* RQ1) What are the different code smells that studies look at?
  + Abbes et al., 2011, Blob and spaghetti antipatterns
    - Languages of systems:
      * Java
    - Domains of systems:
      * Email client
      * Project scheduling and management
      * XML parsing
    - Smells:
      * Blob
      * Spaghetti Code
  + Yamashita and Moonen, 2013, Inter-smell relations on software maintainability
    - Languages of systems:
      * Java
    - Domains of systems:
      * Web-based information systems to keep track of empirical studies and resulting scientific publications
    - Smells:
      * Data Class
      * Data Clump
      * Duplicate code in conditional branches
      * Feature Envy
      * God Class
      * God Method
      * Misplaced Class
      * Refused Bequest
      * Shotgun Surgery
      * Temporary variable used for several purposes
      * Use interface instead of implementation
      * Interface Segregation Principle Violation (ISP Violation)
  + Yamashita et al., 2015, Inter-smell relations in industrial and open-source systems
    - Languages of systems:
      * Java
    - Domains of systems:
      * Research grant application manager
      * Search server
      * Distributed or scalable machine learning algorithms in areas of collaborative filtering, clustering and classification
    - Smells:
      * Refused Parent Bequest
      * Distorted Hierarchy
      * Schizophrenic Class
      * God Class
      * Tradition Breaker
      * Sibling Duplication
      * Data Clumps
      * Blob Operation
      * Feature Envy
      * Shotgun Surgery
      * Internal Duplication
      * Message Chains
      * External Duplication
      * Intensive Coupling
      * Data Class
  + Palomba et al., 2016, Diffusion of test smells in auto-generated test code
    - Languages of systems:
      * Java/JUnit
    - Domains of systems:
      * 110 open-source software projects with tests generated by EvoSuite
    - Smells (test):
      * Mystery Guest
      * Resource Optimism
      * Eager Test
      * Assertion Roulette
      * Indirect Testing
      * For Testers Only (not reported in code smell co-occurrences though)
      * Sensitive Equality
      * Test Code Duplication
  + Palomba et al., 2017, Code smell co-occurrences and association rule learning
    - Languages of systems:
      * Java
    - Domains of systems:
      * 30 open-source software projects
    - Smells:
      * Class Data Should Be Private (CDSBP)
      * Complex Class
      * Feature Envy
      * God Class
      * Inappropriate Intimacy
      * Lazy Class
      * Long Method
      * Long Parameter List (LPL)
      * Message Chain
      * Middle Man
      * Refused Bequest
      * Spaghetti Code
      * Speculative Generality
  + Palomba et al., 2018, Code smell co-occurrence lifecycle
    - Languages of systems:
      * Java
    - Domains of systems:
      * 30 software projects from Apache Software Foundation and Eclipse Development Framework
    - Smells:
      * Blob
      * Class Data Should Be Private (CDSBP)
      * Complex Class
      * Feature Envy
      * Inappropriate Intimacy
      * Lazy Class
      * Long Method
      * Long Parameter List
      * Message Chain
      * Middle Man
      * Refused Bequest
      * Spaghetti Code
      * Speculative Generality
  + Martins et al., 2020, Code smell co-occurrence impact on quality attributes
    - Languages of systems:
      * Java
    - Domains of systems:
      * Electronic dental record
      * Academic offer
      * Warehouse
    - Smells:
      * Feature Envy
      * Dispersed Coupling
      * Intensive Coupling
      * Refuse Parent Bequest
      * Shotgun Surgery
      * Long Method
      * God Class
  + Fontana et al., 2020, Code smell co-occurrences and architecture quality
    - Languages of systems:
      * Java
    - Domains of systems:
      * Qualitas Corpus
    - Smells:
      * God Class
      * Data Class
      * Brain Method
      * Shotgun Surgery
      * Dispersed Coupling
      * Message Chains
  + Ogenrwot et al., 2020, Code smell co-occurrence in desktop and mobile apps
    - Languages of systems:
      * Java
    - Domains of systems:
      * 7 Android apps
      * 5 Desktop systems
    - Design Smells:
      * Long Method
      * Complex Class
      * Long Parameter List
      * Base Class Should Be Abstract
      * Speculative Generality
      * Class Data Should Be Private
      * Many Field Attributes But Not Complex
      * Message Chain
      * Spaghetti Code
      * Refused Parent Bequest
      * Swiss Army Knife
      * Blob
      * Anti-Singleton
      * Large Class
      * Lazy Class
  + Muse et al., 2020, SQL code smell co-occurrences
    - Languages of systems:
      * SQL
      * Java
    - Platforms of systems:
      * Android
      * Desktop
    - Domains of systems:
      * Business
      * Library
      * Multimedia
      * Utility
    - Smells:
      * SQL:
        + Implicit Columns
        + Fear of the Unknown
        + Random Selection
        + Ambiguous Groups
      * Traditional (ones that co-occur with SQL code smells):
        + Long Method
        + Long Parameter List
        + Anti-Singleton
        + Complex Class
  + Bessghaier et al., 2021, Code smell co-occurrence in web apps
    - Languages of systems:
      * PHP
    - Domains:
      * Web applications:
        + Database Management System
        + Content Management System
        + Website Traffic Analysis
        + PHP Development Framework
        + E-Learning Management System
        + Bulletin Board Software
    - Smells:
      * Excessive Number of Children
      * Excessive Depth of Inheritance
      * High Coupling
      * Empty Catch Block
      * Goto Statement
      * High Method Complexity
      * High NPath Complexity
      * Excessive Method Length
      * Excessive Class Length
      * Excessive Parameter List
      * Too Many Public Methods
      * Too Many Methods
  + Martins et al., 2021, Code smell co-occurrence removal impact on quality attributes
    - Languages of systems:
      * Java
    - Domains of systems:
      * Management of Events
      * Extension and Research Projects
      * Risk Management
      * Employee Competency
      * Student Activities
    - Smells:
      * Feature Envy
      * God Class
      * Dispersed Coupling
      * Intensive Coupling
      * Shotgun Surgery
      * Long Method
  + Sobrinho et al., 2021, Inter-play of Large Class, Complex Class and Duplicate Code
    - Languages of systems:
      * Java
    - Domains of systems:
      * ArgoUML: UML Diagramming
      * Lucene: Text Search Engine Library
      * Cassandra: Database Management
      * Hadoop: Distributed Computing
      * Ant: Build Tool and Library
    - Smells:
      * Large Class
      * Complex Class
      * Duplicate Code
  + Hamdi et al., 2021, An Empirical Study on Code Smells Co-occurrences in Android Applications
    - Languages of systems:
      * Java
    - Domains of systems:
      * 1,923 open-source Android apps
    - Smells:
      * Object-Oriented:
        + Blob Class
        + Complex Class
        + Feature Envy
        + Lazy Class
        + Long Method
        + Long Parameter List
        + Message Chain
        + Refused Parent Bequest
        + Spaghetti Code
        + Speculative Generality
      * Android:
        + Data Transmission Without Compression
        + Debuggable Release
        + Durable Wakelock
        + Inefficient Data Format and Parser
        + Inefficient Data Structure
        + Inefficient SQL Query
        + Internal Getter and Setter
        + Leaking Inner Class
        + Leaking Thread
        + Member Ignoring Method
        + No Low Memory Resolver
        + Public Data
        + Rigid Alarm Manager
        + Slow Loop
        + Unclosed Closable
* RQ2) What are the measurements that researchers use to identify code smells?
  + Abbes et al., 2011, Blob and spaghetti antipatterns
    - (No measurements to detect smells, but measurements to measure understanding of code with/without smells)
  + Yamashita and Moonen, 2013, Inter-smell relations on software maintainability
    - (see Appendix here: <https://www.simula.no/sites/default/files/publications/Simula.simula.1494.pdf>)
  + Yamashita et al., 2015, Inter-smell relations in industrial and open-source systems
    - (None mentioned)
  + Palomba et al., 2016, Diffusion of test smells in auto-generated test code
    - (None mentioned)
  + Palomba et al., 2017, Code smell co-occurrences and association rule learning
    - (None mentioned)
  + Palomba et al., 2018, Code smell co-occurrence lifecycle
    - (None mentioned)
  + Martins et al., 2020, Code smell co-occurrence impact on quality attributes
    - (None mentioned for identification, but the following measurements/metrics were used to evaluate internal quality metrics before and after removing code smell co-occurrences):
      * Lack of Cohesion of Methods
      * Coupling Between Objects
      * Average Cyclomatic Complexity
      * Sum Cyclomatic Complexity
      * Nesting
      * Essential Complexity
      * Number of Children
      * Depth of Inheritance Tree
      * Number of Bases Classes
      * Lines of Code
      * Lines with Comments
      * Number of Classes
      * Number of Instance Methods
  + Fontana et al., 2020, Code smell co-occurrences and architecture quality
    - Lines of Code Without Accessor or Mutator Methods
    - Lines of Code
    - Weighted Methods Count of Not Accessor or Mutator Methods
    - Number of Not Accessor or Mutator Methods
    - Tight Class Cohesion
    - Access to Foreign Data
    - Weight of Class
    - Number of Accessor Methods
    - Number of Public Attributes
    - McCabe Cyclomatic Complexity
    - Maximum Nesting Level
    - Number of Local Variables
    - Access to Local Data
    - Changing Classes
    - Changing Methods
    - Number of Called Classes
    - Coupling Intensity
    - Coupling Dispersion
    - Maximum Message Chain Length
    - Mean Message Chain Length
    - Number of Message Chain Statements
  + Ogenrwot et al., 2020, Code smell co-occurrence in desktop and mobile apps
    - (None mentioned)
  + Muse et al., 2020, SQL code smell co-occurrences
    - (None mentioned)
  + Bessghaier et al., 2021, Code smell co-occurrence in web apps
    - (none mentioned for identification; LOC used only for calculating code smell diffusion [i.e. number of smells compared to size of projects])
  + Martins et al., 2021, Code smell co-occurrence removal impact on quality attributes
    - (none mentioned for identification; the following were used only for measuring internal quality before and after removing co-occurrences):
      * Cohesion
        + Lack of Cohesion of Methods
      * Coupling
        + Coupling Between Objects
        + Fan-in
        + Fan-out
      * Complexity
        + Weighted Method Count
        + Sum Cyclomatic Complexity
        + Nesting
        + Essential Complexity
      * Inheritance
        + Number of Children
        + Depth of Inheritance Tree
        + Bases Classes
  + Sobrinho et al., 2021, Inter-play of Large Class, Complex Class and Duplicate Code
    - Cyclomatic Complexity for Complex Class
    - Number of Methods Declared for Large Class
    - The Number of Tokens of Clone for Duplicate Code
  + Hamdi et al., 2021, An Empirical Study on Code Smells Co-occurrences in Android Applications
    - (none mentioned)
* RQ3) What are the tools used to identify code smells?
  + Abbes et al., 2011, Blob and spaghetti antipatterns
    - DECOR
  + Yamashita and Moonen, 2013, Inter-smell relations on software maintainability
    - Borland Together
    - inCode
  + Yamashita et al., 2015, Inter-smell relations in industrial and open-source systems
    - inFusion
    - depFinder (used for dependency analysis to transform resulting graphs)
    - Gephi (“used for visualizing inter-smells to guide code examination”)
  + Palomba et al., 2016, Diffusion of test smells in auto-generated test code
    - Test Smell Detector
  + Palomba et al., 2017, Code smell co-occurrences and association rule learning
    - Semi-manual detection with tool used to discard classes/methods that did not contain code smells
  + Palomba et al., 2018, Code smell co-occurrence lifecycle
    - Semi-manual detection with tool used to discard classes/methods that did not contain code smells
  + Martins et al., 2020, Code smell co-occurrence impact on quality attributes
    - JSpIRIT (semi-automatic approach)
    - JDeodorant
    - Understand tool (used to measure software internal quality attributes after removing co-occurring code smells)
  + Fontana et al., 2020, Code smell co-occurrences and architecture quality
    - Tool utilizing detection rules defined in study using computations from 43 metrics
  + Ogenrwot et al., 2020, Code smell co-occurrence in desktop and mobile apps
    - Pattern Trace Identification, Detection, and Enhancement in Java (Ptidej)
      * Reverse engineering tool including identification algorithms for idioms, micro-patterns, design patterns and design defects
  + Muse et al., 2020, SQL code smell co-occurrences
    - SQL code smell detection:
      * SQLInspect
    - Traditional code smell detection (ones that co-occur with SQL code smells):
      * DECOR
  + Bessghaier et al., 2021, Code smell co-occurrence in web apps
    - PHPMD
  + Martins et al., 2021, Code smell co-occurrence removal impact on quality attributes
    - Understand tool (used to determine number of classes, number of methods and LOC for each project)
    - JDeodorant
    - JSpIRIT
  + Sobrinho et al., 2021, Inter-play of Large Class, Complex Class and Duplicate Code
    - PMD
    - DECOR
  + Hamdi et al., 2021, An Empirical Study on Code Smells Co-occurrences in Android Applications
    - Android-specific smells: aDoctor
    - Object-oriented smells: Organic[[1]](#footnote-0)
* RQ4) What are the algorithms and techniques used to identify code smells?
  + Abbes et al., 2011, Blob and spaghetti antipatterns
    - DEX
  + Yamashita and Moonen, 2013, Inter-smell relations on software maintainability
    - Principal Component Analysis (PCA), using orthogonal rotation (varimax)
    - Factor-Strategy\*
  + Yamashita et al., 2015, Inter-smell relations in industrial and open-source systems
    - PCA, using orthogonal rotation (varimax)
  + Palomba et al., 2016, Diffusion of test smells in auto-generated test code
    - Heuristic metric-based technique
  + Palomba et al., 2017, Code smell co-occurrences and association rule learning
    - Association rule learning
  + Palomba et al., 2018, Code smell co-occurrence lifecycle
    - Manual identification and validation
  + Martins et al., 2020, Code smell co-occurrence impact on quality attributes
    - Individual occurrences of smells detected by automatic tools
    - Co-occurrences of smells identified manually
  + Fontana et al., 2020, Code smell co-occurrences and architecture quality
    - Six code smell detection rules defined based on a computation of 43 metrics
  + Ogenrwot et al., 2020, Code smell co-occurrence in desktop and mobile apps
    - Powered Outer Probabilistic Clustering (POPC)
      * POPC’s output is used to group design smells based on their occurrence
  + Muse et al., 2020, SQL code smell co-occurrences
    - Apriori (for mining code smell co-occurrences)
    - Cramer's V association test (to see if associations between traditional and SQL code smells are statistically significant)
    - Chi-squared test (to see if associations between traditional and SQL code smells are statistically significant)
  + Bessghaier et al., 2021, Code smell co-occurrence in web apps
    - Apriori association rules-mining
    - Chi-squared (to calculate degree of association between code smells)
    - Cramer's V (to calculate degree of association between code smells)
  + Martins et al., 2021, Code smell co-occurrence removal impact on quality attributes
    - Manual:
      * A smell at the class-level and a smell at the method-level of the same class are considered to co-occur
      * Two or more smells affecting a method are considered to co-occur
  + Sobrinho et al., 2021, Inter-play of Large Class, Complex Class and Duplicate Code
    - Logistic Regression Model
    - Odds Ratio
  + Hamdi et al., 2021, An Empirical Study on Code Smells Co-occurrences in Android Applications
    - Apriori
* RQ5) What is the comparison between the results from different studies?
  + Abbes et al., 2011, Blob and spaghetti antipatterns
    - More difficult to understand code when Blob and Spaghetti Code co-occur than when each smell occurs by itself or not at all
  + Yamashita and Moonen, 2013, Inter-smell relations on software maintainability
    - time-consuming changes, changes more prone to defects with Feature Envy and God Method smells
    - changes harder to understand due to ISP Violation and Shotgun Surgery code smells
    - faults that were introduced into one component affected other components due to ISP Violation
    - Data Classes had incoming dependencies from methods with Feature Envy
    - Implementation Instead of Interface not considered problematic
    - maintenance problems not associated with presence of Refused Bequest or absence of Misplaced Class
    - God Method, God Class and Feature Envy have large size and high internal complexity, makes it difficult to understand and change code, makes changes more prone to defects (nicknamed hoarders)
    - Temporal variable used for several purposes and Duplicated code in conditional branches indicate more than one ambiguous context, makes it difficult to understand and change code, makes changes more prone to defects (nicknamed confounders)
    - Shotgun Surgery and ISP Violation can cause unexpected side effects when making changes due to coupling
  + Yamashita et al., 2015, Inter-smell relations in industrial and open-source systems
    - Distinguishes the difference between collocated and coupled code smells:
      * Collocated code smells: two or more smells detected in the same class
      * Coupled code smells: two or more smells detected in artifacts (classes) that are coupled
    - God Class, Feature Envy and Intensive Coupling occur together
      * Validates the following inter-relation findings:
        + Large Class and Feature Envy
        + Large Class and Low Cohesion
        + God Class, Dispersed Coupling and Intensive Coupling
        + God Class and Feature Envy
        + Methods with Feature Envy have Intensive Coupling
        + God Class, Feature Envy, Shotgun Surgery and ISP Violation
      * Occurred in systems from different domains
    - Classes with Data Class did not also have Data Clumps
      * Contradicts other finding that these smells co-occur
        + Reason for difference could be different tools used to detect smells
    - Duplicate code smells (i.e. External Duplication, Internal Duplication) tend to appear with smells related to size and complexity (e.g. Blob Operation, Schizo Class, God Class)
      * Related to findings that “Temporal variable used for several purposes” and “Duplicate code in conditional branches” found together with smells related to size
      * Duplicate code smells may cause system size to increase and lead to code smells related to size
    - Feature Envy and Data Class appeared together
    - Redundant components (i.e. classes with a smell that have incoming or outgoing dependencies to a class with the same smell):
      * Intensive Coupling
      * Message Chains
      * External Duplication
      * Data Clumps
      * Blob Operation
      * Sibling Duplication
      * Refused Parent Bequest
    - Patterns of coupled smells across three systems:
      * Feature Envy to Data Class
      * Data Class and Feature Envy to Class
      * Class to Feature Envy and Message Chains
      * God Class and Schizo Class to Class
      * Internal Duplication and External Duplication to Class
      * Class to Internal Duplication and External Duplication
      * Message Chains to Sibling Duplication
      * Message Chains and Sibling Duplication to Class
      * God Class, Data Class and Message Chains to Class
      * Sibling Duplication to Sibling Duplication
      * Feature Envy and God Class to Class
      * Data Clumps to Data Clumps
    - Inter-smell dependency graph generated to provide a visual aid to guide code review
  + Palomba et al., 2016, Diffusion of test smells in auto-generated test code
    - For Testers Only not reported as code smell co-occurrence
    - All test smells co-occur frequently with Assertion Roulette except For Testers Only
    - Other frequent test smell co-occurrences:
      * Mystery Guest and Resource Optimism
      * Mystery Guest and Indirect Testing
      * Mystery Guest and Eager Test
      * Test Code Duplication and Indirect Testing
      * Test Code Duplication and Eager Test
      * Test Code Duplication and Mystery Guest
      * Test Code Duplication and Sensitive Equality
      * Eager Test and Resource Optimism
      * Eager Test and Indirect Testing
  + Palomba et al., 2017, Code smell co-occurrences and association rule learning
    - Code smell co-occurrences:
      * Message Chains and Complex Class
      * Message Chains and Refused Bequest
      * Long Method and Long Parameter List
      * Long Method and Feature Envy
      * Long Method and Spaghetti Code
      * Inappropriate Intimacy and Feature Envy
  + Palomba et al., 2018, Code smell co-occurrence lifecycle
    - Almost 59% of smelly classes contain more than one kind of code smell instance
    - Code smell co-occurrences:
      * Message Chains and Spaghetti Code
      * Message Chains and Complex Class
      * Message Chains and Blob
      * Message Chains and Refused Bequest
      * Long Method and Feature Envy
      * Long Method and Spaghetti Code
    - Code smells affecting single methods tend to trigger the introduction of class-level
    - Code smells co-occurrences are generally removed together during maintenance and evolution activities
  + Martins et al., 2020, Code smell co-occurrence impact on quality attributes
    - Code smell co-occurrences that tend to occur the most:
      * God Class and Long Method
      * Dispersed Coupling and Long Method
      * Feature Envy and Long Method
      * Dispersed Coupling and Feature Envy
      * Feature Envy and God Class
    - Code smell co-occurrences tend to increase during the evolution of a software project
    - Removing code smell co-occurrences had the following impacts:
      * Negative impacts:
        + Cohesion decreased
        + Coupling increased
      * Positive impacts
        + Complexity decreased
    - The following code smell co-occurrences were the most difficult for developers to remove:
      * God Class and Long Method
      * Dispersed Coupling and Long Method
      * Also difficult:
        + Shotgun Surgery
        + Intensive Coupling
  + Fontana et al., 2020, Code smell co-occurrences and architecture quality
    - Code smell relations:
      * “26% of God Classes access data from Data Classes”
      * 58% “of entities with God Class are called by at least one class affected by code smells”
      * 57% of “God Classes have at least one method affected by at least one of the method code smells”
      * “38% of Data Classes are used by other classes affected by code smells”
        + God class affects most
      * 9.85% of Brain Methods affected by other code smells
      * “16% of Brain Methods are in classes affected by code smells (mainly God Class)”
      * 53% of the classes calling a Shotgun Surgery method are affected by at least one code smell
      * 30% of classes that contain at least one Shotgun Surgery method are affected by a code smell
      * 11% of Shotgun Surgery methods are affected by other code smells
      * 70% of “classes called by Dispersed Coupling methods are affected by code smells”
      * A Dispersed Coupling method calls many other classes with code smells
      * 17% of classes with a Dispersed Coupling method are affected by code smells
      * 8% of Dispersed Coupling methods are affected by code smells
      * 27% of Message Chain methods are affected by at least one other code smell
      * 21% of classes that contain Message Chain methods are affected by code smells
    - Code smell co-occurrences:
      * Less than 10% of methods with code smells were affected by at least one other code smell
      * Brain Method tends to co-occur with other code smells the most
  + Ogenrwot et al., 2020, Code smell co-occurrence in desktop and mobile apps
    - Design smells that occurred the most together:
      * Speculative Generality and Swiss Army Knife
      * Long Parameter List and Class Data Should Be Private
  + Muse et al., 2020, SQL code smell co-occurrences
    - some SQL code smells have statistically significant co-occurrence with traditional code smells, but the association between the smells are not strong
    - Implicit Columns smells are most prevalent SQL code smell, followed by Fear of the Unknown
    - Implicit Columns and Long Method co-occur in all subject systems
    - Implicit Columns and Complex Class co-occur in 3 out of 4 application domains
    - statistically significant associations between SQL and traditional code smells:
      * Implicit Columns and Long Parameter List
      * Implicit Columns and Complex Class
      * Implicit Columns and Anti-Singleton
      * Fear of the Unknown and Long Method
      * Fear of the Unknown and Long Parameter List
      * Fear of the Unknown and Complex Class
      * Fear of the Unknown and Anti-Singleton
  + Bessghaier et al., 2021, Code smell co-occurrence in web apps
    - 39% of fragments with code smells have only one type of smell
    - 61% of fragments with code smells have two or more types of smells
    - Method-level code smells could cause class-level smells
    - Code smells related to complexity and size of code components, like Excessive Class Length and High Method Complexity, have a high degree of association for co-occurrence, but other code smells have a weak degree of association for co-occurrence
    - Code smell co-occurrences:
      * High Method Complexity and Excessive Number of Children
      * High Method Complexity and High Coupling
      * High Method Complexity and Empty Catch Block
      * High Method Complexity and Goto Statement
      * High Method Complexity and High NPath Complexity
      * High Method Complexity and Excessive Method Length
      * High Method Complexity and Excessive Class Length
      * High Method Complexity and Excessive Parameter List
      * High Method Complexity and Too Many Public Methods
      * High Method Complexity and Too Many Methods
      * High NPath Complexity and Goto Statement
      * High NPath Complexity and Too Many Methods
      * Too Many Public Methods and Too Many Methods
      * Excessive Parameter List and Too Many Methods
      * Excessive Parameter List and High Coupling
      * Excessive Parameter List and High NPath Complexity
      * Excessive Class Length and Excessive Method Length
      * Excessive Class Length and Too Many Methods
      * Excessive Class Length and Too Many Public Methods
      * Excessive Method Length and Too Many Public Methods
  + Martins et al., 2021, Code smell co-occurrence removal impact on quality attributes
    - The removal of the following code smell co-occurrences improved all four internal quality attributes, suggesting that these co-occurrences are harmful to software quality:
      * Dispersed Coupling and God Class
      * God Class and Long Method
    - The removal of the following code smell co-occurrences improved cohesion, complexity and coupling:
      * Feature Envy and God Class
      * God Class and Shotgun Surgery
    - The removal of the following code smell co-occurrence had a negative impact on cohesion, complexity and coupling:
      * Feature Envy and Long Method
    - Most harmful code smell co-occurrences according to developers’ perceptions:
      * Feature Envy and God Class
      * Dispersed Coupling and God Class
      * God Class and Long Method
    - Code smell co-occurrences that took the longest time to remove:
      * Intensive Coupling and Long Method
      * Dispersed Coupling and God Class
      * Feature Envy and God Class
      * God Class and Long Method
  + Sobrinho et al., 2021, Inter-play of Large Class, Complex Class and Duplicate Code
    - Prevalence of clones in classes that have only Complex Class does not statistically significantly differ from prevalence of clones in classes where both Large Class and Complex Class co-occur
    - Prevalence of clones in classes that have only Large Class is more related than the prevalence of classes in classes that have both Large Class and Complex Class, but there are other additional factors that explain the prevalence of clones
    - When the smells Large Class and Complex Class co-occur, the prevalence of clones is higher in classes with high intensity of Complex Class than in classes with low intensity of Complex Class, but there are other additional factors that explain the prevalence of clones
  + Hamdi et al., 2021, An Empirical Study on Code Smells Co-occurrences in Android Applications
    - Feature Envy and Long Method
    - Long Method and Long Parameter List
    - Complex Class and Message Chain
    - Complex Class and Feature Envy
    - Long Parameter List and Feature Envy
    - No Low Memory Resolver and Member Ignoring Method
    - Leaking Inner Class and Member Ignoring Method
    - Message Chain and Member Ignoring Method
    - Slow Loop and Member Ignoring Method
    - Complex Class and Member Ignoring Method
    - Long Method and Member Ignoring Method
    - Long Parameter List and Member Ignoring Method

/\*\* NOT INCLUDING \*\*/

* RQ1)
  + Alfadel et al., 2020, Design patterns and code smells
    - Languages of systems:
      * Java
    - Domains of systems:
      * 10 open-source small-to-medium-sized projects
    - Design patterns used to determine correlation with code smells:
      * Creational
        + Abstract Factory
        + Builder
        + Factory Method
        + Prototype
        + Singleton
      * Structural
        + Adapter
        + Bridge
        + Composite
        + Facade
        + Decorator
        + Proxy
      * Behavioral
        + Command
        + Iterator
        + Mediator
        + Memento
        + Observer
        + State
        + Strategy
        + Template Method
        + Visitor
    - Smells:
      * Data Class
      * Data Clumps
      * Refused Parent Bequest
      * Schizophrenic Class
      * Blob Methods
      * Intensive Coupling
      * Sibling Duplication
      * Internal Duplication
      * External Duplication
      * God Class
      * Feature Envy
      * Tradition Breaker
      * Message Chains
  + Tan et al., 2020, Co-occurring technical debt in Python
    - Languages of systems:
      * Python
    - Domains of systems:
      * 20 Python projects from Apache Software Foundation
    - Smells/Technical Debt:
      * Design
      * Code
      * Defect
      * Documentation
      * Test Debt
* RQ2) What are the measurements that researchers use to identify code smells?
  + Tan et al., 2020, Co-occurring technical debt in Python
    - (None mentioned)
* RQ3) What are the tools used to identify code smells?
  + Tan et al., 2020, Co-occurring technical debt in Python
    - SonarQube
* RQ4) What are the algorithms and techniques used to identify code smells?
  + Tan et al., 2020, Co-occurring technical debt in Python
    - Association rule mining, based on the machine learning algorithm Apriori
* RQ5) What is the comparison between the results from different studies?
  + Tan et al., 2020, Co-occurring technical debt in Python
    - More than 90% of Python files have more than one instance of co-occurring technical debt
    - Design Debt items are more likely to co-occur with other similar items
    - Documentation Debt and Test Debt appear in the majority of files
    - Test Debt items co-occur the most frequently with other types
    - Co-occurring technical debt items are usually introduced within the same week
    - One of two co-occurring technical debt items remain for a long time after the other one is removed, except Design Debt items

/\*\* NOT INCLUDING \*\*/

1. https://github.com/opus-research/organic [↑](#footnote-ref-0)