POLYTECHNIQUE
MONTRÉAL

Consensus-Based Recommendation Technique for Software Engineering Applications

Layan Etaiwi
Polytechnique Montréal
Canada

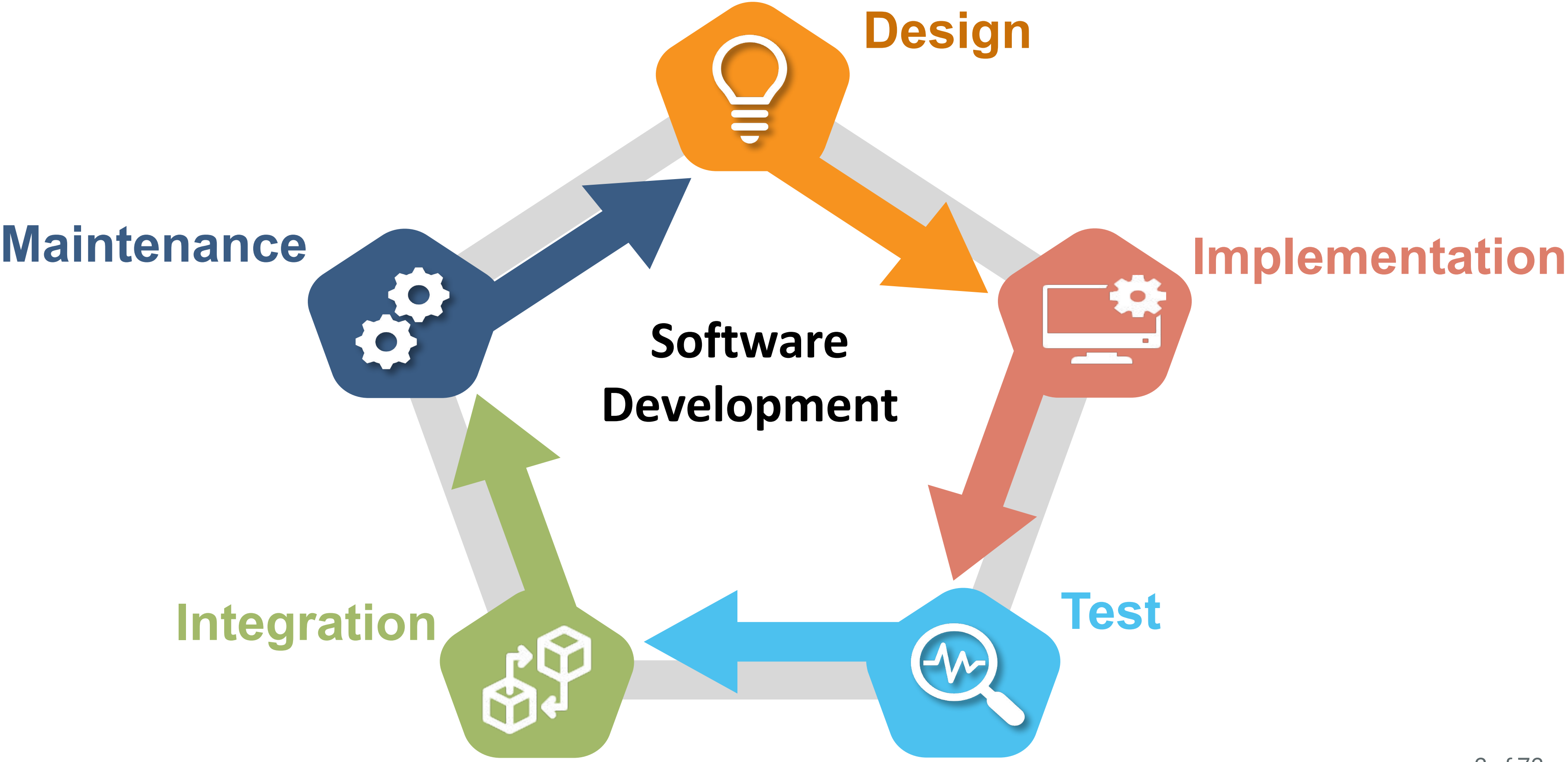
Supervised by:

Yann-Gaël Guéhéneuc
Concordia University

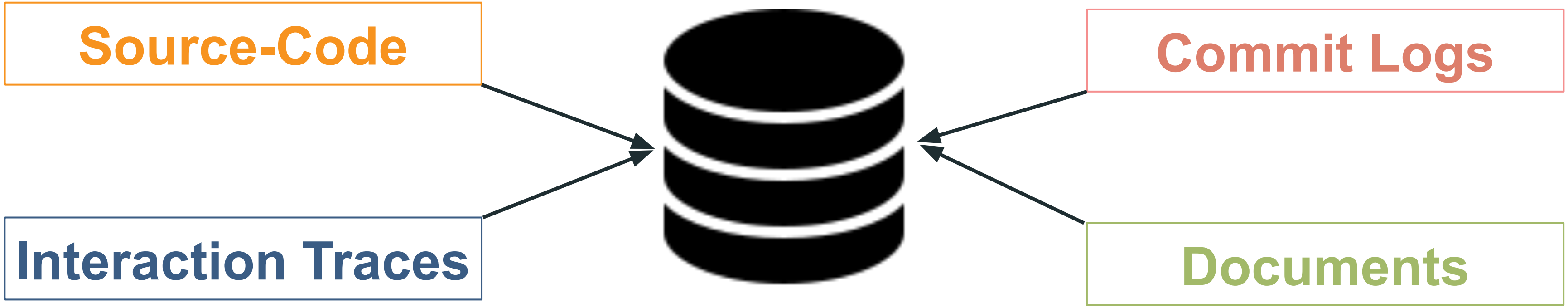
Sylvie Hamel
University of Montréal

Foutse Khomh
Polytechnique Montréal

Thesis Context



Thesis Context



Source-Code

Interaction Traces

Commit Logs

Documents

Thesis Context

Data Growth

Problem

Find specific information that can solve a specific issue, implement a feature, make a decision, etc.

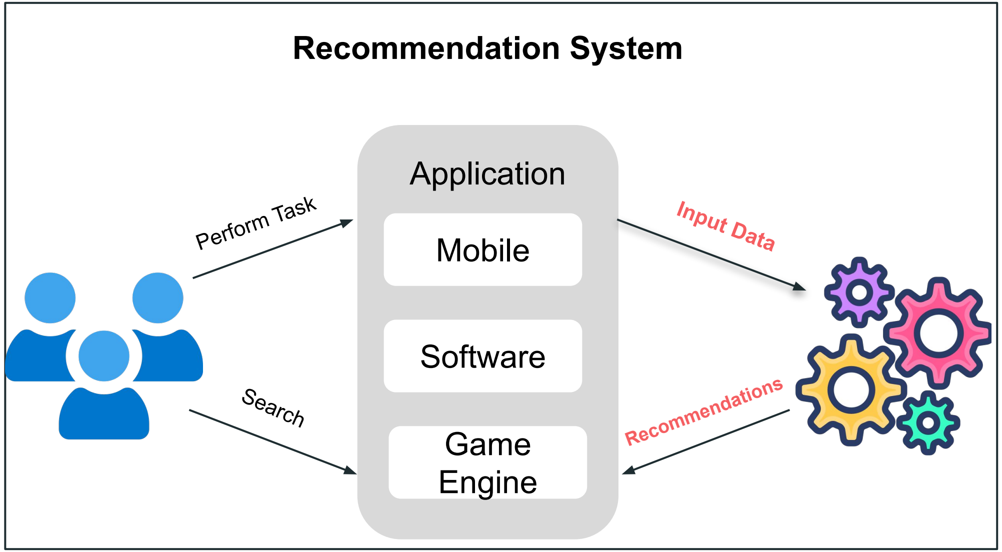
~~Information Retrieval~~
~~Information Filtering~~

Thesis Context

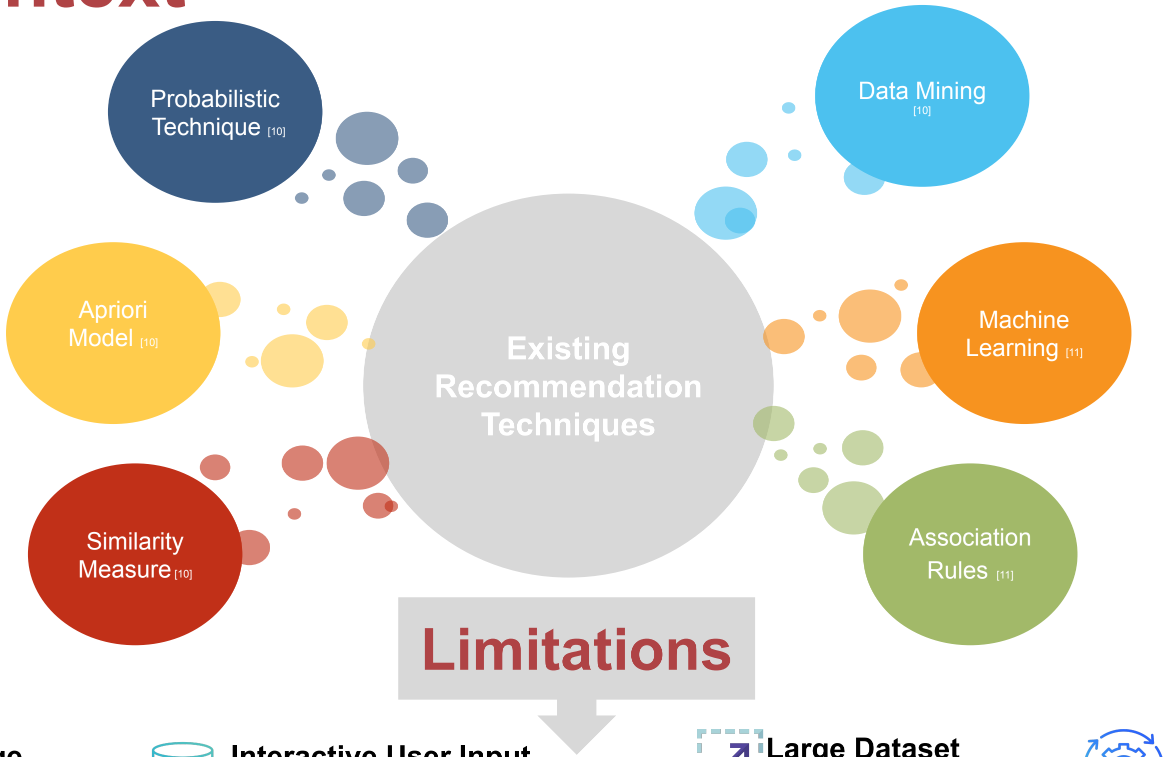
1990s

Solution

Develop a recommendation system that can reduce data overload, deliver the most pertinent data, ease the performance of activities, and improving decision making process



Thesis Context



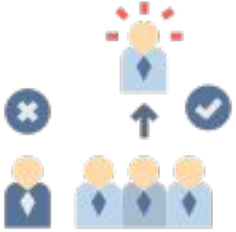
Sufficient Knowledge
Requirement of knowledge about the features of the items in the dataset

Interactive User Input
Requirement of interactive users' input to the recommender, which may suggest unrelated items if the users interact with the "wrong" item

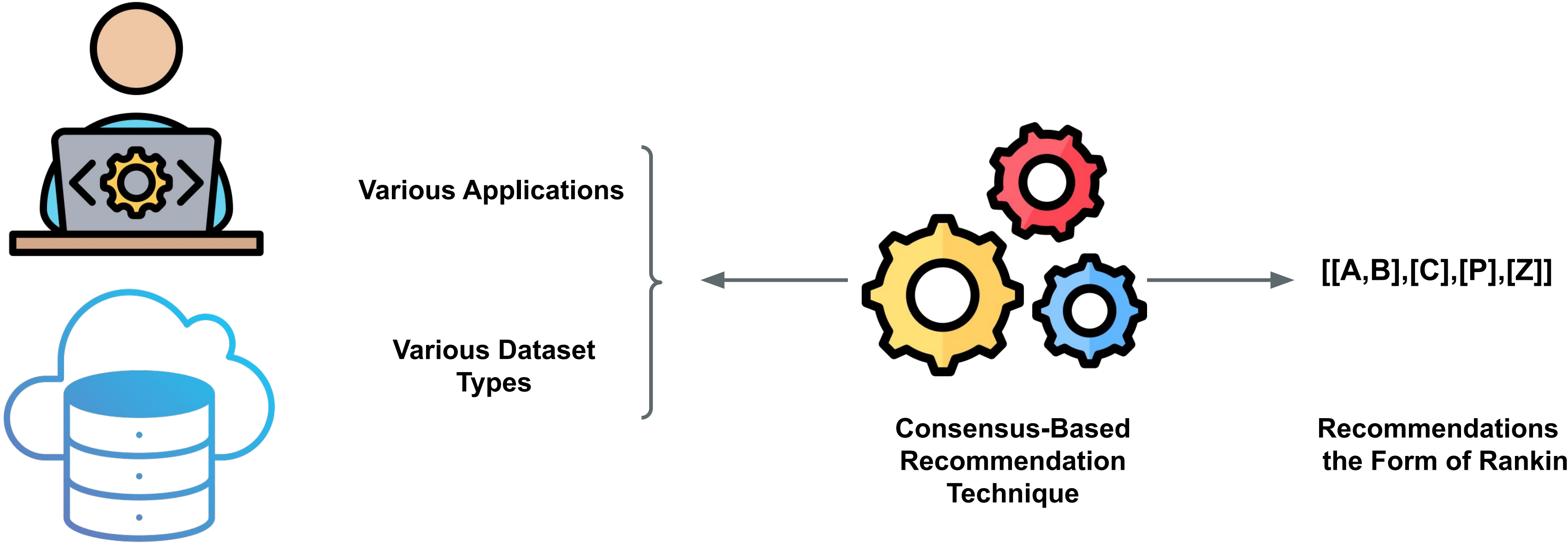
Large Dataset
Requirement of large datasets
≅ 1M [9]

Generalization
Applicability to all data types and software applications

Thesis Statement



We propose a recommendation technique for software engineering based on the **consensus algorithm** and that applies to **various data types** and resolves **various software engineering issues in a variety of applications**



Consensus-Based Recommendation Technique

Recommendations in the Form of Ranking

Research Methodology

Applying the proposed technique to build recommendation approaches that solve issues in…

Mobile Applications

Software Systems

Game Engines

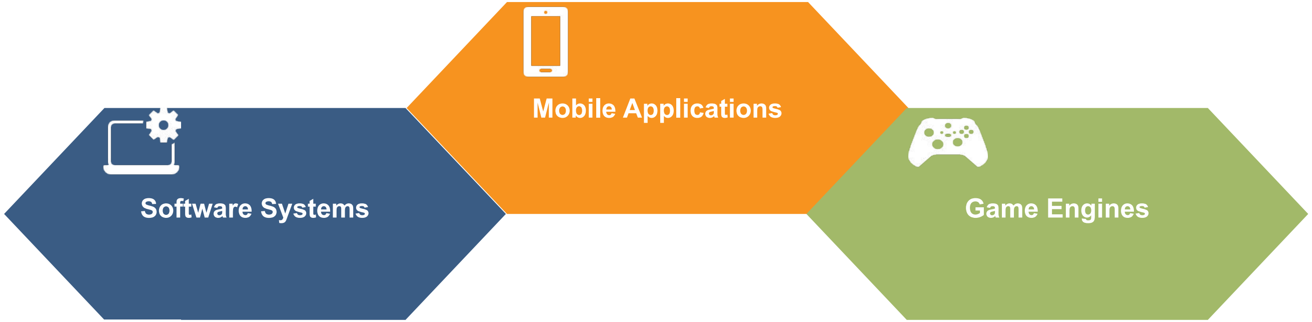
Recommend mobile app reviews to help developers plan their apps' next release

Recommend files-to-edit to help developers complete similar change tasks

Recommend common subsystems to help developers design their game engine architecture

Research Methodology

Applying the proposed technique to build recommendation approaches that solve issues in…



Mobile Applications

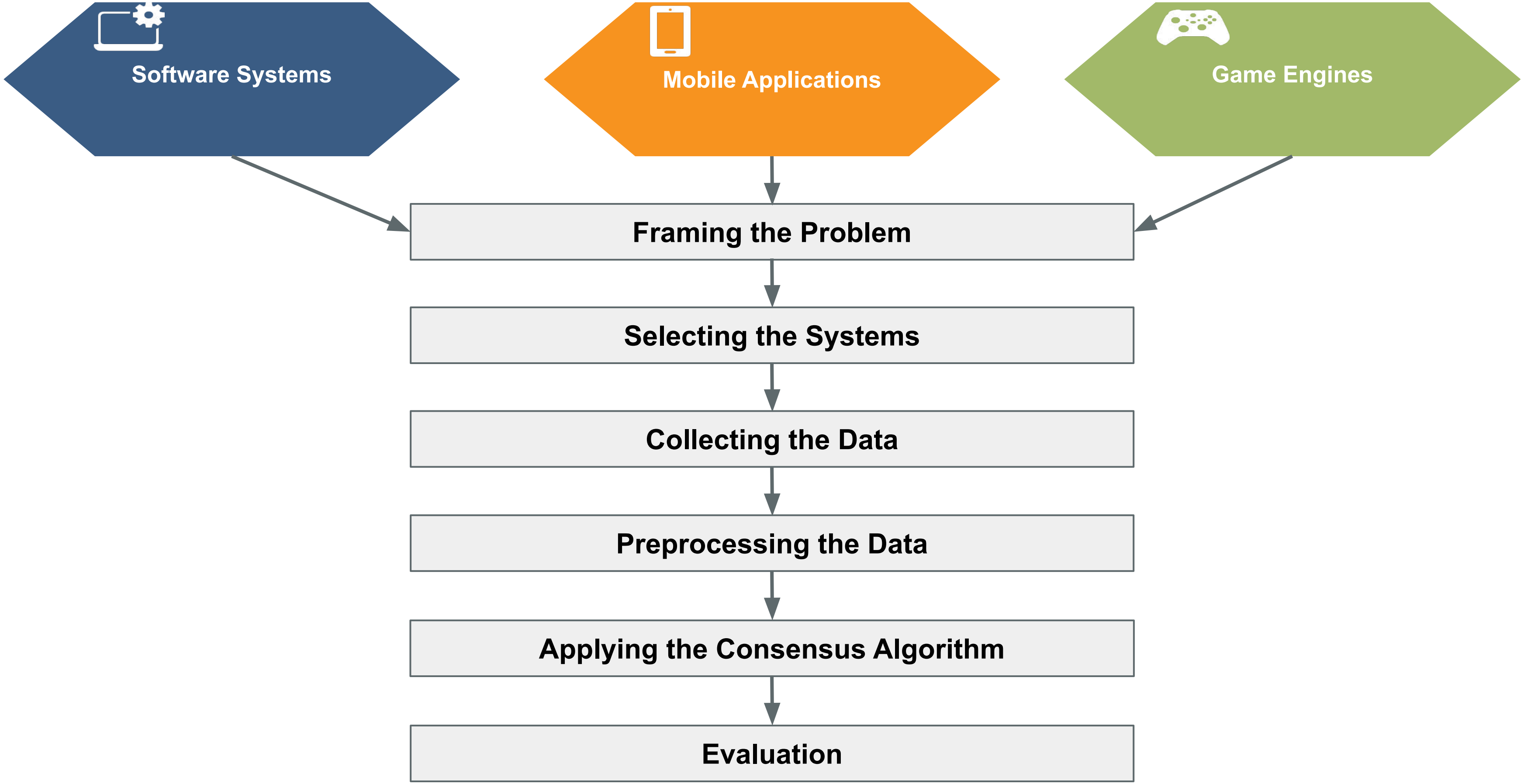
Software Systems

Game Engines

But why these applications?

- Need to help developers prioritize their reviews
- Need to build recommendations for forked software systems
- Need to provide a model of game engine architecture of subsystems

Research Methodology



Software Systems

Mobile Applications

Game Engines

Framing the Problem

Selecting the Systems

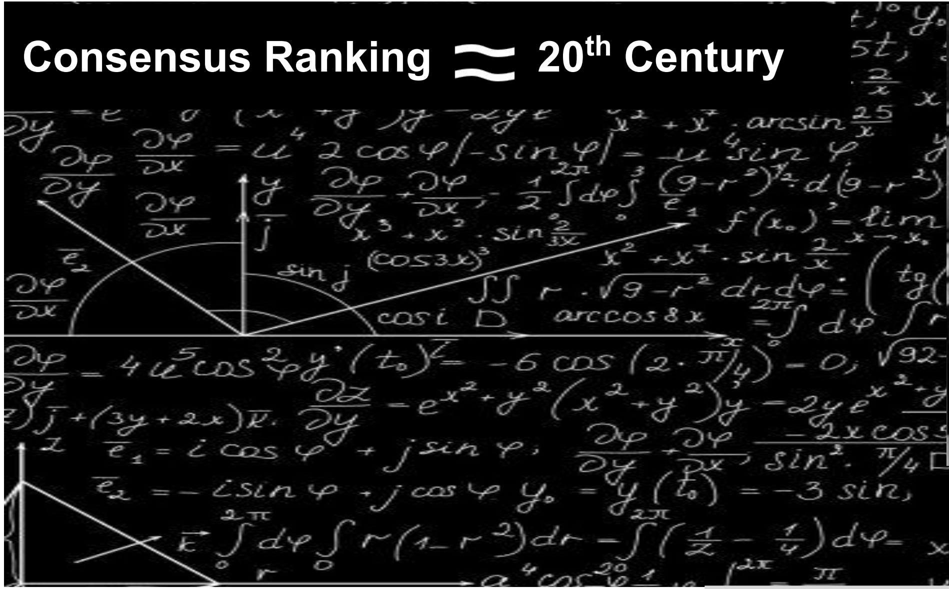
Collecting the Data

Preprocessing the Data

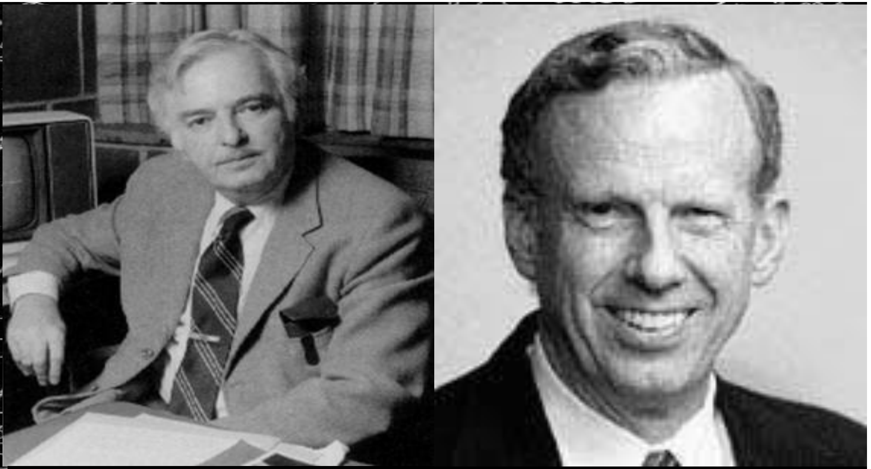
Applying the Consensus Algorithm

Evaluation

Consensus Ranking



Consensus Ranking ≈ 20th Century



John Kemeny & Peyton Young
Kemeny-Young Method - 1959

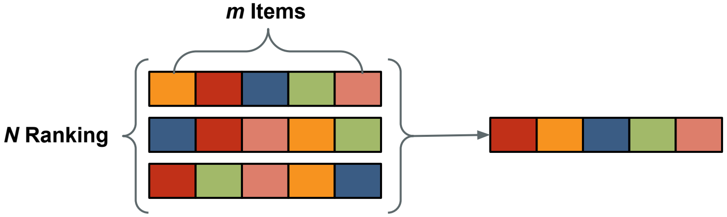
Applications

Election

Social Science

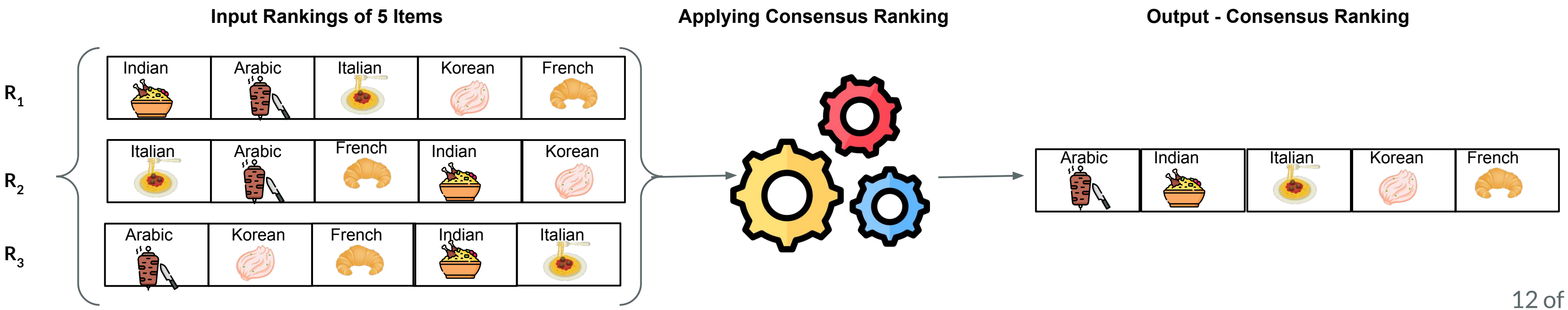
Bioinformatics

Consensus Ranking



Finding a consensus ranking is **aggregating** a set of ***N*** different **rankings** of ***m* items** into one ranking that **orders** the *m* items **closest** to all of the *N* rankings within a specified distance

Example



Input Rankings of 5 Items

R_1	Indian	Arabic	Italian	Korean	French
R_2	Italian	Arabic	French	Indian	Korean
R_3	Arabic	Korean	French	Indian	Italian

Applying Consensus Ranking

Output - Consensus Ranking

Arabic	Indian	Italian	Korean	French

Finding a Consensus Ranking

1 **Generalised Kendall-τ Distance**

Measures the distance between every two rankings in the set of rankings

$$G(R,C) = \#\{(i,j) : i < j \wedge \\ ((R[i] < R[j] \wedge C[i] > C[j]) \vee (R[i] > R[j] \ \wedge \ C[i] < C[j]) \vee \\ (R[i] \neq R[j] \wedge C[i] = C[j]) \vee (R[i] = R[j] \ \wedge \ C[i] \neq C[j]))\}$$

2 **Generalised Kemeny Score**

Finds the sum of the generalised Kendall-τ distance between a ranking and every ranking in the set

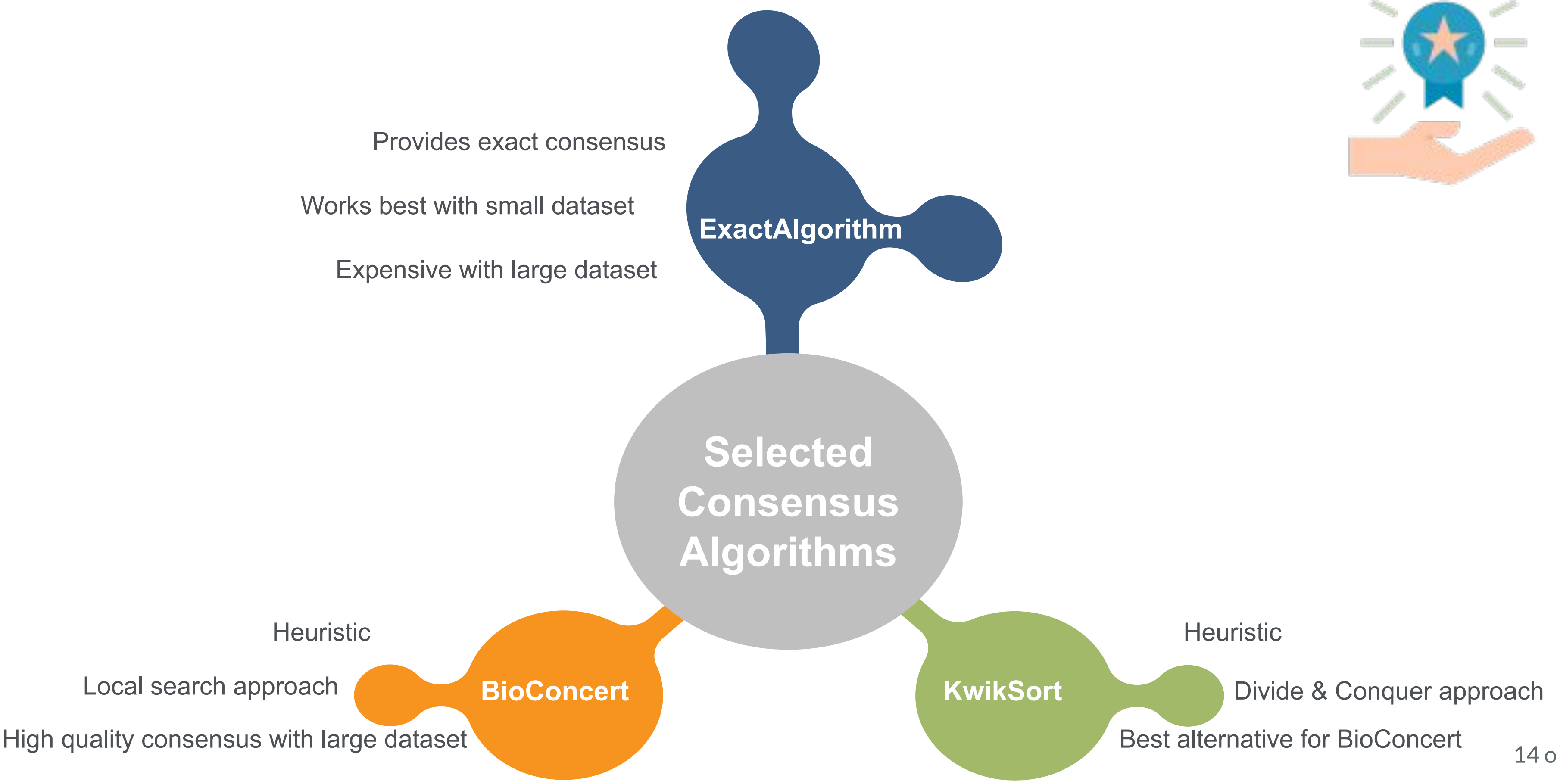
$$\sum_{C \in \mathcal{R}} G(R,C)$$

3 **Consensus Ranking**

Finds the ranking R^* with the smallest generalised Kemeny Score

$$K(R^*, \mathcal{R}) \leq K(R, \mathcal{R})$$

Consensus Algorithms



Provides exact consensus

Works best with small dataset

Expensive with large dataset

Heuristic

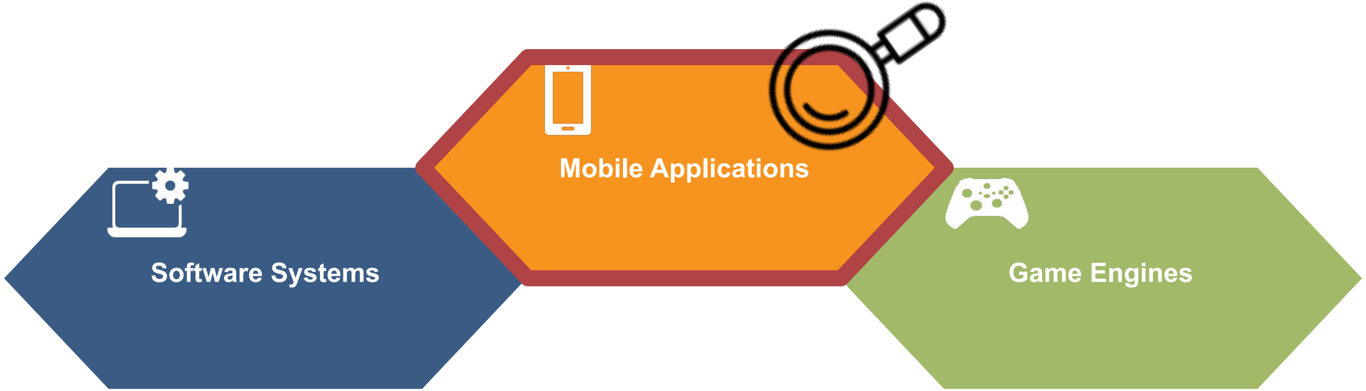
Local search approach

High quality consensus with large dataset

Heuristic

Divide & Conquer approach

Best alternative for BioConcert



Software Systems

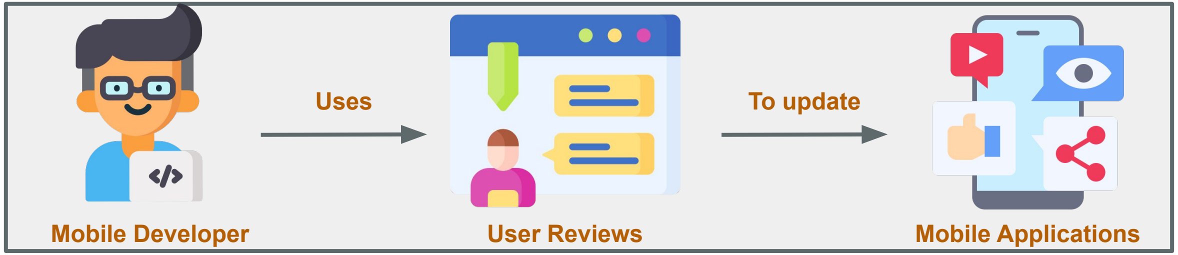
Mobile Applications

Game Engines

Objective

Test and validate the applicability of the consensus algorithm on **mobile app data type** to address **mobile application issues**

Context



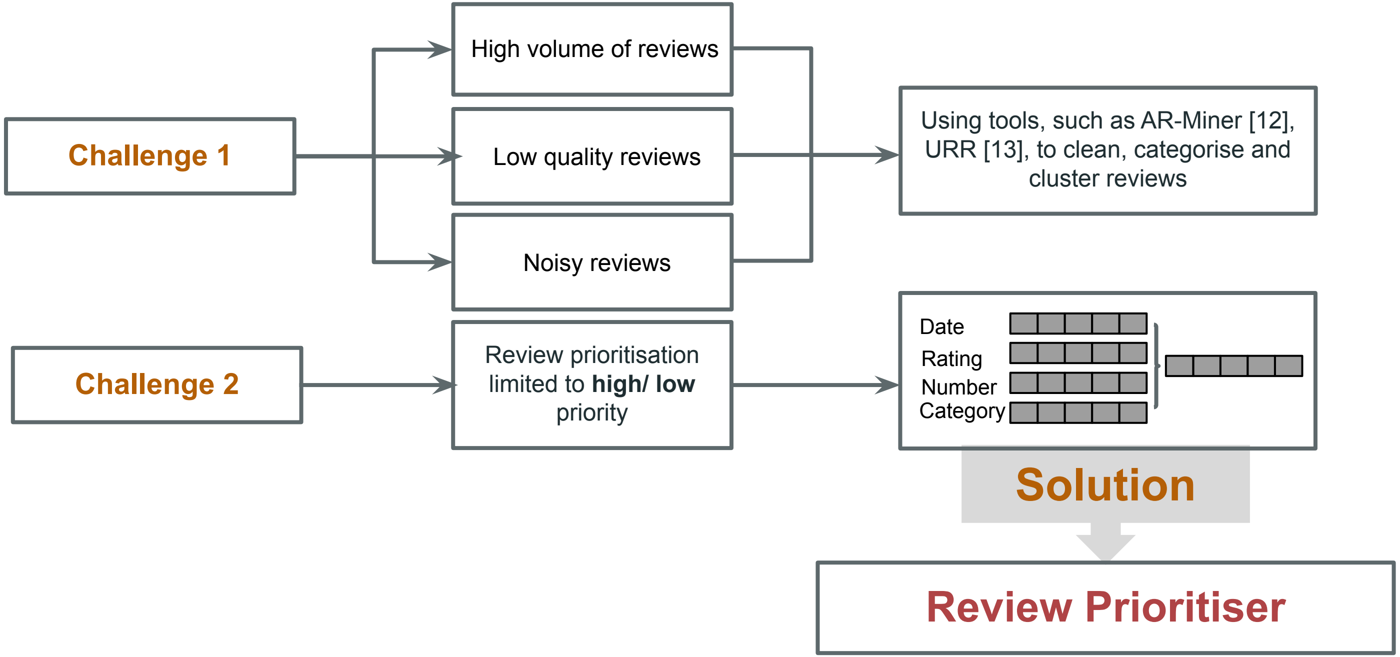
Mobile Developer

Uses

User Reviews

To update

Mobile Applications



Challenge 1

High volume of reviews

Low quality reviews

Noisy reviews

Using tools, such as AR-Miner [12], URR [13], to clean, categorise and cluster reviews

Challenge 2

Review prioritisation limited to **high/ low** priority

Date

Rating

Number

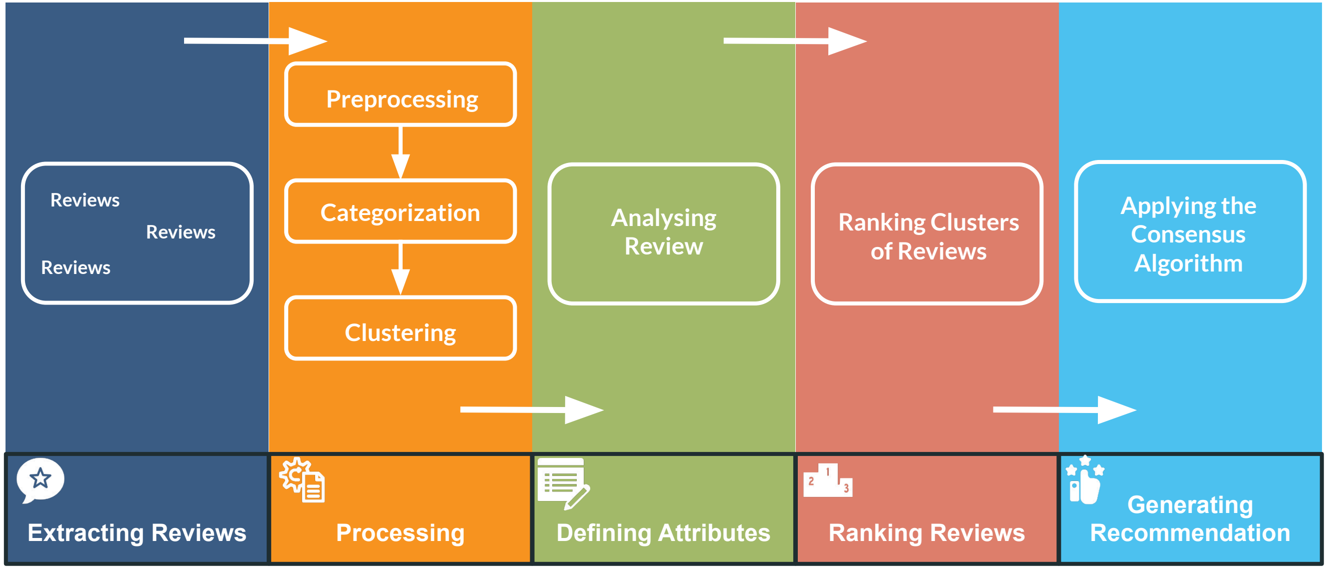
Category

Solution

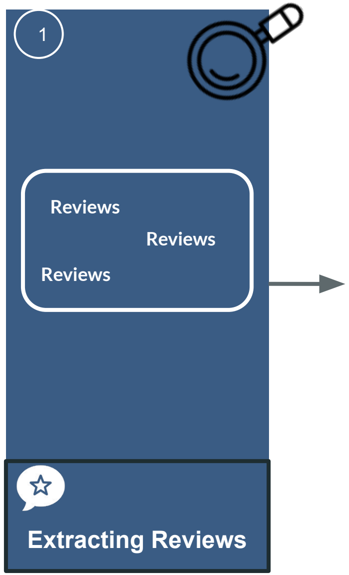
Review Prioritiser

Review Prioritiser (RP)

An approach that uses the **consensus algorithm** to recommend a **prioritised list of user reviews** that could help app developers **planning the next releases** of their apps



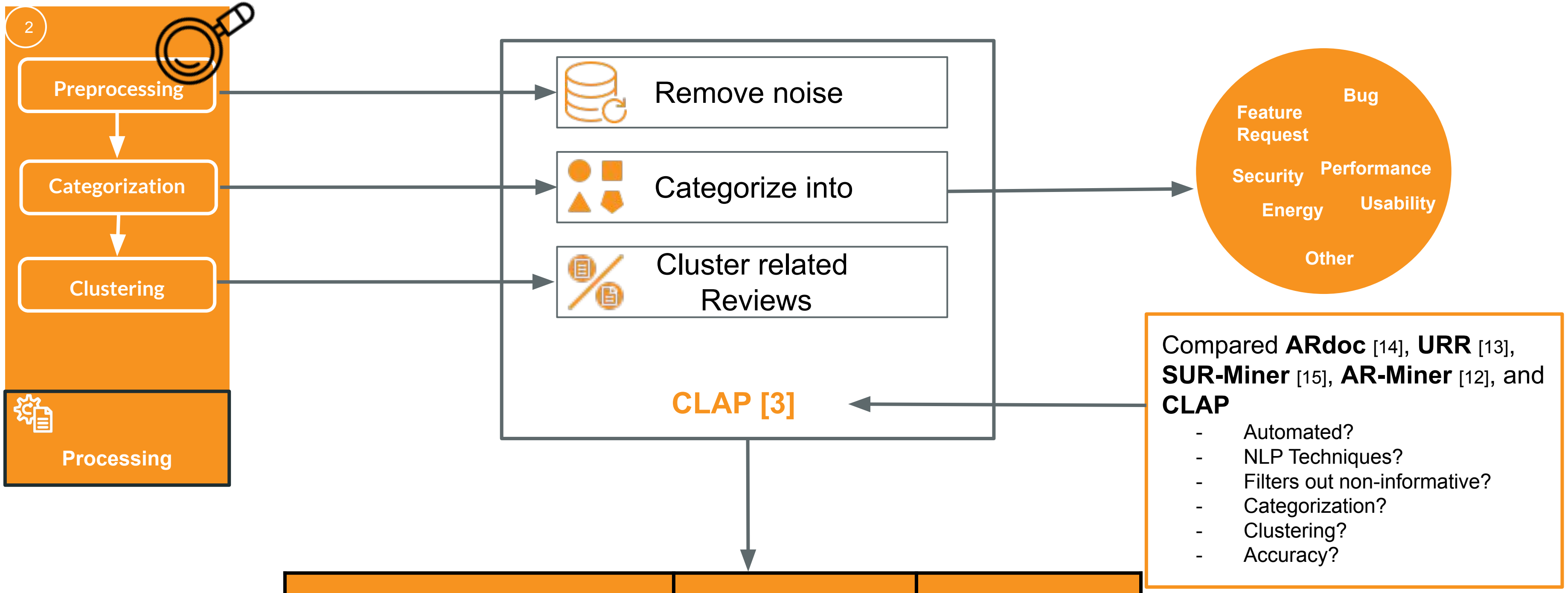
The approach- Extracting Reviews



Publicly Available Dataset [3]
725 Reviews & 14 apps

App Name	Category
BOINC	Education
Lightning Web Browser	Communication
Harvest Moon	Game
Timeriffic	Tools
iFixit	Lifestyle
DuckDuckGo	Tools
eBay	Shopping
Barcode Scanner	Shopping
Ringdroid	Music
2048	Puzzle
Viber	Communication
Dolphin Emulator	Arcade
LinePhone	Communication
WordPress	Productivity

The approach- Processing



2

Preprocessing

Categorization

Clustering

Processing

Remove noise

Categorize into

Cluster related Reviews

CLAP [3]

Bug

Feature Request

Security

Performance

Energy

Usability

Other

Compared **ARdoc** [14], **URR** [13], **SUR-Miner** [15], **AR-Miner** [12], and **CLAP**

- Automated?
- NLP Techniques?
- Filters out non-informative?
- Categorization?
- Clustering?
- Accuracy?

Review	Category	Cluster
Stats broken in last update	Bug	C10
It is slow and buggy	Performance	C17
I need the justify post feature	Feature	C1

The approach- Defining Attributes

3
Analysing Review
Defining Attributes

Cardinality	Number of reviews in each cluster, decreasingly
Oldest Date	Oldest date of review to newest
Average Rating	Low average rating to high
Category	Important to less important

Attributes are references that developers use to order their clusters of reviews

Mobile Applications

The approach- Ranking Reviews

4
Ranking Clusters of Reviews
2 1 3
Ranking Reviews

Image upload error

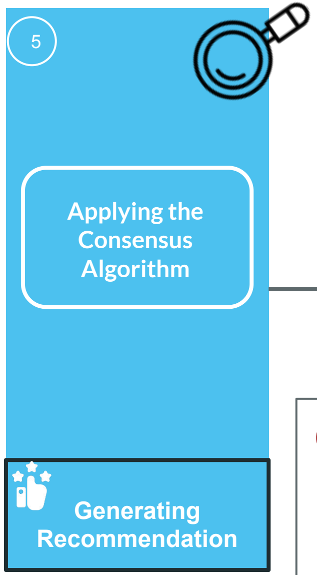
Navigation difficulty

Attribute	Ranked Clustered
Cardinality	[[9],[8,12],[1,4,14,18],[2,5,6,7,10,11,17,3,13,15,16]]
Date	[[2],[9,1,4,14,18,5],[6,10],[12],[8,15],[11,3,13,16],[7,17]]
Average Rating	[[16],[12],[9],[18,15,3,13],[14,11,17],[4],[5,6,10,8,7],[2,1]]
Category	[[16,12,9,15,13,14,11,10,8],[17],[18],[3,4,5,6,7,2,1]]

Ranked Clusters of WordPress reviews

The approach- Applying the Consensus

Mobile Applications



Consensus Algorithms

BioConcert | ExactAlgorithm | KwikSort

R1	[[9],[8,12],[1,4,14,18],[2,5,6,7,10,11,17,3,13,15,16]]
R2	[[2],[9,1,4,14,18,5],[6,10],[12],[8,15],[11,3,13,16],[7,17]]
R3	[[16],[12],[9],[18,15,3,13],[14,11,17],[4],[5,6,10,8,7],[2,1]]
R4	[[16,12,9,15,13,14,11,10,8],[17],[18],[3,4,5,6,7,2,1]]

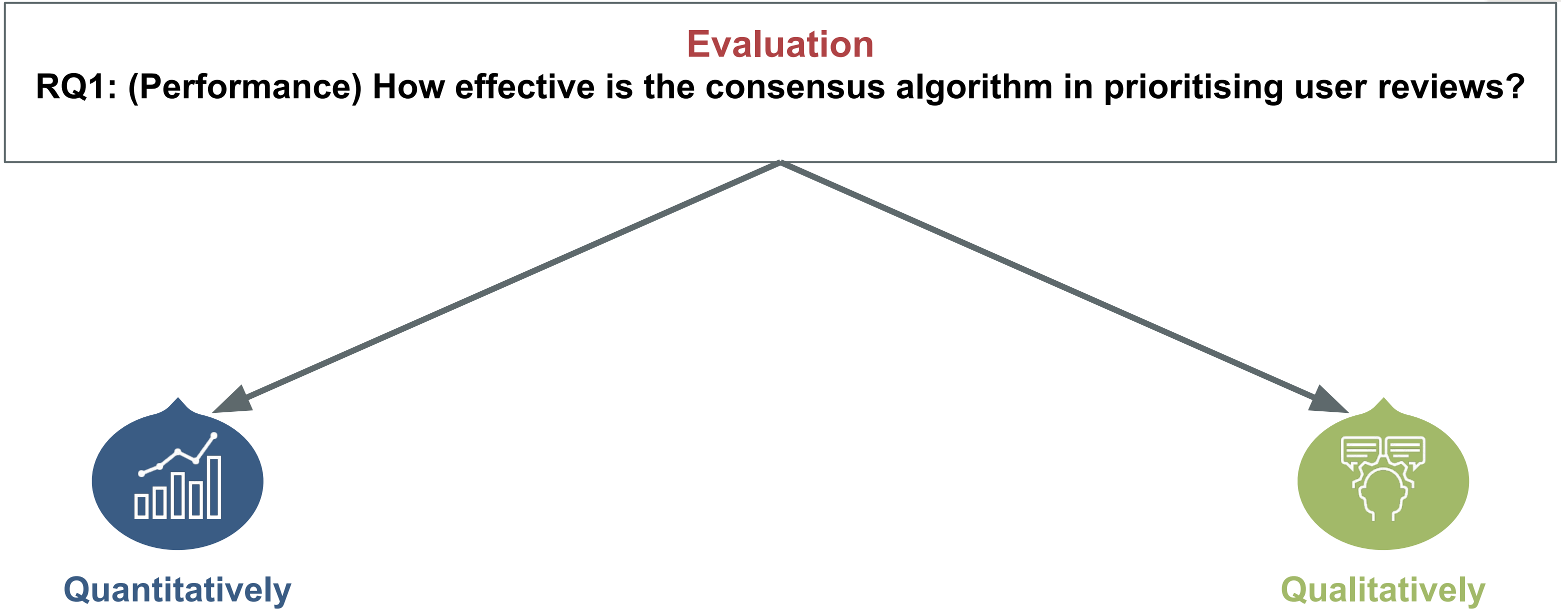
Ranked Clusters of WordPress reviews

Consensus Ranking =

[[9],[12],[14,18],[8],[1,4],[10,11,3,13,15,16],[17],[2,5,6,7]]

[[Upload error], [App login issue], [HTML tags shown, Navigation difficulty], [Web login issue], [Add justify feature, Add text formatting], [Broken stats, Posts hidden, Multiple post edit, Update removes website, Posts not published, Installation issue], [slow], [Add featured image, Allow profile edit, Allow uploading image to posts, Black background theme]]

Evaluation & Results



Evaluation

RQ1: (Performance) How effective is the consensus algorithm in prioritising user reviews?

Quantitatively

Qualitatively

Mobile Applications

Quantitative Evaluation

Kendall Rank Correlation Coefficient

Consensus Ranking & Gold Ranking

Gold Ranking a prioritized ranking of clustered of reviews, defined manually by the apps' developers.

Gold Ranking for WordPress Application

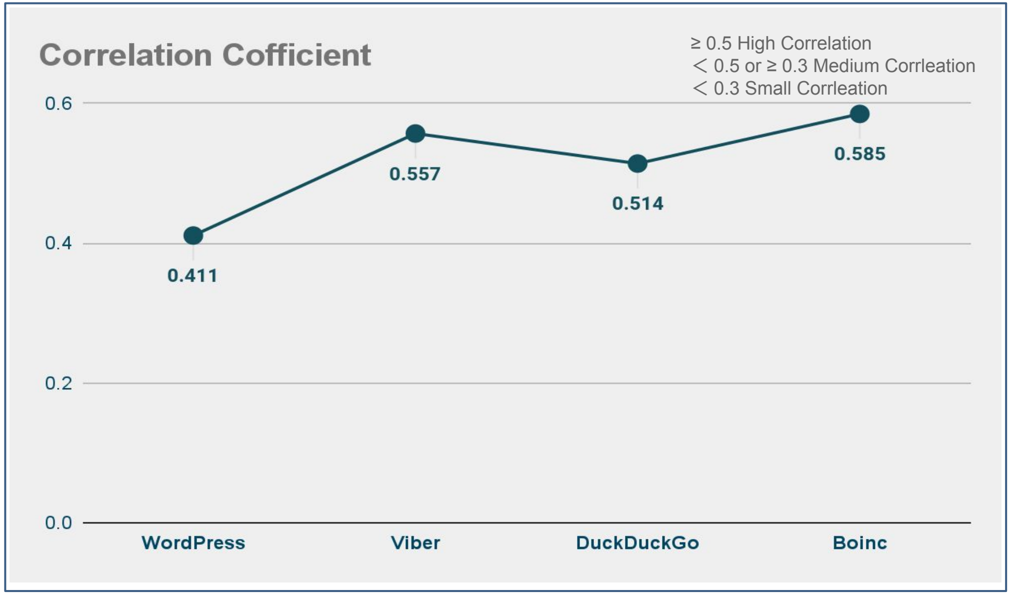
$\mathcal{GR}$ = [[12],[16],[13],[8],[14],[15],[11],[10],[9],[18],[17],[6],[4],[1],[2],[3],[5],[7]]

[[App login issue], [Installation issue], [Update removes website], [Web login issue], [HTML tags shown], [Posts not published], [Posts hidden], [Broken stats], [Upload error], [Navigation difficulty], [slow], [Allow uploading image to posts], [Add text formatting], [Add justify feature], [Add featured image], [Multiple post edit], [Allow profile edit], [Black background theme]]

Quantitative Evaluation

Mobile Applications

RQ1: (Performance) How effective is the consensus algorithm in prioritising user reviews?



The strong and positive correlation demonstrates that the proposed consensus-based technique is effective at prioritising user reviews

Qualitative Evaluation

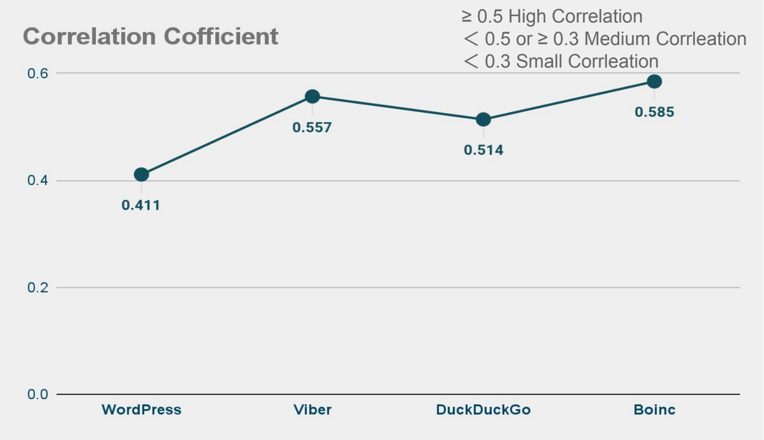
Mobile Applications

“if your app reviews were prioritised in such a consensus ranking, do you believe this ranking would help you plan a successful release and would you plan your next release according to this ranking?'”

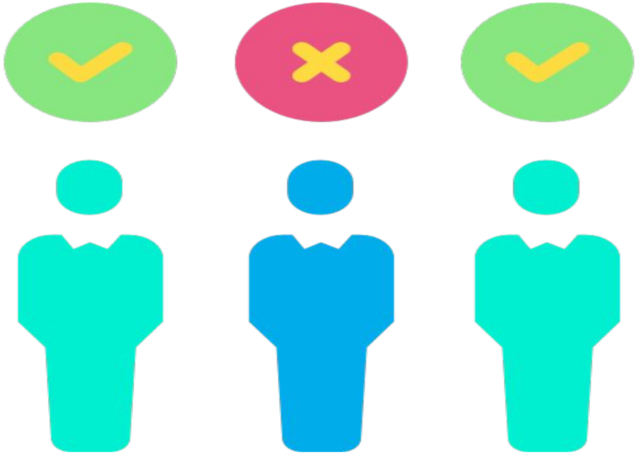
The majority of developers agreed that the consensus-based algorithm generates a meaningful consensus ranking and they would use to plan their next releases

To Conclude

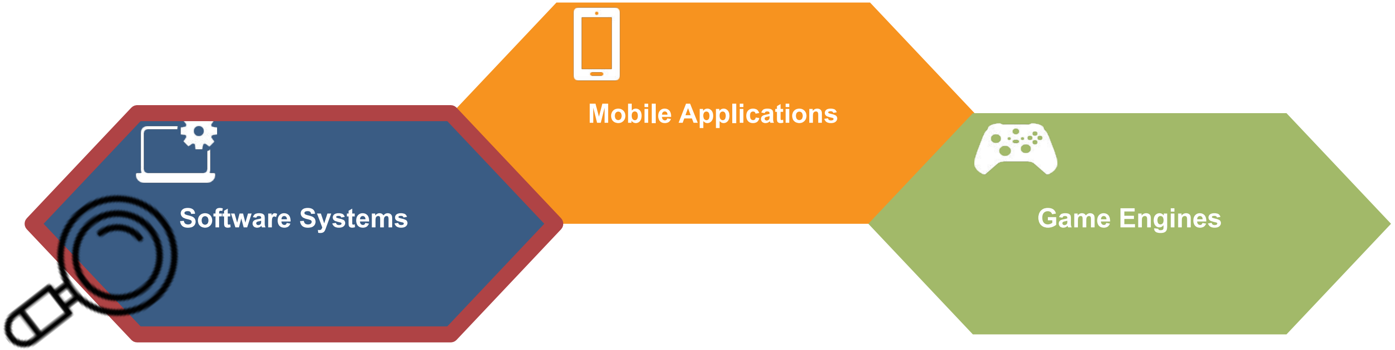
High Correlation



Majority of developers agreed with the recommended prioritization



The consensus-based recommendation technique is effective at prioritising mobile app user reviews and can assist developers in improving their applications.



Software Systems

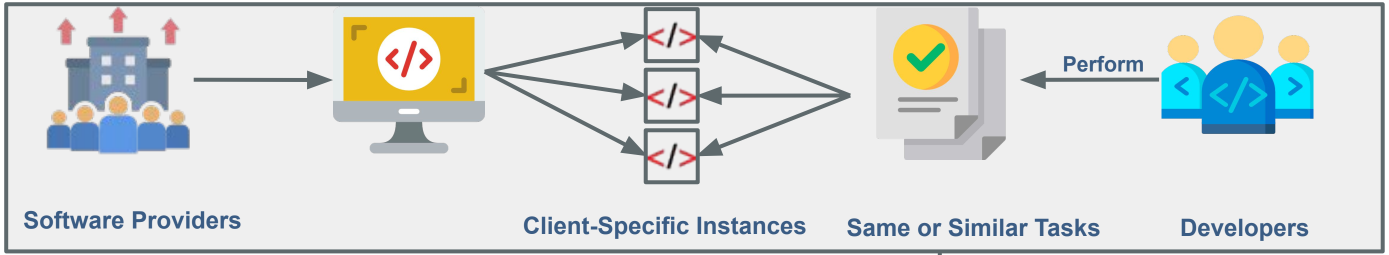
Mobile Applications

Game Engines

Objective

Evaluate the ability of the consensus algorithm to deal with **software data type** to address **software system-related issues**

Context



Tasks that can be implemented on some client's instance and require developers to interact with the same or similar source-code file(s)

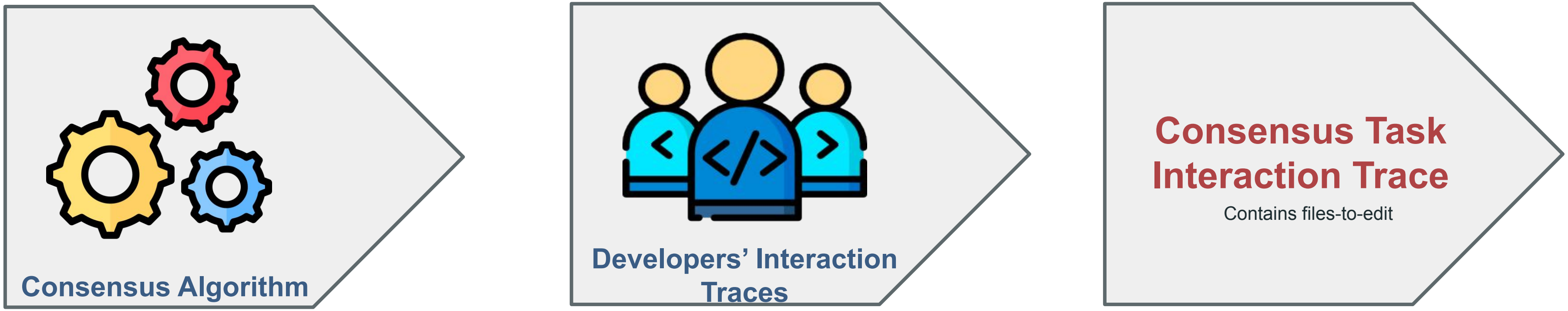
Challenge → More Customization → Complex Software → Difficult to comprehend and maintain

Solution

Consensus Task Interaction Trace Recommender

Consensus Task Interaction Trace Recommender (CITR)

A task-based recommendation approach that uses the **consensus algorithm** to recommend **file(s)-to-edit** based on an aggregated set of **developers' interaction traces**. It helps developers **complete development tasks** successfully in less time, and hence **increases their productivity.**



Consensus Algorithm

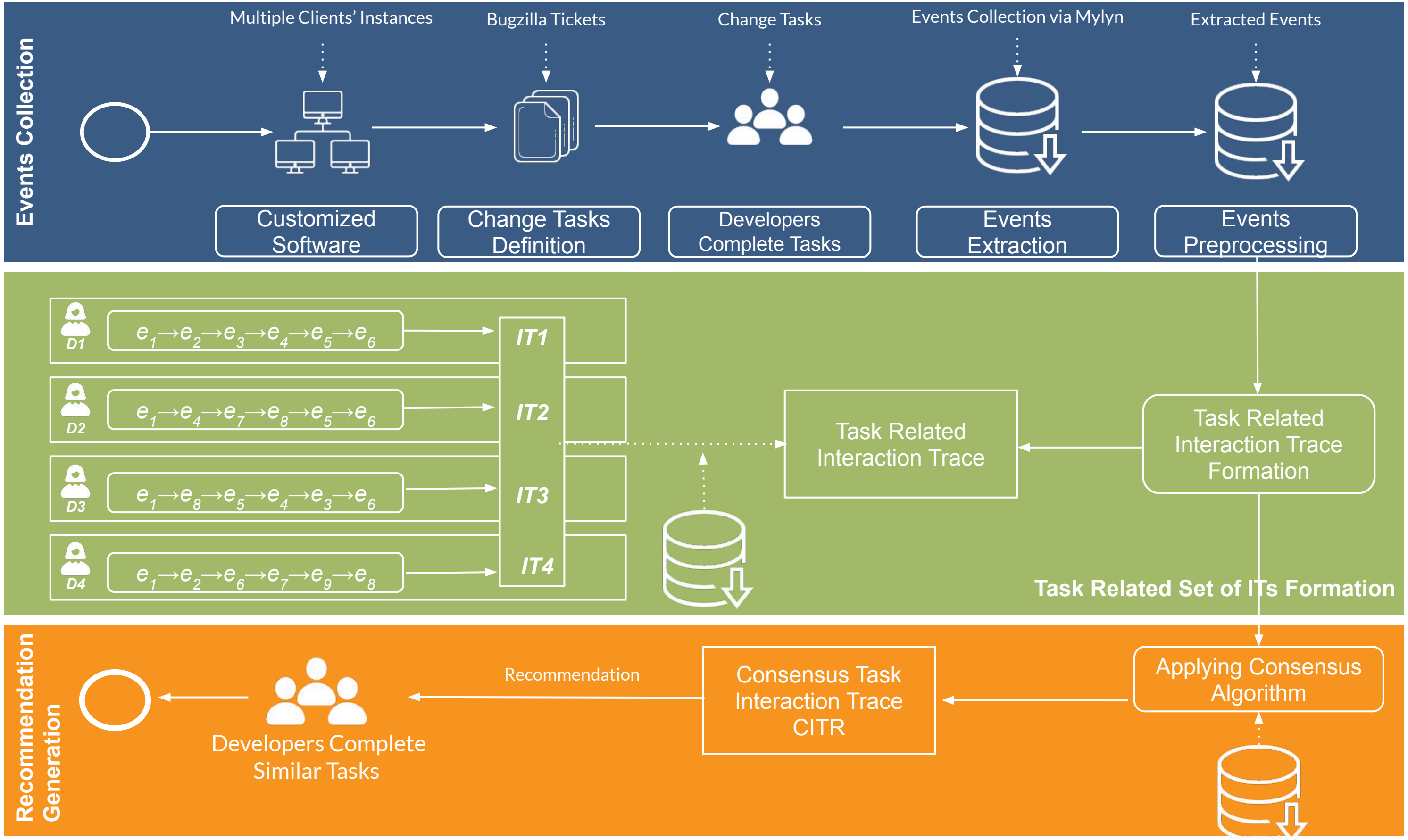
Developers’ Interaction Traces

Consensus Task Interaction Trace

Contains files-to-edit

Consensus Task Interaction Trace Recommender (CITR)

User-Involved Experiment



Events Collection

Multiple Clients' Instances

Bugzilla Tickets

Change Tasks

Events Collection via Mylyn

Extracted Events

Customized Software

Change Tasks Definition

Developers Complete Tasks

Events Extraction

Events Preprocessing

D1 $e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4 \rightarrow e_5 \rightarrow e_6$ IT1

D2 $e_1 \rightarrow e_4 \rightarrow e_7 \rightarrow e_8 \rightarrow e_5 \rightarrow e_6$ IT2

D3 $e_1 \rightarrow e_8 \rightarrow e_5 \rightarrow e_4 \rightarrow e_3 \rightarrow e_6$ IT3

D4 $e_1 \rightarrow e_2 \rightarrow e_6 \rightarrow e_7 \rightarrow e_9 \rightarrow e_8$ IT4

Task Related Interaction Trace

Task Related Interaction Trace Formation

Task Related Set of ITs Formation

Recommendation Generation

Developers Complete Similar Tasks

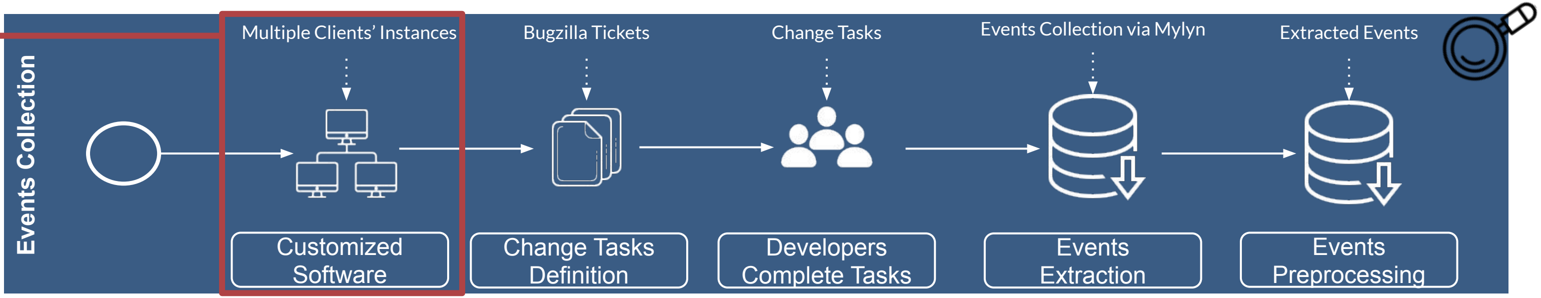
Recommendation

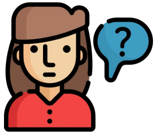
Consensus Task Interaction Trace CITR

Applying Consensus Algorithm

The Approach - System Selection

Eclipse-based plugin, **PDE** (Plug-in Development Environment)

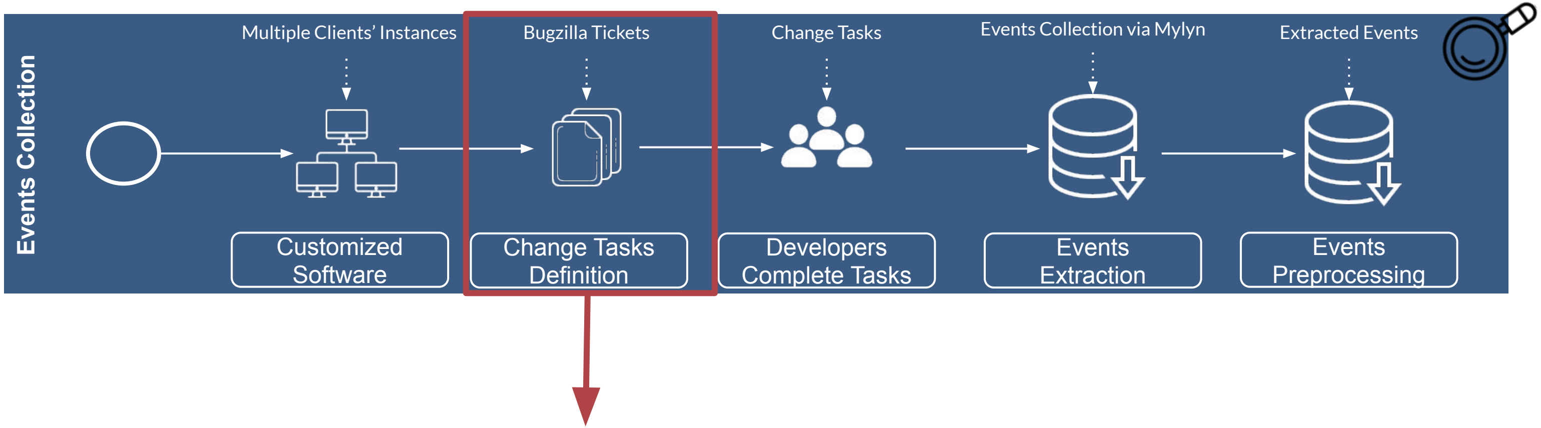




But why this one?

- Open source
- Large base code
- Used in many research studies
- It has a bug tracking system
- Provides Mylyn solution ITs

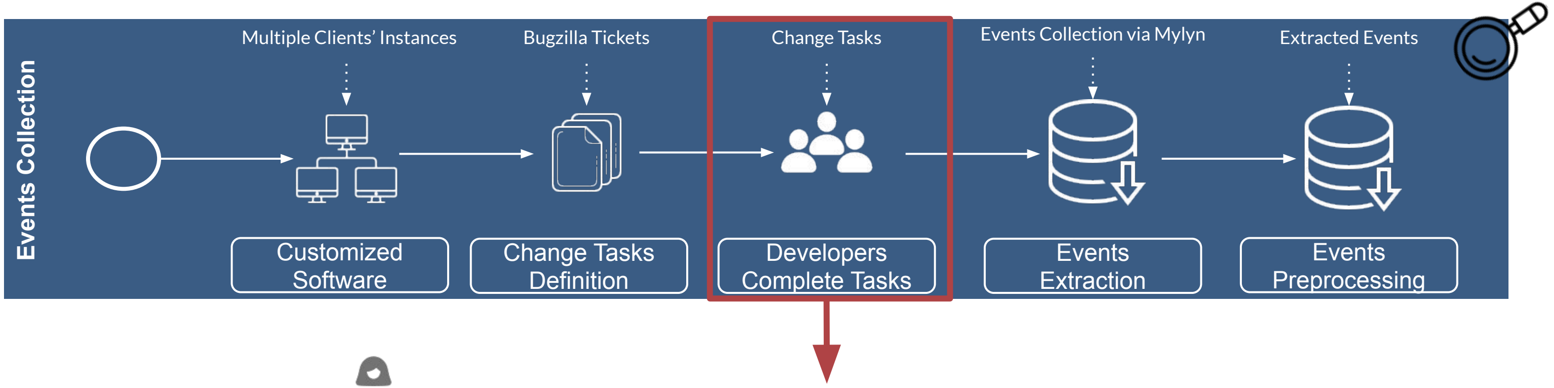
The Approach - Change Tasks Definition



Bugzilla Ticket	Task
304028	Feature properties dialog window has no title
229024	A tab on the overview page shows "?" Instead of API Information
265931	Autostart values are not persisted correctly on the plug-in

Change Tasks Used in the Experiment

The Approach - Tasks Completion

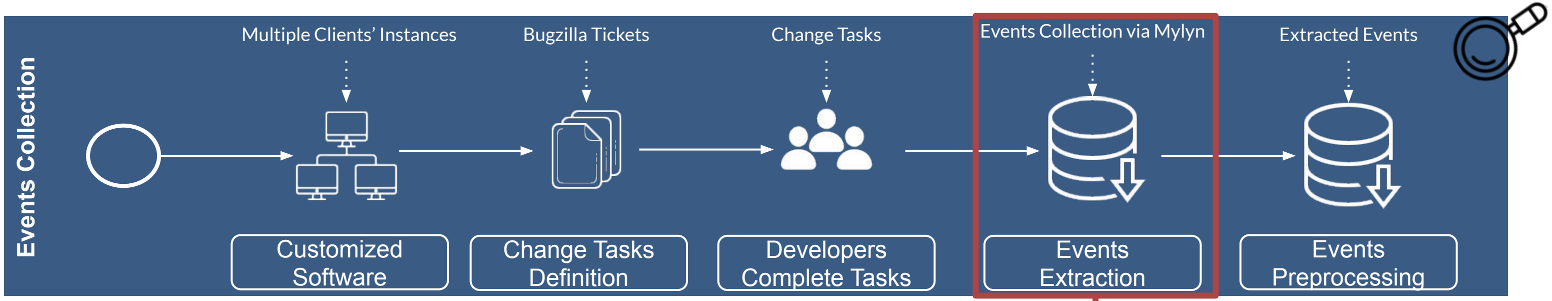


Participants perform Tasks using Eclipse IDE

- 3+ Java experience
- 1+ IDE experience
- 2 female, 5 male
- 1 Post Doc, 3 Ph.D., 3 M.Sc.
- 1-5 years professional experience

Participants Characteristics

The Approach - Events Extraction



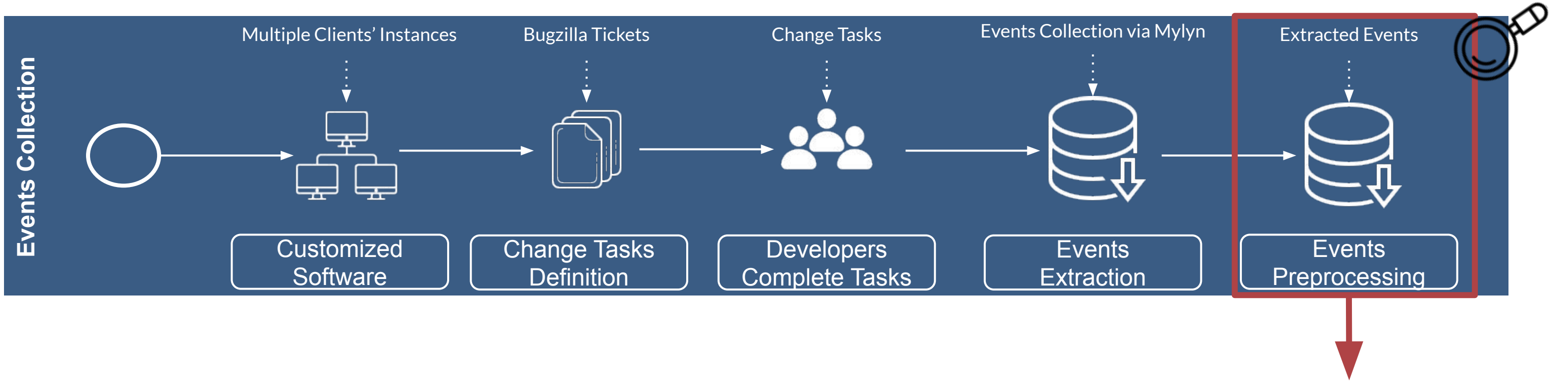
Total of **2390** events

Events are participants' activities on source-code elements (i.e., opening, searching, editing, etc.)

StartDate	EndDate	StructureHandle	Kind
2018-08-08 11:43:44.97	2018-08-08 11:46:09.716	FeatureSection.java	Selection
2018-08-08 11:46:46.918	2018-08-08 11:53:39.320	FeatureSection.handleProperties();	Edit

Example of Extracted Events

The Approach - Events Preprocessing



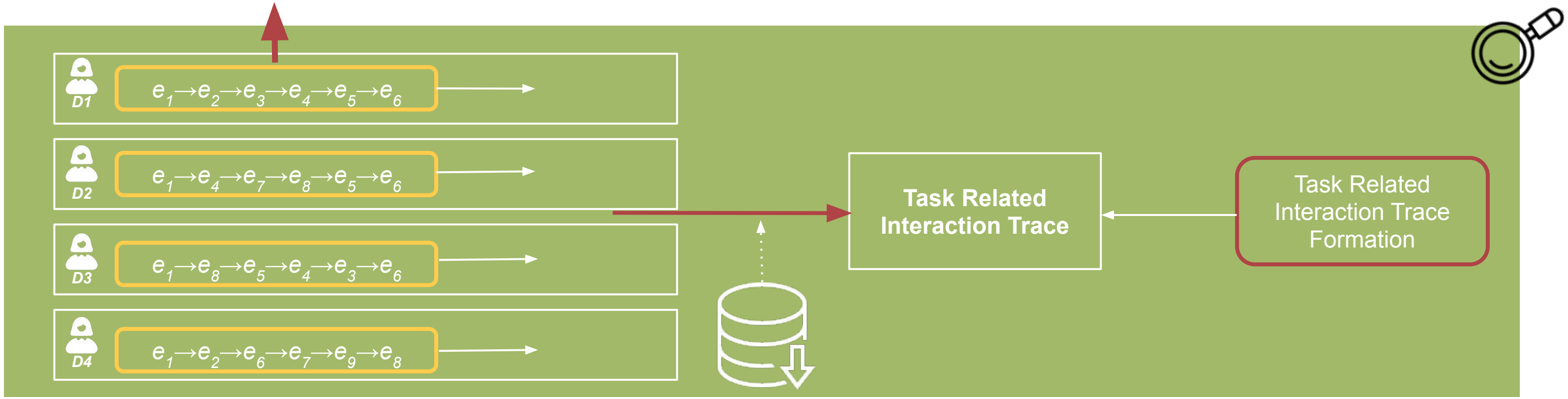
A regular expression based **tool** to remove **noise**, unrelated **JAR files**, **duplicate** events and events with **0-duration**

StructureHandle: =org.eclipse.pde.ui/src&lt;org.eclipse.pde.internal.ui.editor.product{VersionDialog.java[VersionDialog~configureShell~QShell;
CompleteName: org.eclipse.pde.ui.src.org.eclipse.pde.internal.ui.editor.product.VersionDialog.java.VersionDialog.configureShell.QShell

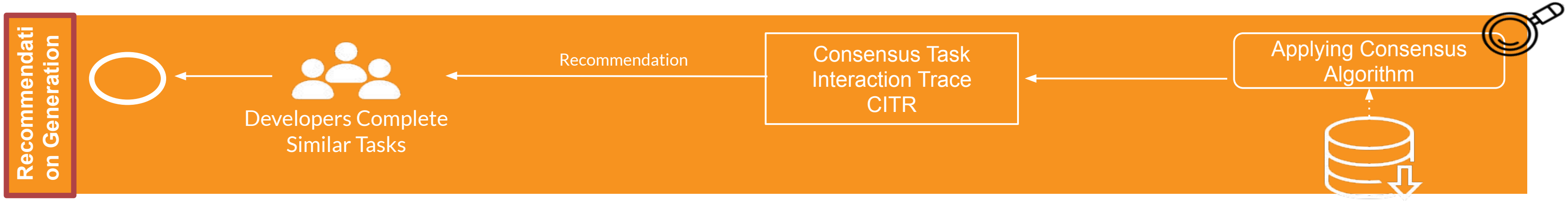
An Event Before & After Noise Removal

The Approach - Task Interaction Trace Formation

E.g. Change Task 1
Participant's set of events from Change Task 1


D1
$e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4 \rightarrow e_5 \rightarrow e_6$
D2
$e_1 \rightarrow e_4 \rightarrow e_7 \rightarrow e_8 \rightarrow e_5 \rightarrow e_6$
D3
$e_1 \rightarrow e_8 \rightarrow e_5 \rightarrow e_4 \rightarrow e_3 \rightarrow e_6$
D4
$e_1 \rightarrow e_2 \rightarrow e_6 \rightarrow e_7 \rightarrow e_9 \rightarrow e_8$
Task Related Interaction Trace
Task Related Interaction Trace Formation

The Approach - Applying the Consensus


Recommendation Generation
Developers Complete Similar Tasks
Recommendation
Consensus Task Interaction Trace CITR
Applying Consensus Algorithm

Consensus Algorithms - BioConcert

D1	[[1],[2],[3],[4],[5],[6],[7]]
D2	[[8],[4],[5],[6],[9],[10]]
D3	[[11],[12],[13],[14],[15],[16],[17],[18],[19],[20],[21],[4],[22],[23],[5],[6]]
D4	[[24],[25],[26],[27],[28],[29],[30],[31],[32],[12],[33],[34],[5],[6],[9],[10]]

Task-Related Interaction Trace

CITR =
[[4], [5], [6], [9, 10], [12]]

[4]	org.eclipse.pde.ui
[5]	Plugin-ConfigurationSection.java
[6]	Plugin-ConfigurationSection.java
[9]	Plug-inConfiguration.java
[10]	Plug-inConfiguration.java
[12]	PluginsTab.java

Translation of the results

Files-to-edit that can help developers complete the same or similar change tasks on other software instances

Evaluation & Results

Quantitatively

RQ1**:** To what degree does CITR recommend relevant files to given change tasks?

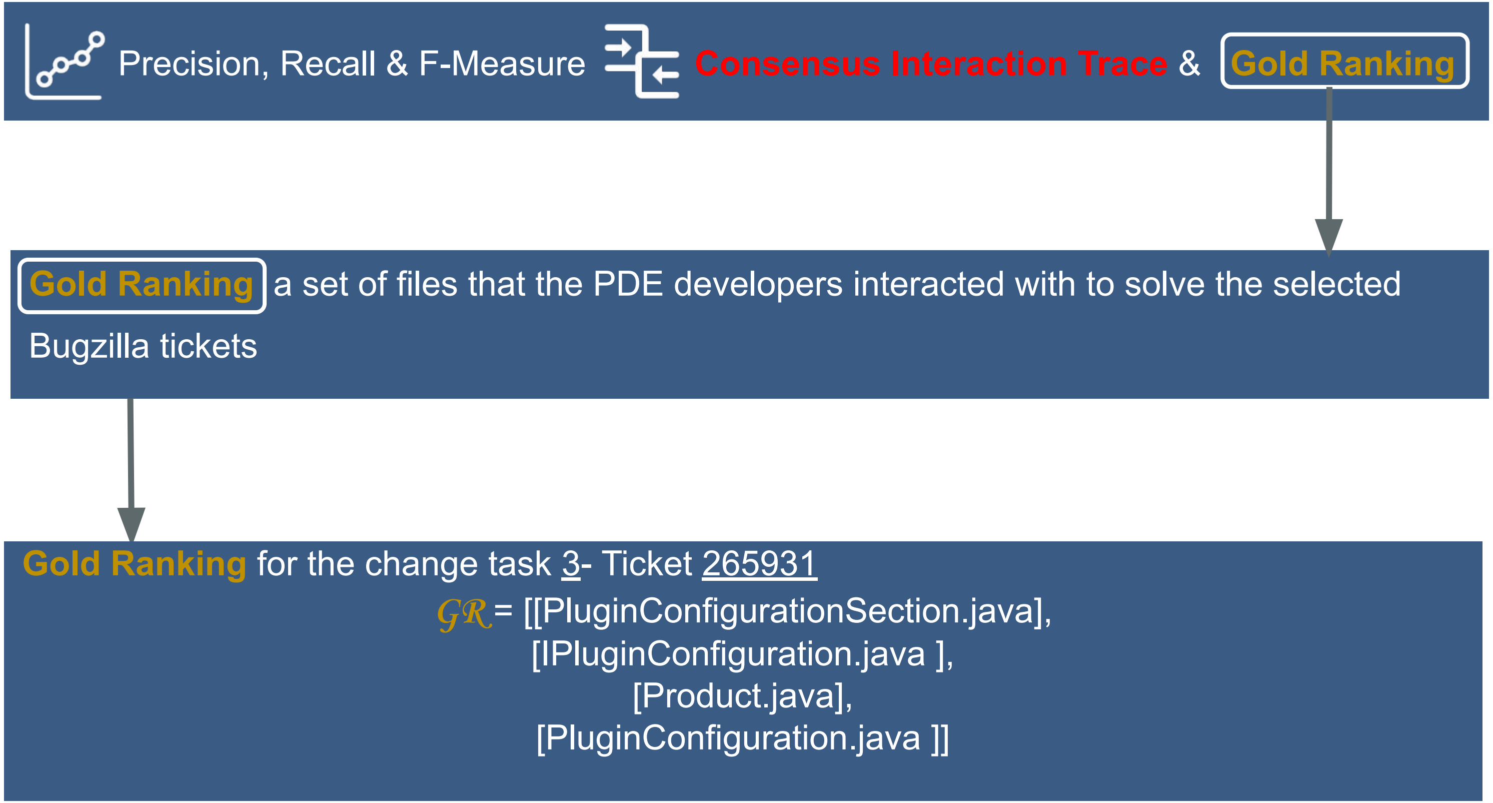
Qualitatively - Between- Subject Experiment

RQ2**:** Given a change task, can CITR help guide developers’ navigation paths to relevant file(s)-to-edit and increase their productivity?

Comparison

RQ3**:** How does CITR compare to MI (Mining Programmer Interaction Histories) in recommending relevant file(s)-to-edit for specific change tasks?

Quantitative



Precision, Recall & F-Measure Consensus Interaction Trace & Gold Ranking

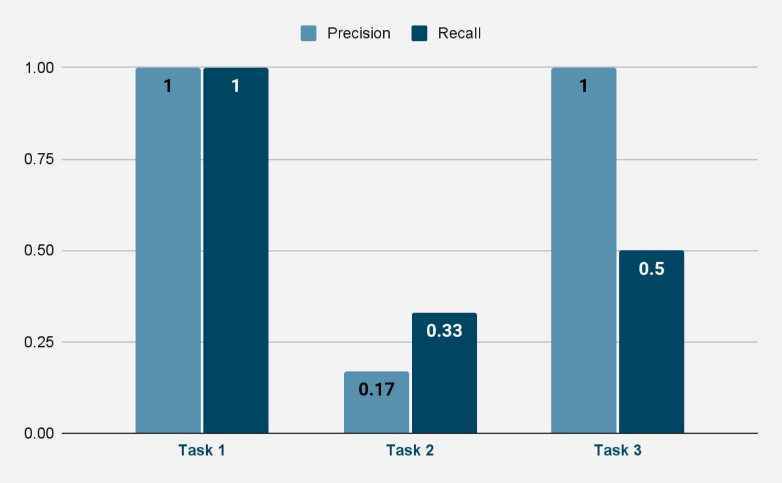
Gold Ranking a set of files that the PDE developers interacted with to solve the selected Bugzilla tickets

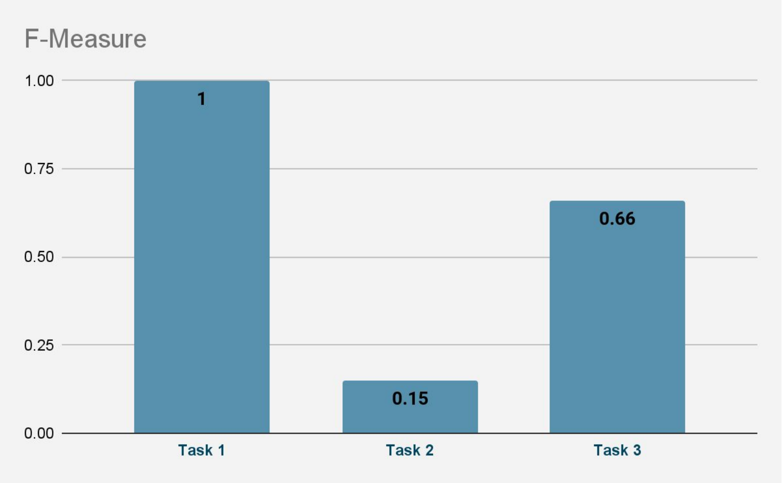
Gold Ranking for the change task 3- Ticket 265931

$\mathcal{GR}$ = [[PluginConfigurationSection.java],
[IPluginConfiguration.java],
[Product.java],
[PluginConfiguration.java]]

Quantitative

***RQ1*: To what degree does CITR recommend relevant files to given change tasks?**





Quantitative

***RQ1*: To what degree does CITR recommend relevant files to given change tasks?**

Gold Ranking	Consensus Interaction Trace
DocSection.java	ConfigurationTab.java
SchemaFormOutlinePage.java	MainTab.java
DocumentSection.java	PluginsTab.java
	TracingTab.java
	PDEUIMessages.java
	DocSection.java

Gold Ranking VS CITC for **Task 2**

CITR achieves high precision, recall, and F-measure and is able to recommend accurate and relevant file(s)-to-edit

Qualitative - Between-Subject Experiment

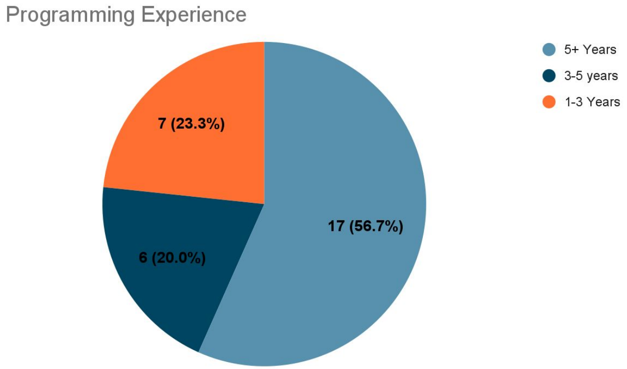
Goal understand to what extent these recommendations can help developers navigate and increase their productivity

Defining Evaluation Change Tasks = Tasks are similar in context to the change tasks

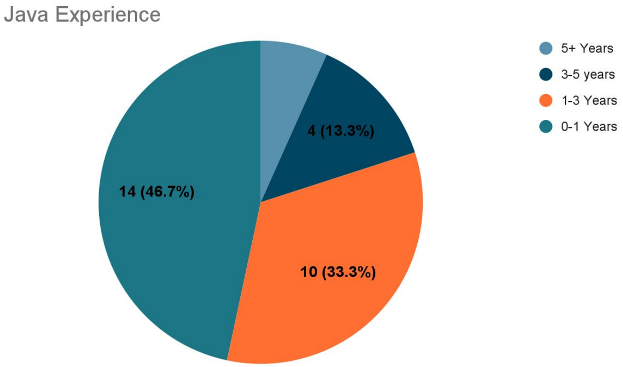
Bugzilla Ticket	Task
269618	Automatic wildcard on plug-ins
144533	Unnecessary white space on configuration tab
88003	Select all property
261878	Prompt to save changes on Plug-ins
171767	Large font on main tab
101516	Sort alphabetically property

Qualitative - Between-Subject Experiment

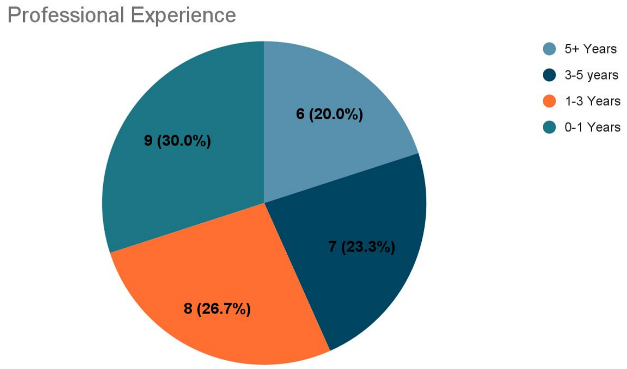
Inviting Developers = 30



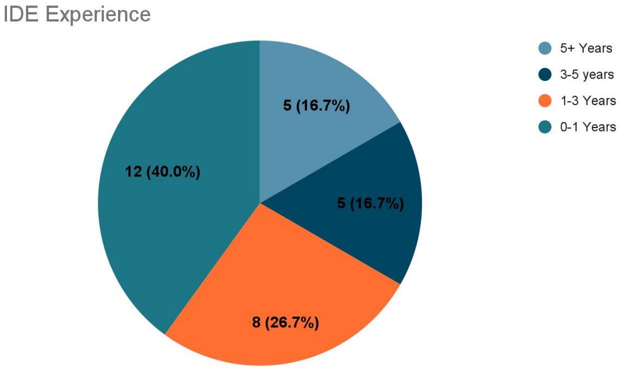
Programming Experience



Java Experience

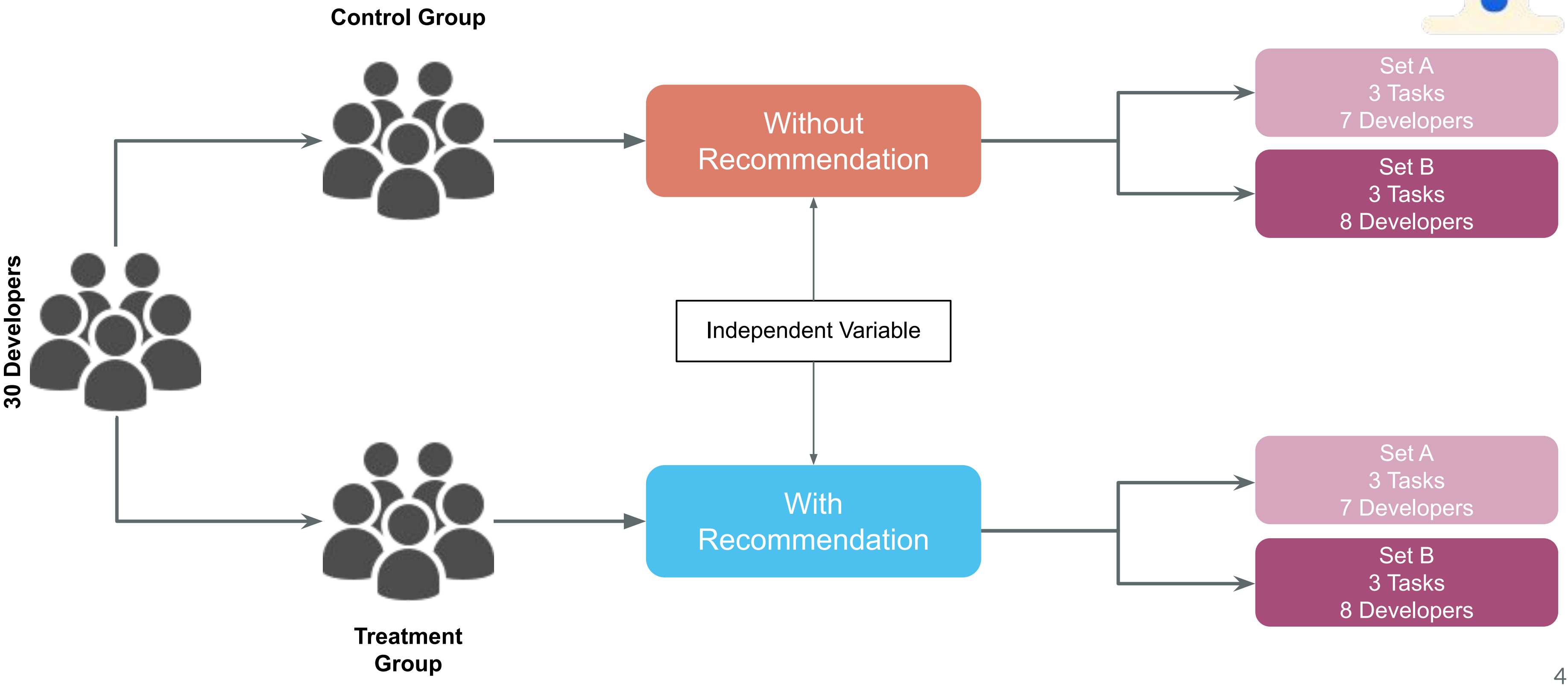


Professional Experience

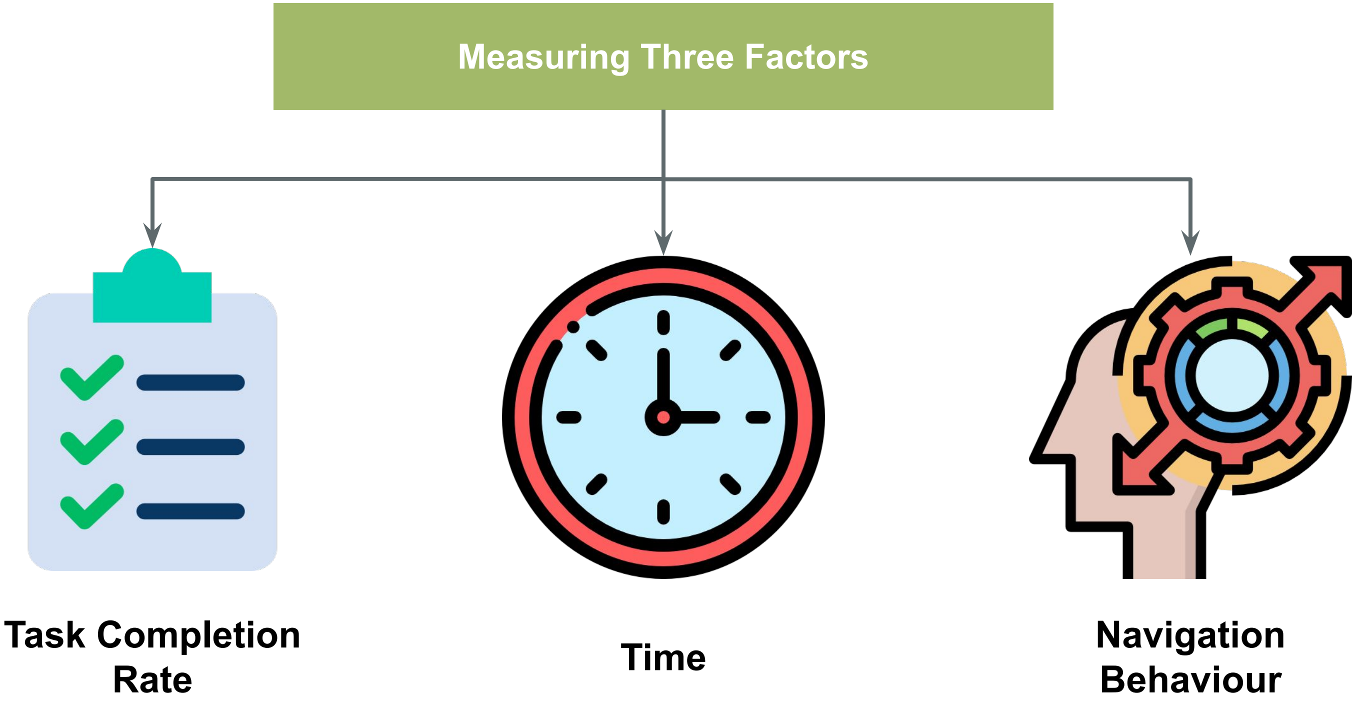


IDE Experience

Qualitative - Between-Subject Experiment



Qualitative - Between-Subject Experiment



Measuring Three Factors

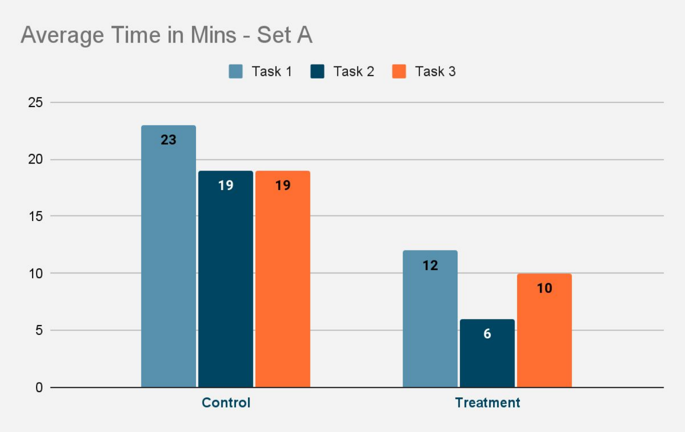
Task Completion Rate

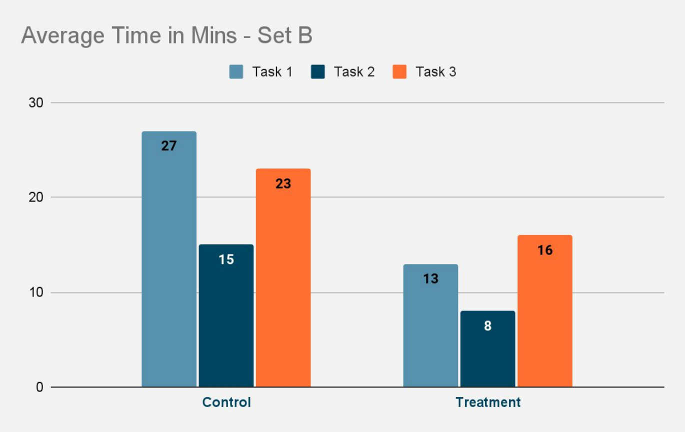
Time

Navigation Behaviour

Qualitative - Between-Subject Experiment

Time = Total time needed to complete each evaluation task

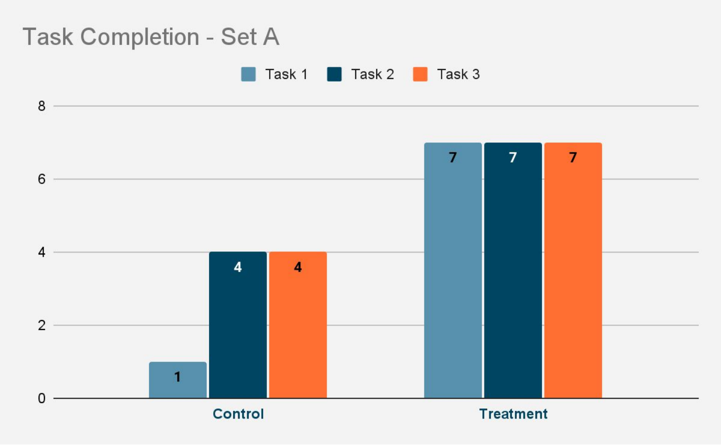


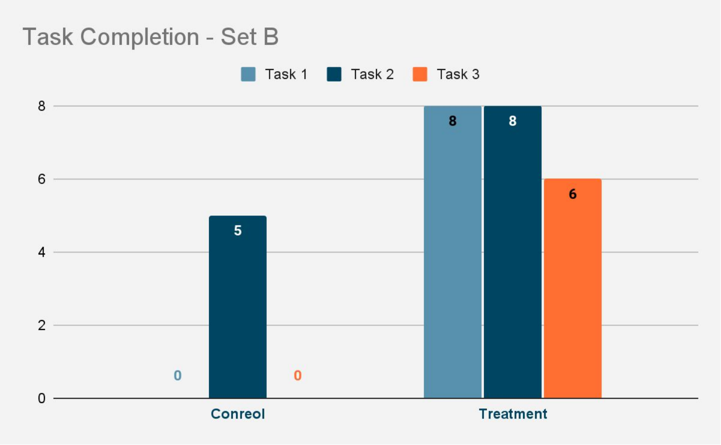


CITR recommendations help developers complete tasks in a shorter time

Qualitative - Between-Subject Experiment

Task Completion = Number of completed tasks.





CITR recommendations help developers complete tasks successfully

Qualitative - Between-Subject Experiment

Navigation Behaviour = Developers following a pattern of structured navigation through source-code and files

***RQ2*: Given a change task, can CITR help guide developers' navigation paths to relevant file(s)-to-edit and increase their productivity?**

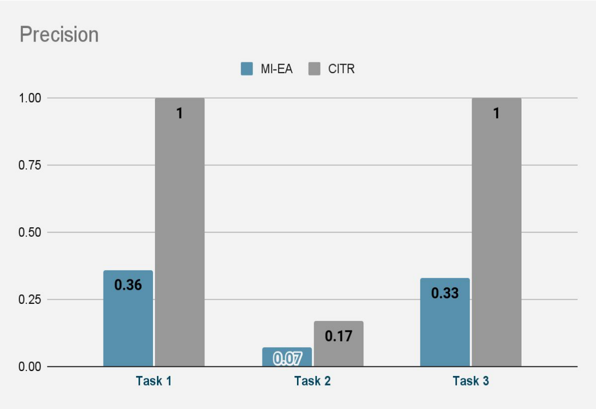
CITR recommendations increases developer's productivity by reducing navigation effort and time, and guiding them into structured navigation

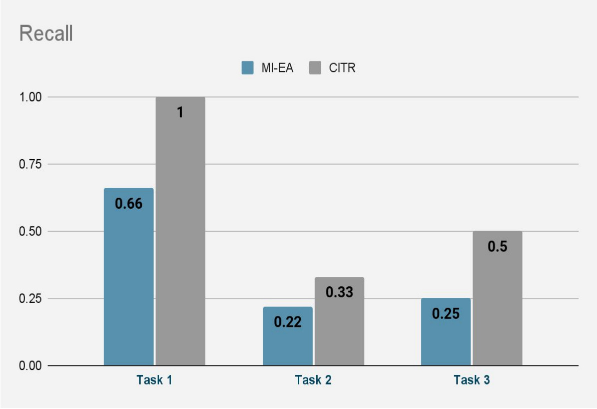
Software Systems

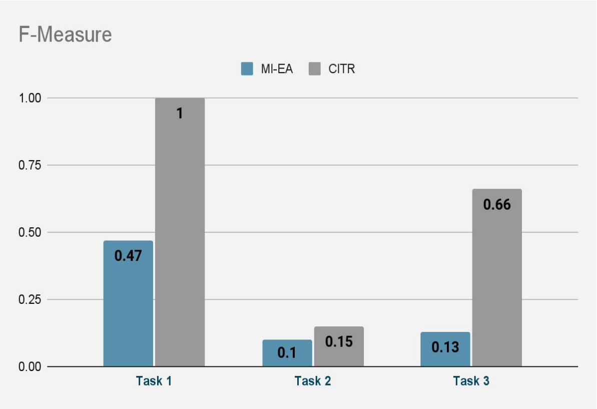
Comparison

***RQ3*: How does CITR compare to MI (Mining Programmer Interaction Histories) in recommending relevant file(s)-to-edit for specific change tasks?**

Goal: Compare **CITR** against **MI** (Mining Programmer Interaction Histories) [4]



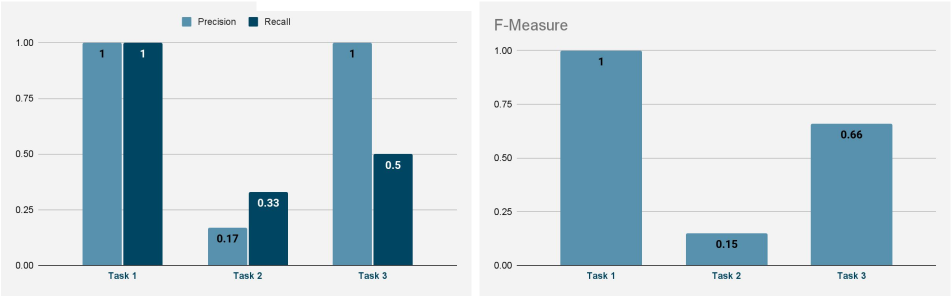




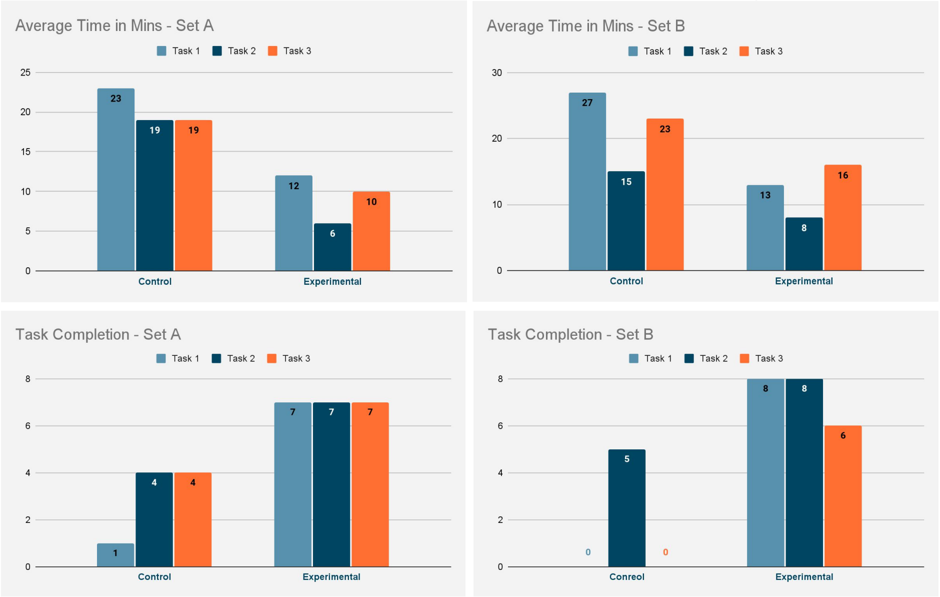
The comparison with MI showed that CITR yield better accuracy and relevant recommendations than MI

To Conclude

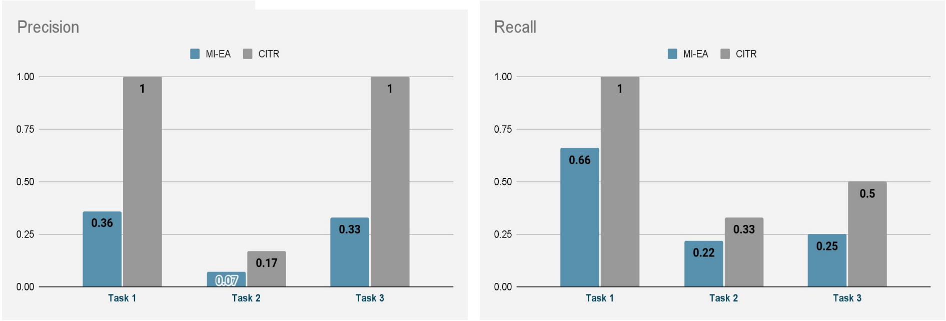
High Accuracy Results



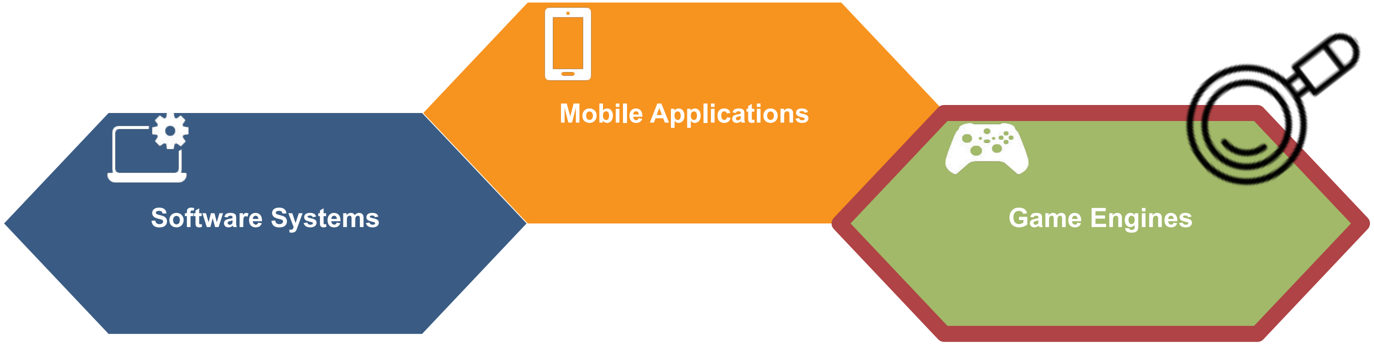
Increased Developers Productivity



Outperformed MI



The consensus-based recommendation technique is effective at providing relevant files to edit, which can boost developer productivity and outperform other approaches



Software Systems

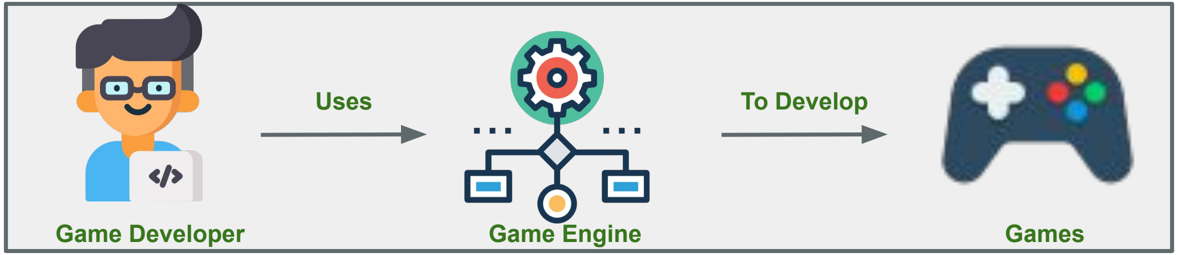
Mobile Applications

Game Engines

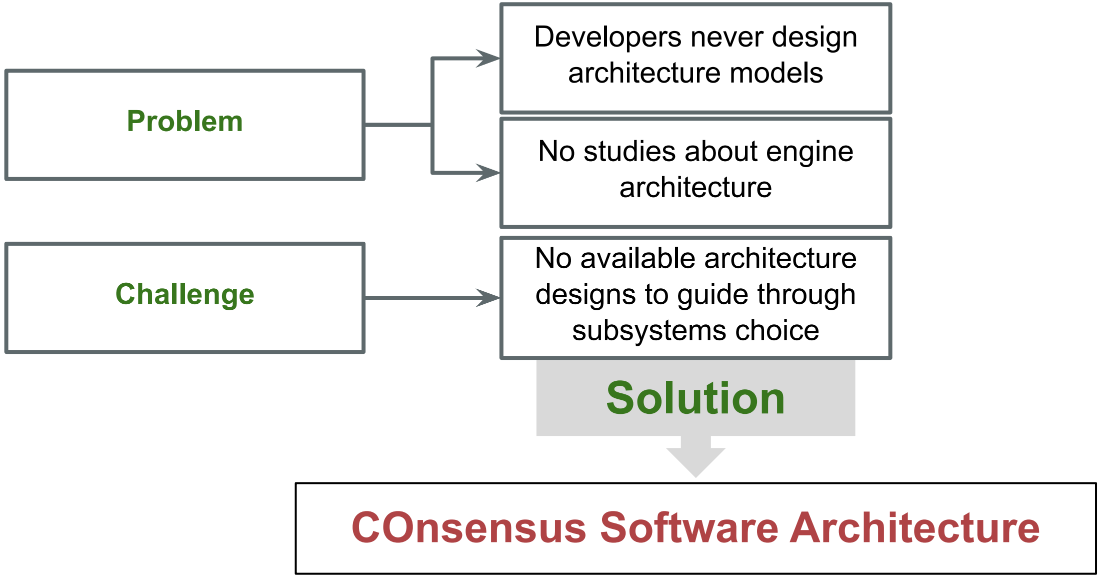
Objective

Evaluate the effectiveness of the consensus algorithm at recommending a **consensus game engine subsystems** to address **game engine-related issues**

Context



Game Engines facilitate the development of games by providing generic, reliable and reusable software subsystems such as a rendering engine, physics engine, audio system, etc.



Problem

Developers never design architecture models

No studies about engine architecture

Challenge

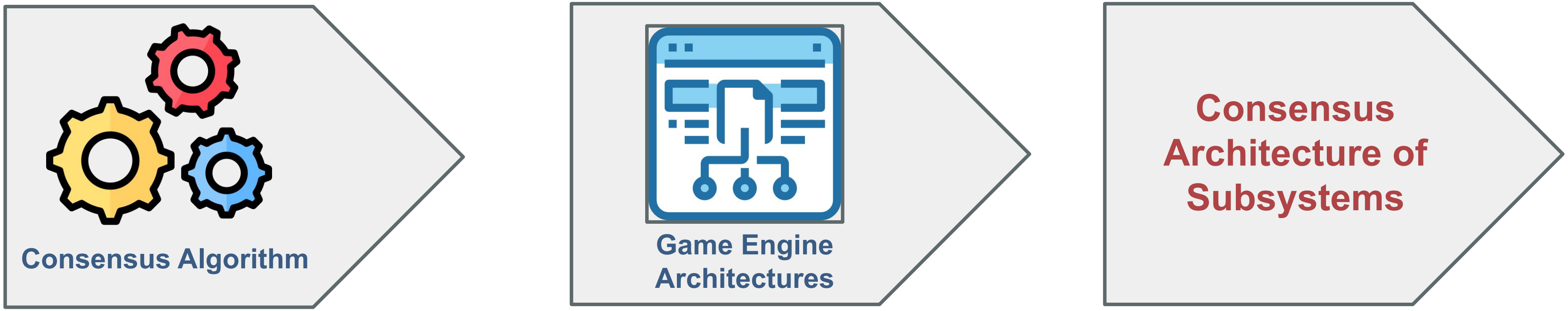
No available architecture designs to guide through subsystems choice

Solution

COnsensus Software Architecture

COnsensus Software Architecture (COSA)

An approach that applies the **consensus algorithm** to a set of game engine architectures to recommend a ranking of **fundamental subsystems**. It helps developers to **decide what subsystems** to include when designing a game engine architecture, and **support reusability and maintenance** by identifying the **most coupled** subsystems.

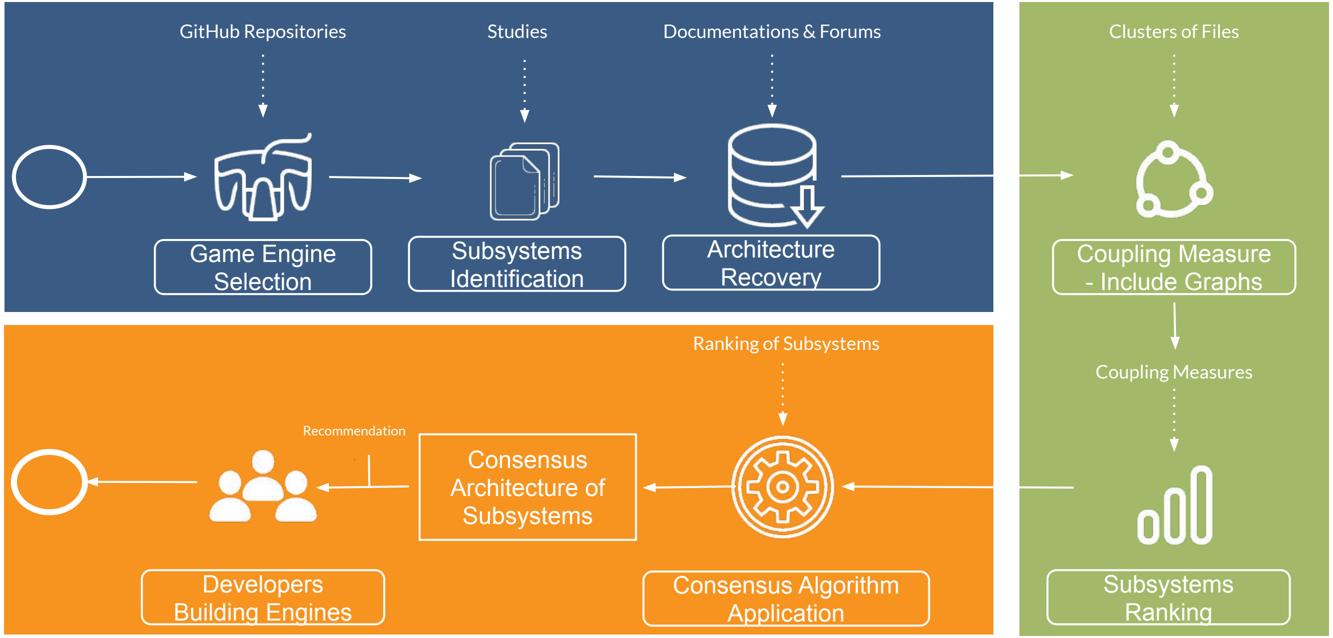


Consensus Algorithm

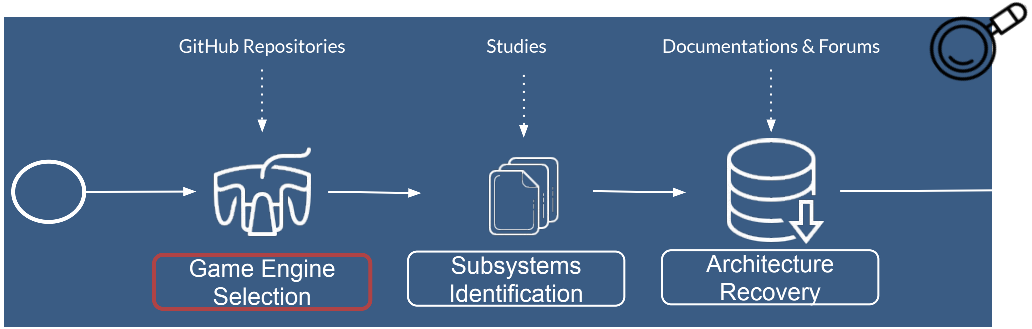
Game Engine Architectures

Consensus Architecture of Subsystems

COnsensus Software Architecture (COSA)

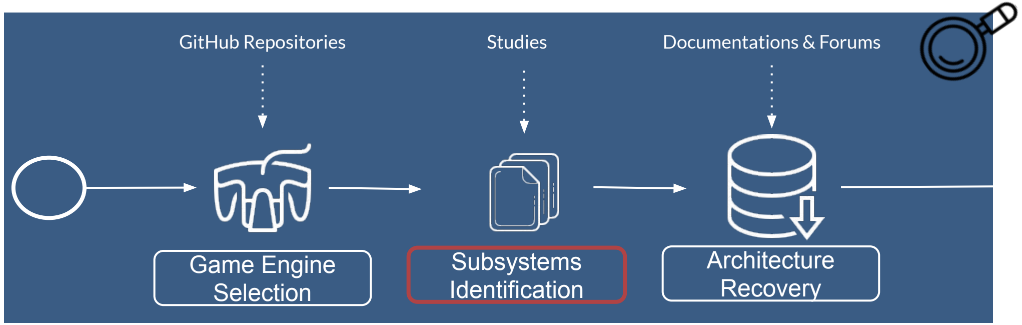


The Approach - Engine Selection

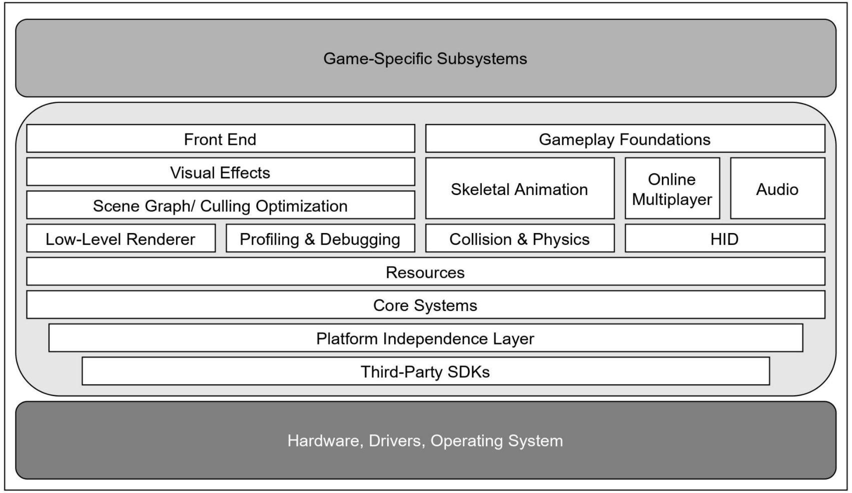


Characteristics	Game Engine
Open Source C++ General-Purpose Highest Forks & Stars Unarchived	UnrealEngine
	godot
	cocos2d-x
	o3de
	Urho3D
	gameplay
	panda3d
	olcPixelGameEngine
	Piccolo
	FlaxEngine

The Approach - Subsystems Identification

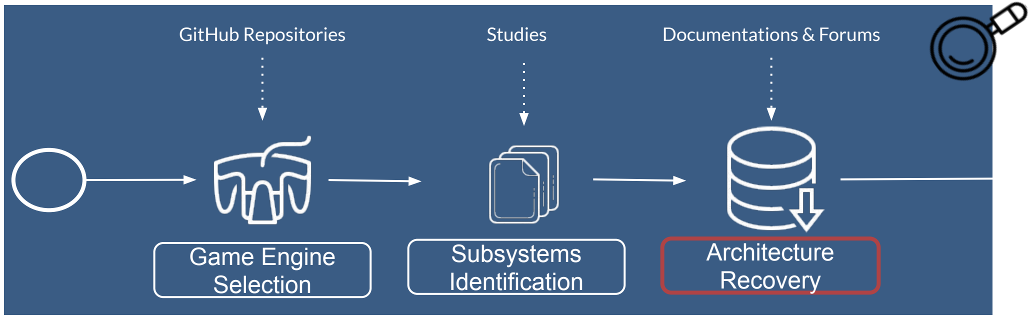


We use the runtime engine architecture defined by Gregory [16] as a guidance for architecture recovery



High-Level Game Engine Architecture [16]

The Approach - Architecture Recovery



Extracting Source-Code Files

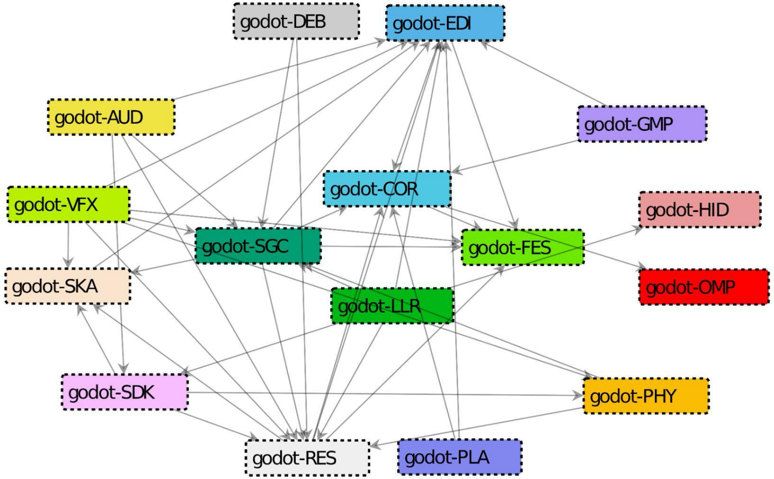
Analysing Files

Clustering Files into Subsystems

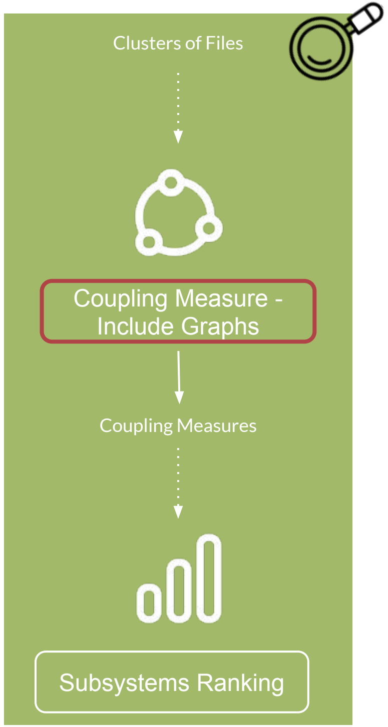
The Approach - Subsystems Coupling Measure

Degree of Coupling?
highly coupled software systems are difficult to maintain, understand, test, or even reuse

Coupling Between Objects (CBO)
A count of the number of classes that are coupled to a particular class



Example of Include Graph for Godot



The Approach - Subsystems Ranking

Tightly Coupled

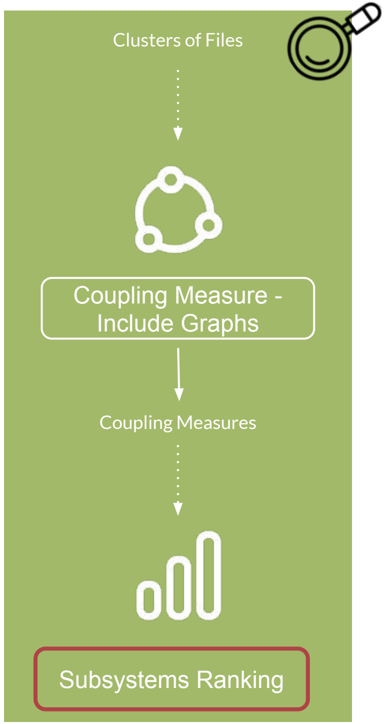
Loosely Coupled

gameplay

gameplay = [Gameplay Foundation], [Resources], [Core Systems], [Skeletal Animation], [Human Interface Devices], [Low-Level Renderer], [Platform Independence Layer], [Collision & Physics], [Front End], [Scene Graph/ Culling], [Audio], [Visual Effects], [Profiling & Debugging]

PICCOLO
Game engine

Piccolo = [Third-Party SDKs], [Core Systems], [Low-Level Renderer], [Resources], [World Editor], [Collision & Physics], [Gameplay Foundation], [Skeletal Animation], [Human Interface Devices], [Platform Independence Layer], [Front End], [Visual Effects]



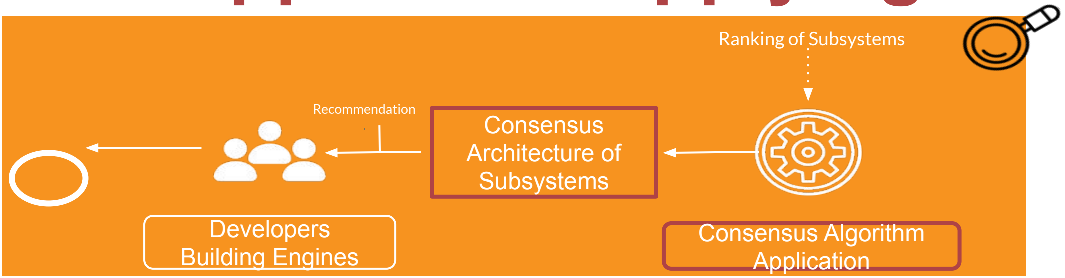
Clusters of Files

Coupling Measure - Include Graphs

Coupling Measures

Subsystems Ranking

The Approach - Applying the Consensus


Ranking of Subsystems
Recommendation
Consensus Architecture of Subsystems
Developers Building Engines
Consensus Algorithm Application

UnrealEngine	[COR],[SDK],[EDI],[FES],[LLR],[SKA],[VFX],[RES],[OMP],[AUD],[DEB],[PLA],[GMP],[PHY],[HID],[SGC]
godot	[SDK],[COR],[EDI],[RES],[FES],[PLA],[PHY],[LLR],[SGC],[VFX],[AUD],[GMP],[OMP],[SKA],[DEB],[HID]
cocos2d-x	[VFX],[COR],[EDI],[LLR],[GMP],[PLA],[FES],[AUD],[PHY],[OMP],[RES],[SKA]
o3de	[EDI],[SKA],[FES],[GMP],[LLR],[COR],[RES],[AUD],[DEB],[OMP],[PHY],[HID],[PLA],[SGC],[SDK]
Urho3D	[SDK],[LLR],[COR],[GMP],[FES],[PHY],[SGC],[AUD],[OMP],[HID],[SKA],[RES]
gameplay	[GMP],[RES],[COR],[SKA],[HID],[LLR],[PLA],[PHY],[FES],[SGC],[AUD],[VFX],[DEB]
panda3d	[COR],[RES],[LLR],[PHY],[EDI],[SGC],[VFX],[OMP],[SKA],[HID],[PLA],[FES],[GMP],[AUD],[DEB]
olcPixelGameEngine	[LLR],[OMP],[AUD],[PHY],[FES]
Piccolo	[SDK],[COR],[LLR],[RES],[EDI],[PHY],[GMP],[SKA],[HID],[PLA],[FES],[VFX]
FlaxEngine	[COR],[LLR],[PLA],[SDK],[GMP],[EDI],[RES],[PHY],[VFX],[DEB],[SKA],[AUD],[HID],[OMP],[FES]

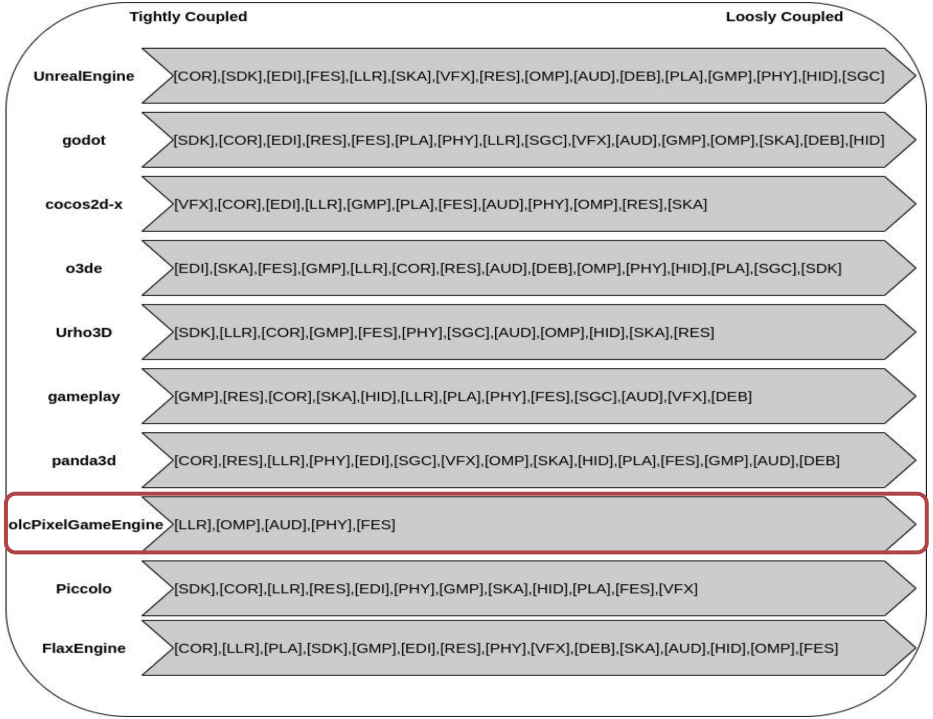
Rankings of Engine Subsystems

Consensus Algorithms - KwikSort

COSA= [[Core Systems], [Low-Level Renderer], [Third-Party SDKs], [World Editor], [Gameplay Foundation], [Resources], [Collision & Physics], [Skeletal Animation], [Platform Independence Layer], [Front End], [Visual Effects], [Audio], [Online Multiplayer], [Profiling & Debugging], [Human Interface Devices], [Scene Graph/ Culling]]

Findings - Commonalities

Game Engines



Tightly Coupled

Loosely Coupled

UnrealEngine	[COR],[SDK],[EDI],[FES],[LLR],[SKA],[VFX],[RES],[OMP],[AUD],[DEB],[PLA],[GMP],[PHY],[HID],[SGC]
godot	[SDK],[COR],[EDI],[RES],[FES],[PLA],[PHY],[LLR],[SGC],[VFX],[AUD],[GMP],[OMP],[SKA],[DEB],[HID]
cocos2d-x	[VFX],[COR],[EDI],[LLR],[GMP],[PLA],[FES],[AUD],[PHY],[OMP],[RES],[SKA]
o3de	[EDI],[SKA],[FES],[GMP],[LLR],[COR],[RES],[AUD],[DEB],[OMP],[PHY],[HID],[PLA],[SGC],[SDK]
Urho3D	[SDK],[LLR],[COR],[GMP],[FES],[PHY],[SGC],[AUD],[OMP],[HID],[SKA],[RES]
gameplay	[GMP],[RES],[COR],[SKA],[HID],[LLR],[PLA],[PHY],[FES],[SGC],[AUD],[VFX],[DEB]
panda3d	[COR],[RES],[LLR],[PHY],[EDI],[SGC],[VFX],[OMP],[SKA],[HID],[PLA],[FES],[GMP],[AUD],[DEB]
olcPixelGameEngine	[LLR],[OMP],[AUD],[PHY],[FES]
Piccolo	[SDK],[COR],[LLR],[RES],[EDI],[PHY],[GMP],[SKA],[HID],[PLA],[FES],[VFX]
FlaxEngine	[COR],[LLR],[PLA],[SDK],[GMP],[EDI],[RES],[PHY],[VFX],[DEB],[SKA],[AUD],[HID],[OMP],[FES]

Game engine architectures are similar in terms of subsystems they include

Findings - Fundamental Subsystems

COSA= [[Core Systems], [Low-Level Renderer], [Third-Party SDKs], [World Editor], [Gameplay Foundation], [Resources], [Collision & Physics], [Skeletal Animation], [Platform Independence Layer], [Front End], [Visual Effects], [Audio], [Online Multiplayer], [Profiling & Debugging], [Human Interface Devices], [Scene Graph/ Culling]]

Consensus Architecture of Subsystems

COSA recommends all subsystems as fundamental, and developers should consider them in their architecture when developing a game engine

Findings - Most Coupled Subsystems

COSA= [[Core Systems], [Low-Level Renderer], [Third-Party SDKs], [World Editor], [Gameplay Foundation], [Resources], [Collision & Physics], [Skeletal Animation], [Platform Independence Layer], [Front End], [Visual Effects], [Audio], [Online Multiplayer], [Profiling & Debugging], [Human Interface Devices], [Scene Graph/ Culling]]

COR and **SDK** are responsible for low-level operations such as memory allocation and file I/O. They serve as support for high-level subsystems such as audio and visual effects

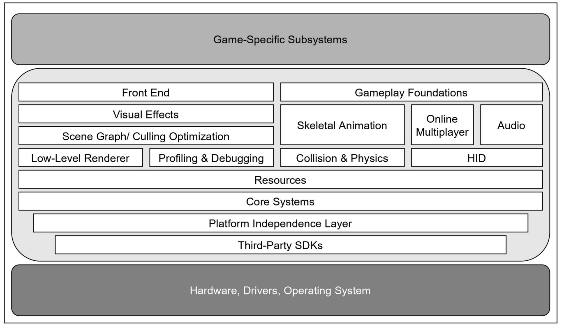
LLR responsible for producing 2D or 3D animated graphics we see on screen in all games

EDI provides a visual interface to many other subsystems

The most coupled subsystems are Core Systems, Low-Level Renderer, 3rd-Party SDKs, and World Editor

Findings - Discovered Subsystems

We discovered subsystems that were never defined in the runtime engine architecture by Gregory [16]



Game-Specific Subsystems

Front End

Gameplay Foundations

Visual Effects

Skeletal Animation

Online Multiplayer

Audio

Scene Graph/ Culling Optimization

Low-Level Renderer

Profiling & Debugging

Collision & Physics

HID

Resources

Core Systems

Platform Independence Layer

Third-Party SDKs

Hardware, Drivers, Operating System

Discovered Subsystem	Game Engine
Code Editor, Multi-User Synchronization, Project Creation, CLI	UnrealEngine, o3de, panda3d
Cache, Source Control	UnrealEngine
Cvars, Graphs (Data Structure), Video Subtitling and Timecoding, Analytics, Media Streaming	FlaxEngine, godot, o3de, panda3d, UnrealEngine
Code Hot Reloading, Visual Scripting, Assembler/Compiler	FlaxEngine, godot, UnrealEngine
Virtual production	UnrealEngine
Screenshot Capture	FlaxEngine
Foliage simulation	UnrealEngine, FlaxEngine
VR, AR, XR	UnrealEngine, godot
Advertisement	UnrealEngine
Cryptography	UnrealEngine, FlaxEngine
Database	UnrealEngine, o3de, Urho3d
Virtualization	UnrealEngine
Cloud Services Integration	o3de

To Conclude

Tightly Coupled

Loosely Coupled

COSA= [[Core Systems], [Low-Level Renderer], [Third-Party SDKs], [World Editor], [Gameplay Foundation], [Resources], [Collision & Physics], [Skeletal Animation], [Platform Independence Layer], [Front End], [Visual Effects], [Audio], [Online Multiplayer], [Profiling & Debugging], [Human Interface Devices], [Scene Graph/ Culling]]

COnsensus Software Architecture (COSA) is successful at recommending the most common subsystems which can help developers decide which subsystems to include when designing a game engine architecture, and support reusability and maintenance by identifying the most coupled subsystems

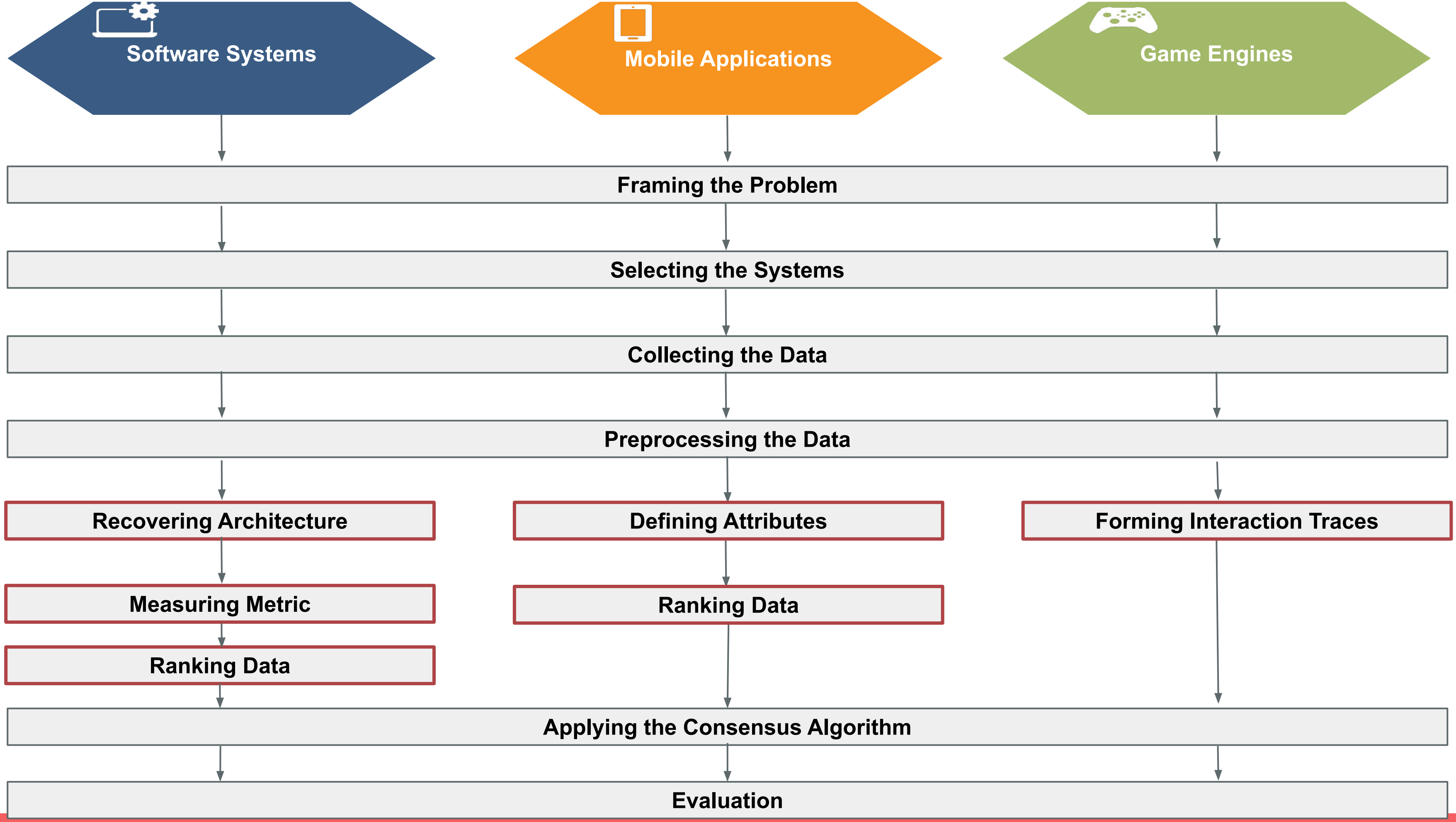
Summary of the Studies

Review Prioritiser (AR) recommends a **prioritised list of user reviews** to help app developers **planning the next releases** of their apps

COnsensus Software Architecture (COSA) recommends a **ranking** of fundamental **subsystems** to help developers **decide which subsystems** to include when designing a game engine architecture

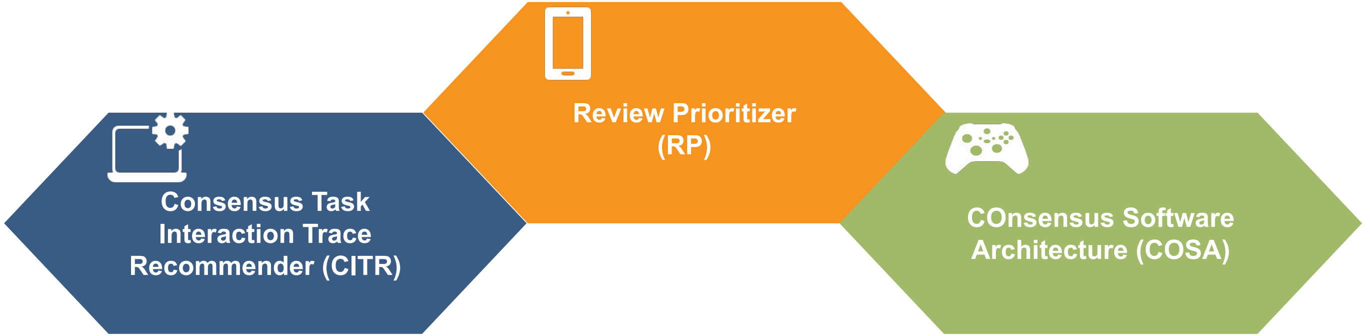
Consensus Task Interaction Trace Recommender (CITR) recommends **file(s)-to-edit** to helps developers **complete development tasks** successfully in less time, and hence **increases their productivity**

Research Methodology



Thesis Conclusion

Thesis Statement. We proposed a recommendation technique for software engineering based on the **consensus algorithm** and that applies to **various data types** and resolves **various software engineering issues in a variety of applications**



Consensus Task Interaction Trace Recommender (CITR)

Review Prioritizer (RP)

COnsensus Software Architecture (COSA)

~~Sufficient Knowledge~~

Content-based techniques require history information or metadata

~~Interactive User Input~~

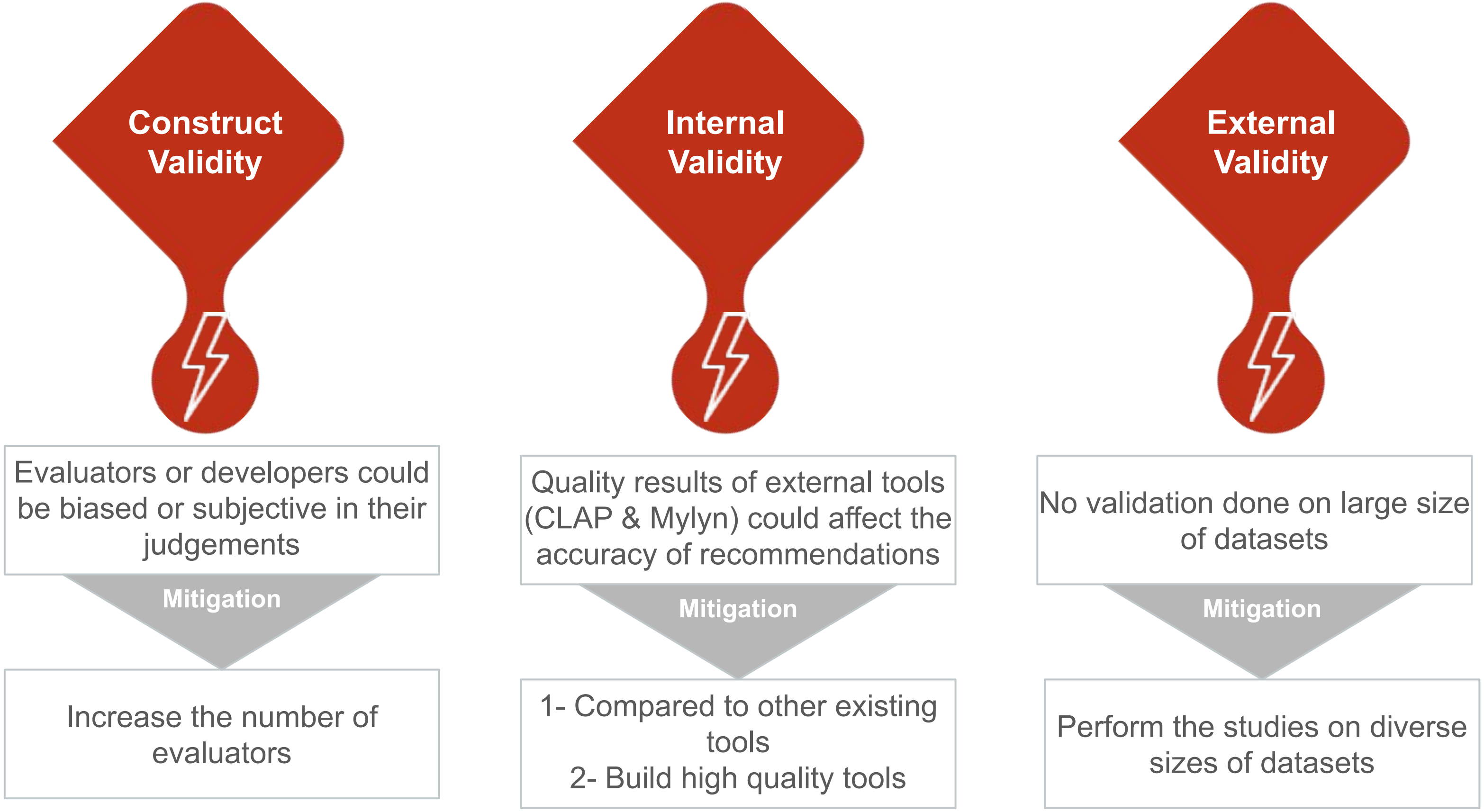
Association rules can only recommend items similar to interactive user input

~~Large Dataset~~

Machine learning based studies required close to 7K reviews to produce recommendations [8,17]

Generalisation

Threats



Construct Validity

Evaluators or developers could be biased or subjective in their judgements

Mitigation

Increase the number of evaluators

Internal Validity

Quality results of external tools (CLAP & Mylyn) could affect the accuracy of recommendations

Mitigation

1- Compared to other existing tools
2- Build high quality tools

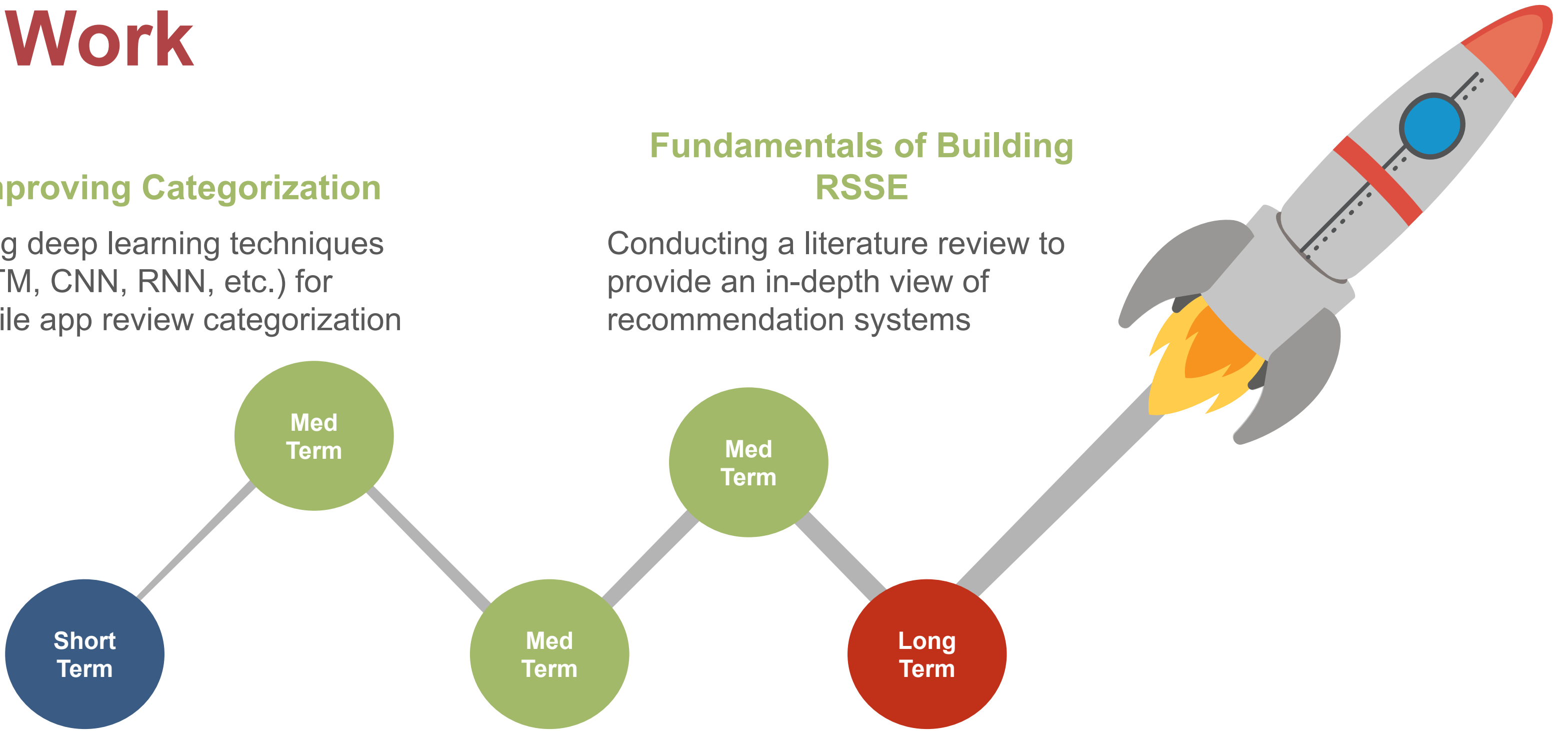
External Validity

No validation done on large size of datasets

Mitigation

Perform the studies on diverse sizes of datasets

Future Work



Research in Other Directions

- Industry applications
- Involving real developers with varying experience
- Larger dataset size

Improving Categorization

Using deep learning techniques (LSTM, CNN, RNN, etc.) for mobile app review categorization

All-In-One Approach

Building a complete tool that can preprocess, categorize, cluster, and prioritize mobile app reviews

Fundamentals of Building RSSE

Conducting a literature review to provide an in-depth view of recommendation systems

Recommendation System

Providing a ready-to-use complete consensus-based recommendation system

The Research Squad

Yann-Gaël Guéhéneuc

Ph.D & eng.
Full Professor
Research Chair
Concordia University

Layan Etaiwi
Polytechnique Montréal

Sylvie Hamel

Ph.D.
Full Professor
Department Chair
University of Montréal

Pascal Sager

M.Sc. Student
Research Assistant
Zürich University of Applied Science (ZHAW)

Gabriel Ullmann

M.Sc. Student
Concordia University

Thank you!

References

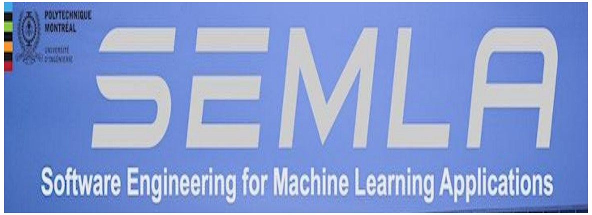
- [1] A. Ali and M. Meilă, “Experiments with kemeny ranking: What works when?” Mathematical Social Sciences, vol. 64, no. 1, pp. 28–40, 2012.
- [2] B. Brancotte et al., “Rank aggregation with ties: Experiments and analysis,” Proc.VLDB Endow., vol. 8, no. 11, p. 1202–1213, jul 2015.
- [3] S. Scalabrino et al., “Listening to the crowd for the release planning of mobile apps,” IEEE Transactions on Software Engineering, vol. 45, no. 1, pp. 68–86, 2017.
- [4] S. Lee et al., “The impact of view histories on edit recommendations,” IEEE Transactions on Software Engineering, vol. 41, no. 3, pp. 314–330, 2015.
- [5] L. C. Briand, J. W. Daly, and J. K. Wust, “A unified framework for coupling measurement in object-oriented systems,” IEEE Transactions on software Engineering, vol. 25, no. 1, pp. 91–121, 1999.
- [6] S. R. Chidamber and C. F. Kemerer, “Towards a metrics suite for object oriented design,” in Conference proceedings on Object-oriented programming systems, languages, and applications, 1991, pp. 197–211.
- [7] J. Gregory, Game engine architecture, 3rd ed. Press, 2018.
- [8] E. Guzman and M. Walid, "How do users like this feature? a fine grained sentiment analysis of app reviews." In *2014 IEEE 22nd international requirements engineering conference (RE)*, pp. 153-162. Ieee, 2014.
- [9]https://graphite-note.com/how-much-data-is-needed-for-machine-learning#:~:text=Generally%20speaking%2C%20the%20rule%20of,100%20rows%20for%20optimal%20results.

References

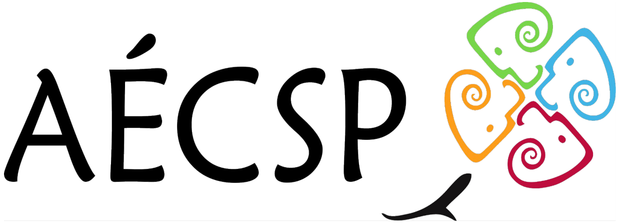
- [10] B. Ashok et al., “Debugadvisor: A recommender system for debugging,” in Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 2009, pp. 373–382.
- [11] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” Egyptian informatics journal, vol. 16, no. 3, pp. 261–273, 2015.
- [12] N. Chen et al., “Ar-miner: mining informative reviews for developers from mobile app marketplace,” in Proceedings of the 36th international conference on software engineering, 2014, pp. 767–778.
- [13] A. Ciurumelea et al., “Analyzing reviews and code of mobile apps for better release planning,” in 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2017, pp. 91–102.
- [14] S. Panichella et al., “Ardoc: App reviews development oriented classifier,” in Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering, 2016, pp. 1023–1027.
- [15] A. Al-Subaihin et al., “App store mining and analysis,” in Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile, 2015, pp. 1–2.
- [16] J. Gregory, Game engine architecture, 3rd ed. Press, 2018.
- [17] S. Panichella et al., “How can i improve my app? classifying user reviews for software maintenance and evolution,” in 2015 IEEE international conference on software maintenance and evolution (ICSME). IEEE, 2015, pp. 281–290.

Academic & Community Involvement

Student Volunteer - 2018



AÉCSP- Interface Tournament Coordinator - 2020



Student Volunteer - 2019

Scholarship

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Government Excellence Scholarship 2020/21