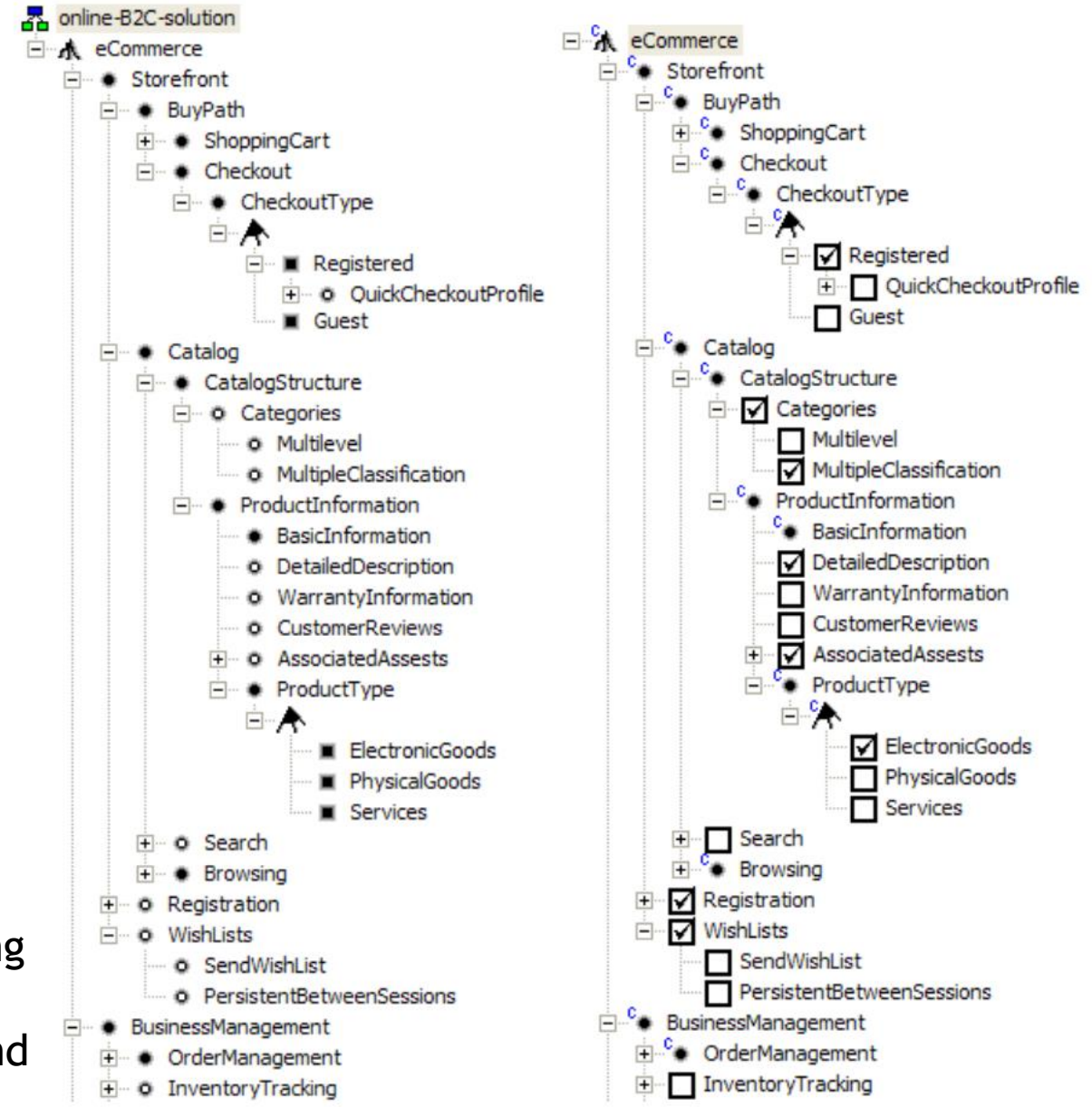
Techniques And Technologies in Variability Management:

From Conditional Compilation, Frame Technology, Framed Aspects, Lightweight Method Towards In-Code Complexity Assessable Constructs

Feature Configuration

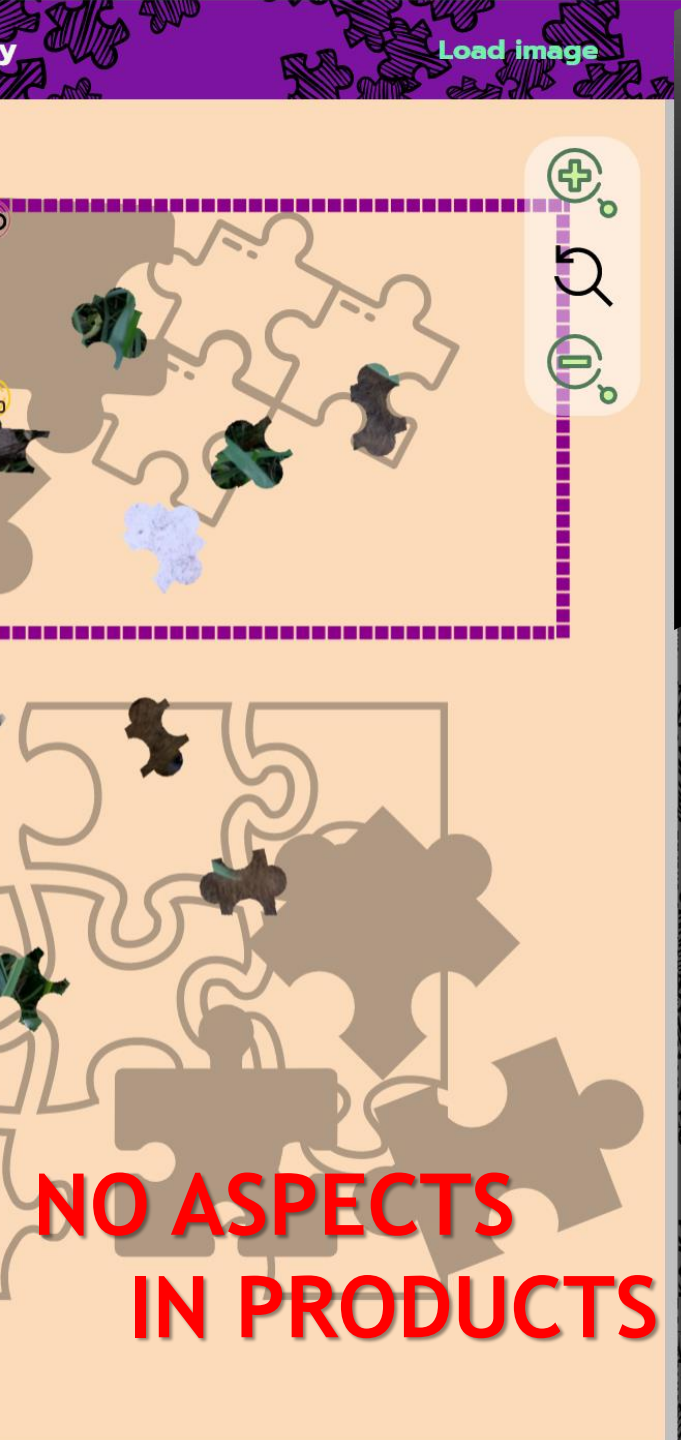


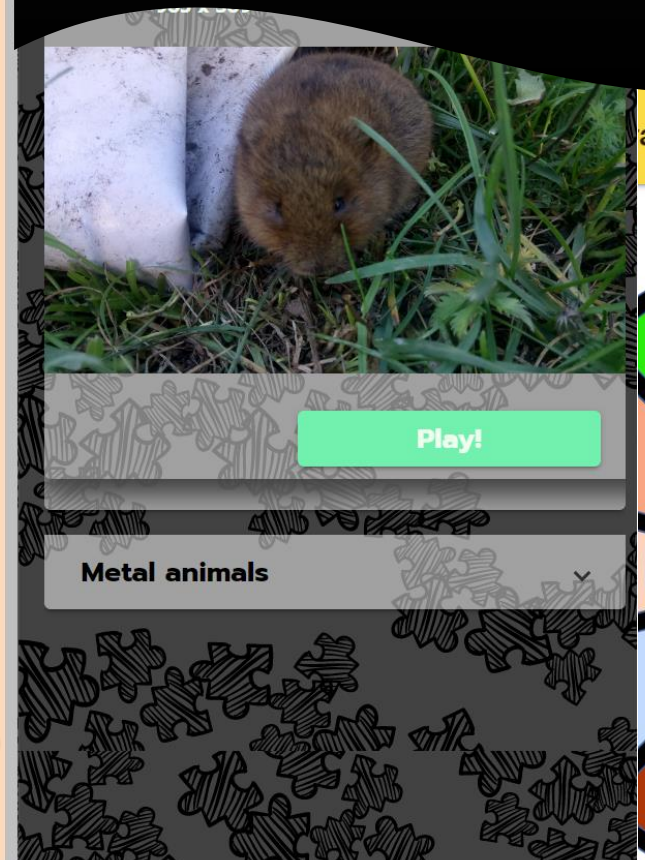
(a) Online B2C Feature Model (b) Feature configuration

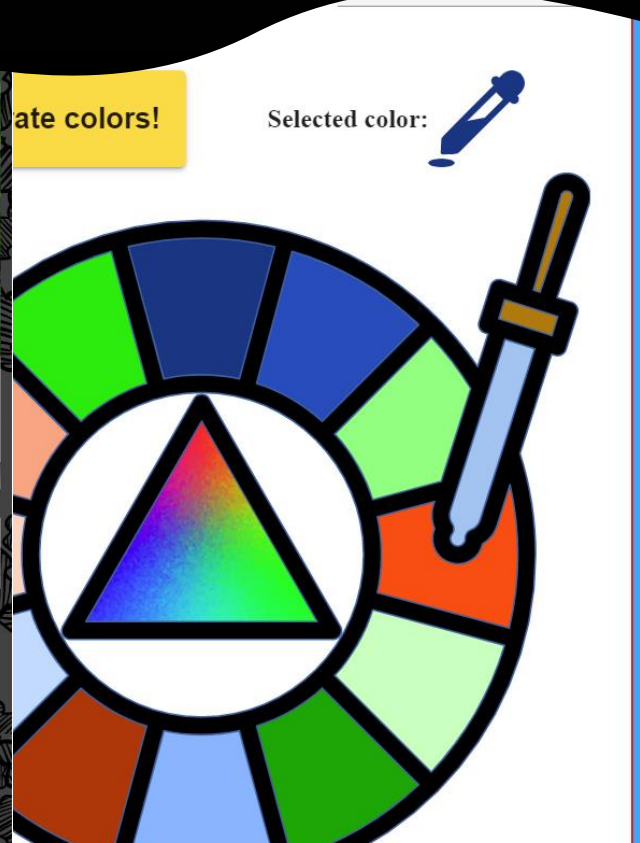
Fig. 1. Sample online B2C feature model and its feature configuration

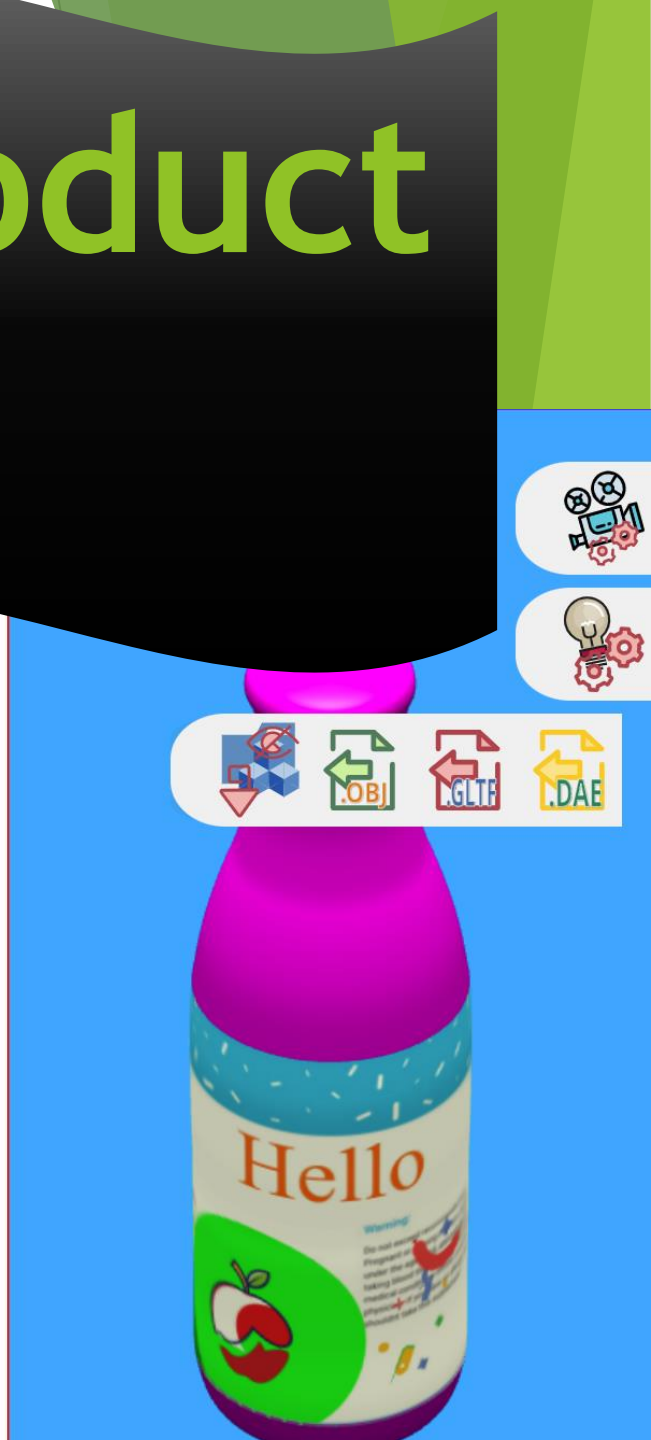
Source: Krzysztof Czarnecki and Michał Antkiewicz. "Map-ping Features to Models: A Template Approach Basedon Superimposed Variants". en. In: Generative Pro-gramming and Component Engineering. Ed. by DavidHutchison et al. Vol. 3676. Series Title: Lecture Notesin Computer Science. Berlin, Heidelberg: SpringerBerlin Heidelberg, 2005, pp. 422–437.

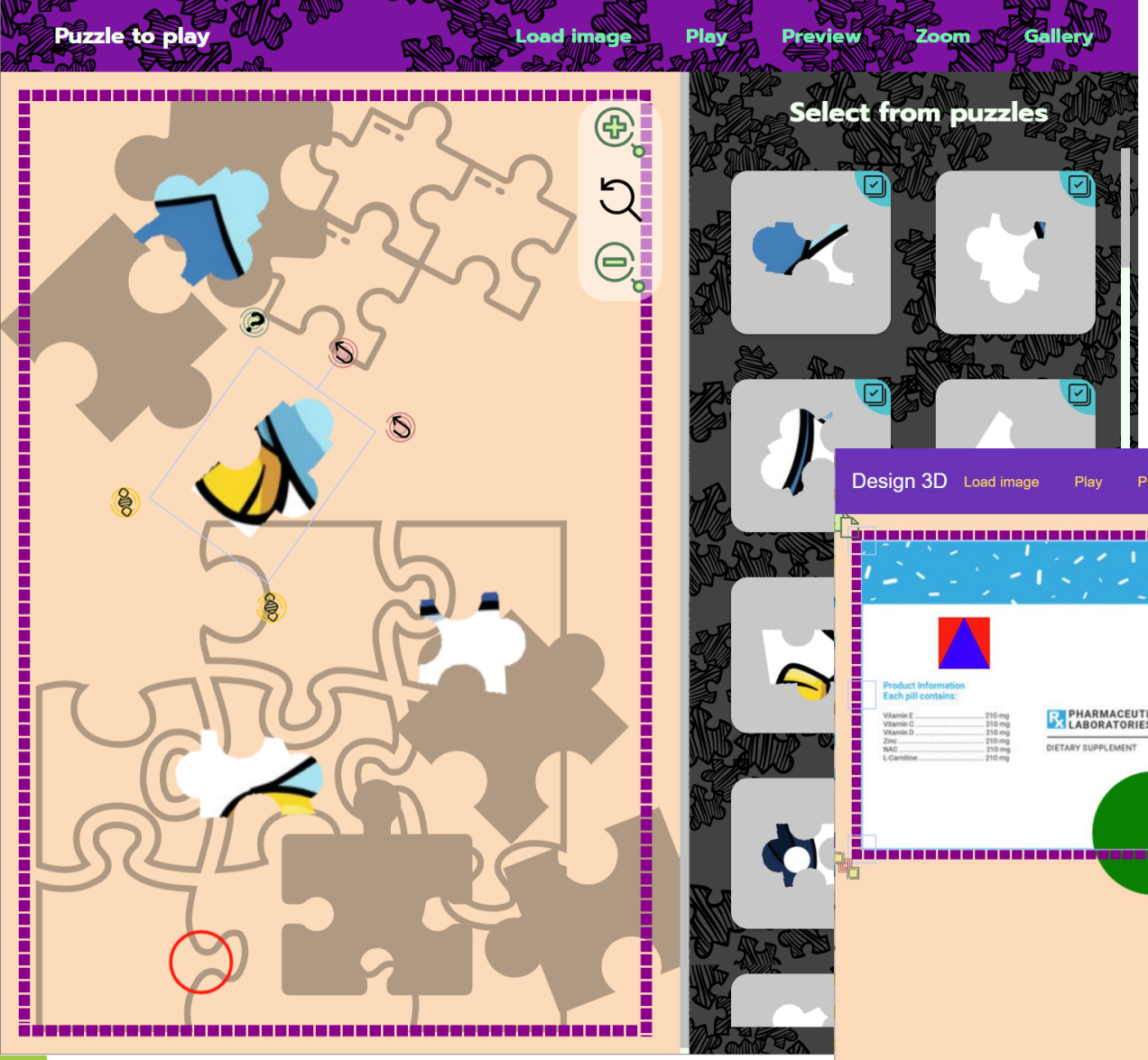
Software product lines



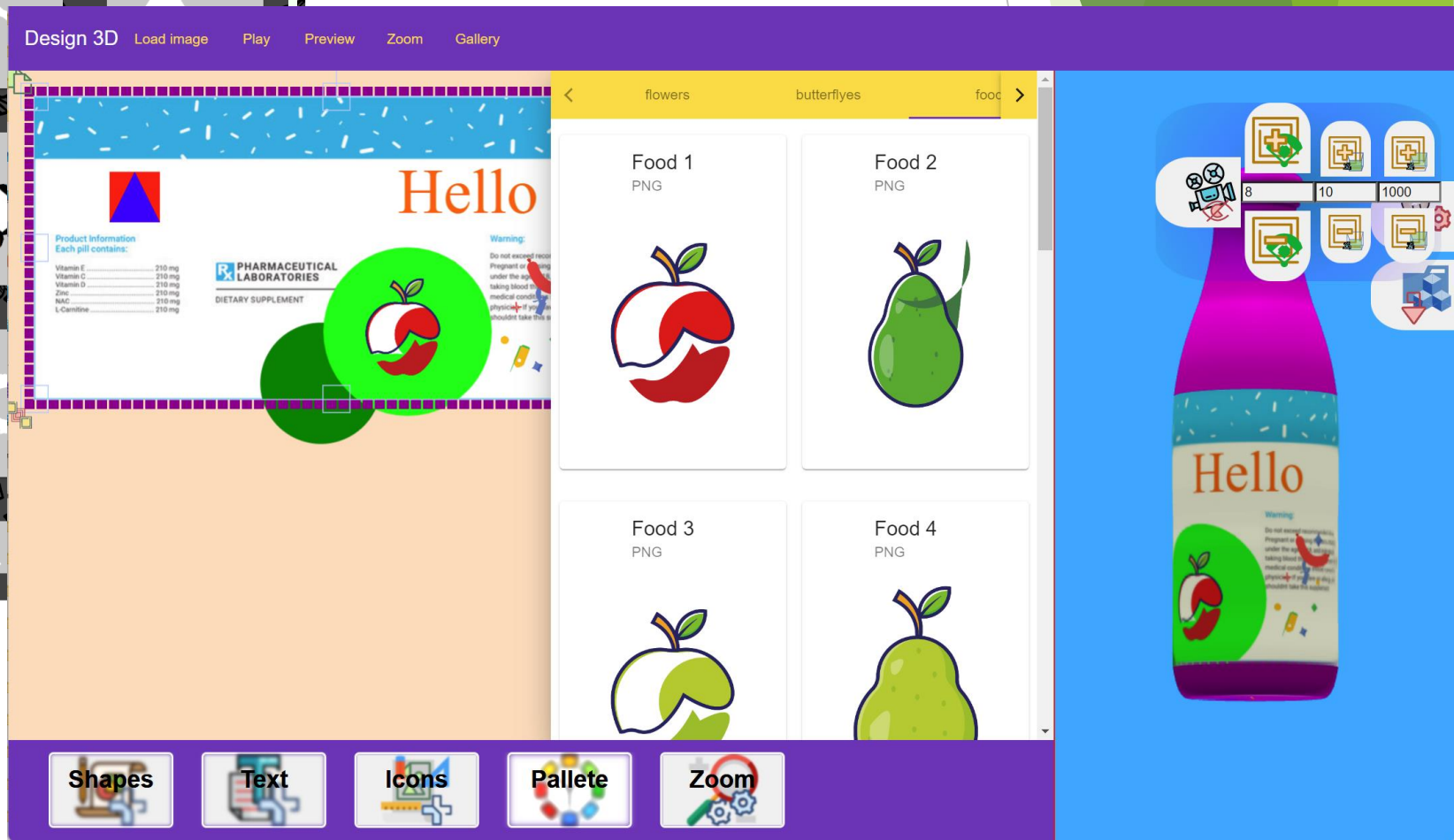






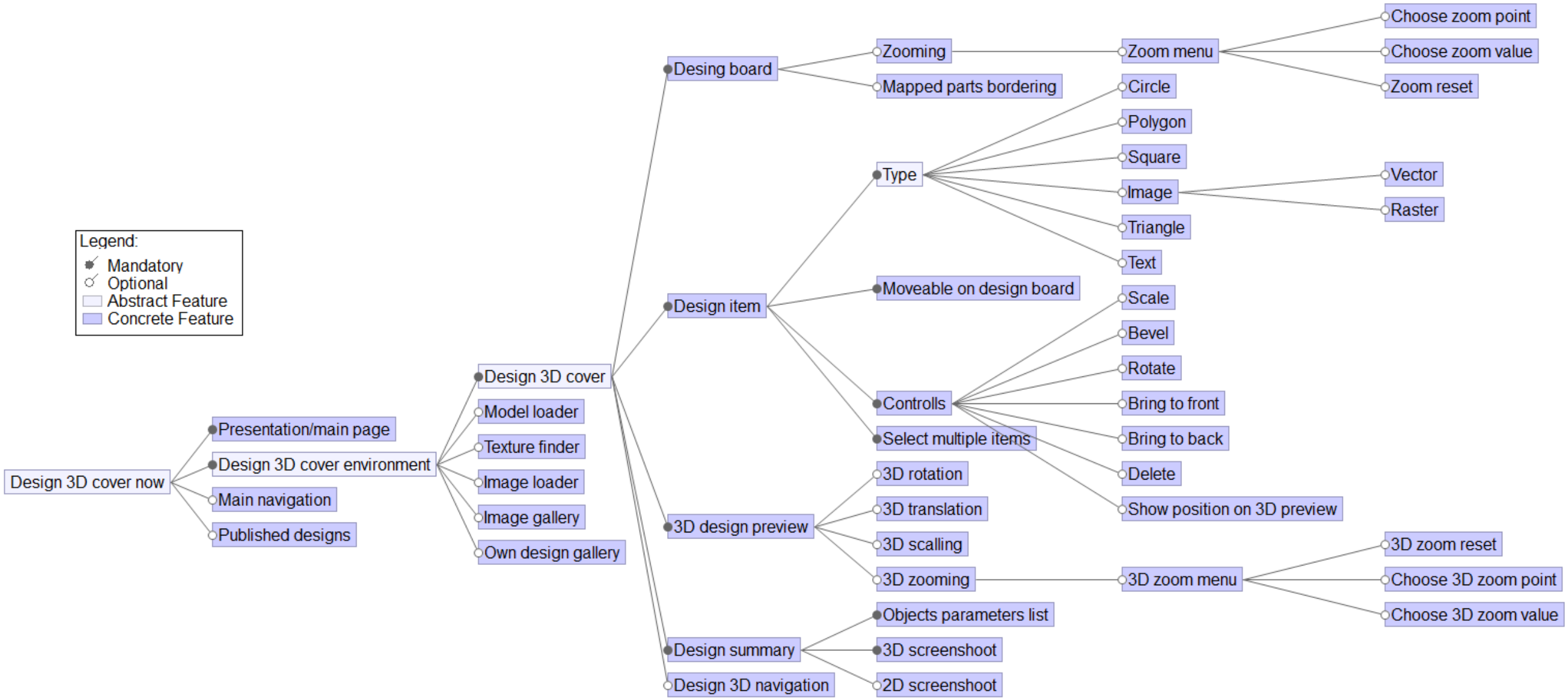


Commonality vs. Variability

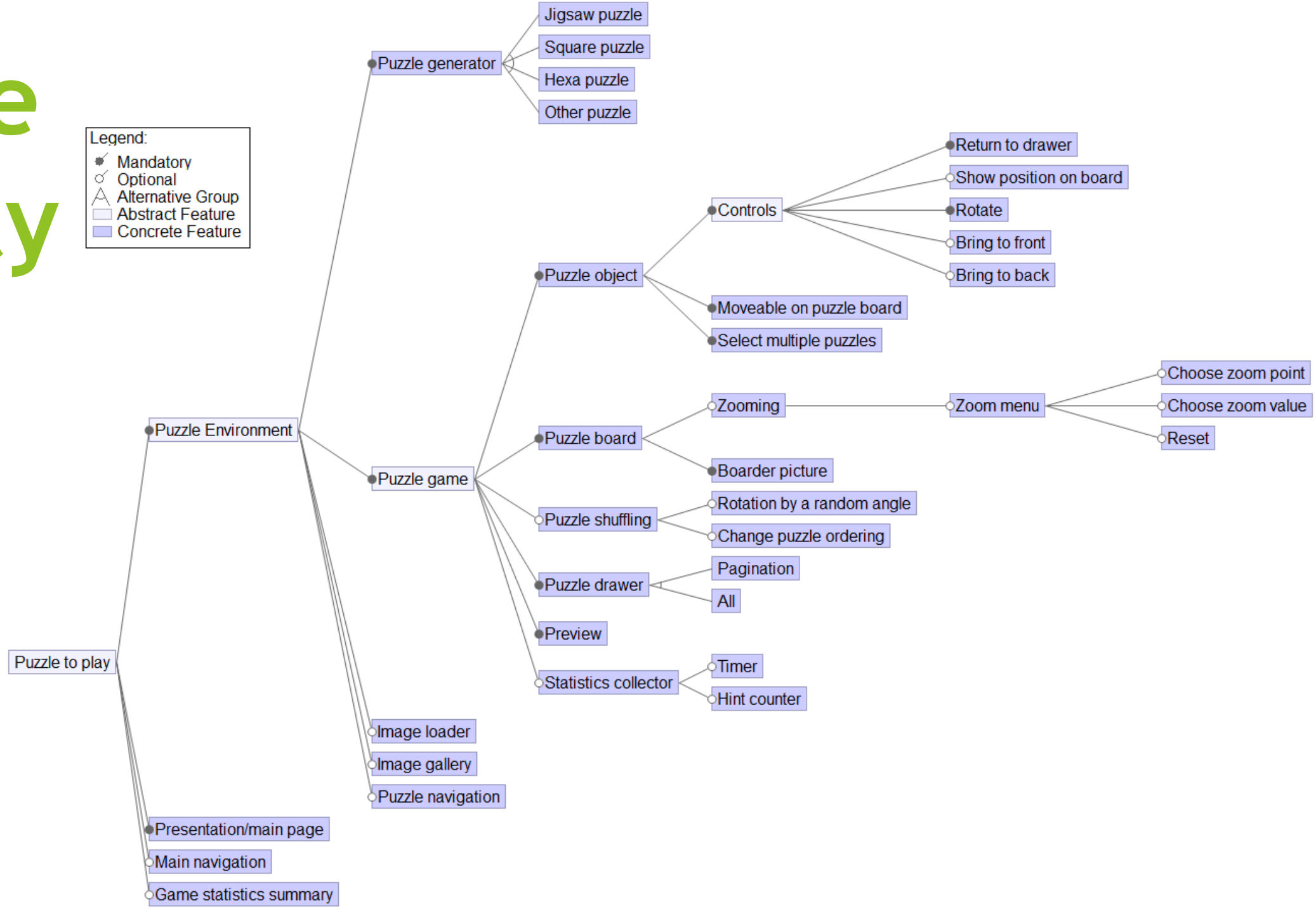


Puzzle app. vs. Desing app.

Design 3D

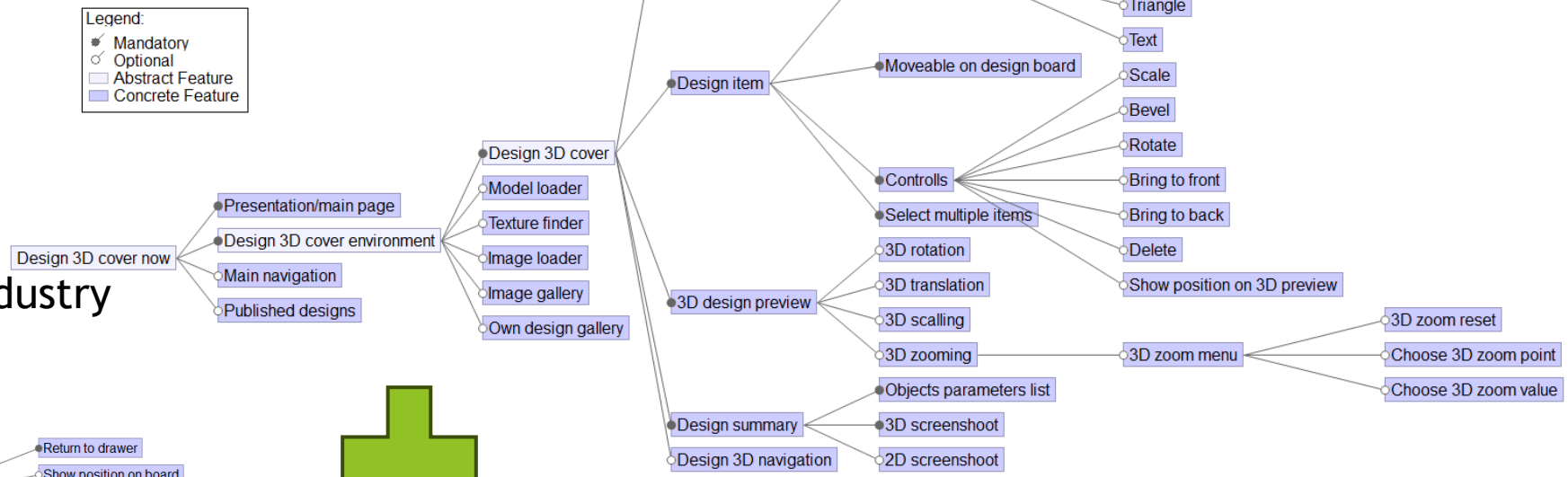

Legend:
Mandatory
Optional
Abstract Feature
Concrete Feature
Design 3D cover now
Presentation/main page
Design 3D cover environment
Main navigation
Published designs
Design 3D cover
Model loader
Texture finder
Image loader
Image gallery
Own design gallery
Desing board
Zooming
Zoom menu
Choose zoom point
Choose zoom value
Zoom reset
Mapped parts bordering
Design item
Type
Circle
Polygon
Square
Image
Vector
Raster
Triangle
Text
Moveable on design board
Controlls
Scale
Bevel
Rotate
Bring to front
Bring to back
Delete
Select multiple items
Show position on 3D preview
3D design preview
3D rotation
3D translation
3D scalling
3D zooming
3D zoom menu
3D zoom reset
Choose 3D zoom point
Choose 3D zoom value
Design summary
Objects parameters list
3D screenshoot
2D screenshoot
Design 3D navigation

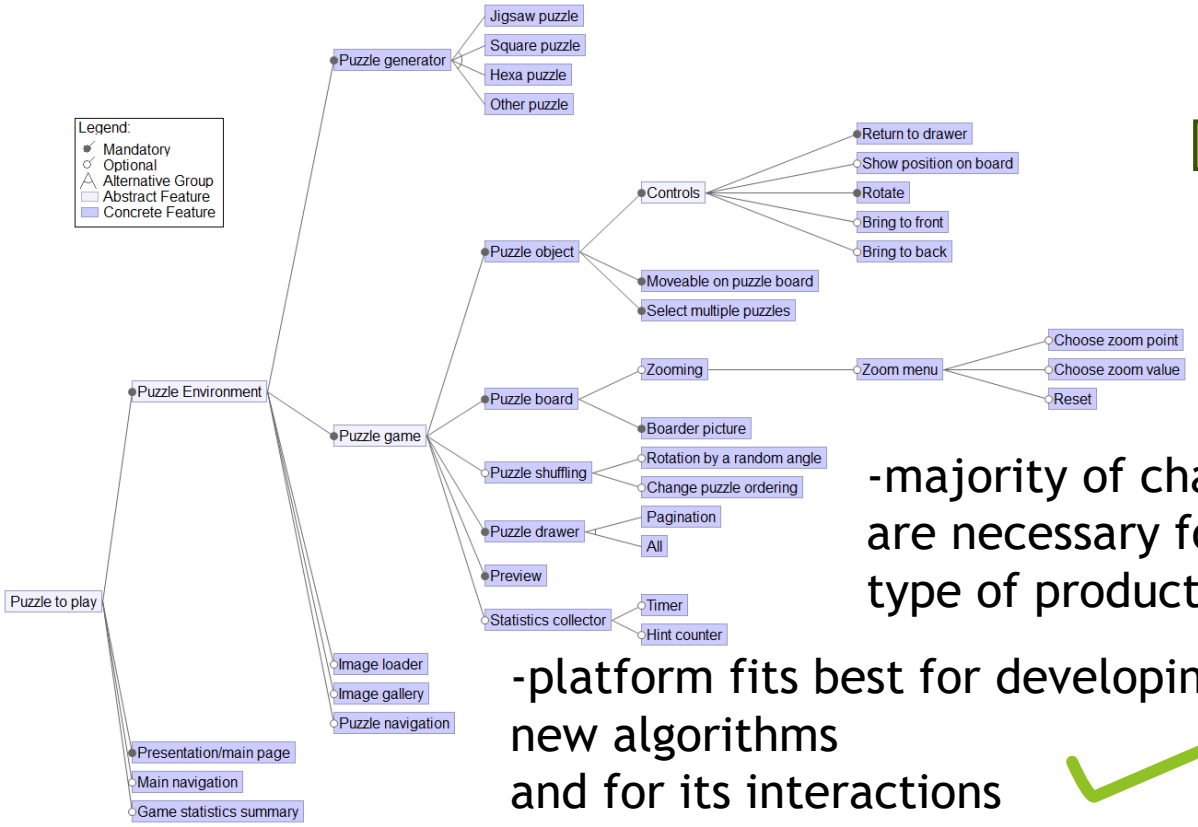
Puzzle To Play



Combining both of their platforms?

-often realized in automobile industry





FOR REUSE

-users wants all features to make final design as good as possible

-majority of changes are necessary for the new type of product

-platform fits best for developing new algorithms and for its interactions

Software Product Line to Produce Stateful Canvas Software Products

Aspect-Oriented Product Lines: Approach

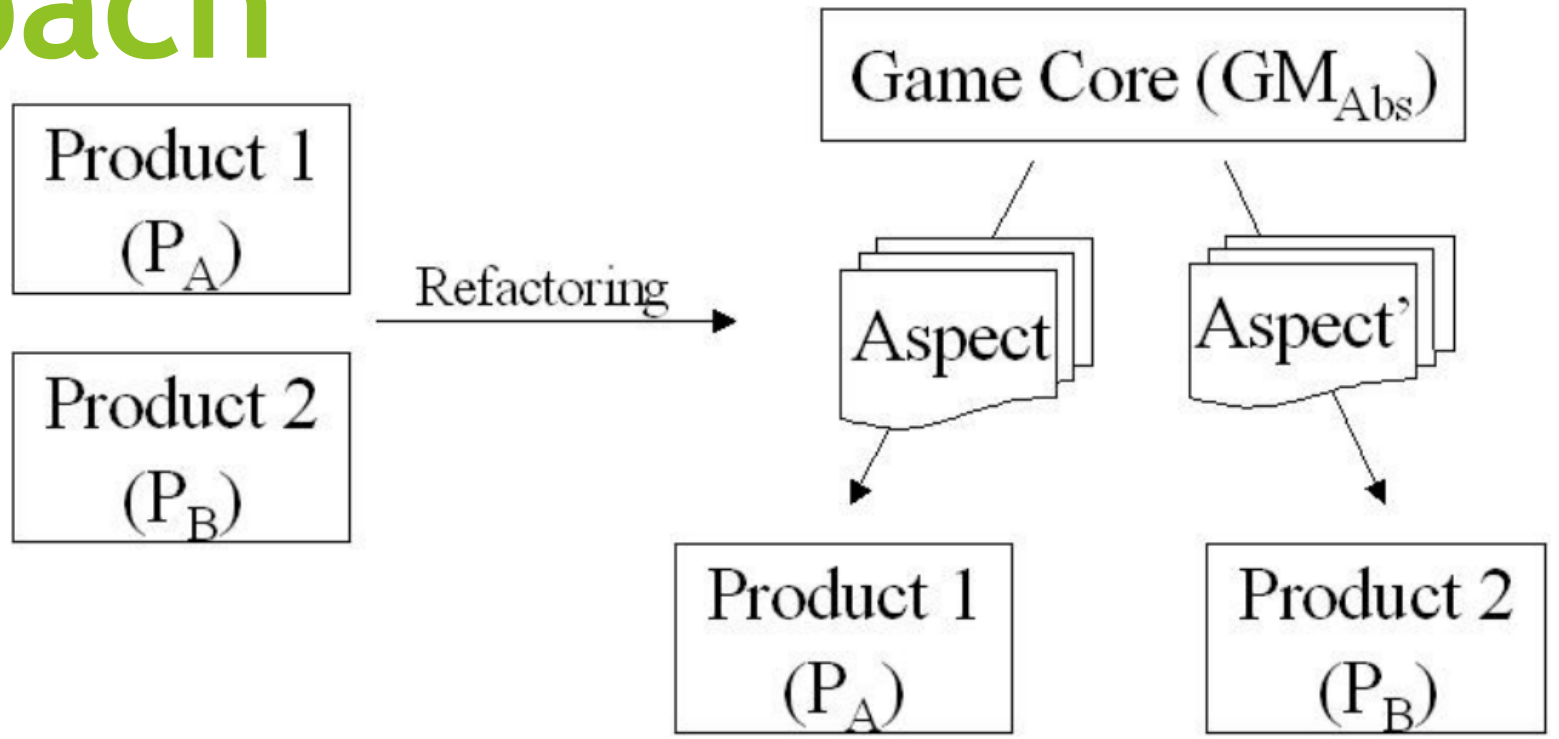
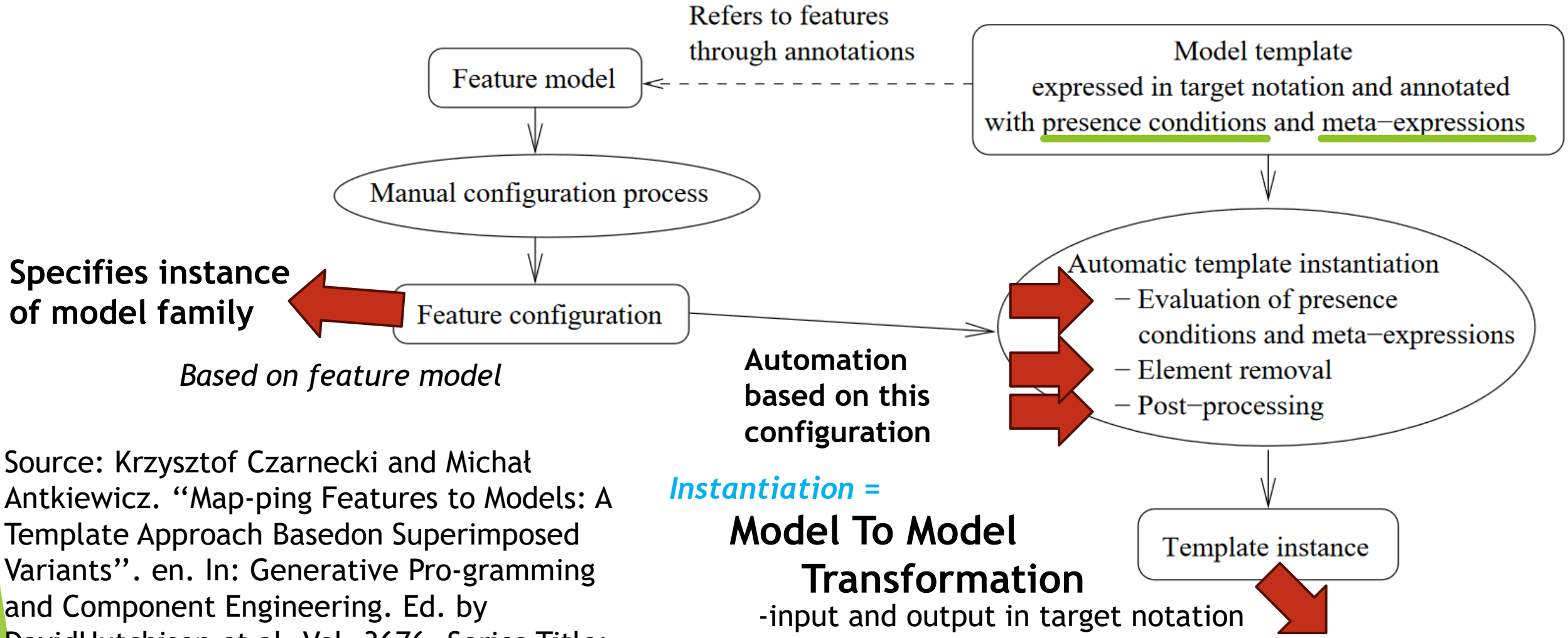


Figure 2: Approach outline

Source: Alves, V., Jr, P.M., Borba, P.: An Incremental Aspect-Oriented ProductLine Method for J2ME Game Development p. 3 (Jan 2004)

Superimposed variants



Specifies instance of model family

Based on feature model

Automation based on this configuration

Instantiation =

Model To Model Transformation

-input and output in target notation

Fig. 2. Overview of the approach

Similar to product derivation

Source: Krzysztof Czarnecki and Michał Antkiewicz. "Map-ping Features to Models: A Template Approach Basedon Superimposed Variants". en. In: Generative Pro-gramming and Component Engineering. Ed. by DavidHutchison et al. Vol. 3676. Series Title: Lecture Notesin Computer Science. Berlin, Heidelberg: SpringerBerlin Heidelberg, 2005, pp. 422–437.

Variability Model

Model

Feature Model

-hierarchic organization of features with constraints on their possible configuration

Model Template

-union of model elements to make valid template instance

Model template elements

annotated with

-defined in terms of feature and feature attributes from feature model
-evaluated in respect to feature configuration

Presence Conditions

-attached to model instance to state if element is present (condition is evaluated as false) or should be removed (false)

Meta Expressions

-used to compute attributes of model elements

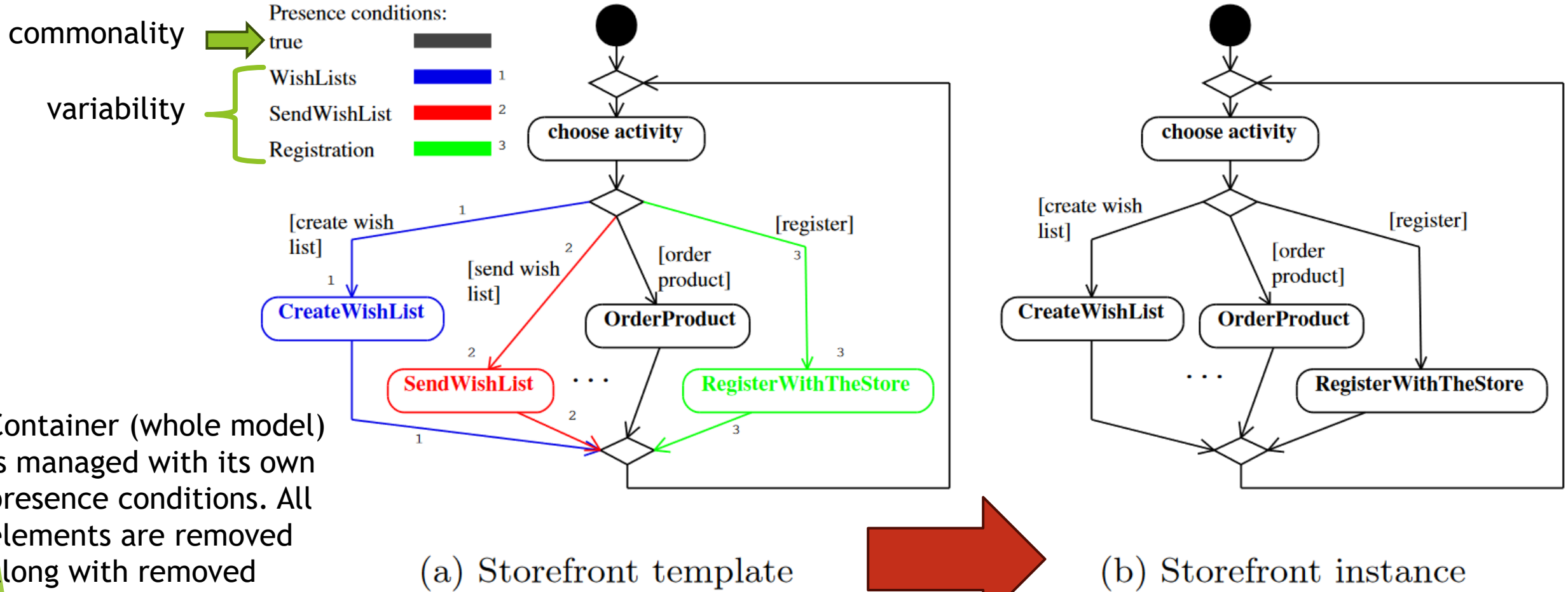
-*element name, return type of operation,...*

Boolean formulas correspond to the features in feature model

Source: Krzysztof Czarnecki and Michał Antkiewicz. “Map-ping Features to Models: A Template Approach Basedon Superimposed Variants”. en. In: Generative Pro-gramming and Component Engineering. Ed. by DavidHutchison et al. Vol. 3676. Series Title: Lecture Notesin Computer Science. Berlin, Heidelberg: SpringerBerlin Heidelberg, 2005, pp. 422–437.

Superimposed Variants: Example

Top-level activity of store-front



Container (whole model) is managed with its own presence conditions. All elements are removed along with removed container.

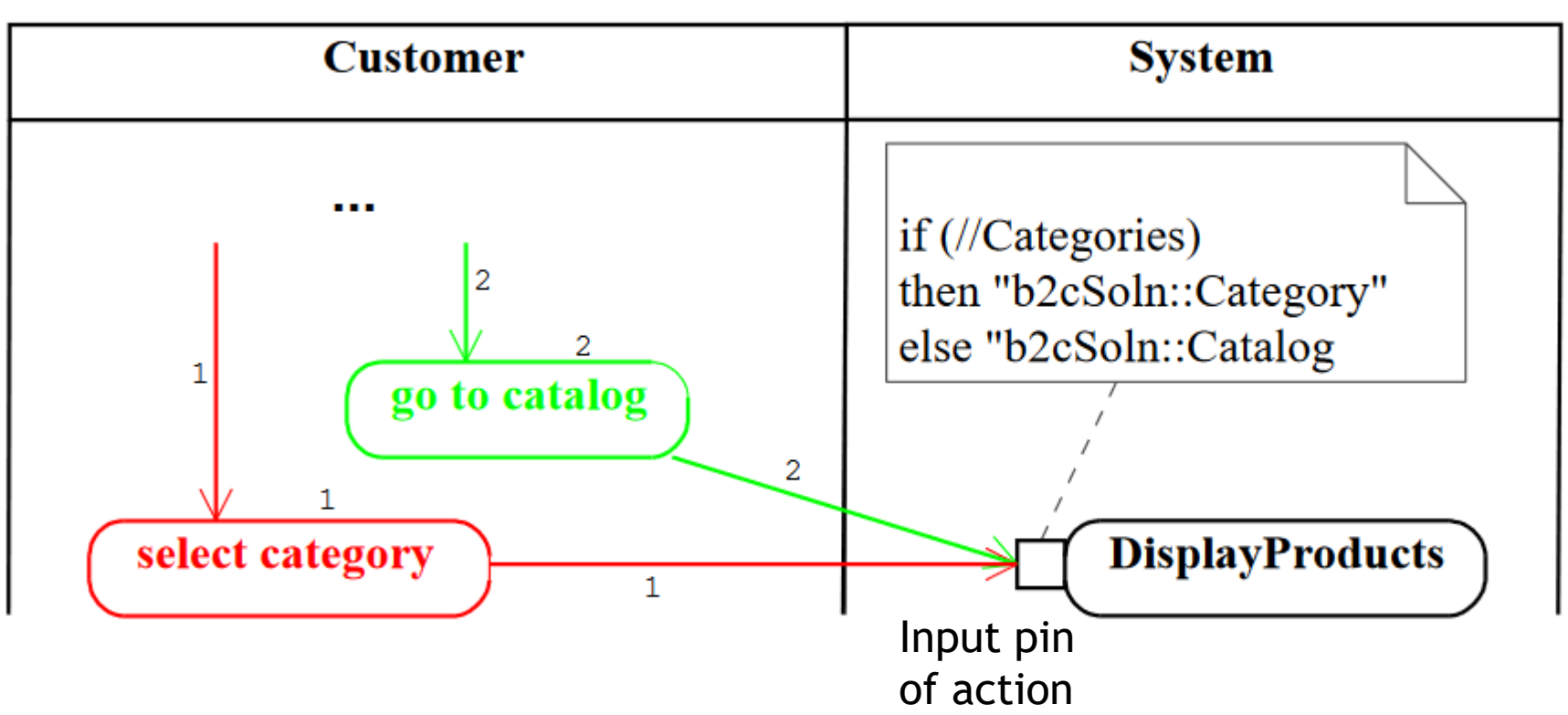
Applicable on XPATH if evaluates on Boolean expression.

instantiation (also derivation in SPL context)

SendWishList not included – as false in configuration

Fig. 3. Sample template activity diagram and its instance

Meta-Expression in Superimposed Variants



Presence conditions:

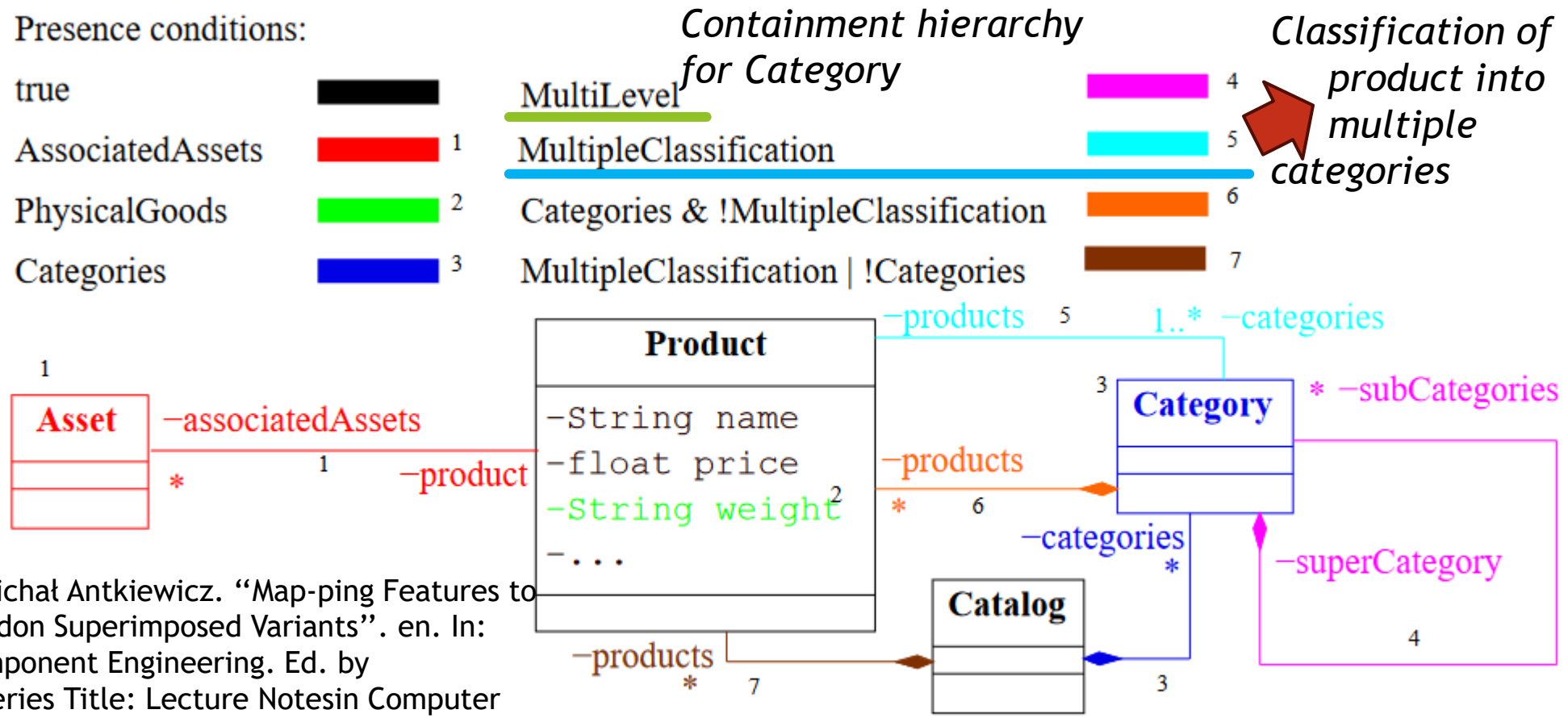
Categories 1

!Categories 2

Source: Krzysztof Czarnecki and Michał Antkiewicz. "Map-ping Features to Models: A Template Approach Basedon Superimposed Variants". en. In: Generative Pro-gramming and Component Engineering. Ed. by DavidHutchison et al. Vol. 3676. Series Title: Lecture Notesin Computer Science. Berlin, Heidelberg: SpringerBerlin Heidelberg, 2005, pp. 422–437.

Fig. 4. Example of a type meta-expression

Class Diagram



Source: Krzysztof Czarnecki and Michał Antkiewicz. "Map-ping Features to Models: A Template Approach Basedon Superimposed Variants". en. In: Generative Pro-gramming and Component Engineering. Ed. by DavidHutchison et al. Vol. 3676. Series Title: Lecture Notesin Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 422–437.

Fig. 5. Example of annotated class diagram

Superimposed Variants: Approach Steps

The realization of our approach for a given target notation involves the following steps:

1. decide on the form of PCs and MEs, for example Boolean formulas and/or XPath expressions;
2. decide on *implicit PCs*. Model elements that are not explicitly annotated by the user will have implicit PCs; implicit PCs will be explained shortly;
3. decide on the annotation mechanism and rendering options for the annotations, e.g., if the target notation is UML, the annotations can be realized as stereotypes; rendering options include labels, icons, and/or coloring;
4. decide on additional processing.

Meta-Expression in Superimposed Variants

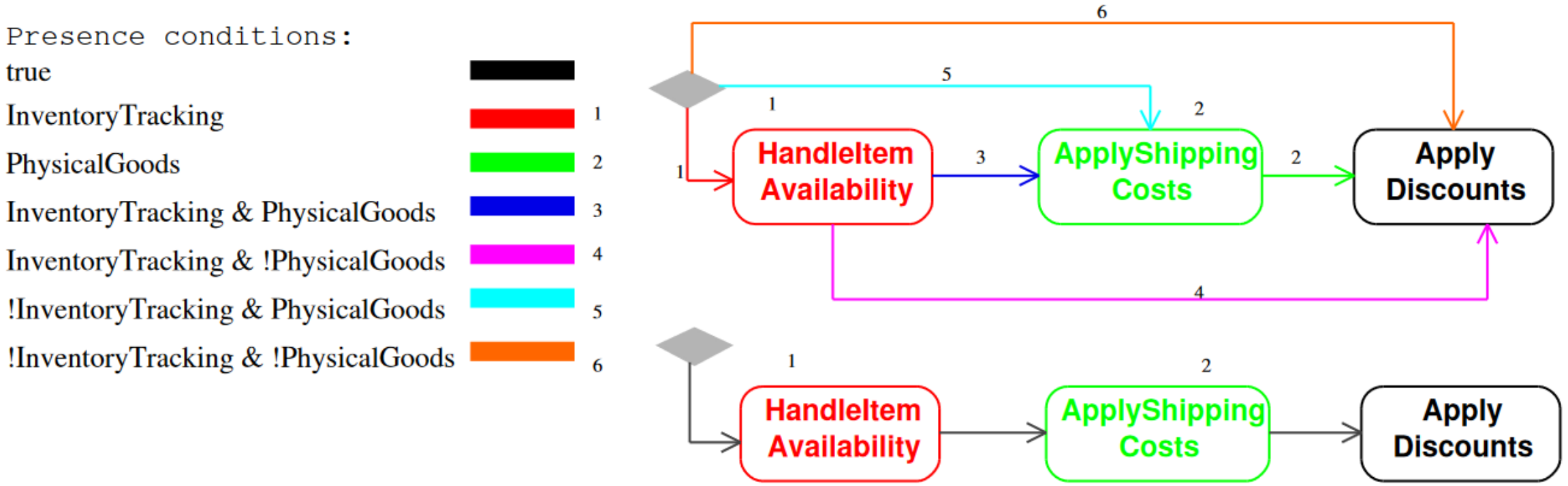


Fig. 6. Model templates with two optional actions without and with automatic flow closure

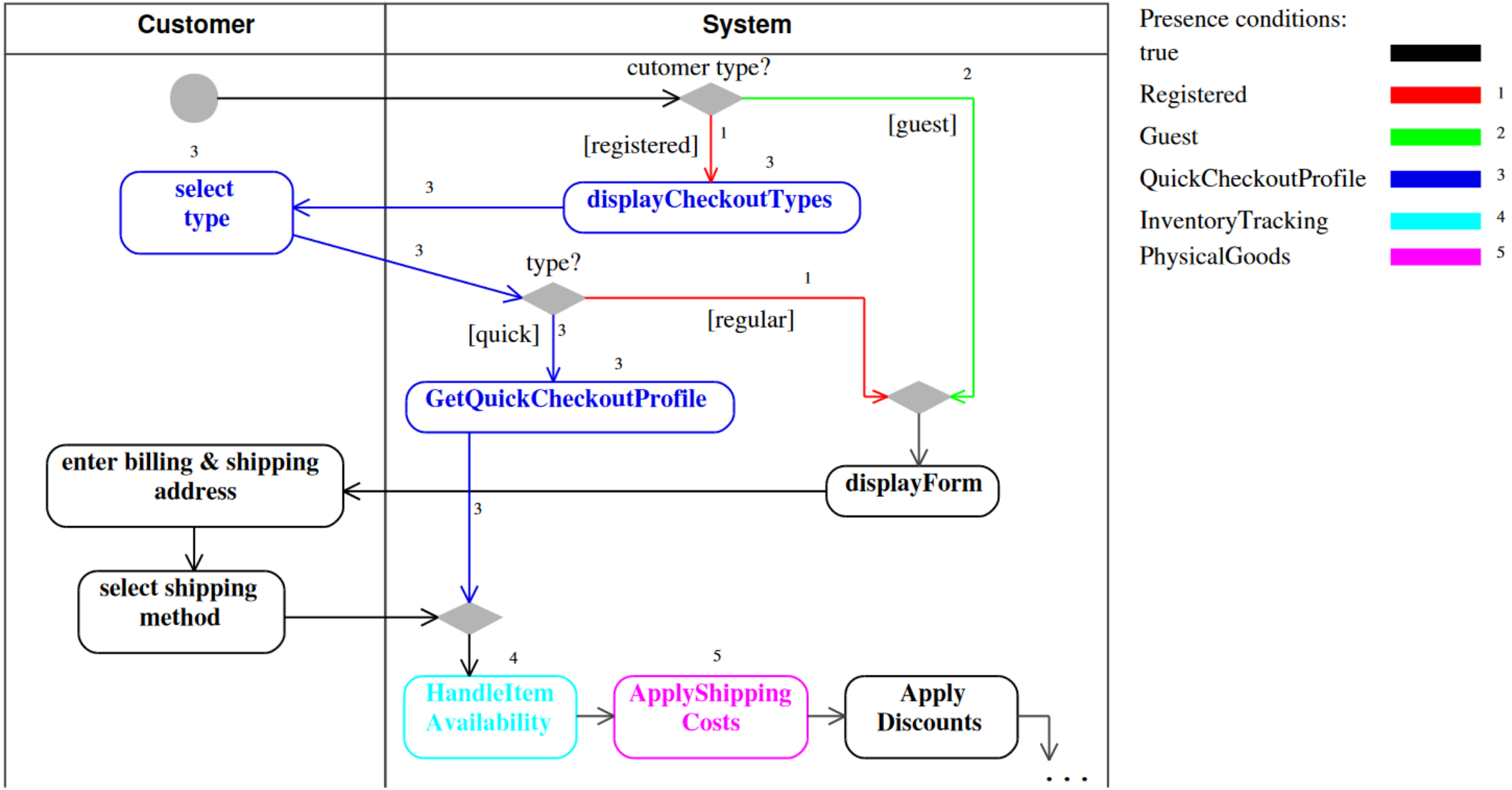


Fig. 7. Checkout Items diagram template

Conditional Compilation

Variability Block 1

Config. Expressions:

copyMedia

Variability Block 2

Config. Expressions:

smsFeature || captureMedia

Variability Block 3

Config. Expressions:

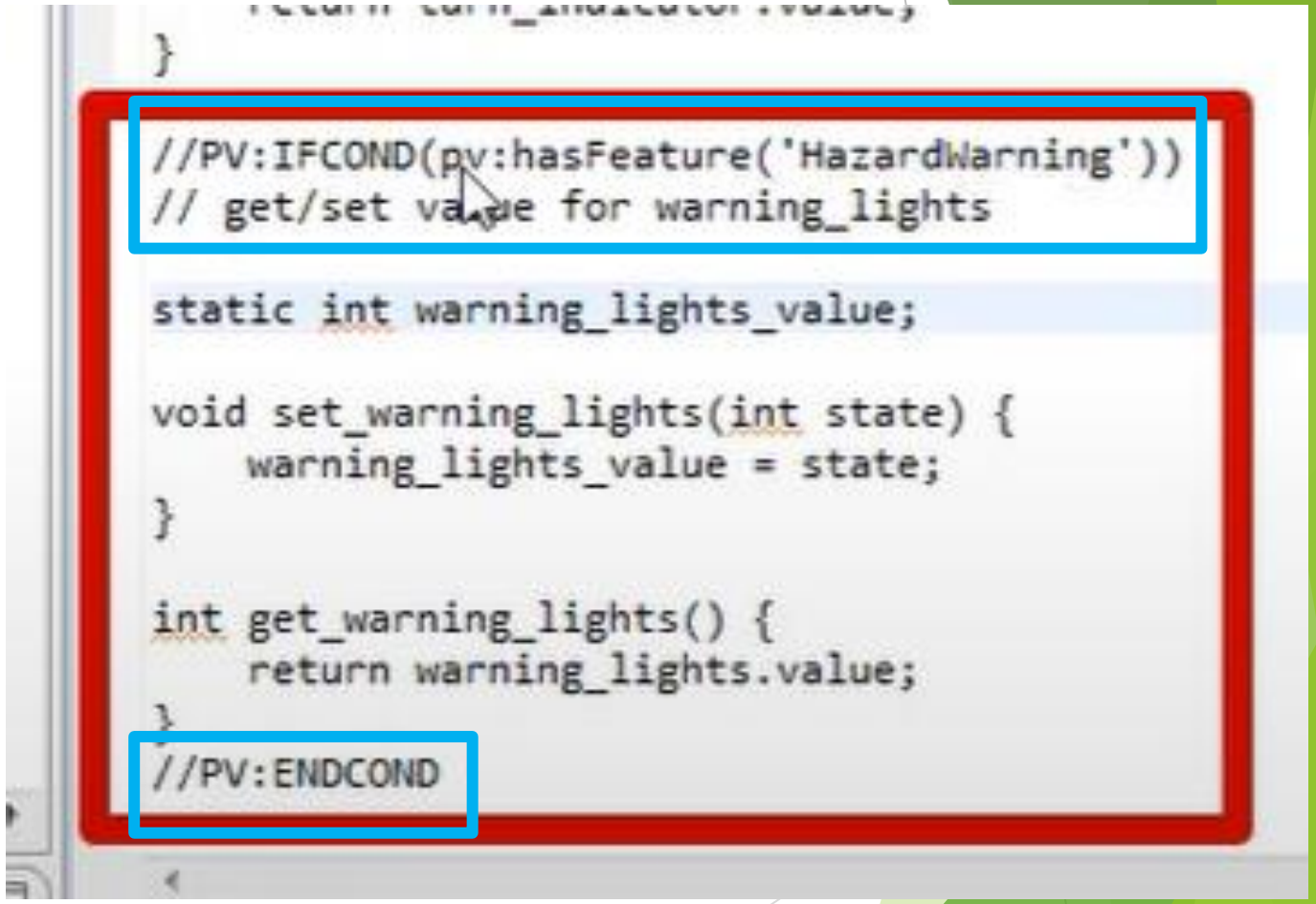
smsFeature || captureMedia

```
01 //#ifdef copyMedia
02 private void processMediaData(String mediaName,
                                 String albumName) {
03   MediaData mediaData = null;
04   //#if smsFeature || captureMedia
05   byte[] mediaByte = getCapturedMedia();
06   if (mediaByte == null)
07   //#endif
08    mediaData = getAlbumData().getMediaInfo(mediaName);
09   //#if smsFeature || captureMedia
10   if (mediaByte != null)
11     getAlbumData().
           addMediaData(mediaName, mediaByte, albumName);
12   else
13   //#endif
14     getAlbumData().addMediaData(mediaData, albumName);
15 }
16 //#endif
```

Figure 1. Variability with conditional compilation

Figueiredo, E., Cacho, N., Sant'Anna, C., Monteiro, M., Kulesza: Evolving software product lines with aspects: An empirical study on design stability. In: Proceedings of 30th international conference on Software engineering, ICSE'08. ACM (2008)

Wrappers in pure::variants



from: pure::systems: PLE & code—managing variability in source code. https: //youtu.be/RlUYjWhJFkM (2020)

Configuration expression in pure::variants

Expression:

```
pv:hasFeature('HazardWarning')
```

Whole variability construct:

```
//PV:IFCOND(pv:hasFeature('HazardWarning'))
// get/set value for warning_lights
```

from: pure::systems: PLE & code—managing variability in source code. https: //youtu.be/RlUYjWhJFkM (2020)

Framed Technology - Overview

- From 1970s
- Language independent textual preprocessor
- To create generalized components
  - Easily adapted or modified to different reuse contexts
  - Based on code templates and specification from developers

Typical Commands/Tags:

<set> - sets a variable

<select> - selects an option

<adapt> - refines a module with new functionality

<while> - creates a loop around repeating code

```
class Editor extends JEditorPane implements HyperlinkListener
  {
  private Network network;
  private Hashtable cache = new Hashtable();
  // .. methods for adding and retrieving data to/from cache
  //.. constructor and editor initialisation

  public void hyperlinkUpdate(HyperlinkEvent e)
    {
    if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED)
      {
      String url = e.getURL().toString();
      Document cachedPage = (Document)getFromCache(url);
      if(cachedPage == null)
        {
        network.requestInfo(this, url);
        addToCache(url, this.getDocument);
        }
       else
        {
        // get record from cache and display it
        this.setDocument((Document)cachedPage.getContent());
        }
      }
    }
  }
```

Source: Loughran, N., Rashid, A., Zhang, W., Jarzabek, S.: Supporting product line evolution with framed aspects p. 5 (2004)

Fig. 1. OO implementation of the Cache feature

pertaining to the hyperlinkEvent requires to update code in both frames

Single System Code - OOP implementation

Framed Technology - variability handled code

```
class Editor extends JEditorPane implements HyperlinkListener
  {
  private Network network;
  <option cache>
  private Hashtable cache = new Hashtable();
  // .. methods for adding and retrieving data to/from cache
  </option>
  //.. constructor and editor initialisation
  public void hyperlinkUpdate(HyperlinkEvent e)
    {
    if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED)
      {
      String url = e.getURL().toString();
      <option cache>
      Document cachedPage = (Document)getFromCache(url);
      if(cachedPage == null)
        {
      </option>
        network.requestInfo(this, url);
        <option cache>
        addToCache(url, this.getDocument);
        }
       else
        {
        // get record from cache and display it
        this.setDocument((Document)cachedPage.getContent());
        }
        </option>
      }
    }
  }
```

Fig. 2. Using frame option tags to identify caching code

Framed Technology - OOP

- Implementation of cache feature using object-oriented programming

```
class Editor extends JEditorPane implements HyperlinkListener
  {
  private Network network;
  private Hashtable cache = new Hashtable();
  // .. methods for adding and retrieving data to/from cache
  //.. constructor and editor initialisation

  public void hyperlinkUpdate(HyperlinkEvent e)
    {
    if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED)
      {
      String url = e.getURL().toString();
      Document cachedPage = (Document)getFromCache(url);
      if(cachedPage == null)
        {
        network.requestInfo(this, url);
        addToCache(url, this.getDocument);
        }
       else
        {
        // get record from cache and display it
        this.setDocument((Document)cachedPage.getContent());
        }
      }
    }
  }
```

Fig. 1. OO implementation of the Cache feature

Source: Loughran, N., Rashid, A., Zhang, W., Jarzabek, S.: Supporting product line evolution with framed aspects p. 5 (2004)

Framed Technology - AOP

- Implementation of cache feature using aspect-oriented programming

UNAVAILABLE PARAMETERIZATION SUPPORT WITH PURE ASPECT-ORIENTED PROGRAMMING

Abstract aspects

+

Concreate aspects (specifying concrete variants)

-can lead to inheritance annomalies

```
aspect CacheAspect
  {
  private Hashtable cache = new Hashtable();
// ..code
  void around(Editor g, String url): args (g,url) &&                    } 1
      call (public void Network.requestInfo(Editor, String))
      {                                                                 } 2
      PageContent cachedPage=(PageContent) cache.get(url);
      if(cachedPage==null)
          {                                                             } 3
           proceed(g,url);
           PageContent page=new PageContent(g.getDocument());
           addToCache(url,page);
           }
      else
           {
           g.setDocument(cachedPage.getContent());
           }
      }
  // inner class for data structures
  }
```

Fig. 3. AOP implementation of the cache using AspectJ

Source: Loughran, N., Rashid, A., Zhang, W., Jarzabek, S.: Supporting product line evolution with framed aspects p. 5 (2004)

Framed Technology With Aspects

-benefitting from the combination of Frame technology and aspects

-processed by **Lancaster Frame Processor (LFP)** [which is essentially a cut down version of the XVCL frame processor]

RESTRICTIONS: -takes only selected frame constructs
-forces programmer to use aspect-oriented techniques

Aspects ⟺ Frame Technology

-to encapsulate and modularize tangled features

-providing parameterization and reconfiguration support for feature aspects

-reduces clutter of template code

-creation of metavariables and options bound to specification from the developer

-to support effective parameterization and reconfiguration

Comparing Frames and AOP

Table 1. Comparing frames and AOP

Capability	Framing with OO	AOP
Configuration Mechanism	Very comprehensive configuration possible	Not supported natively, dependent on IDE
Separation of Concern	Only non crosscutting concerns supported	Addresses problems of crosscutting concerns
Templates	Allows code to be generalised to aid reuse in different contexts	Not supported
Code Generation	Construction time mechanism allows generation of code and refactoring via parameterisation.	Generates code which (in the case of advice) is bound at run time
Language Independence	Supports any textual document and therefore any language	Constrained to implementation language, although language independent AOP forms exist
Use on Legacy Systems	Limited	Supports evolution of legacy systems at source and byte code level
Variation Point Identification	Invasive breakpoints	Non invasive joinpoints
Dynamic Runtime Evolution	Not supported	Possible in JAC and JMangler. Future versions of AspectJ will have support.

Source: Loughran, N., Rashid, A., Zhang, W., Jarzabek, S.: Supporting product line evolution with framed aspects p. 5 (2004)

Framed Aspects

-aspects benefitting from generalization and parameterization
-offering *the best from frames and aspects* such as
-flexibility
-reusability
-evolvability
-improving the integration of features in SPL (crosscutting multiple modules in OO and frames without aspects)

A) CODE AFFECTED WITH FRAMED TAGS

-LOCALIZATION OF CROSSCUTTING CONCERNS
-improving system comprehensibility
-minimising design erosion of architectures

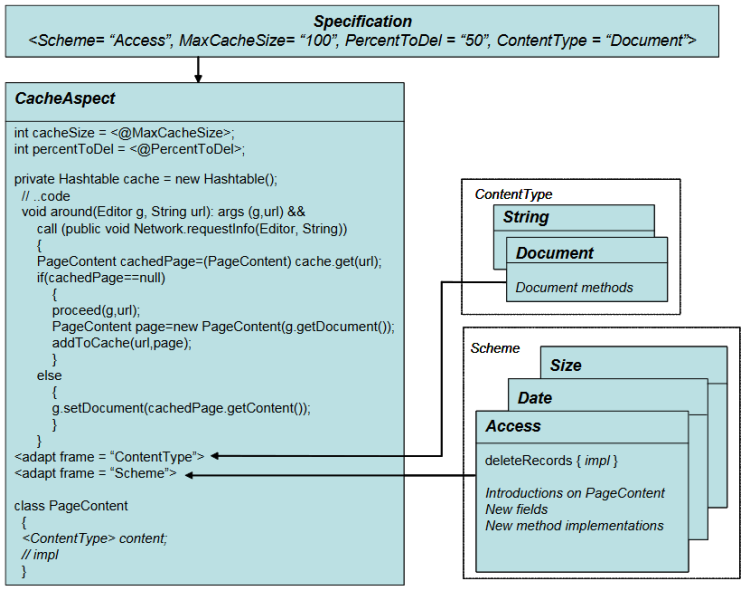


Fig. 4. Using parameterised <adapt> to provide variations in the cache aspect

B) MERGE ALTERNATIVE AND OPTIONAL FEATURES IN TERMS OF CONSTRAINTS

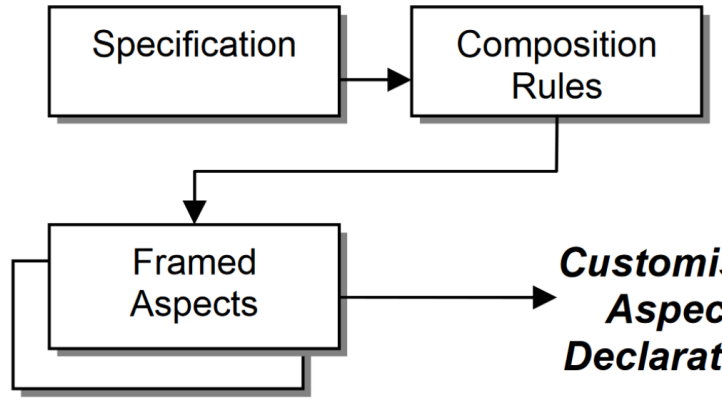


Fig. 2. Framed aspect composition

Source: Loughran, N., Rashid, A., Zhang, W., Jarzabek, S.: Supporting product line evolution with framed aspects p. 5 (2004)

A) CODE AFFECTED WITH FRAMED TAGS

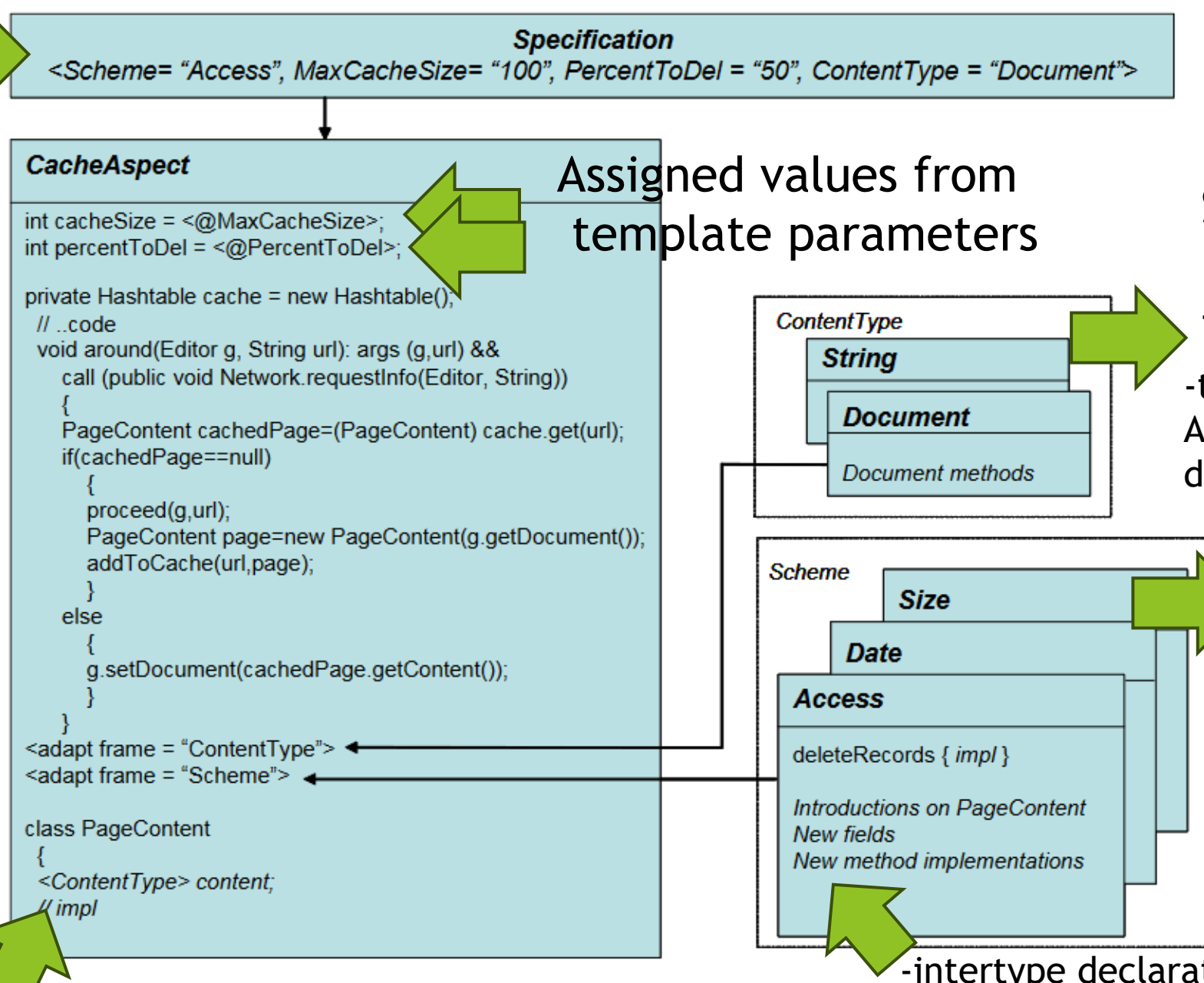
PARAMETERIZED ADAPT:

To frame properties (Scheme, ContentType)

+ incorporation into aspect.

Defining type of object

Configurable parameters



Different Strategies (of content type)

-to do not constrain Aspects on J2SE document

Different Strategies (of scheme)

-predefines data structure within cache and the way how is delete

-intertype declaration

Fig. 4. Using parameterised <adapt> to provide variations in the cache aspect

Framed Aspects

B) MERGE ALTERNATIVE AND OPTIONAL FEATURES IN TERMS OF CONSTRAINTS

-more control over different modules
-for more complex scenarios
-removal more of invasive frame code

-developed methodology to use feature diagram based on FODA:

mapping

REUSABLE COMPONENT → FRAME ASPECT

Source: Loughran, N., Rashid, A., Zhang, W., Jarzabek, S.: Supporting product line evolution with framed aspects p. 5 (2004)

-adapt tags from the framed aspect code to the composition rules)

-the moving of option

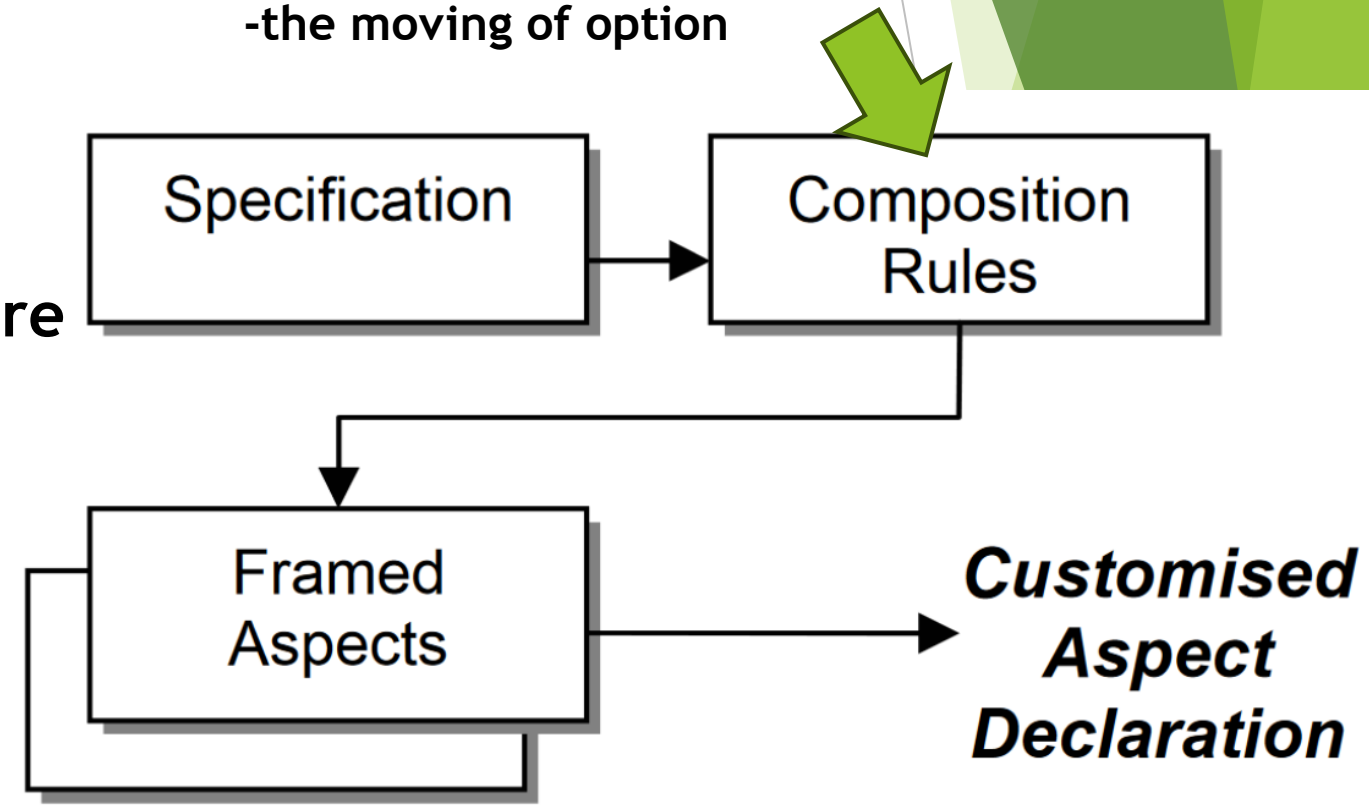


Fig. 2. Framed aspect composition

Framed Aspects Composition

Developer's customization specifications

*-taking incomplete template specification + **filling options and variables that are wished to set***

Maps out possible legal aspect feature:

compositions
combinations
constraints
control flows

How these bound together

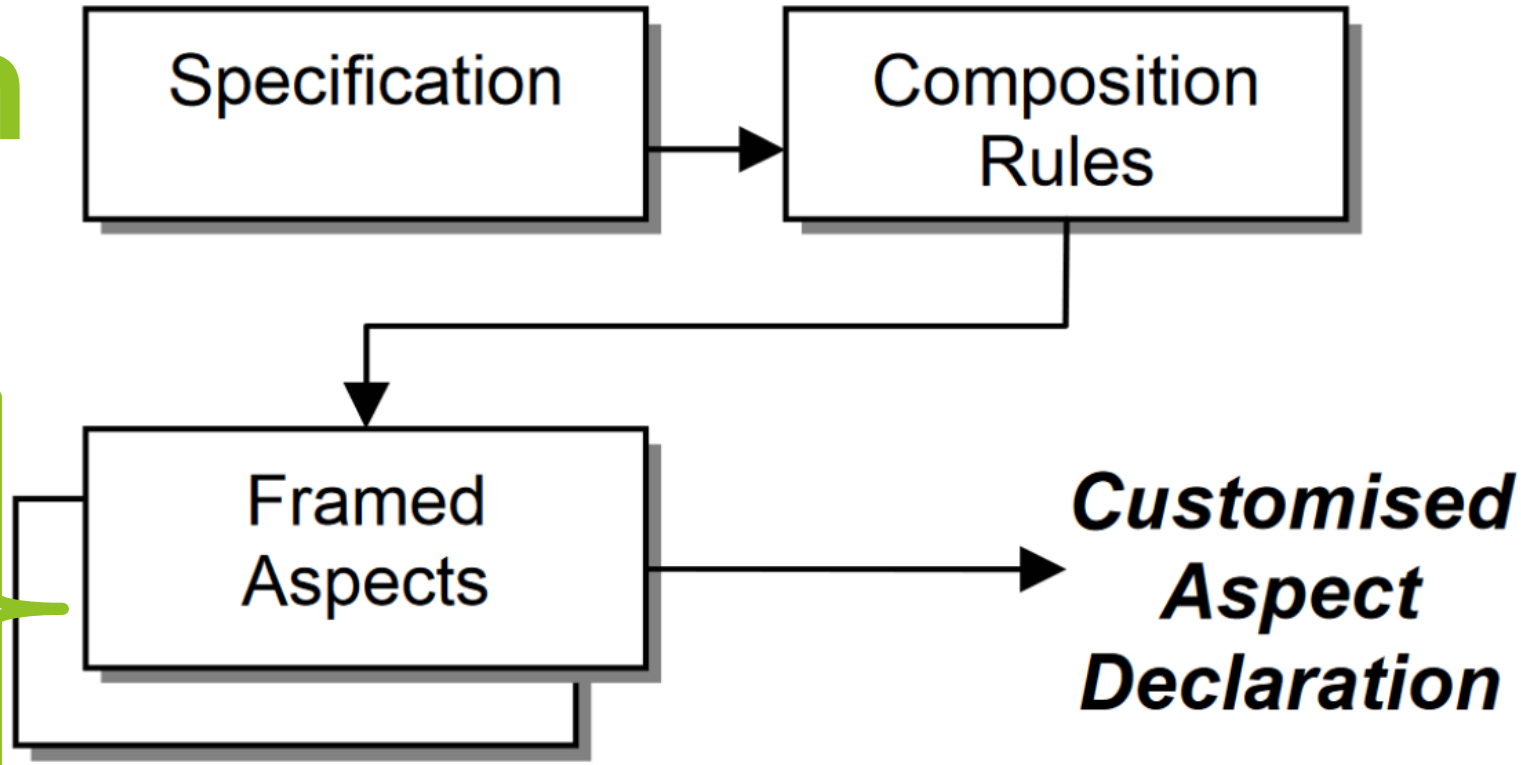


Fig. 2. Framed aspect composition

Made of 3 distinct modules:

A) **Framed Aspect Code**
B) **Composition Rules**
C) **Specification**

Normal aspect code

+

Parameterised aspect code (Framed Aspects)

Process

```
<frame name = "CACHE SPEC">

<select option = "CACHE_TYPE" value = "DATABASE_CACHE" />
<set var = "MAX_CACHE_SIZE" value = "1000" />
<select option = "DELETION_SCHEME" value = "ACCESS" />
<set var = "PERC_TO_DEL" value = "50" />
<set var = "CONN_CLASS" value = "DBConnection" />
<set var = "SEND_QUERY" value = "sendQuery" />
<set var = "REPLY_CLIENT" value = "replyToClient" />
<set var = "DOC_TYPE" value = "String" />

<select option = "WRITABLE" value = "TRUE" />
<select option = "DB_UPDATE_SCHEME" value = "EVERYWRITE" />

<adapt frame = "CACHE_RULES"/>                                    4

</frame>
```

Fig. 9. Typical specification frame for a database

```
<frame name = CACHE_RULES>

<constrain var = "CACHE_TYPE" toSet = "DATABASE_CACHE,WEB_CACHE"/>
<constrain var = "PERC_TO_DEL" toBoundary = "25,100"/>
<constrain var = "DELETION_SCHEME" toSet = "ACCESS,DATE,SIZE"/>
<constrain var = "DB_UPDATE_SCHEME" toSet = "EVERYWRITE,TIME_BASED"/>
<constrain var = "WEB_UPDATE_SCHEME" toSet = "AUTOMATIC,MANUAL"/>                    1

<adapt frame = "CACHE_TYPE"/>
<adapt frame = "DELETION_SCHEME"/>                                                   2

<option name = "MANUAL_CACHE_CLEAR" value = "TRUE">                                  3
 <adapt frame = "MANUAL_CACHE_CLEAR" />
</option>

<option name = "CACHE_TYPE" value = "DATABASE_CACHE">
 <option name = "WRITABLE" value = "TRUE">                                           4
       <adapt frame = "DB_UPDATE_SCHEME"/>
 </option>
</option>
```

Fig. 10. Composition rules for generic cache

```
<frame name = "CACHE">

private int MAX_CACHE_SIZE = <@MAX_CACHE_SIZE>;
private int PERCENTAGE_TO_DEL = <@PERC_TO_DEL>;                                      1
private Hashtable cache = new Hashtable(MAX_CACHE_SIZE);

public void addToCache(String key, <@DOC_TYPE> data)
  {  //..impl.. }

pointcut QUERY_PCT(String key, <@CONN_CLASS> c) : this(c) && args                    2
  (key)&& call(public void<@CONN_CLASS>.<@SEND_QUERY>(String));

pointcut RESULT_PCT(<@DOC_TYPE> data, <@CONN_CLASS> c):
  !within(Cache) && this(c) && args(data) &&                                         3
  call (public void @CONN_CLASS"/>.<@REPLY_CLIENT>(<@DOC_TYPE>));

class CacheDS
  {
  private <@DOC_TYPE> data;
  public CacheDS(<@DOC_TYPE> d)                                                      4
    {
    data = d;
    }
  public <@DOC_TYPE> getData()
    {
    return data;
    }
  }
```

```
<frame name = "WRITABLE">

pointcut DS_INSTANCE_PCT(<@CONN_CLASS> c) : cflow (this(c)) &&                       1
 call(CacheDS.new(..)) ;

pointcut RESULTSET_PCT(<@CONN_CLASS> c) : this(c) &&                                 2
 call (ResultSet Statement.executeQuery(String));

after(<@CONN_CLASS> c) returning(CacheDS cds): DS_INSTANCE_PCT(c)                    3
 {
 cds.setTables(c.getTables());
 }

after(<@CONN_CLASS> c) returning(ResultSet rs): RESULTSET_PCT (c)
 {                                                                                   4
 try
     {
     ResultSetMetaData rsmd = rs.getMetaData();
     c.setTables(getTablesFromMetaData(rsmd));
```

Parame-terized Variants

```
private int MAX_CACHE_SIZE = <@MAX_CACHE_SIZE>;
private int PERC_TO_DEL = <@PERC_TO_DEL>;
pointcut pc1(<@EDITOR_NAME> g,String url):args(g,url) &&
call (public void <@NETWORK_CLASS>.<@REQUEST_MTHD>
  (<@EDITOR_NAME>, String));
void around((<@EDITOR_NAME> g, String url):  pc1(g,url)
   { // impl }
class PageContent
   {
   private <@DOC_TYPE> data;
   private int accesses=0;
   public PageContent(<@DOC_TYPE>  d) { data =d; }
   public <@DOC_TYPE>  getData() { accesses++; return data }
   public int getAccesses() { return accesses; }
   }
```

Fig. 4. Parameterised version of simple caching aspect

- **MAX_CACHE_SIZE:** Sets the maximum size of records the cache will hold.
- **PERC_TO_DEL:** The amount of records to delete when the deletion mechanism is invoked.
- **CONN_CLASS:** The class which contains the methods for sending the query to the database and also sending the results back to the client.
- **SEND_QUERY:** The method which sends the query to the database.
- **REPLY_CLIENT:** The method which sends the result back to the client.
- **DOC_TYPE:** The type of information that is being stored in the cache (e.g. String, Document, CachedResultSet etc.).

Loughran, N., Rashid, A.: Framed aspects: Supporting variability and configurability for AOP. In: Proceedings of 8th International Conference onSoftware Reuse, ICSR 2004. LCNS 3107, Springer, Madrid, Spain (2004)

```
private int MAX_CACHE_SIZE = <@MAX_CACHE_SIZE>;
private int PERC_TO_DEL = <@PERC_TO_DEL>;
pointcut pc1(<@EDITOR_NAME> g,String url):args(g,url) &&
call (public void <@NETWORK_CLASS>.<@REQUEST_MTHD>
  (<@EDITOR_NAME>, String));
void around((<@EDITOR_NAME> g, String url):  pc1(g,url)
   { // impl }
class PageContent
   {
   private <@DOC_TYPE> data;
   private int accesses=0;
   public PageContent(<@DOC_TYPE>  d) { data =d; }
   public <@DOC_TYPE>  getData() { accesses++; return data }
   public int getAccesses() { return accesses; }
   }
```

Fig. 4. Parameterised version of simple caching aspect

-used in any textual representation to make substantiation of type or object,

a method, joinpoint, or pointcut designator

AspectJ Cache Code

```
aspect SimpleCacheAspect
  {
  private int MAX_CACHE_SIZE = 100;
  private int PERC_TO_DEL = 50;
  private Hashtable cache = new Hashtable();

  pointcut pc1(Editor g, String url) : args (g,url) &&
     call (public void Network.requestInfo(Editor, String));

  void around(Editor g, String url):  pc1(g,url)
    {
    PageContent cachedPage=(PageContent) cache.get(url);
    if(cachedPage==null)
      {
      proceed(g,url);
      PageContent page=new PageContent(g.getDocument());
      addToCache(url,page);
      }
    else
      g.setDocument(cachedPage.getContent());
     }
  class PageContent {
    private Document data;
    private int accesses=0;
    public PageContent(Document d) { data = d; }
    public Document getData() { accesses++; return data; }
    public int getAccesses() { return accesses; }
    }
  }
```

Creation of **pc1 pointcut** on the method **requestInfo** in **Network** class.

Around advice which executes whenever the method defined in **pc1** is called. If record doesn't exist in cache, **proceed** with original call then store the result in the cache. Otherwise populate editor with cached content.

Data structure for storing **Document** content of editor pane. The access scheme is incremented every time the document is accessed from the cache.

Fig. 1. A simple editor pane caching aspect in AspectJ

```
<frame name = "CACHE">

private int MAX_CACHE_SIZE = <@MAX_CACHE_SIZE>;
private int PERCENTAGE_TO_DEL = <@PERC_TO_DEL>;
private Hashtable cache = new Hashtable(MAX_CACHE_SIZE);

public void addToCache(String key, <@DOC_TYPE> data)
  {  //..impl.. }

pointcut QUERY_PCT(String key, <@CONN_CLASS> c) : this(c) && args
  (key)&& call(public void<@CONN_CLASS>.<@SEND_QUERY>(String));

pointcut RESULT_PCT(<@DOC_TYPE> data, <@CONN_CLASS> c):
  !within(Cache) && this(c) && args(data) &&
  call (public void @CONN_CLASS"/>.<@REPLY_CLIENT>(<@DOC_TYPE>));

class CacheDS
  {
  private <@DOC_TYPE> data;
  public CacheDS(<@DOC_TYPE> d)
    {
    data = d;
    }
  public <@DOC_TYPE> getData()
    {
    return data;
    }
  }
```

Sets the size of the cache and percentage to be deleted as set by the parameters in the specification.

Creates a pointcut for intercepting the call to the method which executes SQL queries on the database.

Creates a pointcut for intercepting the results sent back to the client.

CacheDS is a data structure for storing the cache results.

Fig. 7. Cache frame

Loughran, N., Rashid, A.: Framed aspects: Supporting variability and configurability for AOP. In: Proceedings of 8th International Conference onSoftware Reuse, ICSR 2004. LCNS 3107, Springer, Madrid, Spain (2004)

Delineating Frame Boundaries

Careful consideration of:

VARIANTS ⟺ SCOPE FOR WHICH ASPECT IS INTENDED

- **1) Creating feature diagram using FODA - discovering variants**

  Feature Approach – natural design method for use with framed aspects

  Characteristics of feature aspect:

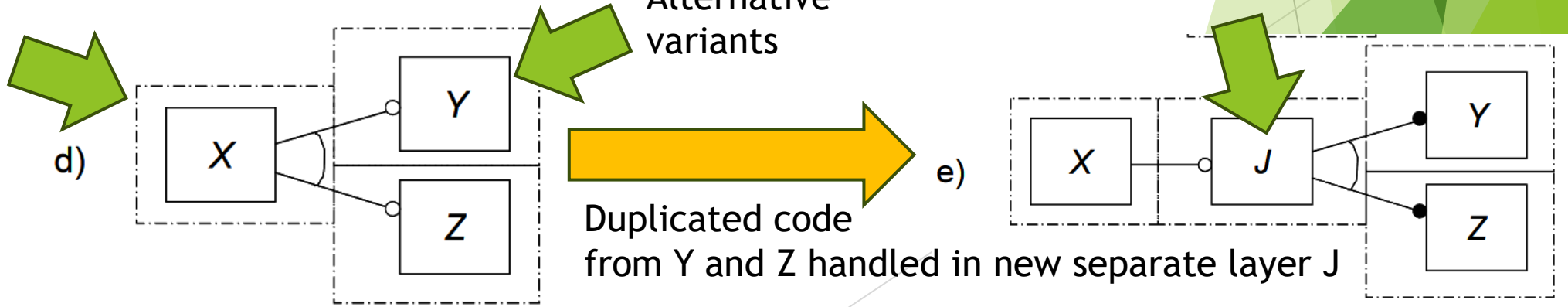
  - Dependencies,
  - Options
  - Alternative characteristics

- **2) Deducing aspects frames by delineating the boundaries between the different options and alternatives in the model**

Duplicated code common to Y and Z (such as an algorithm)

Frame boundaries

Alternative variants



Delineating Frame Boundaries

- **Enhancing modularity and reusability**
  - Allowing component to be framed separately from the main codebase

    REUSABLE IN OTHER CONTEXTS
- **Breaking down large aspect modules**
  - Hiding away less important information from the main concern

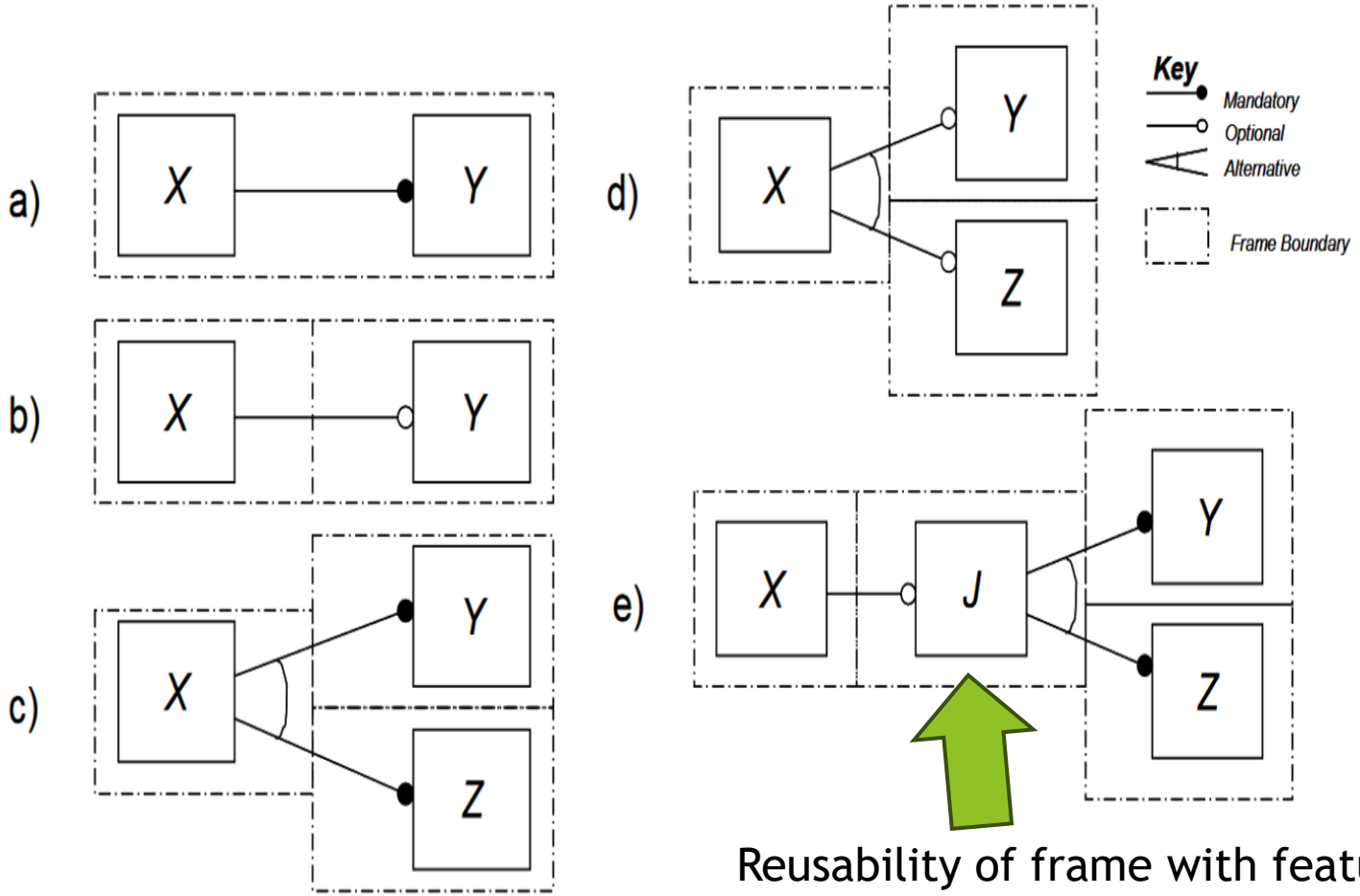
Frame technology ⟺ Aspect Oriented Programming

- **Utilization of frame commands to:**
  - **fine grained variability**
  - **parameterization**
  - **Constraints**

Any programming construct can be parameterized

- **Utilization of aspect-oriented programming to:**
  - **Integrating concern in a non-invasive manner**
  - **Used to make coarser grained functionality**
  - **Used when particular concerns crosscuts multiple modules**

Delineating Frame Boundaries



Reusability of frame with feature J

Fig. 3. Delineating frame boundaries of a) mandatory, b) optional and c) alternative features, and frame refactoring showing d) original and e) transformation.

Loughran, N., Rashid, A.: Framed aspects: Supporting variability and con-figurability for AOP.
In: Proceedings of 8th International Conference onSoftware Reuse, ICSR 2004.
LCNS 3107, Springer, Madrid, Spain (2004)

Delineating Frame Boundaries

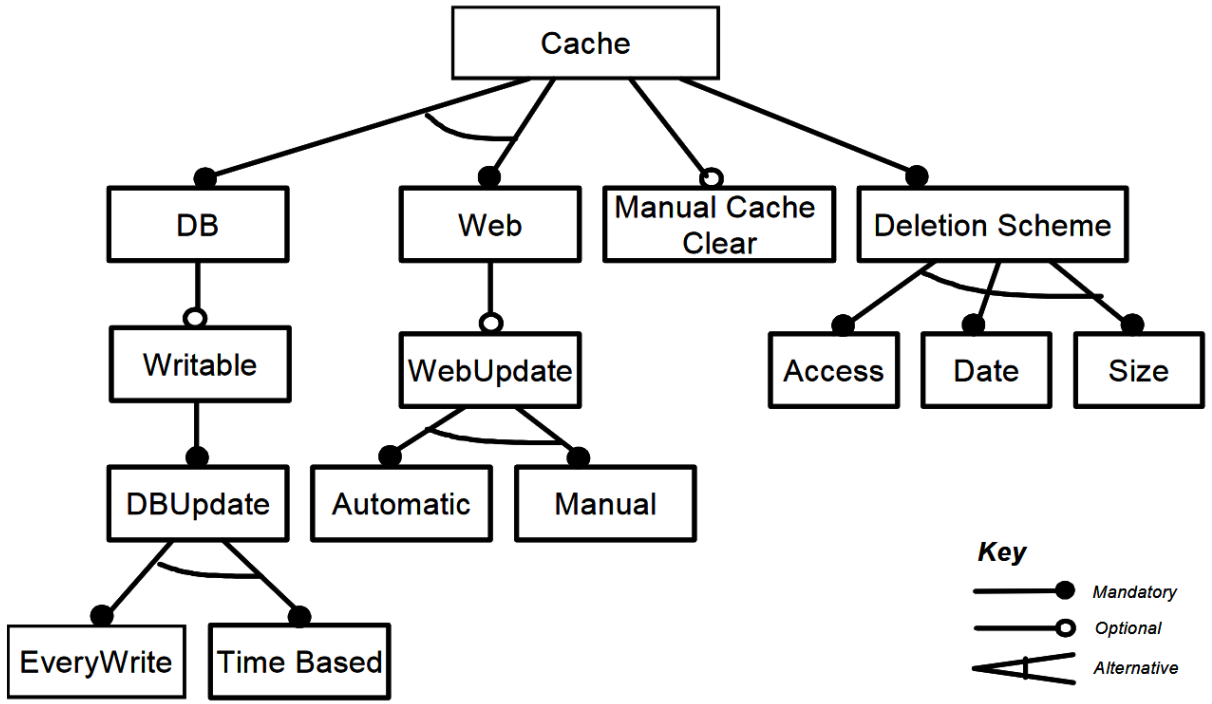


Fig. 5. Feature model for generic cache

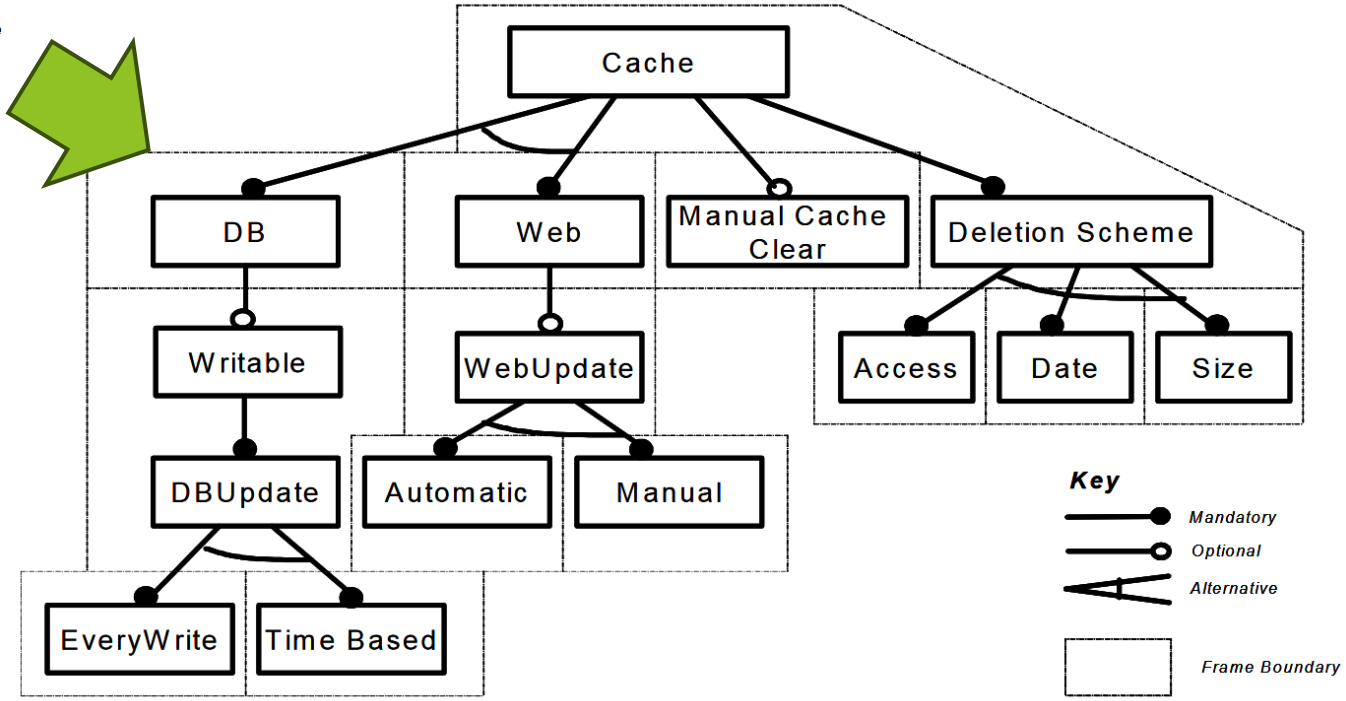


Fig. 6. Delineating frames in the generic cache

Loughran, N., Rashid, A.: Framed aspects: Supporting variability and configurability for AOP. In: Proceedings of 8th International Conference onSoftware Reuse, ICSR 2004. LCNS 3107, Springer, Madrid, Spain (2004)

Writable Frame

-demonstrating strength of framed-aspects over AOP alone and frame technology:

PARAMETERIZATION AND CROSSCUTTING REFINEMENTS ARE ENCAPSULATED WITHIN SINGLE FRAME:

```
<frame name = "WRITABLE">

pointcut DS_INSTANCE_PCT(<@CONN_CLASS> c) : cflow (this(c)) &&
 call(CacheDS.new(..)) ;

pointcut RESULTSET_PCT(<@CONN_CLASS> c) : this(c) &&
 call (ResultSet Statement.executeQuery(String));

after(<@CONN_CLASS> c) returning(CacheDS cds): DS_INSTANCE_PCT(c)
 {
 cds.setTables(c.getTables());
 }

after(<@CONN_CLASS> c) returning(ResultSet rs): RESULTSET_PCT (c)
 {
 try
      {
      ResultSetMetaData rsmd = rs.getMetaData();
     c.setTables(getTablesFromMetaData(rsmd));
      }
 catch(SQLException sqle) {}
 }
```

Pointcut used to trap new instances of CacheDS (data structure for holding the result data to be cached).

Pointcut to capture ResultSet from currently executing query.

Advice which adds tables contained within the executing query by a particular client to the CacheDS data structure

Advice which captures the ResultSet to obtain the ResultSetMetaData and, therefore, the tables used in the resulting query.

Loughran, N., Rashid, A.: Framed aspects: Supporting variability and configurability for AOP. In: Proceedings of 8th International Conference onSoftware Reuse, ICSR 2004. LCNS 3107, Springer, Madrid, Spain (2004)

```
private boolean CacheDS.isValid = true;
private Vector CacheDS.tables;
public void CacheDS.setTables(Vector v)
 {
 tables = v;
 }
public void CacheDS.containsTable(String s)
 {
 if(tables.contains(s)) isValid = false;
 }
public boolean CacheDS.isValid()
 {
 return isValid;
 }
public Vector CacheDS.getTables()
 {
 return tables;
 }
```

Introductions (intertype declaration) into the CacheDS data structure which

- **adds new fields:**
  - -boolean isValid
  - -Vector tables

  and
- **adds new methods:**
  - -void setTables(Vector v)
  - -boolean isValid()
  - -void containsTable(String s)
  - -Vector getTables()

Class provided as parameter

```
private Vector <@CONN_CLASS>.tables = new Vector();
public void <@CONN_CLASS>.setTables(Vector v)
 {
 tables=v;
 }
public Vector <@CONN_CLASS>.getTables()
 {
 return tables;
 }
// methods for getting table names from an SQL query and metadata
```

Class provided as parameter

Introductions (intertype declaration) into the current **CONN_CLASS** to store tables for the current executing query.

Fig. 8. Writable frame

Loughran, N., Rashid, A.: Framed aspects: Supporting variability and configurability for AOP.
In: Proceedings of 8th International Conference on Software Reuse, ICSR 2004.
LCNS 3107, Springer, Madrid, Spain (2004)

Specification Rules

-separated from the main aspect code

-adaptation of framed aspects with required functionality

```
<frame name = "CACHE SPEC">

<select option = "CACHE_TYPE" value = "DATABASE_CACHE" />
<set var = "MAX_CACHE_SIZE" value = "1000" />
<select option = "DELETION_SCHEME" value = "ACCESS" />
<set var = "PERC_TO_DEL" value = "50" />
<set var = "CONN_CLASS" value = "DBConnection" />
<set var = "SEND_QUERY" value = "sendQuery" />
<set var = "REPLY_CLIENT" value = "replyToClient" />
<set var = "DOC_TYPE" value = "String" />

<select option = "WRITABLE" value = "TRUE" />
<select option = "DB_UPDATE_SCHEME" value = "EVERYWRITE"

<adapt frame = "CACHE_RULES"/>

</frame>
```

FINALLY, APPLYING COMPOSITION RULES

1. The database cache option is selected for CACHE_TYPE, 1000 query resultsets can be stored by setting MAX_CACHE_SIZE, DELETION_SCHEME is set to the least accessed option, and PERC_TO_DEL is set to 50%.
2. CONN_CLASS targets a class called DBConnection, the methods for sending queries (sendQuery) to the database and sending the query results back to the client (replyToClient) are bound to SEND_QUERY and REPLY_CLIENT respectively, while the type of data to be stored in the cache, DOC_TYPE, is bound to String.
3. The WRITABLE option is selected and the EVERYWRITE update scheme is chosen.
4. **Specification is processed** by the **composition rules** defined for the cache component **to bind the components together.**

Fig. 9. Typical specification frame for a database

Loughran, N., Rashid, A.: Framed aspects: Supporting variability and con-figurability for AOP. In: Proceedings of 8th International Conference onSoftware Reuse, ICSR 2004. LCNS 3107, Springer, Madrid, Spain (2004)

Composition Rules

Applied according to specification

-separated from the main aspect code → -creation of different rules
-reusing framed aspects in different contexts

- 1. Constraining meta variables to sets or ranges of possible values.
- 2. Adapting mandatory features as defined by the specification.
- 3. Adapting optional features if selected.
- 4. Adaptation rules for the database cache.

```
<frame name = CACHE_RULES>

<constrain var = "CACHE_TYPE" toSet = "DATABASE_CACHE,WEB_CACHE"/>
<constrain var = "PERC_TO_DEL" toBoundary = "25,100"/>
<constrain var = "DELETION_SCHEME" toSet = "ACCESS,DATE,SIZE"/>
<constrain var = "DB_UPDATE_SCHEME" toSet = "EVERYWRITE,TIME_BASED"/>
<constrain var = "WEB_UPDATE_SCHEME" toSet = "AUTOMATIC,MANUAL"/>

<adapt frame = "CACHE_TYPE"/>
<adapt frame = "DELETION_SCHEME"/>

<option name = "MANUAL_CACHE_CLEAR" value = "TRUE">
 <adapt frame = "MANUAL_CACHE_CLEAR" />
</option>

<option name = "CACHE_TYPE" value = "DATABASE_CACHE">
 <option name = "WRITABLE" value = "TRUE">
        <adapt frame = "DB_UPDATE_SCHEME"/>
 </option>
</option>
```

Loughran, N., Rashid, A.: Framed aspects: Supporting variability and configurability for AOP. In: Proceedings of 8th International Conference onSoftware Reuse, ICSR 2004. LCNS 3107, Springer, Madrid, Spain (2004)

Fig. 10. Composition rules for generic cache

Checking Constraints: Example

Specification

```
<frame name = "CACHE SPEC">

<select option = "CACHE_TYPE" value = "DATABASE_CACHE" />
<set var = "MAX_CACHE_SIZE" value = "1000" />
<select option = "DELETION_SCHEME" value = "ACCESS" />
<set var = "PERC_TO_DEL" value = "50" />
<set var = "CONN_CLASS" value = "DBConnection" />
<set var = "SEND_QUERY" value = "sendQuery" />
<set var = "REPLY_CLIENT" value = "replyToClient" />
<set var = "DOC_TYPE" value = "String" />

<select option = "WRITABLE" value = "TRUE" />
<select option = "DB_UPDATE_SCHEME" value = "EVERYWRITE" />

<adapt frame = "CACHE_RULES"/>

</frame>
```

④

Fig. 9. Typical specification frame for a database

Composition Rules

DATABASE_CACHE in [DATABASE_CACHE, WEB_CACHE]

Is constraint

Adapted as mandatory (always)

```
<frame name = CACHE_RULES>

<constrain var = "CACHE_TYPE" toSet = "DATABASE_CACHE,WEB_CACHE"/>
<constrain var = "PERC_TO_DEL" toBoundary = "25,100"/>
<constrain var = "DELETION_SCHEME" toSet = "ACCESS,DATE,SIZE"/>
<constrain var = "DB_UPDATE_SCHEME" toSet = "EVERYWRITE,TIME_BASED"/>
<constrain var = "WEB_UPDATE_SCHEME" toSet = "AUTOMATIC,MANUAL"/>

<adapt frame = "CACHE_TYPE"/>
<adapt frame = "DELETION_SCHEME"/>

<option name = "MANUAL_CACHE_CLEAR" value = "TRUE">
 <adapt frame = "MANUAL_CACHE_CLEAR" />
</option>

<option name = "CACHE_TYPE" value = "DATABASE_CACHE">
 <option name = "WRITABLE" value = "TRUE">
       <adapt frame = "DB_UPDATE_SCHEME"/>
 </option>
</option>
```

① ② ③ ④

Fig. 10. Composition rules for generic cache

Modeling Variability - Types

- A) Modeling variability using parameterization
- B) Modeling variability using information hiding
- C) Modeling variability using inheritance
- D) Modeling variability using variation points

Source: Diana L. Webber, Hassan Gomaa, Modeling variability in software product lines with the variation point model,
Science of Computer Programming, Volume 53, Issue 3, 2004, Pages 305-331,
ISSN 0167-6423, https://doi.org/10.1016/j.scico.2003.04.004

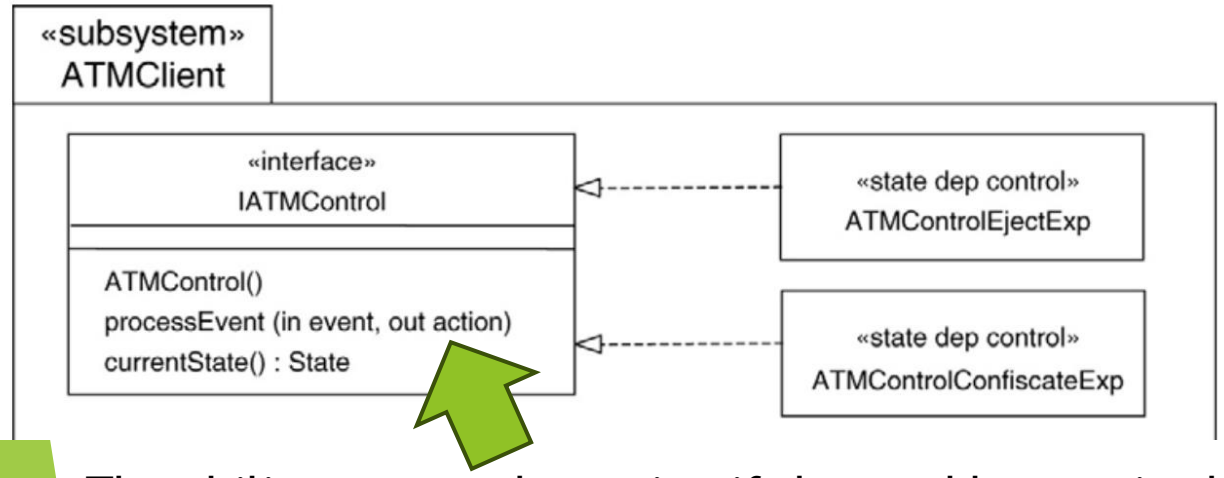
VARIATION POINT

A variation point identifies one or more locations at which the variation will occur

Source: I. Jacobson, M. Griss, P. Jonsson, Software Reuse-Architecture, Process and Organization for Business Success, ACM Press, New York, NY, 1997

Modeling Variability Using Parameterization

-with the Unified Modeling Language (UML) notation



The ability to vary the action if the card has expired

The ability to vary a greeting for display.

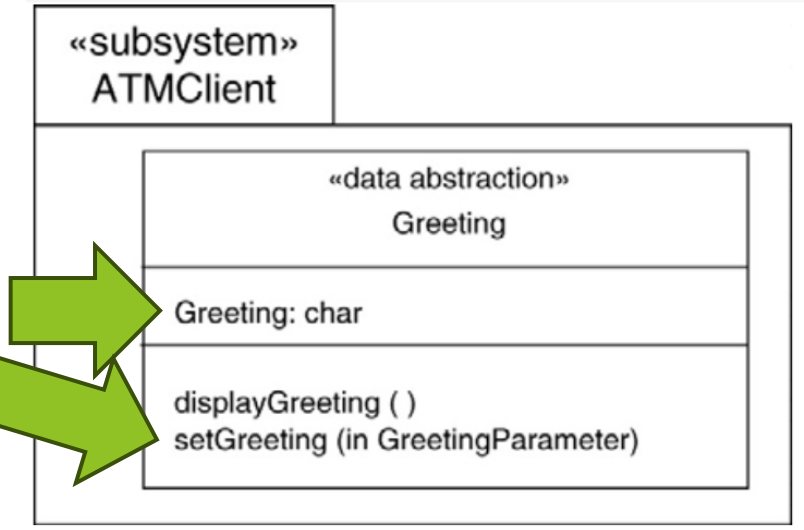


Fig. 1. Greeting—parameterization.

Source: Diana L. Webber, Hassan Gomaa, Modeling variability in software product lines with the variation point model, Science of Computer Programming, Volume 53, Issue 3, 2004, Pages 305-331, ISSN 0167-6423

The ability to vary the language of choice for display.

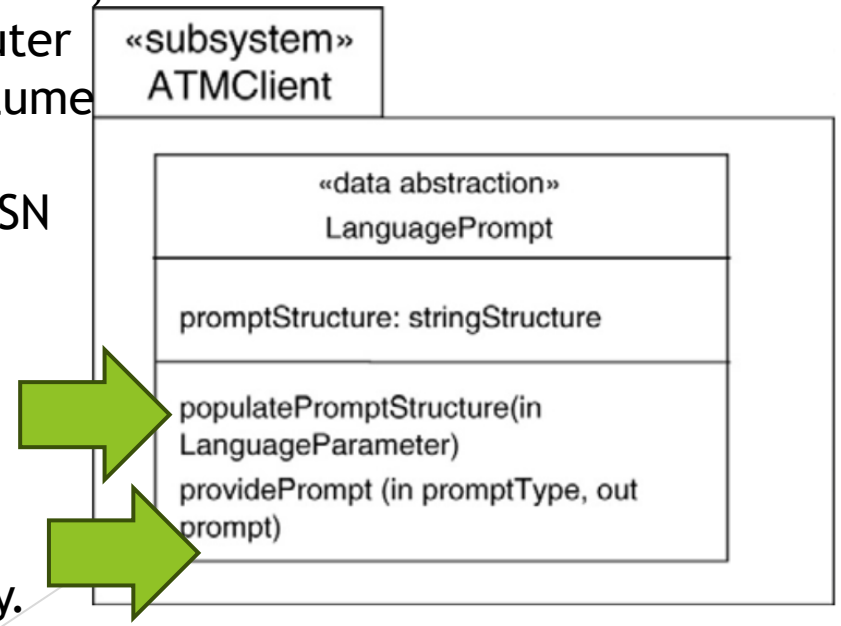


Fig. 2. Language—parameterization.

Modeling variability using information hiding

LIMITED TO SELECT FROM THE SET OF CHOICES

-several version of the same component with the similar interface

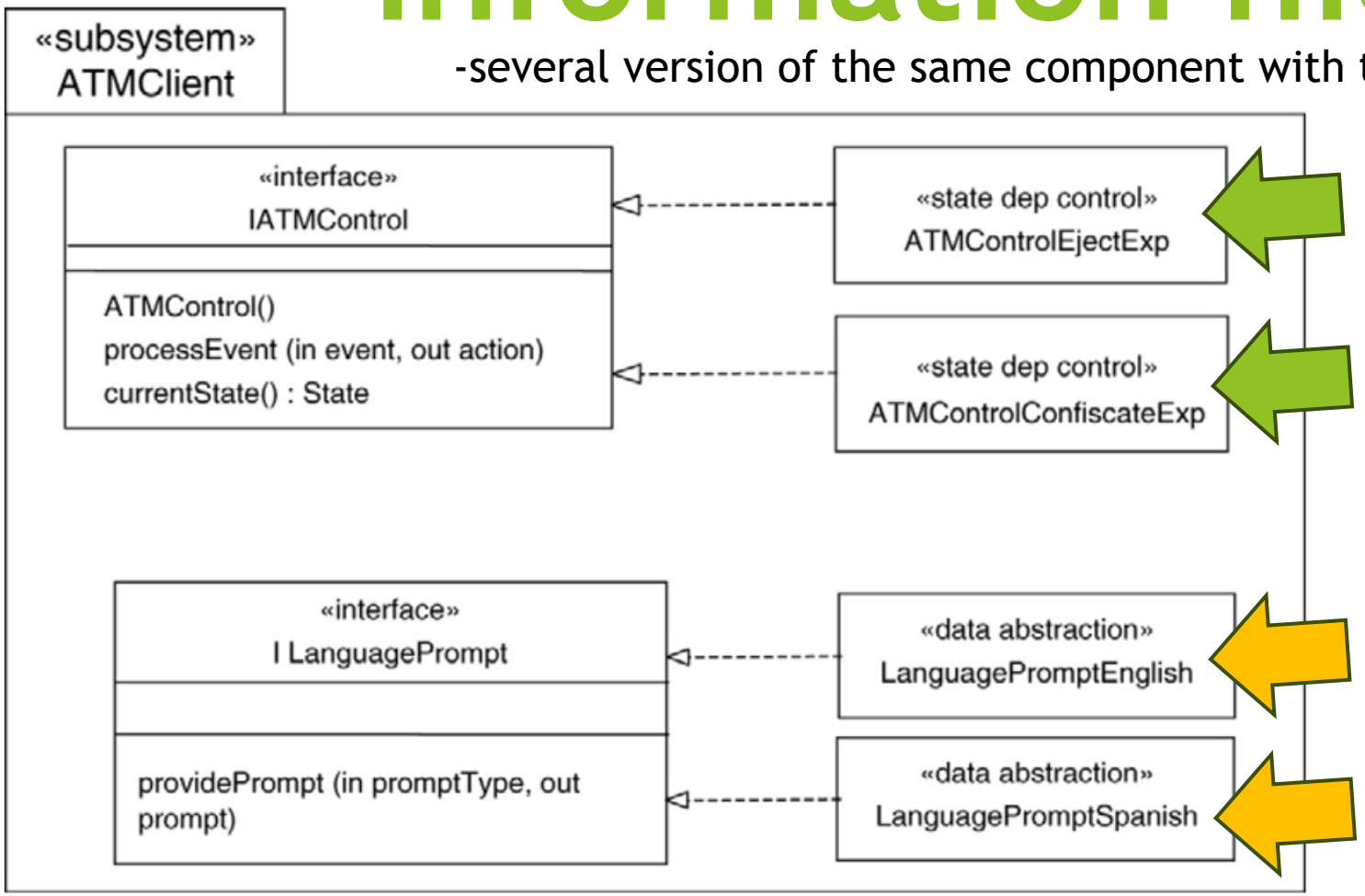


Fig. 4. ExpiredCard—information hiding.

-hiding variability inside each version of the component
VARIANTS → different versions of the same component

-limited to changes inside components, not interfaces

-concerning component version only

-no need to develop new variants

Source: Diana L. Webber, Hassan Gomaa, Modeling variability in software product lines with the variation point model, Science of Computer Programming, Volume 53, Issue 3, 2004, Pages 305-331, ISSN 0167-6423

Modeling Variability Using Inheritance

Source: Diana L. Webber, Hassan Gomaa, Modeling variability in software product lines with the variation point model, Science of Computer Programming, Volume 53, Issue 3, 2004, Pages 305-331, ISSN 0167-6423

-variants do not have to adhere to the same interfaces

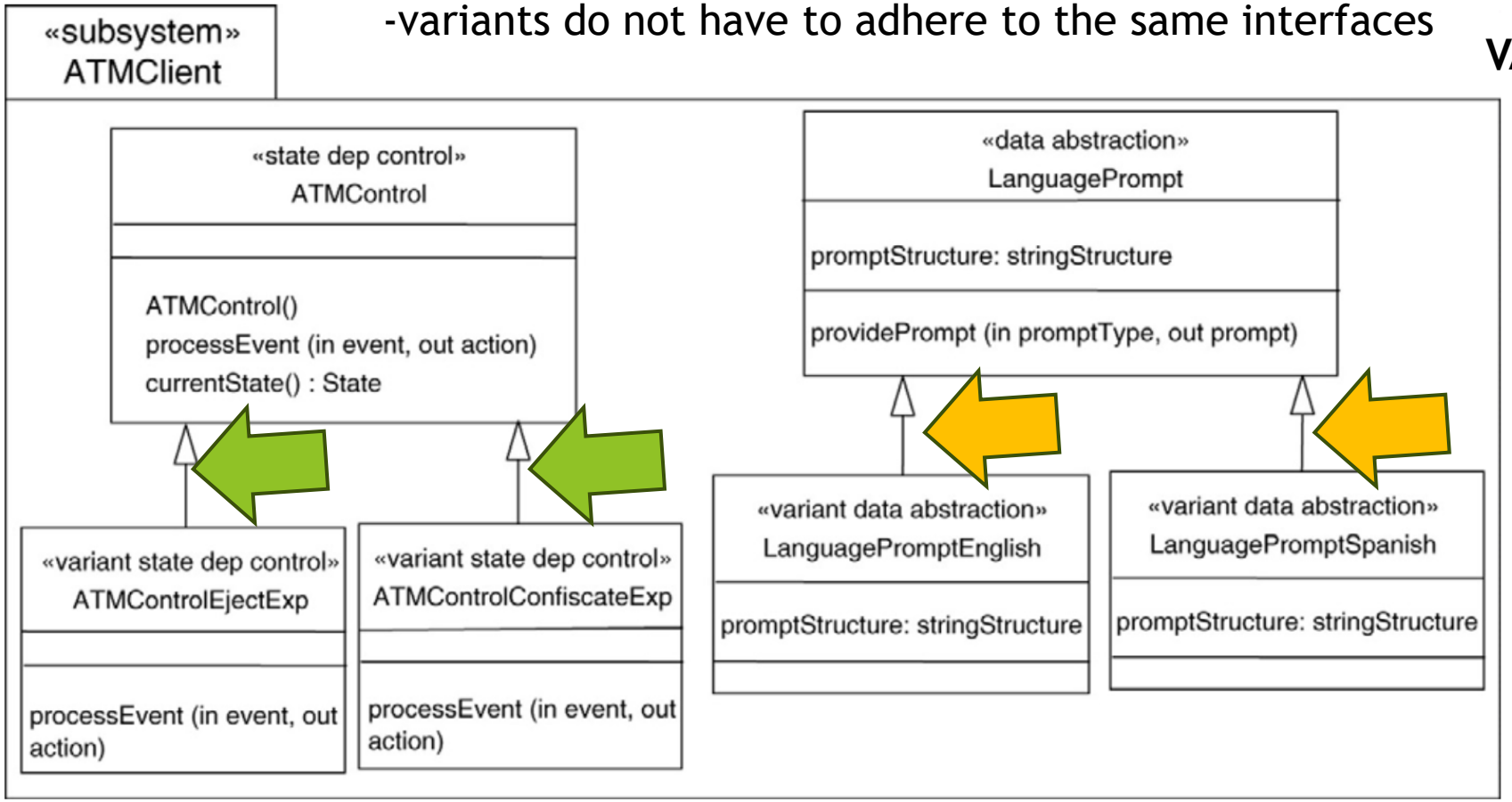


Fig. 6. ExpiredCard—inheritance.

VARIANTS → specializations of other components

-subclass extends the interfaces or superclass with provided new methods and attributes
+ overriding methods

LIMITED TO SELECT FROM THE SET OF CHOICES

-no need to develop new variants

Example: KobrA Approach from PULSE

Modeling Variability Using Variation Points

Source: Diana L. Webber, Hassan Gomaa, Modeling variability in software product lines with the variation point model, Science of Computer Programming, Volume 53, Issue 3, 2004, Pages 305-331, ISSN 0167-6423

-core asset component consists of variation points

USED TO BUILD TARGET SYSTEM COMPONENTS FROM VARIANTS MADE OUT OF THESE VARIATION POINTS

-the most of flexibility: *making unique variants and maintaining them*

PROS

-less resources to develop core assets/common core

-reuser can create new variant not supplied with common core

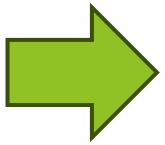
CONST

-requires additional resources to develop the variants as part of core assets

-lack of reusability for reuser to create his own variant

Communicating reuse through following views on variation points:

- 1) Requirements View
- 2) Component Variation Point View
- 3) Static Variation Point View
- 4) Dynamic Variation Point View

NEW VARIANT  CORE ASSETS

HAS TO BE ADDED INTO

-maintenance and management costs

Lightweight method for software product line feature management

Lightweight nature

- Independent of the given programming language
- No assumption about the development process or management is made
- No need for specific DSL and other tools or plugins (but lacks traceability)
- Managed by developers on their own, inside code specifically by annotating variable parts
- Easy to comprehend and use
- Only 3 associated actions given directly by annotation type – should be enough (+ another analytic versions and one recursive version can be perceived)
- Expressions are not only conditional rules but domain knowledge should be inserted
- Should be used in a native and modular way
- The semantics of rules and derivation mechanism can be directly modified by developers according to their needs/observations

Configuration expressions

- Express hierarchy information
- Easy to process by other systems
- Known format
- Not restricted to given parser/given annotation
- Addition information (non-configurational) can be included
- Possibilities of IDE formatting:
  - Hide certain hierarchy levels
  - Hide whole variability information
  - Emphasize on certain:
    - Information
    - Variability relation

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

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

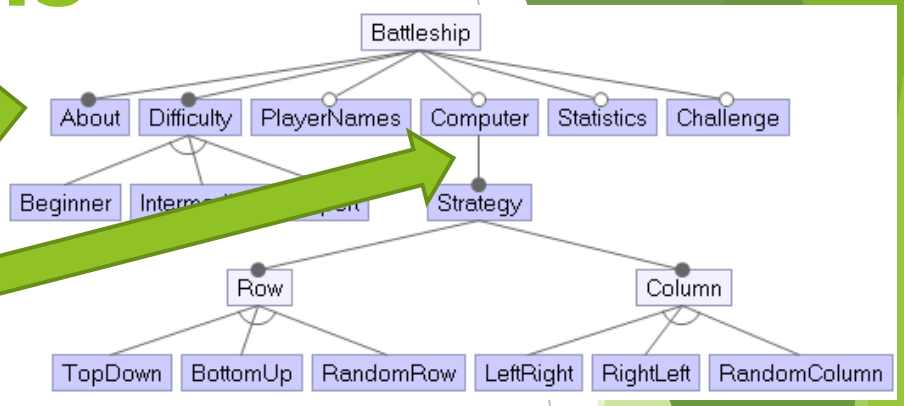
4. If whole is true, then we can copy annotated method

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

Applied annotations types

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

//@{}

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

//@{}

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

//#{}

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

//%{}

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

Variables features can interfere

Setting names for players needs update when computer player is added

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

    Player createdPlayer = proceed();
    createdPlayer.setName(playerNameLine);

    System.out.println(createdPlayer.getName());

    return createdPlayer;
}
```

Variable feature for setting player names

We can't use //@ annotation, because of many different variable features

Method will be included only if all conditions are met (for specific feature)

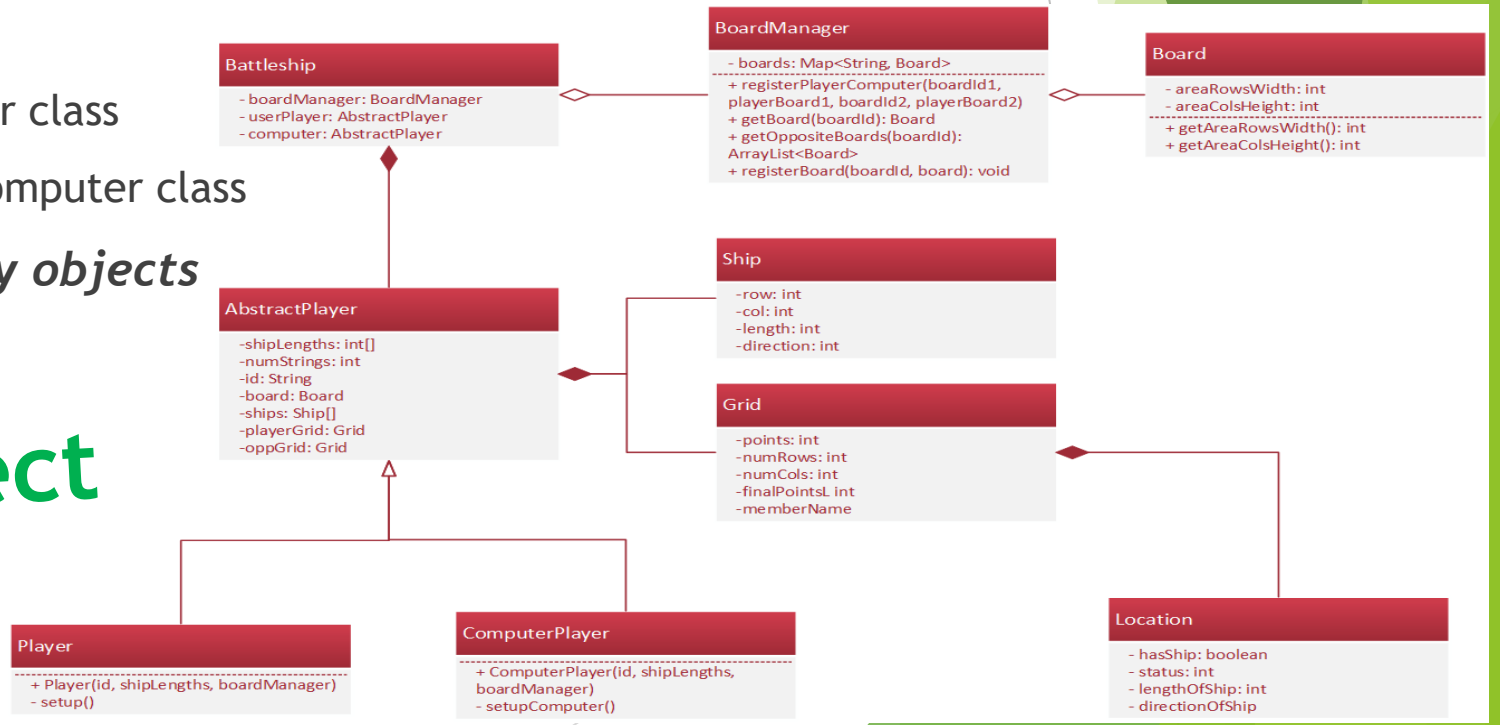
Additional variable feature for managing also computer name

```
//#{"playerNames": "true", "computerOpponent": "true"}
ComputerPlayer around(): call(ComputerPlayer.new(..)) && if(Configuration.playerNames){
    Scanner reader = InputReader.getReader();
    System.out.println("Set computer name:");
    String playerNameLine = reader.nextLine().replace("\n", "");

    ComputerPlayer createdComputerPlayer = proceed();
    createdComputerPlayer.setName(playerNameLine);

    System.out.println(createdComputerPlayer.getName());

    return createdComputerPlayer;
}
```

Object oriented redesign of Battleship game

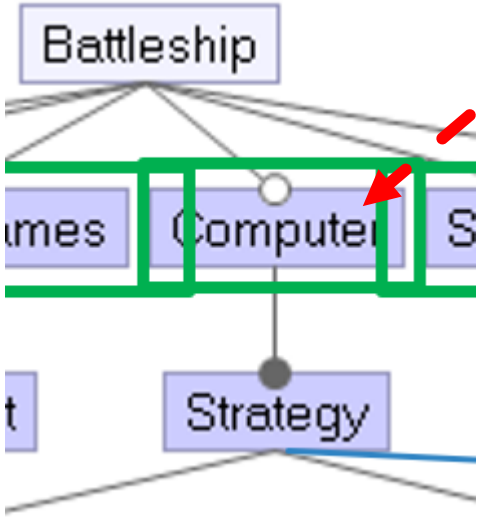
- ***Hardcoded parts should be changed to support configurability***
  - Different lengths of board
  - Support for adding player
- ***Concerns should be separated***
  - Setup of player should be part of player class
  - Setup of computer should be part of computer class
- ***Static methods should be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

Performing refactoring of project

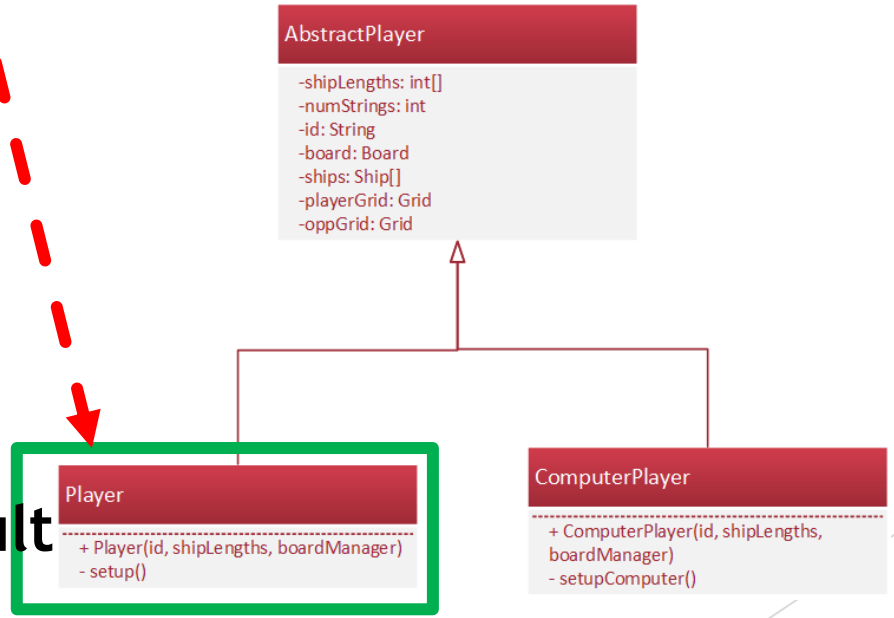


Pattern Cuckoo's egg

```
AbstractPlayer around(String opponentID, int[] playerShips, BoardManager boardManager):
    instantiateComputerInCaseOfPlayer(opponentID, playerShips, boardManager) {
    if (Configuration.computerOpponent) {
        System.out.println("Creating computer ---------- !");
        return new ComputerPlayer("COMPUTER", playerShips, boardManager);
    }

    return proceed(opponentID, playerShips, boardManager);
}
```

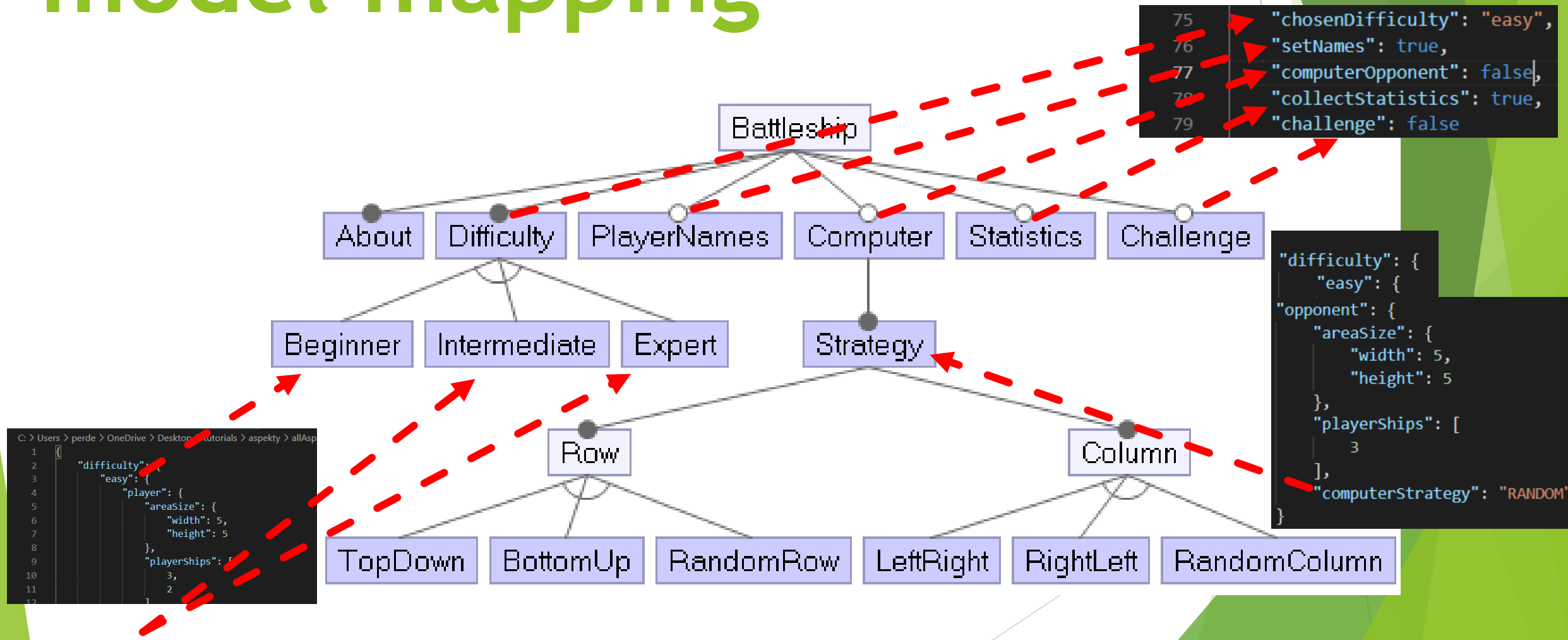


Created by default



Created if condition is met

Config to feature model mapping



Configuration using JSON File

C: > Users > perde > OneDrive > Desktop > tutorials > aspekty > allAsp

```
{
    "difficulty": {
        "easy": {
            "player": {
                "areaSize": {
                    "width": 5,
                    "height": 5
                },
                "playerShips": [
                    3,
                    2
                ]
            },
            "opponent": {
                "areaSize": {
                    "width": 5,
                    "height": 5
                },
                "playerShips": [
                    3
                ],
                "computerStrategy": "RANDOM"
            }
        },
        "hard": {
            "player": {
                "areaSize": {
                    "width": 6,
                    "height": 6
                },
                "playerShips": [
                    3,
                    2,
                    3
                ]
            },
            "opponent": {
                "areaSize": {
                    "width": 10,
                    "height": 10
                },
                "playerShips": [
                    3,
                    2,
                    5,
                    3
                ],
                "computerStrategy": "RANDOM"
            }
        }
    },
    "chosenDifficulty": "easy",
    "setNames": true,
    "computerOpponent": false,
    "collectStatistics": true,
    "challenge": false
}
```

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)

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

//@{}

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

//#{}

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

//%{}

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

Difficulty configuration

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

1. PREPARATION

```
5  public aspect PlayersPrecedence {
6      declare precedence: DifficultyManagement, ComputerInstantiator;
7  }
```

2. POINTCUTS

The same pointcuts
(with other names)

"Hook" functions

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

```
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);
```

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

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)

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

MISS = MOVES - HITS

```
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;
```

```
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

Variable encapsulation problem

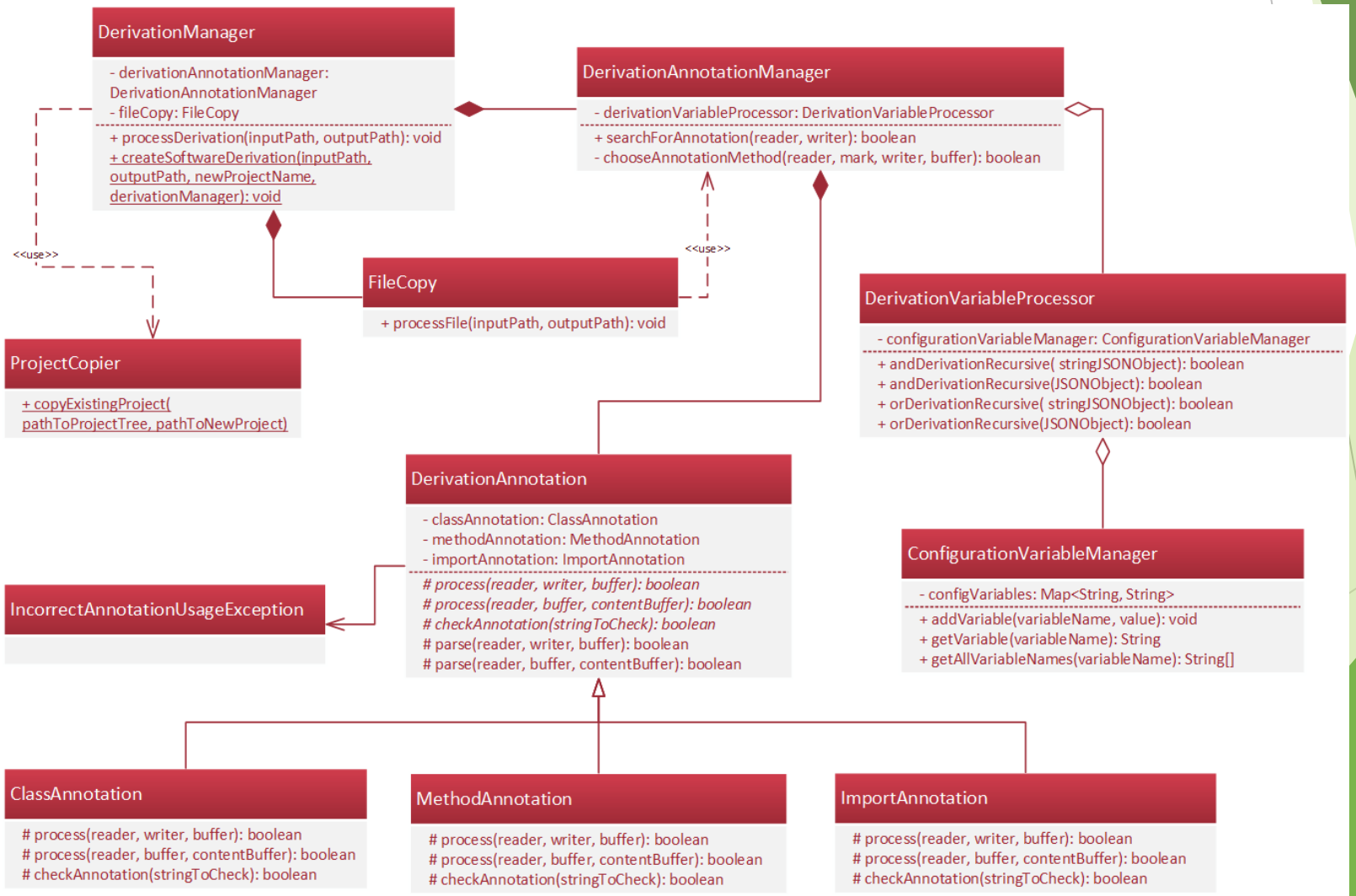
In player instance chooser aspect:

```
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;
```

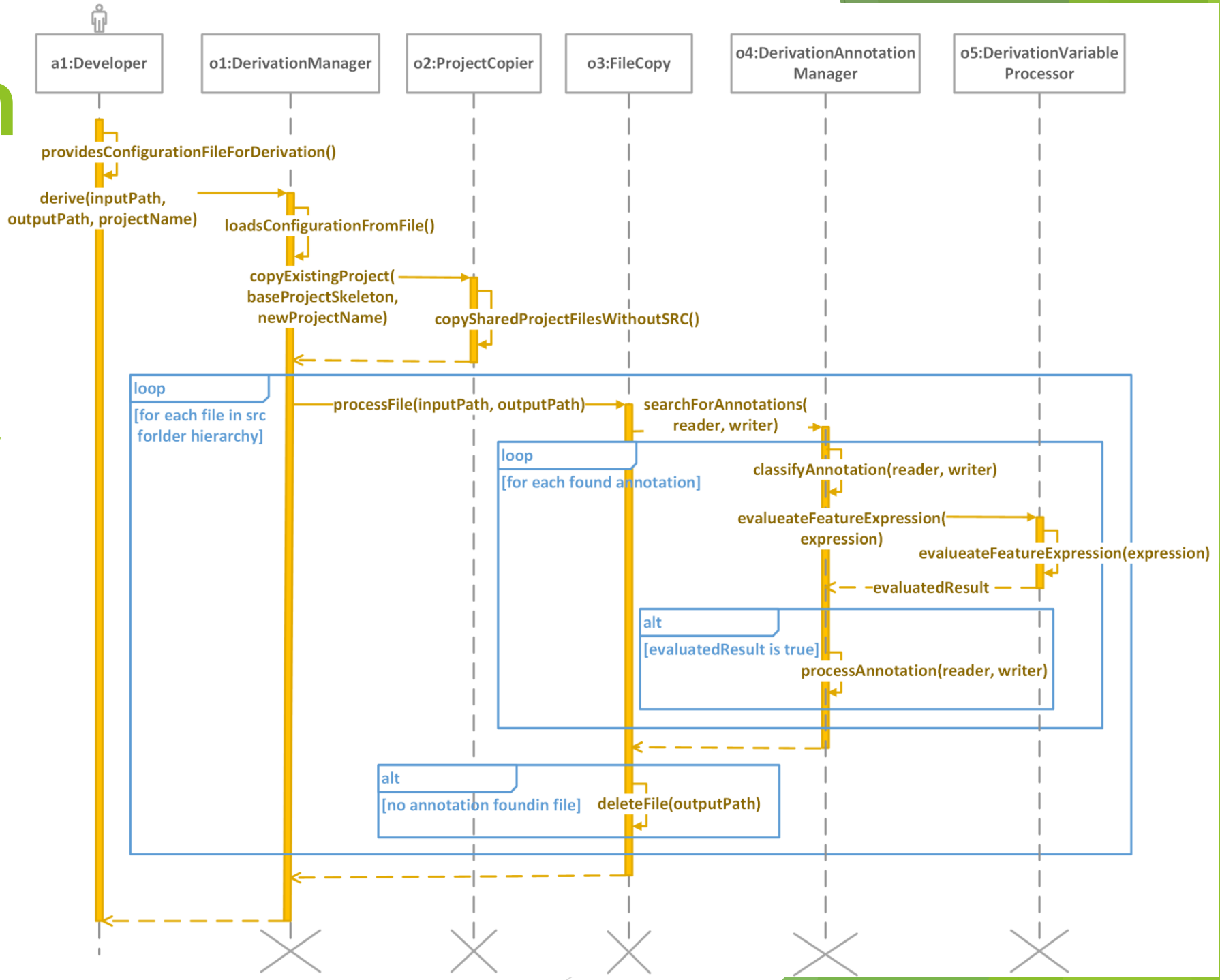
The same problem

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

Derivator - Class Diagram



Evaluation of in-code variability



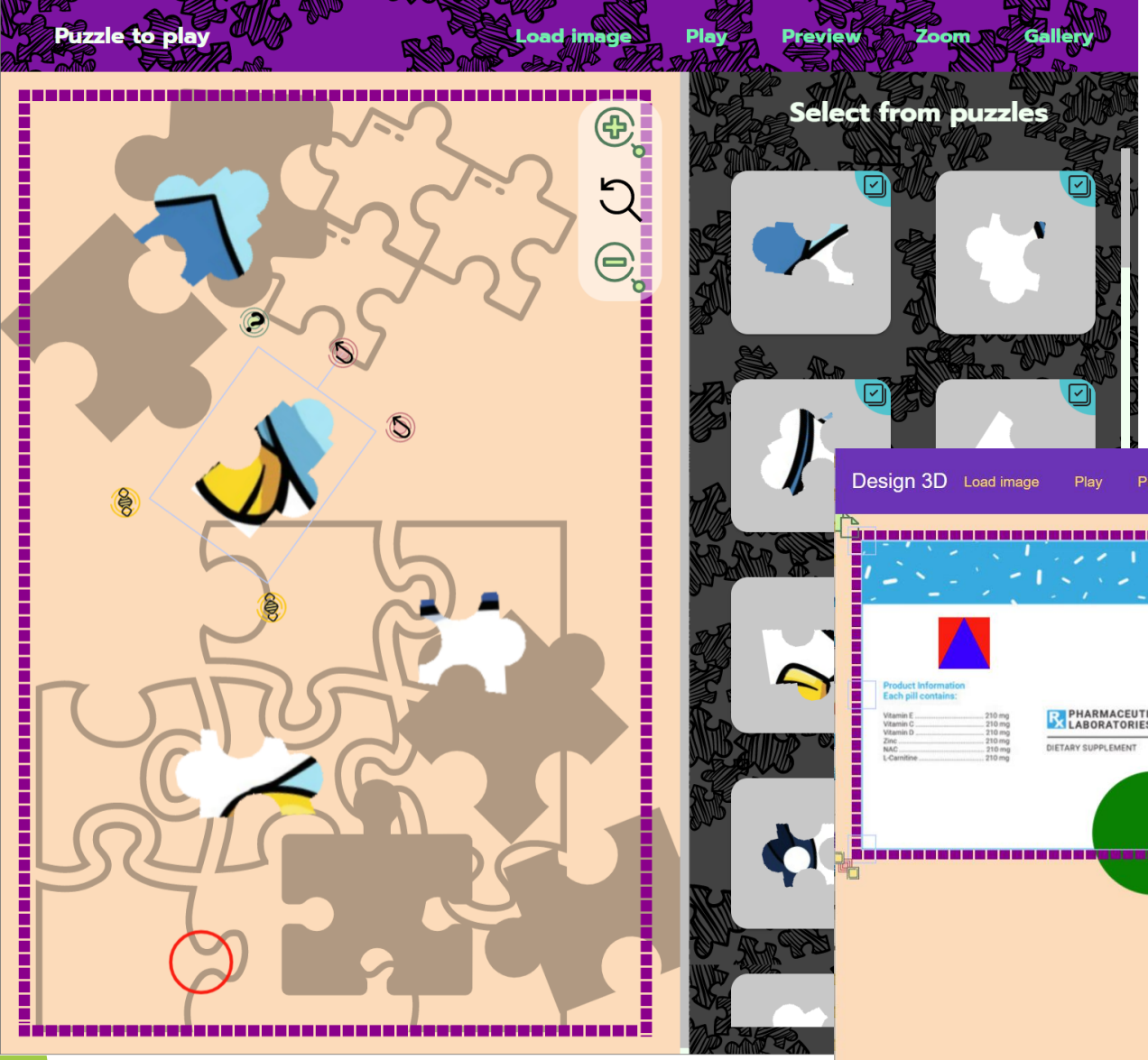
Motivation: Studying the complexity of in-code variability

Measure code complexity of ***…to handle variability***

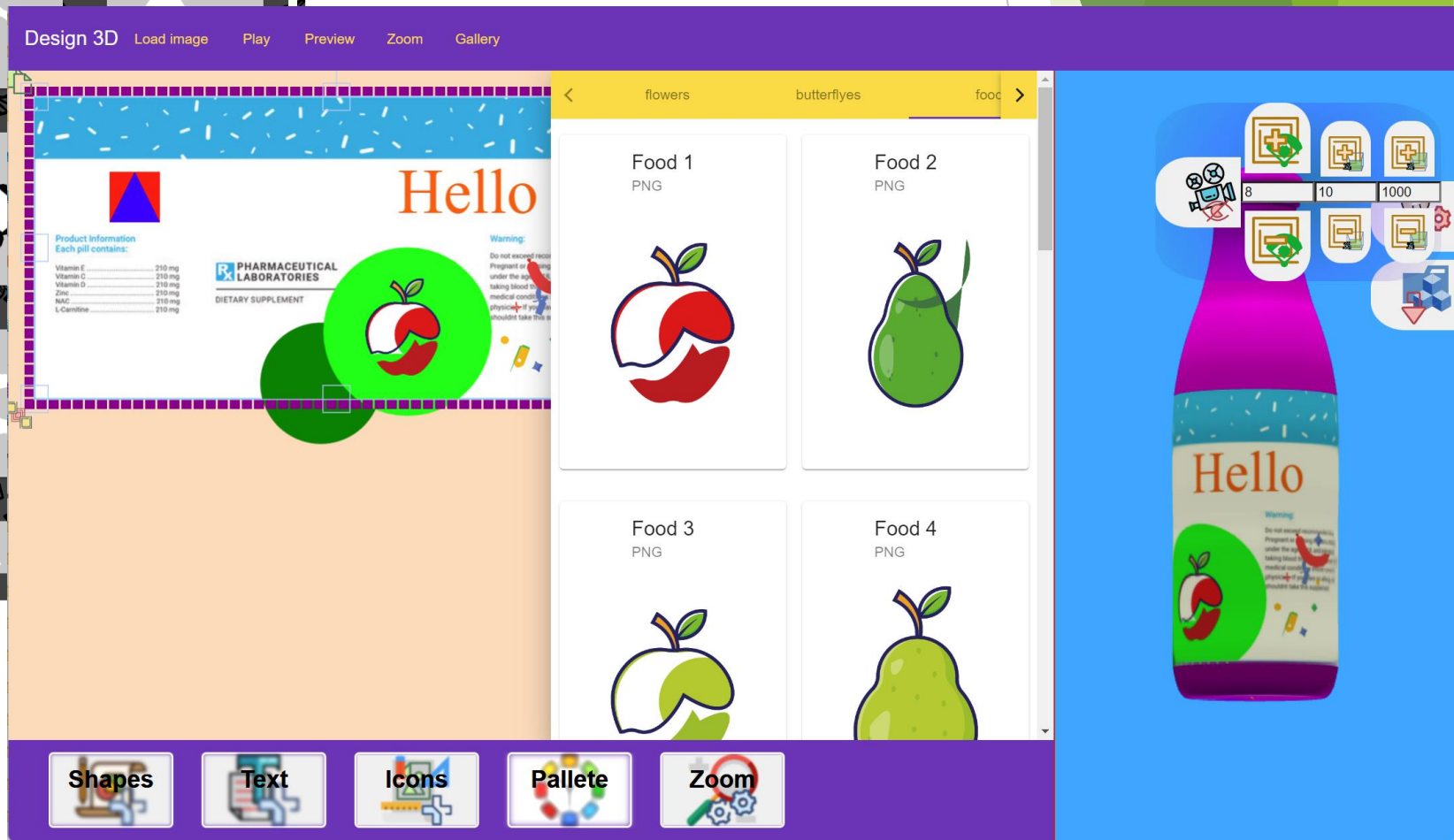
- Code constructs of variability management to handle variability
  - TO FIND LESS COMPLEX CODE CONSTRUCTS
  - TO EVALUATE INCODE EQUIVALENTS OF OUR LIGHTWEIGH METHOD CONSTRUCTS
  - TO DESIGN FILTERING OF VARIABILITY DEPENDENT CONTEXT
  - TO JUSTIFY THE SUPPORT OF LESS COMPLEX VARIABILITY CONSTRUCTS
- Entire variability management
  - TO MEASURE THE INFLUENCE OF CODE COMPLEXITY MEASURES
- Expressions used by variability management to mark variability
  - TO OPTIMIZE THEM
  - TO MAKE THEM MORE COMPREHENSIBLE WHILE PRESERVING FEATURE MODELS IN CODE

Measuring in-code complexity of puzzle to play SPL

Wilcoxon pair test



Commonality vs. Variability



Puzzle app. vs. Desing app.

Cases to evaluate in-code complexity

Case 1: Variability is expressed using detachable decorators

Case 2: Variability is expressed using detachable decorators, but without variability configuration expressions

Case 3: Variability is expressed using wrappers

Case 4: Variability is not expressed at all

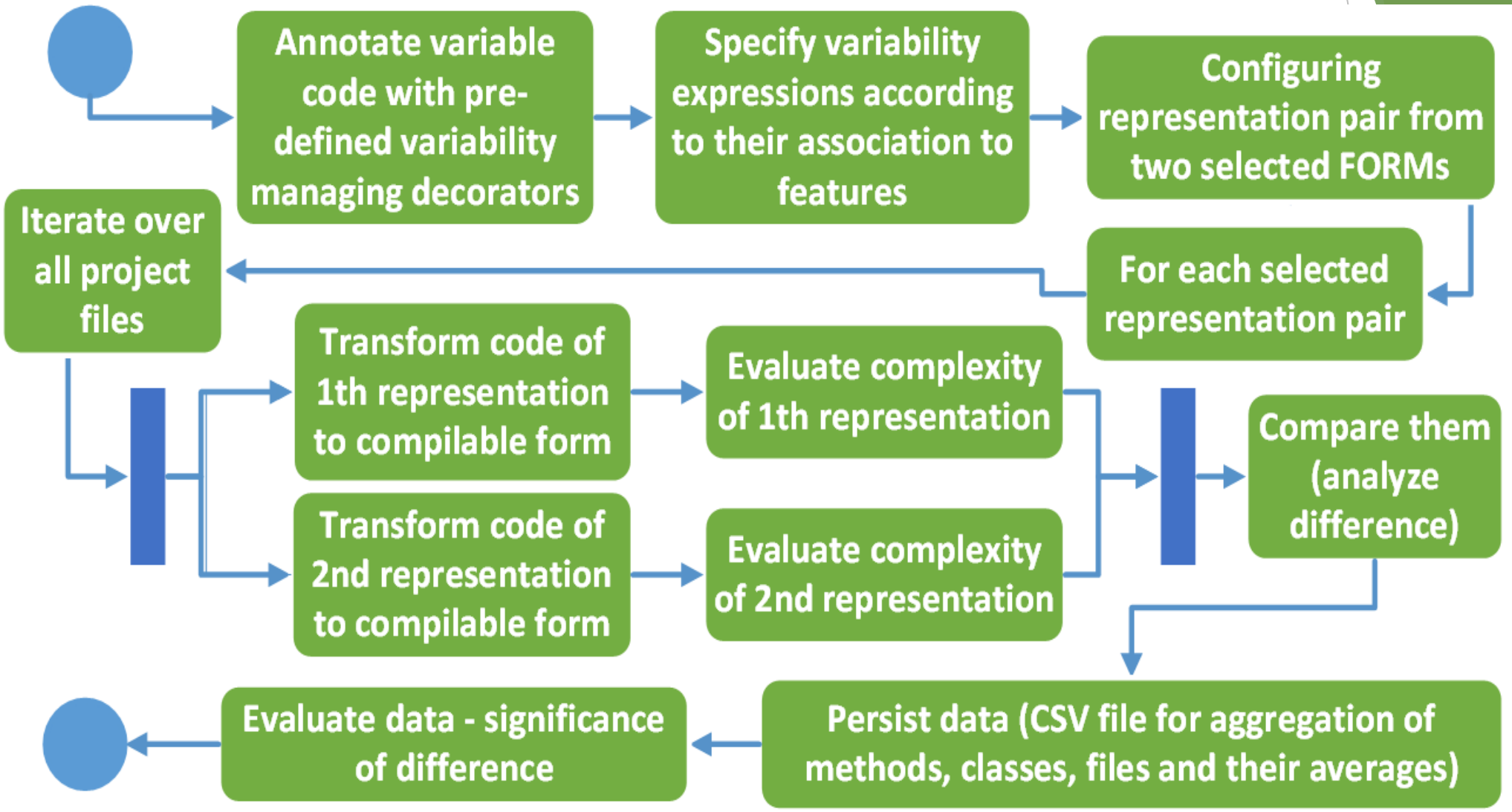
Case 5: Variability is expressed using detachable decorators, but additional unwanted dead code constructs are not included for illegal decorators

```
// @ts-ignore
@DecoratorTypesService.skipLineVariableDeclaration(
        {"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}, "[NOT=let zoomConfig = null;]")
let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom",
        "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};
```

VS

```
EXPRESSION_START50 = {"OR": { "zoomCoordinates": "true", "zoomValue": "true" }};
let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom",
    "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};
ELSE50 = { "ELSE": "~~~~~~~~~~~~~~~~~~~~~~~~" };
let zoomConfig = null;
EXPRESSION_END50 = { "EXPRESSION_END": "------------------------" };
```

Evaluation process



How complex configuration expression are?

Hypothesis 1:

Variability expressions extracted from annotations do not significantly change the complexities of most evaluated metrics.

Configuration expressions

- Express hierarchy information
- Easy to process by other systems
- Known format
- Not restricted to given parser/given annotation
- Addition information (non-configurational) can be included
- Possibilities of IDE formatting:
  - Hide certain hierarchy levels
  - Hide whole variability information
  - Emphasize on certain:
    - Information
    - Variability relation

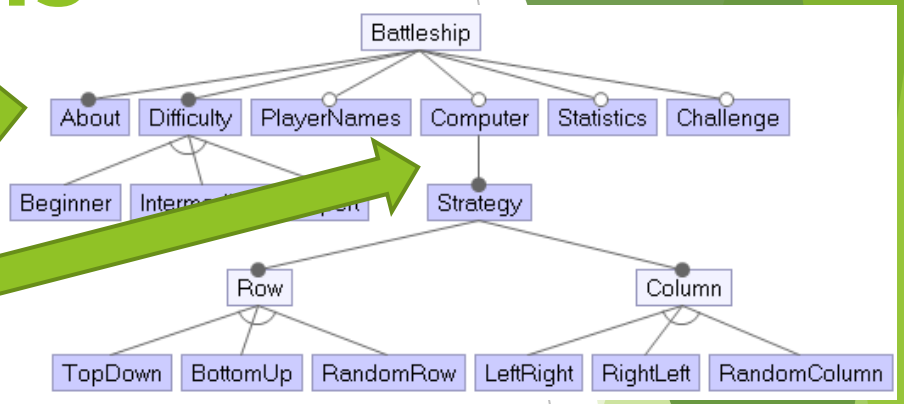
```
{
"AND": {
    "OR": {
        "variable1": "false",
        "AND": {
            "variable2": "true",
            "variable3": "true"
        }
    },
    "variable4": "true"
}
```

Hierarchic nature of configuration expressions

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

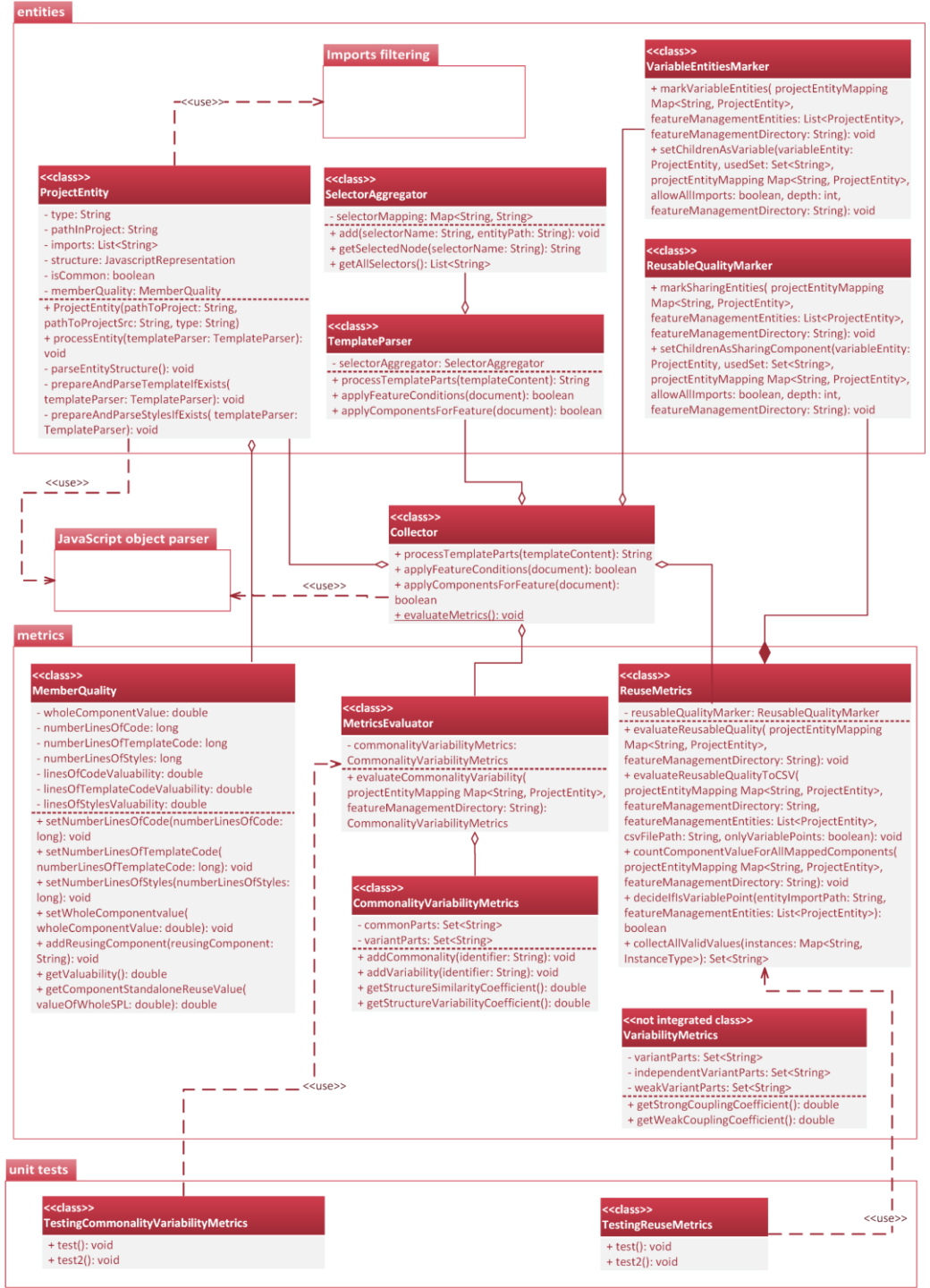
Configuration of the first later

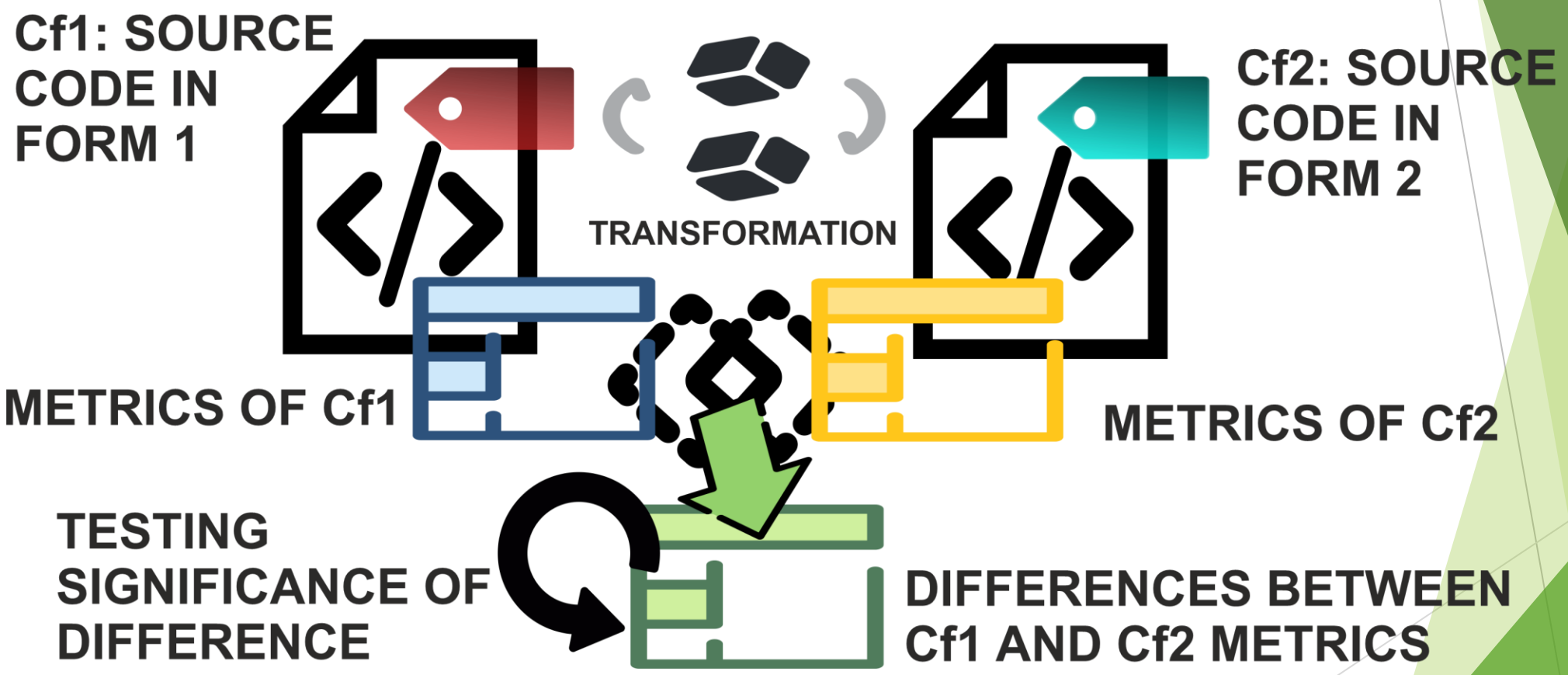
Configuration related to computer as player



Focus during their creation can be on:

- hierarchy levels
- feature groups
- certain hierarchies


entities
Imports filtering
<<use>>
<<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
<<use>>
JavaScript object parser
<<class>>
Collector
+ processTemplateParts(templateContent): String
+ applyFeatureConditions(document): boolean
+ applyComponentsForFeature(document): boolean
+ evaluateMetrics(): void
<<use>>
metrics
<<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
<<use>>
unit tests
<<class>>
TestingCommonalityVariabilityMetrics
+ test(): void
+ test2(): void
<<class>>
TestingReuseMetrics
+ test(): void
+ test2(): void
<<use>>


Cf1: SOURCE CODE IN FORM 1
TRANSFORMATION
Cf2: SOURCE CODE IN FORM 2
METRICS OF Cf1
METRICS OF Cf2
TESTING SIGNIFICANCE OF DIFFERENCE
DIFFERENCES BETWEEN Cf1 AND Cf2 METRICS

Alternative statement (IF-ELSE)

Negative variability

	Initial (compilable) transformation – The first version [base version]	Final (compilable) transformation – The second version [compared version]
SCENARIO 1: *FORM 1 – FORM 2*	// @ts-ignore @DecoratorTypesService.skipLineVariableDeclaration({"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}, "[NOT=let zoomConfig = null;]") let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};	// @ts-ignore @DecoratorTypesService.skipLineVariableDeclaration("[NOT=let zoomConfig = null;]") let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};
SCENARIO 2: *FORM 1 – FORM 4*	// @ts-ignore @DecoratorTypesService.skipLineVariableDeclaration({"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}, "[NOT=let zoomConfig = null;]") let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};	let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};
SCENARIO 3: *FORM 1 – FORM 5*	// @ts-ignore @DecoratorTypesService.skipLineVariableDeclaration({"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}, "[NOT=let zoomConfig = null;]") let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};	// @ts-ignore @DecoratorTypesService.skipLineVariableDeclaration({"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}, "[NOT=let zoomConfig = null;]") let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};
SCENARIO 4: *FORM 3 – FORM 1*	EXPRESSION_START50 = {"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}; let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent}; ELSE50 = { "ELSE": "~~~~~~~~~~~~~~~~~~~~~~~~~~" }; let zoomConfig = null; EXPRESSION_END50 = { "EXPRESSION_END": "--------------------------" };	// @ts-ignore @DecoratorTypesService.skipLineVariableDeclaration({"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}, "[NOT=let zoomConfig = null;]") let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};
SCENARIO 5: FORM 3 – FORM 4	EXPRESSION_START50 = {"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}; let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent}; ELSE50 = { "ELSE": "~~~~~~~~~~~~~~~~~~~~~~~~~~" }; let zoomConfig = null; EXPRESSION_END50 = { "EXPRESSION_END": "--------------------------" };	let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};
SCENARIO 6: *FORM 2 – FORM 4*	// @ts-ignore @DecoratorTypesService.skipLineVariableDeclaration("[NOT=let zoomConfig = null;]") let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};	let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};
SCENARIO 7: *FORM 5 – FORM 4*	// @ts-ignore @DecoratorTypesService.skipLineVariableDeclaration({"OR": { "zoomCoordinates": "true", "zoomValue": "true" }}, "[NOT=let zoomConfig = null;]") let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};	let zoomConfig = {"name": "Zoom", "path": "/puzzle/zoom", "componentPathInModule": "zoom", "componentRef": ZoomManagementComponent};

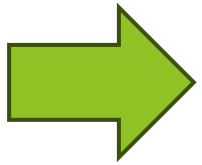
Analogies *…towards code comprehension*

The reason for using the **Halstead measures** in this study is given by the **increase in cognitive processing** demands due to **the number of symbols.**

Schuster, S., Hawelka, S., Himmelstoss, N.A., Richlan, F., Hutzler, F.: The neural correlates of word position and lexical predictability during sentence reading: Evidence from fixation-related fMRI. Language, Cognition and Neuroscience 35(5), 613–624 (Jun 2020)

The cyclomatic complexity was used to **analyze control flows** due to their effects on **rule-guided conditional reasoning.**

No difference in cyclomatic complexity



Annotations with variability expressions should be part of execution flow

Kulakova, E., Aichhorn, M., Schurz, M., Kronbichler, M., Perner, J.: Processing counterfactual and hypothetical conditionals: An fMRI investigation. NeuroImage 72, 265–271 (May 2013)

Hierarchically expressed configuration expressions brings significant complexity.

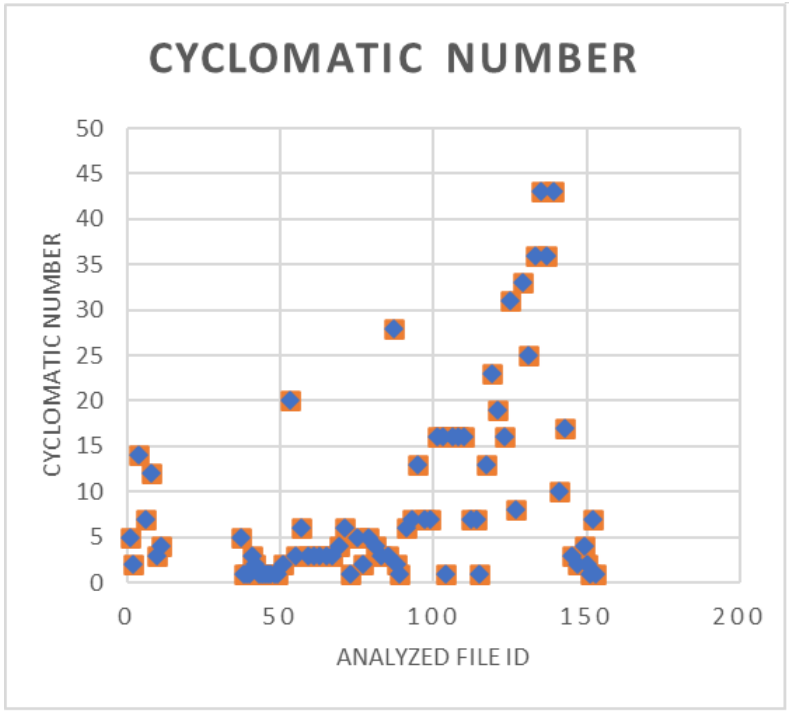
Their code complexity can be used to optimize them for example in automatic evolution process.

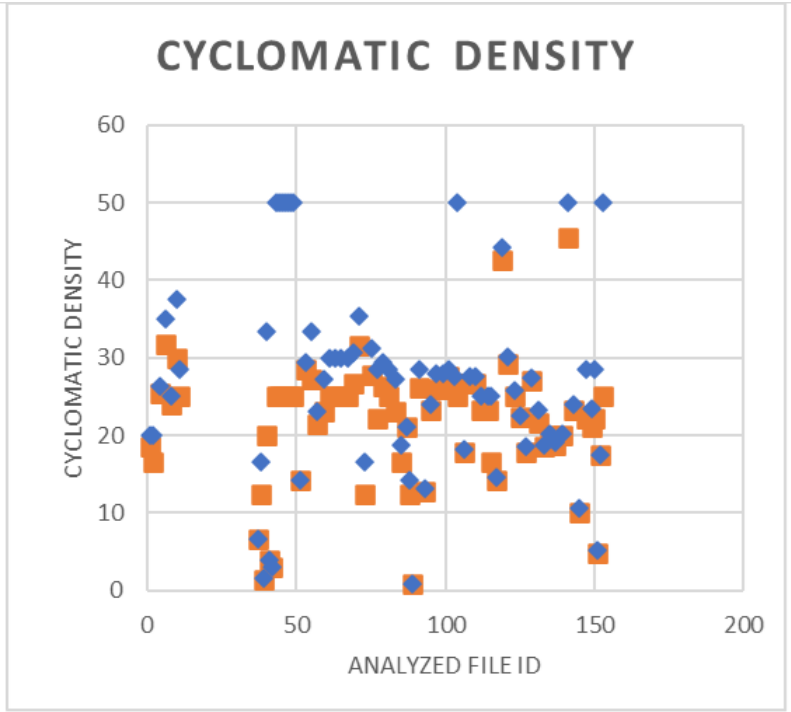
Are traditional wrappers more complex than decorators?

Hypothesis 2: Changing from wrappers to decorators significantly improves the complexity of most evaluated complexity metrics.

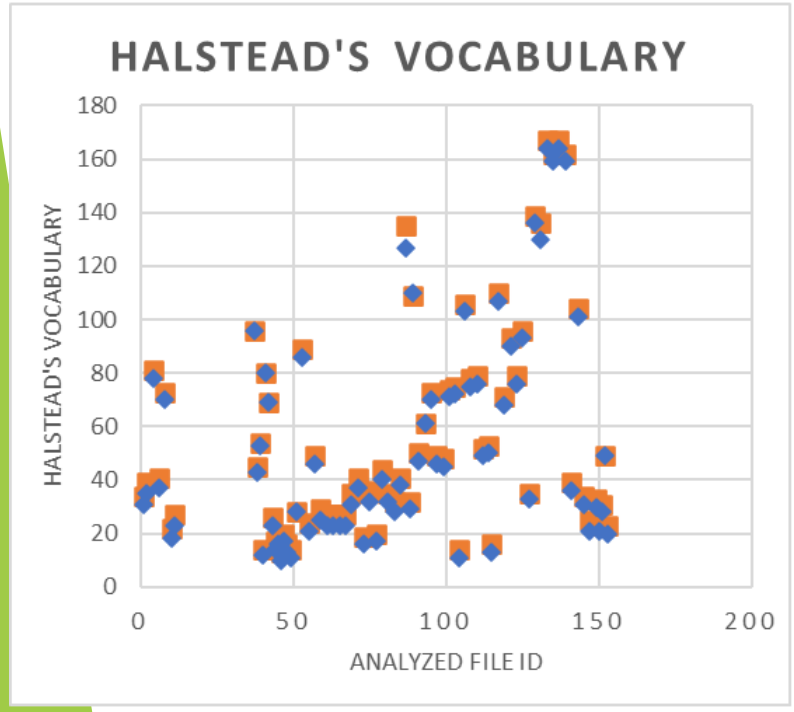
Table 2: Code complexity for Case 3 and 1 compared.

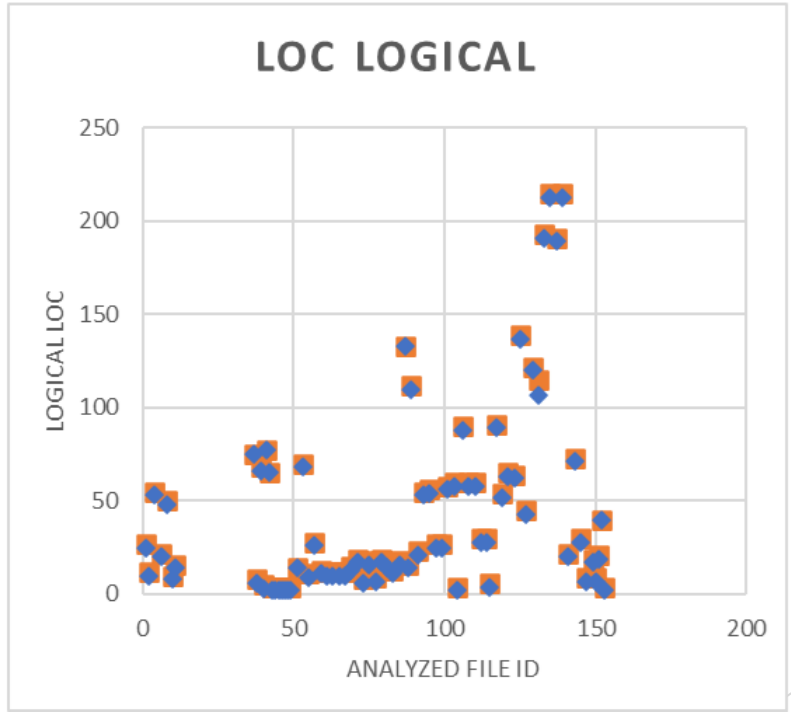
Name of compared metric	**Corr.**	**W**	**p-value**	**95% CI**	**Est.**	**p>0.05**
Cyclomatic Complexity	1.0000	0	1.0000E+00	NaN, NaN	NaN	TRUE
Cyclomatic Density	0.8226	0	3.5776E-13	-4.1959, -2.28	-3.02	**FALSE**
Halstead's Bugs	0.9997	2556	2.4526E-13	0.01, 0.02	0.0141	**FALSE**
Halstead's Difficulty	0.9971	2237	5.9298E-09	0.60, 0.80	0.7390	**FALSE**
Halstead's Effort	0.9988	2386	2.2106E-10	503.04, 1493.10	841.6983	**FALSE**
Halstead's Length	0.9997	2556	9.3382E-17	6.00, 6.00	6.0000	**FALSE**
Halstead's Time	0.9988	2386	2.2106E-10	27.95, 82.95	46.7609	**FALSE**
Halstead's Vocabulary	0.9994	2484	4.2116E-14	3.00, 3.45	3.0000	**FALSE**
Halstead's Volume	0.9997	2556	2.4761E-13	39.32, 45.96	42.2363	**FALSE**
Halstead's Id Dist. Operands	0.9996	2415	2.5713E-16	2.00, 2.00	2.0000	**FALSE**
Halstead's Id Ttl Operands	0.9999	2556	9.3382E-17	2.00, 2.00	2.0000	**FALSE**
Halstead's Id Dist. Operators	0.9886	2030	2.1410E-12	1.00, 1.50	1.0000	**FALSE**
Halstead's Id Ttl Operators	0.9999	2485	9.8502E-17	4.00, 4.00	4.0000	**FALSE**
LOC Physical	0.9998	2556	2.1563E-16	1.00, 1.00	1.0000	**FALSE**
LOC Logical	0.9999	2485	9.8502E-17	2.00, 2.00	2.0000	**FALSE**

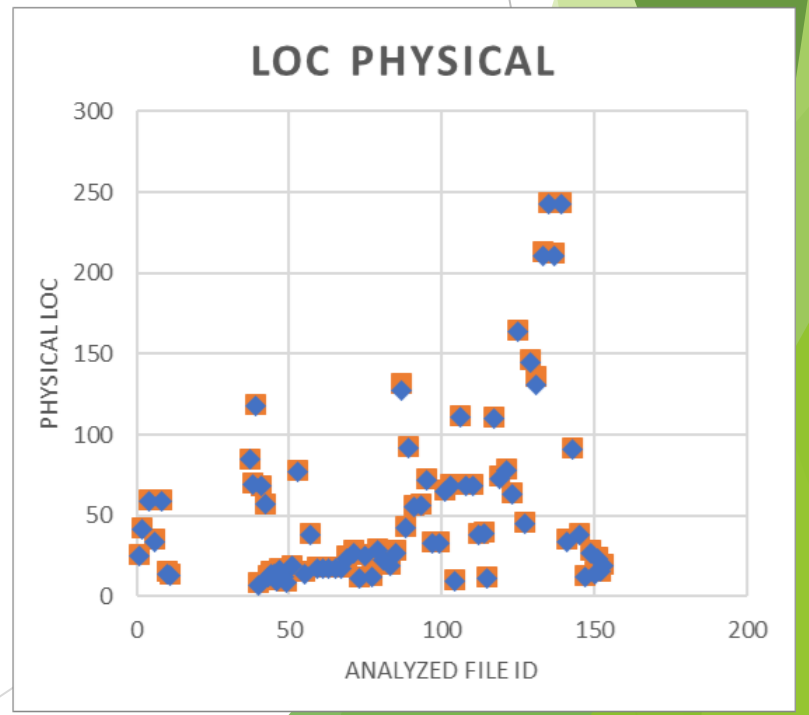

CYCLOMATIC NUMBER
CYCLOMATIC NUMBER
ANALYZED FILE ID
0
5
10
15
20
25
30
35
40
45
50
0
50
100
150
200

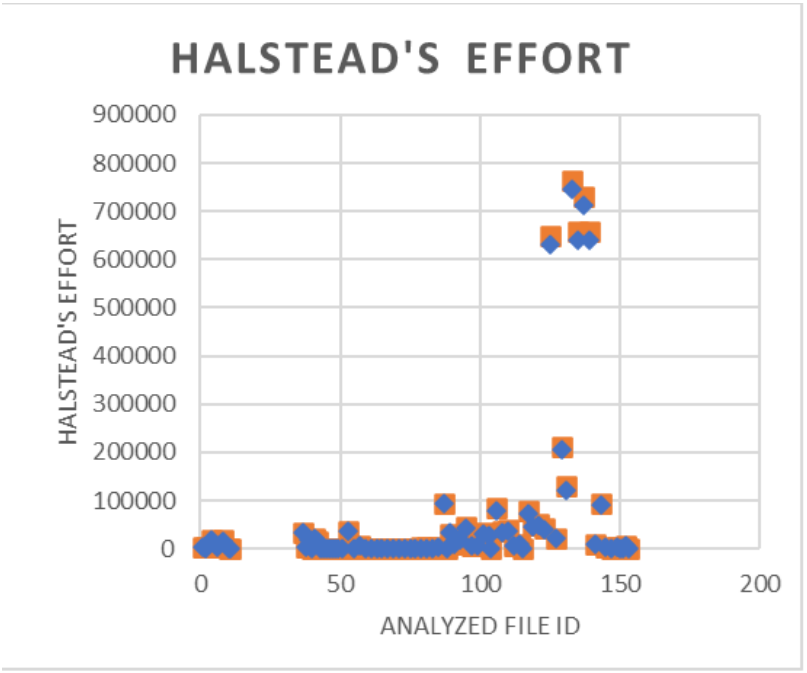

CYCLOMATIC DENSITY
CYCLOMATIC DENSITY
ANALYZED FILE ID
0
10
20
30
40
50
60
0
50
100
150
200

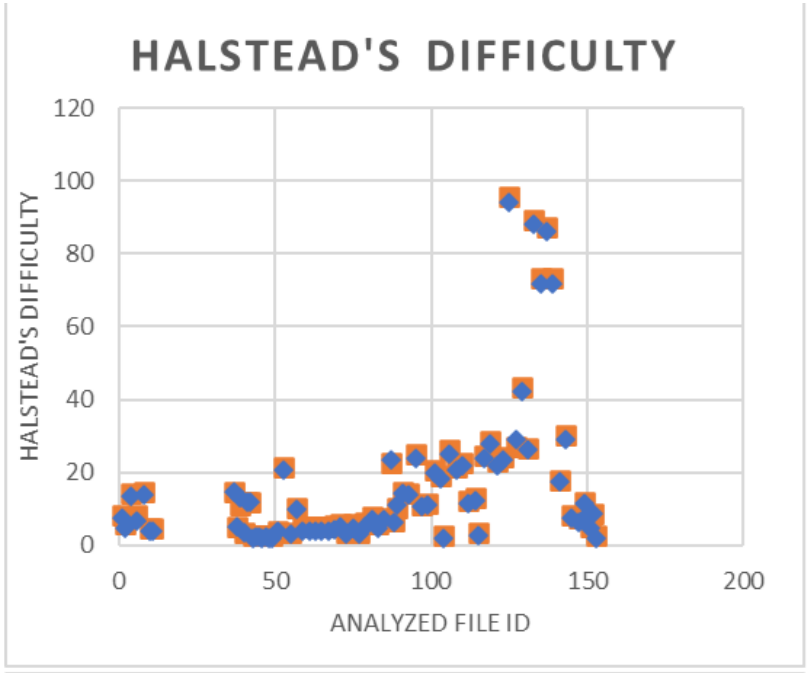

LEGEND
Wrapper-based
Decorator-based

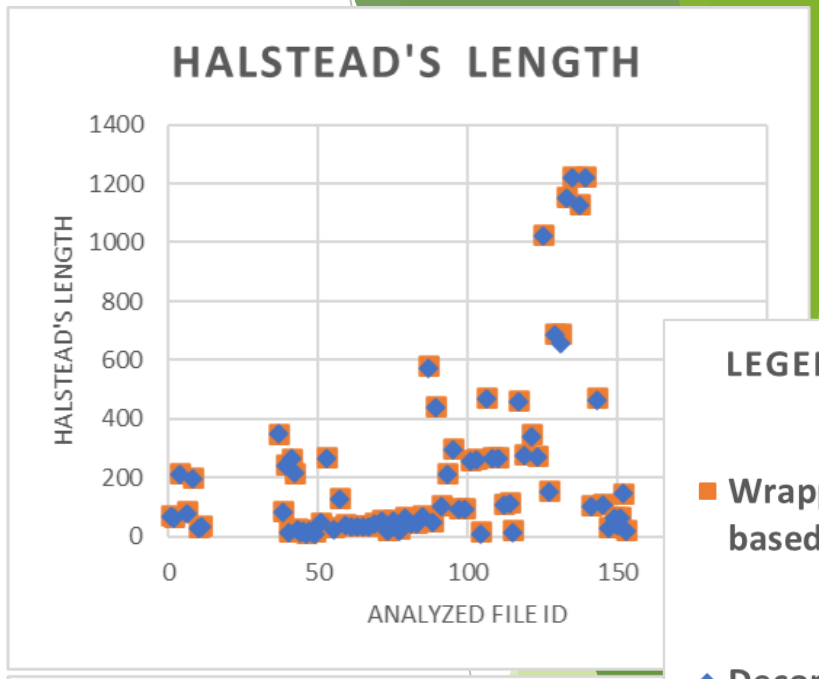

HALSTEAD'S VOCABULARY
HALSTEAD'S VOCABULARY
ANALYZED FILE ID
0
20
40
60
80
100
120
140
160
180
0
50
100
150
200

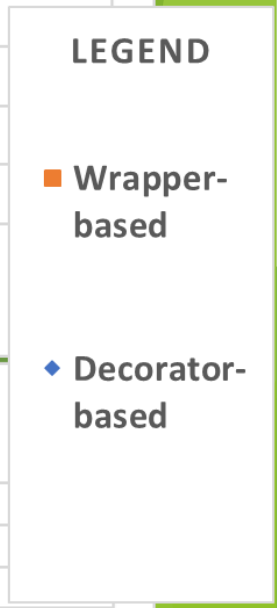

LOC LOGICAL
LOGICAL LOC
ANALYZED FILE ID
0
50
100
150
200
250
0
50
100
150
200

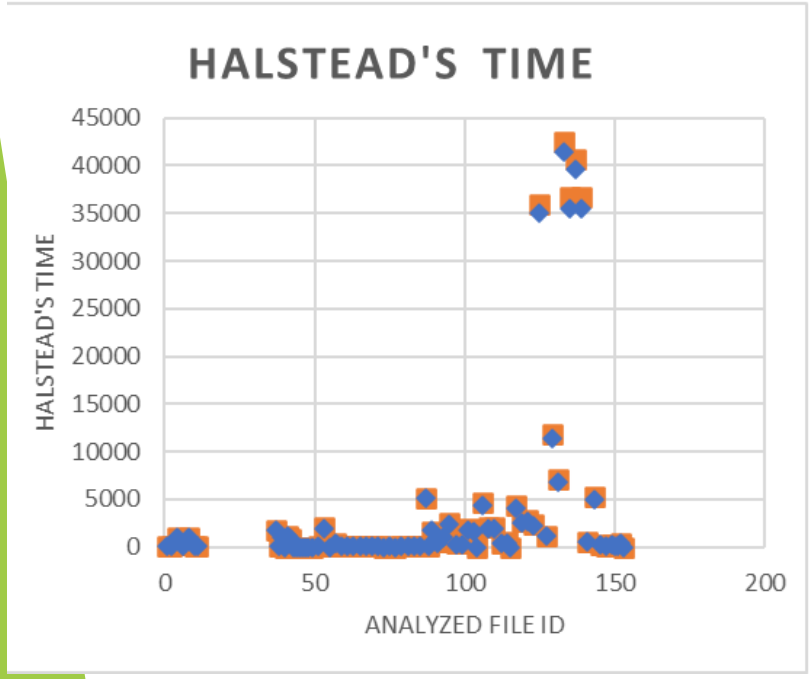

LOC PHYSICAL
PHYSICAL LOC
ANALYZED FILE ID
0
50
100
150
200
250
300
0
50
100
150
200

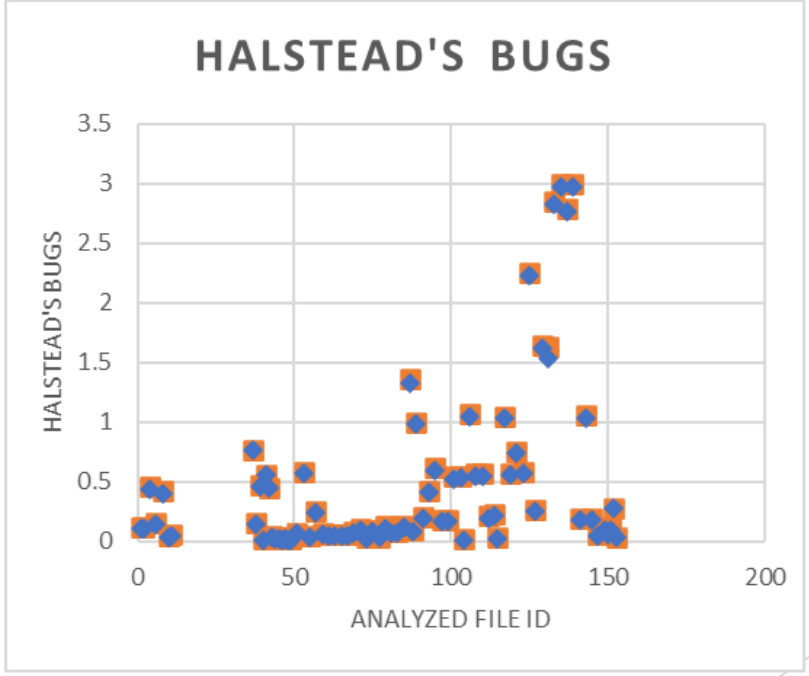

HALSTEAD'S EFFORT
HALSTEAD'S EFFORT
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
0
50
100
150
200
ANALYZED FILE ID

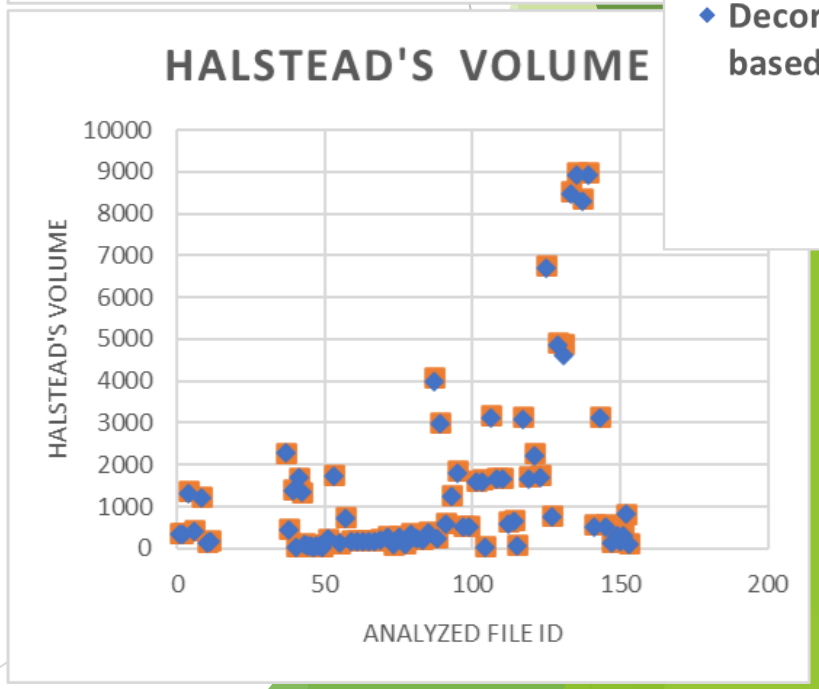

HALSTEAD'S DIFFICULTY
HALSTEAD'S DIFFICULTY
120
100
80
60
40
20
0
0
50
100
150
200
ANALYZED FILE ID


HALSTEAD'S LENGTH
HALSTEAD'S LENGTH
1400
1200
1000
800
600
400
200
0
0
50
100
150
ANALYZED FILE ID


LEGEND
Wrapper-based
Decorator-based


HALSTEAD'S TIME
HALSTEAD'S TIME
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
ANALYZED FILE ID


HALSTEAD'S BUGS
HALSTEAD'S BUGS
3.5
3
2.5
2
1.5
1
0.5
0
0
50
100
150
200
ANALYZED FILE ID


HALSTEAD'S VOLUME
HALSTEAD'S VOLUME
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
ANALYZED FILE ID

Decorators are significantly less complex than wrappers even for a few variable features.

Conventional solutions based on wrappers such as pure::variants or conditional compilation can be enhanced to managing features in code.

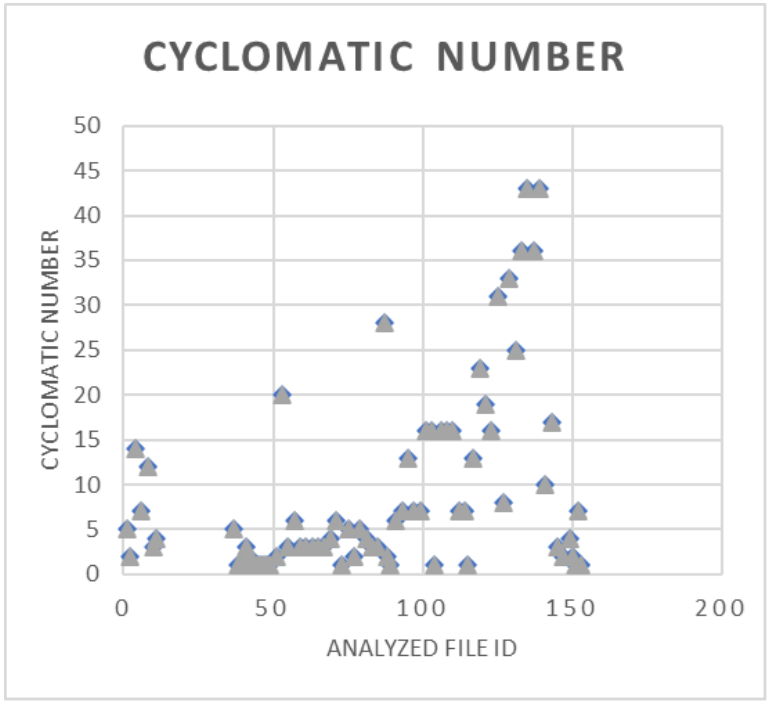
Is complexity of variability management significant for used code complexity measures?

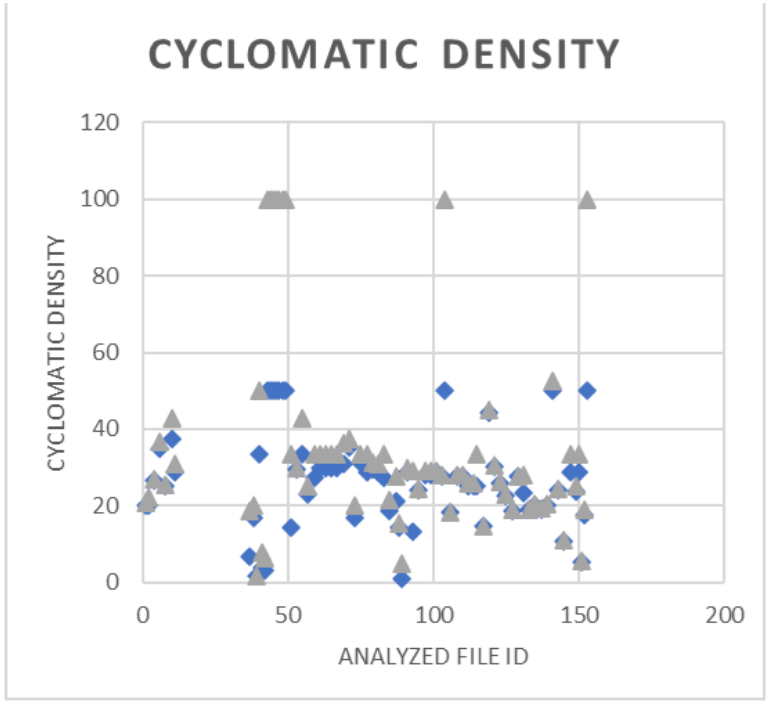
…is less complex decorator-detachable version insignificant to most of code-complexity measures?

Hypothesis 3: Removal of all variability constructs from Case 1 does not significantly change at least one of the evaluated complexity metrics.

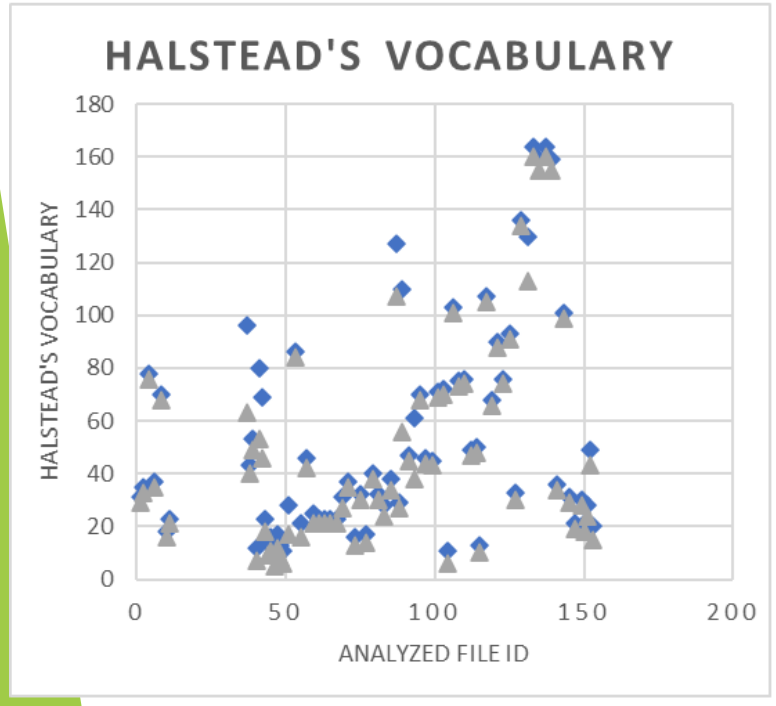
Table 3: Code complexity for Case 1 and 4 compared.

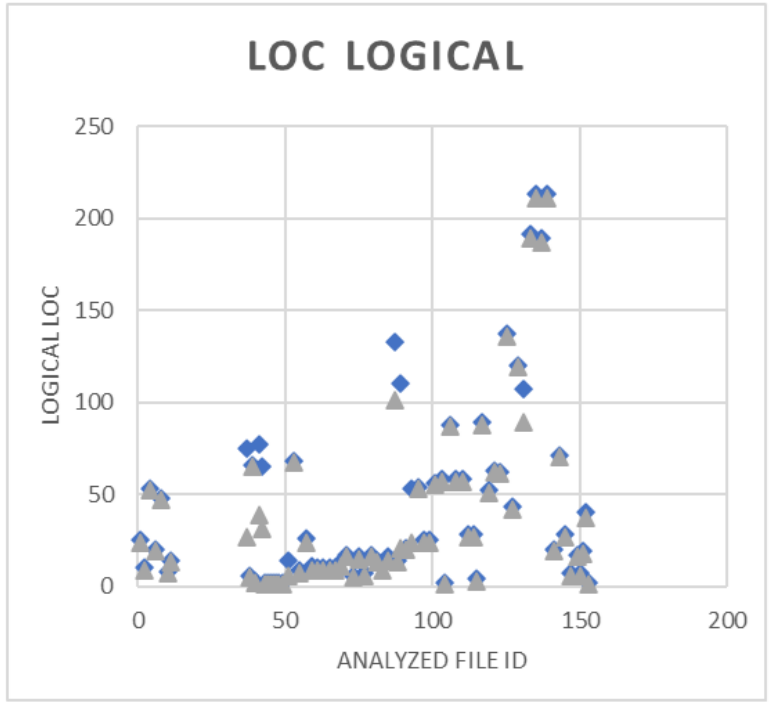
Name of compared metric	**Corr.**	**W**	**p-value**	**95% CI**	**Est.**	**p>0.05**
Cyclomatic Complexity	1.0000	0	NaN	NaN, NaN	NaN	TRUE
Cyclomatic Density	0.9264	0	3.6200E-14	-4.63, -2.05)	-2.9825	FALSE
Halstead's Bugs	0.9900	2926	3.6200E-14	0.01, 0.02	0.0130	FALSE
Halstead's Difficulty	0.9921	1489	8.9495E-01	-0.19, 0.52	0.0280	**TRUE**
Halstead's Effort	0.9920	2486	1.2000E-07	123.31, 199.05	155.8794	FALSE
Halstead's Length	0.9885	2926	1.2500E-15	5.00, 6.50	5.0001	FALSE
Halstead's Time	0.9920	2486	1.2000E-07	6.85, 11.06	8.6572	FALSE
Halstead's Vocabulary	0.9880	2926	1.2900E-14	3.00, 3.50	3.0000	FALSE
Halstead's Volume	0.9903	2926	3.6700E-14	35.18, 45.23	38.6939	FALSE
Halstead's Id Dist. Operands	0.9855	2926	1.2500E-15	2.00, 3.00	2.0001	FALSE
Halstead's Id Ttl Operands	0.9891	2926	1.2500E-15	2.00, 3.00	2.0001	FALSE
Halstead's Id Dist. Operators	0.9928	406	2.6600E-06	1.50, 2.00	1.9999	FALSE
Halstead's Id Ttl Operators	0.9863	2926	1.2500E-15	3.00, 3.50	3.0001	FALSE
LOC Physical	0.9904	2926	1.0300E-16	2.00, 2.00	2.0000	FALSE
LOC Logical	0.9734	2926	1.2500E-15	1.00, 1.50	1.0001	FALSE

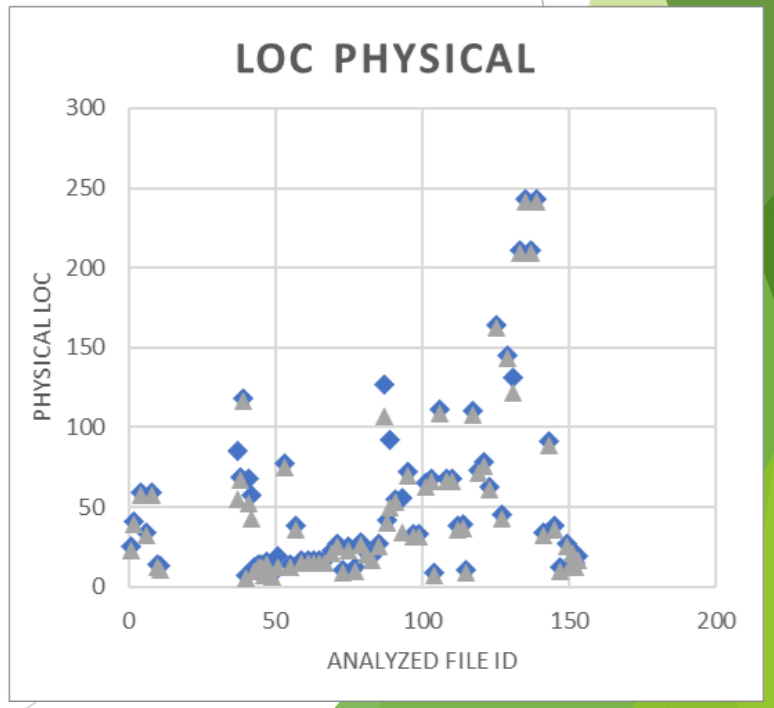

CYCLOMATIC NUMBER
CYCLOMATIC NUMBER
ANALYZED FILE ID
0
5
10
15
20
25
30
35
40
45
50
0
50
100
150
200

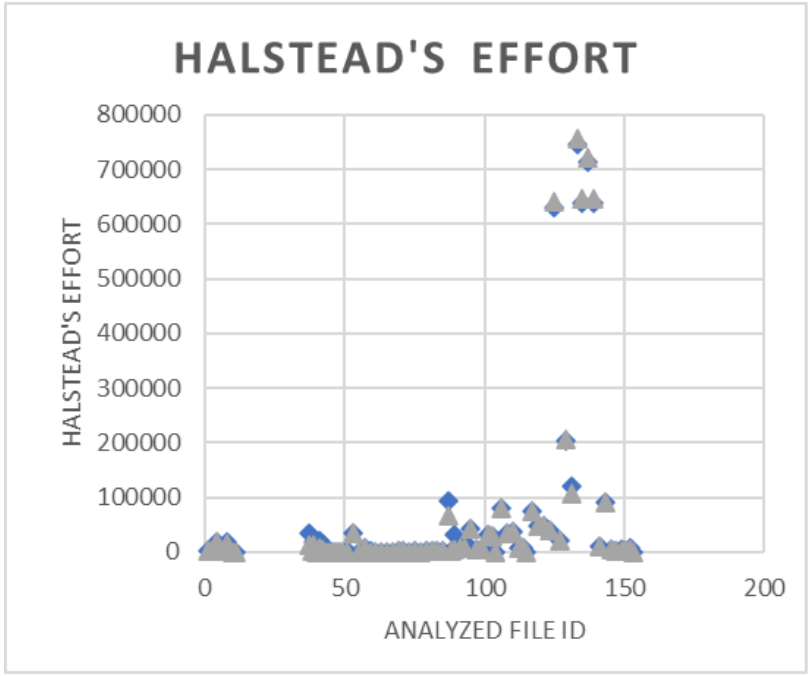

CYCLOMATIC DENSITY
CYCLOMATIC DENSITY
ANALYZED FILE ID
0
20
40
60
80
100
120
0
50
100
150
200

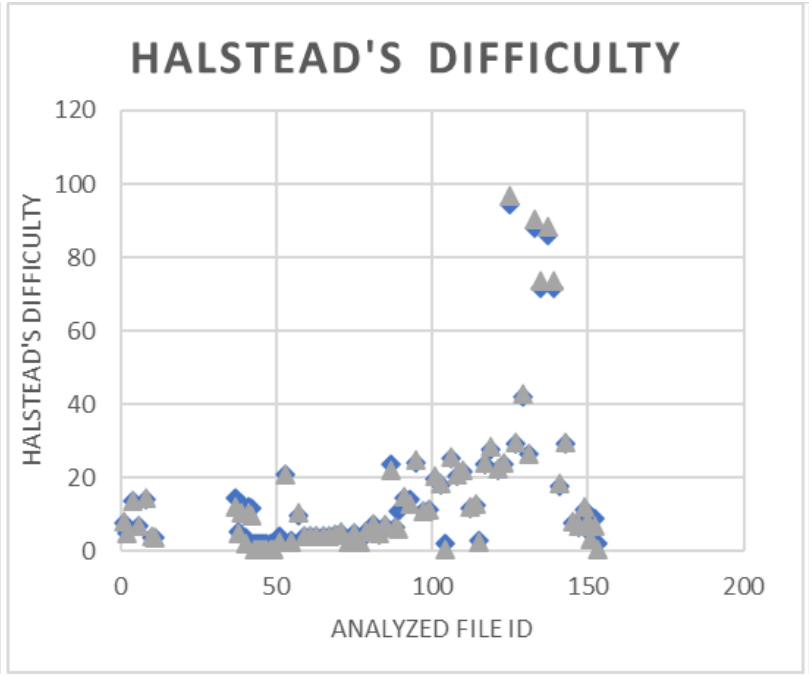

LEGEND
Decorator-based
Without variability

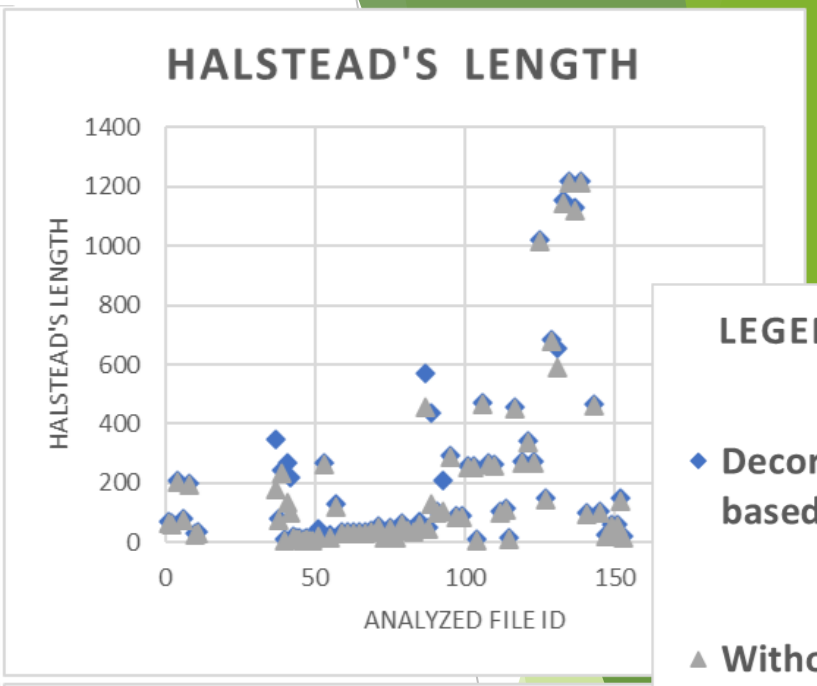

HALSTEAD'S VOCABULARY
HALSTEAD'S VOCABULARY
ANALYZED FILE ID
0
20
40
60
80
100
120
140
160
180
0
50
100
150
200

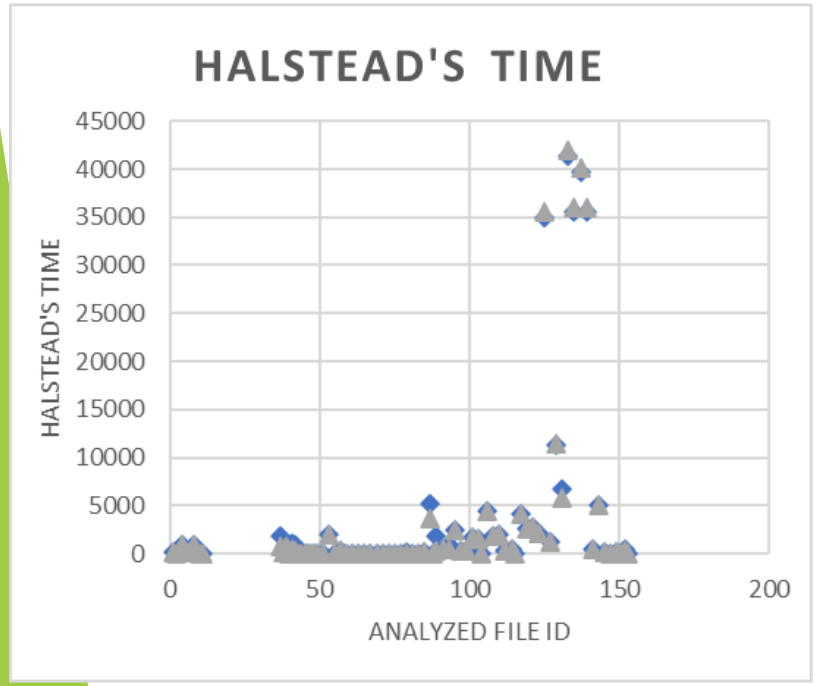

LOC LOGICAL
LOGICAL LOC
ANALYZED FILE ID
0
50
100
150
200
250
0
50
100
150
200

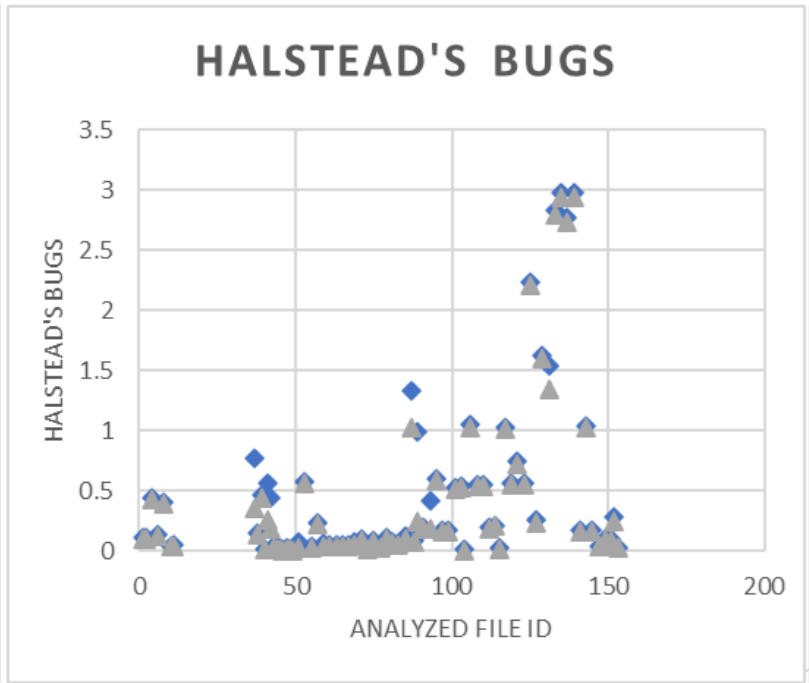

LOC PHYSICAL
PHYSICAL LOC
ANALYZED FILE ID
0
50
100
150
200
250
300
0
50
100
150
200

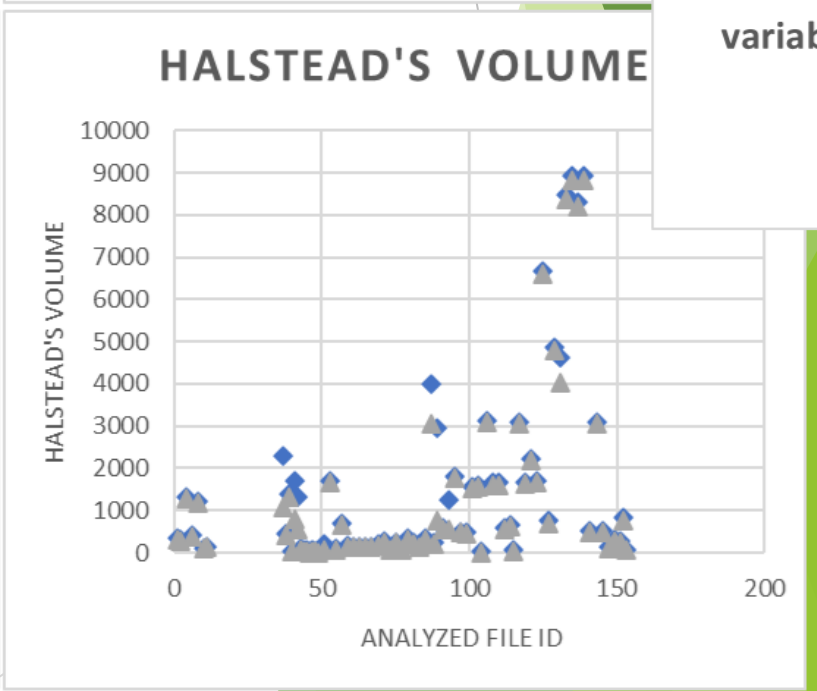

HALSTEAD'S EFFORT
HALSTEAD'S EFFORT
800000
700000
600000
500000
400000
300000
200000
100000
0
0
50
100
150
200
ANALYZED FILE ID

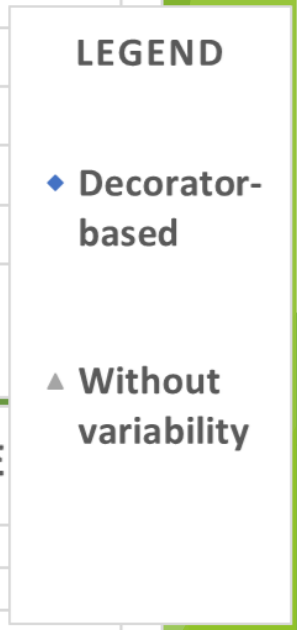

HALSTEAD'S DIFFICULTY
HALSTEAD'S DIFFICULTY
120
100
80
60
40
20
0
0
50
100
150
200
ANALYZED FILE ID


HALSTEAD'S LENGTH
HALSTEAD'S LENGTH
1400
1200
1000
800
600
400
200
0
0
50
100
150
ANALYZED FILE ID

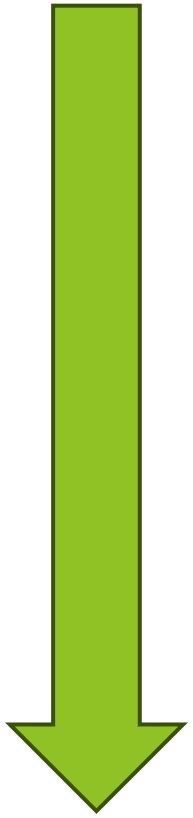

HALSTEAD'S TIME
HALSTEAD'S TIME
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
ANALYZED FILE ID


HALSTEAD'S BUGS
HALSTEAD'S BUGS
3.5
3
2.5
2
1.5
1
0.5
0
0
50
100
150
200
ANALYZED FILE ID


HALSTEAD'S VOLUME
HALSTEAD'S VOLUME
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
ANALYZED FILE ID


LEGEND
Decorator-based
Without variability

Narrowing test focused on already supported decorators in TypeScript



The code complexity change after removal of files with most of (unconvertable decorators into) wrappers

Table 4: Code complexity for Case 1 and 4 compared without most of the files with wrappers.

Name of compared metric	**Corr.**	**W**	**p-value**	**95% CI**	**Est.**	**p>0.05**
Cyclomatic Complexity	1.0000	0	NaN	NaN, NaN	NaN	TRUE
Cyclomatic Density	0.9277	0	***1.6427E-12***	-4.58, -1.81	-2.7695	FALSE
Halstead's Bugs	0.9969	2211	***1.6442E-12***	0.01, 0.01	0.0120	FALSE
Halstead's Difficulty	0.9948	886	**1.6171E-01**	-0.28, 0.11	-0.1545	TRUE
Halstead's Effort	0.9979	1787	***1.3588E-05***	102.09, 152.93	126.98	FALSE
Halstead's Length	0.9965	2211	**2.5043E-14**	5.00, 5.00	5.0000	FALSE
Halstead's Time	0.9979	1787	***1.3588E-05***	5.67, 8.50	7.0545	FALSE
Halstead's Vocabulary	0.9963	2211	**4.1183E-13**	2.50, 3.50	2.9999	FALSE
Halstead's Volume	0.9969	2211	***1.6743E-12***	32.77, 38.41	35.4739	FALSE
Halstead's Id Dist. Operands	0.9953	2211	**2.5043E-14**	2.00, 2.00	2.0001	FALSE
Halstead's Id. Ttl Operands	0.9954	2211	**2.5043E-14**	2.00, 2.00	2.0001	FALSE
Halstead's Id. Dist. Operators	0.9958	171	***1.4868E-04***	2.00, 3.00	2.0000	FALSE
Halstead's Id. Ttl Operators	0.9953	2211	**2.5043E-14**	3.00, 3.00	3.0001	FALSE
LOC Physical	0.9980	2211	7.4931E-16	2.00, 2.00	2.0000	FALSE
LOC Logical	0.9958	2211	**2.5043E-14**	1.00, 1.00	1.0001	FALSE

The variability management significantly influences the code complexity

…the annotated code of fragments of variable features should be filtered according to particular manipulations with code to decrease it

Is dead code introduced with use of illegal decorators in TypeScript significant for used code complexity measures?

…should some of illegal decorators by supported in future version for variability management?

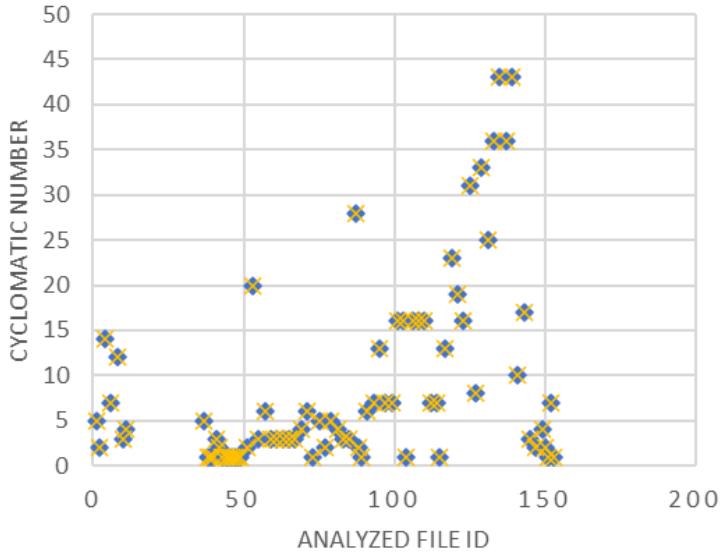
Hypothesis 4:

Unwanted dead code constructs significantly change complexity measured by most evaluated complexity metrics.

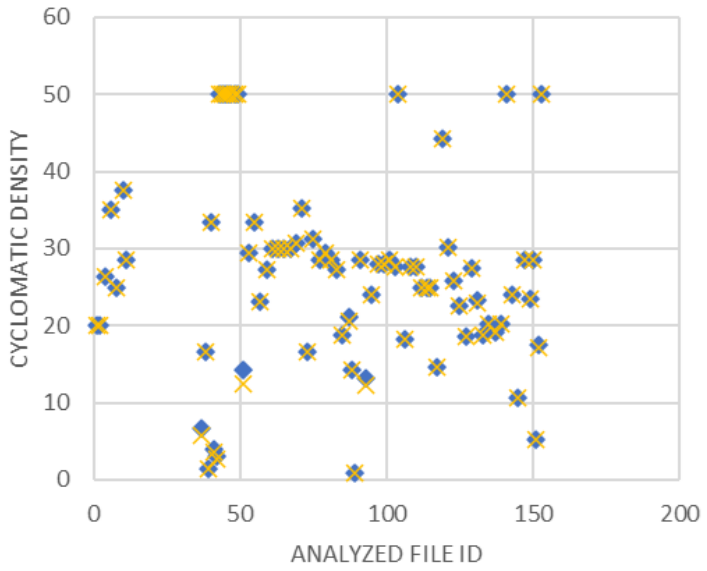
Table 5: Code complexity for Case 5 and 1 compared.

Name of compared metric	**Corr.**	**W**	**p-value**	**95% CI**	**Est.**	**p>0.05**
Cyclomatic Complexity	1.0000	0	1.0000	NaN, NaN	NaN	TRUE
Cyclomatic Density	1.0000	0	0.0092	-1.11, -0.25	-0.495	**FALSE**
Halstead's Bugs	0.9998	45	0.0092	0.01, 0.04	0.0205	**FALSE**
Halstead's Difficulty	0.9999	40	0.0440	0.01, 0.53	0.2210	**FALSE**
Halstead's Effort	0.9996	45	0.0092	551.81, 2876.1	1199	**FALSE**
Halstead's Length	0.9998	45	0.0091	4.00, 16.00	9.0001	**FALSE**
Halstead's Time	0.9996	45	0.0092	30.66, 159.78	66.603	**FALSE**
Halstead's Vocabulary	0.9999	45	0.0034	NaN, NaN	1.0000	**FALSE**
Halstead's Volume	0.9998	45	0.0092	32.04, 108.08	62.861	**FALSE**
Halstead's Id Dist. Operands	0.9999	45	0.0034	NaN, NaN	1.0000	**FALSE**
Halstead's Id Ttl Operands	0.9996	45	0.0091	2.00, 8.00	4.5000	**FALSE**
Halstead's Id Dist. Operator	1.0000	0	1.0000	NaN, NaN	NaN	TRUE
Halstead's Id Ttl Operators	0.9999	45	0.0091	2.00, 8.00	4.5000	**FALSE**
LOC Physical	0.9990	45	0.0092	2.50, 12.00	5.5000	**FALSE**
LOC Logical	0.9996	45	0.0091	2.00, 8.00	4.5000	**FALSE**

CYCLOMATIC NUMBER



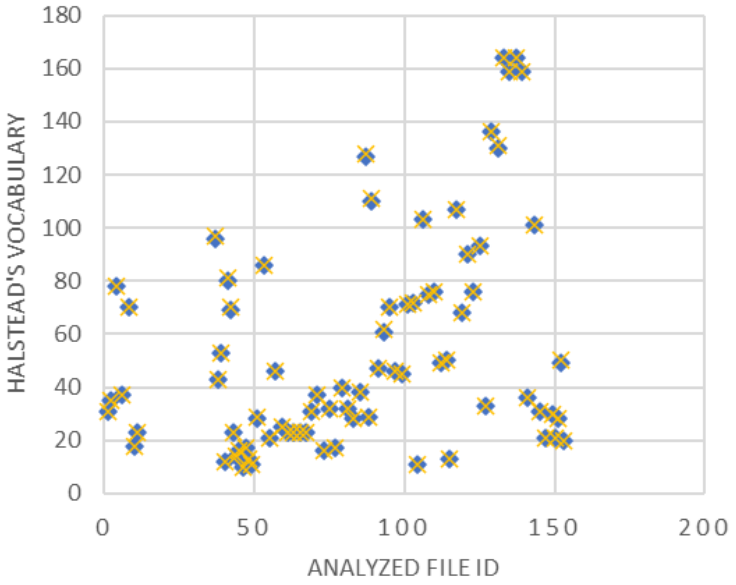
CYCLOMATIC DENSITY



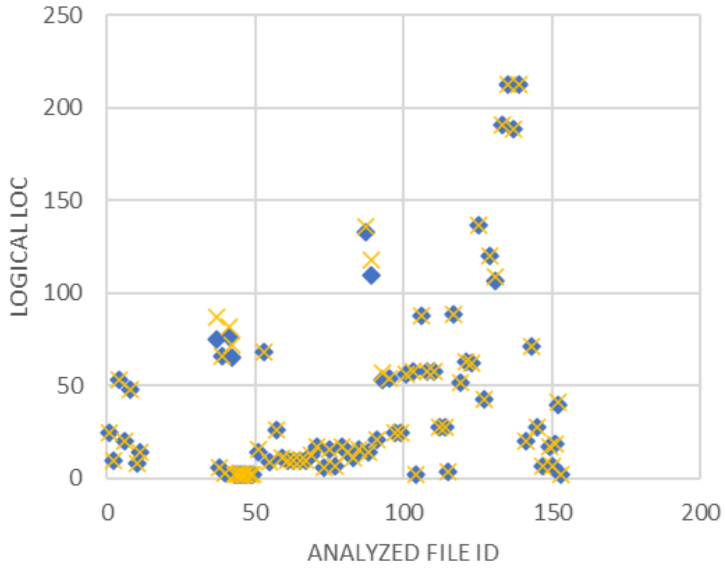
LEGEND

- Decorator-based
- Decorator-based +dead code

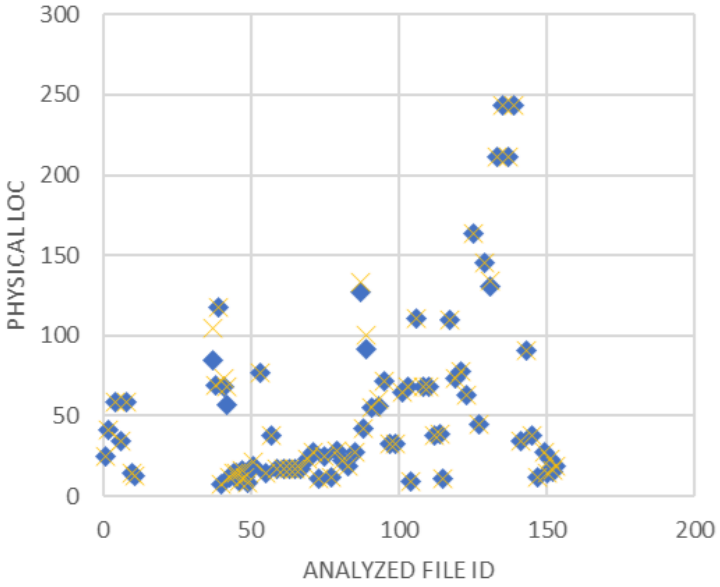
HALSTEAD'S VOCABULARY

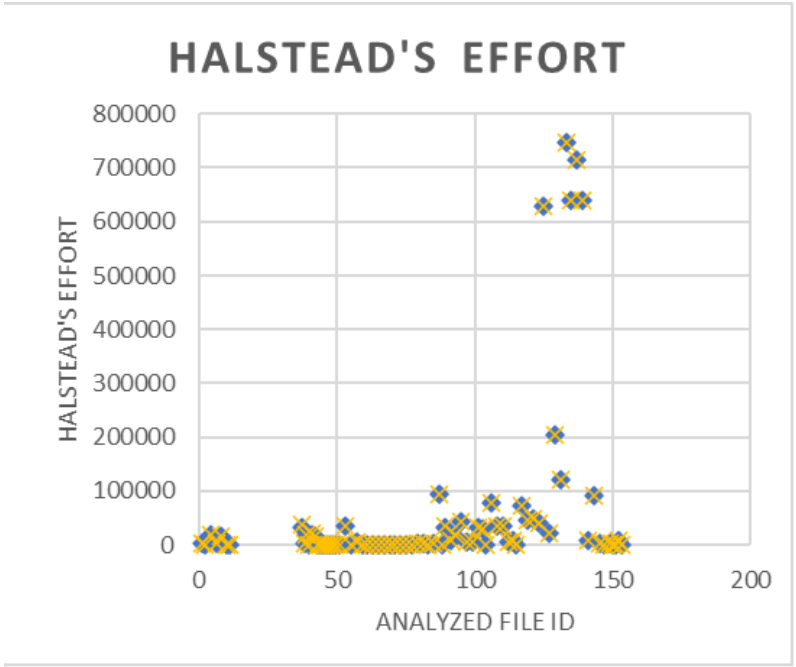


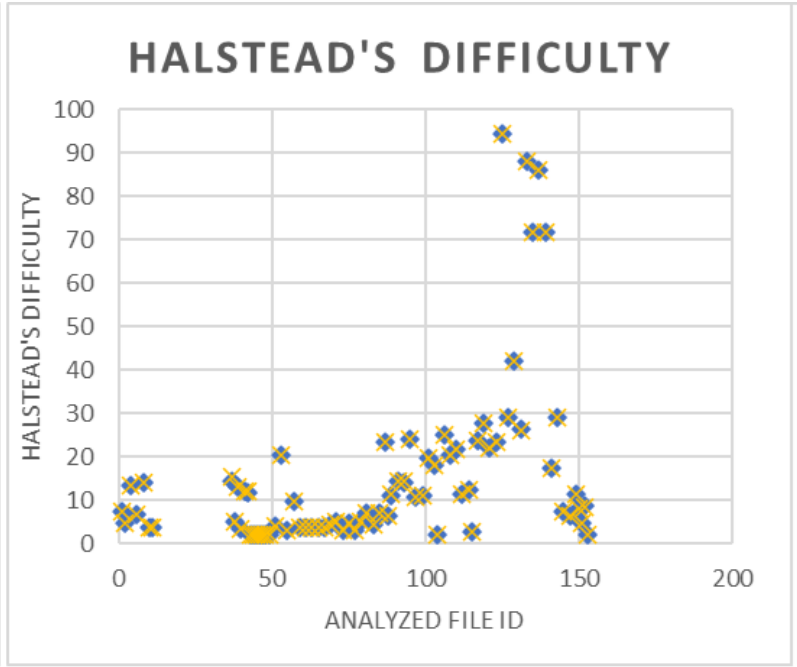
LOC LOGICAL

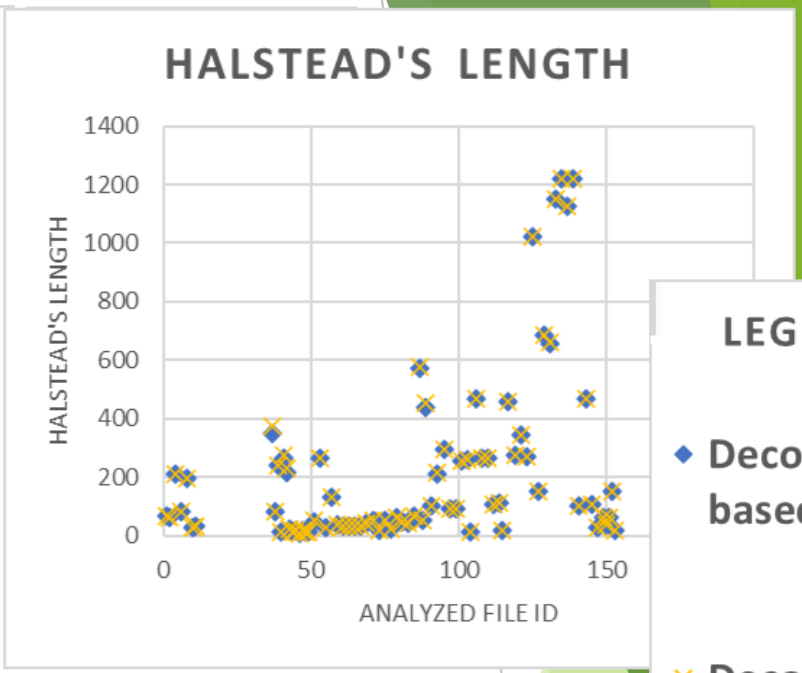


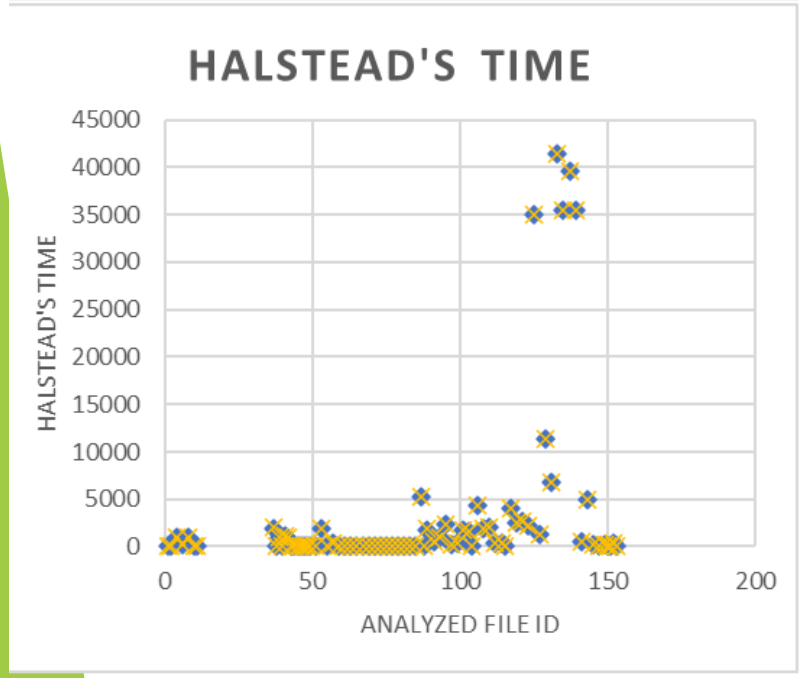
LOC PHYSICAL

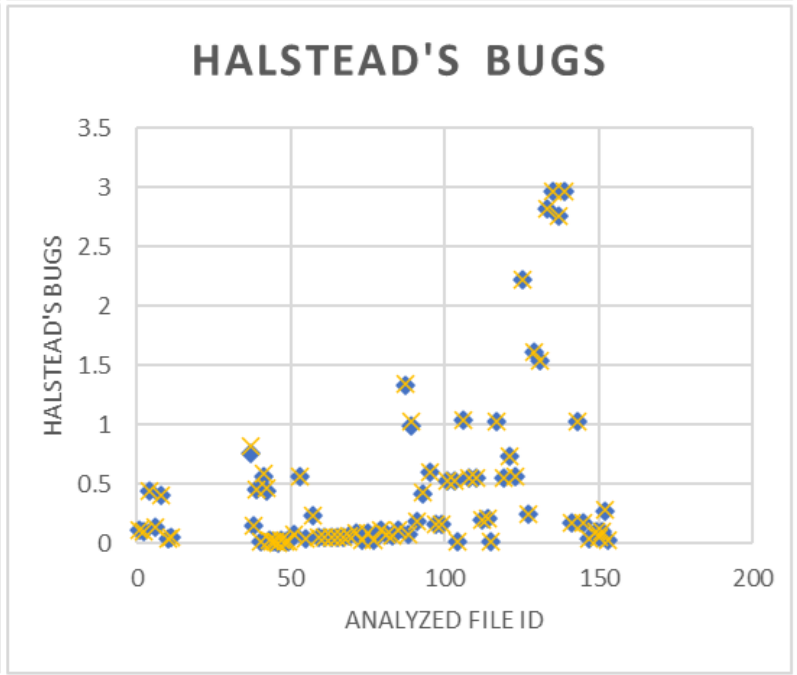


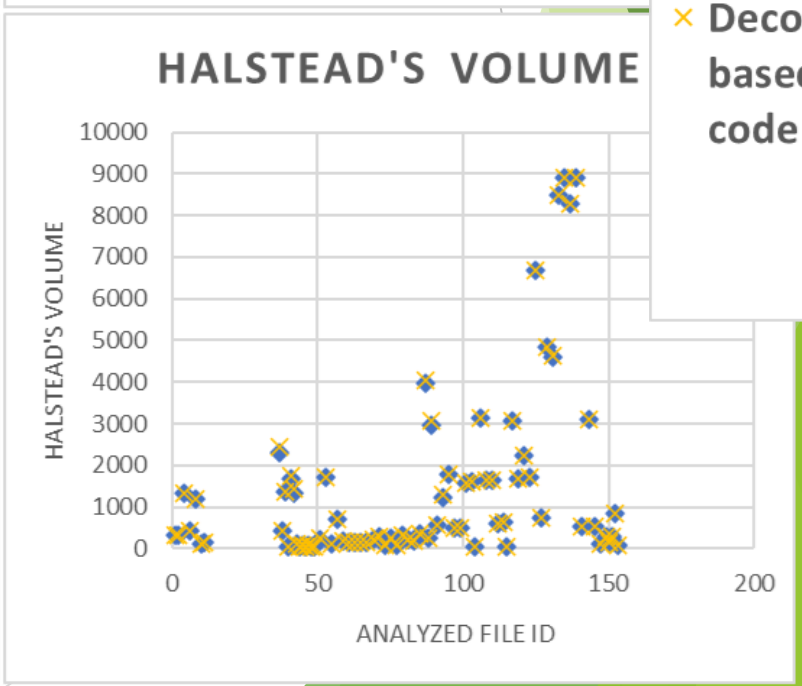












LEGEND

- Decorator-based
- Decorator-based +dead code

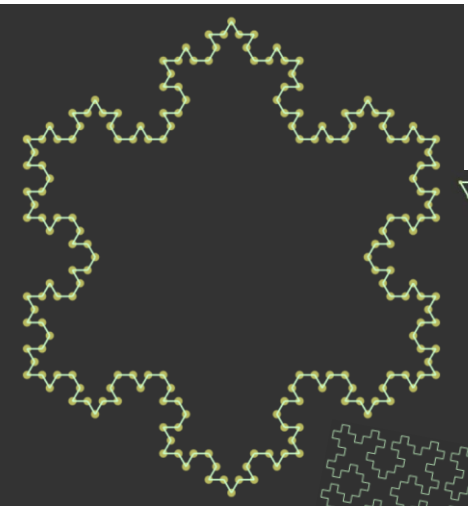
The support of function and import decorators is necessary to reduce dead code with borderline but still significant impact on code complexity.

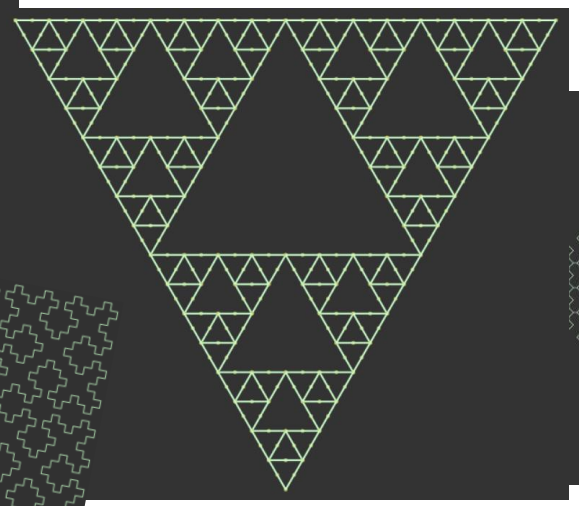
Future work

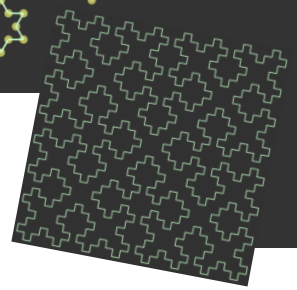
- Optimizing configuration expressions on fractals where variability is modeled and managed in large (many features)
  - To suit interacting features
  - To suit features on the same layer
  - To suit features on a particular hierarchy tree
  - combinations of approaches above
- Introducing variability filtering according to features, concerns, and code complexity
- Observing a variability-oriented cyclomatic number by introducing program flow that directly contains variability conditions taken from configuration expressions
- Evaluate other code complexity measures and observe their influence on user cognitive processing and comprehension

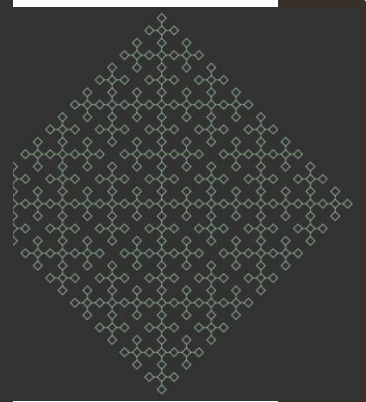
Optimalization of configuration expressions in large

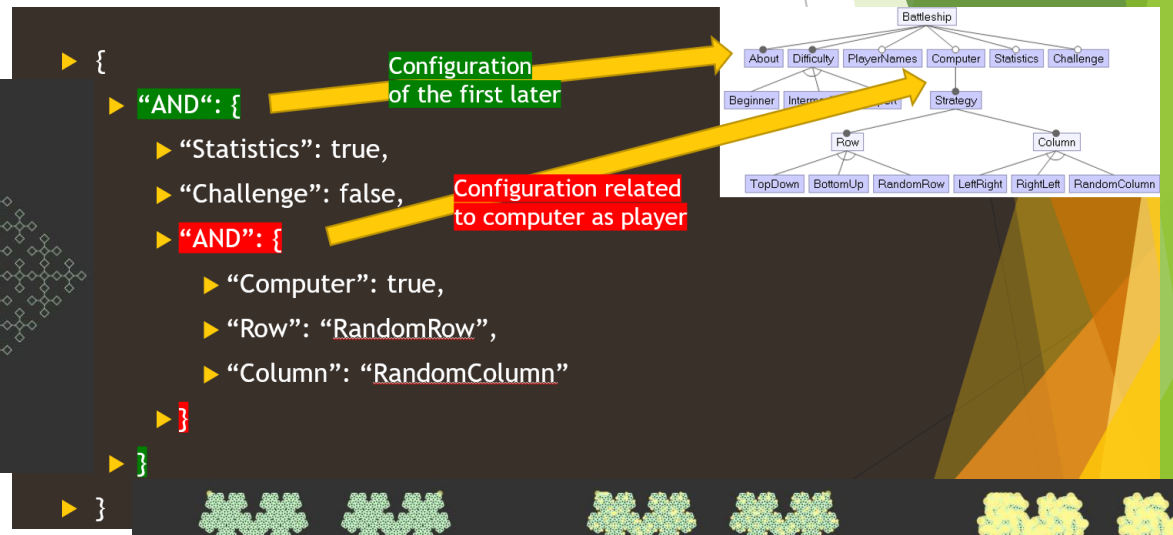
...creating hierarchic structures and evaluating the comprehension of feature models in code



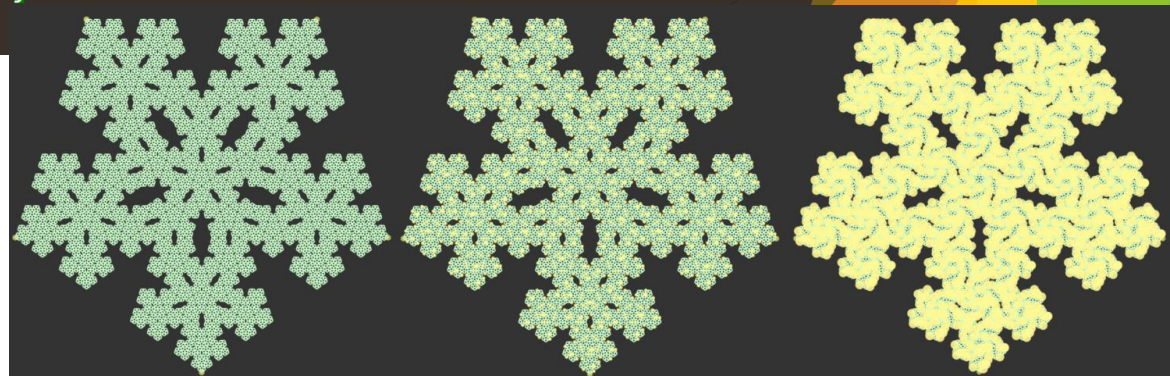


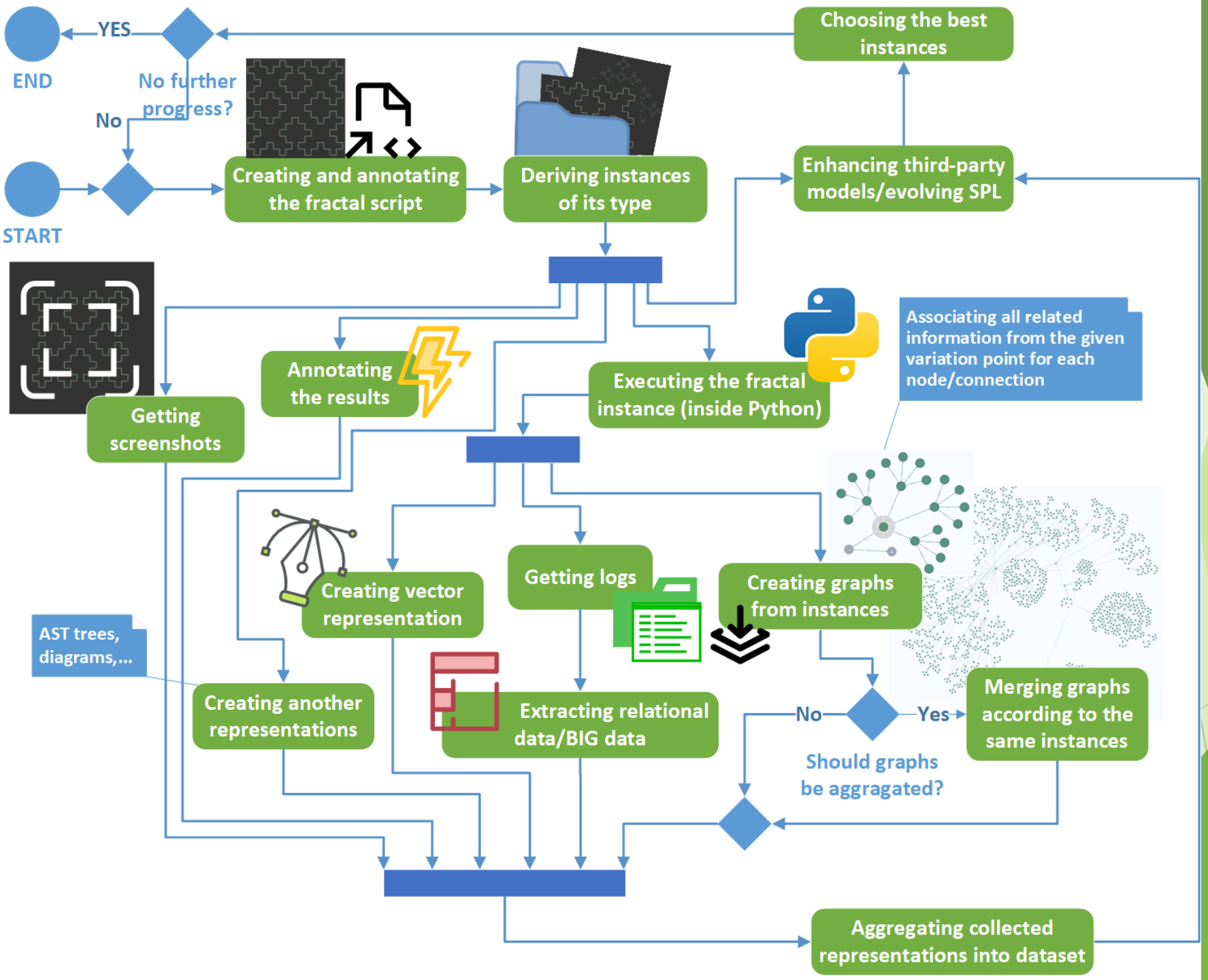






via automated software product line evolution focused on generating fractal shapes




YES
END
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 aggragated?
Merging graphs according to the same instances
Aggregating collected representations into dataset

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