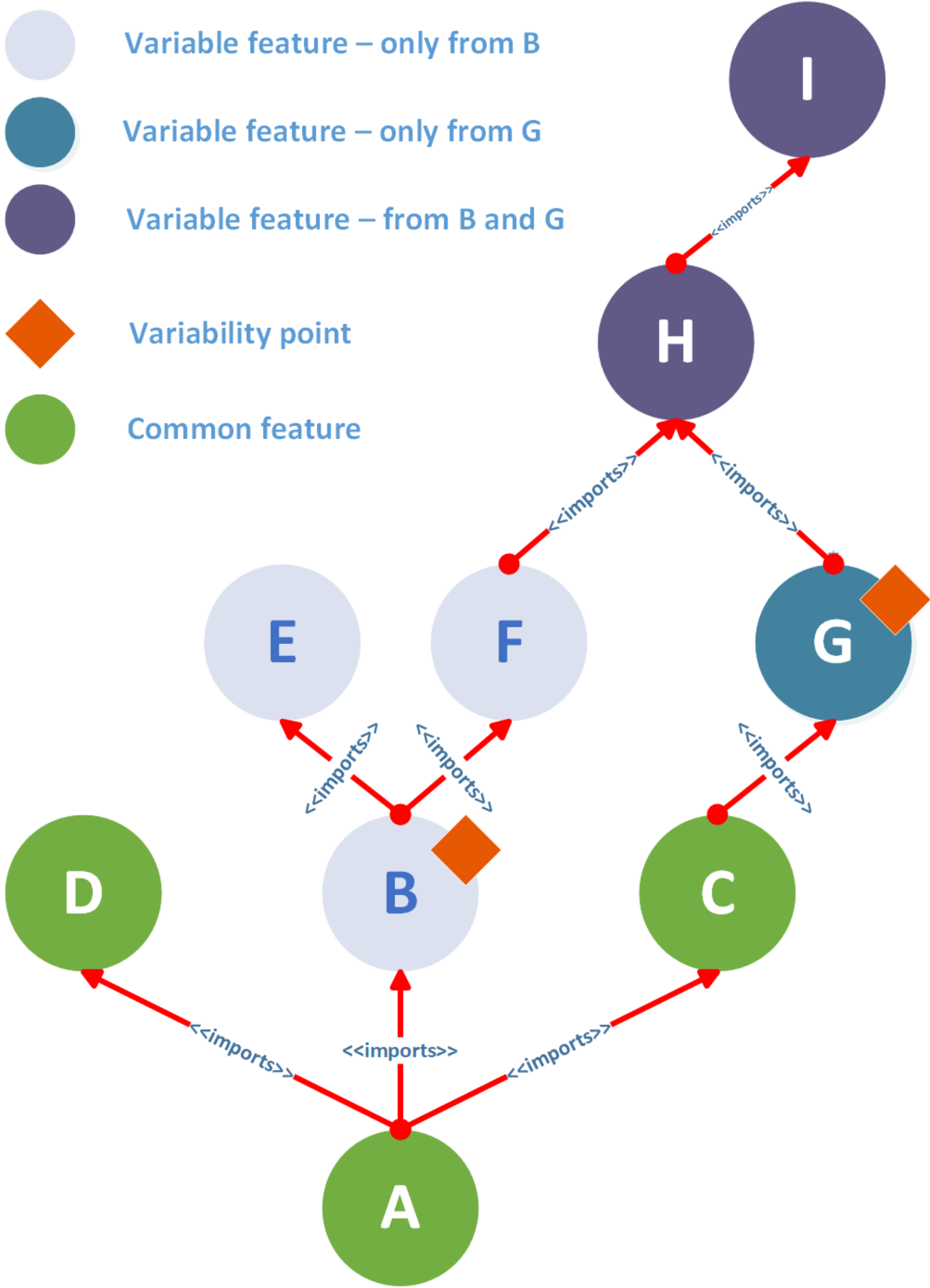
Structure variability coefficient

Tao Zhang, Lei Deng, Jian Wu, Qiaoming Zhou, and Chunyan Ma. 2008.
Some Metrics for Accessing Quality of Product Line Architecture.
In 2008 International Conference on Computer Science
and Software Engineering. IEEE, Wuhan, China, 500–503.
https://doi.org/10.1109/CSSE.2008.500

SSC = 1 - SVC

For components

Mutually conditional



Variable feature – only from B
Variable feature – only from G
Variable feature – from B and G
Variability point
Common feature
I
H
E
F
G
D
B
C
A
<<imports>>

Measuring Reuse Rate

$$CRR = \frac{\Sigma_i\ Ex|M_i|}{|M|}$$

Component reuse rate

1 if component is included in given member i otherwise 0 (interior)

(If component is interior then $Ex|M_i|$ = 1 otherwise 0)

Number of all members of SPL

Usable when all ARCHITECTURES SHOULD BE DERIVED

CRR for common components will be 100%
- in all derivations (architectures)

The higher CRR of the component,
the more important for SPL is
(for reuse)

$$RBR = \frac{\Sigma_k\ \mathrm{Cost}\ C_k}{\Sigma_j\ \mathrm{Cost}\ C}$$

Reuse benefit rate

Quality of given product line member k

Quality of all components in SPL

The higher RBR, the more reusable SPL is (the more members has)

Measuring variability

DEPENDANCE RELATIONS

CONSTRAINTS

Value of one variability
point affects another

STRONG COUPLED
VARIABILITY POINTS

STRONG COUPLING

More difficult to bind variability point in PLA

$$SCC = 1 - \frac{|IVP|}{|VP|}$$

Strong coupling coefficient

Independent variability points
(*no dependence relation with others*
– *no value of any variability point affects another one*)

Number of variability points

Less difficulty to bind variability point in PLA

WEAK COUPLING

More difficult variability design of PLA

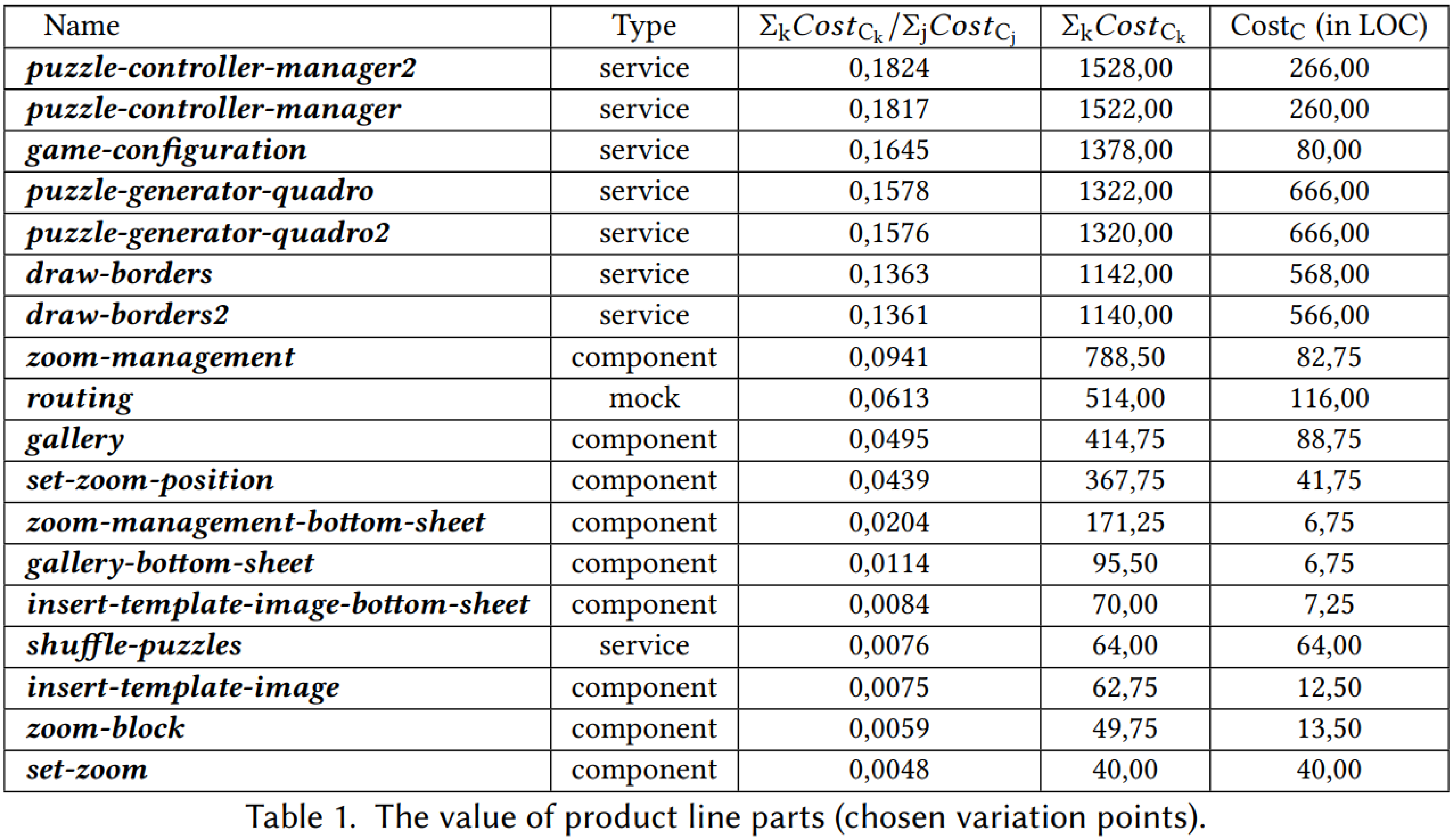
$$WCC = \frac{|CVP|}{|VP|}$$

Weak coupling coefficient

Weak coupling variability points
(*2 or more variability points controlling guard condition*
of some variability point(s), components,…)

Number of variability points

Tao Zhang, Lei Deng, Jian Wu, Qiaoming Zhou, and Chunyan Ma. 2008.
Some Metrics for Accessing Quality of Product Line Architecture.
In 2008 International Conference on Computer Science
and Software Engineering. IEEE, Wuhan, China, 500–503.
https://doi.org/10.1109/CSSE.2008.500

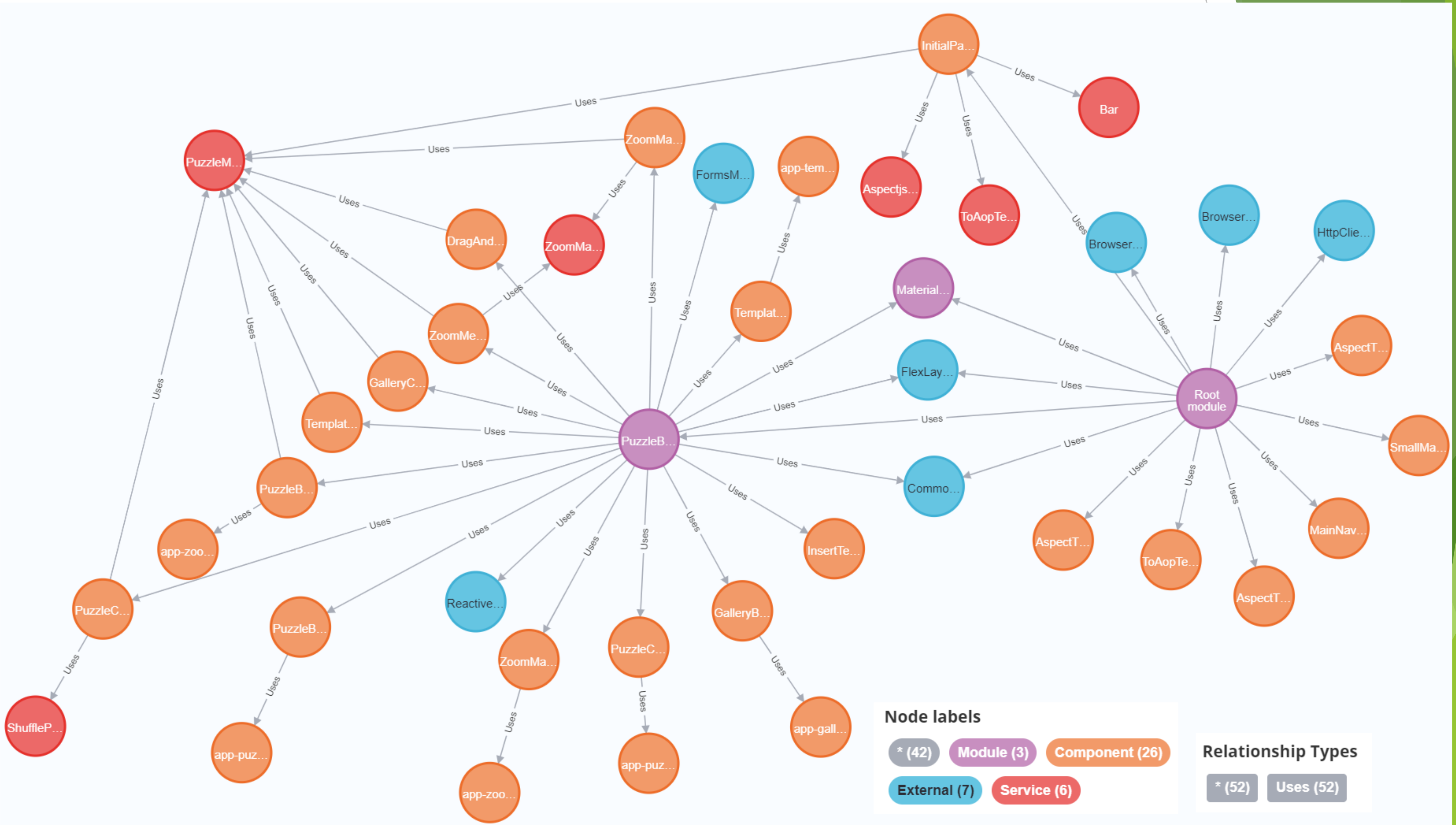


Name	Type	$\Sigma_k Cost_{C_k} / \Sigma_j Cost_{C_j}$	$\Sigma_k Cost_{C_k}$	$Cost_C$ (in LOC)
puzzle-controller-manager2	service	0,1824	1528,00	266,00
puzzle-controller-manager	service	0,1817	1522,00	260,00
game-configuration	service	0,1645	1378,00	80,00
puzzle-generator-quadro	service	0,1578	1322,00	666,00
puzzle-generator-quadro2	service	0,1576	1320,00	666,00
draw-borders	service	0,1363	1142,00	568,00
draw-borders2	service	0,1361	1140,00	566,00
zoom-management	component	0,0941	788,50	82,75
routing	mock	0,0613	514,00	116,00
gallery	component	0,0495	414,75	88,75
set-zoom-position	component	0,0439	367,75	41,75
zoom-management-bottom-sheet	component	0,0204	171,25	6,75
gallery-bottom-sheet	component	0,0114	95,50	6,75
insert-template-image-bottom-sheet	component	0,0084	70,00	7,25
shuffle-puzzles	service	0,0076	64,00	64,00
insert-template-image	component	0,0075	62,75	12,50
zoom-block	component	0,0059	49,75	13,50
set-zoom	component	0,0048	40,00	40,00

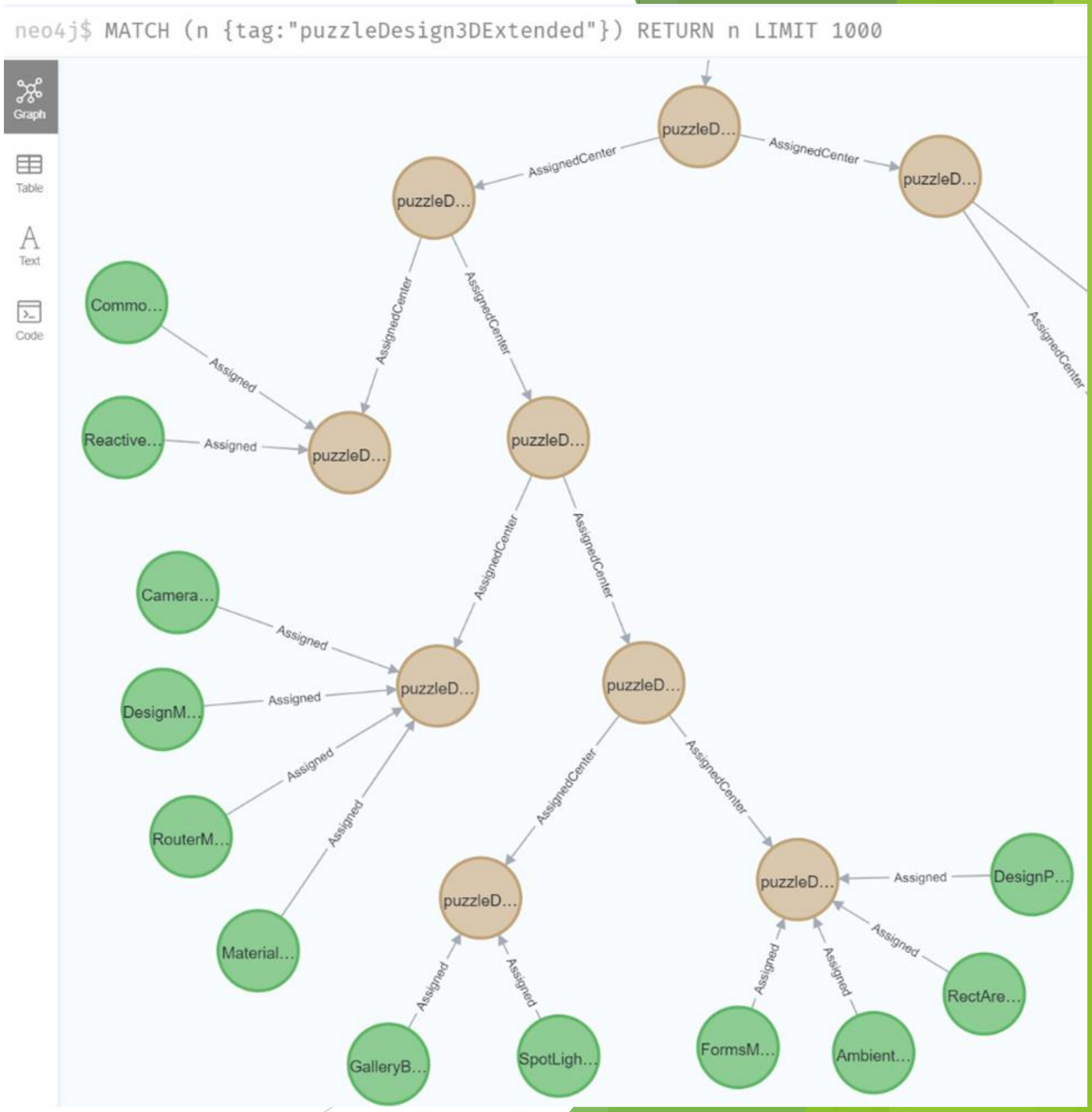
Table 1. The value of product line parts (chosen variation points).

Where The Cost of all components in SPL = Σ_j Cost C = 8378,25

Visualization - Puzzle to play - original data



Results of:
1. Graph merging
2. Hierarchical clustering

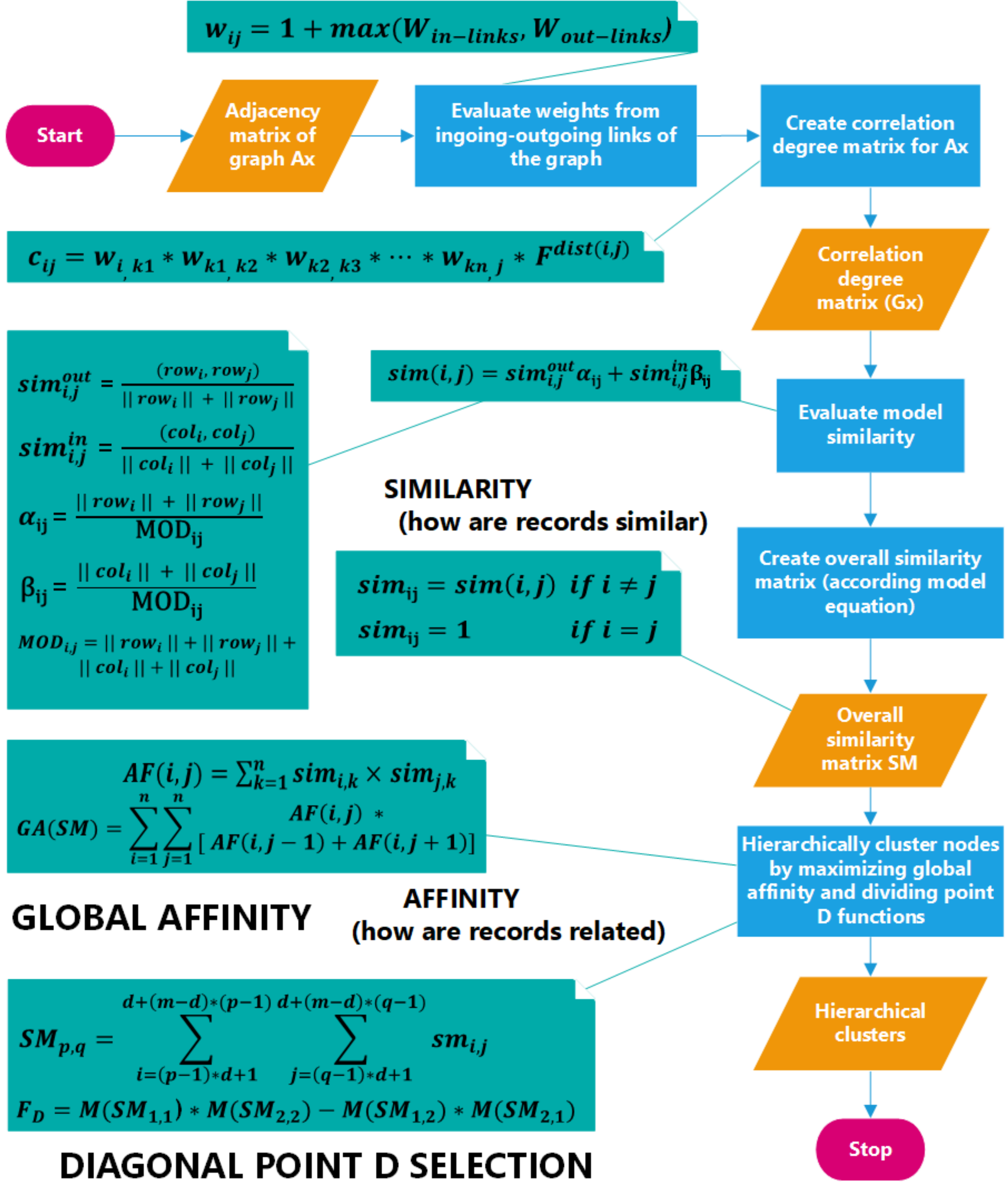


```
neo4j$ MATCH (n {tag:"puzzleDesign3DExtended"}) RETURN n LIMIT 1000
```

Matrix-based hierarchical clustering

Based on ingoing and outgoing connections/links

- HOU, Jingyu, Yanchun ZHANG a Jinli CAO, 2003. Web Page Clustering: A Hyperlink-Based Similarity and Matrix-Based Hierarchical Algorithms. V: Xiaofang ZHOU, Maria E. ORLOWSKA a Yanchun ZHANG, ed. Web Technologies and Applications [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, Lecture Notes in Computer Science, s. 201–212 [cit. 3.12.2022]. ISBN 978-3-540-02354-8. Dostupné na: doi:10.1007/3-540-36901-5_22



Start

Adjacency matrix of graph Ax

Evaluate weights from ingoing-outgoing links of the graph

$$w_{ij} = 1 + max(W_{in-links}, W_{out-links})$$

Create correlation degree matrix for Ax

$$c_{ij} = w_{i\_k1} * w_{k1\_k2} * w_{k2\_k3} * \cdots * w_{kn\_j} * F^{dist(i,j)}$$

Correlation degree matrix (Gx)

Evaluate model similarity

$$sim(i,j) = sim^{out}_{i,j}\alpha_{ij} + sim^{in}_{i,j}\beta_{ij}$$

$$sim^{out}_{i,j} = \frac{(row_i, row_j)}{\| row_i \| + \| row_j \|}$$

$$sim^{in}_{i,j} = \frac{(col_i, col_j)}{\| col_i \| + \| col_j \|}$$

$$\alpha_{ij} = \frac{\| row_i \| + \| row_j \|}{\mathrm{MOD}_{ij}}$$

$$\beta_{ij} = \frac{\| col_i \| + \| col_j \|}{\mathrm{MOD}_{ij}}$$

$$MOD_{i,j} = \| row_i \| + \| row_j \| + \| col_i \| + \| col_j \|$$

SIMILARITY
(how are records similar)

Create overall similarity matrix (according model equation)

$$sim_{ij} = sim(i,j) \quad if\ i \neq j$$
$$sim_{ij} = 1 \quad if\ i = j$$

Overall similarity matrix SM

Hierarchically cluster nodes by maximizing global affinity and dividing point D functions

$$AF(i,j) = \sum_{k=1}^{n} sim_{i,k} \times sim_{j,k}$$

$$GA(SM) = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{AF(i,j)\ *}{[AF(i,j-1) + AF(i,j+1)]}$$

GLOBAL AFFINITY

AFFINITY
(how are records related)

$$SM_{p,q} = \sum_{i=(p-1)*d+1}^{d+(m-d)*(p-1)} \sum_{j=(q-1)*d+1}^{d+(m-d)*(q-1)} sm_{i,j}$$

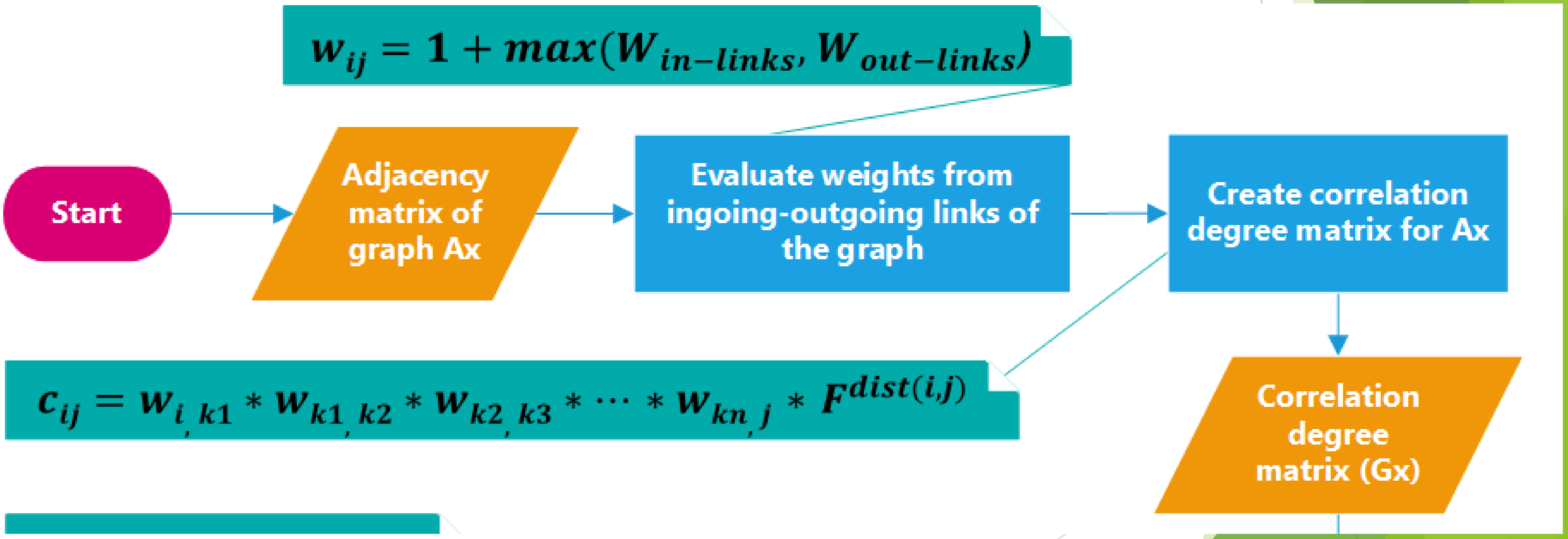
$$F_D = M(SM_{1,1}) * M(SM_{2,2}) - M(SM_{1,2}) * M(SM_{2,1})$$

DIAGONAL POINT D SELECTION

Hierarchical clusters

Stop

Initialization based on ingoing and outgoing links



$$w_{ij} = 1 + max(W_{in-links}, W_{out-links})$$

Start

Adjacency matrix of graph Ax

Evaluate weights from ingoing-outgoing links of the graph

Create correlation degree matrix for Ax

Correlation degree matrix (Gx)

$$c_{ij} = w_{i, k1} * w_{k1, k2} * w_{k2, k3} * \cdots * w_{kn, j} * F^{dist(i,j)}$$

Evaluating model similarity

$$sim_{i,j}^{out} = \frac{(row_i, row_j)}{\| row_i \| + \| row_j \|}$$

$$sim_{i,j}^{in} = \frac{(col_i, col_j)}{\| col_i \| + \| col_j \|}$$

$$\alpha_{ij} = \frac{\| row_i \| + \| row_j \|}{\mathrm{MOD}_{ij}}$$

$$\beta_{ij} = \frac{\| col_i \| + \| col_j \|}{\mathrm{MOD}_{ij}}$$

$$MOD_{i,j} = \| row_i \| + \| row_j \| + \| col_i \| + \| col_j \|$$

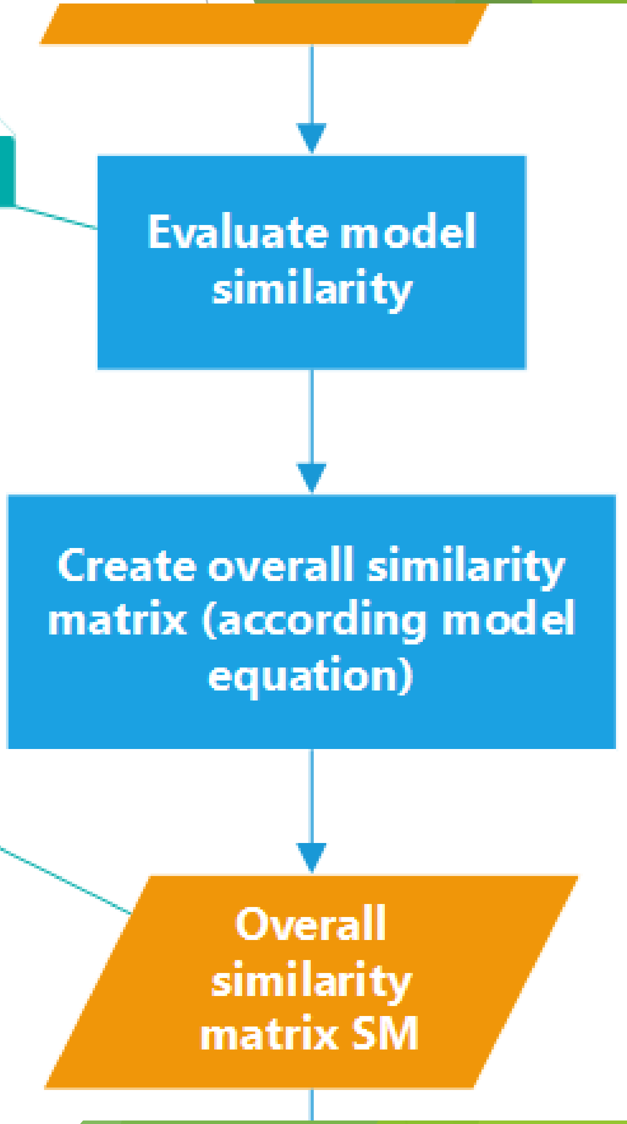
$$sim(i,j) = sim_{i,j}^{out} \alpha_{ij} + sim_{i,j}^{in} \beta_{ij}$$

SIMILARITY

(how are records similar)

$$\begin{array}{ll} sim_{ij} = sim(i,j) & if\ i \neq j \\ sim_{ij} = 1 & if\ i = j \end{array}$$

$$AF(i,j) = \sum_{k=1}^{n} sim_{i,k} \times sim_{j,k}$$



Evaluate model similarity

Create overall similarity matrix (according model equation)

Overall similarity matrix SM

Hierarchical matrix-based clustering

$$AF(i,j) = \sum_{k=1}^{n} sim_{i,k} \times sim_{j,k}$$

$$GA(SM) = \sum_{i=1}^{n}\sum_{j=1}^{n} \begin{array}{c} AF(i,j) \ * \\ [\,AF(i,j-1) + AF(i,j+1)] \end{array}$$

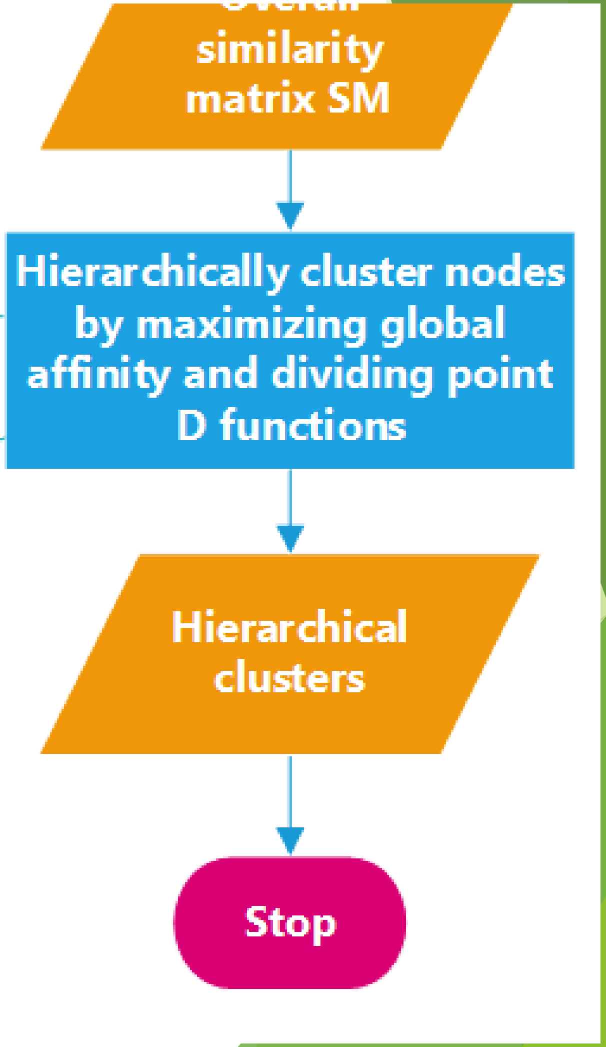
GLOBAL AFFINITY

AFFINITY
(how are records related)

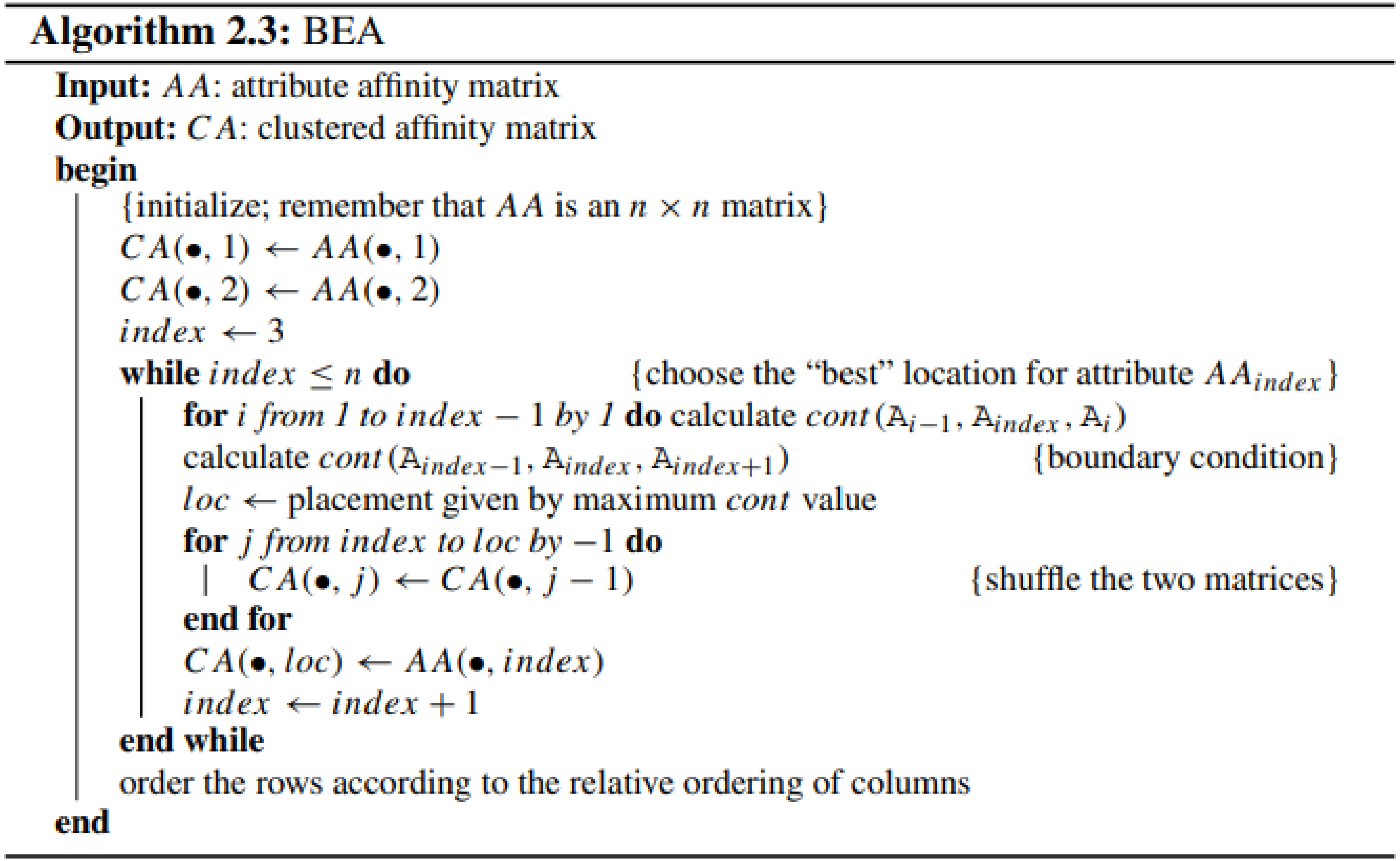
$$SM_{p,q} = \sum_{i=(p-1)*d+1}^{d+(m-d)*(p-1)} \ \sum_{j=(q-1)*d+1}^{d+(m-d)*(q-1)} sm_{i,j}$$

$$F_D = M(SM_{1,1}) * M(SM_{2,2}) - M(SM_{1,2}) * M(SM_{2,1})$$

DIAGONAL POINT D SELECTION


overall similarity matrix SM
Hierarchically cluster nodes by maximizing global affinity and dividing point D functions
Hierarchical clusters
Stop

Energy-bond algorithm



Algorithm 2.3: BEA

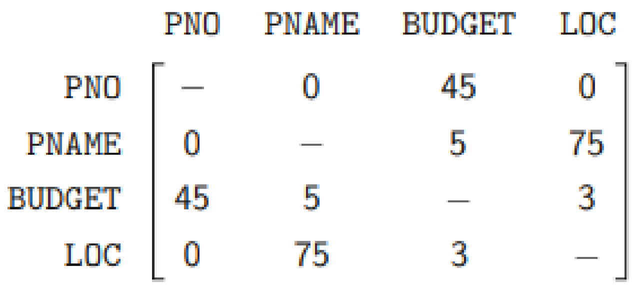
```
Input: $AA$: attribute affinity matrix
Output: $CA$: clustered affinity matrix
begin
    {initialize; remember that $AA$ is an $n \times n$ matrix}
    $CA(\bullet, 1) \leftarrow AA(\bullet, 1)$
    $CA(\bullet, 2) \leftarrow AA(\bullet, 2)$
    $index \leftarrow 3$
    while $index \leq n$ do          {choose the "best" location for attribute $AA_{index}$}
        for $i$ from 1 to $index - 1$ by 1 do calculate $cont(A_{i-1}, A_{index}, A_i)$
        calculate $cont(A_{index-1}, A_{index}, A_{index+1})$          {boundary condition}
        $loc \leftarrow$ placement given by maximum $cont$ value
        for $j$ from $index$ to $loc$ by $-1$ do
            $CA(\bullet, j) \leftarrow CA(\bullet, j-1)$          {shuffle the two matrices}
        end for
        $CA(\bullet, loc) \leftarrow AA(\bullet, index)$
        $index \leftarrow index + 1$
    end while
    order the rows according to the relative ordering of columns
end
```

M. T. Özsu, P. Valduriez, Principles of Distributed Database Systems, Springer International Publishing, Cham, 2020.

Energy-bond algorithm

1. MAXIMIZATION

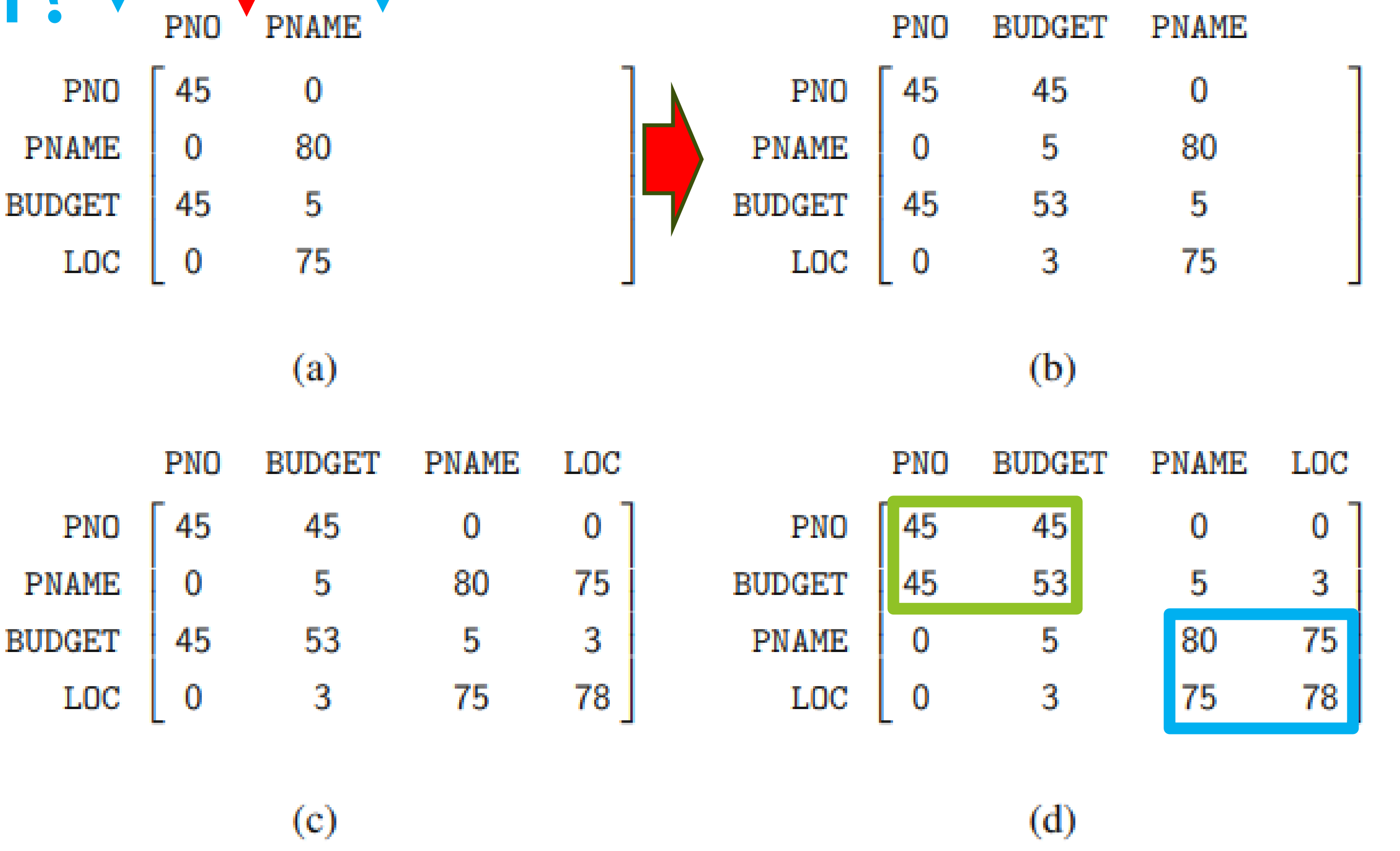
As unit test



$$\begin{array}{c|cccc} & \text{PNO} & \text{PNAME} & \text{BUDGET} & \text{LOC} \\ \hline \text{PNO} & - & 0 & 45 & 0 \\ \text{PNAME} & 0 & - & 5 & 75 \\ \text{BUDGET} & 45 & 5 & - & 3 \\ \text{LOC} & 0 & 75 & 3 & - \end{array}$$

Fig. 2.13 Attribute affinity matrix

BUDGET?



$$\begin{array}{c|cccc} & \text{PNO} & \text{PNAME} & & \\ \hline \text{PNO} & 45 & 0 & & \\ \text{PNAME} & 0 & 80 & & \\ \text{BUDGET} & 45 & 5 & & \\ \text{LOC} & 0 & 75 & & \end{array}$$

(a)

$$\begin{array}{c|cccc} & \text{PNO} & \text{BUDGET} & \text{PNAME} & \\ \hline \text{PNO} & 45 & 45 & 0 & \\ \text{PNAME} & 0 & 5 & 80 & \\ \text{BUDGET} & 45 & 53 & 5 & \\ \text{LOC} & 0 & 3 & 75 & \end{array}$$

(b)

$$\begin{array}{c|cccc} & \text{PNO} & \text{BUDGET} & \text{PNAME} & \text{LOC} \\ \hline \text{PNO} & 45 & 45 & 0 & 0 \\ \text{PNAME} & 0 & 5 & 80 & 75 \\ \text{BUDGET} & 45 & 53 & 5 & 3 \\ \text{LOC} & 0 & 3 & 75 & 78 \end{array}$$

(c)

$$\begin{array}{c|cccc} & \text{PNO} & \text{BUDGET} & \text{PNAME} & \text{LOC} \\ \hline \text{PNO} & 45 & 45 & 0 & 0 \\ \text{BUDGET} & 45 & 53 & 5 & 3 \\ \text{PNAME} & 0 & 5 & 80 & 75 \\ \text{LOC} & 0 & 3 & 75 & 78 \end{array}$$

(d)

Fig. 2.14 Calculation of the clustered affinity (CA) matrix

$$\begin{aligned} cont(\mathrm{A}_i, \mathrm{A}_k, \mathrm{A}_j) &= AM_{new} - AM_{old} \\ &= 2bond(\mathrm{A}_i, \mathrm{A}_k) + 2bond(\mathrm{A}_k, \mathrm{A}_j) - 2bond(\mathrm{A}_i, \mathrm{A}_j) \end{aligned}$$

$$bond(\mathrm{A}_x, \mathrm{A}_y) = \sum_{z=1}^{n} aff(\mathrm{A}_z, \mathrm{A}_x) aff(\mathrm{A}_z, \mathrm{A}_y)$$

For 0-3-1:

$$\begin{aligned} cont(\mathrm{A}_0, \mathrm{BUDGET}, \mathrm{PNO}) = {} & 2bond(\mathrm{A}_0, \mathrm{BUDGET}) + 2bond(\mathrm{BUDGET}, \mathrm{PNO}) \\ & - 2bond(\mathrm{A}_0, \mathrm{PNO}) \end{aligned}$$

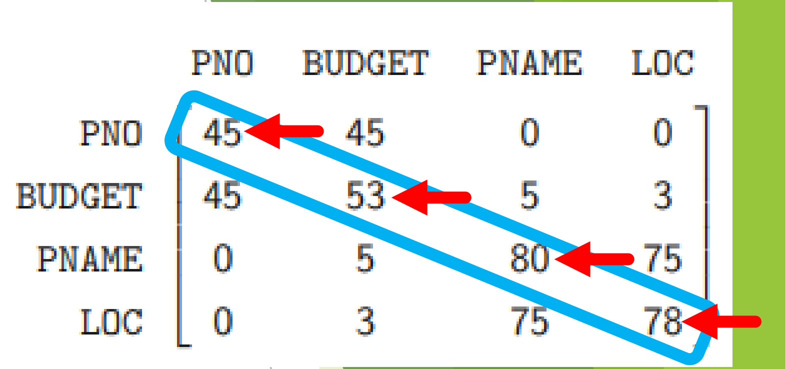
$$\begin{aligned} bond(\mathrm{A}_0, \mathrm{PNO}) &= bond(\mathrm{A}_0, \mathrm{BUDGET}) = 0 \\ bond(\mathrm{BUDGET}, \mathrm{PNO}) &= 45 * 45 + 5 * 0 + 53 * 45 + 3 * 0 = 4410 \end{aligned}$$

$$cont(\mathrm{A}_0, \mathrm{BUDGET}, \mathrm{PNO}) = 8820$$

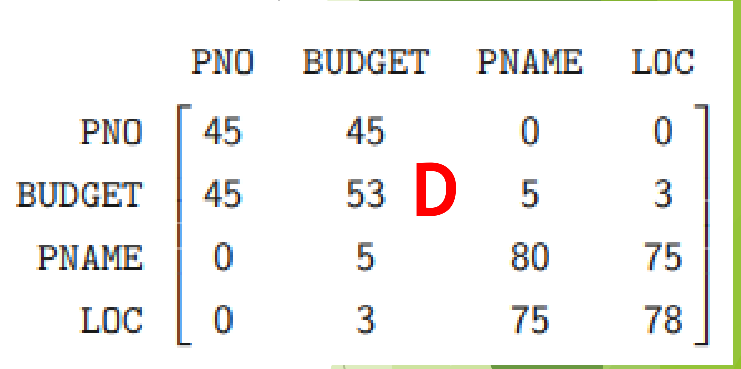
M. T. Özsu, P. Valduriez, Principles of Distributed Database Systems, Springer International Publishing, Cham, 2020.

Diagonal Point D selection

2. MAXIMIZATION



$$
\begin{array}{c|cccc}
 & PNO & BUDGET & PNAME & LOC \\
\hline
PNO & 45 & 45 & 0 & 0 \\
BUDGET & 45 & 53 & 5 & 3 \\
PNAME & 0 & 5 & 80 & 75 \\
LOC & 0 & 3 & 75 & 78
\end{array}
$$



$$
\begin{array}{c|cccc}
 & PNO & BUDGET & PNAME & LOC \\
\hline
PNO & 45 & 45 & 0 & 0 \\
BUDGET & 45 & 53 & 5 & 3 \\
PNAME & 0 & 5 & 80 & 75 \\
LOC & 0 & 3 & 75 & 78
\end{array}
$$

$$
SM = (sm_{i,j})_{m \times m} = \left[\begin{array}{c:c} SM_{1,1} & SM_{1,2} \\ \hdashline \multicolumn{2}{c}{D} \\ SM_{2,1} & SM_{2,2} \end{array}\right]_{m \times m}
$$

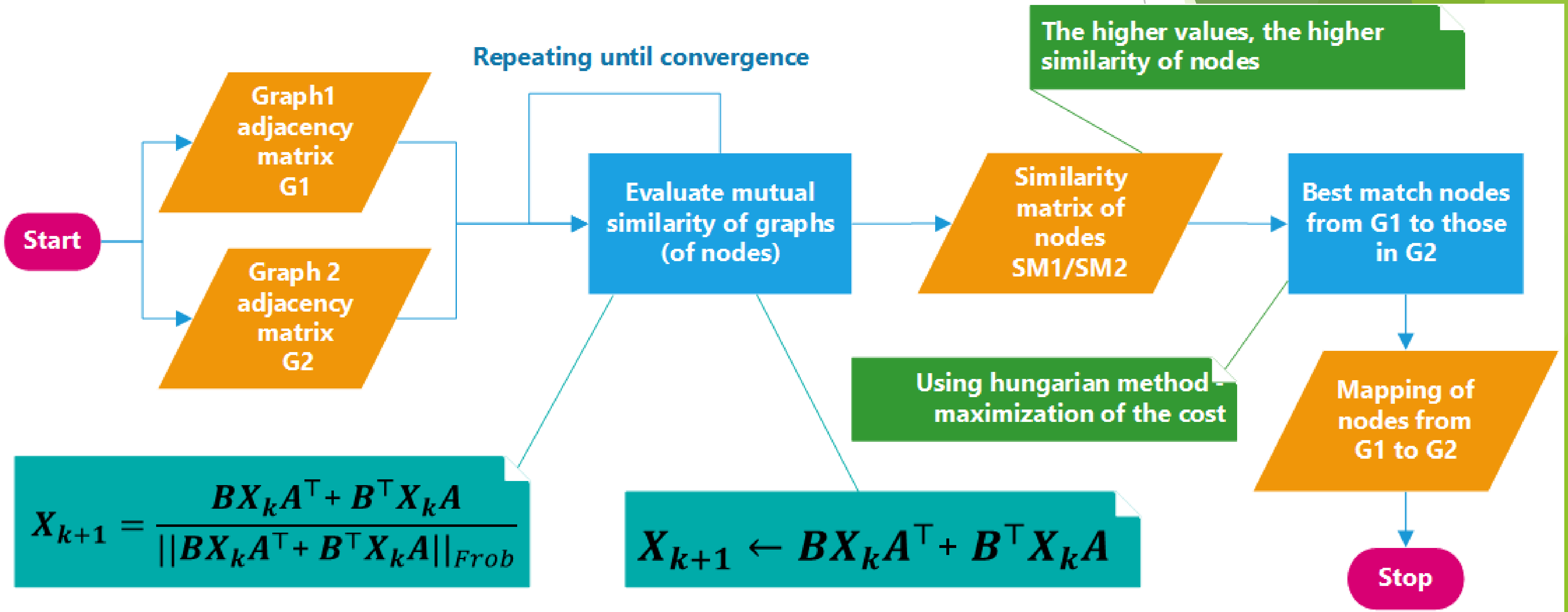
$$
F_D = M(SM_{1,1})^* \, M(SM_{2,2}) - M(SM_{1,2})^* \, M(SM_{2,1}).
$$

$$
M(SM_{p.q}) = \sum_{i=(p-1)*d+1}^{d+(m-d)*(p-1)} \sum_{j=(q-1)*d+1}^{d+(m-d)*(q-1)} sm_{i,j} , \qquad 1 \le p, q \le 2,
$$

J. Hou, Y. Zhang, J. Cao, Web page clustering: A hyperlink-based similarity and matrixbased hierarchical algorithms, in: Web Technologies and Applications, volume 2642, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 201–212.

Matrix-based graph matching based on node similarity

BLONDEL, Vincent D., Anahí GAJARDO, Maureen HEYMANS, Pierre SENELLART a Paul VAN DOOREN, 2004. A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching. SIAM Review [online]. 2004, roč. 46, č. 4, s. 647–666. ISSN 0036-1445, 1095-7200. Dostupné na: doi:10.1137/S0036144502415960


Repeating until convergence
The higher values, the higher similarity of nodes
Start
Graph1 adjacency matrix G1
Graph 2 adjacency matrix G2
Evaluate mutual similarity of graphs (of nodes)
Similarity matrix of nodes SM1/SM2
Best match nodes from G1 to those in G2
Using hungarian method - maximization of the cost
Mapping of nodes from G1 to G2
Stop

$$X_{k+1} = \frac{BX_kA^\top + B^\top X_kA}{||BX_kA^\top + B^\top X_kA||_{Frob}}$$

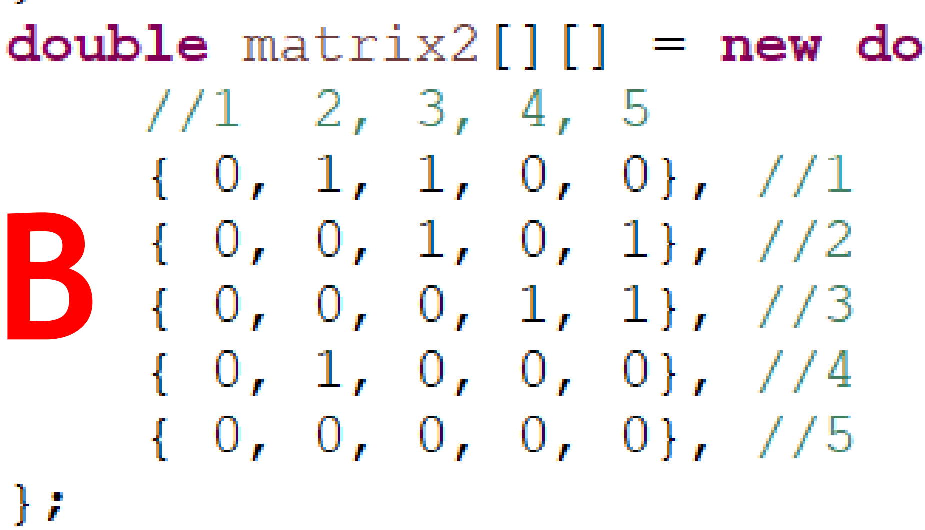
$$X_{k+1} \leftarrow BX_kA^\top + B^\top X_kA$$

ZAGER, Laura A. a George C. VERGHESE, 2008. Graph similarity scoring and matching. Applied Mathematics Letters [online]. 2008, roč. 21, č. 1, s. 86–94. ISSN 08939659. Dostupné na: doi:10.1016/j.aml.2007.01.006

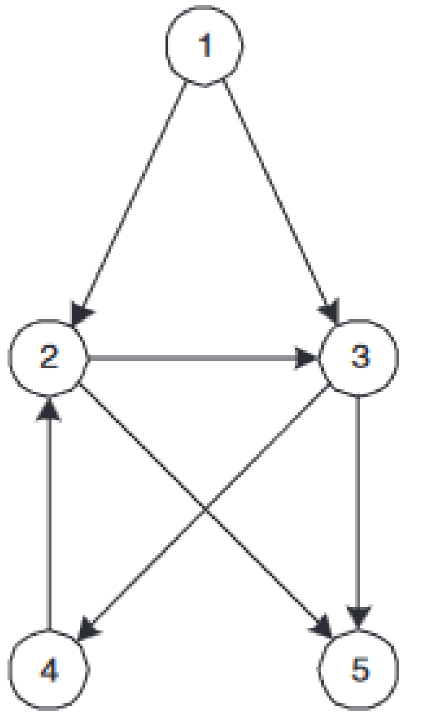
Tested convergence

According to authentic papers

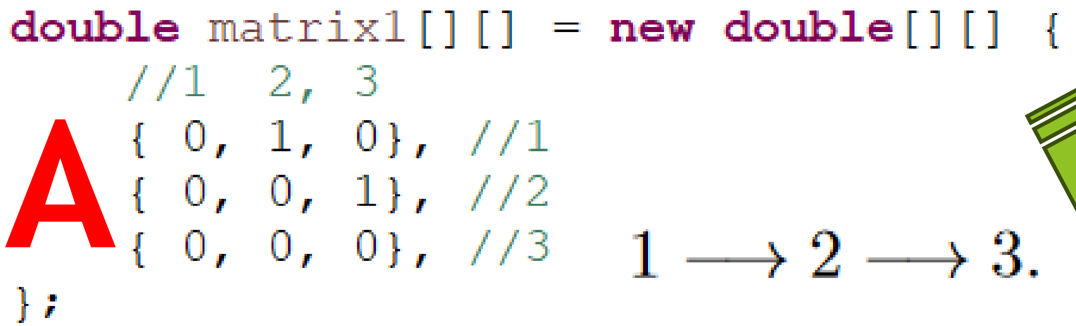
B



```
double matrix2[][] = new double[][]
    //1  2, 3, 4, 5
    { 0, 1, 1, 0, 0}, //1
    { 0, 0, 1, 0, 1}, //2
    { 0, 0, 0, 1, 1}, //3
    { 0, 1, 0, 0, 0}, //4
    { 0, 0, 0, 0, 0}, //5
};
```



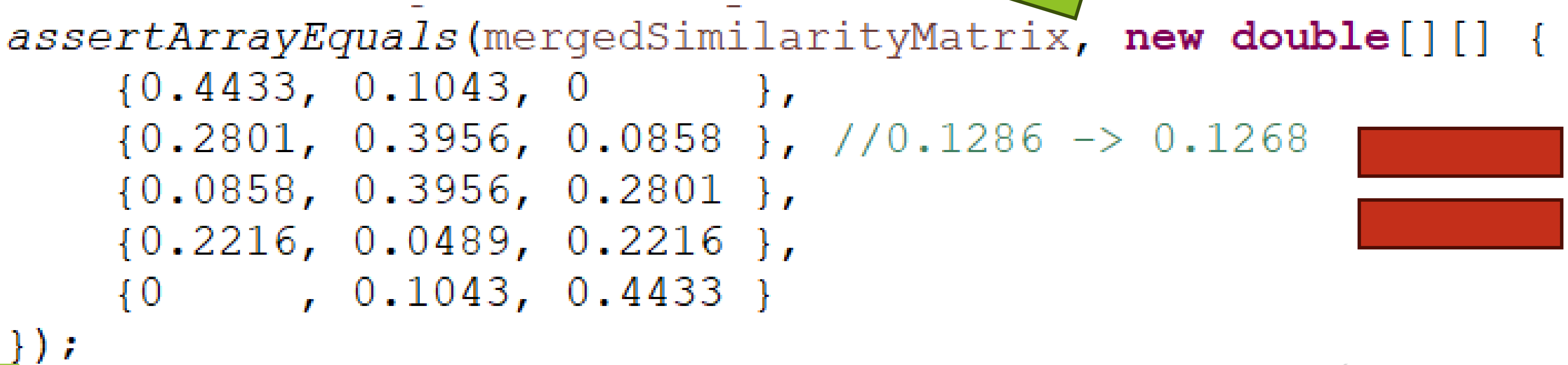
A



```
double matrix1[][] = new double[][] {
    //1  2, 3
    { 0, 1, 0}, //1
    { 0, 0, 1}, //2
    { 0, 0, 0}, //3
};
```

$$1 \longrightarrow 2 \longrightarrow 3.$$

$$X_{k+1} \leftarrow BX_kA^\top + B^\top X_kA$$

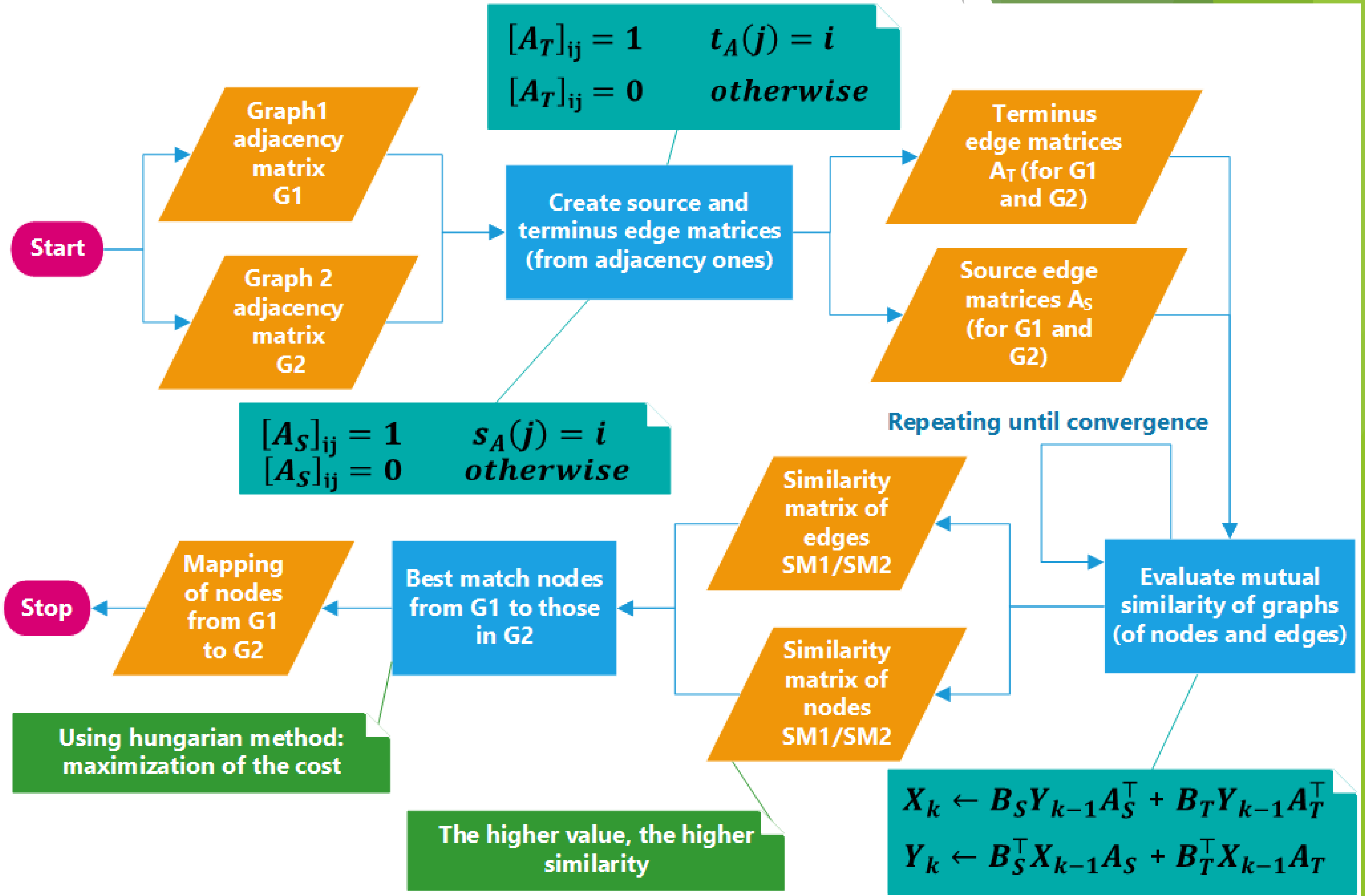


```
assertArrayEquals(mergedSimilarityMatrix, new double[][] {
        {0.4433, 0.1043, 0        },
        {0.2801, 0.3956, 0.0858 }, //0.1286 -> 0.1268
        {0.0858, 0.3956, 0.2801 },
        {0.2216, 0.0489, 0.2216 },
        {0       , 0.1043, 0.4433 }
});
```

$$\begin{bmatrix} 0.4433 & 0.1043 & 0 \\ 0.2801 & 0.3956 & 0.0858 \\ 0.0858 & 0.3956 & 0.2801 \\ 0.2216 & 0.0489 & 0.2216 \\ 0 & 0.1043 & 0.4433 \end{bmatrix}$$

Matrix-based graph matching based on node-edge similarity

ZAGER, Laura A. a George C. VERGHESE, 2008. Graph similarity scoring and matching. Applied Mathematics Letters [online]. 2008, roč. 21, č. 1, s. 86–94. ISSN 08939659. Dostupné na: doi:10.1016/j.aml.2007.01.006



Start

Graph1 adjacency matrix G1

Graph 2 adjacency matrix G2

Create source and terminus edge matrices (from adjacency ones)

$$[A_T]_{ij} = 1 \qquad t_A(j) = i$$
$$[A_T]_{ij} = 0 \qquad otherwise$$

$$[A_S]_{ij} = 1 \qquad s_A(j) = i$$
$$[A_S]_{ij} = 0 \qquad otherwise$$

Terminus edge matrices A_T (for G1 and G2)

Source edge matrices A_S (for G1 and G2)

Repeating until convergence

Evaluate mutual similarity of graphs (of nodes and edges)

$$X_k \leftarrow B_S Y_{k-1} A_S^{\top} + B_T Y_{k-1} A_T^{\top}$$
$$Y_k \leftarrow B_S^{\top} X_{k-1} A_S + B_T^{\top} X_{k-1} A_T$$

Similarity matrix of edges SM1/SM2

Similarity matrix of nodes SM1/SM2

The higher value, the higher similarity

Best match nodes from G1 to those in G2

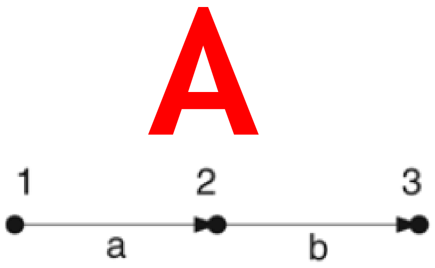
Using hungarian method: maximization of the cost

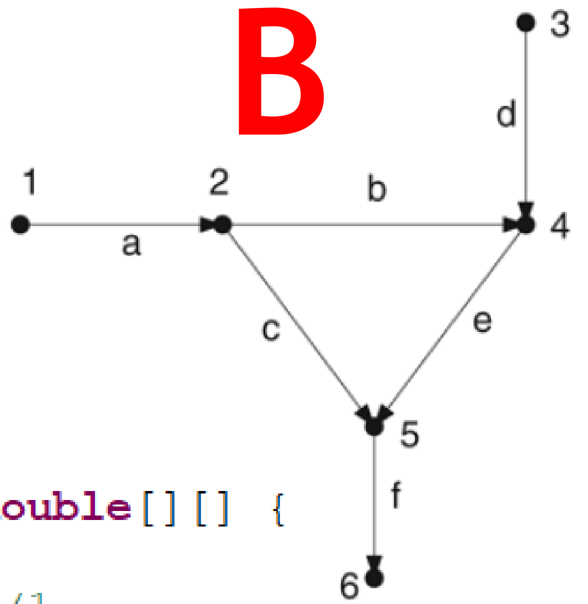
Mapping of nodes from G1 to G2

Stop

Tested convergence

According to authentic papers





```
double matrix1[][] = new double[][] {
        //1  2, 3
        { 0, 1, 0}, //1
        { 0, 0, 1}, //2
        { 0, 0, 0}, //3
};
```

```
assertArrayEquals(vertexMatrix, new double[][] {
    {0.124, 0,      0},
    {0.348, 0.444, 0},        //0.445 -> 0.444
    {0.157, 0.054, 0},
    {0.094, 0.564, 0.192},    //0.563 -> 0.564; 0.193
    {0     , 0.338, 0.389},   //0.340 -> 0.389
    {0     , 0     , 0.094}
}
```

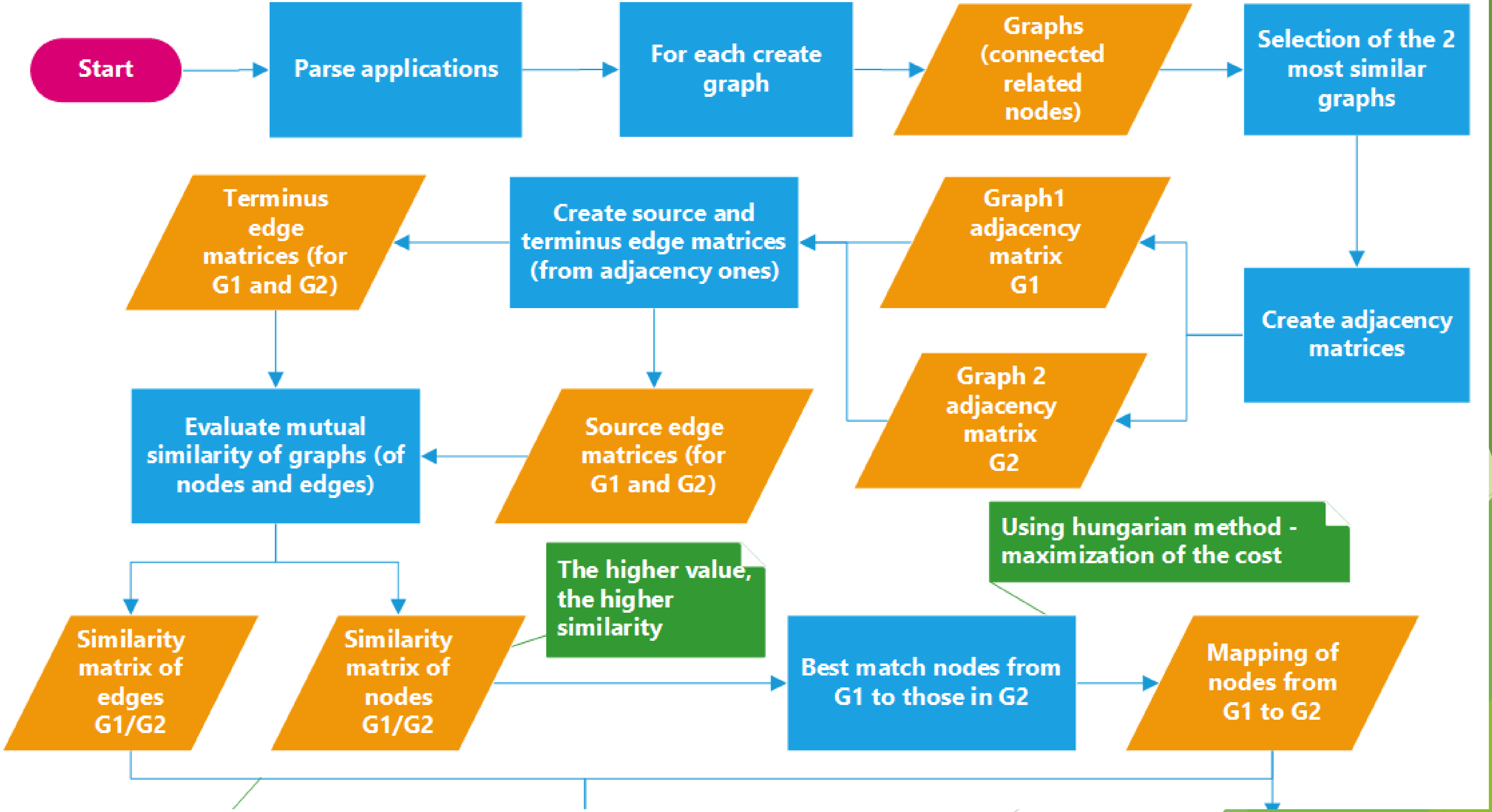
```
double matrix2[][] = new double[][] {
        //1  2, 3, 4, 5, 6
        { 0, 1, 0, 0, 0, 0}, //1
        { 0, 0, 0, 1, 1, 0}, //2
        { 0, 0, 0, 1, 0, 0}, //3
        { 0, 0, 0, 0, 1, 0}, //4
        { 0, 0, 0, 0, 0, 1}, //5
        { 0, 0, 0, 0, 0, 0}  //6
};
```

```
assertArrayEquals(ed
    {0.265, 0},
    {0.426, 0.297},
    {0.320, 0.389},
    {0.336, 0.115},
    {0.202, 0.445},
    {0     , 0.202},
```

Nodes	1	2	3
1	0.124	0	0
2	0.348	0.445	0
3	0.157	0.054	0
4	0.094	0.563	0.193
5	0	0.338	0.340
6	0	0	0.094

Edges	*a*	*b*
a	0.265	0
b	0.426	0.297
c	0.320	0.389
d	0.336	0.115
e	0.202	0.445
f	0	0.202

Integration of matrix-based methods



Start

Parse applications

For each create graph

Graphs (connected related nodes)

Selection of the 2 most similar graphs

Create adjacency matrices

Graph1 adjacency matrix G1

Graph 2 adjacency matrix G2

Create source and terminus edge matrices (from adjacency ones)

Terminus edge matrices (for G1 and G2)

Source edge matrices (for G1 and G2)

Evaluate mutual similarity of graphs (of nodes and edges)

Similarity matrix of edges G1/G2

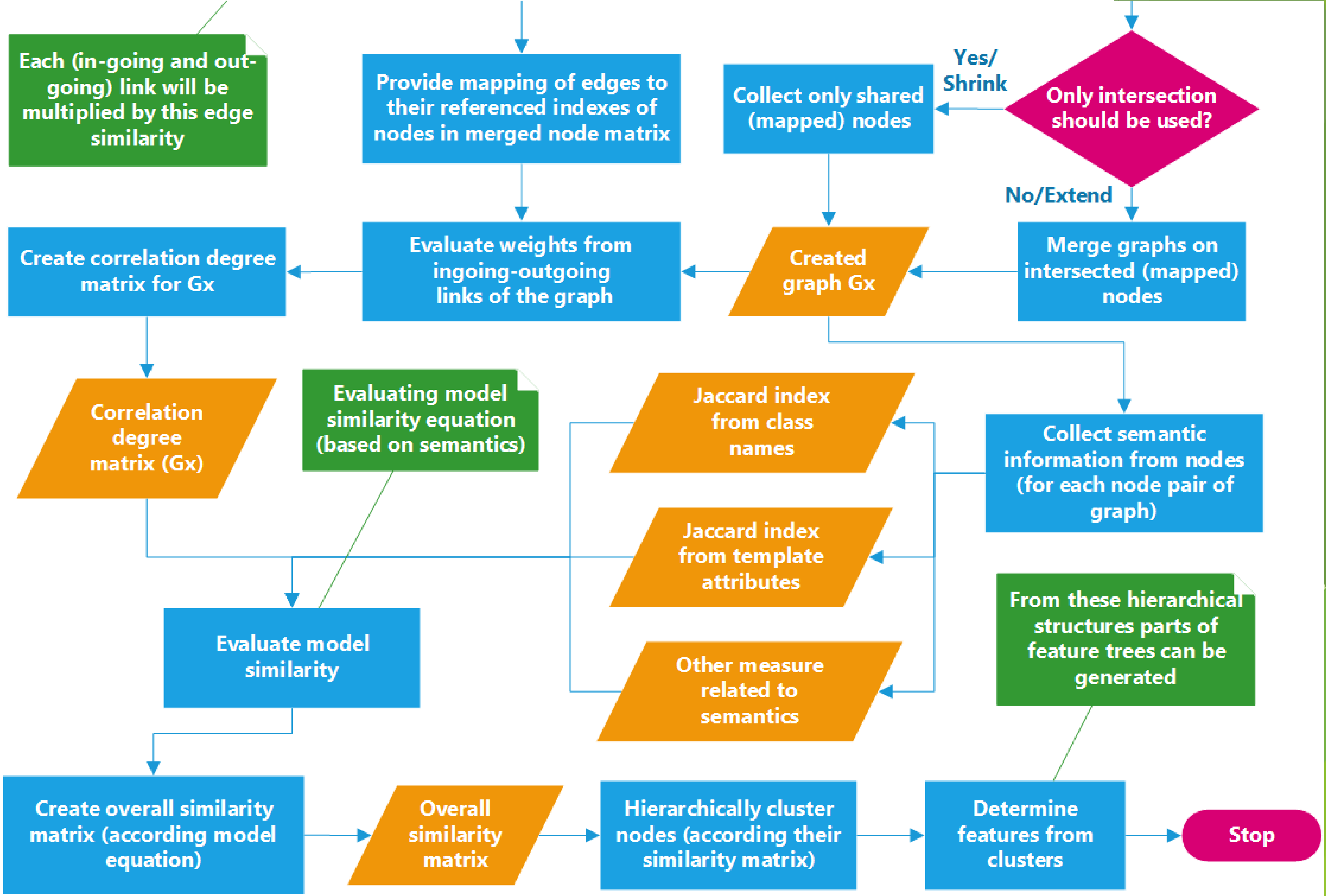
Similarity matrix of nodes G1/G2

The higher value, the higher similarity

Best match nodes from G1 to those in G2

Using hungarian method - maximization of the cost

Mapping of nodes from G1 to G2



Only intersection should be used?

Yes/ Shrink

Collect only shared (mapped) nodes

No/Extend

Merge graphs on intersected (mapped) nodes

Created graph Gx

Provide mapping of edges to their referenced indexes of nodes in merged node matrix

Evaluate weights from ingoing-outgoing links of the graph

Each (in-going and out-going) link will be multiplied by this edge similarity

Create correlation degree matrix for Gx

Correlation degree matrix (Gx)

Collect semantic information from nodes (for each node pair of graph)

Jaccard index from class names

Jaccard index from template attributes

Other measure related to semantics

Evaluating model similarity equation (based on semantics)

Evaluate model similarity

Create overall similarity matrix (according model equation)

Overall similarity matrix

Hierarchically cluster nodes (according their similarity matrix)

Determine features from clusters

From these hierarchical structures parts of feature trees can be generated

Stop

Model similarity

$$sim(i,j) = \underbrace{\alpha_{ij} * sim_{ij}^{in} + \beta_{ij} * sim_{ij}^{out}}_{\text{Structural information}} + \underbrace{\gamma_{ij} * sim_{ij}^{sem1} + ... + \epsilon_{ij} * sim_{ij}^{sem2}}_{\text{Semantic information}}$$

CREATING MULTI-CONTENT AND MULTI-PURPOSE FRACTAL DATASET

MULTI-CONTENT

-JSON data from variability points
-raster screenshots/images
-vector SVG structure information
-table from the variability information itself
-data from recursion

MULTI-PURPOSE

-aesthetic evaluation
-comparing the same models on
different data formats
-SPL evolution through variability points evaluation
– if they should be included or merged
-associating products with their generators/software parts
-generate the similar fractals using GANS

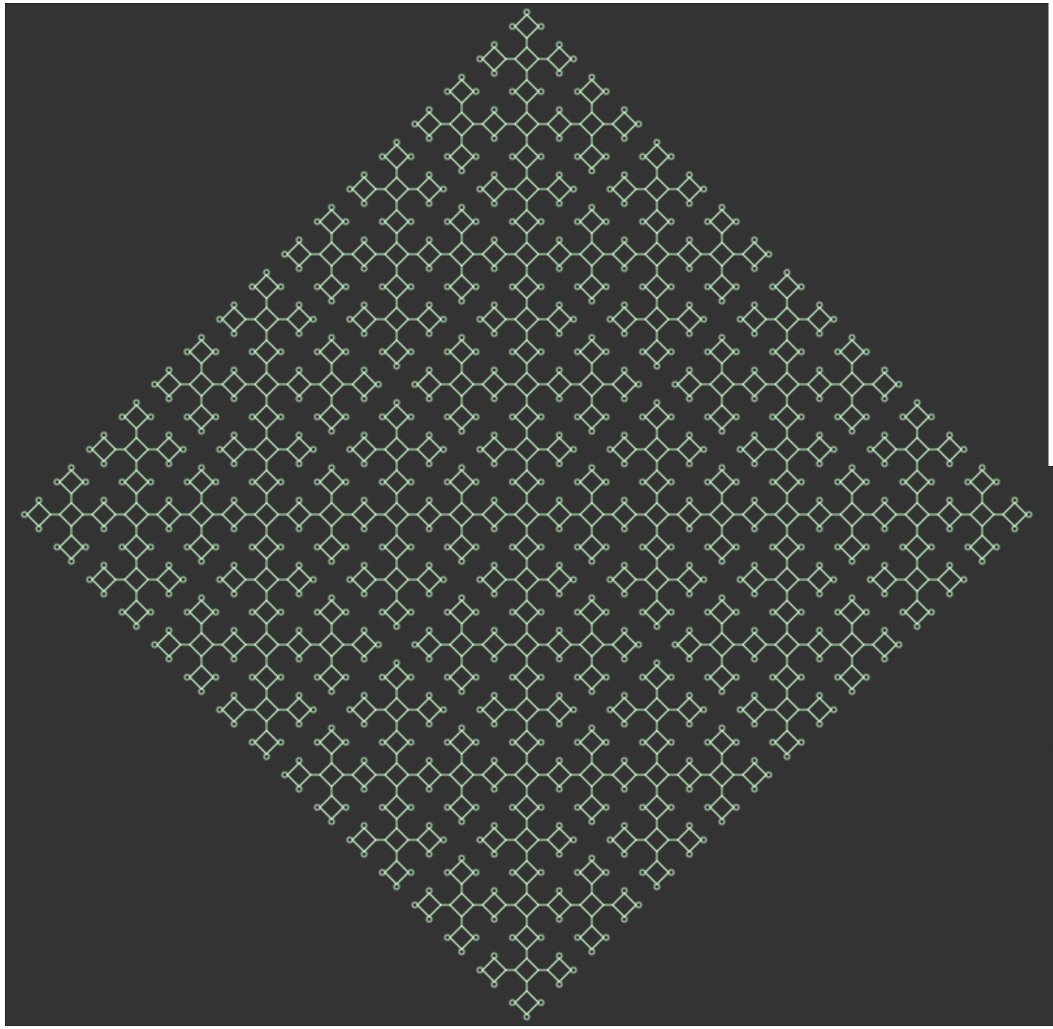
Many of them can be often generalized

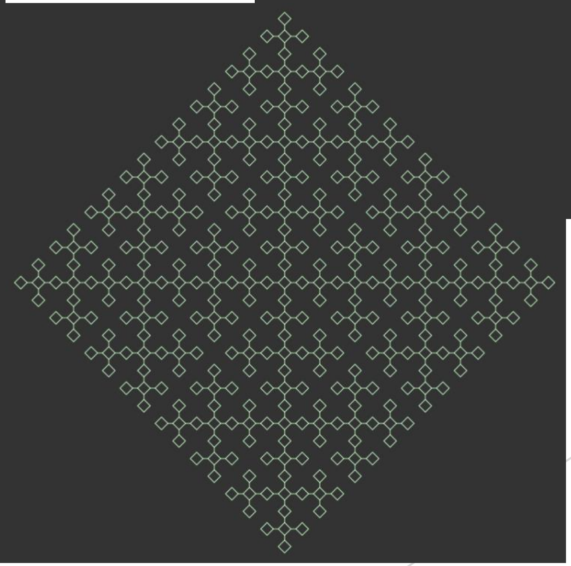
RECURSION IN SPL

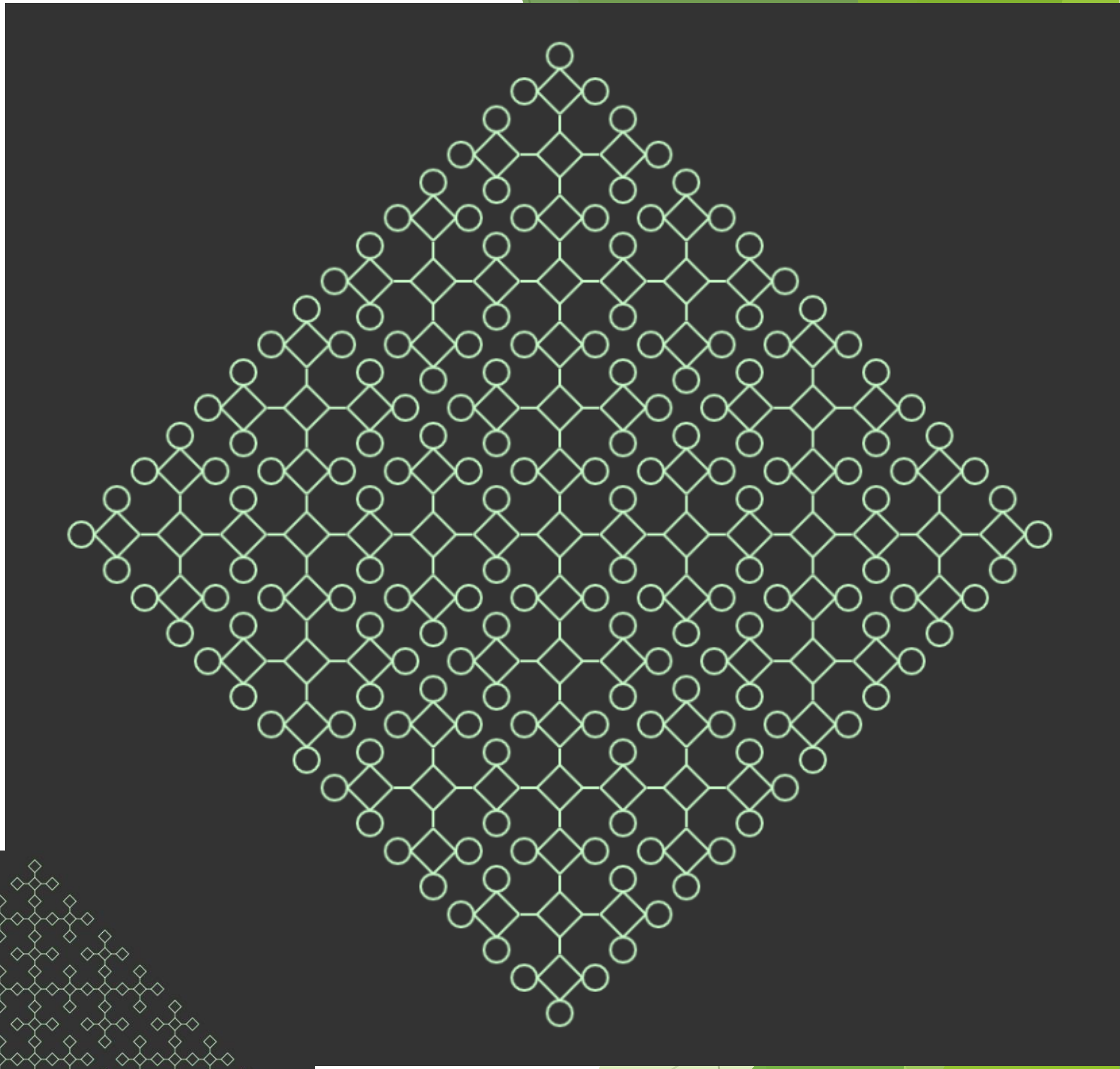
The same code parts are repeatedly **reused** - **with different values**

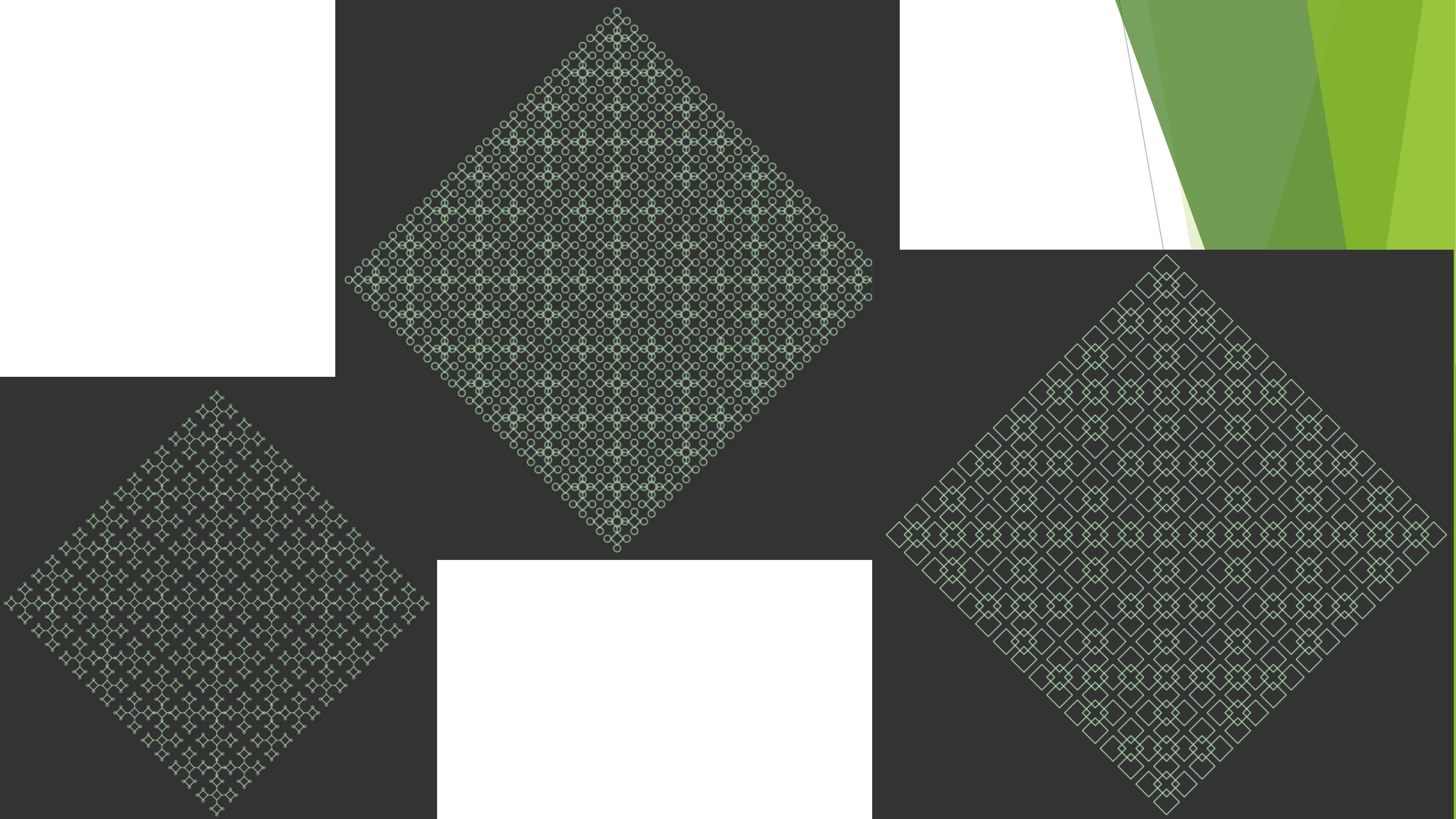
HOW IT AFFECTS VARIABILITY MODELING?

Given samples of one type









Many types

Different approaches how to generate

The Need for framework

extension for based on previous work, but focused mainly on variability

Iteratively building and enhancing framework

Variability configuration

- Repeating the same code fragments
  - – addinional for cyckle with different range in each iteration
- Combining many code parts
- Excluding optional code parts from some derivations

Logging

- variables
- function parameters
- permutation of variables
  - Recursion depth is the most important dependency
- precalculating values to log them together

Evaluating customized dataset

- Manual annotations – based on own aesthetics
- Used third party model

- -comparing different fractal representations/formats:
  - Vector graphics – whole structure is written as text .SVG
  - Raster graphics
  - Information from variability points

- **inserts knowledge from structure of program generator itself into data**

Already 378 fractals generated from one file

-based on variability points permutations and recursion

Can be more, but...

we bring:

assymetry, chaos, standalone lines creating non-fractal shape

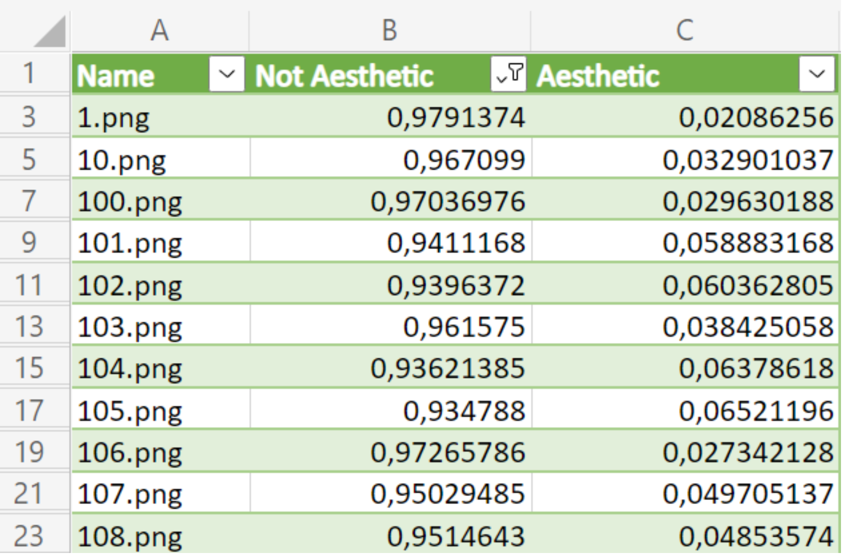
EASY TO EXECUTE AND ANALYZE FRACTAL SCRIPT IN MANY PROGRAMMING LANGUAGES ***js2py for Python***

Will it help to enhance third party models and systems?

-improve their accuracy

Can actual results from model be used as label values?

For evaluation
– learning with teacher needs annotated data



	A	B	C
1	Name	Not Aesthetic	Aesthetic
3	1.png	0,9791374	0,02086256
5	10.png	0,967099	0,032901037
7	100.png	0,97036976	0,029630188
9	101.png	0,9411168	0,058883168
11	102.png	0,9396372	0,060362805
13	103.png	0,961575	0,038425058
15	104.png	0,93621385	0,06378618
17	105.png	0,934788	0,06521196
19	106.png	0,97265786	0,027342128
21	107.png	0,95029485	0,049705137
23	108.png	0,9514643	0,04853574

Evaluated model data

Are all values of the same value?

Yes if borders of images are filled to the same size

No, if we take original images
– how far fractal can be extended

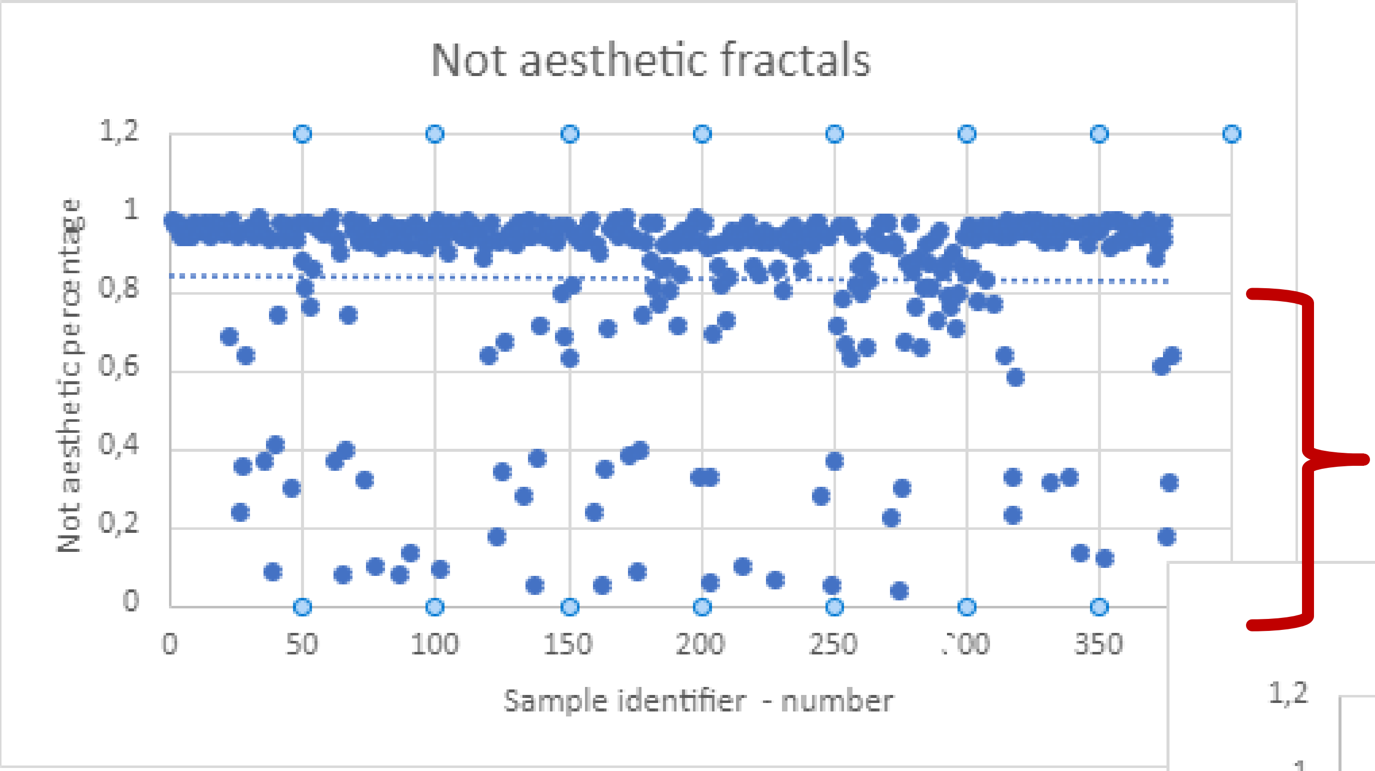
Is model suitable for evaluating fractals?

Maybe yes, but evaluation is also focused mainly on:
colors, golden cut, perspective,
view of the spectator/camera

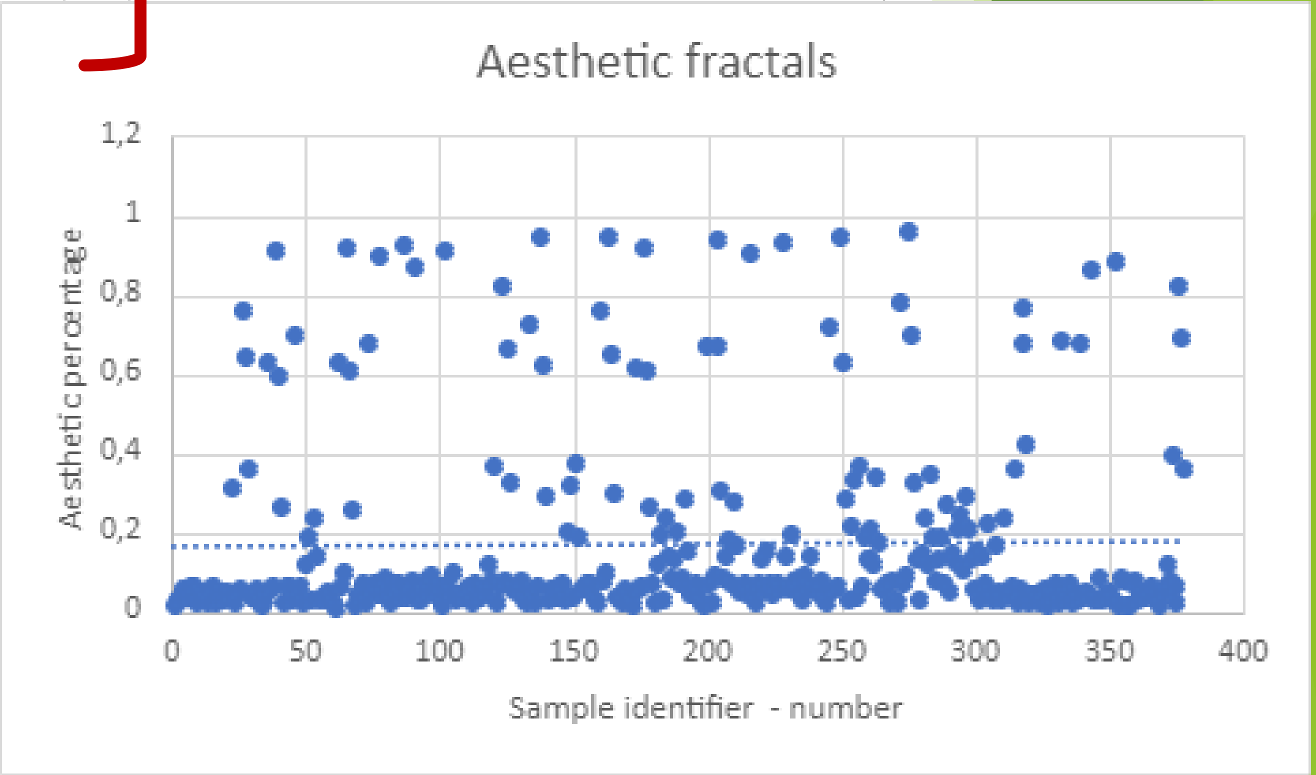
OWN MODEL IS REQUIRED

-restrict it on shapes only/mainly

-better if deformations were detected and evaluated accordingly



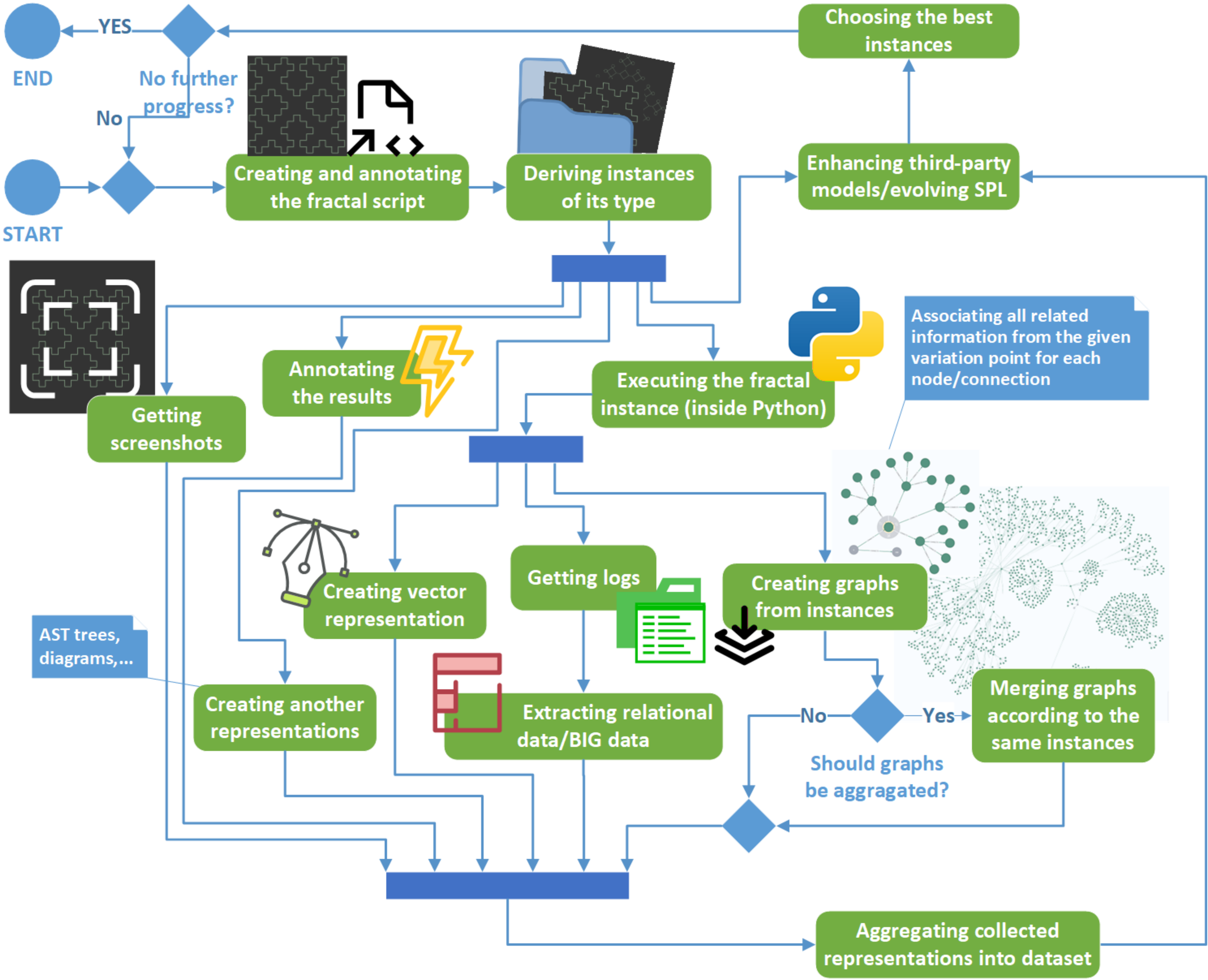
Can serve as restricted „reference“ for further evaluation



Why fractals?

Are they necessary? – YES

- Multiple format representations (vector, raster, text)
    - All images can be converted to SVG, but not all are suitable as shapes – bigger, better more points – image quality
- We can use them as already created “products”
- No other dependencies - easy to execute code and get values from the execution
- Code that is executed repeatedly
- Variability management on lower levels (code level) – components are not suitable
- Variability reaches a “high degree” – almost everything is variability
    - No reuse? – NO in recursion there is high reuse, also across all types of derivations
- Many samples can be generated – also merging existing ones
    - Thousands – already hundreds of quality ones from one type
    - Not all are aesthetic or interesting



START

END

No further progress?

YES

No

Creating and annotating the fractal script

Deriving instances of its type

Choosing the best instances

Enhancing third-party models/evolving SPL

Getting screenshots

Annotating the results

Executing the fractal instance (inside Python)

Associating all related information from the given variation point for each node/connection

Creating vector representation

Getting logs

Creating graphs from instances

AST trees, diagrams,…

Creating another representations

Extracting relational data/BIG data

Should graphs be aggregated?

No

Yes

Merging graphs according to the same instances

Aggregating collected representations into dataset

Method based on annotations and aspects - *recursive extension*

still compilable

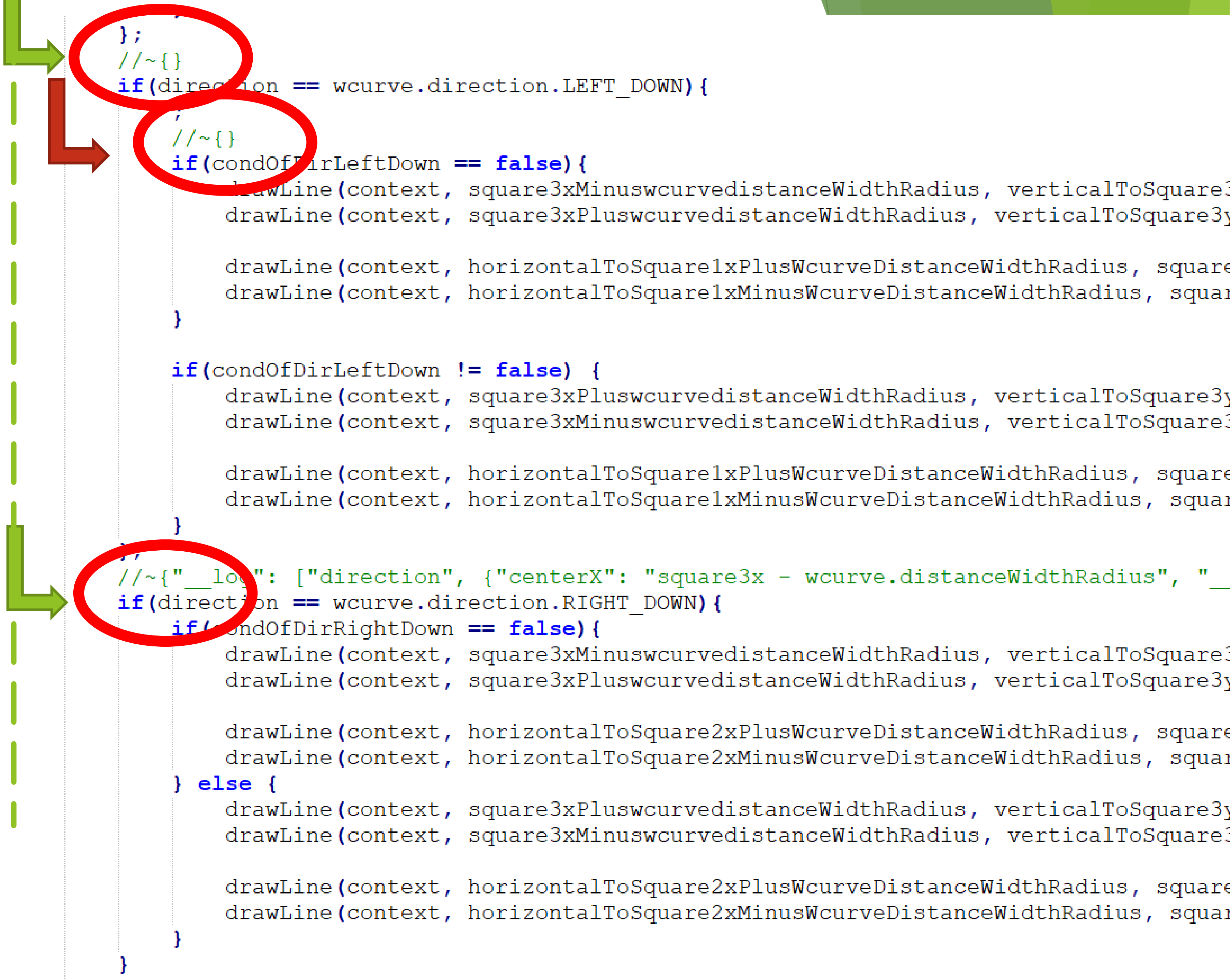
Only one small script is enough

for 378 samples, but generating fractals

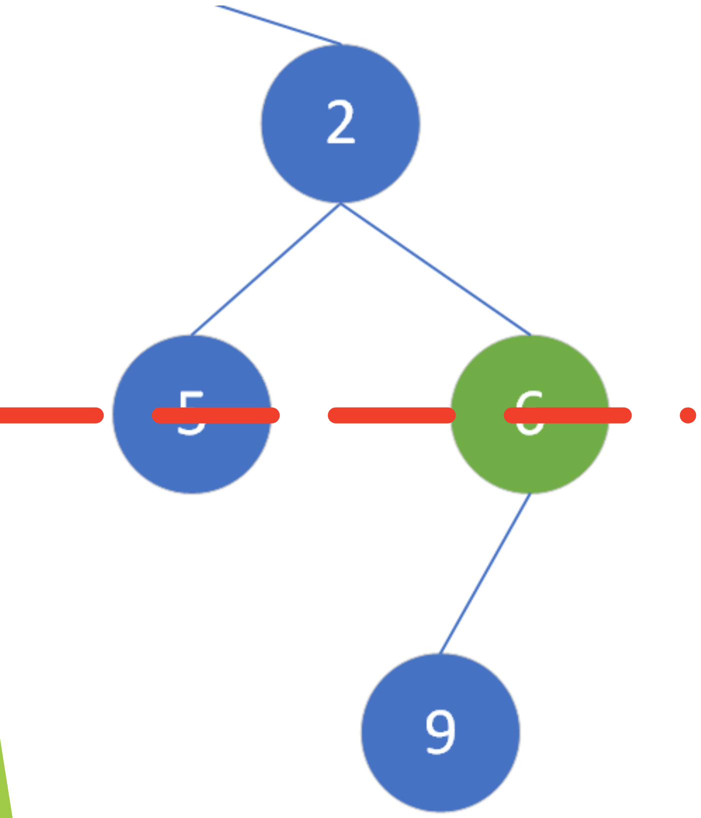
in for cycles still produces

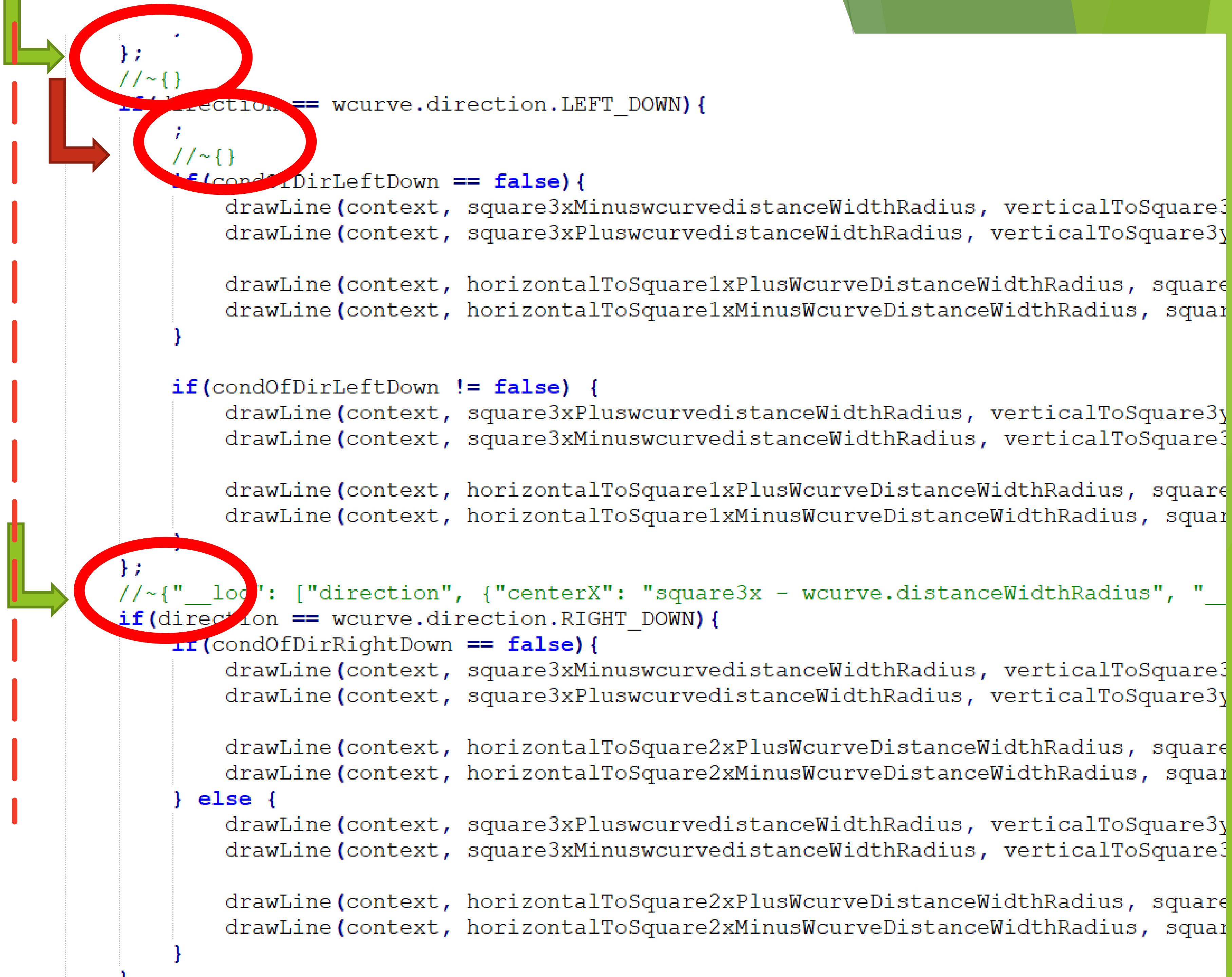
a few same shapes

-translation is not productive



```
        ;
    };
    //~{}
    if(direction == wcurve.direction.LEFT_DOWN){
        ;
        //~{}
        if(condOfDirLeftDown == false){
            drawLine(context, square3xMinuswcurvedistanceWidthRadius, verticalToSquare3
            drawLine(context, square3xPluswcurvedistanceWidthRadius, verticalToSquare3y

            drawLine(context, horizontalToSquare1xPlusWcurveDistanceWidthRadius, square
            drawLine(context, horizontalToSquare1xMinusWcurveDistanceWidthRadius, squar
        }

        if(condOfDirLeftDown != false) {
            drawLine(context, square3xPluswcurvedistanceWidthRadius, verticalToSquare3y
            drawLine(context, square3xMinuswcurvedistanceWidthRadius, verticalToSquare3

            drawLine(context, horizontalToSquare1xPlusWcurveDistanceWidthRadius, square
            drawLine(context, horizontalToSquare1xMinusWcurveDistanceWidthRadius, squar
        }
    };
    //~{"__log": ["direction", {"centerX": "square3x - wcurve.distanceWidthRadius", "__
    if(direction == wcurve.direction.RIGHT_DOWN){
        if(condOfDirRightDown == false){
            drawLine(context, square3xMinuswcurvedistanceWidthRadius, verticalToSquare3
            drawLine(context, square3xPluswcurvedistanceWidthRadius, verticalToSquare3y

            drawLine(context, horizontalToSquare2xPlusWcurveDistanceWidthRadius, square
            drawLine(context, horizontalToSquare2xMinusWcurveDistanceWidthRadius, squar
        } else {
            drawLine(context, square3xPluswcurvedistanceWidthRadius, verticalToSquare3y
            drawLine(context, square3xMinuswcurvedistanceWidthRadius, verticalToSquare3

            drawLine(context, horizontalToSquare2xPlusWcurveDistanceWidthRadius, square
            drawLine(context, horizontalToSquare2xMinusWcurveDistanceWidthRadius, squar
        }
    }
```

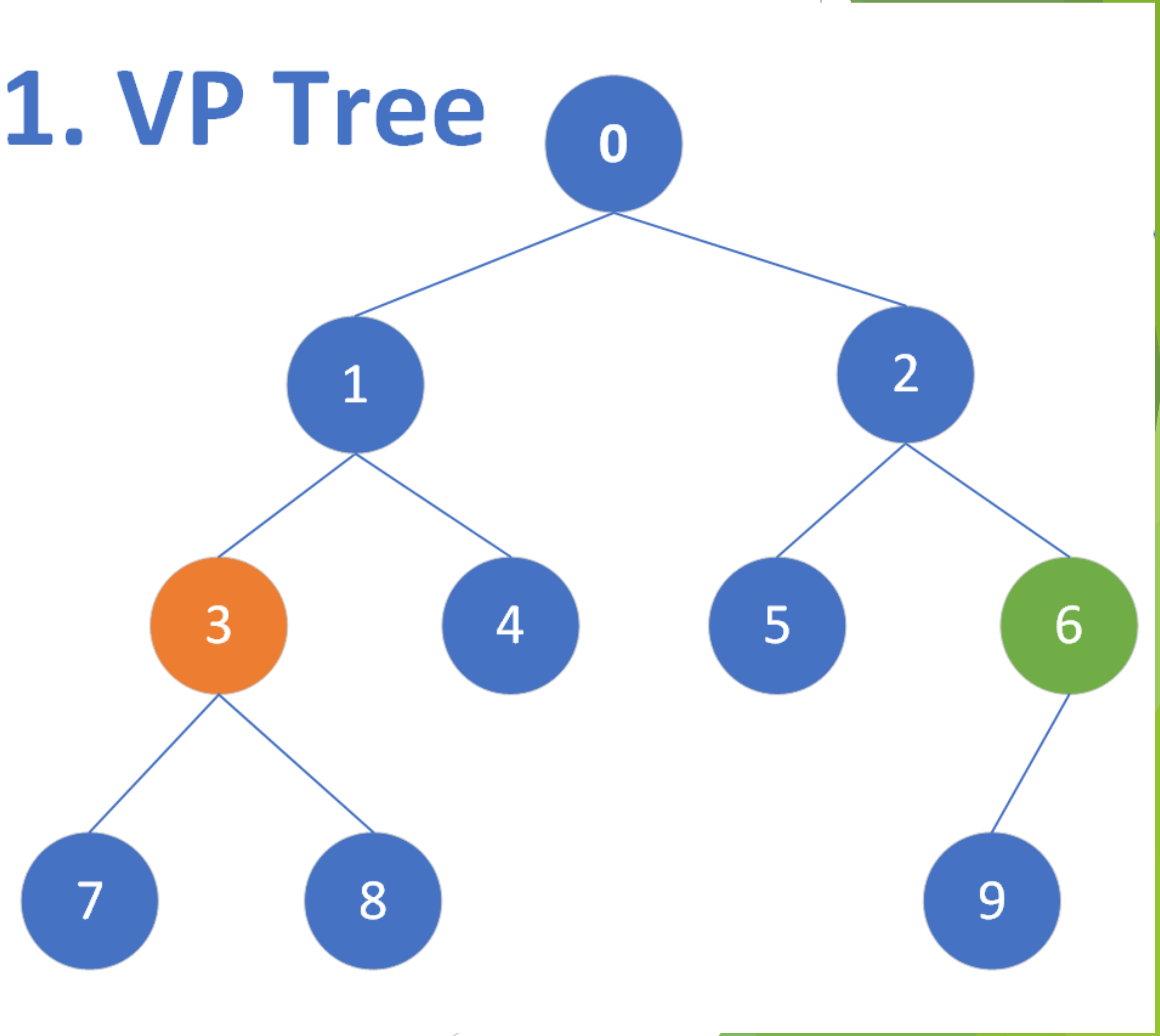




```
    ;
};
//~{}
if(direction == wcurve.direction.LEFT_DOWN){
    ;
    //~{}
    if(condOfDirLeftDown == false){
        drawLine(context, square3xMinuswcurvedistanceWidthRadius, verticalToSquare3
        drawLine(context, square3xPluswcurvedistanceWidthRadius, verticalToSquare3y

        drawLine(context, horizontalToSquare1xPlusWcurveDistanceWidthRadius, square
        drawLine(context, horizontalToSquare1xMinusWcurveDistanceWidthRadius, squar
    }

    if(condOfDirLeftDown != false) {
        drawLine(context, square3xPluswcurvedistanceWidthRadius, verticalToSquare3y
        drawLine(context, square3xMinuswcurvedistanceWidthRadius, verticalToSquare3

        drawLine(context, horizontalToSquare1xPlusWcurveDistanceWidthRadius, square
        drawLine(context, horizontalToSquare1xMinusWcurveDistanceWidthRadius, squar
    }
};
//~{"__loc": ["direction", {"centerX": "square3x - wcurve.distanceWidthRadius", "__
if(direction == wcurve.direction.RIGHT_DOWN){
    if(condOfDirRightDown == false){
        drawLine(context, square3xMinuswcurvedistanceWidthRadius, verticalToSquare3
        drawLine(context, square3xPluswcurvedistanceWidthRadius, verticalToSquare3y

        drawLine(context, horizontalToSquare2xPlusWcurveDistanceWidthRadius, square
        drawLine(context, horizontalToSquare2xMinusWcurveDistanceWidthRadius, squar
    } else {
        drawLine(context, square3xPluswcurvedistanceWidthRadius, verticalToSquare3y
        drawLine(context, square3xMinuswcurvedistanceWidthRadius, verticalToSquare3

        drawLine(context, horizontalToSquare2xPlusWcurveDistanceWidthRadius, square
        drawLine(context, horizontalToSquare2xMinusWcurveDistanceWidthRadius, squar
    }
}
```

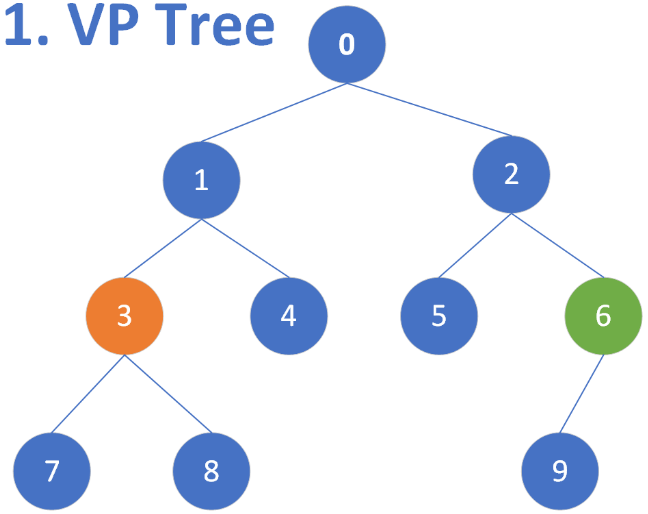
Creating tree from these points

- //~{}

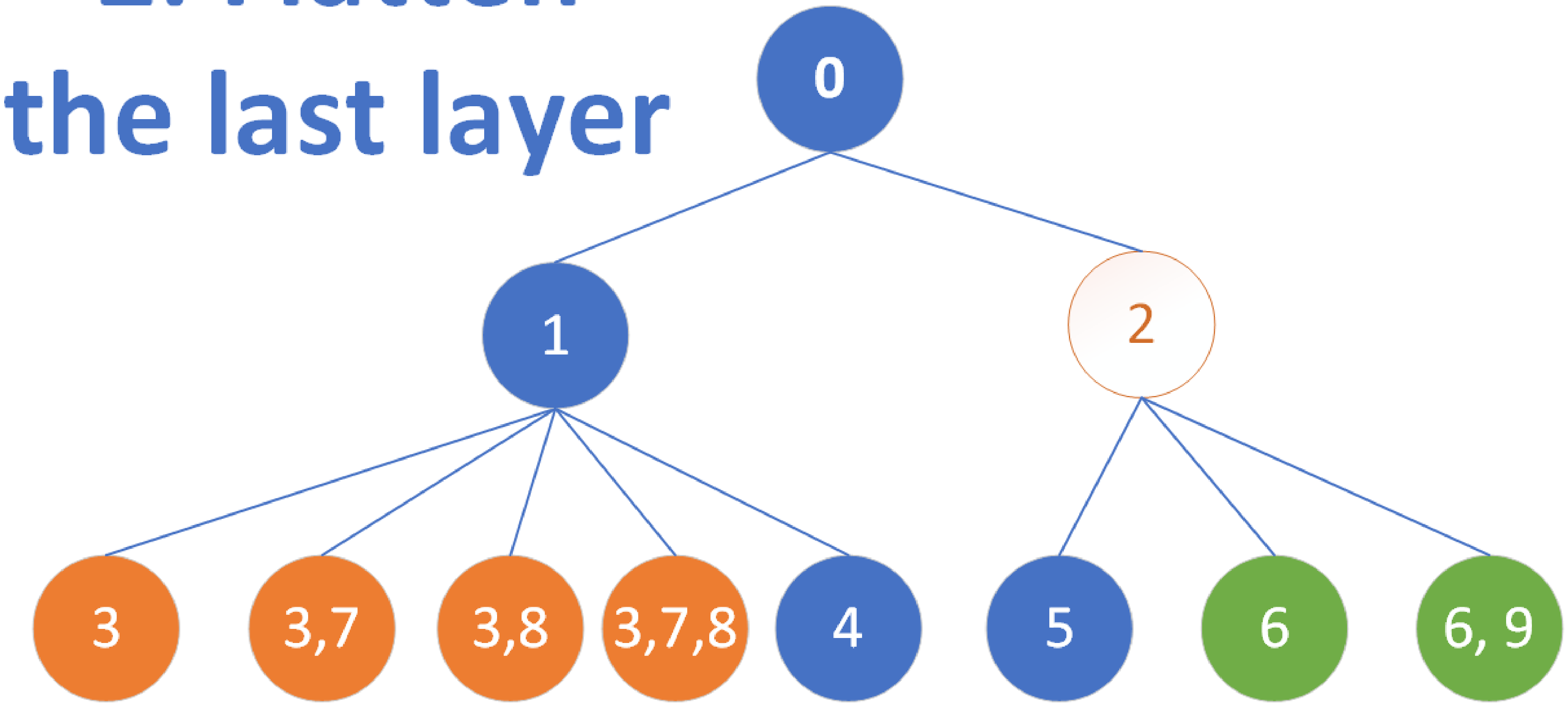


1. VP Tree

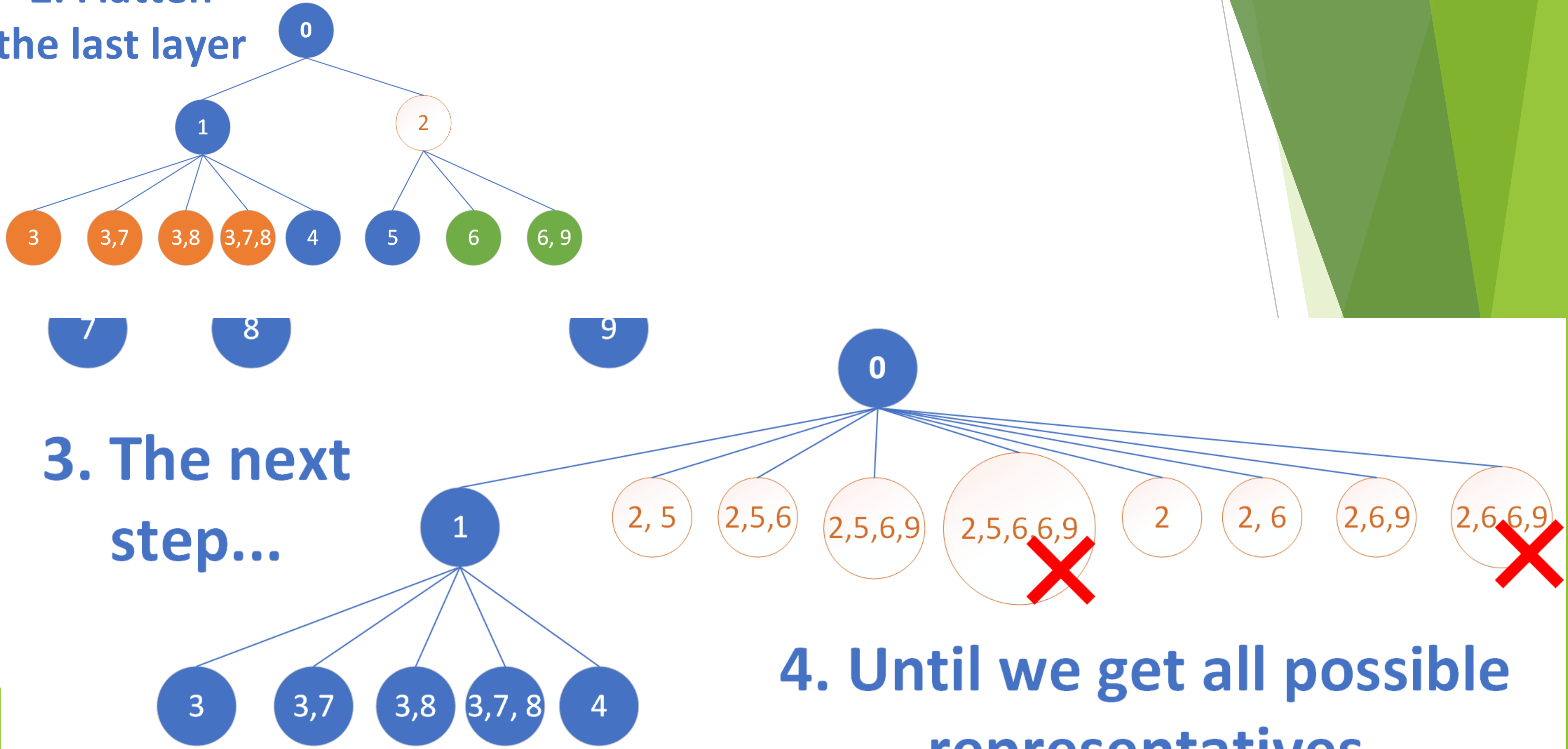
1. VP Tree



2. Flatten the last layer



2. Flatten the last layer



3. The next step...

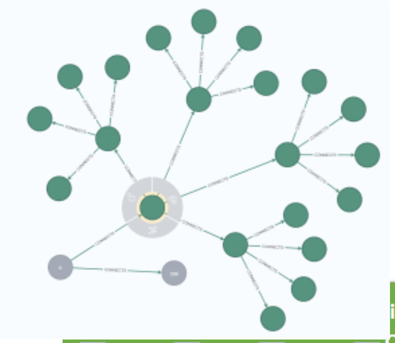
4. Until we get all possible representatives

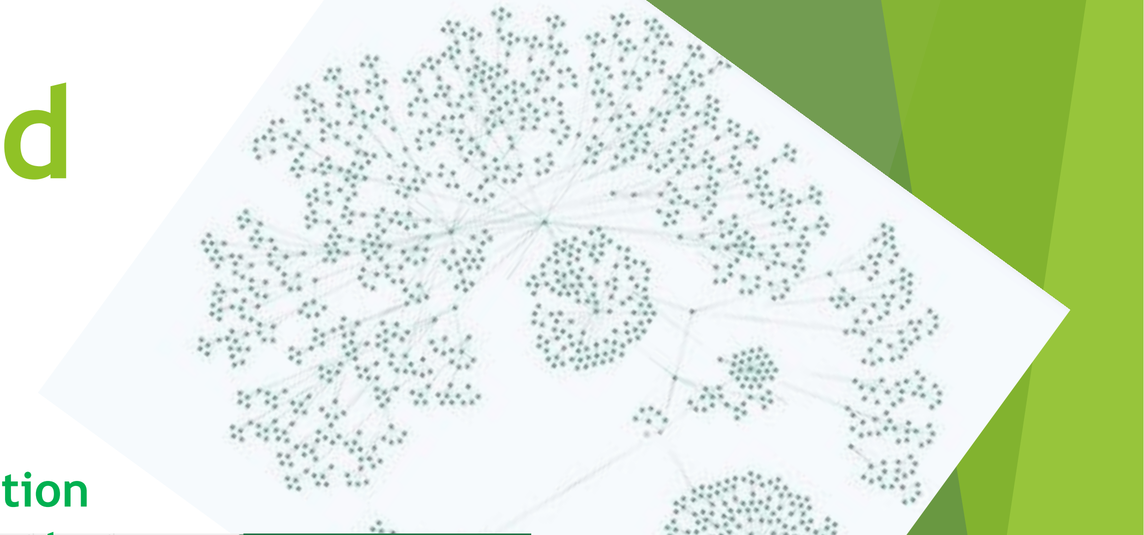
Creating the best representations

... according to the given requirements for model construction and evaluation of aesthetics ...

Raster screenshots

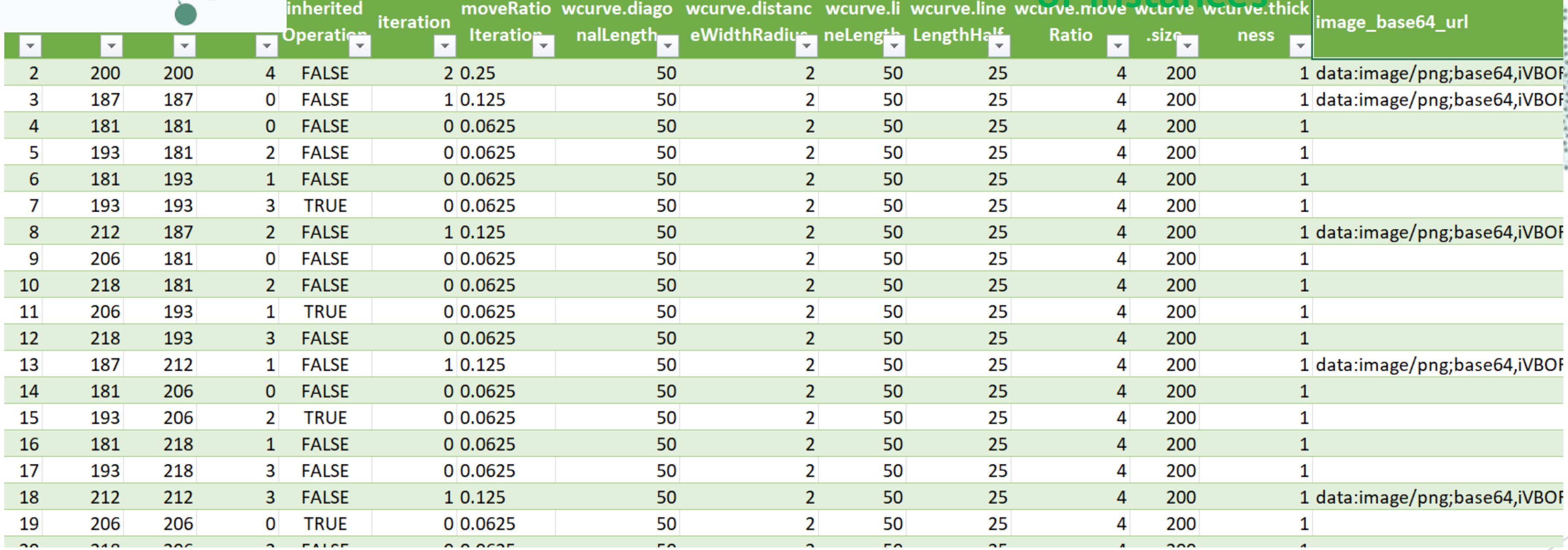
Graph data - nodes and connections



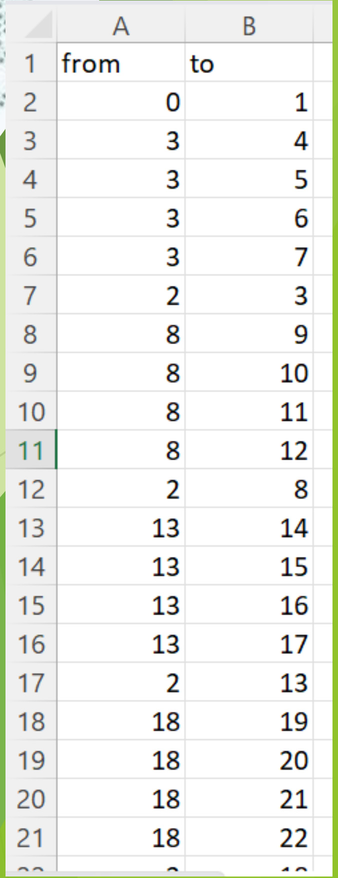


One instance

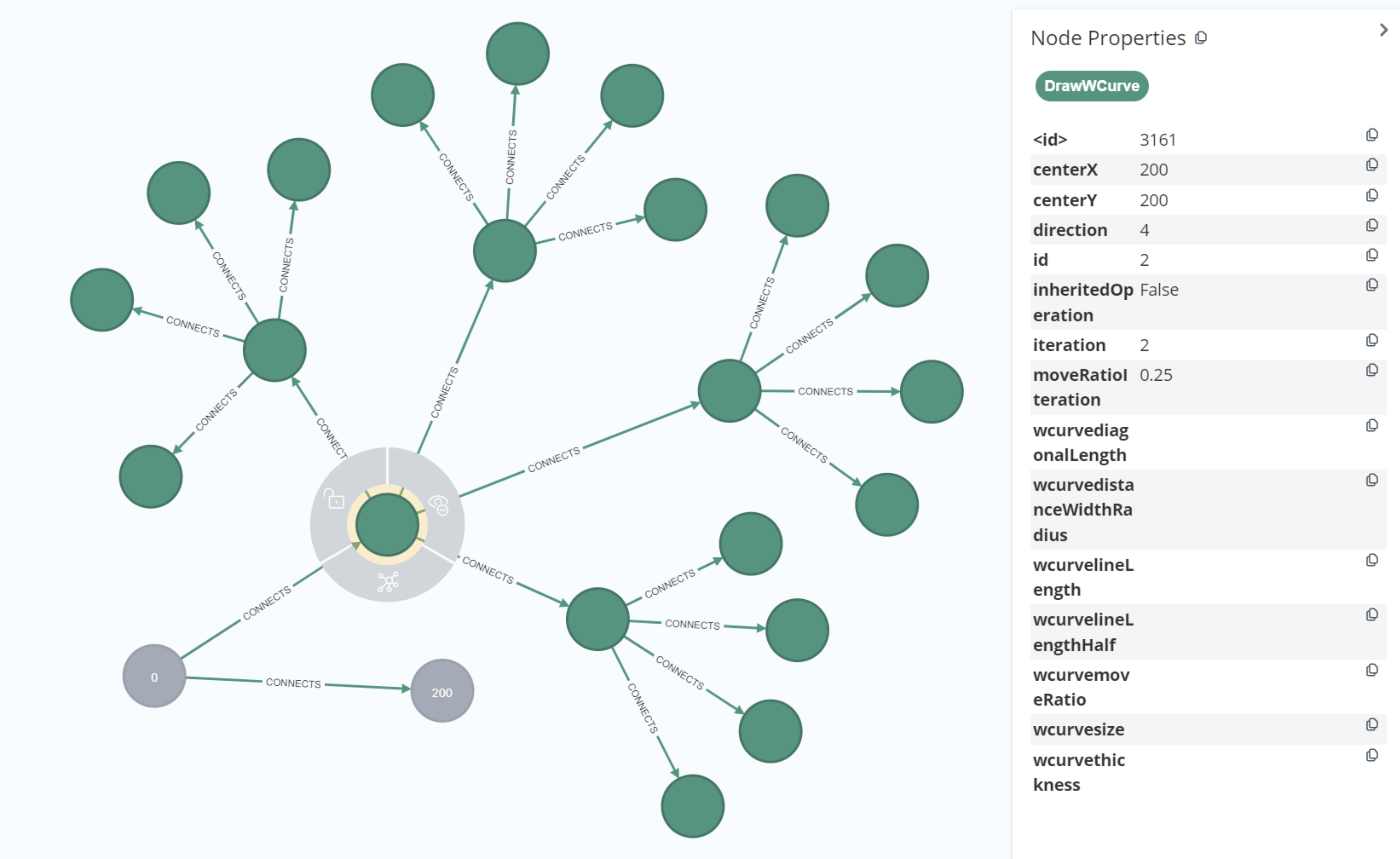
Aggregation of instances



				inherited Operation	iteration	moveRatio Iteration	wcurve.diagonalLength	wcurve.distanceWidthRadius	wcurve.lineLength	wcurve.lineLengthHalf	wcurve.moveRatio	wcurve.size	wcurve.thickness	image_base64_url
2	200	200	4	FALSE	2	0.25	50	2	50	25	4	200	1	data:image/png;base64,iVBOR
3	187	187	0	FALSE	1	0.125	50	2	50	25	4	200	1	data:image/png;base64,iVBOR
4	181	181	0	FALSE	0	0.0625	50	2	50	25	4	200	1	
5	193	181	2	FALSE	0	0.0625	50	2	50	25	4	200	1	
6	181	193	1	FALSE	0	0.0625	50	2	50	25	4	200	1	
7	193	193	3	TRUE	0	0.0625	50	2	50	25	4	200	1	
8	212	187	2	FALSE	1	0.125	50	2	50	25	4	200	1	data:image/png;base64,iVBOR
9	206	181	0	FALSE	0	0.0625	50	2	50	25	4	200	1	
10	218	181	2	FALSE	0	0.0625	50	2	50	25	4	200	1	
11	206	193	1	TRUE	0	0.0625	50	2	50	25	4	200	1	
12	218	193	3	FALSE	0	0.0625	50	2	50	25	4	200	1	
13	187	212	1	FALSE	1	0.125	50	2	50	25	4	200	1	data:image/png;base64,iVBOR
14	181	206	0	FALSE	0	0.0625	50	2	50	25	4	200	1	
15	193	206	2	TRUE	0	0.0625	50	2	50	25	4	200	1	
16	181	218	1	FALSE	0	0.0625	50	2	50	25	4	200	1	
17	193	218	3	FALSE	0	0.0625	50	2	50	25	4	200	1	
18	212	212	3	FALSE	1	0.125	50	2	50	25	4	200	1	data:image/png;base64,iVBOR
19	206	206	0	TRUE	0	0.0625	50	2	50	25	4	200	1	



	A	B
1	from	to
2	0	1
3	3	4
4	3	5
5	3	6
6	3	7
7	2	3
8	8	9
9	8	10
10	8	11
11	8	12
12	2	8
13	13	14
14	13	15
15	13	16
16	13	17
17	2	13
18	18	19
19	18	20
20	18	21
21	18	22



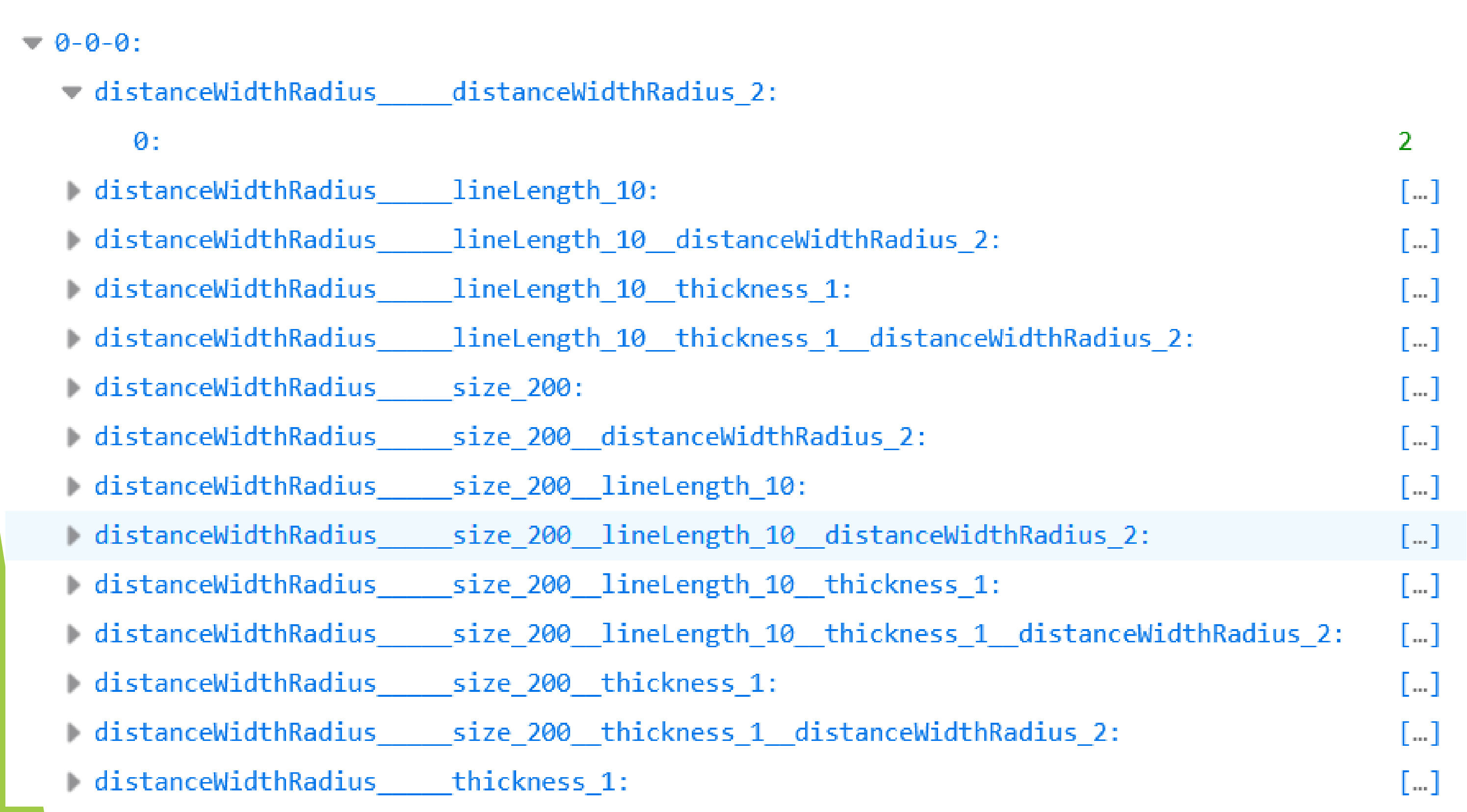
Node Properties

DrawWCurve

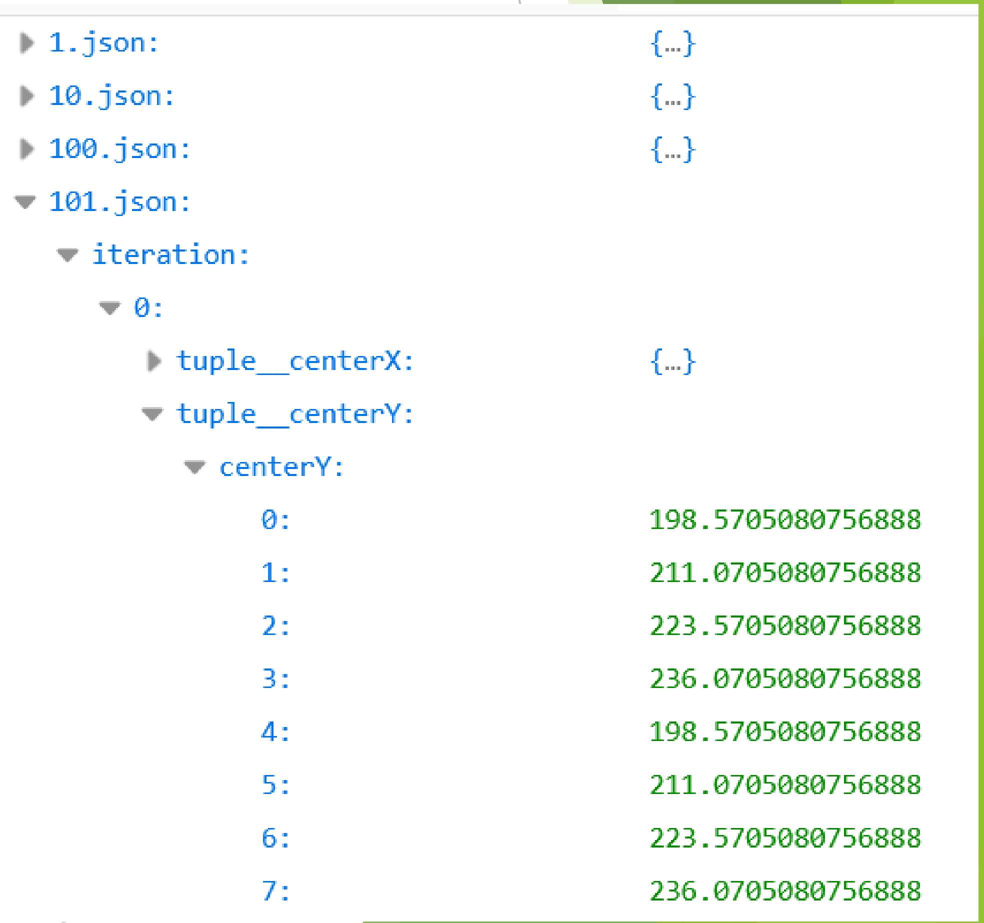
<id>	3161
centerX	200
centerY	200
direction	4
id	2
inheritedOperation	False
iteration	2
moveRatioIteration	0.25
wcurvediagonalLength	
wcurvedistanceWidthRadius	
wcurvelineLength	
wcurvelineLengthHalf	
wcurvemoveRatio	
wcurvesize	
wcurvethickness	

Semi-structured data - variable dependencies

recursion depth as reusability of the components:



```
▼ 0-0-0:
  ▼ distanceWidthRadius_____distanceWidthRadius_2:
      0:                                                                  2
  ▶ distanceWidthRadius_____lineLength_10:                                [...]
  ▶ distanceWidthRadius_____lineLength_10__distanceWidthRadius_2:         [...]
  ▶ distanceWidthRadius_____lineLength_10__thickness_1:                   [...]
  ▶ distanceWidthRadius_____lineLength_10__thickness_1__distanceWidthRadius_2:   [...]
  ▶ distanceWidthRadius_____size_200:                                     [...]
  ▶ distanceWidthRadius_____size_200__distanceWidthRadius_2:              [...]
  ▶ distanceWidthRadius_____size_200__lineLength_10:                      [...]
  ▶ distanceWidthRadius_____size_200__lineLength_10__distanceWidthRadius_2:   [...]
  ▶ distanceWidthRadius_____size_200__lineLength_10__thickness_1:         [...]
  ▶ distanceWidthRadius_____size_200__lineLength_10__thickness_1__distanceWidthRadius_2:   [...]
  ▶ distanceWidthRadius_____size_200__thickness_1:                        [...]
  ▶ distanceWidthRadius_____size_200__thickness_1__distanceWidthRadius_2: [...]
  ▶ distanceWidthRadius_____thickness_1:                                  [...]
```



```
▶ 1.json:                          {...}
▶ 10.json:                         {...}
▶ 100.json:                        {...}
▼ 101.json:
  ▼ iteration:
    ▼ 0:
      ▶ tuple__centerX:            {...}
      ▼ tuple__centerY:
        ▼ centerY:
            0:                     198.5705080756888
            1:                     211.0705080756888
            2:                     223.5705080756888
            3:                     236.0705080756888
            4:                     198.5705080756888
            5:                     211.0705080756888
            6:                     223.5705080756888
            7:                     236.0705080756888
```

Step wise logistic regression

Without images, on structured data – dependencies on recursion depth as separate columns

```
multinom(perceivedAesthetics ~ ., family=multinomial,
            data = variablePointDataTrain[usedColnames]) %>% stepAIC(trace = FALSE, direction="both")
```

Evaluating accuracy:

```
    print(mean(predictedValues == observedValues))
```

Restricted to the maximal number of 110 columns

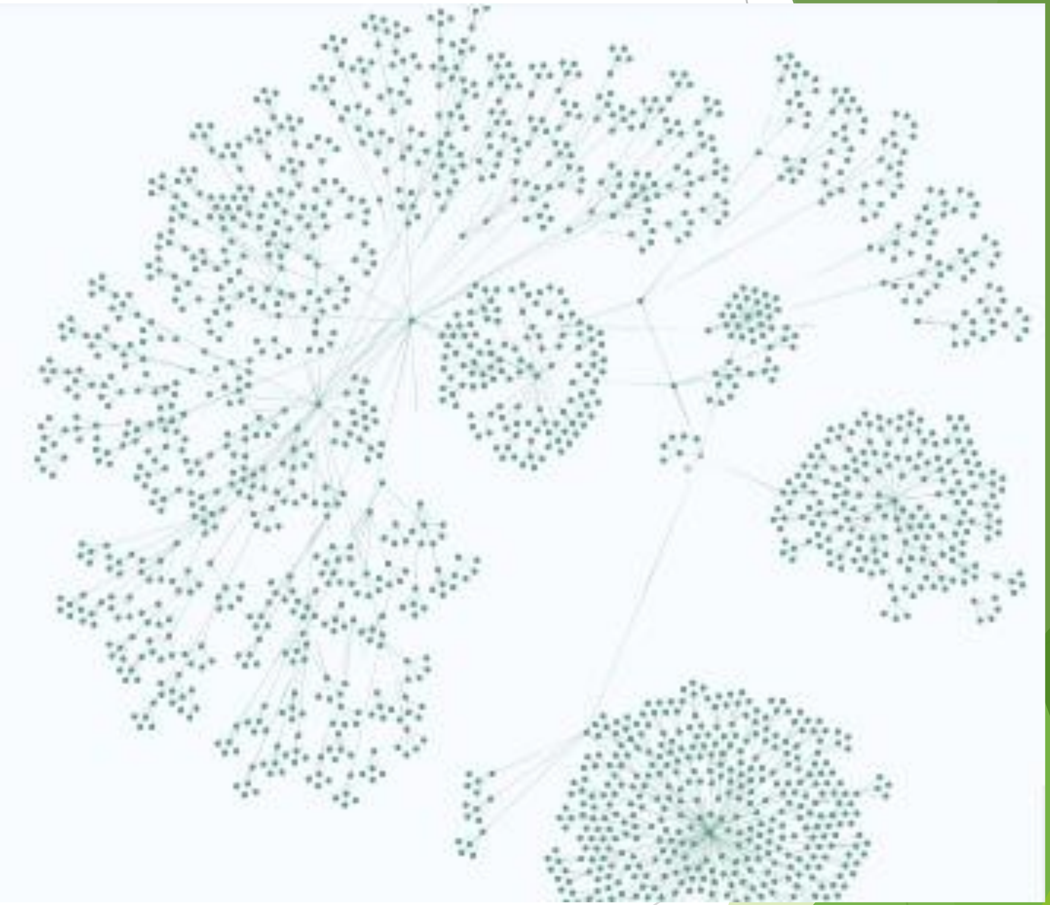
for this small evaluated dataset … **NOT ENOUGH**

Test ACC: 0.3421 GOOD

GNN – accuracy and loss

Without images, on graph data

```
Model: "gnn_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 preprocess (Sequential)     (38376, 32)               1396

 graph_conv1 (GraphConvLayer  multiple                  5888
 )

 graph_conv2 (GraphConvLayer  multiple                  5888
 )

 postprocess (Sequential)    (38376, 32)               2368

 logits (Dense)              multiple                  330

=================================================================
Total params: 15,870
Trainable params: 15,028
Non-trainable params: 842
_________________________________________________________________
```



- **Test ACC: 0.7133**

Formatted data according domain knowledge

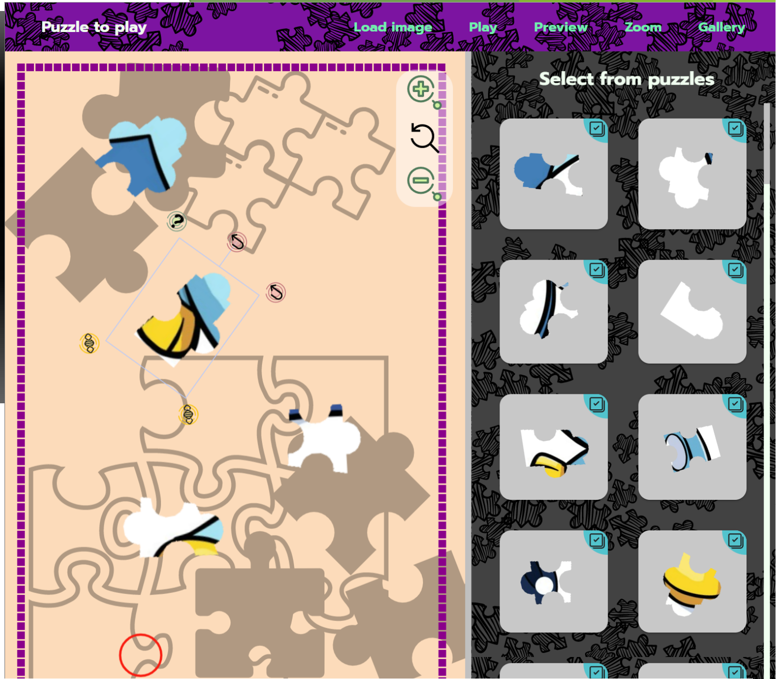
Finding the best model for aesthetics assignment

Used model	Accuracy one user	Accuracy AI LeNet model
LeNet (input size 28x28)	0,3158	0,8487
LeNet (input size 600x600)	0,3421	0,8618
LeNet multimodel for image with coordinates (input size 600x600)	0,2697	0,7961
Multinomial logistic regression based on coordinates	0,3618	0,8092
Stepwise logistic regression based on coordinates (backward)	0,3421	0,7894
Stepwise logistic regression based on coordinates (forward)	0,4539	0,7960
Stepwise logistic regression based on coordinates (both)	0,4539	0,7961
Graph neural network + coordinates	0,7133	0,6999

Results

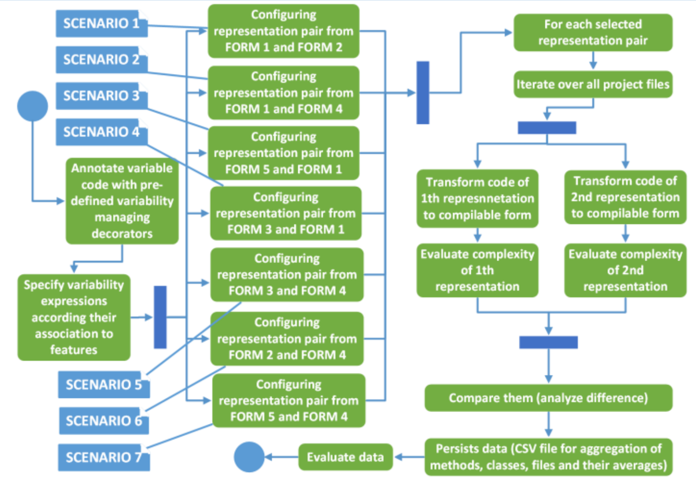
```
{
"AND": {
    "OR": {
        "variable1": "false",
        "AND": {
            "variable2": "true",
            "variable3": "true"
        }
    },
    "variable4": "true"
}
```

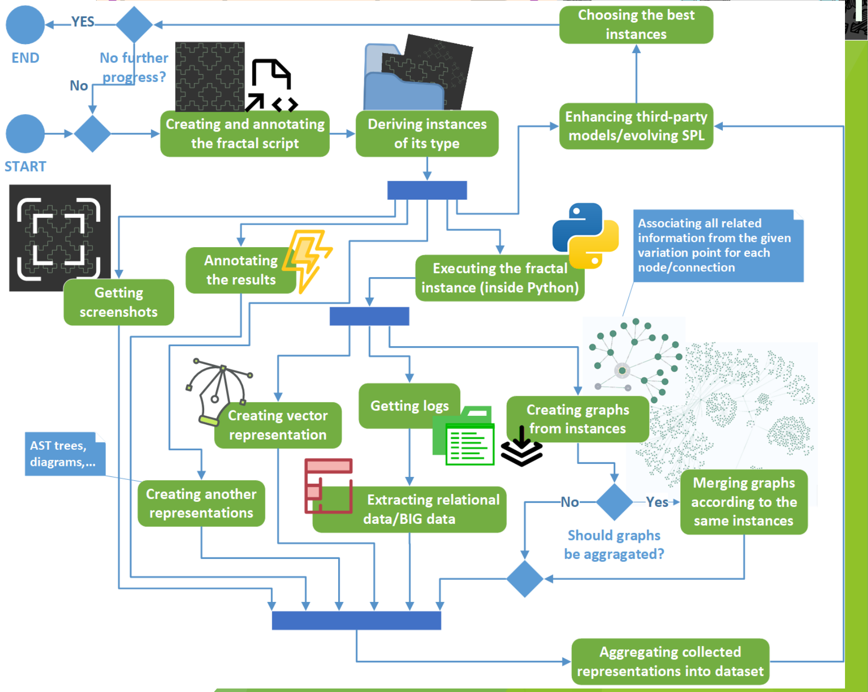
```
@Annotation.classVP()
    variable1 = -4;
    ...
```



The hierarchically-expressed representation of variation points effectively drives the development processes by forcing its use to build modular and reusable code fragments and enabling to automatically derive resulting products according to their concisely expressed configuration which is preserved in code with the possibility to model them dynamically, collect them into a dataset, select them, and iteratively customize them in the software product line evolution process according to structural and semantic knowledge.

FORM 1: *variability managing decorators are used as much as possible in a form that is possible to analyze*
FORM 2: *as form 1, but without variability configuration expressions*
FORM 3: *variability managing decorators are used only marginally in a form that is possible to analyze (as attr. decorators)*
FORM 4: *all variability annotations are removed, variable code is preserved, without unwanted code (applied only if possible)*
FORM 5: *as form 1, but with additional unwanted dead code constructs needed for illegal decorators*


SCENARIO 1
SCENARIO 2
SCENARIO 3
SCENARIO 4
Annotate variable code with pre-defined variability managing decorators
Specify variability expressions according their association to features
SCENARIO 5
SCENARIO 6
SCENARIO 7
Configuring representation pair from FORM 1 and FORM 2
Configuring representation pair from FORM 1 and FORM 4
Configuring representation pair from FORM 5 and FORM 1
Configuring representation pair from FORM 3 and FORM 1
Configuring representation pair from FORM 3 and FORM 4
Configuring representation pair from FORM 2 and FORM 4
Configuring representation pair from FORM 5 and FORM 4
For each selected representation pair
Iterate over all project files
Transform code of 1th represnnetation to compilable form
Transform code of 2nd representation to compilable form
Evaluate complexity of 1th representation
Evaluate complexity of 2nd representation
Compare them (analyze difference)
Persists data (CSV file for aggregation of methods, classes, files and their averages)
Evaluate data


END
YES
No further progress?
No
START
Creating and annotating the fractal script
Deriving instances of its type
Choosing the best instances
Enhancing third-party models/evolving SPL
Getting screenshots
Annotating the results
Executing the fractal instance (inside Python)
Associating all related information from the given variation point for each node/connection
Creating vector representation
Getting logs
Creating graphs from instances
AST trees, diagrams,...
Creating another representations
Extracting relational data/BIG data
No
Yes
Should graphs be aggregated?
Merging graphs according to the same instances
Aggregating collected representations into dataset

$$sim(i,j) = \underbrace{\alpha_{ij} * sim_{ij}^{in} + \beta_{ij} * sim_{ij}^{out}}_{\text{Structural information}} + \underbrace{\gamma_{ij} * sim_{ij}^{sem1} + ... + \epsilon_{ij} * sim_{ij}^{sem2}}_{\text{Semantic information}}$$

Resulting capabilities

- To study managed software product line evolution in its automated form
  - with the possibility to integrate it with available evolution algorithms
  - Applied principles of variability modeling, knowledge modeling, and feature interactions (from data of resulting products)
  - use machine learning/deep learning marginally by applying a wide range of features
  - fast and cheap way to observe different possibilities
- To study managed software product lines in large
- Possibility to analyze restricted use of annotations (our approach) applied in variation points and available actions to preserve modularity, native development (exchangeable with decorators in TypeScript), and comprehensive code
- Multi-content and Multi-Purpose dataset built from knowledge based on similarity in product family + capability to compare and design different models
  - No existing one which contains various formats accompanied with launchable applications/products exists (according to our observations on Kaggle or from the internet)
- Identified extensions to expressions inside annotations to fill gaps during product instantiation

Future work

- Automatically evolve fractal products with the help of the extracted knowledge
- Continue manually evolve stateful canvas-based SPL
- Evaluate recreated annotations into TypeScript decorators in comparison with code without them (modularity, coupling, and possible applications of aspects)
- Provide functionality to automatically insert these decorators into AST of TypeScript code
- Design other advanced models capable to evaluate quality according to the requirements including GANs and Transformers
- Tune mechanism to generate different semantic and structural views according to instantiated products
- Another possibility: Model a given knowledge further (in knowledge bases)

What next?

- Implement and compare other mechanisms for variability management such as pure::variants
- Extend the solution to support new variants and evaluate its quality
- Build GAN to generate similar fractals – analyze the impact of the product in SPL evolution
  - try to design variation points based on the best ones

Published and presented articles on conferences

MADEISD 2023, SQAMIA 2023

- J. Perdek and V. Vranić. Lightweight Aspect-Oriented Software Product Lines with Automated Product Derivation. 5th Workshop on Modern Approaches in Data Engineering and Information System Design, MADEISD 2023, a part of 27th European Conference on Advances in Databases and Information Systems, ADBIS 2023. Barcelona, Spain, 2023. Accepted (A-).
- J. Perdek and V. Vranić. Matrix Based Approach for Structural and Semantic Analysis Supporting Software Product Line Evolution. 10th Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, SQAMIA 2023. Bratislava, Slovakia, 2023. Accepted (A-).

Bibliography

- BEUCHE, Danilo a Mark DALGARNO, 2006. Software Product Line Engineering with Feature Models. 2006, s. 7.
- BOTTERWECK, Goetz, Kwanwoo LEE a Steffen THIEL, 2009. Automating Product Derivation in Software Product Line Engineering. 2009, s. 6.
- KASTNER, Christian, Sven APEL a Don BATORY, 2007. A Case Study Implementing Features Using AspectJ. V: *11th International Software Product Line Conference (SPLC 2007)*: *11th International Software Product Line Conference (SPLC 2007)* [online]. Kyoto, Japan: IEEE, s. 223–232 [cit. 30.9.2021]. ISBN 978-0-7695-2888-5. Dostupné na: doi:10.1109/SPLINE.2007.12
- LADDAD, Ramnivas, 2003. *AspectJ in action: practical aspect-oriented programming*. Greenwich, CT: Manning. ISBN 978-1-930110-93-9.
- PELÁNEK, Radek, 2012. *Programátorská cvičebnice*. 1. vydání. Brno: Computer press. ISBN 978-80-251-3751-2.
- VRANIC, Valentino a Roman TÁBORSKÝ, 2016. Features as transformations: A generative approach to software development. *Computer Science and Information Systems* [online]. 2016, roč. 13, č. 3, s. 759–778. ISSN 1820-0214, 2406-1018. Dostupné na: doi:10.2298/CSIS160128027V
- YOUNG, Trevor J a B MATH, 1999. Using AspectJ to Build a Software Product Line for Mobile Devices. 1999, s. 73.

- Mohammad Abu-Matar and Hassan Gomaa. 2011. Variability Modeling for Service Oriented Product Line Architectures. In 2011 15th International Software Product Line Conference. IEEE, Munich, Germany, 110–119. https://doi.org/10.1109/SPLC.2011.26
- Hwi Ahn and Sungwon Kang. 2011. Analysis of Software Product Line Architecture Representation Mechanisms. In 2011 Ninth International Conference on Software Engineering Research, Management and Applications. IEEE, Baltimore, MD, USA, 219–226. https://doi.org/10.1109/SERA.2011.22
- S.A. Ajila. 2005. Reusing Base-product Features to develop Product Line Architecture. In IRI -2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005. IEEE, Las Vegas, NV, USA, 288–293. https://doi.org/10.1109/ IRI-05.2005.1506488
- Samuel A Ajila and Patrick J Tierney. 2002. The FOOM Method – Modeling Software Product Lines in Industrial Settings. (2002), 11.
- Vander Alves, Pedro Matos Jr, and Paulo Borba. 2004. An Incremental Aspect-Oriented Product Line Method for J2ME Game Development. (2004), 3.
- Vander Alves, Pedro Matos, Leonardo Cole, Alexandre Vasconcelos, Paulo Borba, and Geber Ramalho. 2007. Extracting and Evolving Code in Product Lines with Aspect-Oriented Programming. In Transactions on Aspect-Oriented Software Development IV, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Awais Rashid, and Mehmet Aksit (Eds.). Vol. 4640. Springer Berlin Heidelberg, Berlin, Heidelberg, 117–142. https://doi.org/10.1007/978-3-540-77042-8_5 Series Title: Lecture Notes in Computer Science.

- Fazal-e Amin, Ahmad Kamil Mahmood, and Alan Oxley. 2010. A Review on Aspect Oriented Implementation of Software Product Lines Components. Information Technology Journal 9, 6 (Aug. 2010), 1262–1269. https://doi.org/10.3923/itj.2010.1262.1269
- Michalis Anastasopoulos and Dirk Muthig. 2004. An Evaluation of Aspect-Oriented Programming as a Product Line Implementation Technology. In Software Reuse: Methods, Techniques, and Tools, Jan Bosch and Charles Krueger (Eds.). Vol. 3107. Springer Berlin Heidelberg, Berlin, Heidelberg, 141–156. https://doi.org/10.1007/978-3-540-27799-6_12 Series Title: Lecture Notes in Computer Science
- Sven Apel, Thomas Leich, and Gunter Saake. 2006. Aspectual mixin layers: aspects and features in concert. In Proceedings of the 28th international conference on Software engineering. ACM, Shanghai China, 122–131. https://doi.org/10.1145/1134285.1134304
- U. Aßmann. 2003. Invasive Software Composition. Springer-Verlag, Berlin, Heidelberg
- M.A. Babar. 2004. Scenarios, Quality Attributes, and Patterns: Capturing and Using their Synergistic Relationships for Product Line Architectures. In 11th Asia-Pacific Software Engineering Conference. IEEE, Busan, Korea, 574–578. https://doi.org/10.1109/APSEC.2004.91
- Felix Bachmann and Len Bass. 2001. Managing Variability in Software Architectures. (2001), 7.
- L. Balzerani, D. Di Ruscio, A. Pierantonio, and G. De Angelis. 2005. A product line architecture for web applications. In Proceedings of the 2005 ACM symposium on Applied computing - SAC ’05. ACM Press, Santa Fe, New Mexico, 1689. https://doi.org/10.1145/1066677.1067059

- Gérald Barré. 2018. Aspect Oriented Programming in TypeScript. https://www.meziantou.net/aspect-oriented-programmingin-typescript.htm
- Don Batory, Rich Cardone, and Yannis Smaragdakis. 2000. Object-Oriented Frameworks and Product Lines. In Software Product Lines, Patrick Donohoe (Ed.). Springer US, Boston, MA, 227–247. https://doi.org/10.1007/978-1-4615-4339-8_13
- Joachim Bayer, Oliver Flege, and Cristina Gacek. 2000. Creating Product Line Architectures. In Software Architectures for Product Families, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Frank van der Linden (Eds.). Vol. 1951. Springer Berlin Heidelberg, Berlin, Heidelberg, 210–216. https://doi.org/10.1007/978-3-540-44542-5_23 Series Title: Lecture Notes in Computer Science.
- Ivo Augusto Bertoncello, Marcelo Oliveira Dias, Patrick H. S. Brito, and Cecília M. F. Rubira. 2008. Explicit exception handling variability in component-based product line architectures. In Proceedings of the 4th international workshop on Exception handling - WEH '08. ACM Press, Atlanta, Georgia, 47–54. https://doi.org/10.1145/1454268.1454275
- Vinicius Bischoff, Kleinner Farias, Lucian José Gonçales, and Jorge Luis Victória Barbosa. 2019. Integration of feature models: A systematic mapping study. Information and Software Technology 105 (Jan. 2019), 209–225. https://doi.org/10. 1016/j.infsof.2018.08.016
- Lynne Blair and Jianxiong Pang. 2003. Aspect-Oriented Solutions to Feature Interaction Concerns using AspectJ. (2003), 17.
- Jan Bosch. 2000. Design & Use of Software Architectures—Adopting and Evolving a Product Line Approach.

- Jan Bosch, Gert Florijn, Danny Greefhorst, Juha Kuusela, J. Henk Obbink, and Klaus Pohl. 2002. Variability Issues in Software Product Lines. In Software Product-Family Engineering, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Frank van der Linden (Eds.). Vol. 2290. Springer Berlin Heidelberg, Berlin, Heidelberg, 13–21. https://doi.org/10.1007/3-540-47833-7_3 Series Title: Lecture Notes in Computer Science

- Jonathan Cardoso. 2021. How To Use Decorators in TypeScript. https://www.digitalocean.com/community/tutorials/howto-use-decorators-in-typescript

- João M.P. Cardoso, Tiago Carvalho, José G.F. Coutinho, Wayne Luk, Ricardo Nobre, Pedro Diniz, and Zlatko Petrov. 2012. LARA: an aspect-oriented programming language for embedded systems. In Proceedings of the 11th annual international conference on Aspect-oriented Software Development - AOSD ’12. ACM Press, Potsdam, Germany, 179. https://doi.org/10.1145/2162049.2162071

- Adrian Colyer, Awais Rashid, and Gordon Blair. 2004. On the Separation of Concerns in Program Families. (2004), 11

- Tung M. Dao and Kyo C. Kang. 2010. Mapping Features to Reusable Components: A Problem Frames-Based Approach. In Software Product Lines: Going Beyond, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Jan Bosch, and Jaejoon Lee (Eds.). Vol. 6287. Springer Berlin Heidelberg, Berlin, Heidelberg, 377–392. https://doi.org/10.1007/978-3-642-15579-6_26 Series Title: Lecture Notes in Computer Science.

- Ebru Dincel, Nenad Medvidovic, and André van der Hoek. 2002. Measuring Product Line Architectures. In Software Product-Family Engineering, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Frank van der Linden (Eds.). Vol. 2290. Springer Berlin Heidelberg, Berlin, Heidelberg, 346–352. https://doi.org/10.1007/3-540-47833-7_31 Series Title: Lecture Notes in Computer Science.

- Chethana Kuloor Armin Eberlein. 2002. Requirements Engineering for Software Product Lines. (2002), 12

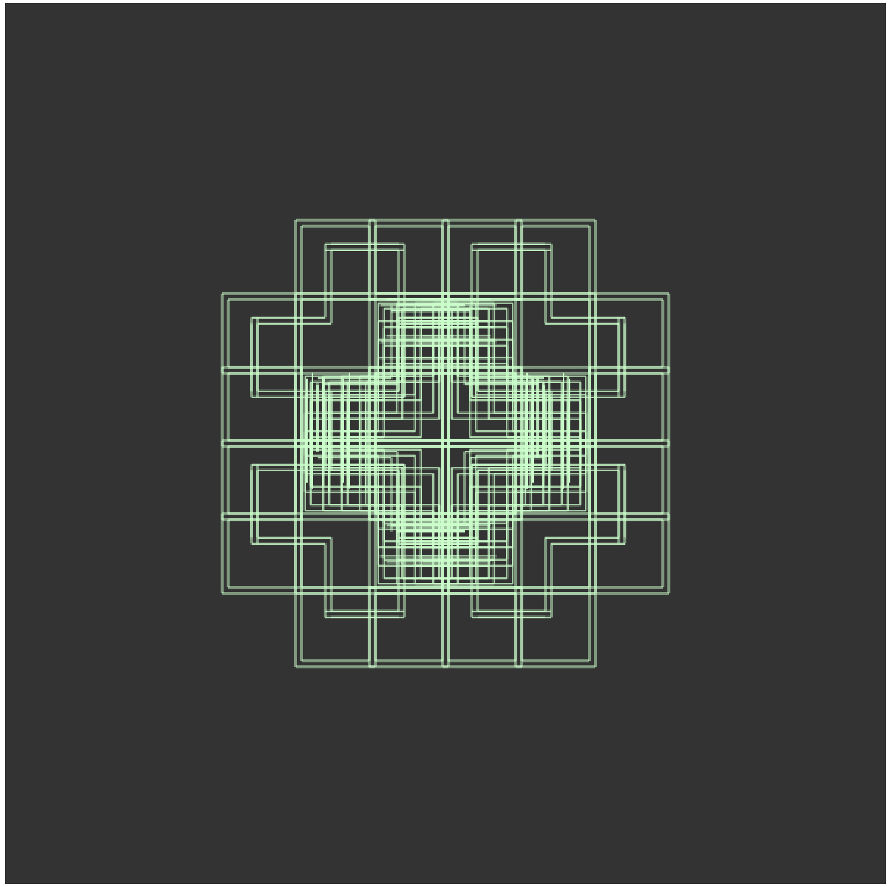
- Eun Sook Cho, Min Sun Kim, and Soo Dong Kim. 2001. Component metrics to measure component quality. In Proceedings Eighth Asia-Pacific Software Engineering Conference. IEEE Comput. Soc, Macao, China, 419–426. https://doi.org/10.1109/ APSEC.2001.991509
- Eduardo Figueiredo, Nelio Cacho, Claudio Sant'Anna, Mario Monteiro, Uira Kulesza, Alessandro Garcia, Sergio Soares, Fabiano Ferrari, Safoora Khan, Fernando Castor Filho, and Francisco Dantas. 2008. Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability. (2008), 10.
- Robert E. Filman and Daniel P. Friedman. 2000. Aspect-Oriented Programming is Quantification and Obliviousness. In Proceedings of the Workshop on Advanced Separation of Concerns in Object-Oriented Systems, ACM Conference on ObjectOriented Programming, Systems, Languages, and Applications, OOPSLA 2000. Minneapolis, Minnesota USA. RIACS Technical Report 01.12, 2001.
- Critina Gacek and Michalis Anastasopoules. 2001. Implementing product line variabilities. In Proceedings of the 2001 symposium on Software reusability putting software reuse in context - SSR '01. ACM Press, Toronto, Ontario, Canada, 109–117. https://doi.org/10.1145/375212.375269
- R.L. Glass and I. Vessey. 1998. Focusing on the application domain: everyone agrees it's vital, but who's doing anything about it?. In Proceedings of the Thirty-First Hawaii International Conference on System Sciences, Vol. 3. IEEE Comput. Soc, Kohala Coast, HI, USA, 187–196. https://doi.org/10.1109/HICSS.1998.656141
- Sebastian Gunther and Thorsten Berger. 2008. Service-Oriented Product Lines: Towards a Development Process and Feature Management Model for Web Services. (2008), 6.
- Stefan Hanenberg, Christian Oberschulte, and Rainer Unland. 2003. Refactoring of Aspect-Oriented Software. (2003), 18.

- Jan Hannemann and Gregor Kiczales. 2002. Design Pattern Implementation in Java and AspectJ. (Nov. 2002), 13.
- Wenhao Huang, Chengwan He, and Zheng Li. 2015. A Comparison of Implementations for Aspect-Oriented JavaScript:. Zhengzhou, China. https://doi.org/10.2991/csic-15.2015.9
- Renien John Joseph. 2015. Single Page Application and Canvas Drawing. International journal of Web & Semantic Technology 6, 1 (Jan. 2015), 29–37. https://doi.org/10.5121/ijwest.2015.6103
- Critina Gacek and Michalis Anastasopoules. 2001. Implementing product line variabilities. In Proceedings of the 2001 symposium on Software reusability putting software reuse in context - SSR ’01. ACM Press, Toronto, Ontario, Canada, 109–117. https://doi.org/10.1145/375212.375269
- K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report. Carnegie-Mellon University Software Engineering Institute
- Christian Kastner, Sven Apel, and Don Batory. 2007. A Case Study Implementing Features Using AspectJ. In 11th International Software Product Line Conference (SPLC 2007). IEEE, Kyoto, Japan, 223–232. https://doi.org/10.1109/SPLINE.2007.12
- Elizabeth A Kendall. 1999. Role Model Designs and Implementations with Aspect-oriented Programming. (1999), 17

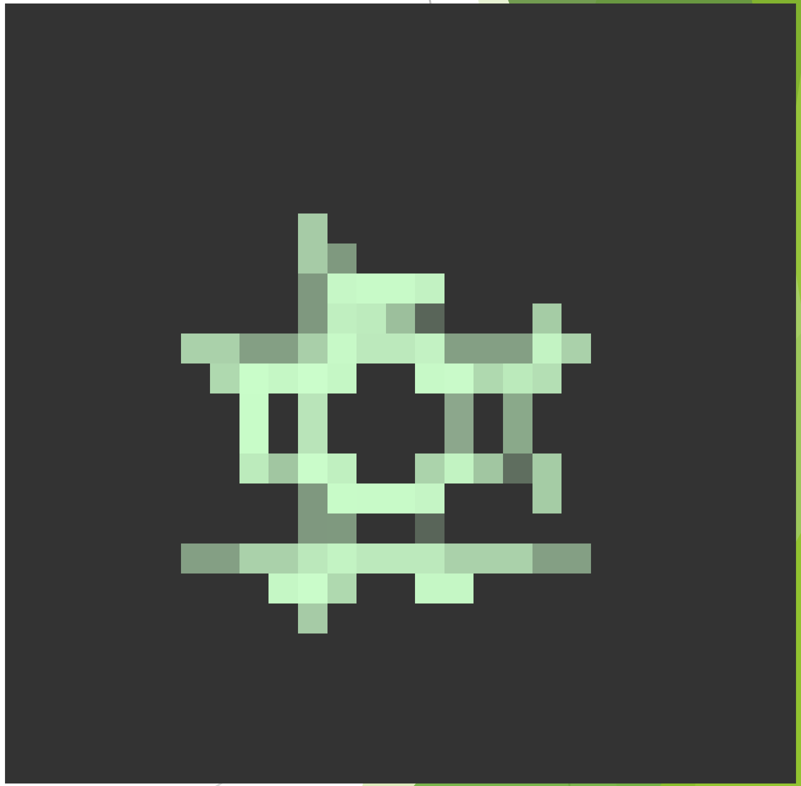
- Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. 2001. An Overview of AspectJ. In ECOOP 2001 — Object-Oriented Programming, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Jørgen Lindskov Knudsen (Eds.). Vol. 2072. Springer Berlin Heidelberg, Berlin, Heidelberg, 327–354. https://doi.org/10. 1007/3-540-45337-7_18 Series Title: Lecture Notes in Computer Science.
- Jan Kohut and Valentino Vranic. 2010. Guidelines for using aspects in product lines. In 2010 IEEE 8th International Symposium on Applied Machine Intelligence and Informatics (SAMI). IEEE, Herlany, 183–188. https://doi.org/10.1109/SAMI.2010. 5423741

Evaluating aesthetic perception on third party model - bias

What model sees:



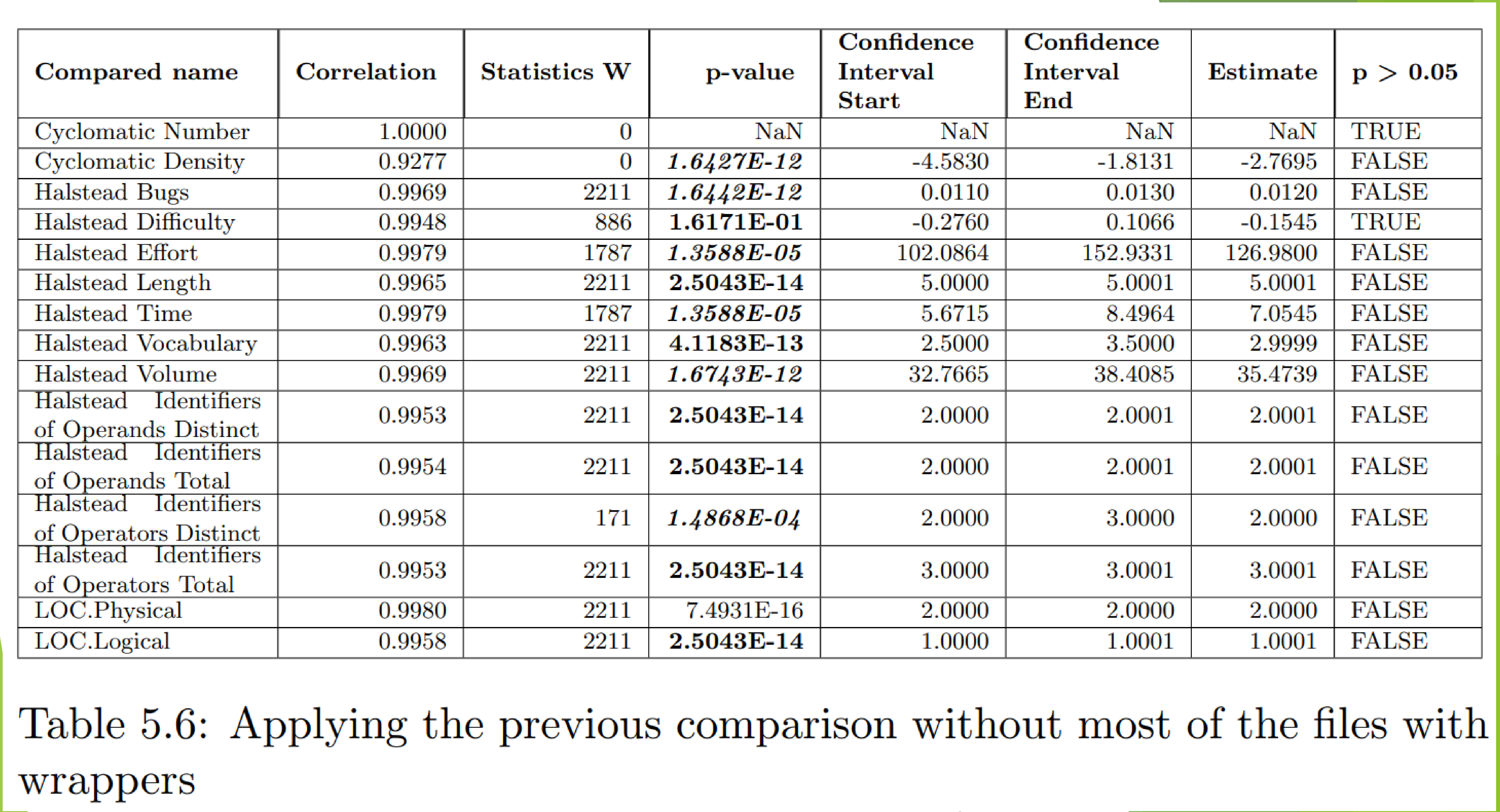
28 X 28 px



600 X 600 px

https://github.com/vatsal-rooprai/Image-Aesthetic-Evaluation

The **hierarchically-expressed** representation of *variation points* effectively drives the development processes **by forcing its use** to build **modular and reusable code fragments** and enabling to **automatically** derive resulting products according to their **concisely expressed configuration** which is **preserved in code** with the possibility to **model them dynamically**, **collect them into a dataset**, **select them**, and **iteratively** **customize them** in the **software product line evolution** process according to **structural and semantic knowledge**

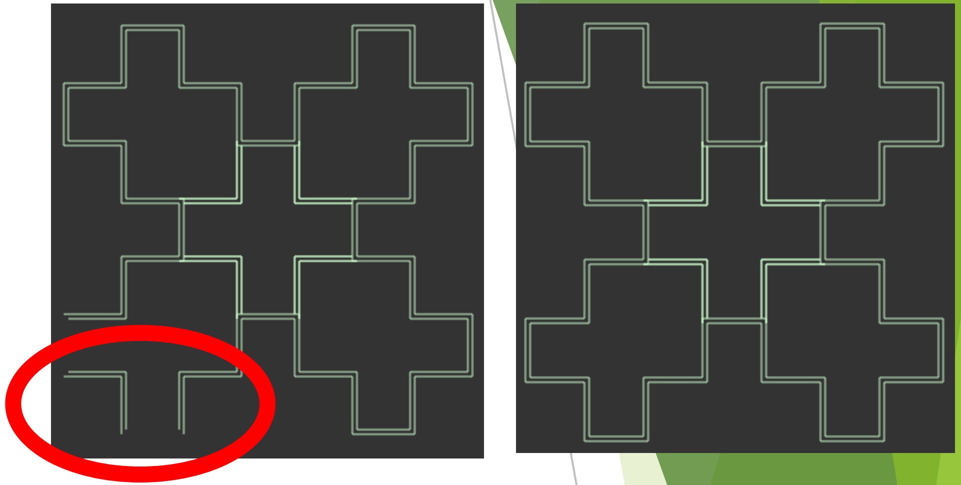


Compared name	**Correlation**	**Statistics W**	**p-value**	**Confidence Interval Start**	**Confidence Interval End**	**Estimate**	**p > 0.05**
Cyclomatic Number	1.0000	0	NaN	NaN	NaN	NaN	TRUE
Cyclomatic Density	0.9277	0	***1.6427E-12***	-4.5830	-1.8131	-2.7695	FALSE
Halstead Bugs	0.9969	2211	***1.6442E-12***	0.0110	0.0130	0.0120	FALSE
Halstead Difficulty	0.9948	886	**1.6171E-01**	-0.2760	0.1066	-0.1545	TRUE
Halstead Effort	0.9979	1787	***1.3588E-05***	102.0864	152.9331	126.9800	FALSE
Halstead Length	0.9965	2211	**2.5043E-14**	5.0000	5.0001	5.0001	FALSE
Halstead Time	0.9979	1787	***1.3588E-05***	5.6715	8.4964	7.0545	FALSE
Halstead Vocabulary	0.9963	2211	**4.1183E-13**	2.5000	3.5000	2.9999	FALSE
Halstead Volume	0.9969	2211	***1.6743E-12***	32.7665	38.4085	35.4739	FALSE
Halstead Identifiers of Operands Distinct	0.9953	2211	**2.5043E-14**	2.0000	2.0001	2.0001	FALSE
Halstead Identifiers of Operands Total	0.9954	2211	**2.5043E-14**	2.0000	2.0001	2.0001	FALSE
Halstead Identifiers of Operators Distinct	0.9958	171	***1.4868E-04***	2.0000	3.0000	2.0000	FALSE
Halstead Identifiers of Operators Total	0.9953	2211	**2.5043E-14**	3.0000	3.0001	3.0001	FALSE
LOC.Physical	0.9980	2211	7.4931E-16	2.0000	2.0000	2.0000	FALSE
LOC.Logical	0.9958	2211	**2.5043E-14**	1.0000	1.0001	1.0001	FALSE

Table 5.6: Applying the previous comparison without most of the files with wrappers

Results after SPL creation

doubled W-curves

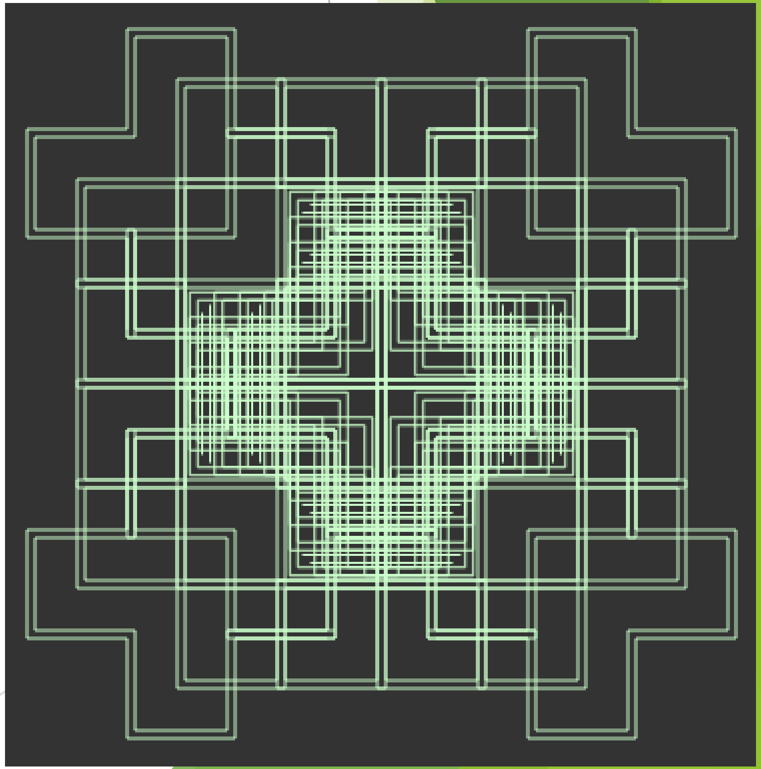


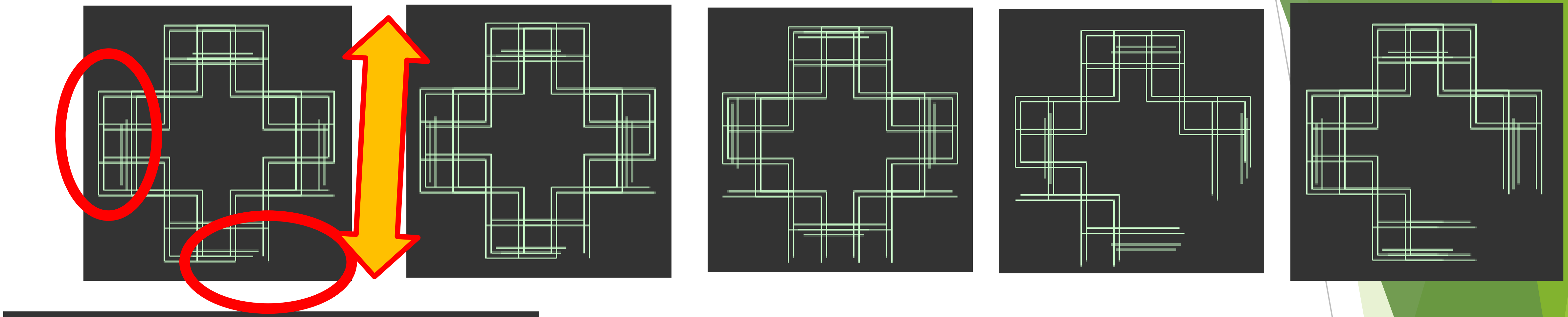
- THE RESULT SHOULD REMAIN A FRACTAL
- SYMMETRY IS THE BEST
- ASYMMETRY OFTEN DOES NOT LOOK SO GOOD

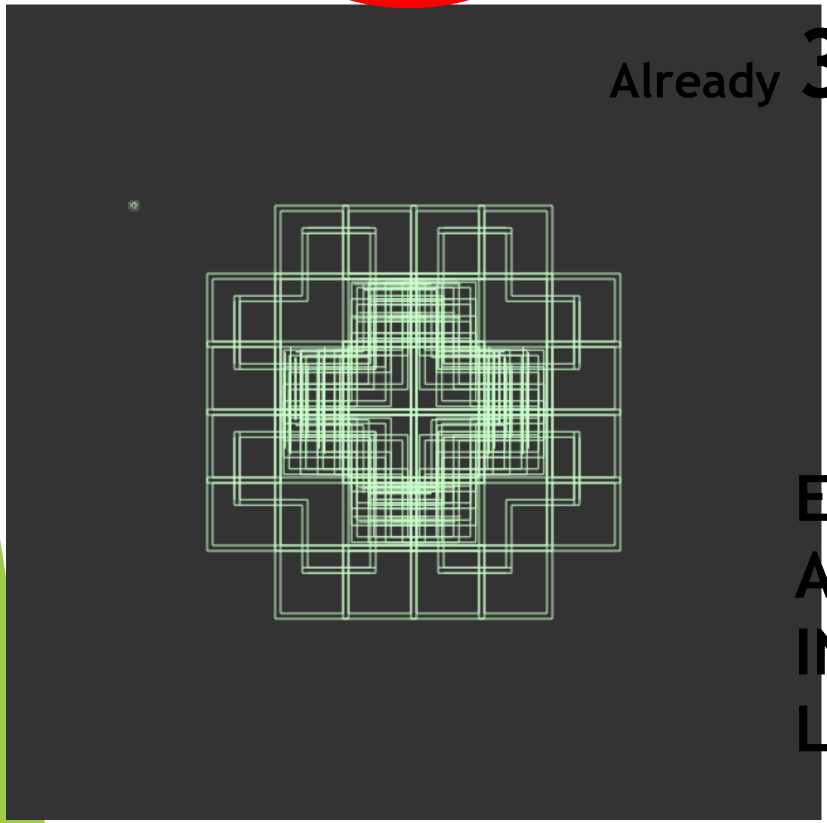
  Some results are enhanced,
  if another recursion functionality depends on it

- MAKING MORE INSTANCES OFTEN RESULTS IN CHAOS IN A FEW PLACES IN THE IMAGE

- LOGGING CAN PROVIDE „FACTORIALS“ OF DATA
- GENERATED DATA ARE ONLY IN FORM OF KEY-VALUE PAIRS







Already **378** fractals generated from one file

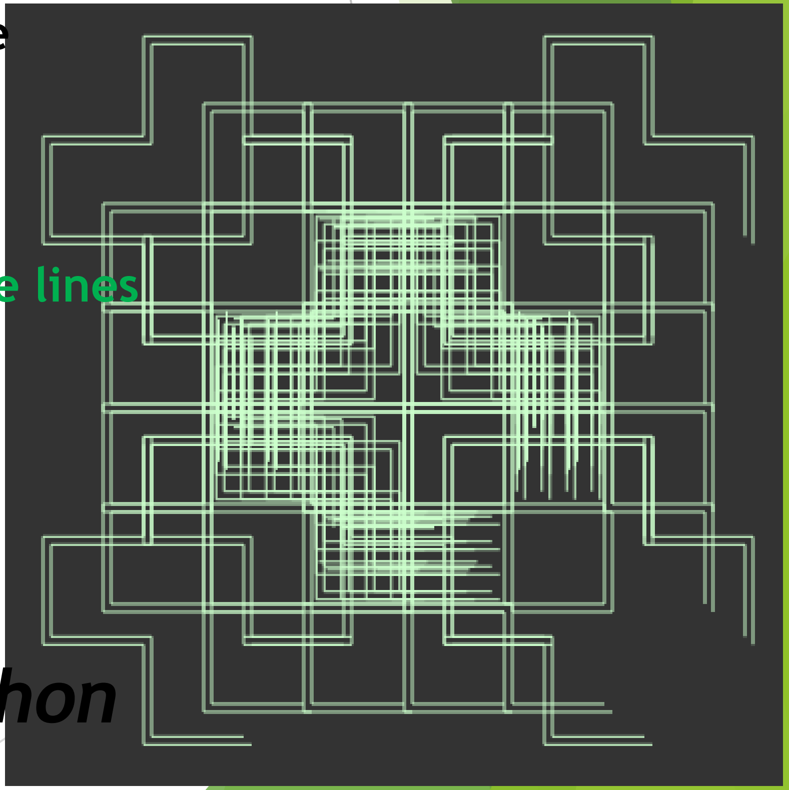
Can be more, but...

we bring:

asymmetry, chaos, standalone lines

creating non-fractal shape

EASY TO EXECUTE AND ANALYZE FRACTAL SCRIPT IN MANY PROGRAMMING LANGUAGES ***js2py for Python***



Results

The proper representation of software knowledge in place of variation points

essential information (knowledge) about software put inside annotation or found in their place (a form of tracing from lit.)

Annotated by our annotations

drives the

effective modularization and reuse

of software parts in the form of

automatically derived
*resulting **products***

Restricted use of our annotations (their actions from lit.)
to force organize variable code in a native and modular way
in parallel with the help of the aspects

The derivation process is automated

while the

subsequent extraction

with optional aggregation

of this

The mechanism is adapted to extract given information from code fragment (also dynamic one)

Also knowledge from heterogeneous applications can be analyzed with the rest of the software family

knowledge and associated information
supports decision-making about the evolution

of the software product line mainly based on

differences between variants

Knowledge can be connected and used in various models that are designed for automated decision-making about SPL evolution, its evaluation

Knowledge mainly captures differences between members

Studying the SPL evolution and variability

Less rigorous evaluations of variability management

→

General handling of the variability is still not fully covered/supported by variability management

-knowledge modeling,
-applying principles of variability modeling
-simulating feature interactions

…to handle variability

M. Galster, D. Weyns, D. Tofan, B. Michalik, and P. Avgeriou. Variability in software systems—a systematic literature review. IEEE Transactions on Software Engineering, 40(3):282–306, 2014.

various models and data representations are required for this purpose

Other possibilities

- Data should be used further to ***detect defects*** and ***provide quality assurance*** between selected variants

L. Chen, M. Ali Babar, and N. Ali. Variability management in software product lines: A systematic review. pages 81–90, 01 2009.

Evaluation of our Angular SPL

$$SSC = \frac{|Cc|}{|Cc|+|Cv|} = \frac{29}{29+40} = 0{,}4209$$

The more commonality, the…

BETTER REUSE OF ASSETS ACROSS PRODUCT FAMILY MEMBERS (PRODUCTS)

$$SVC = \frac{|Cv|}{|Cv|+|Cc|} = \frac{40}{40+29} = 0{,}57971$$

The more variability, the…

BETTER USER MENTAL MODEL SUPPORT

If some features are conditionally common, then in our assumption are evaluated as variable

Sum of components k

$$RBR = \frac{\Sigma_k \text{ Cost } C_k}{\Sigma_j \text{ Cost } C}$$

Sum of all components in SPL

Value of given component C is measured by LOC (the lines of code)

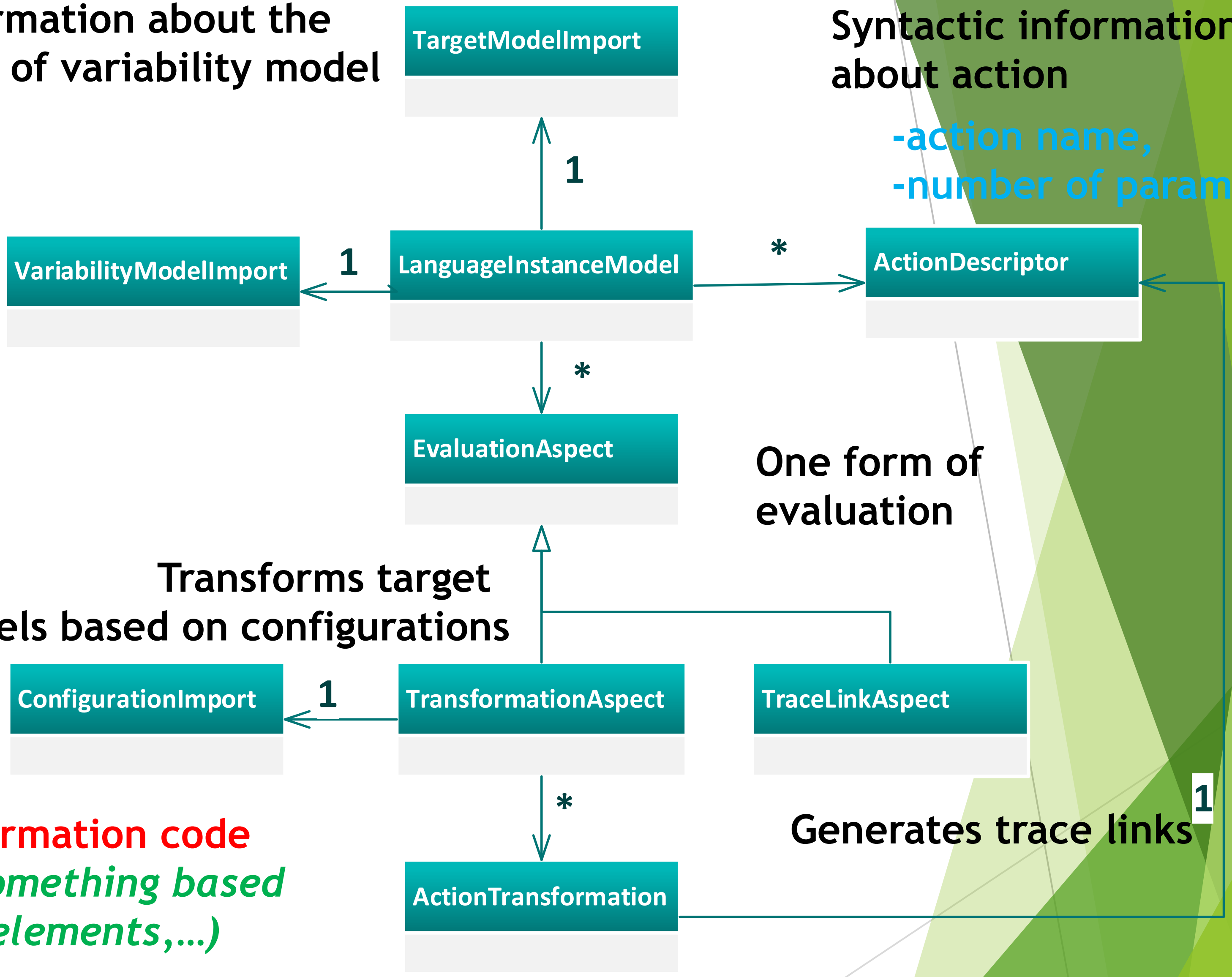
TypeScript code (fc=3), ***template code*** (ft=2) ***styles*** (fs=0.25)

Additive results for variation points are shown in the next table

Agenda

- Software product lines – what are they used for?
- Motivation – research on variability in parallel with software quality, and extraction of knowledge from related products
- Resolving commonality and variability in TypeScript stateful applications
- Evaluating the effectiveness of software product line establishment
  - presented on prepared stateful canvas-based TypeScript SPA product line
- Supporting product line evolution by extraction and comprehension of knowledge from related software products (presented on the real use-case)
  - presented on prepared fractal recursion-based product line
- Various data representations of software product features and capabilities
- Evaluation of models for aesthetics assignment and quality of resulting products
- Results and future work, Bibliography

Meta-model for VML language instance descriptions



Information about the type of variability model

Syntactic information about action

-action name,
-number of parameters

One form of evaluation

Transforms target models based on configurations

Generates trace links

Additional information for action associated with transformation of target models

VML LANGUAGES

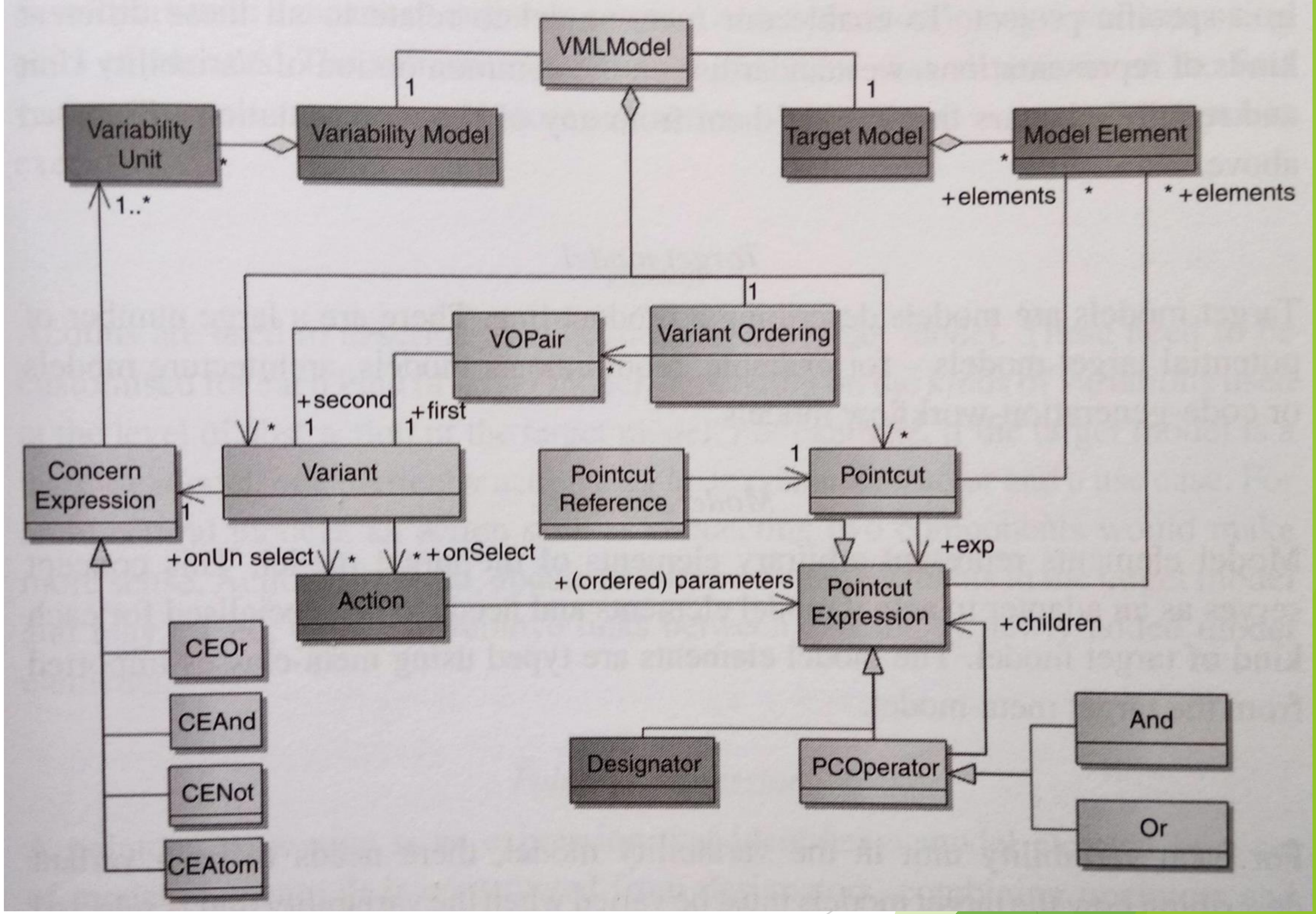
- VML4RE
- VML4Arch

Model-transformation code

(parser, something based on model elements,…)

Rashid, A., Royer, J., & Rummler, A. (Eds.). (2011). *Aspect-Oriented, Model-Driven Software Product Lines: The AMPLE Way*. Cambridge: Cambridge University Press. doi:10.1017/CBO9781139003629

Meta-model for variability management



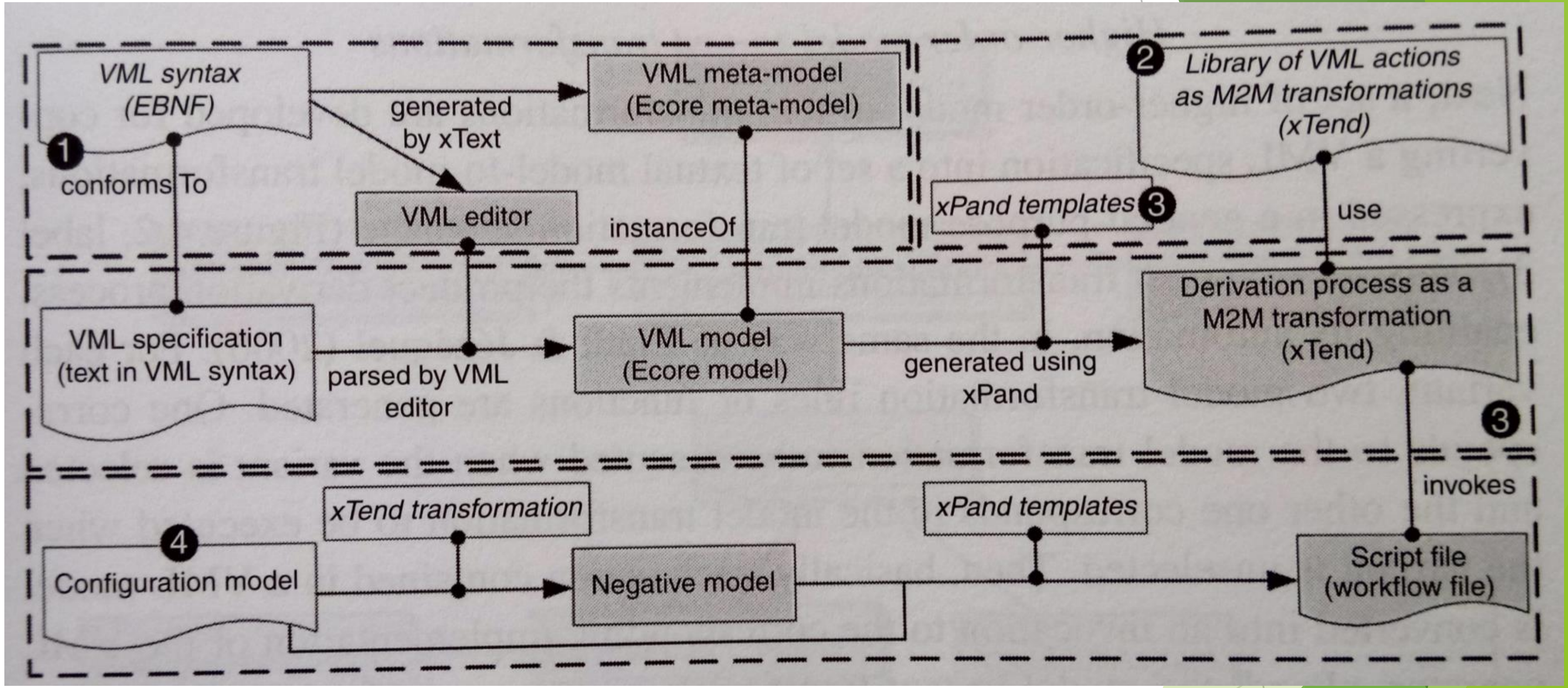
Rashid, A., Royer, J., & Rummler, A. (Eds.). (2011). *Aspect-Oriented, Model-Driven Software Product Lines: The AMPLE Way*. Cambridge: Cambridge University Press. doi:10.1017/CBO9781139003629

VML language - the process

TRANSFORMATION LANGUAGES

xTend – model-to-text

xPand – model-to-model



Rashid, A., Royer, J., & Rummler, A. (Eds.). (2011). *Aspect-Oriented, Model-Driven Software Product Lines: The AMPLE Way*. Cambridge: Cambridge University Press. doi:10.1017/CBO9781139003629

Domain analysis

Creation of features

Problems of given solution

- Created only mandatory features in a way that not provides:
  - product derivation
  - Voluntary features in configurable way
  - Hardcoded functionality – needs refactoring
  - Option to choose from options (in case of difficulty in game)
- Only console environment – (we will not remake)
- No code reuse – repetition on many places
- Still not extensive game –
  (but real application for given domain)
- Lack of encapsulation and object oriented features
  - Needs divide static method to appropriate classes
  - Needs manage access from parent object
  - Business concerns are not fully separated

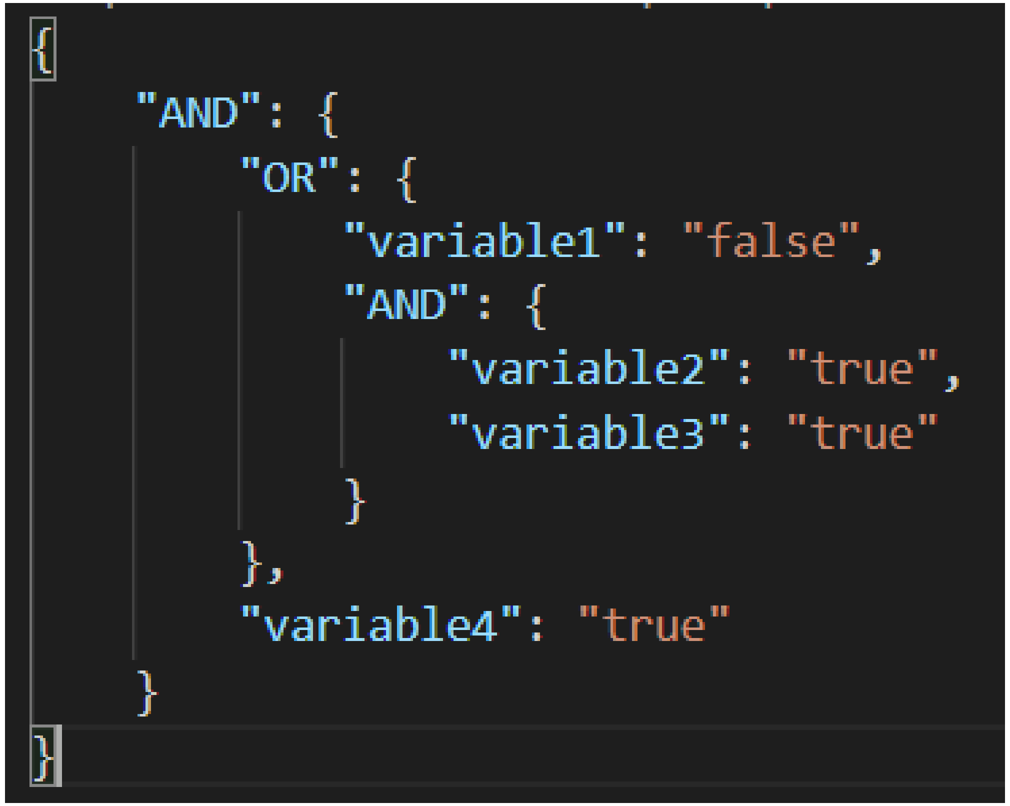
Design with aspects as voluntary functionality

- ***Aspect can be removed from execution - variable functionality***
- ***Aspect can intercepts points in execution and helps to derive product***

- **Good to extend functionality in various ways**
  - Add voluntary features
  - Choosing specific strategy from strategy options – from mandatory ones too
  - Enhance necessary functionality on existing classes (includes classes of additional features)

AND or OR JSON TREE

- (variable1 OR (Variable2 AND variable3)) AND variable4



```
{
    "AND": {
        "OR": {
            "variable1": "false",
            "AND": {
                "variable2": "true",
                "variable3": "true"
            }
        },
        "variable4": "true"
    }
}
```

2. If given variable variable1 is false in config then
OR is true, otherwise remaining branches should be true

1. If given variables in config are both true, then
AND above is true

3. If given variable variable4 is true in config and
whole OR is true, then parent AND is true

4. If whole is true, then we can copy annotated method

Applied annotations types

//@{}

For whole class/aspect/interface

Copying of whole file with class

//#{}

For class/aspect method only

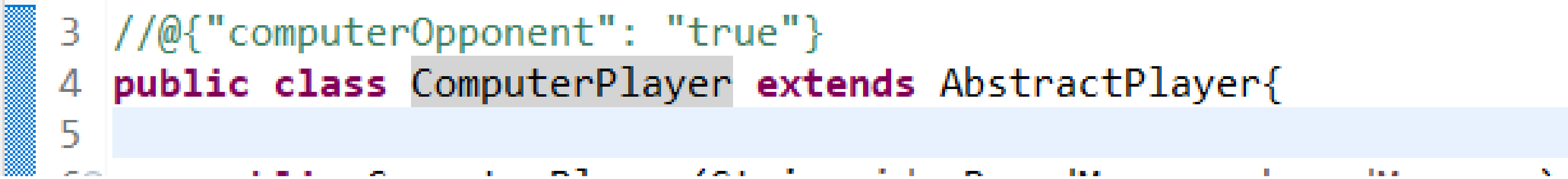
Copying of given method

//%{}

For import statement only

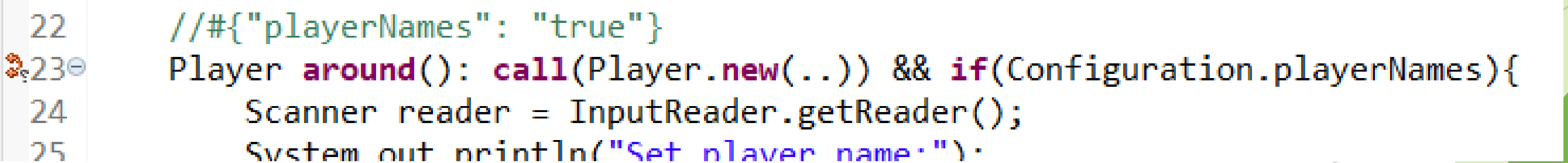
Copying of given import

//@{}



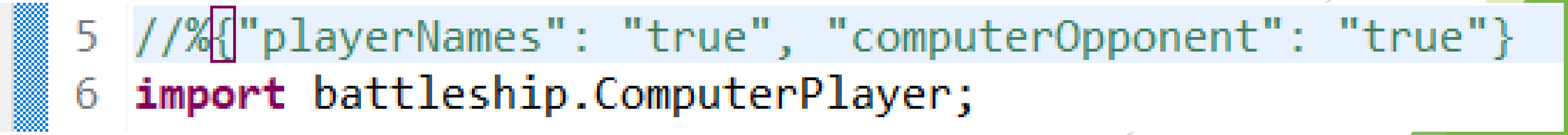
```
//@{"computerOpponent": "true"}
public class ComputerPlayer extends AbstractPlayer{

```

//#{}



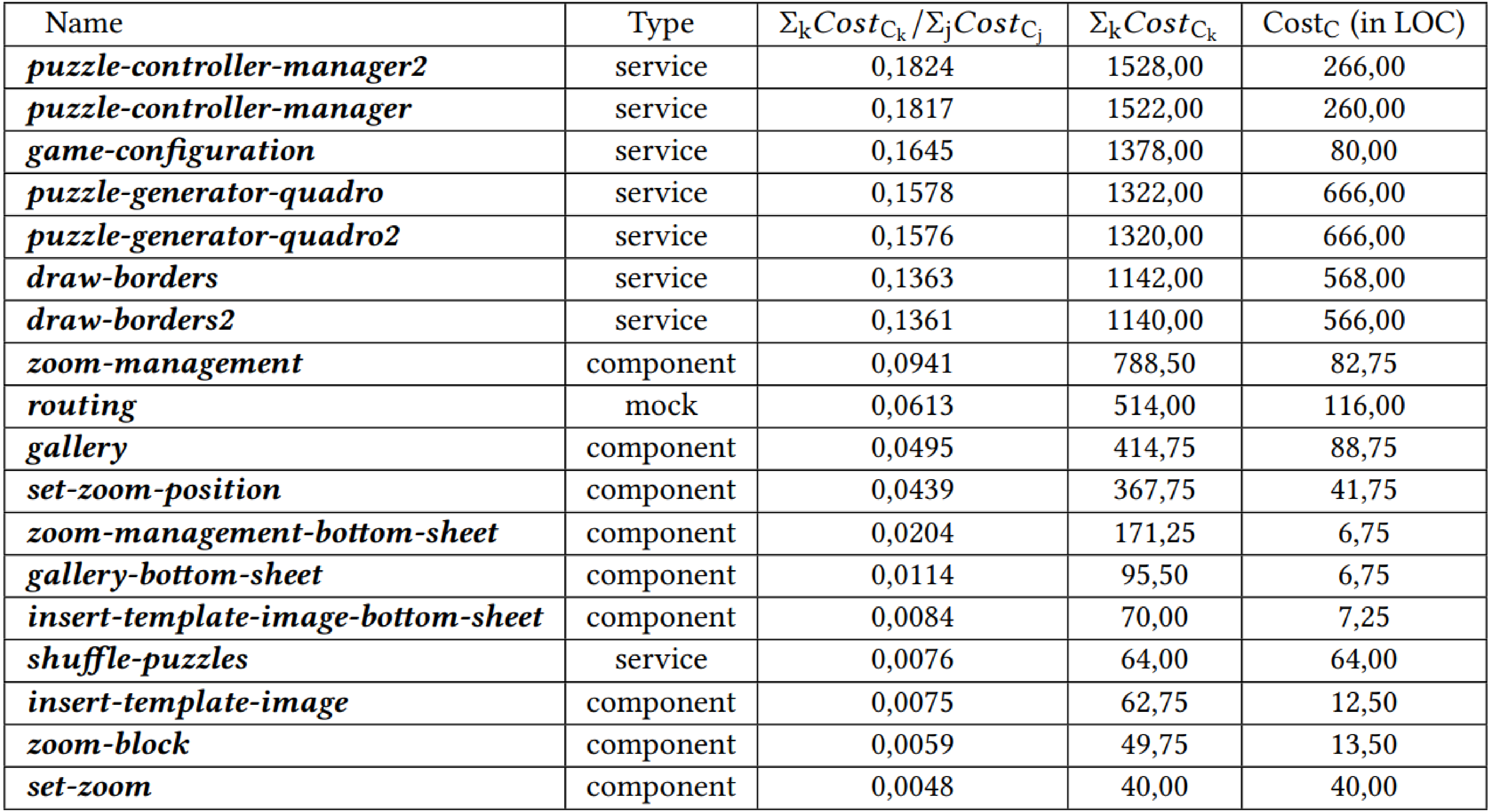
```
    //#{"playerNames": "true"}
    Player around(): call(Player.new(..)) && if(Configuration.playerNames){
        Scanner reader = InputReader.getReader();
        System.out.println("Set player name:");
```

//%{}



```
//%{"playerNames": "true", "computerOpponent": "true"}
import battleship.ComputerPlayer;
```

Evaluation



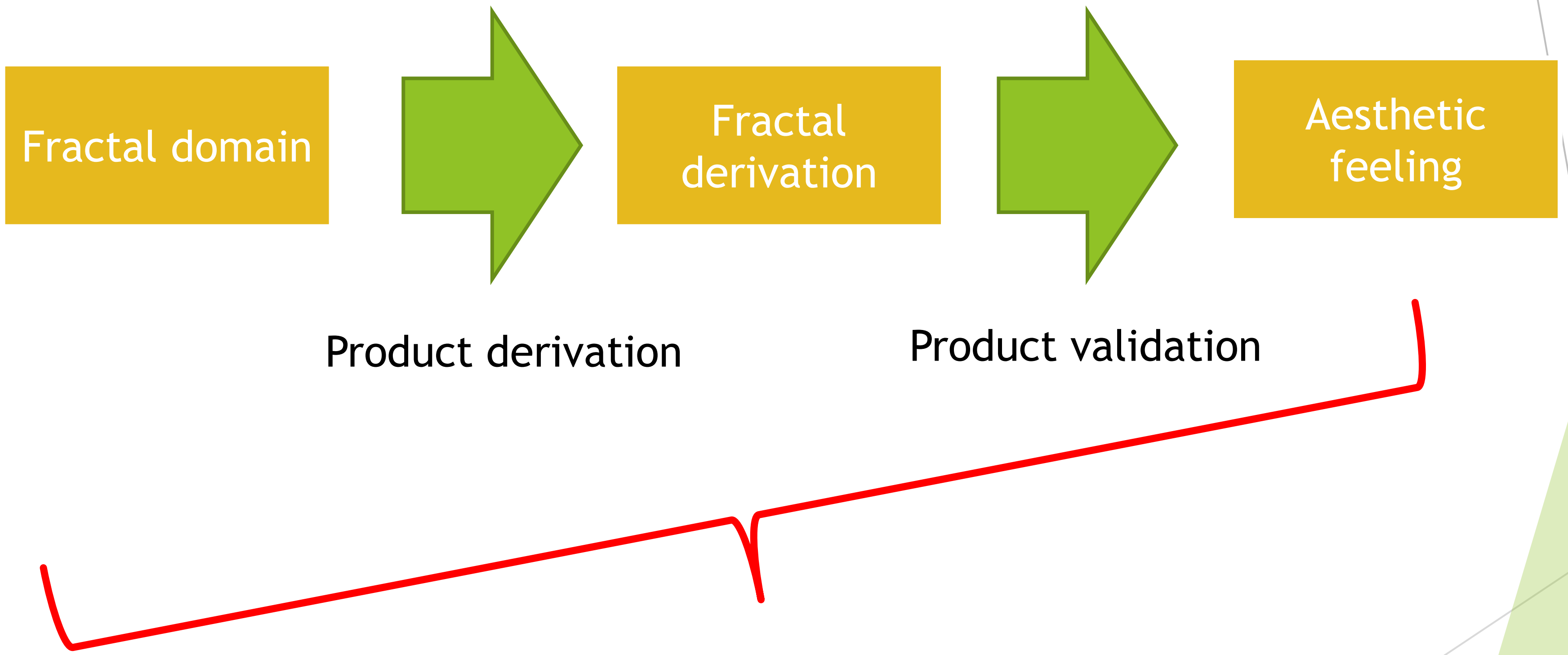
Name	Type	$\Sigma_k Cost_{C_k} / \Sigma_j Cost_{C_j}$	$\Sigma_k Cost_{C_k}$	$Cost_C$ (in LOC)
puzzle-controller-manager2	service	0,1824	1528,00	266,00
puzzle-controller-manager	service	0,1817	1522,00	260,00
game-configuration	service	0,1645	1378,00	80,00
puzzle-generator-quadro	service	0,1578	1322,00	666,00
puzzle-generator-quadro2	service	0,1576	1320,00	666,00
draw-borders	service	0,1363	1142,00	568,00
draw-borders2	service	0,1361	1140,00	566,00
zoom-management	component	0,0941	788,50	82,75
routing	mock	0,0613	514,00	116,00
gallery	component	0,0495	414,75	88,75
set-zoom-position	component	0,0439	367,75	41,75
zoom-management-bottom-sheet	component	0,0204	171,25	6,75
gallery-bottom-sheet	component	0,0114	95,50	6,75
insert-template-image-bottom-sheet	component	0,0084	70,00	7,25
shuffle-puzzles	service	0,0076	64,00	64,00
insert-template-image	component	0,0075	62,75	12,50
zoom-block	component	0,0059	49,75	13,50
set-zoom	component	0,0048	40,00	40,00

Table 1. The value of product line parts (chosen variation points).

Where The Cost of all components in SPL = Σ_j Cost C = 8378,25

Application on fractals

Many possible derivations of fractals



EXTENSIVE SOLUTION SPACE

A need for best product derivator

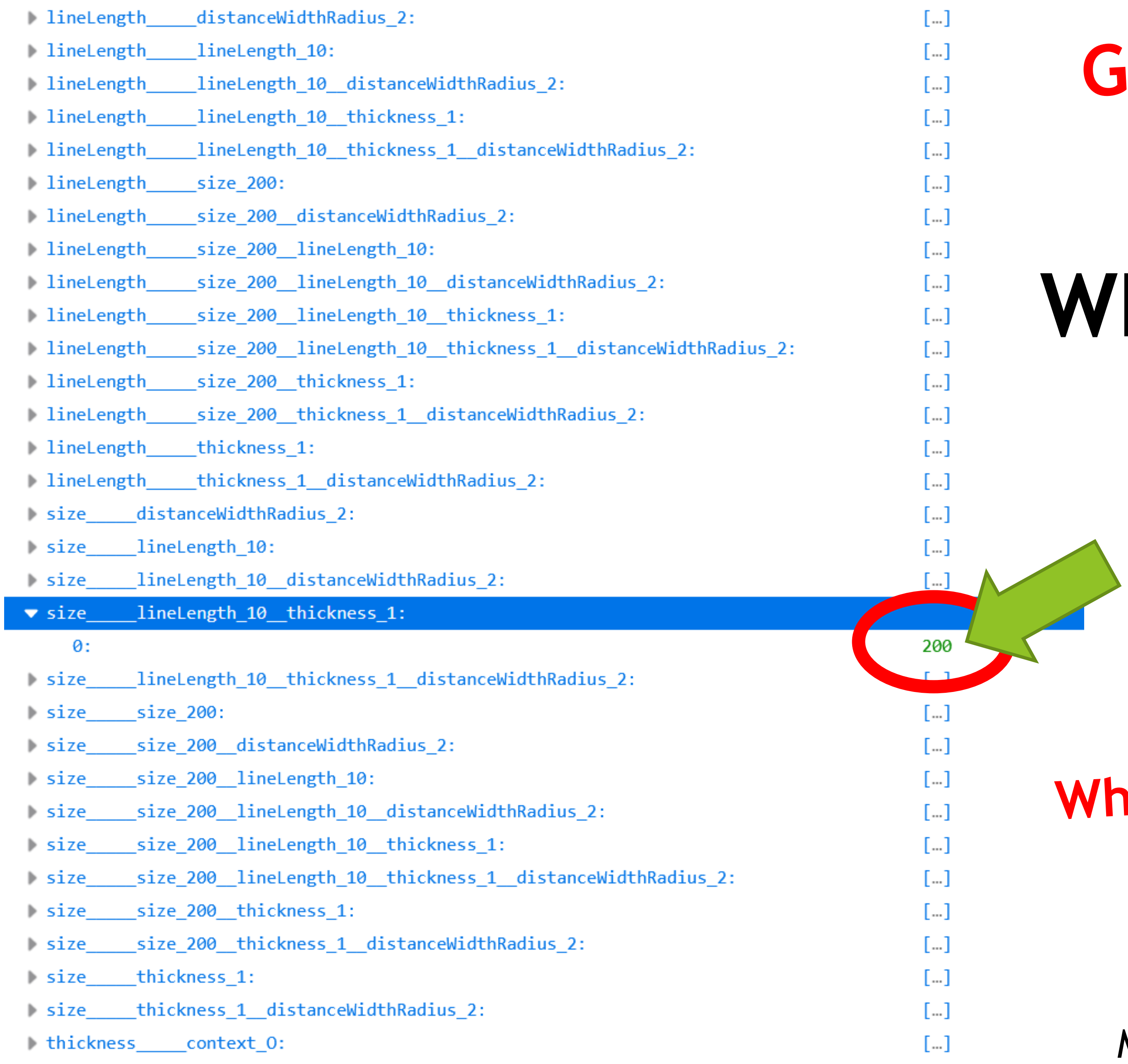
How to catch all feature variability?

When domain is focused on our aesthetic perception

In there suitable feature diagram?

A need to generate all possible derivations.

Can they include only mathematical model?



```
▶ lineLength_____distanceWidthRadius_2:                                                   […]
▶ lineLength_____lineLength_10:                                                           […]
▶ lineLength_____lineLength_10__distanceWidthRadius_2:                                    […]
▶ lineLength_____lineLength_10__thickness_1:                                              […]
▶ lineLength_____lineLength_10__thickness_1__distanceWidthRadius_2:                       […]
▶ lineLength_____size_200:                                                                […]
▶ lineLength_____size_200__distanceWidthRadius_2:                                         […]
▶ lineLength_____size_200__lineLength_10:                                                 […]
▶ lineLength_____size_200__lineLength_10__distanceWidthRadius_2:                          […]
▶ lineLength_____size_200__lineLength_10__thickness_1:                                    […]
▶ lineLength_____size_200__lineLength_10__thickness_1__distanceWidthRadius_2:             […]
▶ lineLength_____size_200__thickness_1:                                                   […]
▶ lineLength_____size_200__thickness_1__distanceWidthRadius_2:                            […]
▶ lineLength_____thickness_1:                                                             […]
▶ lineLength_____thickness_1__distanceWidthRadius_2:                                      […]
▶ size_____distanceWidthRadius_2:                                                         […]
▶ size_____lineLength_10:                                                                 […]
▶ size_____lineLength_10__distanceWidthRadius_2:                                          […]
▼ size_____lineLength_10__thickness_1:
     0:                                                                                   200
▶ size_____lineLength_10__thickness_1__distanceWidthRadius_2:                             […]
▶ size_____size_200:                                                                      […]
▶ size_____size_200__distanceWidthRadius_2:                                               […]
▶ size_____size_200__lineLength_10:                                                       […]
▶ size_____size_200__lineLength_10__distanceWidthRadius_2:                                […]
▶ size_____size_200__lineLength_10__thickness_1:                                          […]
▶ size_____size_200__lineLength_10__thickness_1__distanceWidthRadius_2:                   […]
▶ size_____size_200__thickness_1:                                                         […]
▶ size_____size_200__thickness_1__distanceWidthRadius_2:                                  […]
▶ size_____thickness_1:                                                                   […]
▶ size_____thickness_1__distanceWidthRadius_2:                                            […]
▶ thickness_____context_0:                                                                […]
```

Generated permutations

Not all are suitable,
but for completeness....

WHY? Changeability inside recursion

Variable **size** has value **200**,
As the same as values of variables:

lineLength = 10
thickness = 1

What we should be focusing on?

Dependency of recursion depth

More so than other variables inside recursion

What next with fractals?

- Analyze already harvested content to observe if catched variability can be used to improve (in automatic way):
  - Accuracy of third party systems (evaluating aesthetics)
  - Variability points – decomposing them, adding new ones, checking their suitability

Related to evaluation/statistics? -mainly to variability points in general

-contingency/pivot tables

-association tables

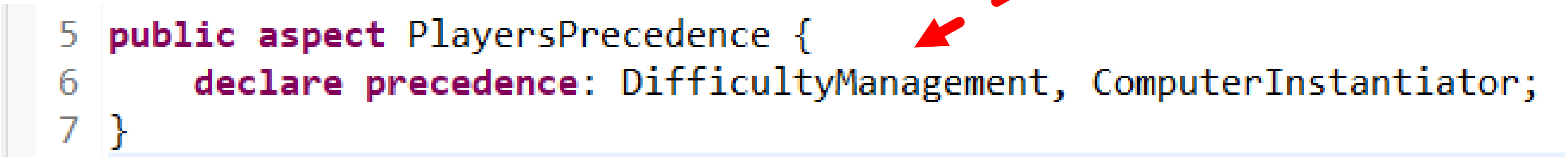
-agreement studies *Topic in statistics course*

- Build own model for fractals only – RESEARCH THE EVOLUTION OF SPL
- Build GAN to generate similar fractals – analyze impact of product in SPL evolution
  - try to design variability points based on the best ones

Difficulty configuration

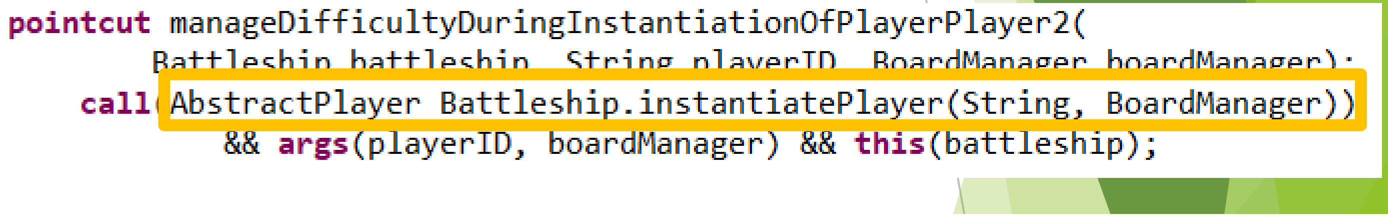
Prepare configuration (with difficulty settings) before creating player's specific instance

1. PREPARATION



```
public aspect PlayersPrecedence {
    declare precedence: DifficultyManagement, ComputerInstantiator;
}
```

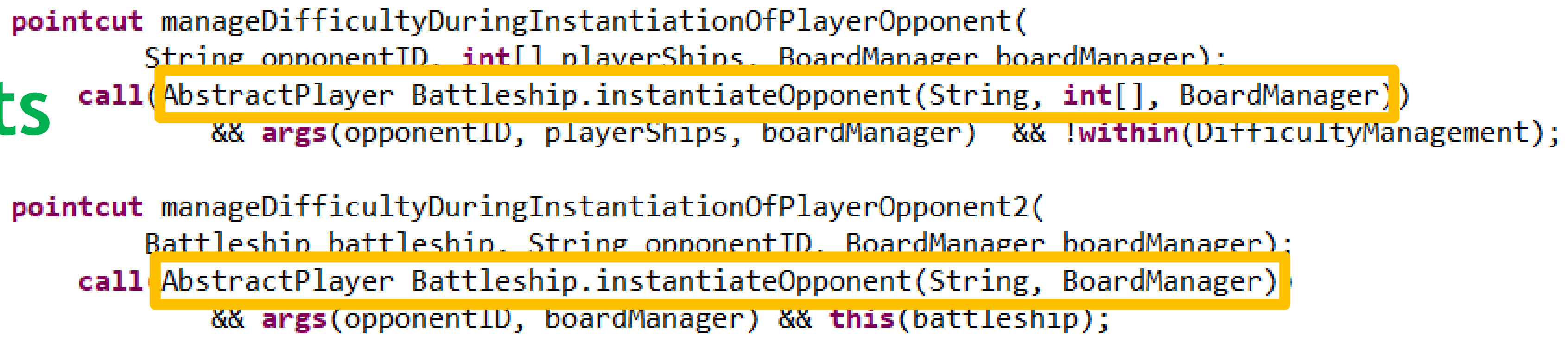
2. POINTCUTS



```
pointcut manageDifficultyDuringInstantiationOfPlayerPlayer2(
        Battleship battleship, String playerID, BoardManager boardManager):
    call(AbstractPlayer Battleship.instantiatePlayer(String, BoardManager))
        && args(playerID, boardManager) && this(battleship);
```

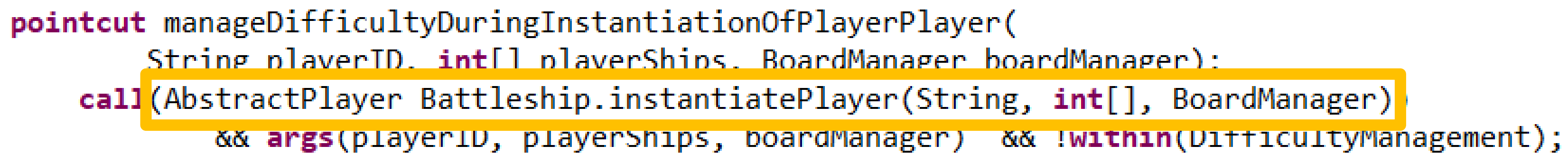
The same pointcuts

(with other names)



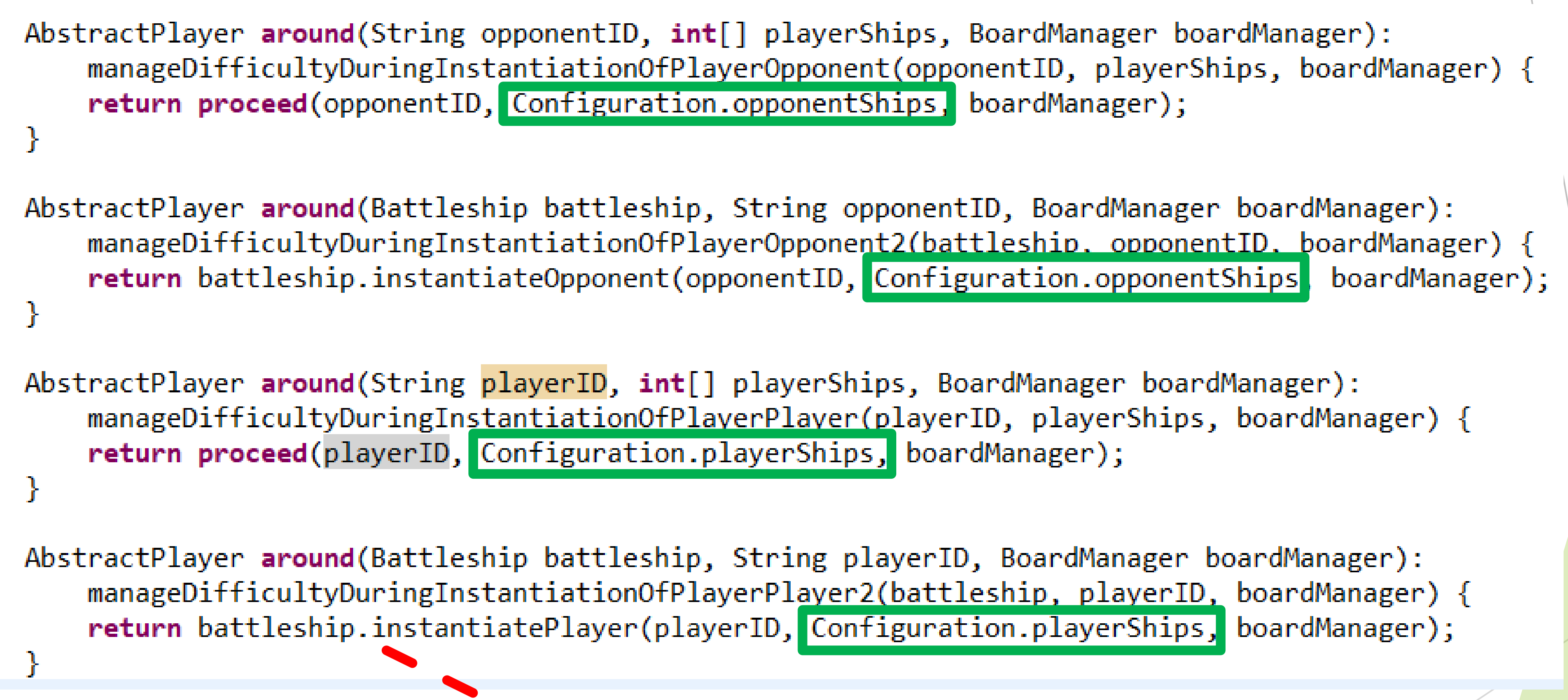
```
pointcut manageDifficultyDuringInstantiationOfPlayerOpponent(
        String opponentID, int[] playerShips, BoardManager boardManager):
    call(AbstractPlayer Battleship.instantiateOpponent(String, int[], BoardManager))
        && args(opponentID, playerShips, boardManager)  && !within(DifficultyManagement);

pointcut manageDifficultyDuringInstantiationOfPlayerOpponent2(
        Battleship battleship, String opponentID, BoardManager boardManager):
    call(AbstractPlayer Battleship.instantiateOpponent(String, BoardManager))
        && args(opponentID, boardManager) && this(battleship);
```

"Hook" functions



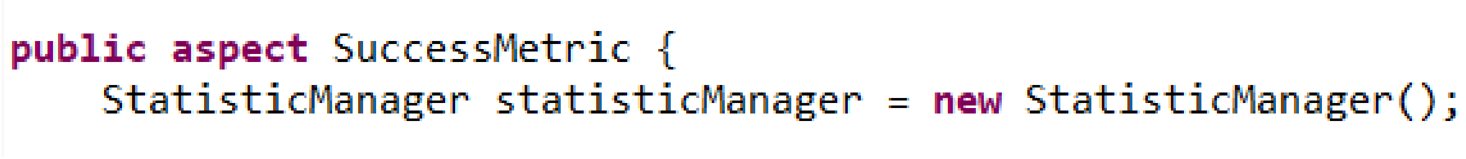
```
pointcut manageDifficultyDuringInstantiationOfPlayerPlayer(
        String playerID, int[] playerShips, BoardManager boardManager):
    call(AbstractPlayer Battleship.instantiatePlayer(String, int[], BoardManager))
        && args(playerID, playerShips, boardManager)  && !within(DifficultyManagement);
```

3. APPLYING CONFIGURATION VALUES

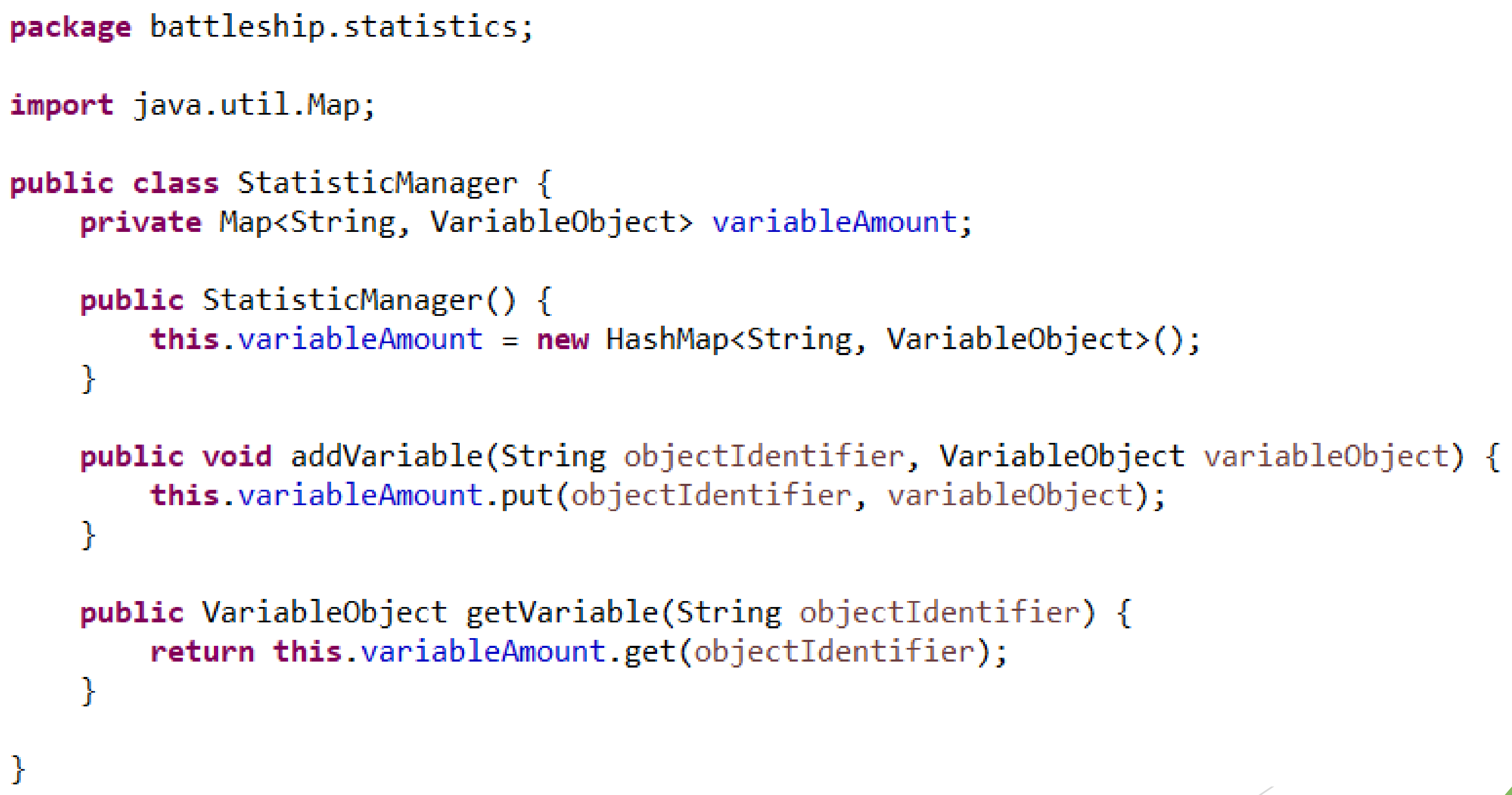


```
AbstractPlayer around(String opponentID, int[] playerShips, BoardManager boardManager):
    manageDifficultyDuringInstantiationOfPlayerOpponent(opponentID, playerShips, boardManager) {
    return proceed(opponentID, Configuration.opponentShips, boardManager);
}

AbstractPlayer around(Battleship battleship, String opponentID, BoardManager boardManager):
    manageDifficultyDuringInstantiationOfPlayerOpponent2(battleship, opponentID, boardManager) {
    return battleship.instantiateOpponent(opponentID, Configuration.opponentShips, boardManager);
}

AbstractPlayer around(String playerID, int[] playerShips, BoardManager boardManager):
    manageDifficultyDuringInstantiationOfPlayerPlayer(playerID, playerShips, boardManager) {
    return proceed(playerID, Configuration.playerShips, boardManager);
}

AbstractPlayer around(Battleship battleship, String playerID, BoardManager boardManager):
    manageDifficultyDuringInstantiationOfPlayerPlayer2(battleship, playerID, boardManager) {
    return battleship.instantiatePlayer(playerID, Configuration.playerShips, boardManager);
}
```

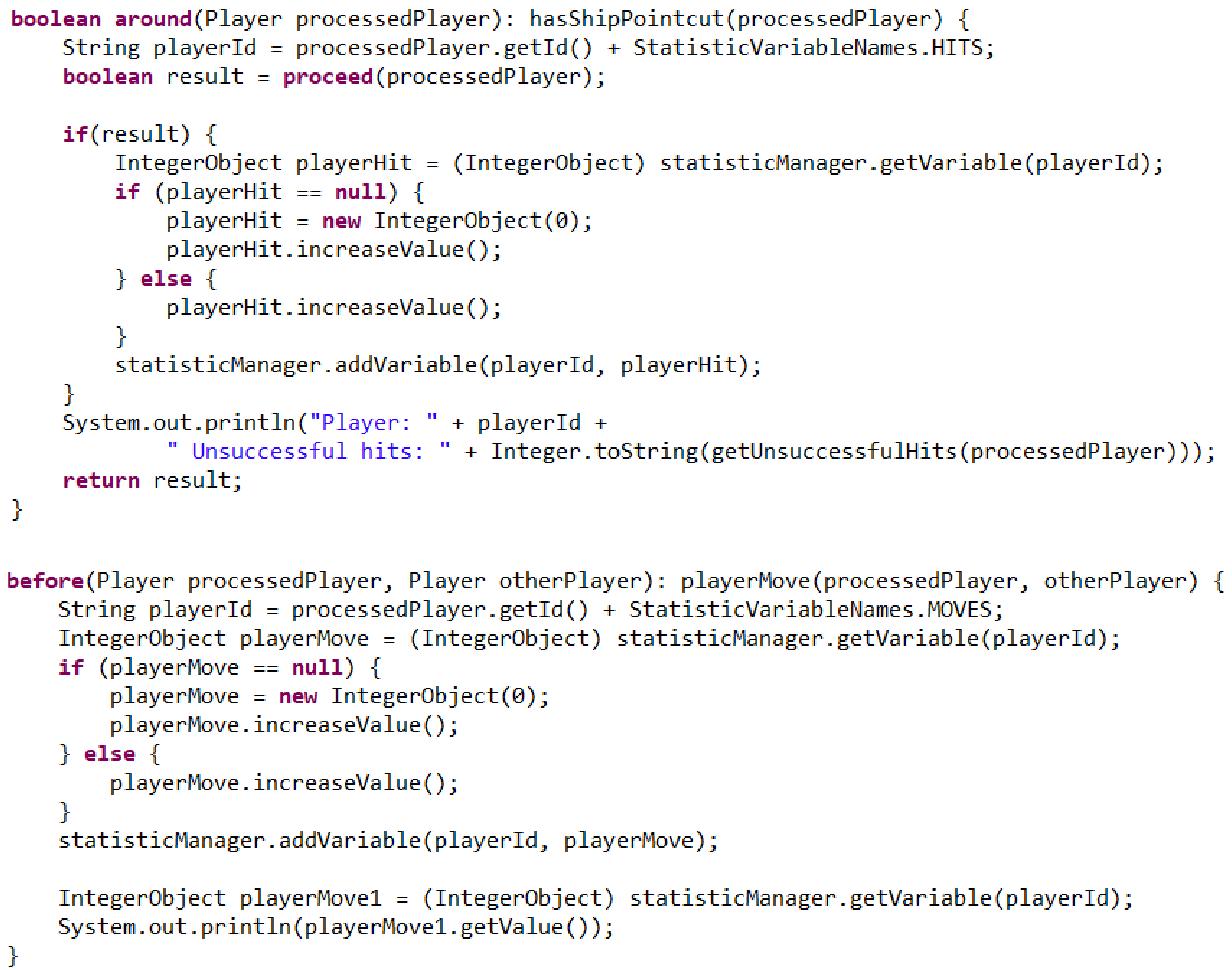
Calling the method with the same name but other arguments,
to apply other aspect managing player’s instance (showed previously)



```
public aspect SuccessMetric {
    StatisticManager statisticManager = new StatisticManager();
```



```
package battleship.statistics;

import java.util.Map;

public class StatisticManager {
    private Map<String, VariableObject> variableAmount;

    public StatisticManager() {
        this.variableAmount = new HashMap<String, VariableObject>();
    }

    public void addVariable(String objectIdentifier, VariableObject variableObject) {
        this.variableAmount.put(objectIdentifier, variableObject);
    }

    public VariableObject getVariable(String objectIdentifier) {
        return this.variableAmount.get(objectIdentifier);
    }

}
```

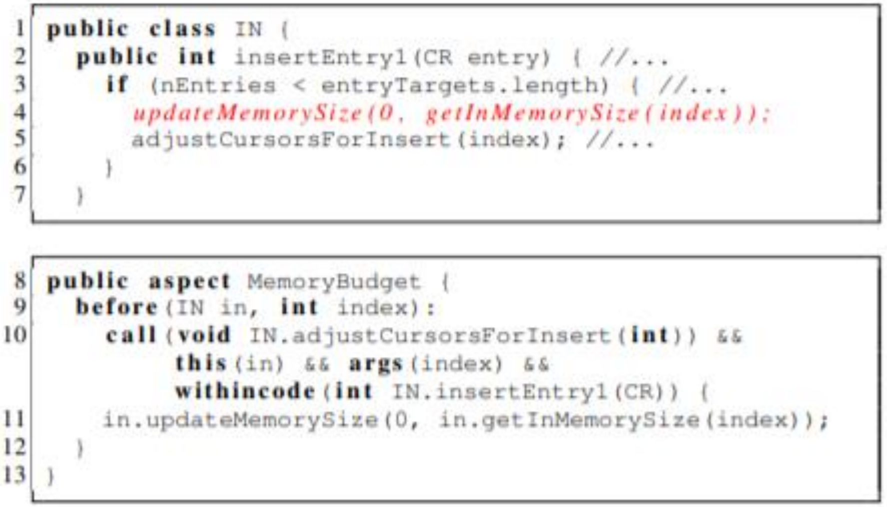


```
boolean around(Player processedPlayer): hasShipPointcut(processedPlayer) {
    String playerId = processedPlayer.getId() + StatisticVariableNames.HITS;
    boolean result = proceed(processedPlayer);

    if(result) {
        IntegerObject playerHit = (IntegerObject) statisticManager.getVariable(playerId);
        if (playerHit == null) {
            playerHit = new IntegerObject(0);
            playerHit.increaseValue();
        } else {
            playerHit.increaseValue();
        }
        statisticManager.addVariable(playerId, playerHit);
    }
    System.out.println("Player: " + playerId +
            " Unsuccessful hits: " + Integer.toString(getUnsuccessfulHits(processedPlayer)));
    return result;
}


before(Player processedPlayer, Player otherPlayer): playerMove(processedPlayer, otherPlayer) {
    String playerId = processedPlayer.getId() + StatisticVariableNames.MOVES;
    IntegerObject playerMove = (IntegerObject) statisticManager.getVariable(playerId);
    if (playerMove == null) {
        playerMove = new IntegerObject(0);
        playerMove.increaseValue();
    } else {
        playerMove.increaseValue();
    }
    statisticManager.addVariable(playerId, playerMove);

    IntegerObject playerMove1 = (IntegerObject) statisticManager.getVariable(playerId);
    System.out.println(playerMove1.getValue());
}
```

Software design according feature diagram

- Given functionality can spread trough whole system – in not modular systems
  - This functionality can be voluntary – marked in feature diagram this way
- For using aspects codes should be created according some principles
- How to derive product with / without given feature if feature has many classes and its implementation can include aspects too

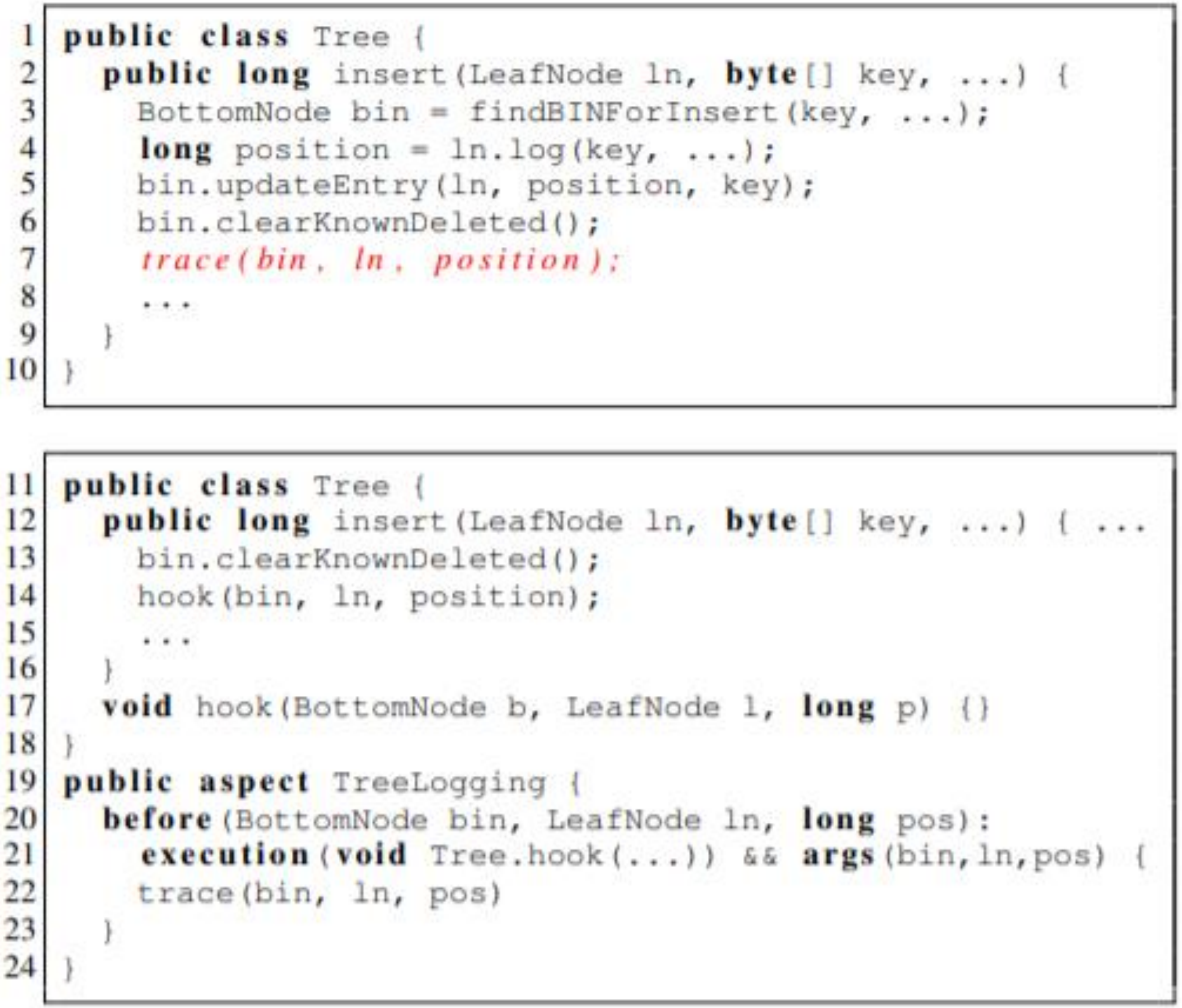
NEED TO KNOW CERTAIN DOMAIN



```
public class IN {
  public int insertEntry1(CR entry) { //...
    if (nEntries < entryTargets.length) { //...
      updateMemorySize(0, getInMemorySize(index));
      adjustCursorsForInsert(index); //...
    }
  }
```

```
public aspect MemoryBudget {
  before(IN in, int index):
    call(void IN.adjustCursorsForInsert(int)) &&
         this(in) && args(index) &&
         withincode(int IN.insertEntry1(CR)) {
    in.updateMemorySize(0, in.getInMemorySize(index));
  }
}
```

Figure 3. *Extract Before Call* **Refactoring.**



```
public class Tree {
  public long insert(LeafNode ln, byte[] key, ...) {
    BottomNode bin = findBINForInsert(key, ...);
    long position = ln.log(key, ...);
    bin.updateEntry(ln, position, key);
    bin.clearKnownDeleted();
    trace(bin, ln, position);
    ...
  }
}
```

```
public class Tree {
  public long insert(LeafNode ln, byte[] key, ...) { ...
    bin.clearKnownDeleted();
    hook(bin, ln, position);
    ...
  }
  void hook(BottomNode b, LeafNode l, long p) {}
}
public aspect TreeLogging {
  before(BottomNode bin, LeafNode ln, long pos):
    execution(void Tree.hook(...)) && args(bin,ln,pos) {
    trace(bin, ln, pos)
  }
}
```

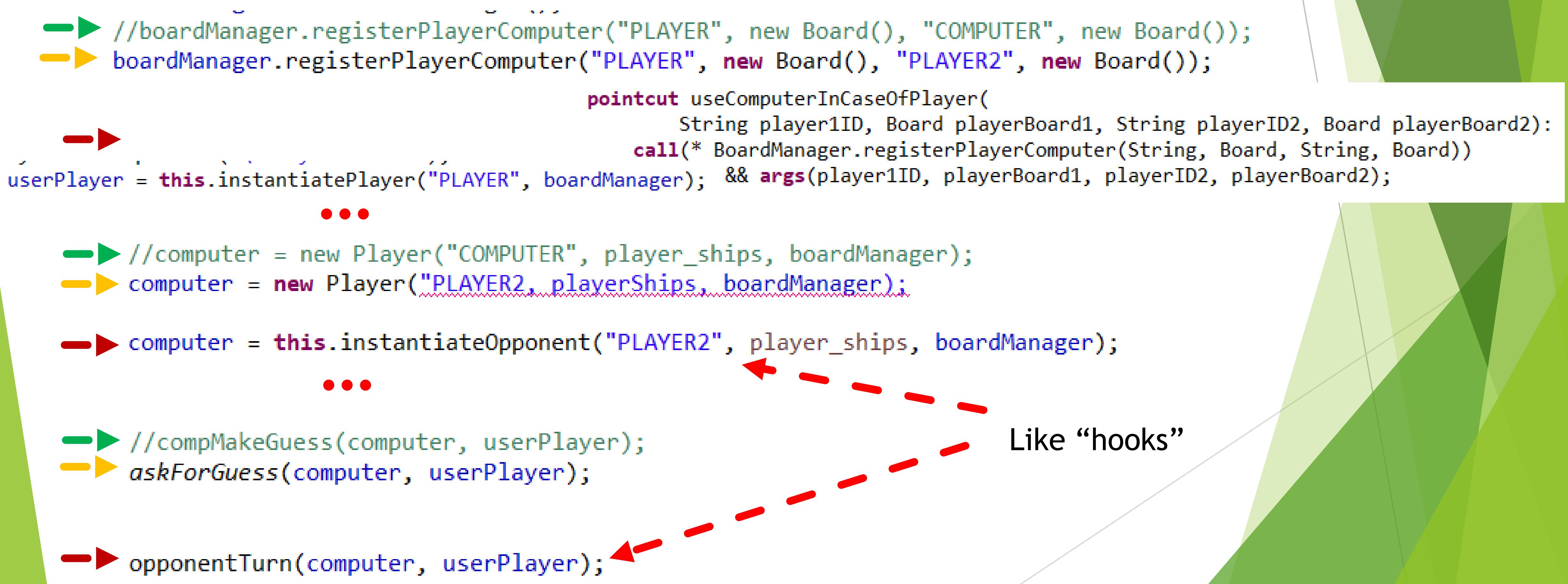
Figure 5. Local Variables Access Problem.

Adding support for computer or user opponent

🟨 *Player as opponent*

🟩 *Computer as opponent*

🟥 Changes to use both – aspect use



🟩
```
//boardManager.registerPlayerComputer("PLAYER", new Board(), "COMPUTER", new Board());
```
🟨
```
boardManager.registerPlayerComputer("PLAYER", new Board(), "PLAYER2", new Board());
```
🟥
```
userPlayer = this.instantiatePlayer("PLAYER", boardManager);
```
```
pointcut useComputerInCaseOfPlayer(
        String player1ID, Board playerBoard1, String playerID2, Board playerBoard2):
    call(* BoardManager.registerPlayerComputer(String, Board, String, Board))
    && args(player1ID, playerBoard1, playerID2, playerBoard2);
```

…

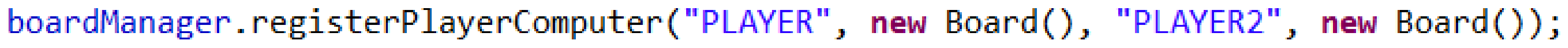
🟩
```
//computer = new Player("COMPUTER", player_ships, boardManager);
```
🟨
```
computer = new Player("PLAYER2, playerShips, boardManager);
```
🟥
```
computer = this.instantiateOpponent("PLAYER2", player_ships, boardManager);
```

…

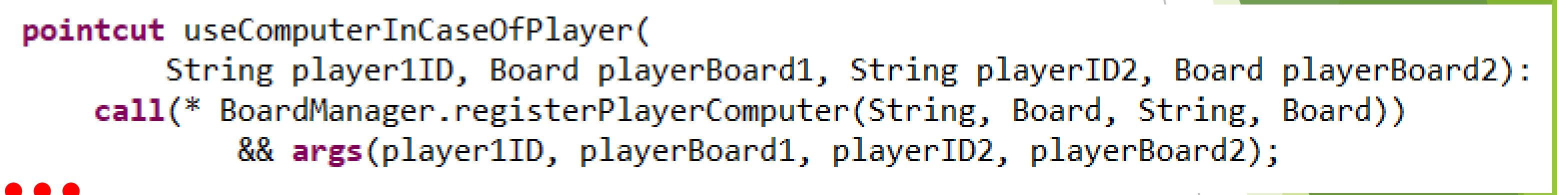
🟩
```
//compMakeGuess(computer, userPlayer);
```
🟨
```
askForGuess(computer, userPlayer);
```
🟥
```
opponentTurn(computer, userPlayer);
```

Like “hooks”

Mapping of pointcuts

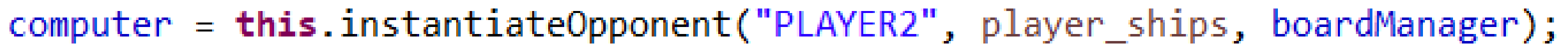


```
boardManager.registerPlayerComputer("PLAYER", new Board(), "PLAYER2", new Board());
```

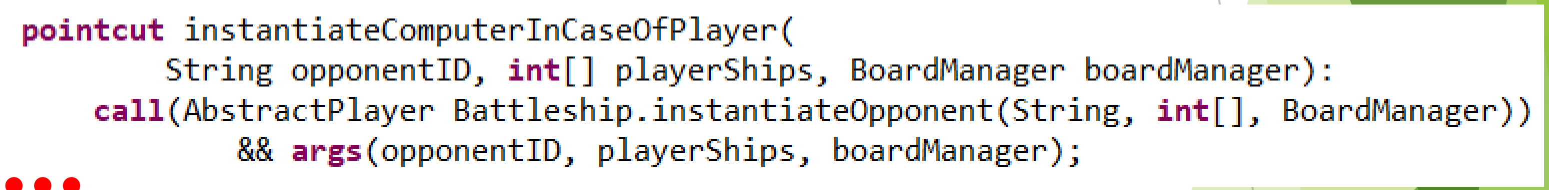


```
pointcut useComputerInCaseOfPlayer(
        String player1ID, Board playerBoard1, String playerID2, Board playerBoard2):
    call(* BoardManager.registerPlayerComputer(String, Board, String, Board))
        && args(player1ID, playerBoard1, playerID2, playerBoard2);
```

●●●

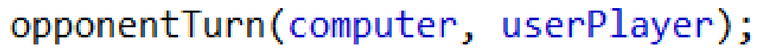


```
computer = this.instantiateOpponent("PLAYER2", player_ships, boardManager);
```

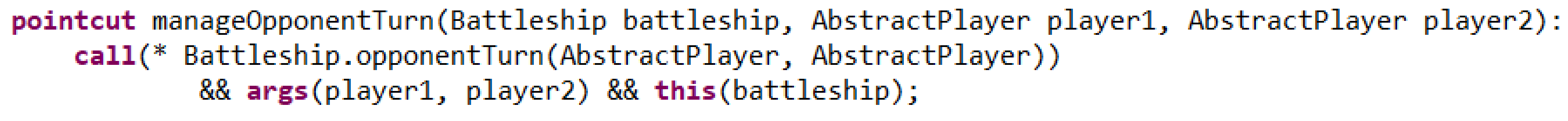


```
pointcut instantiateComputerInCaseOfPlayer(
        String opponentID, int[] playerShips, BoardManager boardManager):
    call(AbstractPlayer Battleship.instantiateOpponent(String, int[], BoardManager))
        && args(opponentID, playerShips, boardManager);
```

●●●



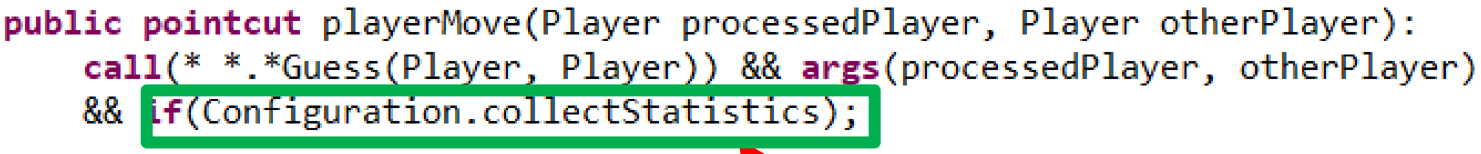
```
opponentTurn(computer, userPlayer);
```



```
pointcut manageOpponentTurn(Battleship battleship, AbstractPlayer player1, AbstractPlayer player2):
    call(* Battleship.opponentTurn(AbstractPlayer, AbstractPlayer))
        && args(player1, player2) && this(battleship);
```

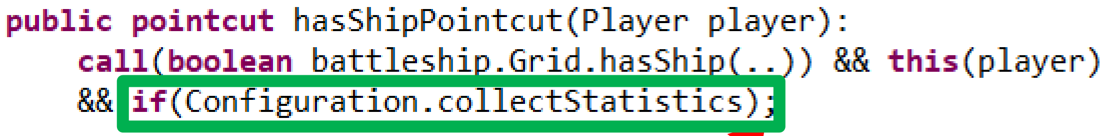
Statistics configuration

MOVES



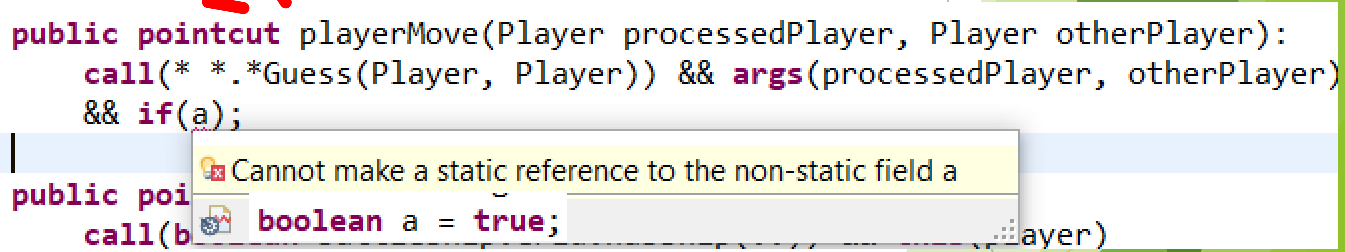
```
public pointcut playerMove(Player processedPlayer, Player otherPlayer):
    call(* *.*Guess(Player, Player)) && args(processedPlayer, otherPlayer)
    && if(Configuration.collectStatistics);
```

HITS



```
public pointcut hasShipPointcut(Player player):
    call(boolean battleship.Grid.hasShip(..)) && this(player)
    && if(Configuration.collectStatistics);
```

Statistics observation are gathered
if value of variable from config file is True

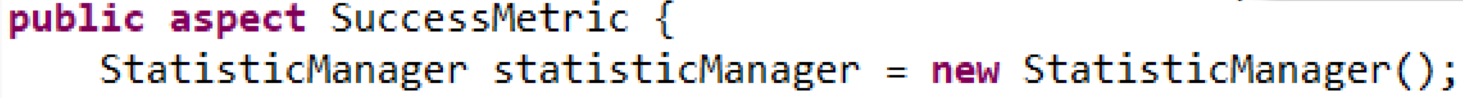


```
public pointcut playerMove(Player processedPlayer, Player otherPlayer):
    call(* *.*Guess(Player, Player)) && args(processedPlayer, otherPlayer)
    && if(a);

public poi
    call(b                                        ayer)
```

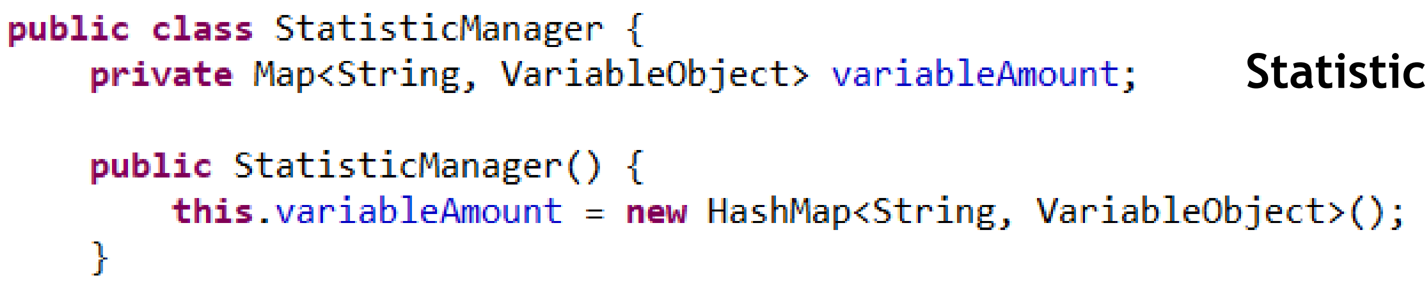
Cannot make a static reference to the non-static field a

```
boolean a = true;
```

MISS = MOVES - HITS

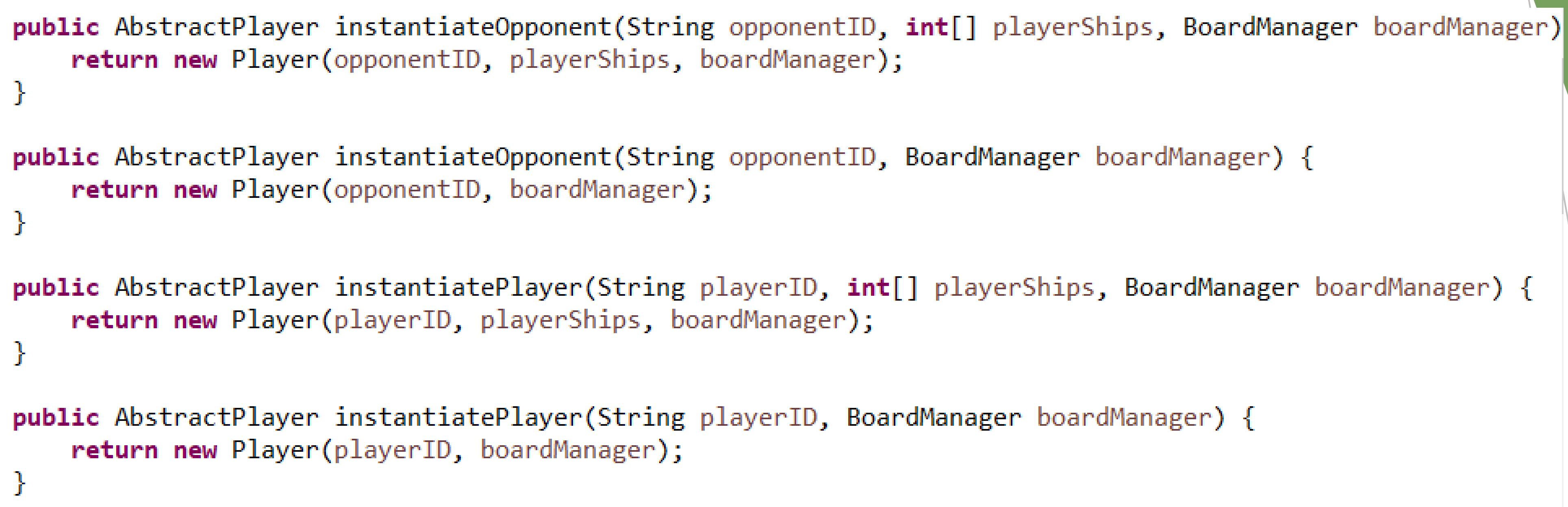


```
public aspect SuccessMetric {
    StatisticManager statisticManager = new StatisticManager();
```



```
public class StatisticManager {
    private Map<String, VariableObject> variableAmount;

    public StatisticManager() {
        this.variableAmount = new HashMap<String, VariableObject>();
    }
```

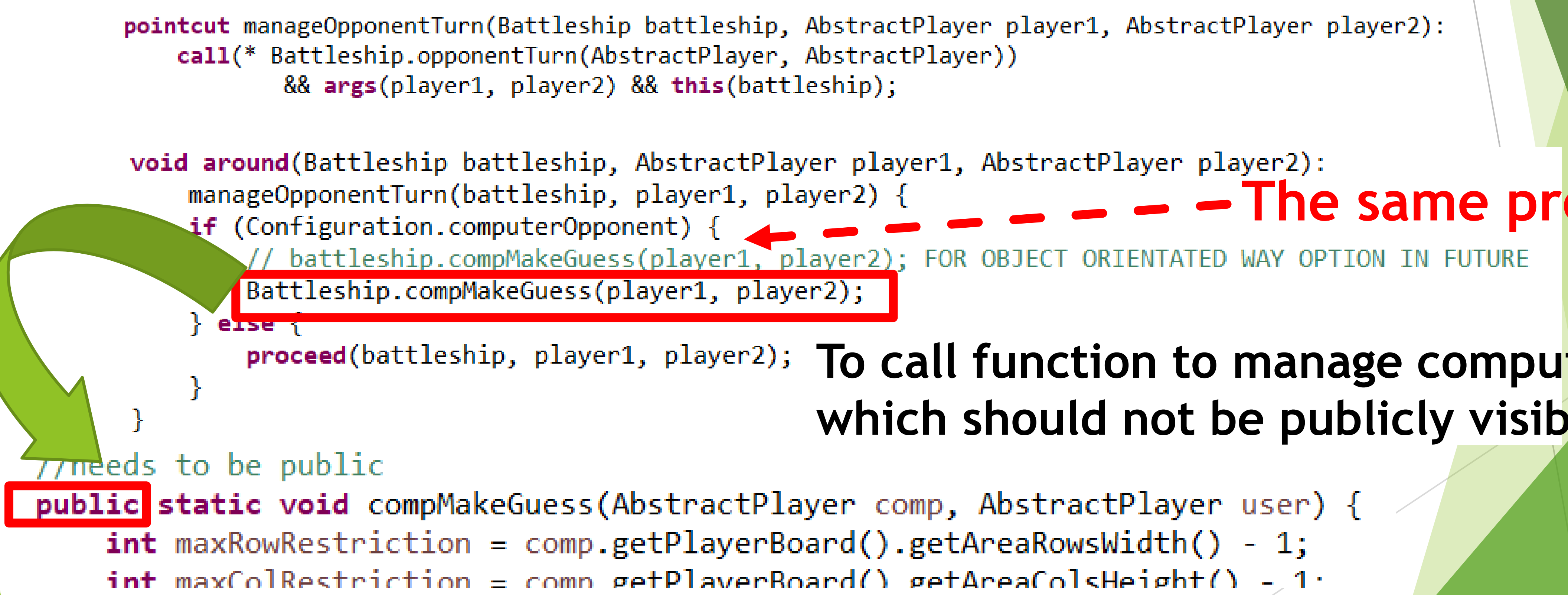
Statistics objects are stored in hash-map



```
public AbstractPlayer instantiateOpponent(String opponentID, int[] playerShips, BoardManager boardManager)
    return new Player(opponentID, playerShips, boardManager);
}

public AbstractPlayer instantiateOpponent(String opponentID, BoardManager boardManager) {
    return new Player(opponentID, boardManager);
}

public AbstractPlayer instantiatePlayer(String playerID, int[] playerShips, BoardManager boardManager) {
    return new Player(playerID, playerShips, boardManager);
}

public AbstractPlayer instantiatePlayer(String playerID, BoardManager boardManager) {
    return new Player(playerID, boardManager);
}
```

Variable encapsulation problem

In player instance chooser aspect:

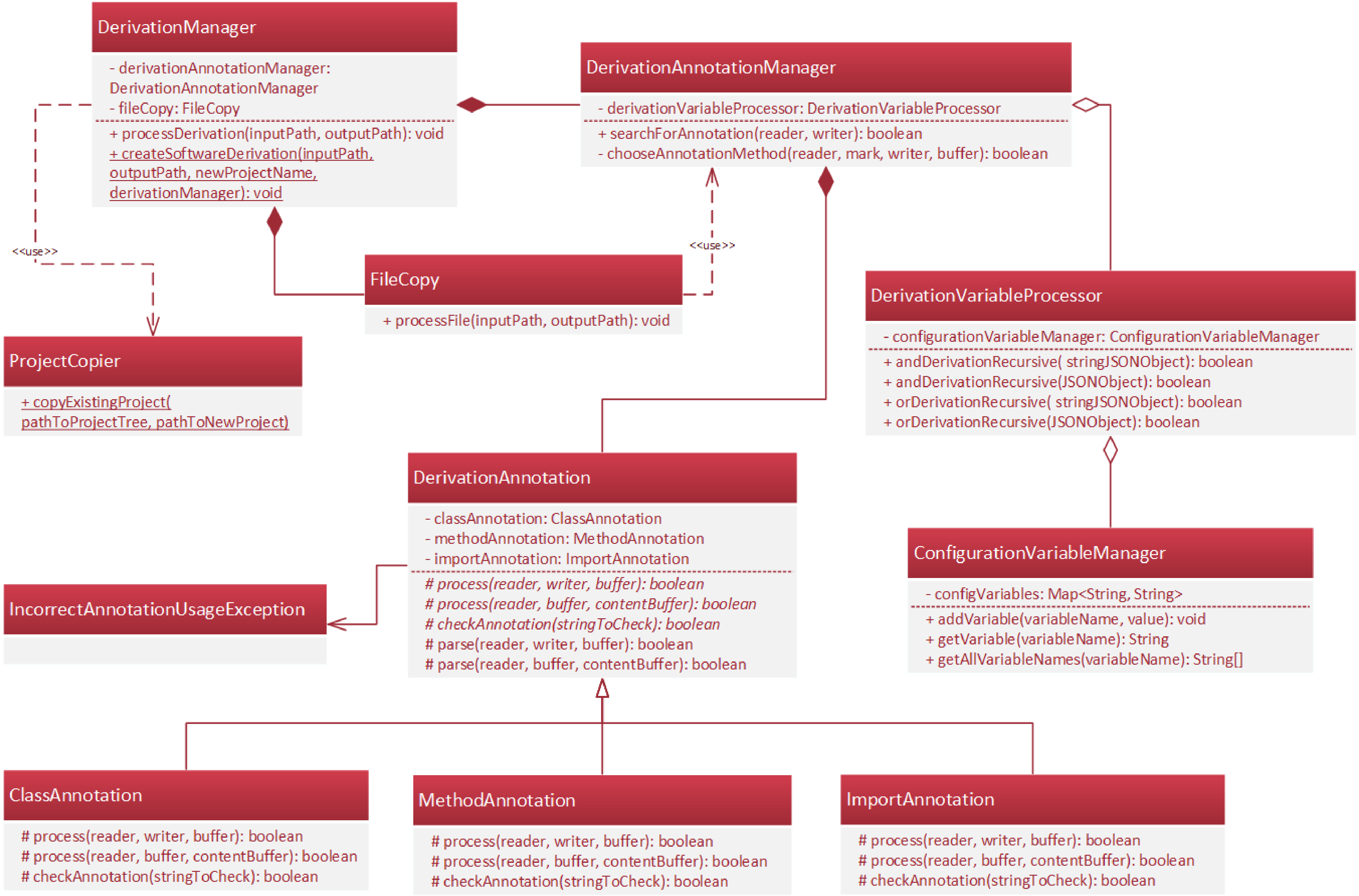


```
pointcut manageOpponentTurn(Battleship battleship, AbstractPlayer player1, AbstractPlayer player2):
    call(* Battleship.opponentTurn(AbstractPlayer, AbstractPlayer))
        && args(player1, player2) && this(battleship);

void around(Battleship battleship, AbstractPlayer player1, AbstractPlayer player2):
    manageOpponentTurn(battleship, player1, player2) {
    if (Configuration.computerOpponent) {
        // battleship.compMakeGuess(player1, player2); FOR OBJECT ORIENTATED WAY OPTION IN FUTURE
        Battleship.compMakeGuess(player1, player2);
    } else {
        proceed(battleship, player1, player2);
    }
}
```

```
//needs to be public
public static void compMakeGuess(AbstractPlayer comp, AbstractPlayer user) {
    int maxRowRestriction = comp.getPlayerBoard().getAreaRowsWidth() - 1;
    int maxColRestriction = comp.getPlayerBoard().getAreaColsHeight() - 1;
```

The same problem

To call function to manage computer guess, which should not be publicly visible



DerivationManager

- derivationAnnotationManager: DerivationAnnotationManager
- fileCopy: FileCopy

+ processDerivation(inputPath, outputPath): void
+ createSoftwareDerivation(inputPath, outputPath, newProjectName, derivationManager): void

DerivationAnnotationManager

- derivationVariableProcessor: DerivationVariableProcessor

+ searchForAnnotation(reader, writer): boolean
- chooseAnnotationMethod(reader, mark, writer, buffer): boolean

FileCopy

+ processFile(inputPath, outputPath): void

ProjectCopier

+ copyExistingProject(pathToProjectTree, pathToNewProject)

DerivationVariableProcessor

- configurationVariableManager: ConfigurationVariableManager

+ andDerivationRecursive(stringJSONObject): boolean
+ andDerivationRecursive(JSONObject): boolean
+ orDerivationRecursive(stringJSONObject): boolean
+ orDerivationRecursive(JSONObject): boolean

DerivationAnnotation

- classAnnotation: ClassAnnotation
- methodAnnotation: MethodAnnotation
- importAnnotation: ImportAnnotation

# process(reader, writer, buffer): boolean
# process(reader, buffer, contentBuffer): boolean
# checkAnnotation(stringToCheck): boolean
parse(reader, writer, buffer): boolean
parse(reader, buffer, contentBuffer): boolean

IncorrectAnnotationUsageException

ConfigurationVariableManager

- configVariables: Map<String, String>

+ addVariable(variableName, value): void
+ getVariable(variableName): String
+ getAllVariableNames(variableName): String[]

ClassAnnotation

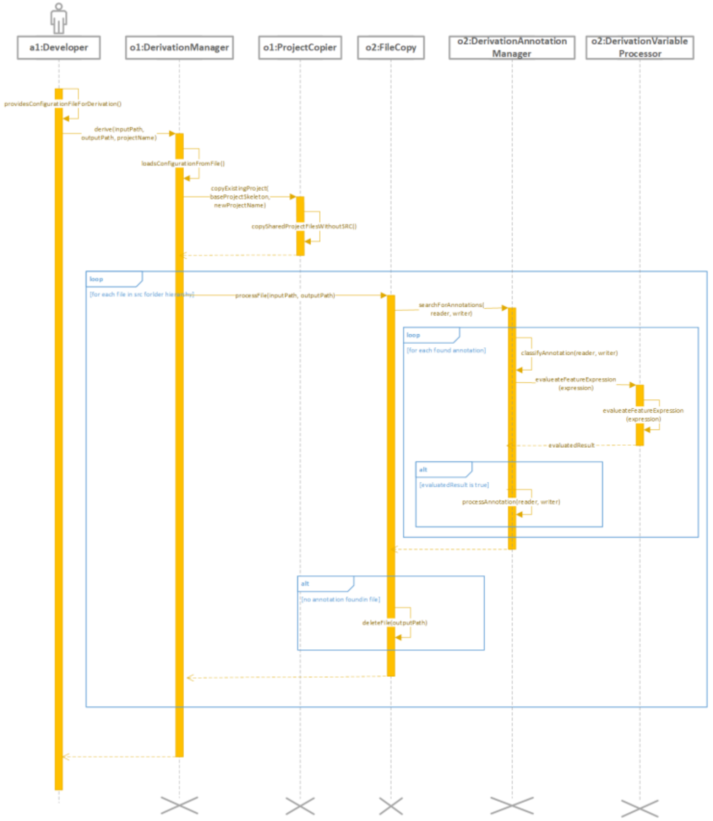
process(reader, writer, buffer): boolean
process(reader, buffer, contentBuffer): boolean
checkAnnotation(stringToCheck): boolean

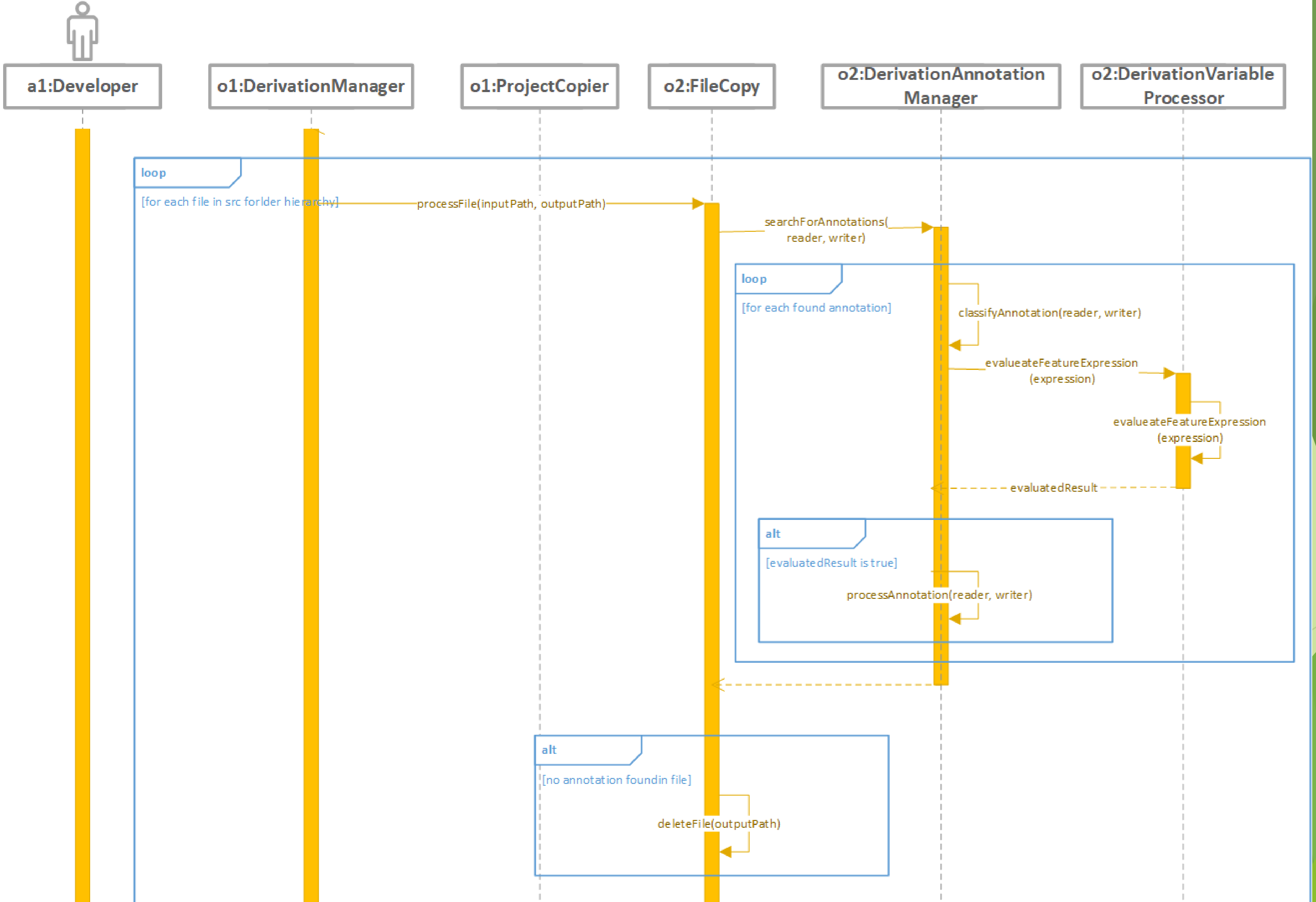
MethodAnnotation

process(reader, writer, buffer): boolean
process(reader, buffer, contentBuffer): boolean
checkAnnotation(stringToCheck): boolean

ImportAnnotation

process(reader, writer, buffer): boolean
process(reader, buffer, contentBuffer): boolean
checkAnnotation(stringToCheck): boolean


a1:Developer
o1:DerivationManager
o1:ProjectCopier
o2:FileCopy
o2:DerivationAnnotation Manager
o2:DerivationVariable Processor
providesConfigurationFileForDerivation()
derive(inputPath, outputPath, projectName)
loadsConfigurationFromFile()
copyExistingProject(baseProjectSkeleton, newProjectName)
copySharedProjectFilesWithoutSRC()
loop
[for each file in src forlder hierarchy]
processFile(inputPath, outputPath)
searchForAnnotations(reader, writer)
loop
[for each found annotation]
classifyAnnotation(reader, writer)
evalueateFeatureExpression (expression)
evalueateFeatureExpression (expression)
evaluatedResult
alt
[evaluatedResult is true]
processAnnotation(reader, writer)
alt
[no annotation foundin file]
deleteFile(outputPath)



a1:Developer

o1:DerivationManager

o1:ProjectCopier

o2:FileCopy

o2:DerivationAnnotation Manager

o2:DerivationVariable Processor

loop

[for each file in src forlder hierarchy]

processFile(inputPath, outputPath)

searchForAnnotations(reader, writer)

loop

[for each found annotation]

classifyAnnotation(reader, writer)

evalueateFeatureExpression (expression)

evalueateFeatureExpression (expression)

evaluatedResult

alt

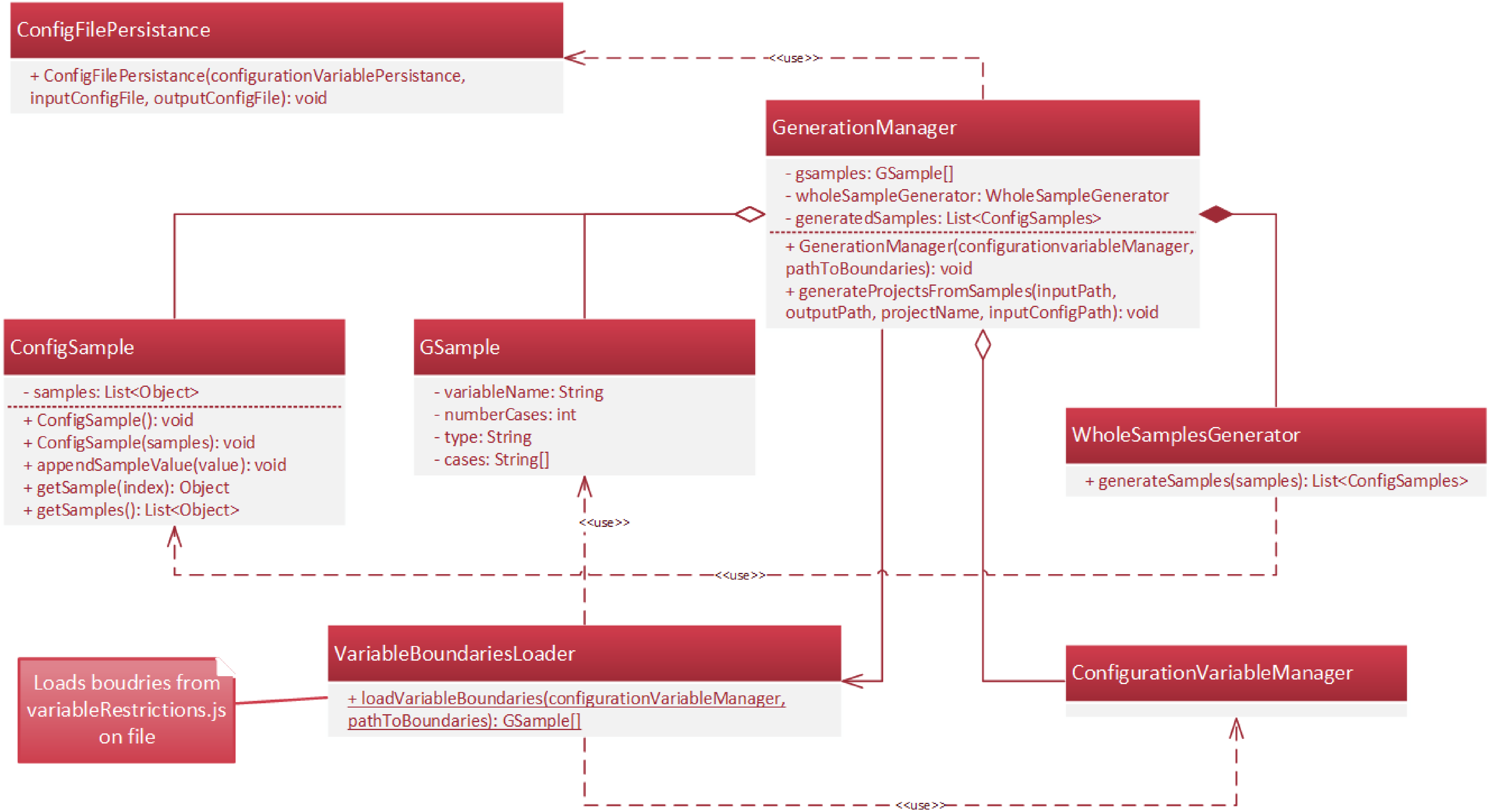
[evaluatedResult is true]

processAnnotation(reader, writer)

alt

[no annotation foundin file]

deleteFile(outputPath)



ConfigFilePersistance

+ ConfigFilePersistance(configurationVariablePersistance, inputConfigFile, outputConfigFile): void

<<use>>

GenerationManager

- gsamples: GSample[]
- wholeSampleGenerator: WholeSampleGenerator
- generatedSamples: List<ConfigSamples>

+ GenerationManager(configurationvariableManager, pathToBoundaries): void
+ generateProjectsFromSamples(inputPath, outputPath, projectName, inputConfigPath): void

ConfigSample

- samples: List<Object>

+ ConfigSample(): void
+ ConfigSample(samples): void
+ appendSampleValue(value): void
+ getSample(index): Object
+ getSamples(): List<Object>

GSample

- variableName: String
- numberCases: int
- type: String
- cases: String[]

WholeSamplesGenerator

+ generateSamples(samples): List<ConfigSamples>

<<use>>

<<use>>

Loads boudries from variableRestrictions.js on file

VariableBoundariesLoader

+ loadVariableBoundaries(configurationVariableManager, pathToBoundaries): GSample[]

ConfigurationVariableManager

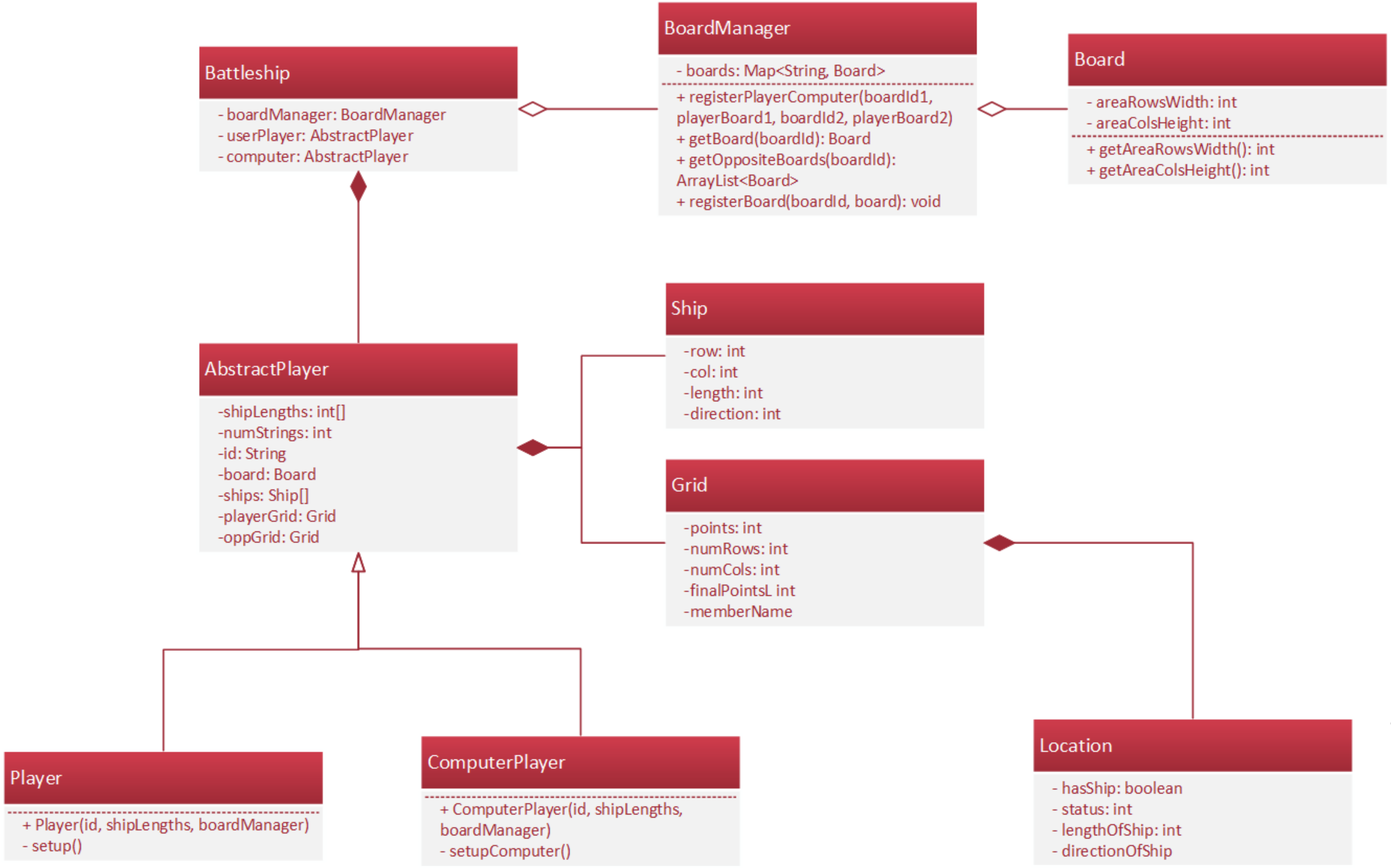
<<use>>

Object oriented redesign

- ***Hardcoded parts should be changed to support configurability***
  - Different lengths of board
  - Support for adding player
- ***Concerns should be separated***
  - Setup of player should be part of player class
  - Setup of computer should be part of computer class
- ***Static methods should be replaced by objects***

Performing refactoring of project

Schema after refactoring



Battleship

- boardManager: BoardManager
- userPlayer: AbstractPlayer
- computer: AbstractPlayer

BoardManager

- boards: Map<String, Board>

+ registerPlayerComputer(boardId1, playerBoard1, boardId2, playerBoard2)
+ getBoard(boardId): Board
+ getOppositeBoards(boardId): ArrayList<Board>
+ registerBoard(boardId, board): void

Board

- areaRowsWidth: int
- areaColsHeight: int

+ getAreaRowsWidth(): int
+ getAreaColsHeight(): int

AbstractPlayer

-shipLengths: int[]
-numStrings: int
-id: String
-board: Board
-ships: Ship[]
-playerGrid: Grid
-oppGrid: Grid

Ship

-row: int
-col: int
-length: int
-direction: int

Grid

-points: int
-numRows: int
-numCols: int
-finalPointsL int
-memberName

Player

+ Player(id, shipLengths, boardManager)
- setup()

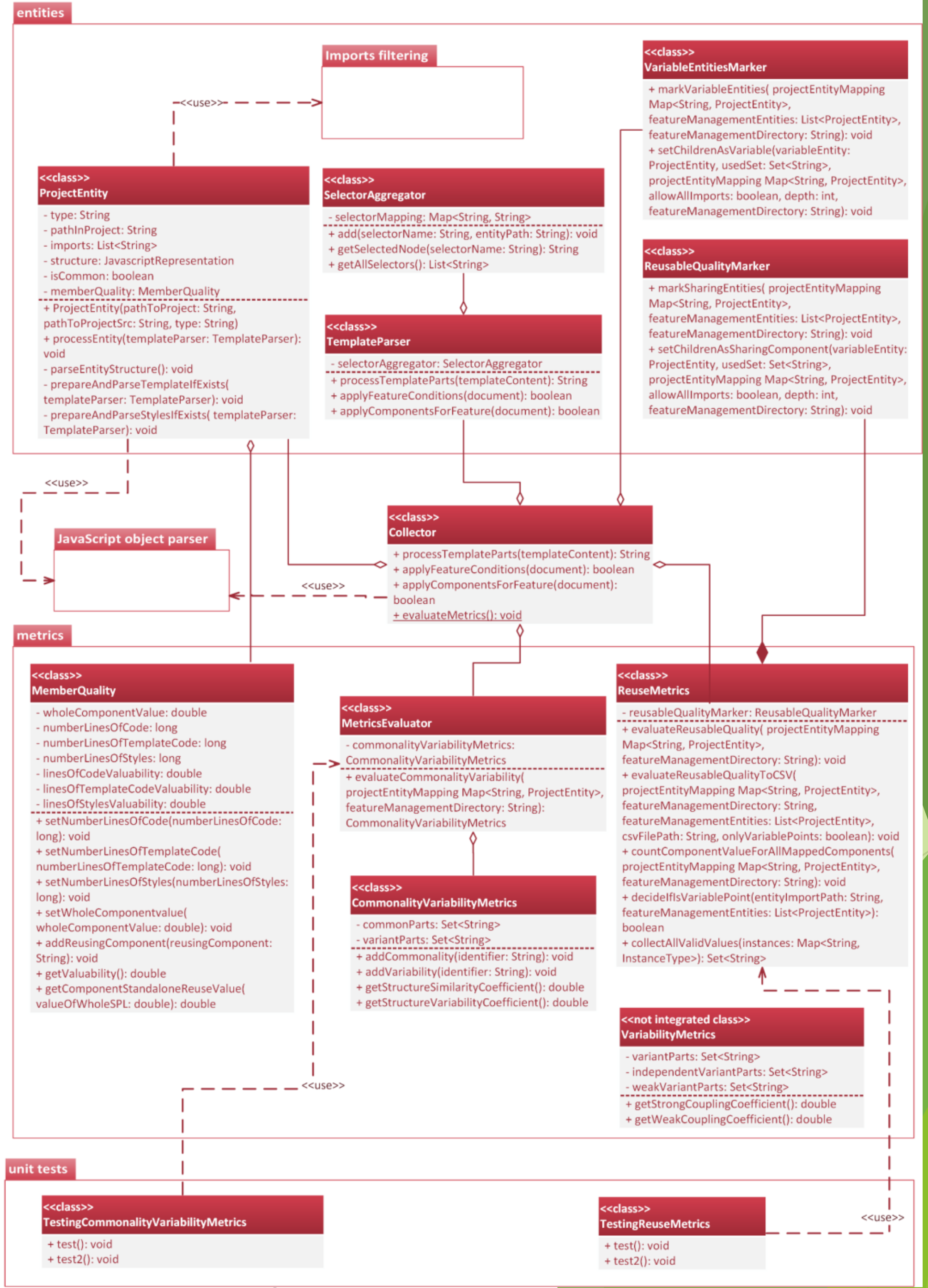
ComputerPlayer

+ ComputerPlayer(id, shipLengths, boardManager)
- setupComputer()

Location

- hasShip: boolean
- status: int
- lengthOfShip: int
- directionOfShip

Quality checker structure



<<class>>
ProjectEntity

- type: String
- pathInProject: String
- imports: List<String>
- structure: JavascriptRepresentation
- isCommon: boolean
- memberQuality: MemberQuality

+ ProjectEntity(pathToProject: String, pathToProjectSrc: String, type: String)
+ processEntity(templateParser: TemplateParser): void
- parseEntityStructure(): void
- prepareAndParseTemplateIfExists(templateParser: TemplateParser): void
- prepareAndParseStylesIfExists(templateParser: TemplateParser): void

<<class>>
SelectorAggregator

- selectorMapping: Map<String, String>

+ add(selectorName: String, entityPath: String): void
+ getSelectedNode(selectorName: String): String
+ getAllSelectors(): List<String>

<<class>>
TemplateParser

- selectorAggregator: SelectorAggregator

+ processTemplateParts(templateContent): String
+ applyFeatureConditions(document): boolean
+ applyComponentsForFeature(document): boolean

<<class>>
VariableEntitiesMarker

+ markVariableEntities(projectEntityMapping Map<String, ProjectEntity>, featureManagementEntities: List<ProjectEntity>, featureManagementDirectory: String): void
+ setChildrenAsVariable(variableEntity: ProjectEntity, usedSet: Set<String>, projectEntityMapping Map<String, ProjectEntity>, allowAllImports: boolean, depth: int, featureManagementDirectory: String): void

<<class>>
ReusableQualityMarker

+ markSharingEntities(projectEntityMapping Map<String, ProjectEntity>, featureManagementEntities: List<ProjectEntity>, featureManagementDirectory: String): void
+ setChildrenAsSharingComponent(variableEntity: ProjectEntity, usedSet: Set<String>, projectEntityMapping Map<String, ProjectEntity>, allowAllImports: boolean, depth: int, featureManagementDirectory: String): void

<<class>>
Collector

+ processTemplateParts(templateContent): String
+ applyFeatureConditions(document): boolean
+ applyComponentsForFeature(document): boolean
+ evaluateMetrics(): void

<<class>>
MemberQuality

- wholeComponentValue: double
- numberLinesOfCode: long
- numberLinesOfTemplateCode: long
- numberLinesOfStyles: long
- linesOfCodeValuability: double
- linesOfTemplateCodeValuability: double
- linesOfStylesValuability: double

+ setNumberLinesOfCode(numberLinesOfCode: long): void
+ setNumberLinesOfTemplateCode(numberLinesOfTemplateCode: long): void
+ setNumberLinesOfStyles(numberLinesOfStyles: long): void
+ setWholeComponentvalue(wholeComponentValue: double): void
+ addReusingComponent(reusingComponent: String): void
+ getValuability(): double
+ getComponentStandaloneReuseValue(valueOfWholeSPL: double): double

<<class>>
MetricsEvaluator

- commonalityVariabilityMetrics: CommonalityVariabilityMetrics

+ evaluateCommonalityVariability(projectEntityMapping Map<String, ProjectEntity>, featureManagementDirectory: String): CommonalityVariabilityMetrics

<<class>>
CommonalityVariabilityMetrics

- commonParts: Set<String>
- variantParts: Set<String>

+ addCommonality(identifier: String): void
+ addVariability(identifier: String): void
+ getStructureSimilarityCoefficient(): double
+ getStructureVariabilityCoefficient(): double

<<class>>
ReuseMetrics

- reusableQualityMarker: ReusableQualityMarker

+ evaluateReusableQuality(projectEntityMapping Map<String, ProjectEntity>, featureManagementDirectory: String): void
+ evaluateReusableQualityToCSV(projectEntityMapping Map<String, ProjectEntity>, featureManagementDirectory: String, featureManagementEntities: List<ProjectEntity>, csvFilePath: String, onlyVariablePoints: boolean): void
+ countComponentValueForAllMappedComponents(projectEntityMapping Map<String, ProjectEntity>, featureManagementDirectory: String): void
+ decideIfIsVariablePoint(entityImportPath: String, featureManagementEntities: List<ProjectEntity>): boolean
+ collectAllValidValues(instances: Map<String, InstanceType>): Set<String>

<<not integrated class>>
VariabilityMetrics

- variantParts: Set<String>
- independentVariantParts: Set<String>
- weakVariantParts: Set<String>

+ getStrongCouplingCoefficient(): double
+ getWeakCouplingCoefficient(): double

<<class>>
TestingCommonalityVariabilityMetrics

+ test(): void
+ test2(): void

<<class>>
TestingReuseMetrics

+ test(): void
+ test2(): void

Evaluating customized dataset

- Manual annotations – based on own aesthetics
- Used third party model

- -comparing different fractal representations/formats:
  - Vector graphics – whole structure is written as text .SVG
  - Raster graphics
  - Information from variation points

- **inserts knowledge from structure of program generator itself into data**

Already **378** fractals generated from one file

-based on permutations of variation points and recursion

Can be more, but...

we bring:

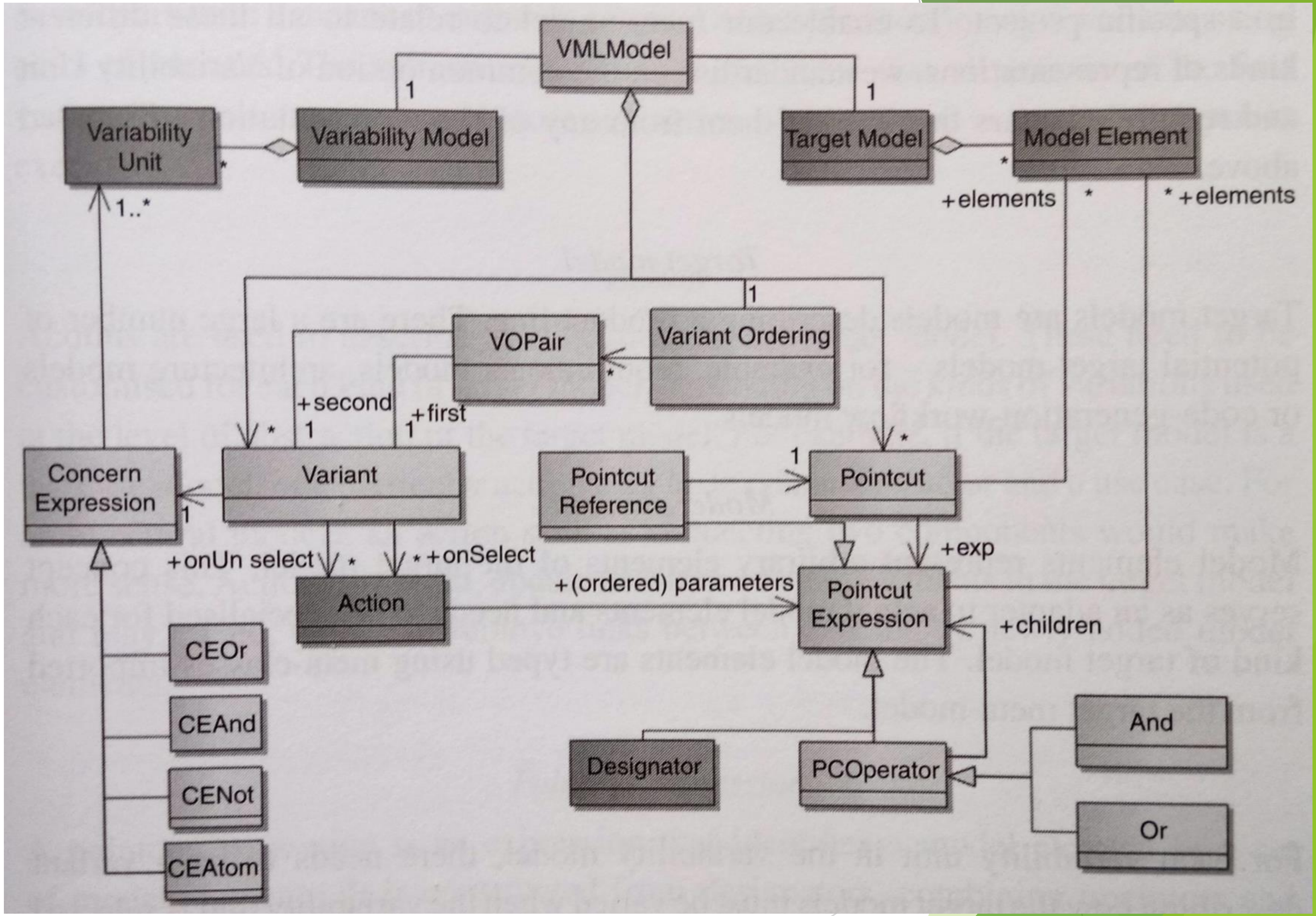
asymmetry, chaos, standalone lines creating non-fractal shape

EASY TO EXECUTE AND ANALYZE FRACTAL SCRIPT IN MANY PROGRAMMING LANGUAGES ***js2py* for Python**

Will it help to enhance third party models and systems?

-improve their accuracy

Meta-model Concepts



Rashid, A., Royer, J., & Rummler, A. (Eds.). (2011). *Aspect-Oriented, Model-Driven Software Product Lines: The AMPLE Way*. Cambridge: Cambridge University Press. doi:10.1017/CBO9781139003629