Design Pattern Decay

A Study of Design Pattern Grime and Its Impact on Quality and Technical Debt

Isaac D. Griffith

MONTANA STATE UNIVERSITY

Gianforte School of Computing

Technical Debt

- 1992 - Ward Cunningham coined the term **Technical Debt** describing a financial metaphor explaining the need to refactor to stakeholders

Technical Debt

- 1992 - Ward Cunningham coined the term **Technical Debt** describing a financial metaphor explaining the need to refactor to stakeholders

... 20 years later ...

Technical Debt

- 1992 - Ward Cunningham coined the term **Technical Debt** describing a financial metaphor explaining the need to refactor to stakeholders

... 20 years later ...

2012 - CAST Research Group Findings

In an analysis of 745 applications comprising 365 MLOC, on average, there is **$3.61 of Technical Debt per LOC**

Technical Debt

Definition

"In software intensive systems, technical debt–is a design or implementation construct that is expedient in the short term, but sets up a technical context that can make a future change more costly or impossible. Technical debt is a contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability"[a]

[a] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman, Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162), Dagstuhl Reports 6 (2016), no. 4, 110–138

Technical Debt

Definition

"In software intensive systems, technical debt–is a design or implementation construct that is expedient in the short term, but sets up a technical context that can make a future change more costly or impossible. Technical debt is a contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability"[a]

[a] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman, Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162), Dagstuhl Reports 6 (2016), no. 4, 110–138

Properties

- **Principal** - The effort required to refactor the software and remove the debt.

Technical Debt

Definition

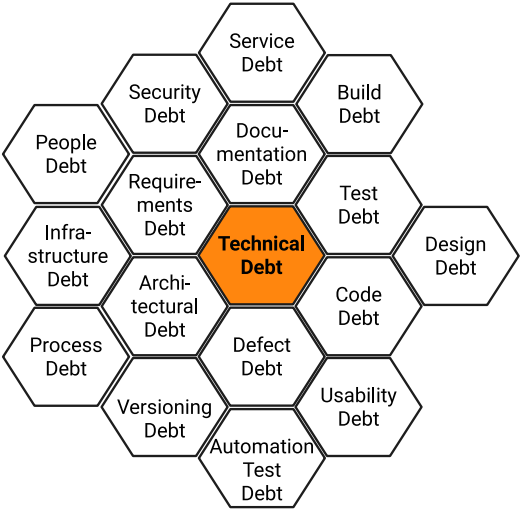
"In software intensive systems, technical debt–is a design or implementation construct that is expedient in the short term, but sets up a technical context that can make a future change more costly or impossible. Technical debt is a contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability"[a]

[a] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman, Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162), Dagstuhl Reports 6 (2016), no. 4, 110–138

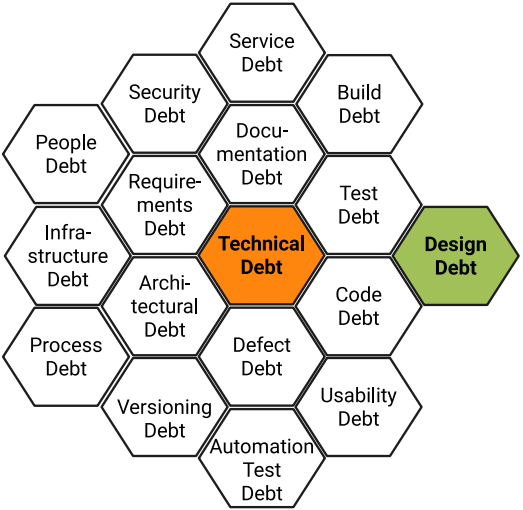
Properties

- **Principal** - The effort required to refactor the software and remove the debt.
- **Interest** - The increase in development effort caused by debt affected components which have yet to be refactored.

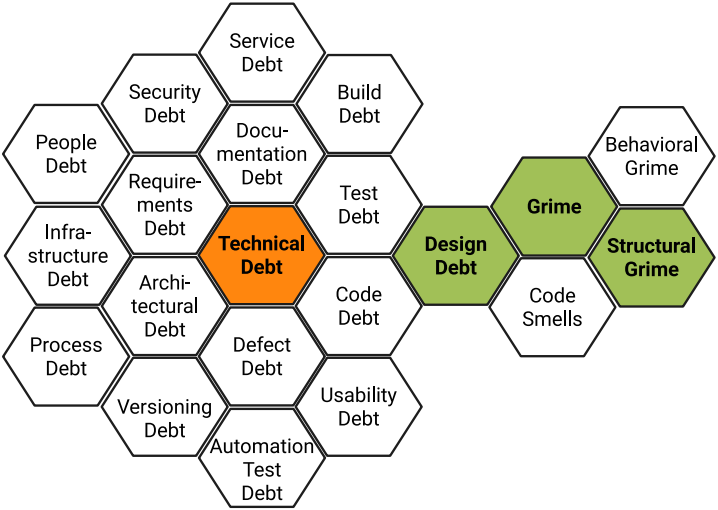
Technical Debt



Technical Debt



Technical Debt



Design Pattern Grime

Definition

The accumulation of unnecessary or unrelated software artifacts within the classes of a design pattern instance.

Design Pattern Grime

Definition

The accumulation of unnecessary or unrelated software artifacts within the classes of a design pattern instance.

- **Structural Grime** - Izurieta and Bieman

Design Pattern Grime

Definition

The accumulation of unnecessary or unrelated software artifacts within the classes of a design pattern instance.

- **Structural Grime** - Izurieta and Bieman
  - **Modular Structural Grime** - build-up of relationships involving one or more pattern classes that are unnecessary to facilitate the operation of a pattern

Design Pattern Grime

Definition

The accumulation of unnecessary or unrelated software artifacts within the classes of a design pattern instance.

- **Structural Grime** - Izurieta and Bieman
  - **Modular Structural Grime** - build-up of relationships involving one or more pattern classes that are unnecessary to facilitate the operation of a pattern
  - **Class Structural Grime** - build-up of fields/methods within pattern classes that are unnecessary to facilitate the operation of a pattern

Design Pattern Grime

Definition

The accumulation of unnecessary or unrelated software artifacts within the classes of a design pattern instance.

- **Structural Grime** - Izurieta and Bieman
  - **Modular Structural Grime** - build-up of relationships involving one or more pattern classes that are unnecessary to facilitate the operation of a pattern
  - **Class Structural Grime** - build-up of fields/methods within pattern classes that are unnecessary to facilitate the operation of a pattern
  - **Organizational Structural Grime** - the unnecessary distribution of pattern instance classes across namespaces or packages

Design Pattern Grime

Definition

The accumulation of unnecessary or unrelated software artifacts within the classes of a design pattern instance.

- **Structural Grime** - Izurieta and Bieman
  - **Modular Structural Grime** - build-up of relationships involving one or more pattern classes that are unnecessary to facilitate the operation of a pattern
  - **Class Structural Grime** - build-up of fields/methods within pattern classes that are unnecessary to facilitate the operation of a pattern
  - **Organizational Structural Grime** - the unnecessary distribution of pattern instance classes across namespaces or packages
- **Behavioral Grime** - Reimanis and Izurieta

Design Pattern Grime

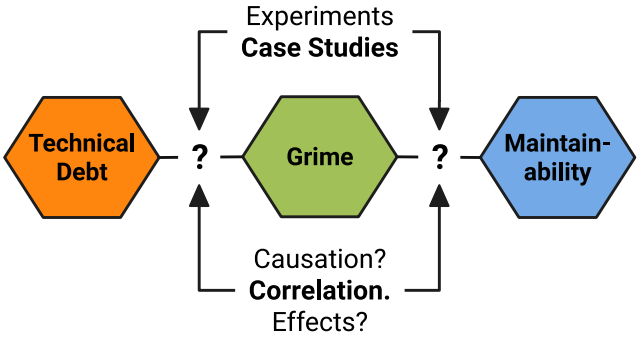
Definition

The accumulation of unnecessary or unrelated software artifacts within the classes of a design pattern instance.

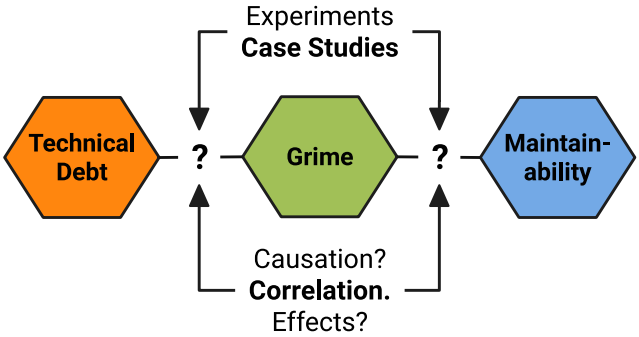
- **Structural Grime** - Izurieta and Bieman
  - **Modular Structural Grime** - build-up of relationships involving one or more pattern classes that are unnecessary to facilitate the operation of a pattern
  - **Class Structural Grime** - build-up of fields/methods within pattern classes that are unnecessary to facilitate the operation of a pattern
  - **Organizational Structural Grime** - the unnecessary distribution of pattern instance classes across namespaces or packages
- **Behavioral Grime** - Reimanis and Izurieta
  - Buildup of errant behavioral elements within a pattern instance (i.e. *Excessive Actions* or *Improper Order of Sequences*)

Problem Statement

Problem Statement



Problem Statement



What is the nature of the relationships between structural design pattern grime, software maintainability, and technical debt measurement?

Design Science

An approach whereby stakeholders, serving human purposes, create or improve ”things” within technological solutions

Design Science

An approach whereby stakeholders, serving human purposes, create or improve ”things” within technological solutions

Design Science Components:

- **Design Problems:** “call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals”

Design Science

An approach whereby stakeholders, serving human purposes, create or improve ”things” within technological solutions

Design Science Components:

- **Design Problems:** “call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals”
  - Result in the creation of an artifact

Design Science

An approach whereby stakeholders, serving human purposes, create or improve ”things” within technological solutions

Design Science Components:

- **Design Problems:** “call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals”
  - Result in the creation of an artifact
  - Addressed by following a *design cycle*

Design Science

An approach whereby stakeholders, serving human purposes, create or improve ”things” within technological solutions

Design Science Components:

- **Design Problems:** “call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals”
  - Result in the creation of an artifact
  - Addressed by following a *design cycle*
- **Knowledge Questions:** “do not call for a change but ask for knowledge about the world as it is”

Design Science

An approach whereby stakeholders, serving human purposes, create or improve ”things” within technological solutions

Design Science Components:

- **Design Problems:** “call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals”
  - Result in the creation of an artifact
  - Addressed by following a *design cycle*
- **Knowledge Questions:** “do not call for a change but ask for knowledge about the world as it is”
  - Result in a proposition

Design Science

An approach whereby stakeholders, serving human purposes, create or improve ”things” within technological solutions

Design Science Components:

- **Design Problems:** “call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals”
  - Result in the creation of an artifact
  - Addressed by following a *design cycle*
- **Knowledge Questions:** “do not call for a change but ask for knowledge about the world as it is”
  - Result in a proposition
  - Addressed by following an *empirical cycle*

Phase 01 - Extending Grime Taxonomies

RG1: Analyze design patterns to elaborate on the complete taxonomy of Class and Organizational Grime

Phase 01 - Extending Grime Taxonomies

RG1: Analyze design patterns to elaborate on the complete taxonomy of Class and Organizational Grime

Research Questions

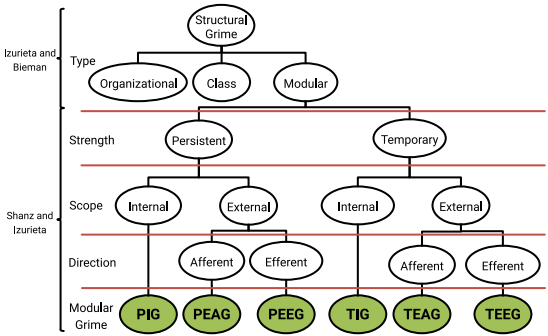
- **RQ1.1:** What are the types of Class Grime?
- **RQ1.2:** What are the types of Organizational Grime?

Phase 01 - Extending Grime Taxonomies

RG1: Analyze design patterns to elaborate on the complete taxonomy of Class and Organizational Grime

Research Questions

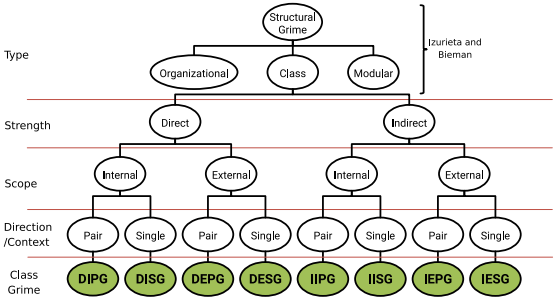
- **RQ1.1:** What are the types of Class Grime?
- **RQ1.2:** What are the types of Organizational Grime?



Class Grime

RQ1.1: What are the types of Class Grime?

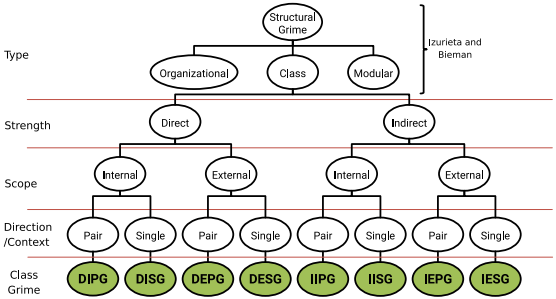
Taxonomy:



Class Grime

RQ1.1: What are the types of Class Grime?

Taxonomy:



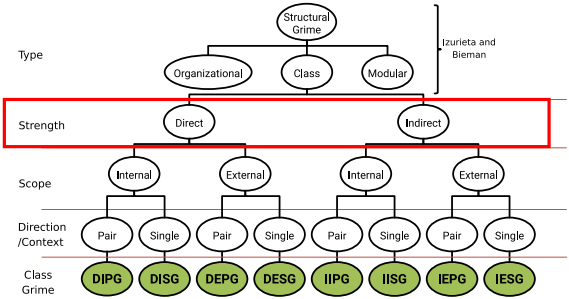
Basis:

- **Design Principles:** *SRP* and *HC*

Class Grime

RQ1.1: What are the types of Class Grime?

Taxonomy:



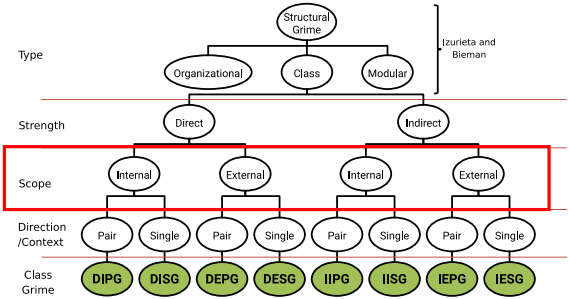
Basis:

- **Design Principles:** *SRP* and *HC*
- **Strength:** Attribute use by Method(s)

Class Grime

RQ1.1: What are the types of Class Grime?

Taxonomy:



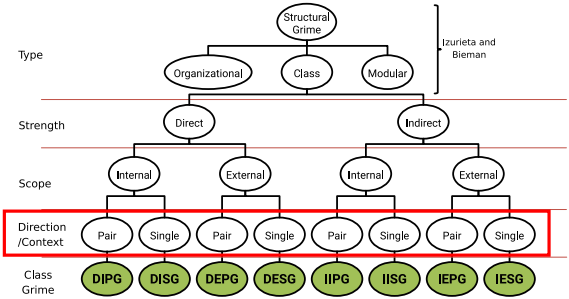
Basis:

- **Design Principles:** *SRP* and *HC*
- **Strength:** Attribute use by Method(s)
- **Scope:** Method(s) fulfill a pattern role or not

Class Grime

RQ1.1: What are the types of Class Grime?

Taxonomy:



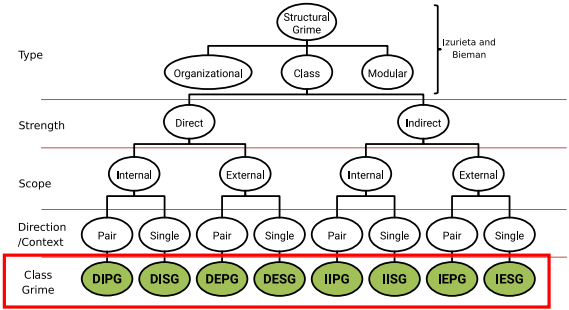
Basis:

- **Design Principles:** *SRP* and *HC*
- **Strength:** Attribute use by Method(s)
- **Scope:** Method(s) fulfill a pattern role or not
- **Context:** Defined by the cohesion metric used

Class Grime

RQ1.1: What are the types of Class Grime?

Taxonomy:



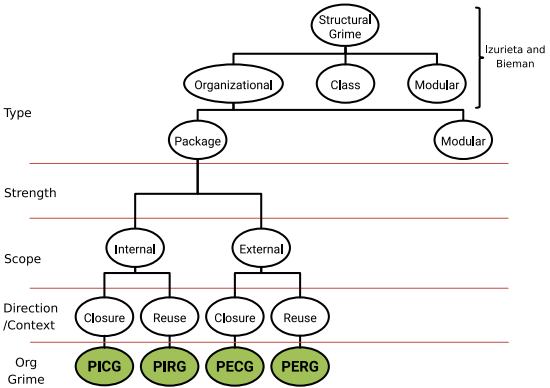
Basis:

- **Design Principles:** *SRP* and *HC*
- **Strength:** Attribute use by Method(s)
- **Scope:** Method(s) fulfill a pattern role or not
- **Context:** Defined by the cohesion metric used
  - Pair: Tight Class Cohesion (TCC)
  - Single: Ratio of Cohesive Interactions (RCI)

Package Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:

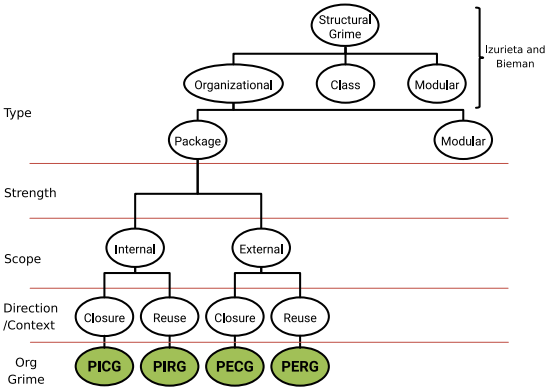


Basis:

Package Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



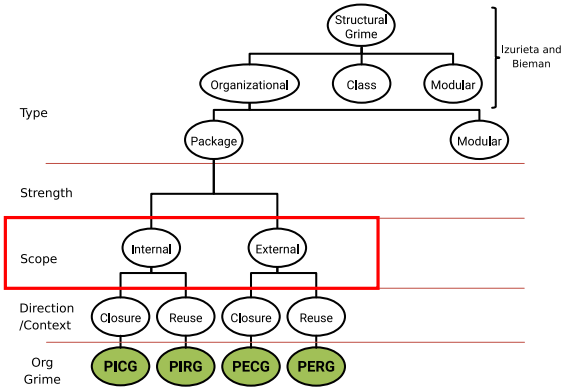
Basis:

- **Principles:** CCP and CRP

Package Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



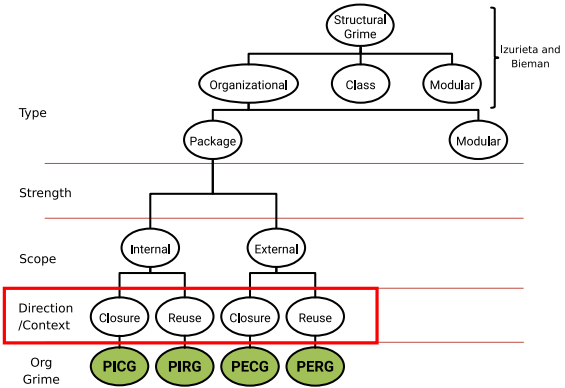
Basis:

- **Principles:** CCP and CRP
- **Scope:** Whether a class in a package is a member of the pattern

Package Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



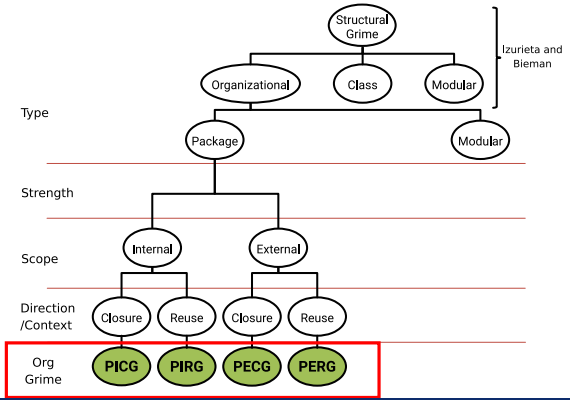
Basis:

- **Principles:** CCP and CRP
- **Scope:** Whether a class in a package is a member of the pattern
- **Context:** Considers how well a class fits within the package

Package Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



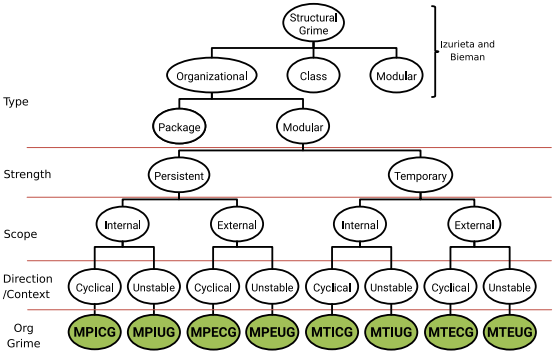
Basis:

- **Principles:** CCP and CRP
- **Scope:** Whether a class in a package is a member of the pattern
- **Context:** Considers how well a class fits within the package
  - Closure - as measured by $CohesionQ$
  - Reuse - as measured by $CouplingQ$

Modular Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:

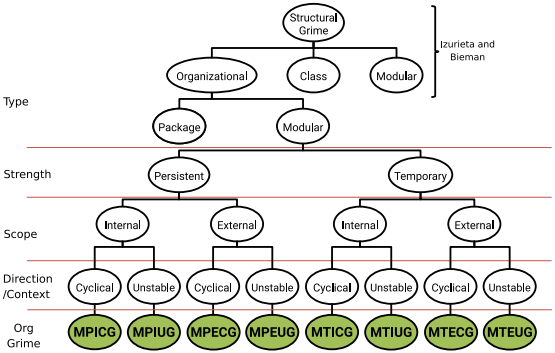


Basis:

Modular Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



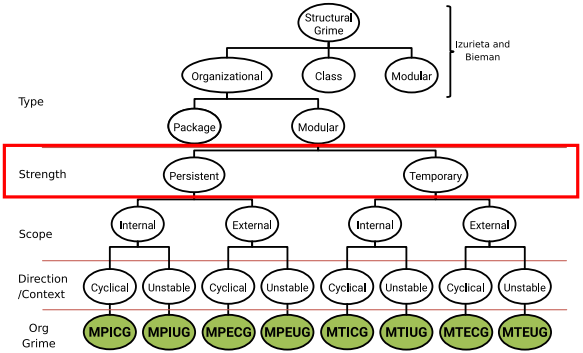
Basis:

- **Principles:** *ADP* and *SDP*

Modular Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



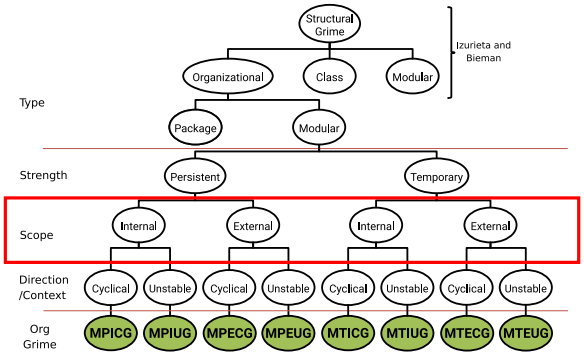
Basis:

- **Principles:** *ADP* and *SDP*
- **Strength:** Type of coupling underlying package dependencies

Modular Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



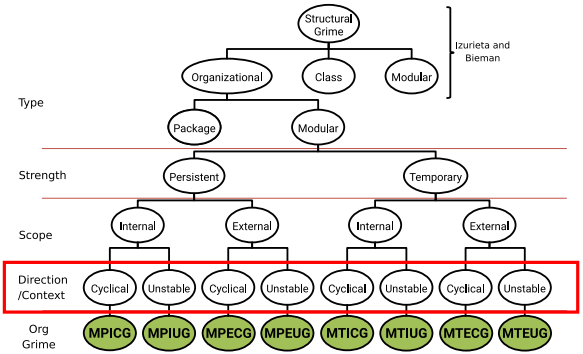
Basis:

- **Principles:** *ADP* and *SDP*
- **Strength:** Type of coupling underlying package dependencies
- **Scope:** Considers class dependency forming class membership within the pattern instance

Modular Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



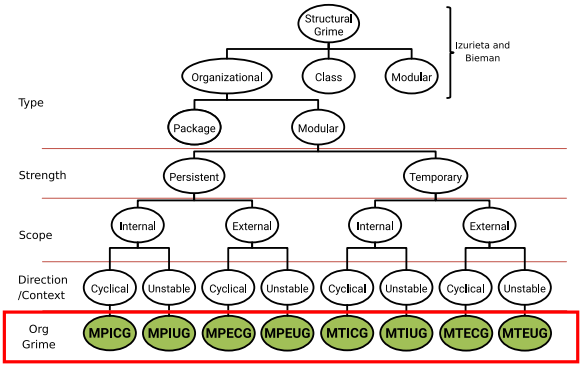
Basis:

- **Principles:** *ADP* and *SDP*
- **Strength:** Type of coupling underlying package dependencies
- **Scope:** Considers class dependency forming class membership within the pattern instance
- **Context:** Does the dependency create a cycle or reduce stability

Modular Organizational Grime

RQ1.2: What are the types of Organizational Grime?

Taxonomy:



Basis:

- **Principles:** *ADP* and *SDP*
- **Strength:** Type of coupling underlying package dependencies
- **Scope:** Considers class dependency forming class membership within the pattern instance
- **Context:** Does the dependency create a cycle or reduce stability

Phase 02 - Simulation Experiments

RG2: Analyze design pattern instances afflicted with **design pattern grime** for the purpose of evaluation with respect to the **ISO/IEC 25010 Maintainability sub-characteristics**, from the perspective of researchers, in the context of generated Java™ design pattern instances.

Phase 02 - Simulation Experiments

RG2: Analyze design pattern instances afflicted with **design pattern grime** for the purpose of evaluation with respect to the **ISO/IEC 25010 Maintainability sub-characteristics**, from the perspective of researchers, in the context of generated Java™ design pattern instances.

RG3: Analyze design pattern instances afflicted with **design pattern grime** for the purpose of evaluation with respect to the **Technical Debt Principal and Interest**, from the perspective of researchers, in the context of generated Java™ design pattern instances.

Phase 02 - Simulation Experiments

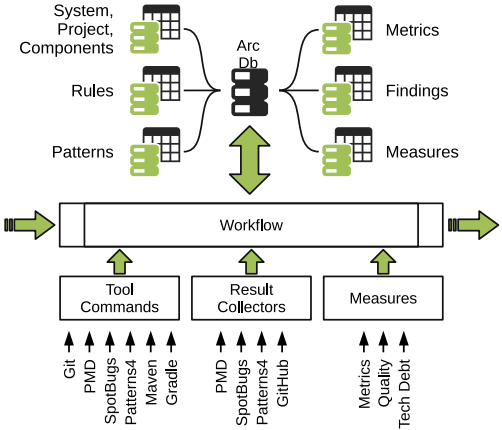
RG2: Analyze design pattern instances afflicted with **design pattern grime** for the purpose of evaluation with respect to the **ISO/IEC 25010 Maintainability sub-characteristics**, from the perspective of researchers, in the context of generated Java™ design pattern instances.

RG3: Analyze design pattern instances afflicted with **design pattern grime** for the purpose of evaluation with respect to the **Technical Debt Principal and Interest**, from the perspective of researchers, in the context of generated Java™ design pattern instances.

DP1: How can we conduct a full-factorial experiment to evaluate the relationship between grime and quality and grime and technical debt when we are unsure how often design pattern grime occurs?

The Arc Framework

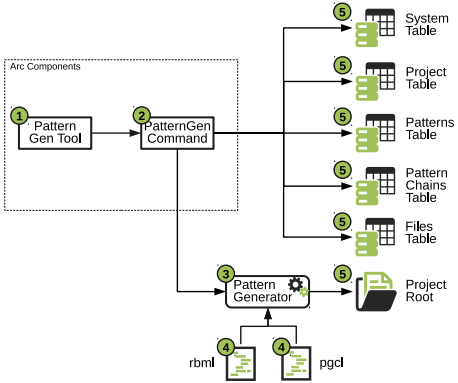
DP1.1 How do we automate the data collection and experimentation process?



Pattern Generation and Grime Injection

DP1.2 How can we generate design pattern instances?

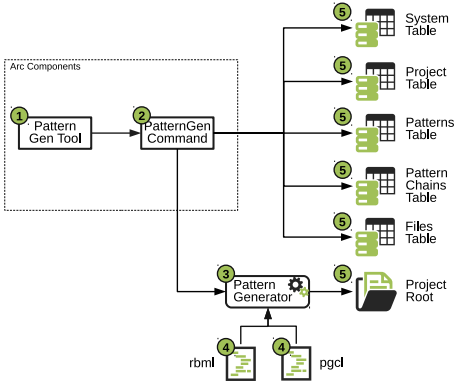
Pattern Generation Integration:



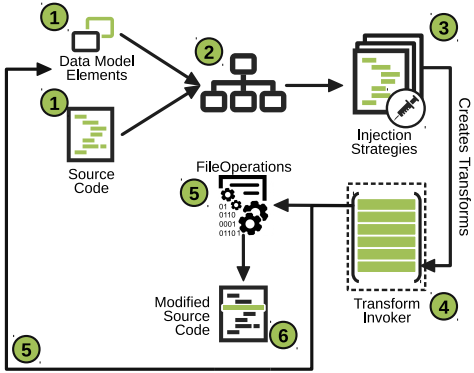
Pattern Generation and Grime Injection

DP1.3 How can we ensure design patterns have the correct type of grime?

Pattern Generation Integration:



Grime Injection Operation:

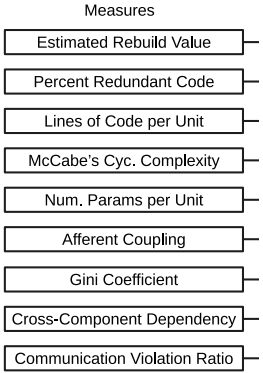


Measuring Software Quality

DP1.4 How do we measure the quality of a pattern instance?

Measuring Software Quality

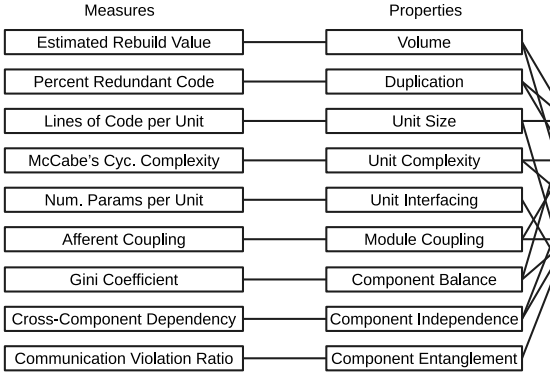
DP1.4 How do we measure the quality of a pattern instance?



Calibrated using SIG's prescribed method across 106 Open Source Projects from the *Qualitas.Class Corpus*

Measuring Software Quality

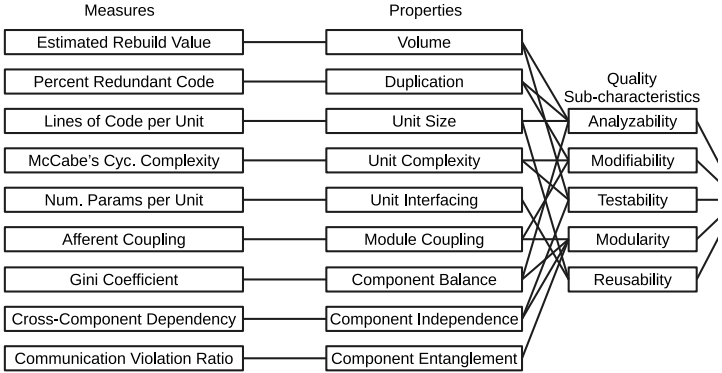
DP1.4 How do we measure the quality of a pattern instance?



Calibrated using SIG's prescribed method across 106 Open Source Projects from the *Qualitas.Class Corpus*

Measuring Software Quality

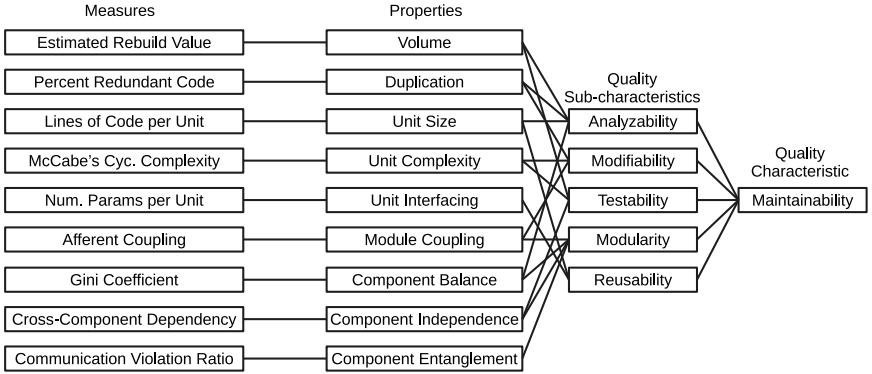
DP1.4 How do we measure the quality of a pattern instance?



Calibrated using SIG's prescribed method across 106 Open Source Projects from the *Qualitas.Class Corpus*

Measuring Software Quality

DP1.4 How do we measure the quality of a pattern instance?



Calibrated using SIG's prescribed method across 106 Open Source Projects from the *Qualitas.Class Corpus*

Measuring Technical Debt

DP1.5 How do we measure the technical debt of a pattern instance?

Measuring Technical Debt

DP1.5 How do we measure the technical debt of a pattern instance?

Nugroho et al.'s Calculations

- **TD Principal** (*Repair Effort (RE)*):
  $RE = RF * RV * RA$
  - Rebuild Value, $RV = SS * TF$
  - RF - Rework Fraction
  - SS - System Size in KLOC
  - TF - Technology Factor
  - RA - Refactoring Adjustment

Measuring Technical Debt

DP1.5 How do we measure the technical debt of a pattern instance?

Nugroho et al.'s Calculations

- **TD Principal** (*Repair Effort (RE)*):
  $RE = RF * RV * RA$
  - Rebuild Value, $RV = SS * TF$
  - RF - Rework Fraction
  - SS - System Size in KLOC
  - TF - Technology Factor
  - RA - Refactoring Adjustment

- **TD Interest (TDI)**:
  $TDI = ME_I - ME_C$
  - ME_I = Maintenance Effort at Ideal Quality Level
  - ME_C = Maintenance Effort at Current Quality Level

  $ME = \frac{MF * RV}{QF}$
  - MF = yearly maintenance effort of the system (0.15)
  - RV = Rebuild Value
  - QF = Quality Factor

  $QF = 2^{((QualityLevel-3)/2)}$

Research Questions

- **RQ2(3):** How does each type of grime affect software product maintainability (technical debt estimate)?

Research Questions

- **RQ2(3):** How does each type of grime affect software product maintainability (technical debt estimate)?
  - **RQ2(3).1:** How does each type of Grime affect design pattern quality for each of the selected maintainability sub-characteristics (technical debt principal and interest)?

Research Questions

- **RQ2(3):** How does each type of grime affect software product maintainability (technical debt estimate)?
  - **RQ2(3).1:** How does each type of Grime affect design pattern quality for each of the selected maintainability sub-characteristics (technical debt principal and interest)?
  - **RQ2(3).2:** What level of injection severity affects a change in design pattern quality for each of the maintainability sub-characteristics (technical debt principal and interest)?

Research Questions

- **RQ2(3):** How does each type of grime affect software product maintainability (technical debt estimate)?
  - **RQ2(3).1:** How does each type of Grime affect design pattern quality for each of the selected maintainability sub-characteristics (technical debt principal and interest)?
  - **RQ2(3).2:** What level of injection severity affects a change in design pattern quality for each of the maintainability sub-characteristics (technical debt principal and interest)?
  - **RQ2(3).3:** What is the difference between the effects of the grime types and their subtypes on maintainability sub-characteristics (technical debt principal and interest)?

Experimental Design

- Design Type: Factorial

Experimental Design

- Design Type: Factorial
- Replication Size: 2496 Units

Experimental Design

- Design Type: Factorial
- Replication Size: 2496 Units
  - 26 Grime Types for Injection

Experimental Design

- Design Type: Factorial
- Replication Size: 2496 Units
  - 26 Grime Types for Injection
  - 16 Design Pattern Types

Pattern Types	
Adapter	Observer
Bridge	Prototype
Chain of Responsibility	Proxy
Command	Singleton
Composite	State
Decorator	Strategy
Flyweight	Template Method
Factory Method	Visitor

Experimental Design

- Design Type: Factorial
- Replication Size: 2496 Units
  - 26 Grime Types for Injection
  - 16 Design Pattern Types
  - 6 Levels of Injection Severity

Pattern Types	
Adapter	Observer
Bridge	Prototype
Chain of Responsibility	Proxy
Command	Singleton
Composite	State
Decorator	Strategy
Flyweight	Template Method
Factory Method	Visitor

Level	Max % Grime Injected
0	0% (Control)
1	$\leq$ 15%
2	$\leq$ 30%
3	$\leq$ 45%
4	$\leq$ 60%
5	$\leq$ 75%

Experimental Design

- Design Type: Factorial
- Replication Size: 2496 Units
  - 26 Grime Types for Injection
  - 16 Design Pattern Types
  - 6 Levels of Injection Severity

Pattern Types	
Adapter	Observer
Bridge	Prototype
Chain of Responsibility	Proxy
Command	Singleton
Composite	State
Decorator	Strategy
Flyweight	Template Method
Factory Method	Visitor

Level	Max % Grime Injected
0	0% (Control)
1	$\leq$ 15%
2	$\leq$ 30%
3	$\leq$ 45%
4	$\leq$ 60%
5	$\leq$ 75%

Experimental Design

- Design Type: Factorial
- Replication Size: 2496 Units
  - 26 Grime Types for Injection
  - 16 Design Pattern Types
  - 6 Levels of Injection Severity

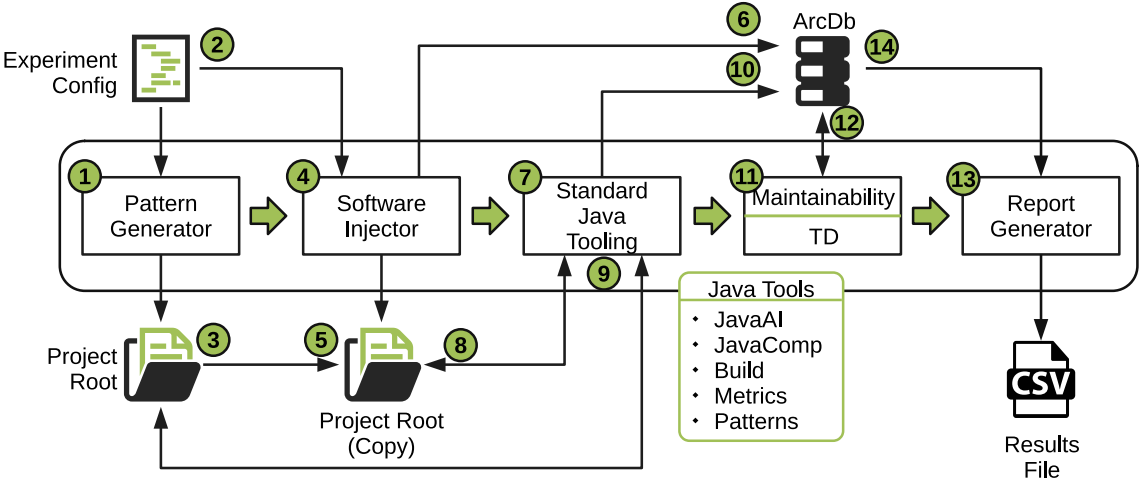
Pattern Types	
Adapter	Observer
Bridge	Prototype
Chain of Responsibility	Proxy
Command	Singleton
Composite	State
Decorator	Strategy
Flyweight	Template Method
Factory Method	Visitor

Level	Max % Grime Injected
0	0% (Control)
1	$\leq$ 15%
2	$\leq$ 30%
3	$\leq$ 45%
4	$\leq$ 60%
5	$\leq$ 75%

Analysis:

- Each experiment was analyzed using a Permutation F-Test
- Followed by an analysis of interactions between Pattern Type, Injection Type and Injection Severity
- Here we present 3 of the 7 Experiments

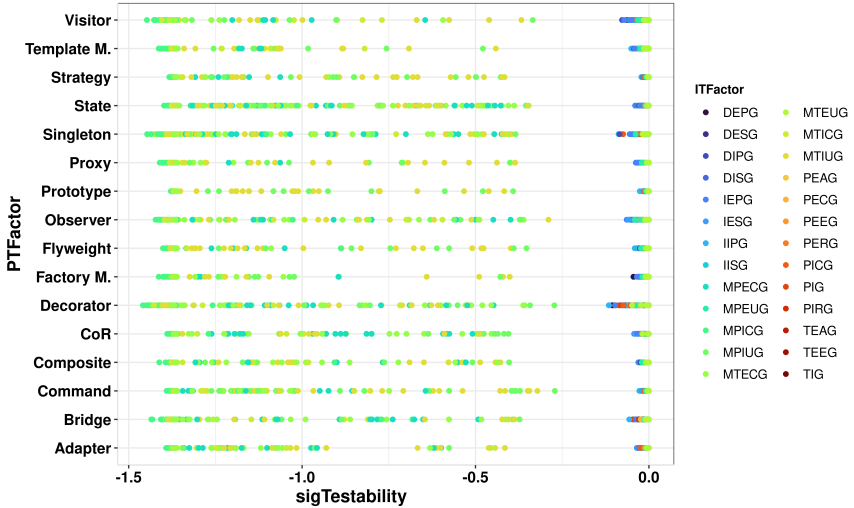
Data Collection



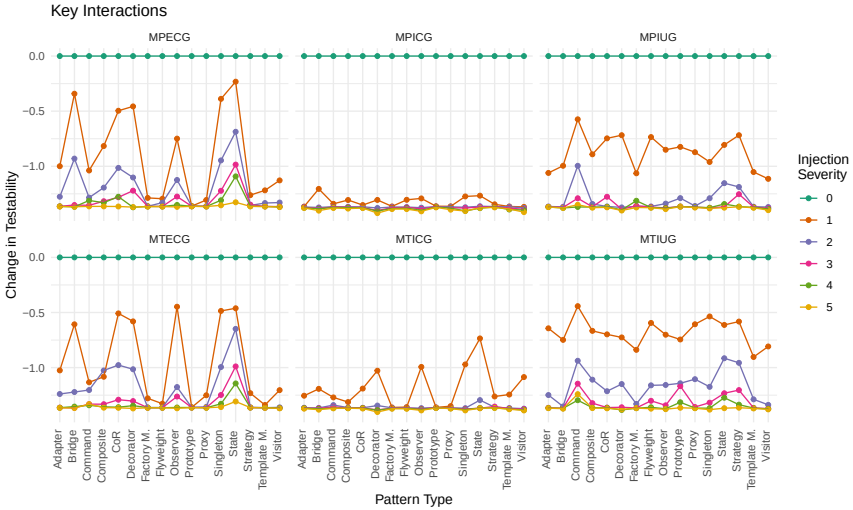
Testability

Experiment

Testability



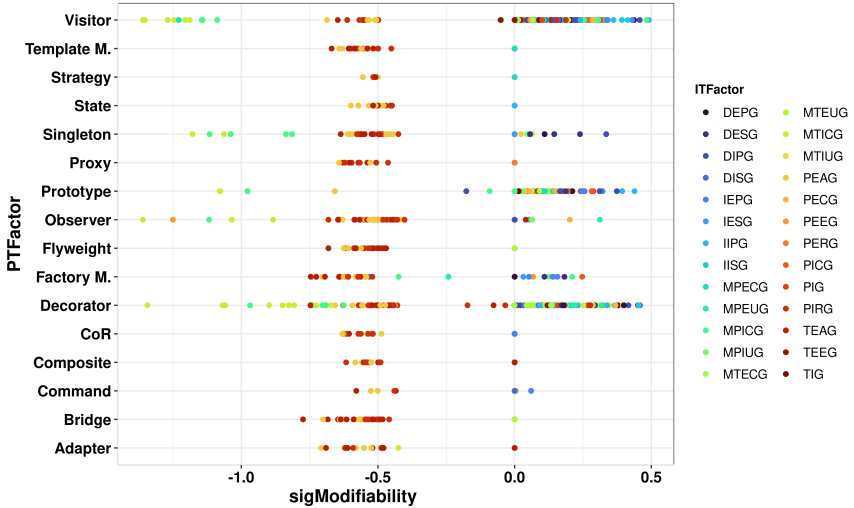
Testability



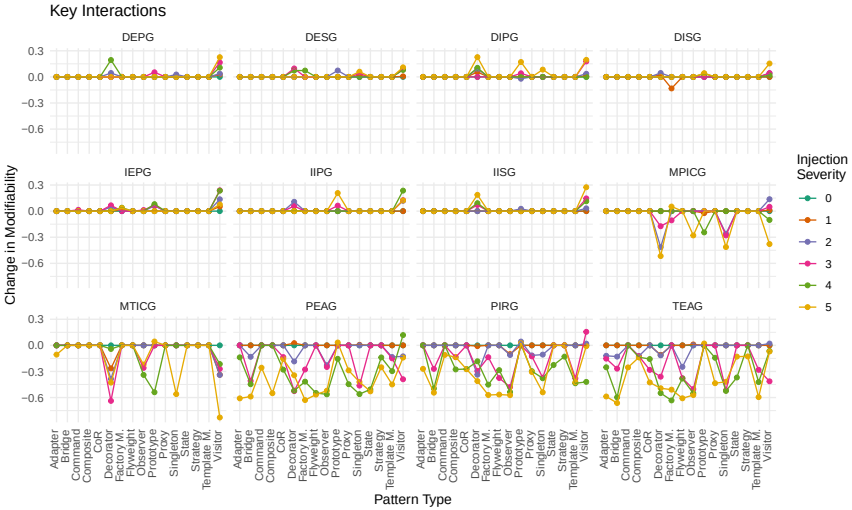
Modifiability

Experiment

Modifiability



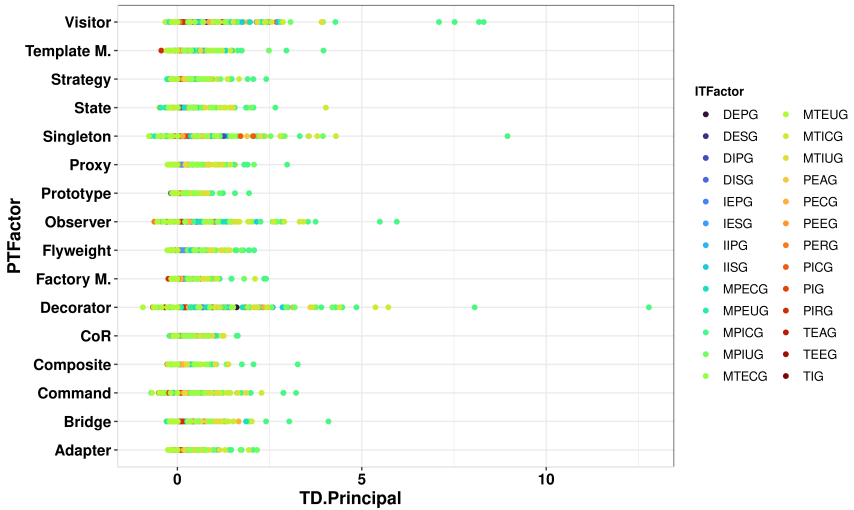
Modifiability



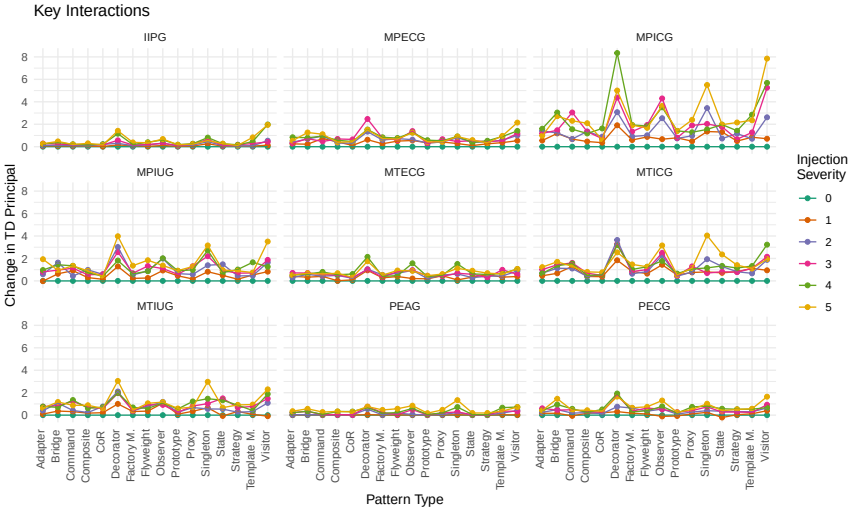
TD Principal

Experiment

TD Principal



TD Principal



Phase 03 - Verification Study

RG4: Analyze design pattern instances for the purpose of comparing injected and observed instances of grime with respect to their ISO/IEC 25010 Maintainability subcharacteristics attributes and Technical Debt Principal and Interest from the perspective of researchers in the context of open source Java™ software projects.

Phase 03 - Verification Study

RG4: Analyze design pattern instances for the purpose of comparing injected and observed instances of grime with respect to their ISO/IEC 25010 Maintainability subcharacteristics attributes and Technical Debt Principal and Interest from the perspective of researchers in the context of open source Java™ software projects.

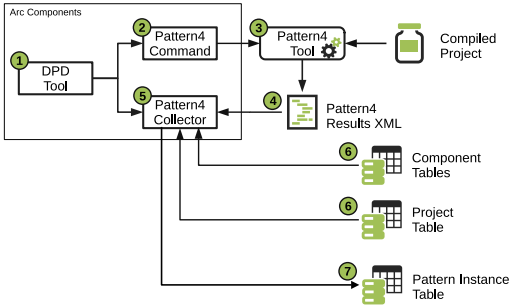
DP2: How do we verify the results of the experiments with real software?

Design Pattern Detection

DP2.1: How do we incorporate design pattern detection into the Arc Framework?

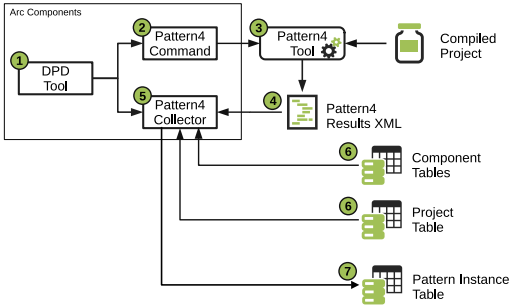
Design Pattern Detection

DP2.1: How do we incorporate design pattern detection into the Arc Framework?



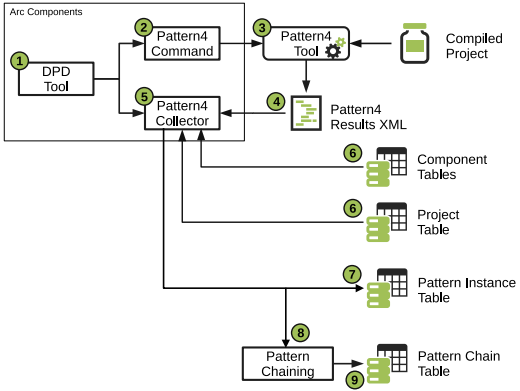
Design Pattern Detection

DP2.2: How do we track instances of design patterns across versions of a software system?



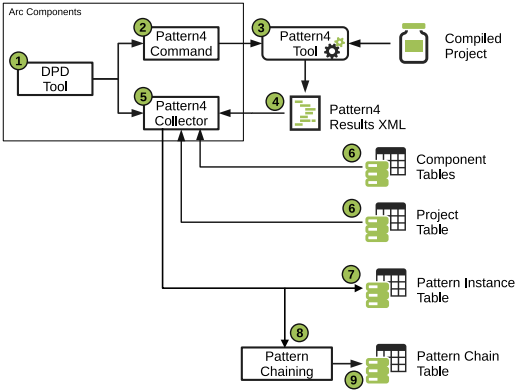
Design Pattern Detection

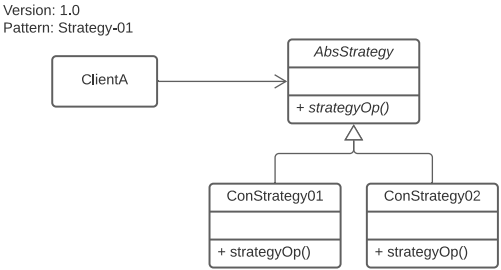
DP2.2: How do we track instances of design patterns across versions of a software system?



Design Pattern Detection

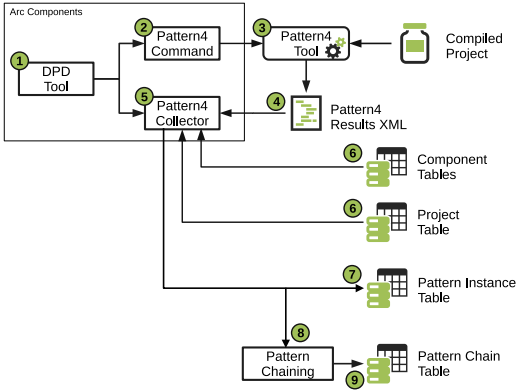
DP2.2: How do we track instances of design patterns across versions of a software system?

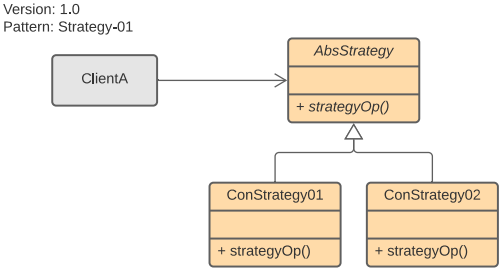




Design Pattern Detection

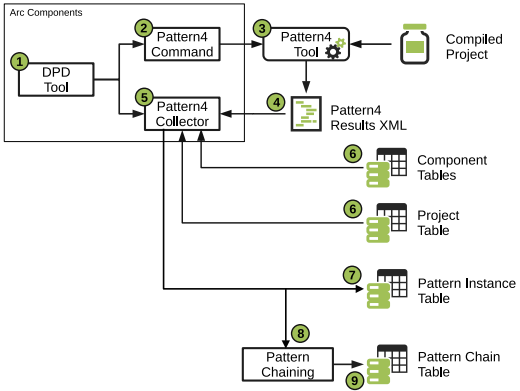
DP2.2: How do we track instances of design patterns across versions of a software system?

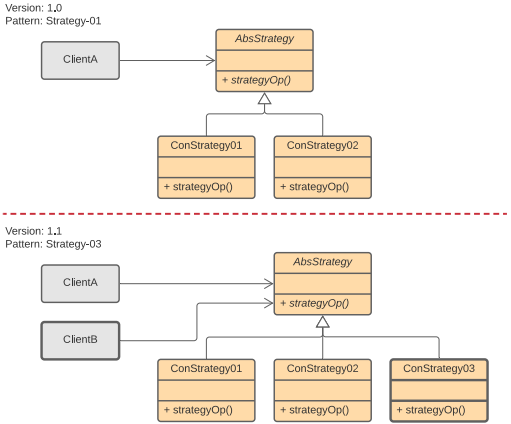




Design Pattern Detection

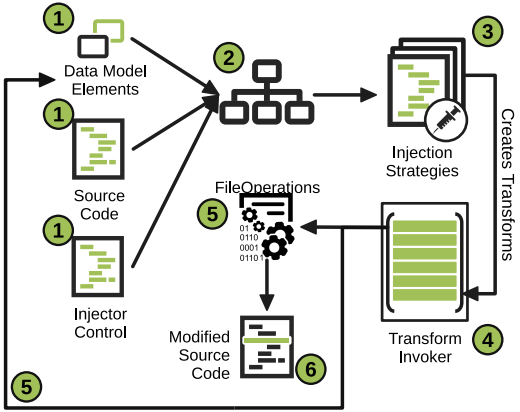
DP2.2: How do we track instances of design patterns across versions of a software system?





Controlled Injection

DP2.3: How do we modify the Software Injector to allow for controlled injection to mimic the change in grime that occurred between versions?

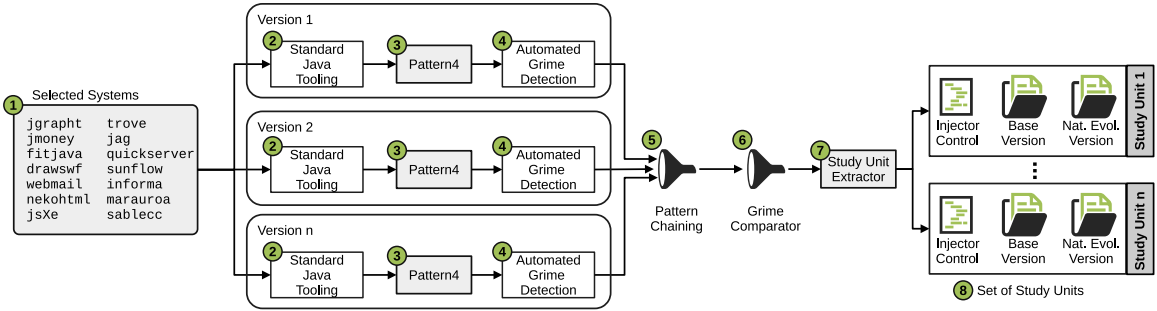


Research Questions

RQ4: Do observed and injected grime have a similar effect on the Maintainability sub-characteristics and Technical Debt Principal and Interest?

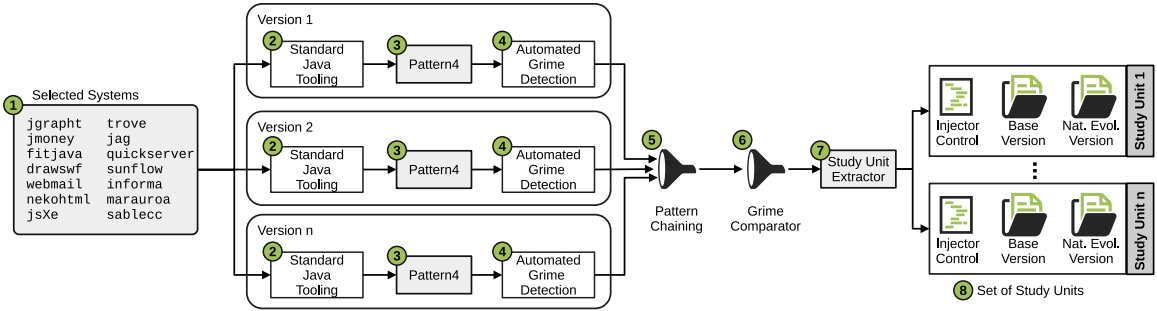
Data Collection (1)

Phase 1: Study Unit Extraction



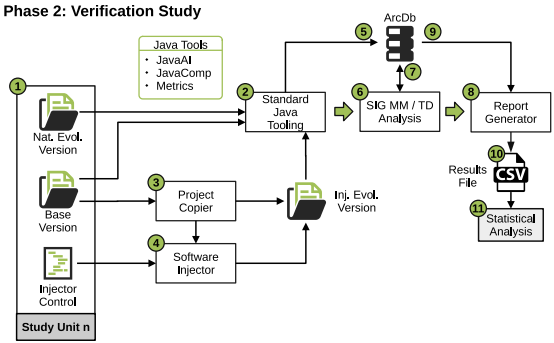
Data Collection (1)

Phase 1: Study Unit Extraction

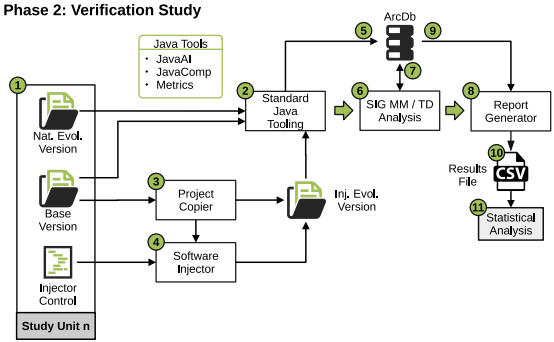


Total units found: 8

Data Collection (2)



Data Collection (2)



- Next we map the change in each characteristic, Δc to a nominal scale for comparison:

$$map(\Delta c) = \begin{cases} L & \text{if } \Delta c < 0 \\ E & \text{if } \Delta c = 0 \\ G & \text{if } \Delta c > 0 \end{cases}$$

Where Δc is the difference between:

- Natural Evolution and Base Version
- Injected Evolution and Base Version

Results

	Study Units															
	1 - State		2 - State		3 - Sngltn		4 - Adapt		5 - TM		6 - TM		7 - State		8 - Sngltn	
	Nat	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*
Ana	E	G	L	L	G	L	E	E	L	G	L	L	L	L	G	L
Test	E	L	L	L	L	L	E	L	G	L	G	L	G	L	G	L
Modul	E	G	L	L	E	L	E	L	G	L	L	L	L	L	E	L
Modif	E	E	E	E	E	E	E	G	G	G	G	G	E	E	G	G
Reuse	E	G	G	G	G	L	E	L	G	L	G	L	L	L	L	L
TDP	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L
TDI	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L

Results

What is the agreement between natural and injection evolution?

	Study Units															
	1 - State		2 - State		3 - Sngltn		4 - Adapt		5 - TM		6 - TM		7 - State		8 - Sngltn	
	Nat	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*
Ana	E	G	L	L	G	L	E	E	L	G	L	L	L	L	G	L
Test	E	L	L	L	L	L	E	L	G	L	G	L	G	L	G	L
Modul	E	G	L	L	E	L	E	L	G	L	L	L	L	L	E	L
Modif	E	E	E	E	E	E	E	G	G	G	G	G	E	E	G	G
Reuse	E	G	G	G	G	L	E	L	G	L	G	L	L	L	L	L
TDP	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L
TDI	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L

Results

What is the agreement between natural and injection evolution?

	Study Units															
	1 - State		2 - State		3 - Snglton		4 - Adapt		5 - TM		6 - TM		7 - State		8 - Snglton	
	Nat	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*	*Nat*	*Inj*
Ana	E	G	L	L	G	L	E	E	L	G	L	L	L	L	G	L
Test	E	L	L	L	L	L	E	L	G	L	G	L	G	L	G	L
Modul	E	G	L	L	E	L	E	L	G	L	L	L	L	L	E	L
Modif	E	E	E	E	E	E	E	G	G	G	G	G	E	E	G	G
Reuse	E	G	G	G	G	L	E	L	G	L	G	L	L	L	L	L
TDP	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L
TDI	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L

	Attribute						
	Ana	Test	Modul	Modif	Reuse	TDP	TDI
κ	0.16	0.0	0.048	0.75	0.091	-0.077	-0.077
Agreement	Slight	None	Slight	Substantial	Slight	None	None

Results

How does natural and injected evolution compare to experimental results?

	Study Units															
	1 - State		2 - State		3 - Snglton		4 - Adapt		5 - TM		6 - TM		7 - State		8 - Snglton	
	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj
Ana	E	G	L	L	G	L	E	E	L	G	L	L	L	L	G	L
Test	E	L	L	L	L	L	E	L	G	L	G	L	G	L	G	L
Modul	E	G	L	L	E	L	E	L	G	L	L	L	L	L	E	L
Modif	E	E	E	E	E	E	E	G	G	G	G	G	E	E	G	G
Reuse	E	G	G	G	G	L	E	L	G	L	G	L	L	L	L	L
TDP	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L
TDI	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L

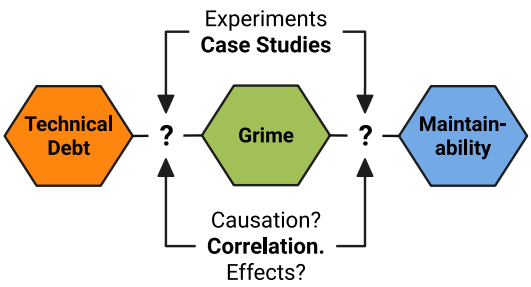
Results

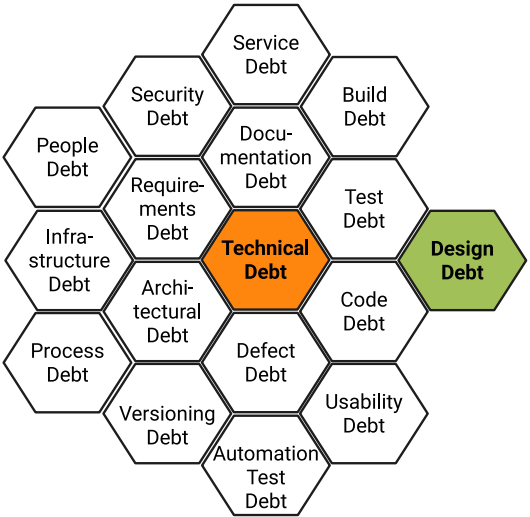
How does natural and injected evolution compare to experimental results?

	Study Units															
	1 - State		2 - State		3 - Sngltn		4 - Adapt		5 - TM		6 - TM		7 - State		8 - Sngltn	
	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj	Nat	Inj
Ana	E	G	L	L	G	L	E	E	L	G	L	L	L	L	G	L
Test	E	L	L	L	L	L	E	L	G	L	G	L	G	L	G	L
Modul	E	G	L	L	E	L	E	L	G	L	L	L	L	L	E	L
Modif	E	E	E	E	E	E	E	G	G	G	G	G	E	E	G	G
Reuse	E	G	G	G	G	L	E	L	G	L	G	L	L	L	L	L
TDP	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L
TDI	E	G	G	G	G	L	E	L	G	L	G	L	G	L	G	L

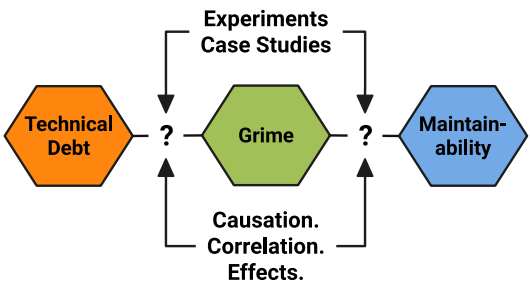
- 39.583% of the natural evolution results reflect the experimental results
- 52.083% of the injected evolution results reflect the experimental results

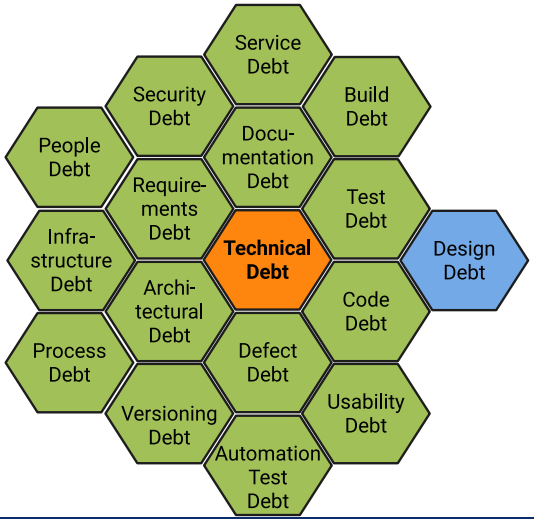
Impacts


Experiments
Case Studies
Technical Debt
?
Grime
?
Maintain-ability
Causation?
Correlation.
Effects?

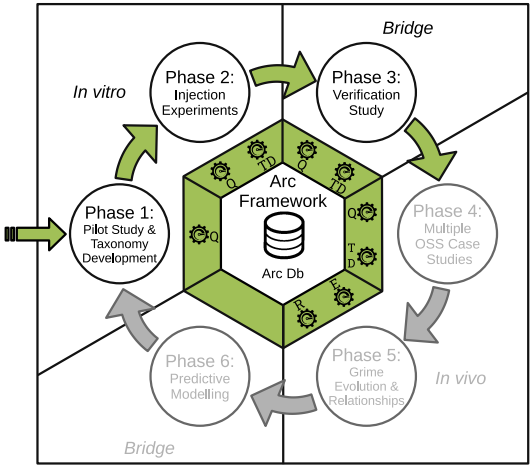

Service Debt
Security Debt
Build Debt
People Debt
Docu-mentation Debt
Require-ments Debt
Test Debt
Infra-structure Debt
Technical Debt
Design Debt
Archi-tectural Debt
Code Debt
Process Debt
Defect Debt
Versioning Debt
Usability Debt
Automation Test Debt

Impacts


Experiments
Case Studies
Technical Debt
?
Grime
?
Maintain-ability
Causation.
Correlation.
Effects.


Service Debt
Security Debt
Build Debt
People Debt
Docu-mentation Debt
Require-ments Debt
Test Debt
Infra-structure Debt
Technical Debt
Design Debt
Archi-tectural Debt
Code Debt
Process Debt
Defect Debt
Versioning Debt
Usability Debt
Automation Test Debt

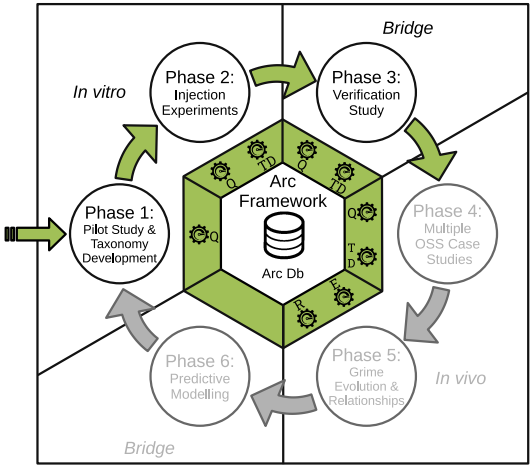
Impacts



Researchers:

- Presents a promising approach to explore and understand SE phenomena

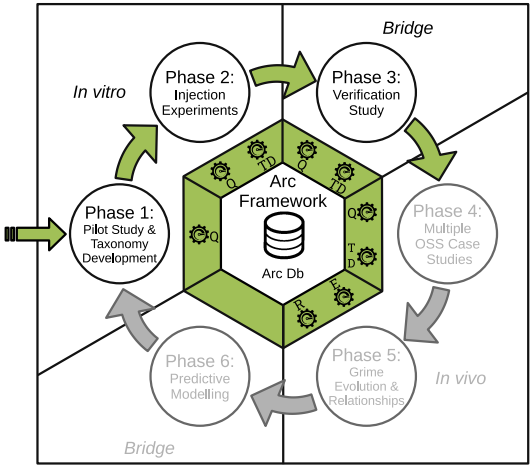
Impacts



Researchers:

- Presents a promising approach to explore and understand SE phenomena
- Bridge the gap between empirical methods allows a combination of results

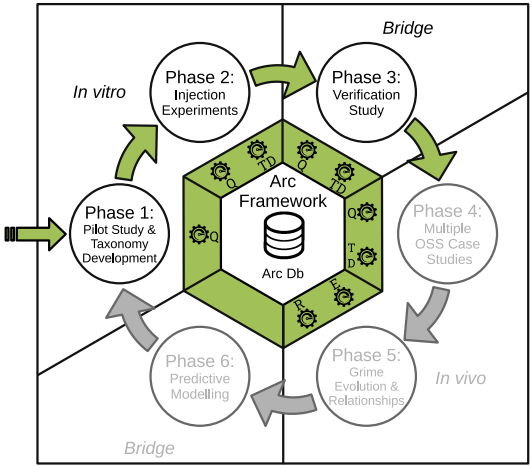
Impacts



Researchers:

- Presents a promising approach to explore and understand SE phenomena
- Bridge the gap between empirical methods allows a combination of results
- Confirmed prior results concerning the effects of Modular Grime on Testability and TD Principal

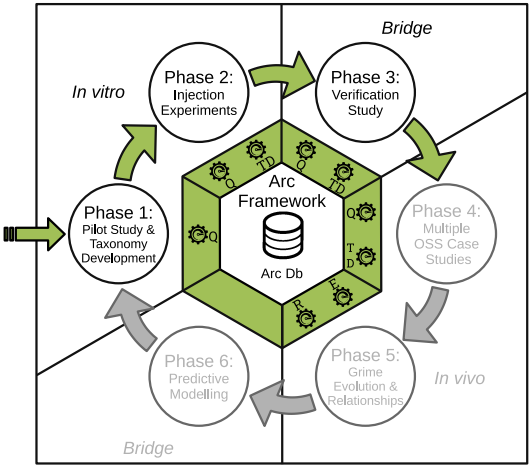
Impacts



Researchers:

- Presents a promising approach to explore and understand SE phenomena
- Bridge the gap between empirical methods allows a combination of results
- Confirmed prior results concerning the effects of Modular Grime on Testability and TD Principal
- Approach to inject and detect design pattern grime for experimental purposes

Impacts



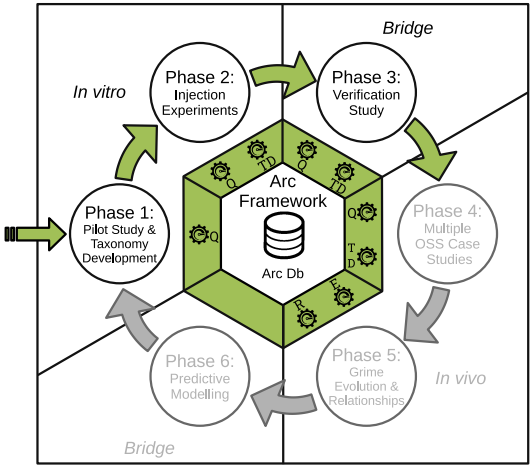
Researchers:

- Presents a promising approach to explore and understand SE phenomena
- Bridge the gap between empirical methods allows a combination of results
- Confirmed prior results concerning the effects of Modular Grime on Testability and TD Principal
- Approach to inject and detect design pattern grime for experimental purposes

Practitioners:

- Open implementation of the SIG Maintainability Model

Impacts



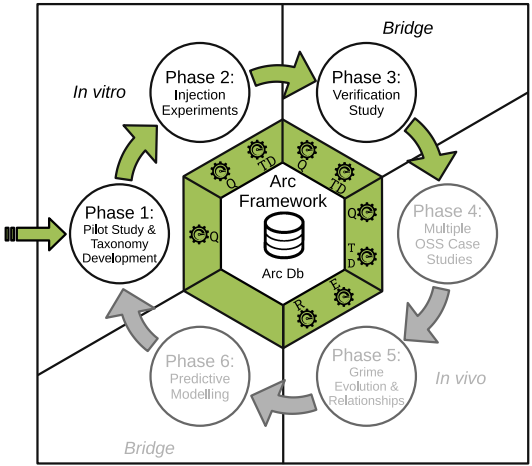
Researchers:

- Presents a promising approach to explore and understand SE phenomena
- Bridge the gap between empirical methods allows a combination of results
- Confirmed prior results concerning the effects of Modular Grime on Testability and TD Principal
- Approach to inject and detect design pattern grime for experimental purposes

Practitioners:

- Open implementation of the SIG Maintainability Model

Impacts



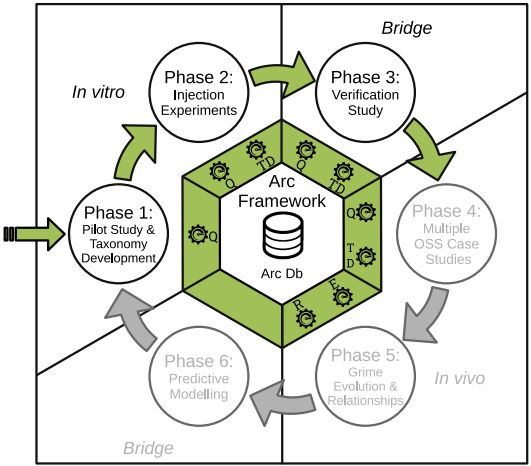
Researchers:

- Presents a promising approach to explore and understand SE phenomena
- Bridge the gap between empirical methods allows a combination of results
- Confirmed prior results concerning the effects of Modular Grime on Testability and TD Principal
- Approach to inject and detect design pattern grime for experimental purposes

Practitioners:

- Open implementation of the SIG Maintainability Model
- First known implementation of Nugroho et al.'s TD Measurements

Impacts



Researchers:

- Presents a promising approach to explore and understand SE phenomena
- Bridge the gap between empirical methods allows a combination of results
- Confirmed prior results concerning the effects of Modular Grime on Testability and TD Principal
- Approach to inject and detect design pattern grime for experimental purposes

Practitioners:

- Open implementation of the SIG Maintainability Model
- First known implementation of Nugroho et al.'s TD Measurements
- **Modifyiability**: Specific grime types should be of concern for instances of the Decorator, Observer, Singleton, and Visitor patterns.

Threats to Validity

Construct Validity

- Experiments and Validation Study:
  - Use of modified SIG Maintainability model with linear projection during rating phase.
- Validation Study:
  - Assumption of accumulative changes in grime between versions

Threats to Validity

Construct Validity

- Experiments and Validation Study:
  - Use of modified SIG Maintainability model with linear projection during rating phase.
- Validation Study:
  - Assumption of accumulative changes in grime between versions

Content Validity

- Experiments:
  - Only covered 16 of the 23 GoF patterns

Threats to Validity

Construct Validity

- Experiments and Validation Study:
  - Use of modified SIG Maintainability model with linear projection during rating phase.
- Validation Study:
  - Assumption of accumulative changes in grime between versions

Content Validity

- Experiments:
  - Only covered 16 of the 23 GoF patterns

External Validity

- Experiments:
  - Injection used generated rather than real systems
  - Does not account for Behavioral or Unknown types of grime
  - Only analyzed the Java language
- Validation Study:
  - Only Open-Source Java Software Systems
  - Small sample size
  - Limited coverage of grime types
  - Limited coverage of pattern types

Related Publications

- Manuscript in progress.

Related Publications

- Manuscript in progress.
- Taxonomies
  - Izurieta C., Reimanis D., **Griffith I.**, Schanz T., "Structural and Behavioral Taxonomies of Design Pattern Grime Evolution". In Proceedings of the 12th Seminar on Advanced Techniques & Tools for Software Evolution, SATToSE 2019. Bolzano, Italy, July 8-10, 2019.

Related Publications

- Manuscript in progress.
- Taxonomies
  - Izurieta C., Reimanis D., **Griffith I.**, Schanz T., "Structural and Behavioral Taxonomies of Design Pattern Grime Evolution". In Proceedings of the 12th Seminar on Advanced Techniques & Tools for Software Evolution, SATToSE 2019. Bolzano, Italy, July 8-10, 2019.
- Quality Modeling
  - **Griffith I.**, Izurieta C., Huvaere C., "An Industry Perspective to Comparing the SQALE and Quamoco Software Quality Models," 11th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, Canada, November 9-10, 2017.

Related Publications

- Manuscript in progress.
- Taxonomies
  - Izurieta C., Reimanis D., **Griffith I.**, Schanz T., "Structural and Behavioral Taxonomies of Design Pattern Grime Evolution". In Proceedings of the 12th Seminar on Advanced Techniques & Tools for Software Evolution, SATToSE 2019. Bolzano, Italy, July 8-10, 2019.
- Quality Modeling
  - **Griffith I.**, Izurieta C., Huvaere C., "An Industry Perspective to Comparing the SQALE and Quamoco Software Quality Models," 11th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, Canada, November 9-10, 2017.
- Technical Debt Measurement Implementation
  - **Griffith I.**, Reimanis D., Izurieta C., Codabux Z., Deo A., and Williams B., "The Correspondence between Software Quality Models and Technical Debt Estimation Approaches". In Proceedings of the 6th International Workshop on Managing Technical Debt. Victoria, British Columbia, Canada, September 30, 2014.

Related Publications

- Software Injection and Initial Class Grime Taxonomy
  - **Griffith I.** and Izurieta C., "Design Pattern Decay: The Case for Class Grime". In Proceedings of the 8th International Symposium on Empirical Software Engineering and Measurement. Torino, Italy, September 18–19, 2014.

Related Publications

- Software Injection and Initial Class Grime Taxonomy
  - **Griffith I.** and Izurieta C., "Design Pattern Decay: The Case for Class Grime". In Proceedings of the 8th International Symposium on Empirical Software Engineering and Measurement. Torino, Italy, September 18–19, 2014.
- The Overall Method
  - **Griffith I.** and Izurieta C., "Design Pattern Decay: An Extended Taxonomy and Empirical Study of Grime and its Impact on Design Pattern Evolution". The 11th International Doctoral Symposium on Empirical Software Engineering. Baltimore, Maryland, October 9, 2013.

Related Publications

- Software Injection and Initial Class Grime Taxonomy
  - **Griffith I.** and Izurieta C., "Design Pattern Decay: The Case for Class Grime". In Proceedings of the 8th International Symposium on Empirical Software Engineering and Measurement. Torino, Italy, September 18–19, 2014.
- The Overall Method
  - **Griffith I.** and Izurieta C., "Design Pattern Decay: An Extended Taxonomy and Empirical Study of Grime and its Impact on Design Pattern Evolution". The 11th International Doctoral Symposium on Empirical Software Engineering. Baltimore, Maryland, October 9, 2013.
- Arc Data Model and Metrics
  - **Griffith I.**, Wahl S., and Izurieta C., "Evolution of Legacy System Comprehensibility through Automated Refactoring". In Proceedings of IEEE ACM MALETS 2011 International Workshop on Machine Learning Technologies in Software Engineering. In association with the 26th International Conference on Automated Software Engineering, ASE 2011. Lawrence, Kansas, November 7–12, 2011.
  - **Griffith I.**, Wahl S., and Izurieta C., "TrueRefactor: An Automated Refactoring Tool to Improve Legacy System and Application Comprehensibility". In Proceedings of ISCA 24th International Conference on Computer Applications in Industry and Engineering, CAINE '11. Honolulu, Hawaii, November 2011.

Future Work

Short-term:

- Identification and correction of the underlying issues leading to the results of the verification study

Future Work

Short-term:

- Identification and correction of the underlying issues leading to the results of the verification study
  - Improve Taxonomy Definitions (if needed)
  - Improve Injection Strategies and Process
  - Re-execute Phase 02 and 03 Studies

Future Work

Short-term:

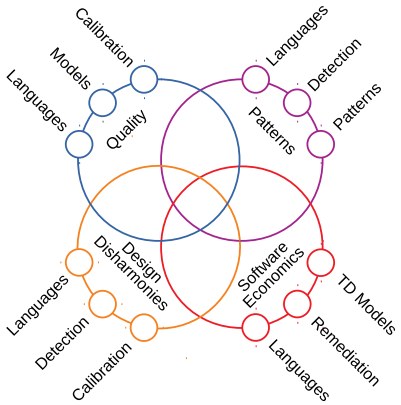
- Identification and correction of the underlying issues leading to the results of the verification study
  - Improve Taxonomy Definitions (if needed)
  - Improve Injection Strategies and Process
  - Re-execute Phase 02 and 03 Studies

Future Work

Short-term:

- Identification and correction of the underlying issues leading to the results of the verification study
  - Improve Taxonomy Definitions (if needed)
  - Improve Injection Strategies and Process
  - Re-execute Phase 02 and 03 Studies

Long-term:



Thank You!

Are there any questions?