SyDRA: An Exploratory Approach to Game Engine Architecture Recovery

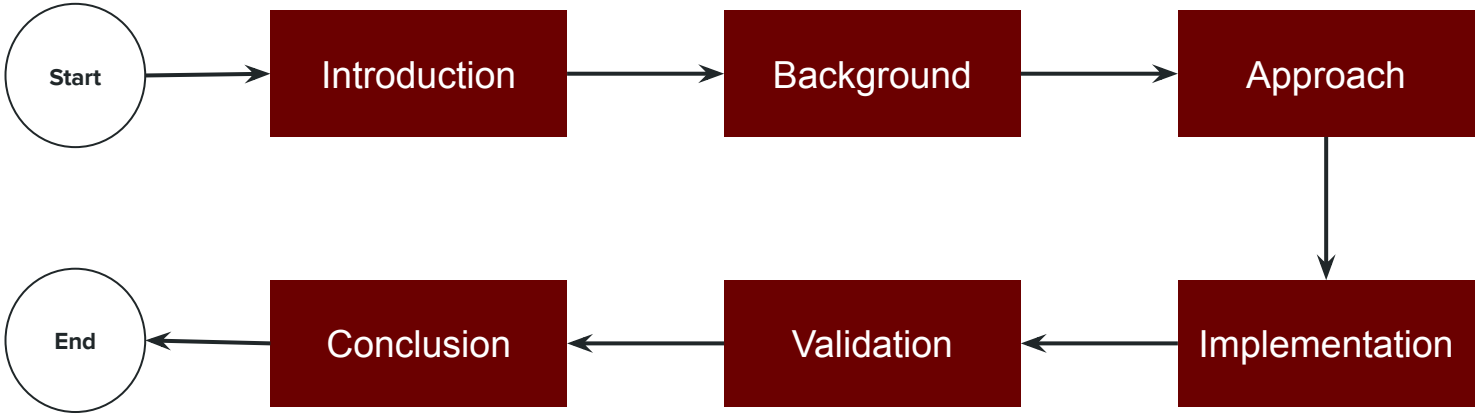
Gabriel C. Ullmann
Concordia University

Supervised By:

Dr. Yann-Gaël Guéhéneuc
Concordia University

Dr. Fabio Petrillo
École de Technologie Supérieure

Summary

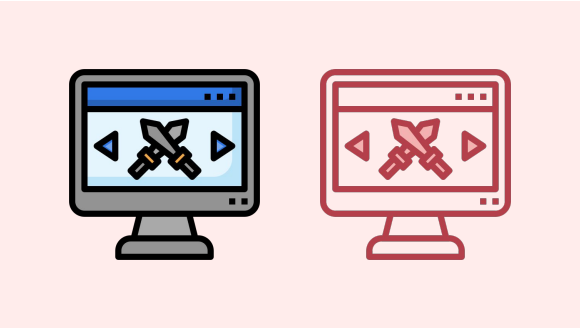

Start
Introduction
Background
Approach
Implementation
Validation
Conclusion
End

1. Introduction

What is a Game Engine?

Game-making

A tool to facilitate video game development

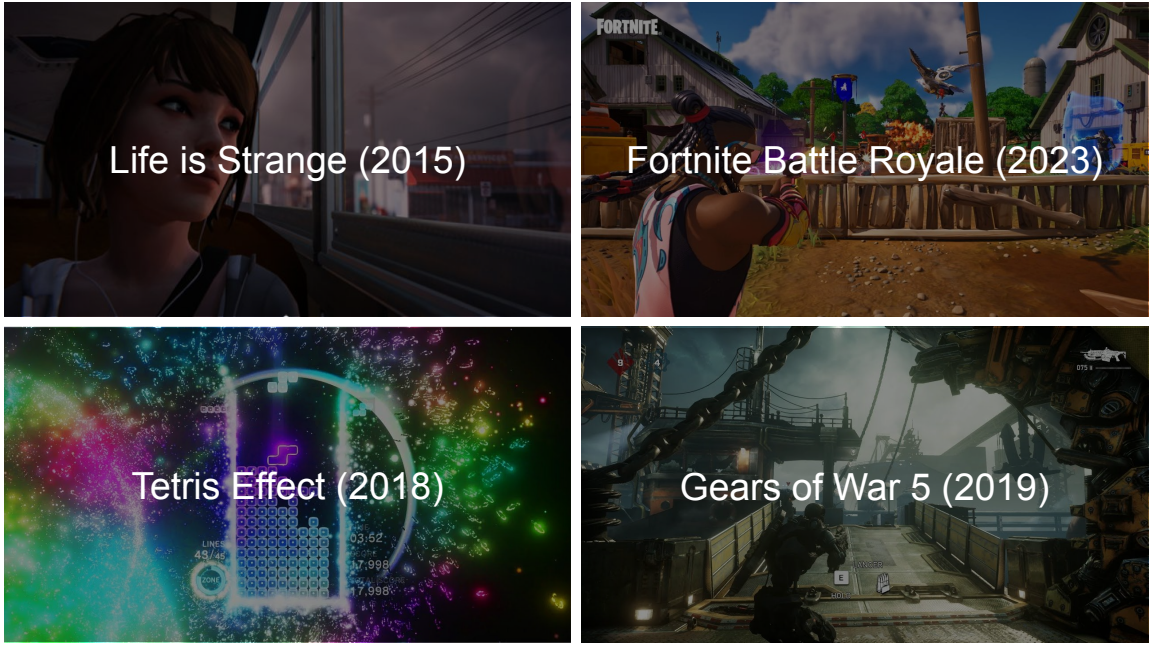


Flexibility

Reusable and extendable subsystems

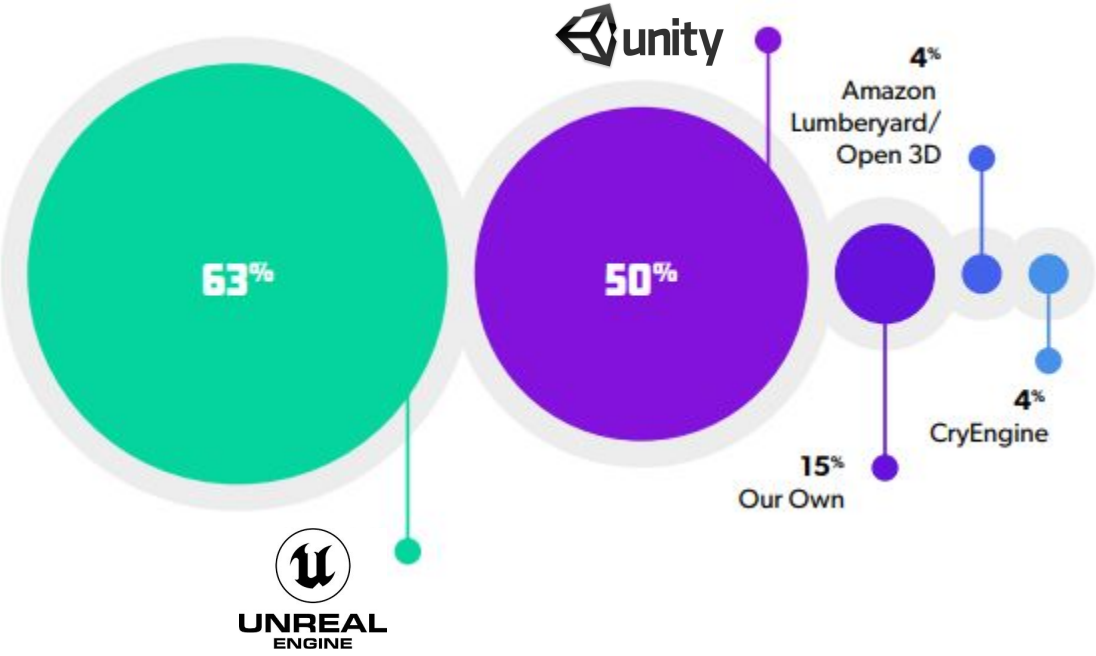
GIF from Unreal Engine 4.27 release notes.

What Can They Do?



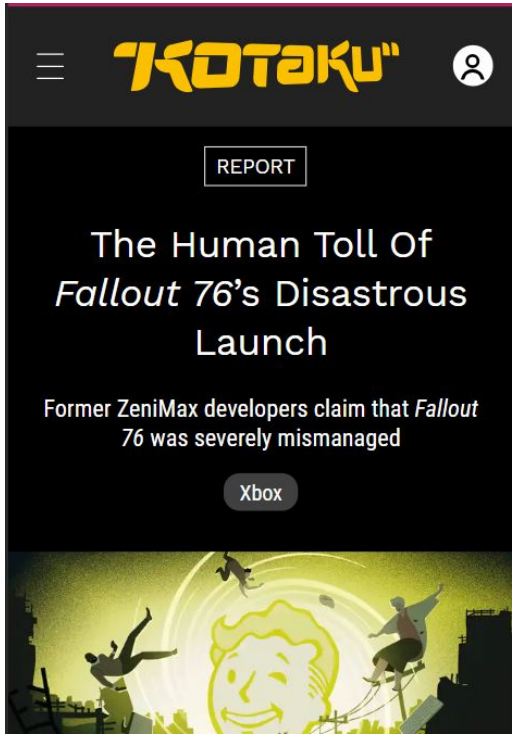
Games made with different versions of Unreal Engine.

What Game Engines Are Currently Used?



Adapted from 2022 Game Development Trends & Forecast by Perforce.

What Are The Problems?



Problem 1: Changes Are Time-Consuming


GAMERANT Log in

Home > Gaming News

Former Dragon Age Producer Reflects On BioWare's Problems

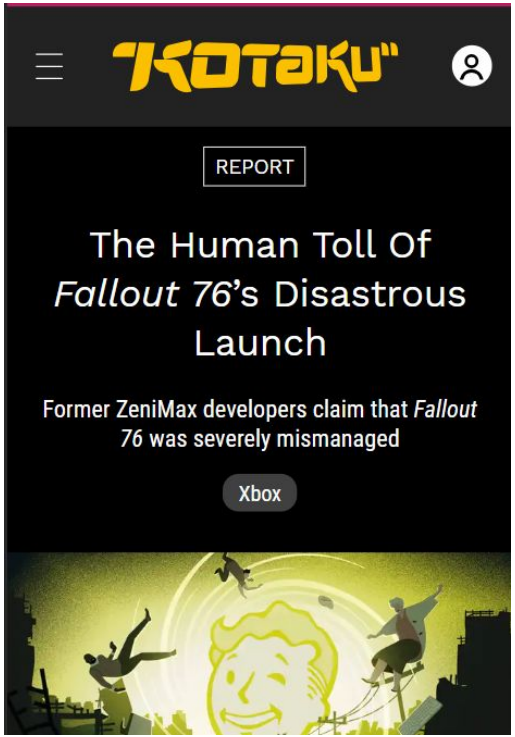
Former Executive Producer Mark Darrah releases a video that offers insight into what went wrong at BioWare during and after Dragon Age: Inquisition.

BY CHRISTIAN MILLER PUBLISHED NOV 10, 2022

"[Dragon Age's executive producer] went on to say that developing the tools for Frostbite took up about a third of the project's development time"

By Christian Miller for GameRant, 2022.

Problem 2: Changes Cause Bugs



"According to one source who worked on the game, *Fallout 76*'s tools were so poorly optimized that simply updating the build could break it, which could add even more production pressure on the developers."

By Sisi Jiang for Kotaku, 2023.

Problem 3: Game Engines Are Complex Systems


ACM DL DIGITAL LIBRARY

ISEC

SHORT-PAPER

What's Inside Unreal Engine? - A Curious Gaze!

Authors: Vartika Agrahari, Sridhar Chimalakonda Authors Info & Claims

ISEC 2021: 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) • February 2021 • Article No.: 21 • Pages 1–5 • https://doi.org/10.1145/3452383.3452404

"Unreal Engine is a complex software project consisting of 203 releases with around 215,000 commits. (...) "

V. Agrahari and S. Chimalakonda. "What's Inside Unreal Engine? - A Curious Gaze!". In 14th Innovations in Software Engineering Conference, 2021.

Problem 3: Game Engines Are Complex Systems


ACM DIGITAL LIBRARY

ISEC

SHORT-PAPER

What's Inside Unreal Engine? - A Curious Gaze!

Authors: Vartika Agrahari, Sridhar Chimalakonda Authors Info & Claims

ISEC 2021: 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) • February 2021 • Article No.: 21 • Pages 1–5 • https://doi.org/10.1145/3452383.3452404

"A prerequisite for integration and extension is the comprehension of the software. To understand the architecture, we should identify the architectural patterns involved and how they are coupled."

V. Agrahari and S. Chimalakonda. "What's Inside Unreal Engine? - A Curious Gaze!". In 14th Innovations in Software Engineering Conference, 2021.

Thesis Statement

SyDRA

Subs**y**stem **D**ependency **R**ecovery **A**pproach

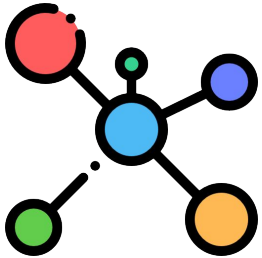
Proposal → **Implementation** → **Evaluation**

Present a 6-step architecture recovery approach

Apply to 10 open-source game engines

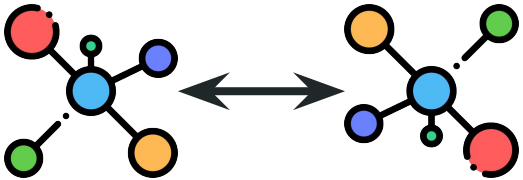
Via literature comparison and user study

Research Questions



RQ1

Which subsystems are more often coupled with one another?



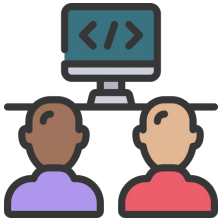
RQ2

Do game engines share subsystem coupling patterns?

Research Evaluation

Qualitative

To what extent do the game engines that we analysed with **SyDRA** match the software architecture literature?



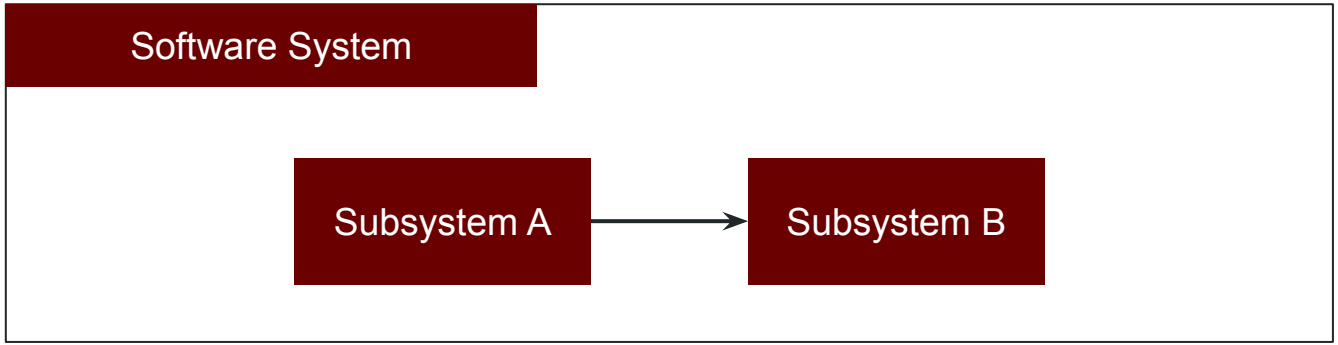
User Study

Does **SyDRA** help developers understand and maintain the architecture of game engines?

2. Background

Software Architecture

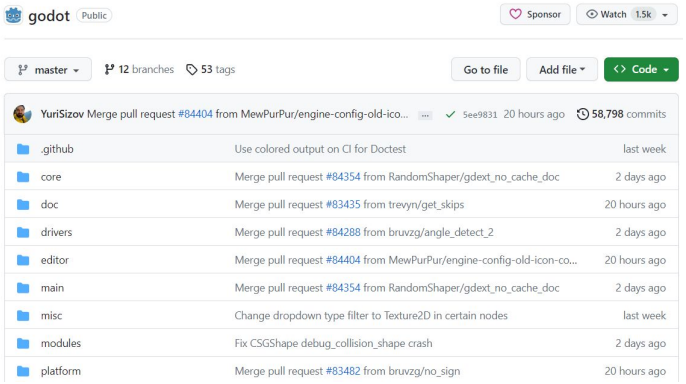
"Software architecture is the highest-level **breakdown** of a **system** into its **parts**."

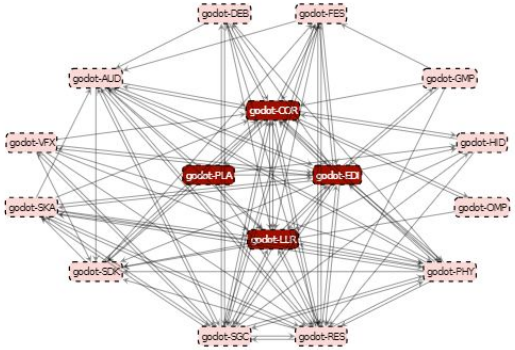


M. Fowler. "Patterns of enterprise application architecture", 2003.

Software Architecture Recovery

"The **extraction** of architectural descriptions **from a system implementation**"





I. Bowman et al. "Linux as a case study: its extracted software architecture". In Proceedings of the 21st International Conference on Software Engineering (ICSE), 1999.

Software Architecture Recovery - Inputs

Source Code

Textual Information

Dynamic Information

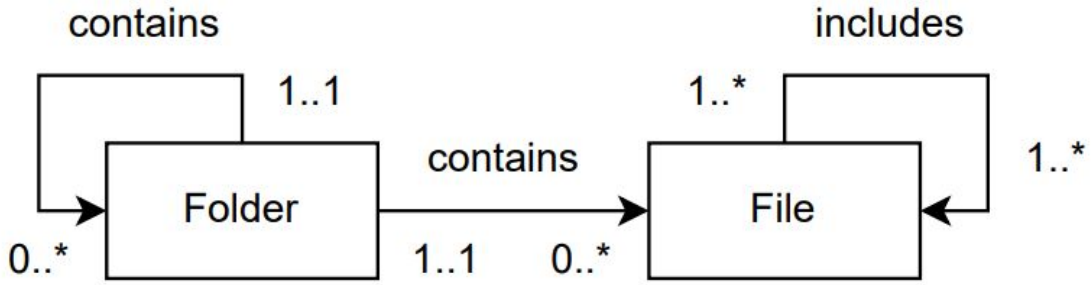
Physical Organization

Human Expertise

Historical Data

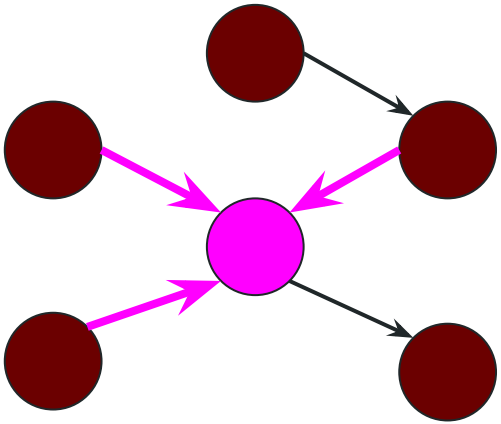
Software Architecture Recovery - Metamodels

"A metamodel describes the **possible structures** which can be expressed in a programming language"



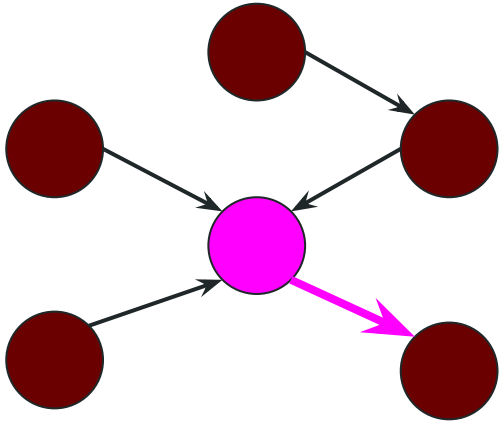
H. Vangheluwe et al. "Meta-Models are models too". In Proceedings of the IEEE Winter Simulation Conference, 2002.

Software Architecture Recovery - Graph Analysis



In-degree

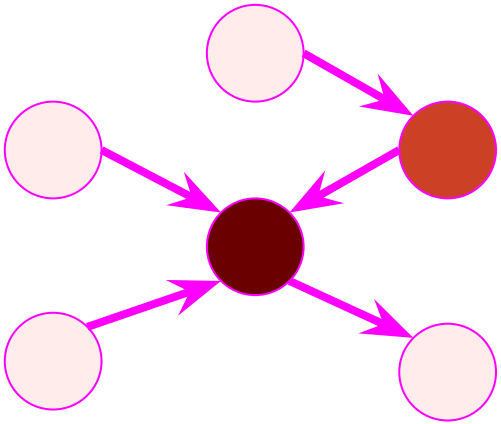
Incoming edge count



Out-degree

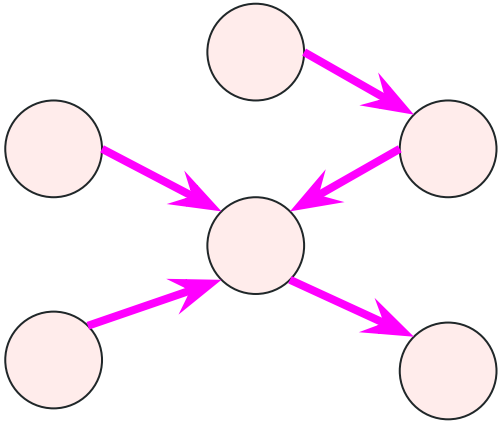
Outgoing edge count

Software Architecture Recovery - Graph Analysis



Betweenness Centrality

How frequently a node is positioned between other nodes



Density

Proportion of possible incoming/outgoing edges

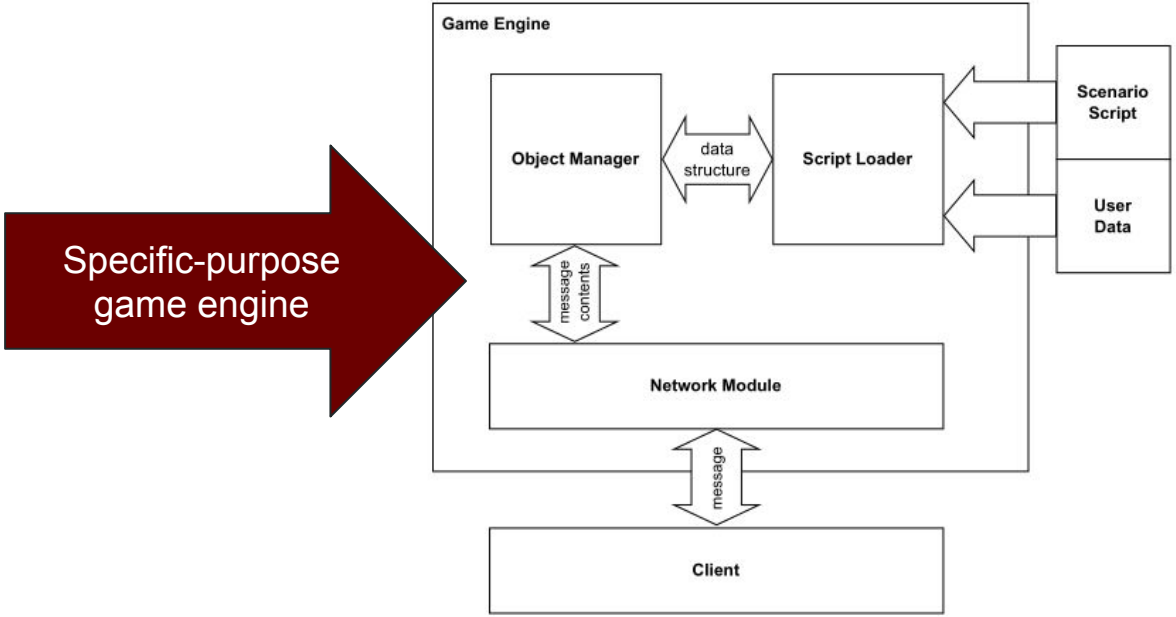
Betweenness Centrality

This metric is measured with the number of shortest paths (between any couple of nodes in the graphs) that passes through the target node *u* (denoted $\sigma_{v,w}(u)$). This score is moderated by the total number of shortest paths existing between any couple of nodes of the graph (denoted $\sigma_{v,w}$).

$$B(u) = \sum_{u \neq v \neq w} \frac{\sigma_{v,w}(u)}{\sigma_{v,w}}$$

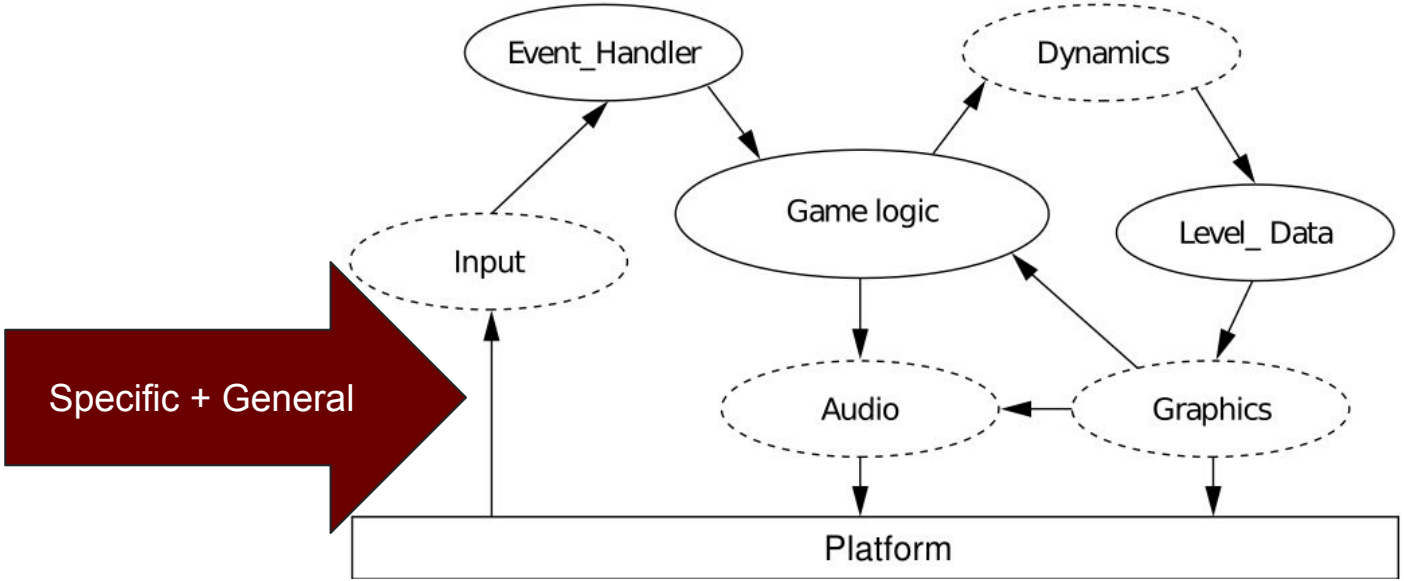
C. Perez et al. "Graph Creation and Analysis for Linking Actors: Application to Social Data". In Automating Open Source Intelligence, 2016

Game Engine Architectures



J. Park and C. Park. "Development of a multiuser and multimedia game engine based on TCP/IP". In Proceedings of IEEE PACRIM, 1997.

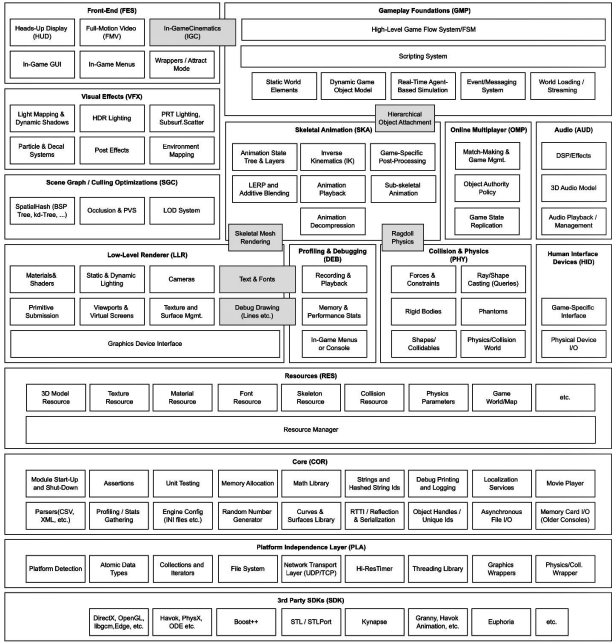
Game Engine Architectures



L. Bishop et al. "Designing a PC game engine". In IEEE Computer Graphics and Applications, 1998.

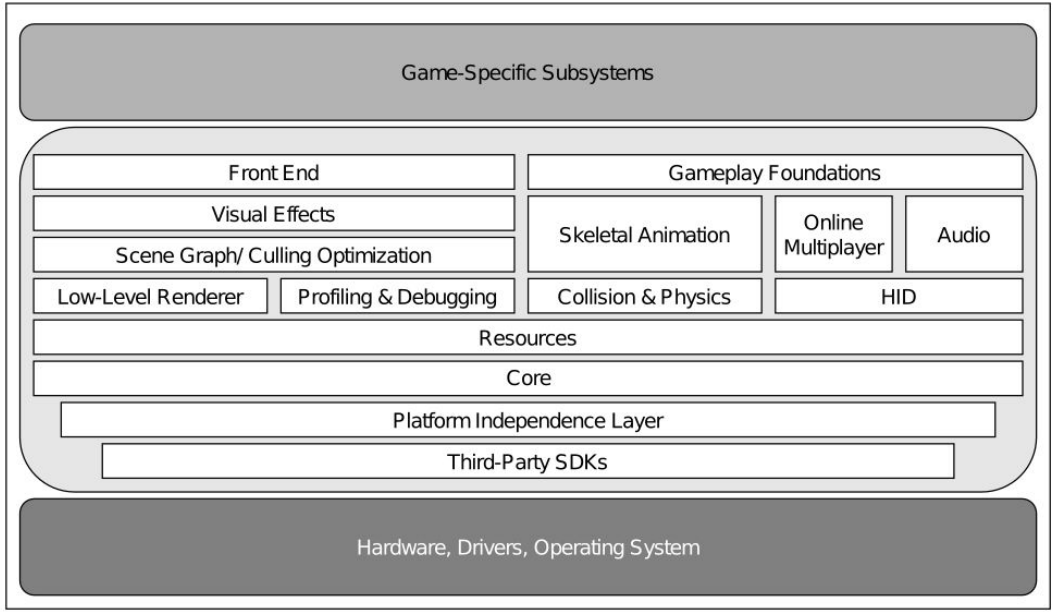
Game Engine Architectures

Expanded and layered



J. Gregory. "Game Engine Architecture", 1st edition, 2009.

Reference Game Engine Architecture: Gregory

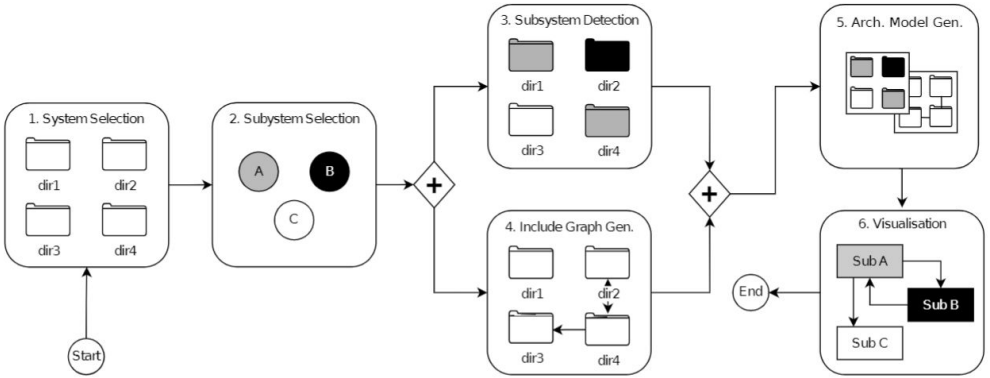


Adapted from J. Gregory. "Game Engine Architecture", 2018.

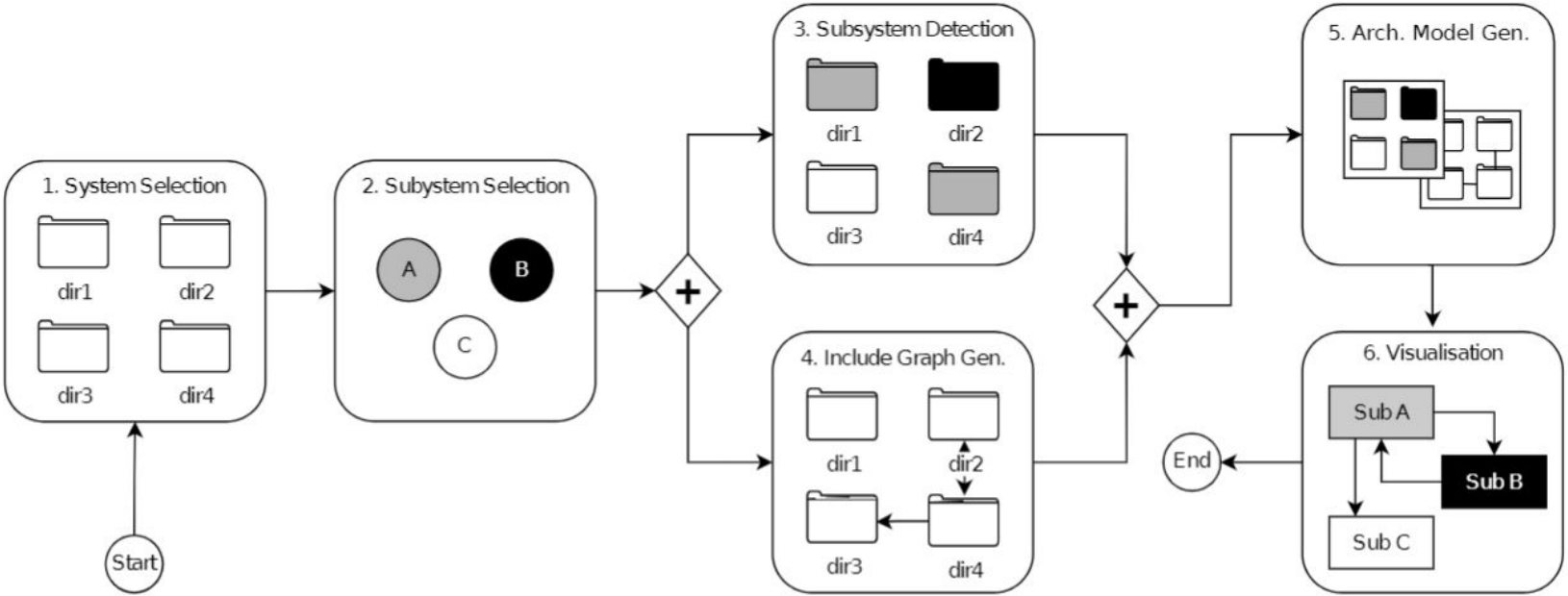
3. Approach

Steps of the SyDRA

1. System selection
2. Subsystem selection
3. Subsystem detection
4. Include graph generation
5. Moose model generation
6. Architectural model visualisation



Steps of the SyDRA

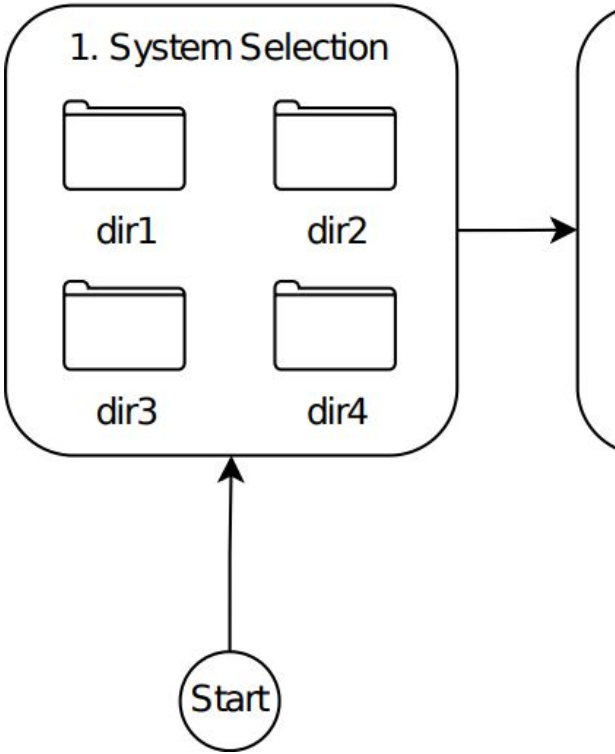


4. Implementation

System Selection

10 game engines from GitHub which are:

- Publicly available
- General-purpose
- Popular (number of stars + forks)
- Written in C++



System Selection

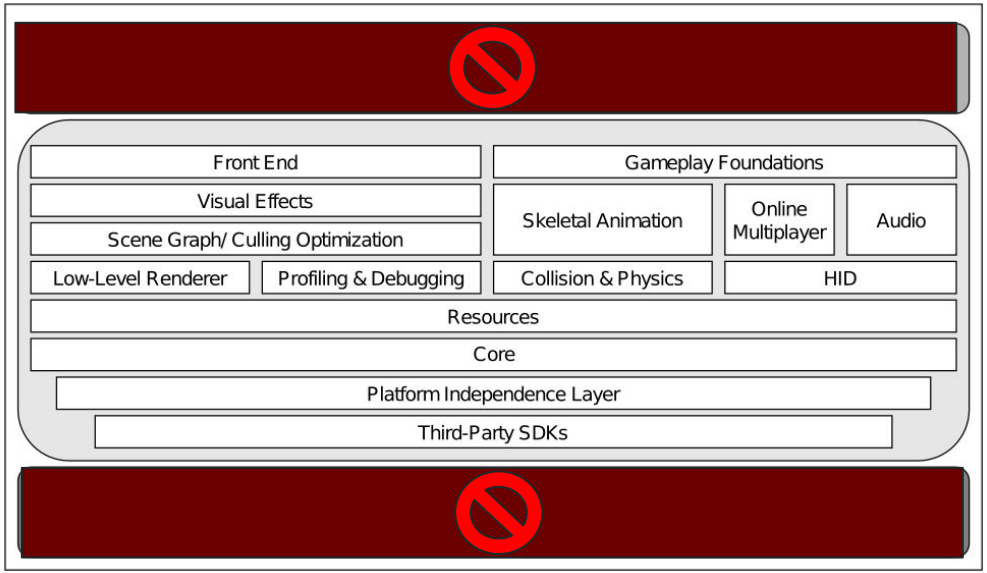
olcPixelGameEngine

Logos for the 10 selected game engines as provided by their official website or GitHub page.

Subsystem Selection

16 subsystems from Gregory:

- AKA: modules, components
- Generalisation in mind
- Functionality descriptions

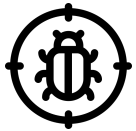


Adapted from J. Gregory. "Game Engine Architecture", 2018.

Subsystem Selection

Audio (AUD)

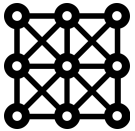
Core (COR)



Debug (DEB)

Editor (EDI)

Front End (FES)



Gameplay (GMP)

Inputs (HID)

Renderer (LLR)

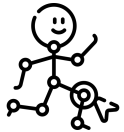
Multipl. (OMP)

Physics (PHY)

Platforms (PLA)

Resources (RES)

3rd Party (SDK)



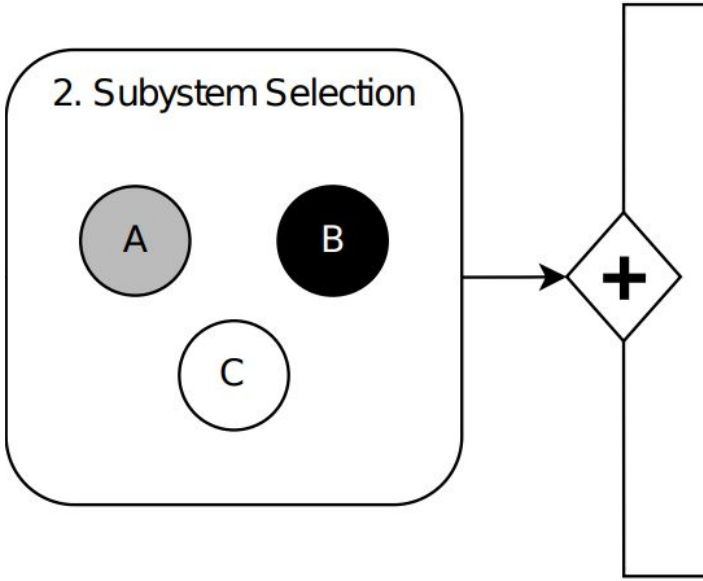
Animation (SKA)

Scene (SGC)

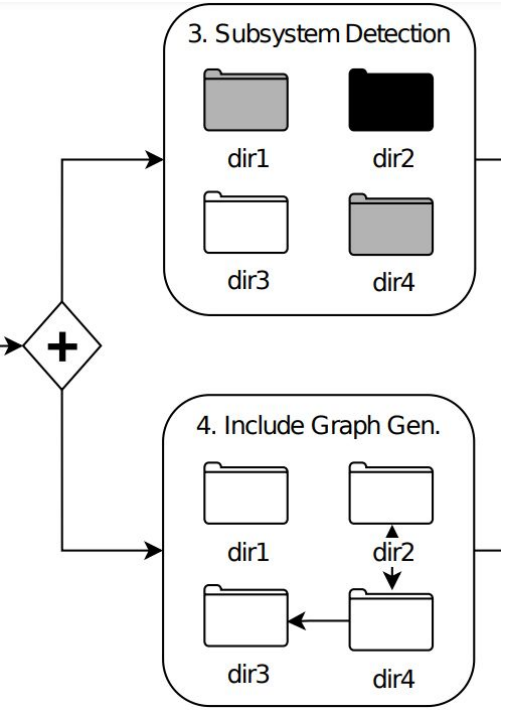
Visuals (VFX)

Subsystem Selection

Abbrev.	Name
AUD	Audio
COR	Core Systems
DEB	Profiling and Debugging
EDI	World Editor
FES	Front End
GMP	Gameplay Foundations
HID	Human Interface Devices
LLR	Low-Level Renderer
OMP	Online Multiplayer
PHY	Collision and Physics
PLA	Platform Independence Layer
RES	Resources (Game Assets)
SDK	Third-party SDKs
SGC	Scene graph/culling optimizations
SKA	Skeletal Animation
VFX	Visual Effects



Subsystem Detection and Include Graph Generation



Clustering files/folders by:

- Naming
- Children naming
- Documentation
- Source Code

Generate an include graph:

- We used cinclude2dot
- We resolved some absolute include paths manually

Include path resolution issues: example

engine.cpp

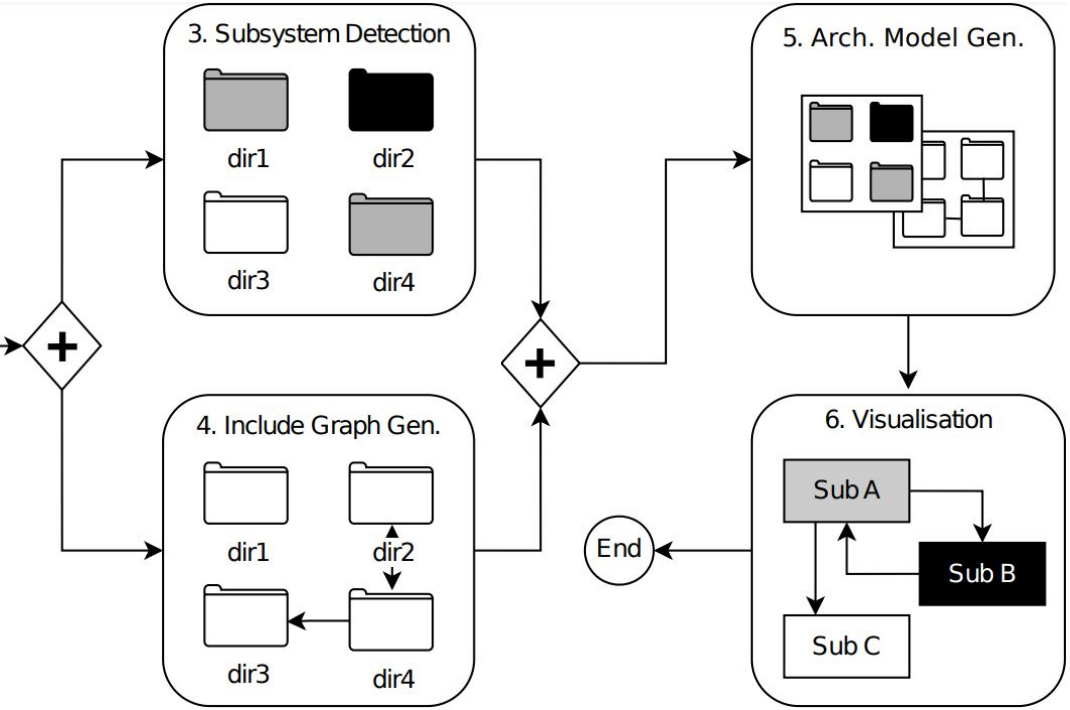
```
#include bla.h

-------------------------------
-------------------------------
-------------------------------
-------------------------------
-------------------------------
-------------------------------
-------------------------------
---------------------------
```

Where is bla.h?

- Same folder
- Other folder(s) defined in the build config files (e.g. CMakeLists.txt)
  - These folders could be in another repository (e.g. third-party libraries)
- Some files are generated during build (e.g. based on templates)
- P.S: not every game engine uses CMake

Architectural Model Generation and Visualisation



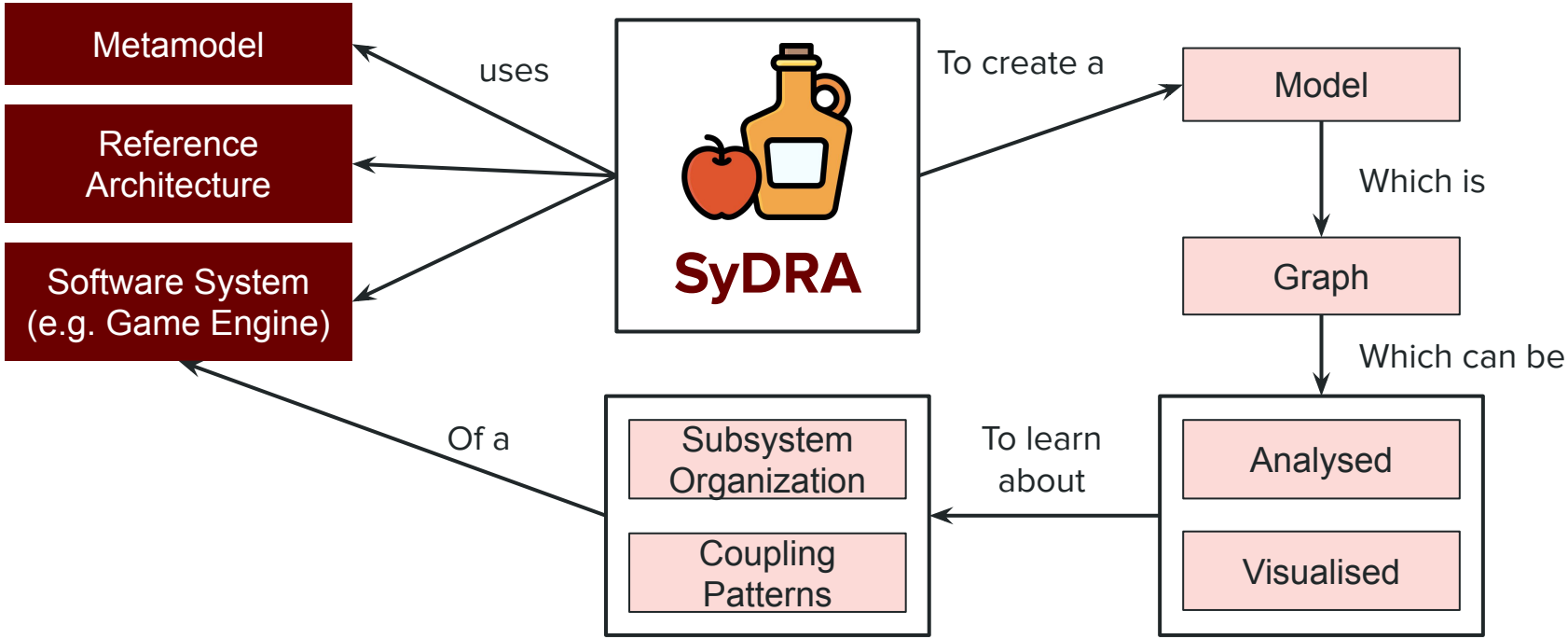
Generated using:

Moose, a platform for software analysis; FamixCpp.

Visualised as:

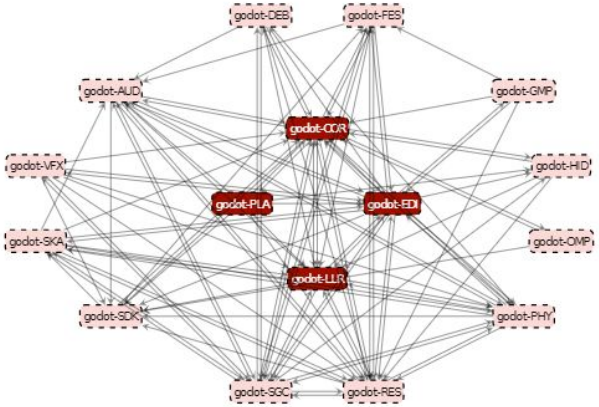
Architectural Map (graph) + derived visualisations

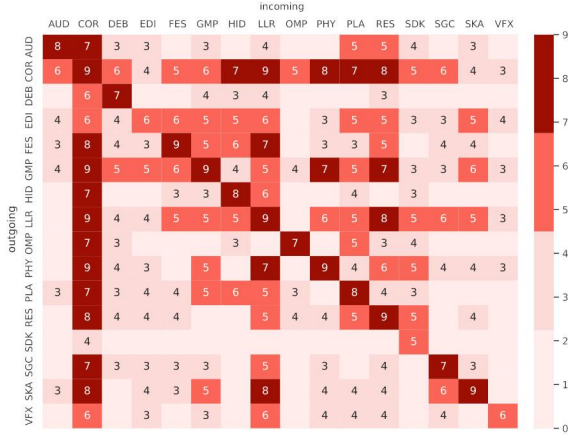
SyDRA: Background x Approach



5. Results

Results





RQ1

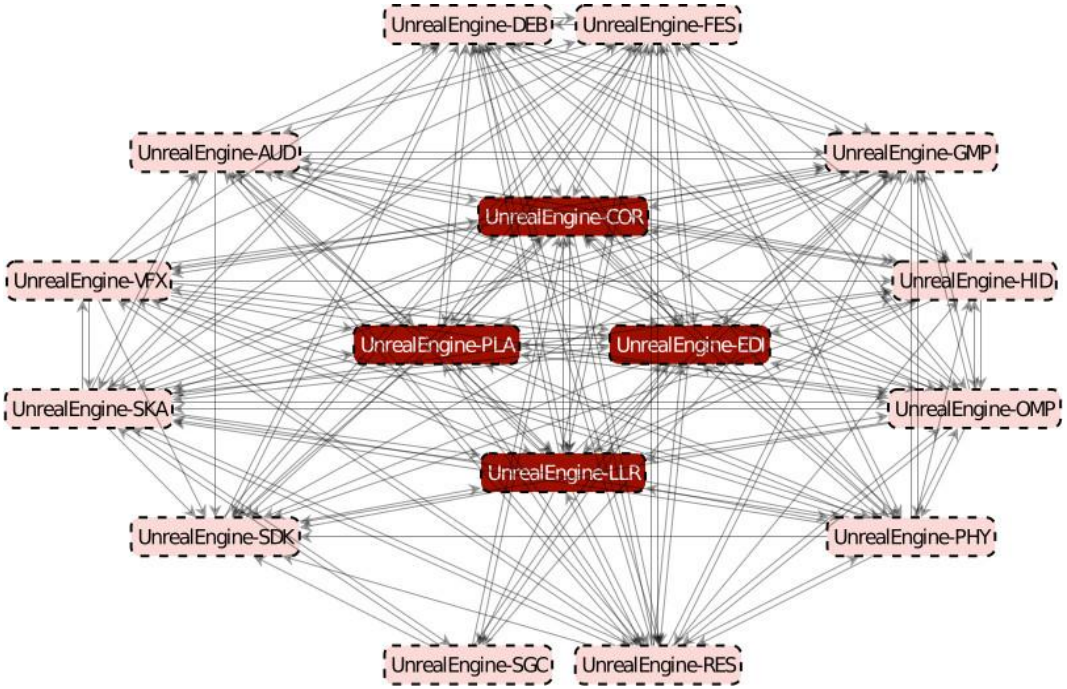
Which subsystems more often couple with one another?

RQ2

Do game engines share subsystem coupling patterns?

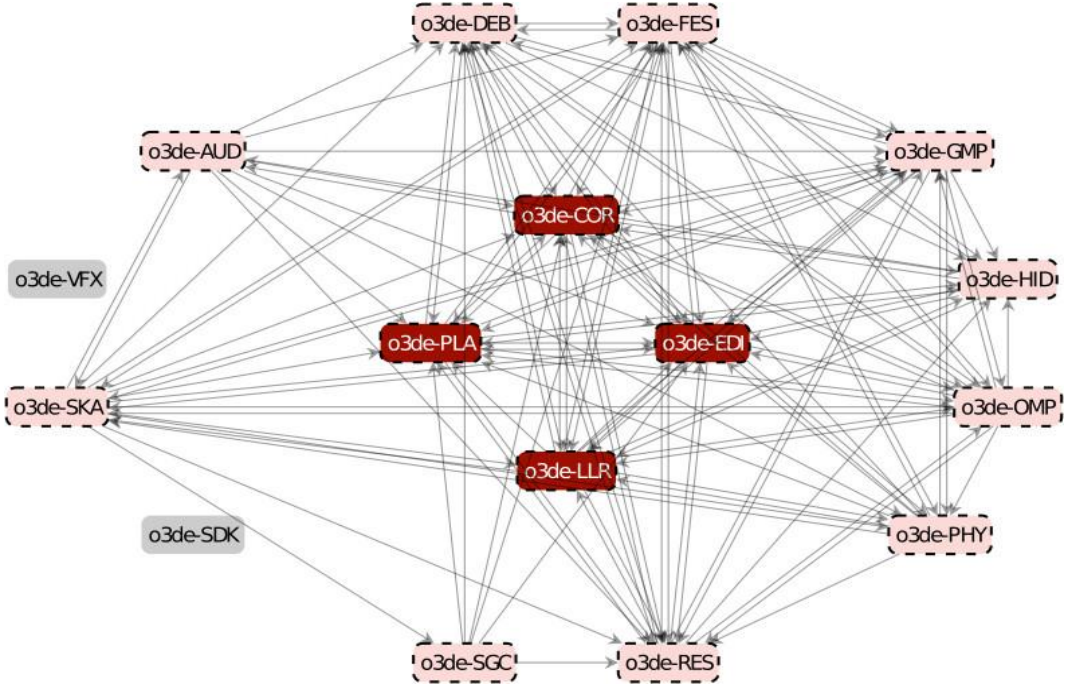
Graph analysis: in-degree, centrality and density

RQ1: Which Subsystems Are More Often Coupled With One Another?



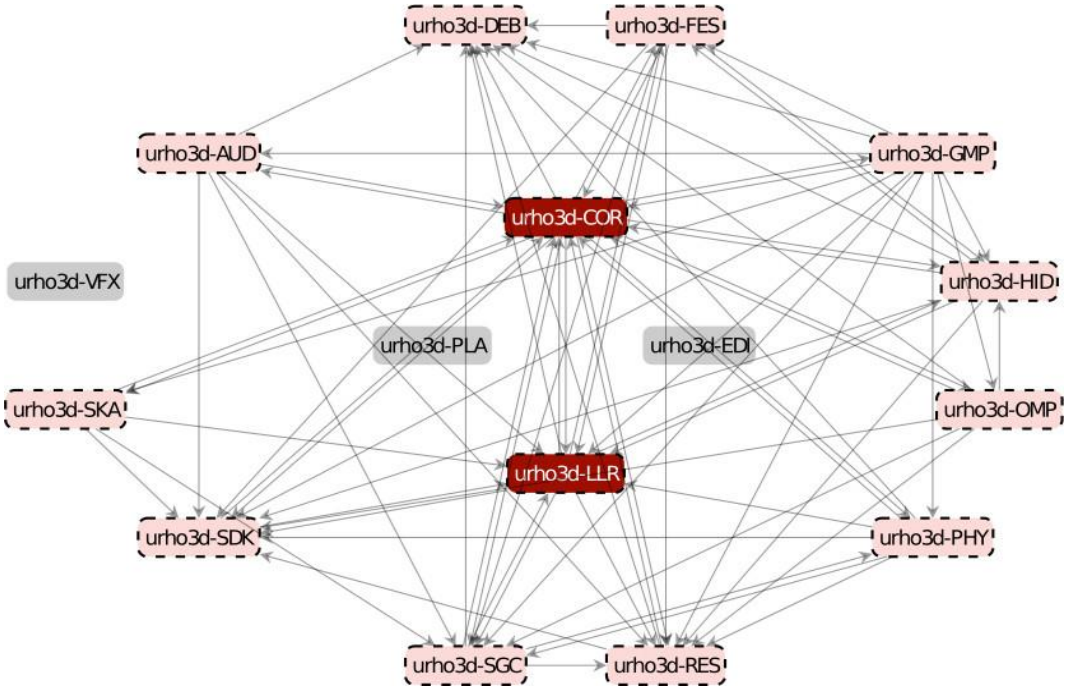
Unreal Engine

RQ1: Which Subsystems Are More Often Coupled With One Another?



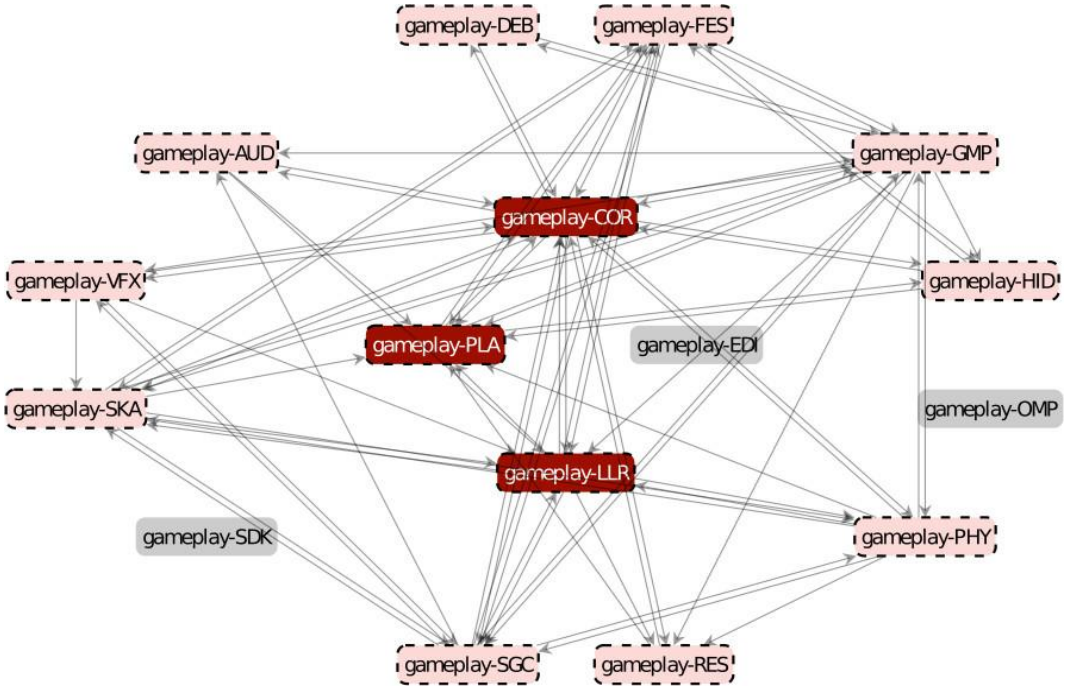
O3DE

RQ1: Which Subsystems Are More Often Coupled With One Another?



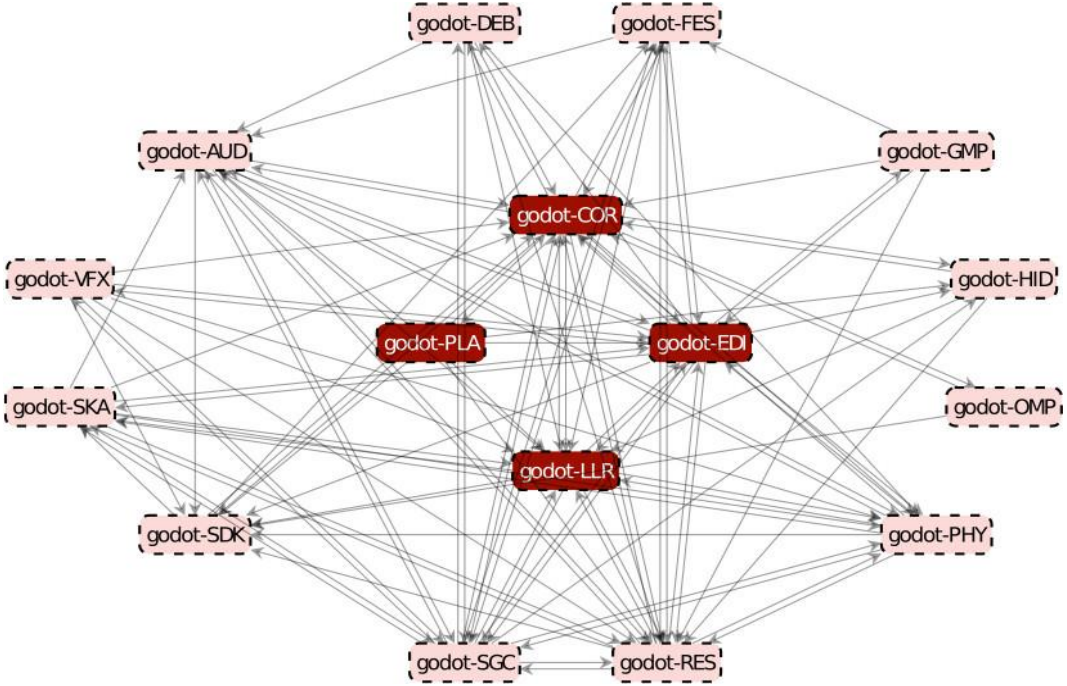
Urho3D

RQ1: Which Subsystems Are More Often Coupled With One Another?



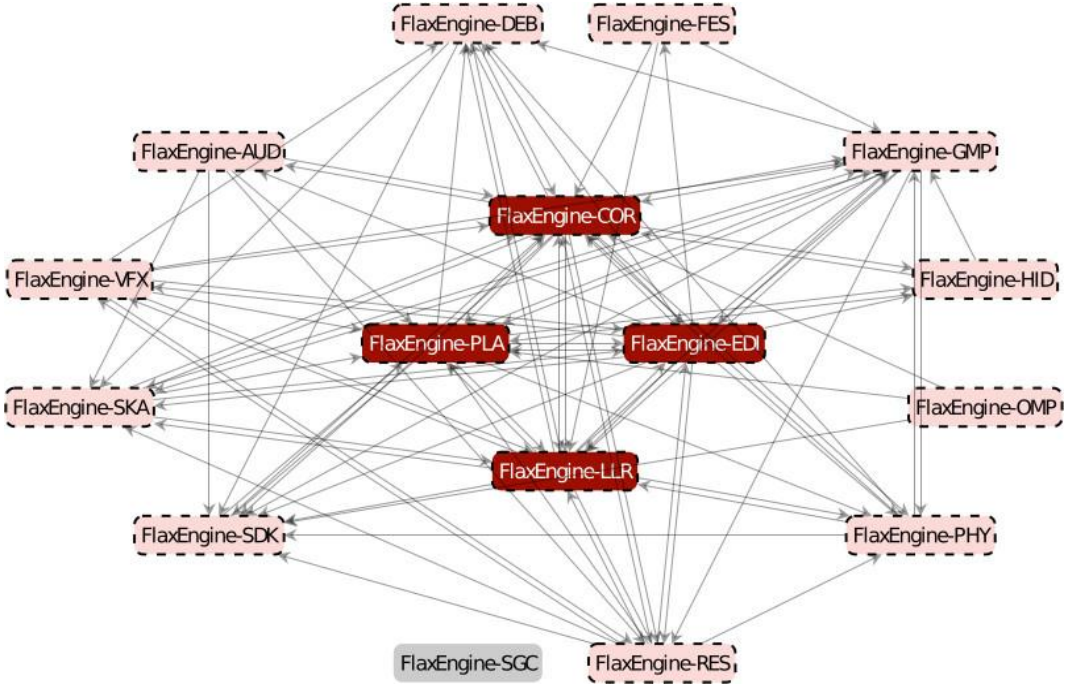
Gameplay3D

RQ1: Which Subsystems Are More Often Coupled With One Another?



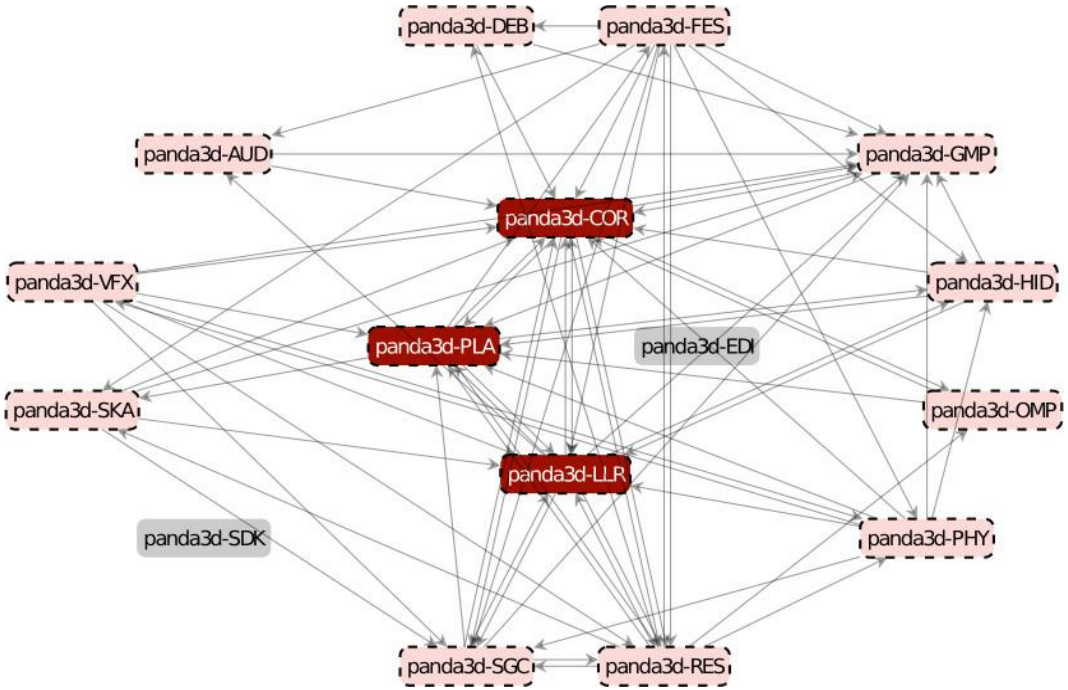
Godot

RQ1: Which Subsystems Are More Often Coupled With One Another?



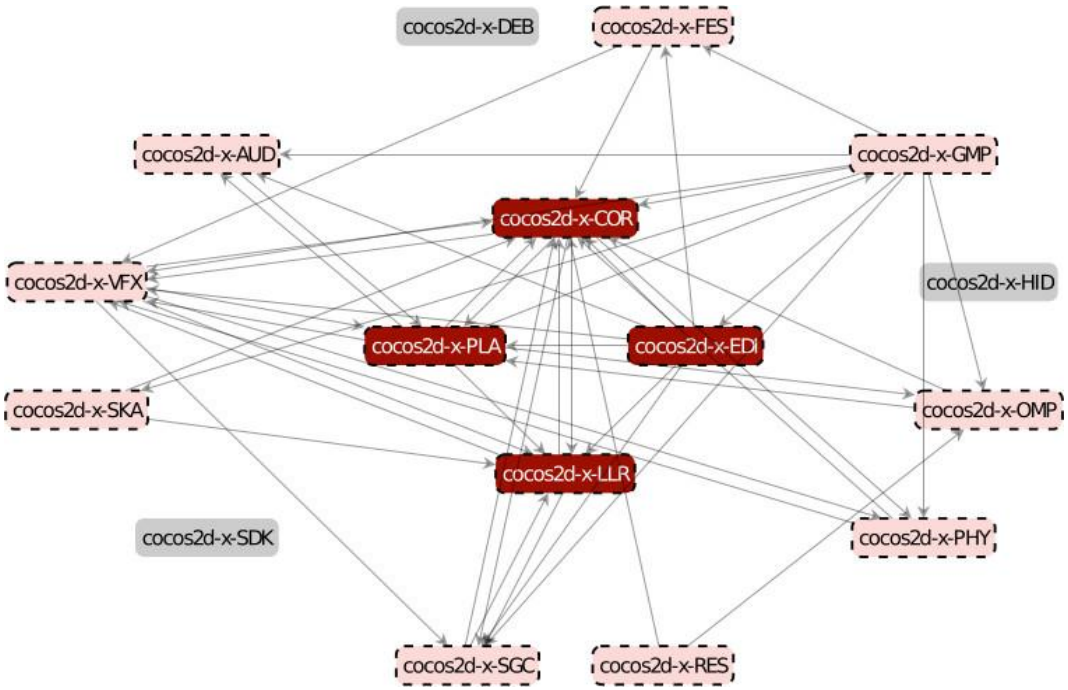
FlaxEngine

RQ1: Which Subsystems Are More Often Coupled With One Another?



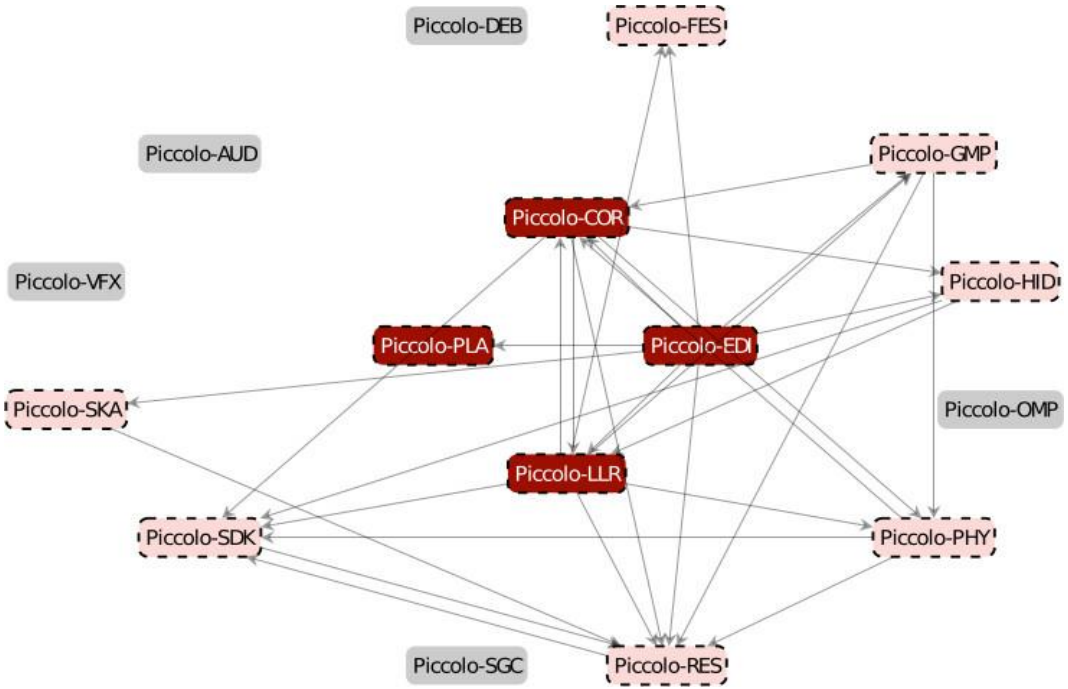
Panda3D

RQ1: Which Subsystems Are More Often Coupled With One Another?



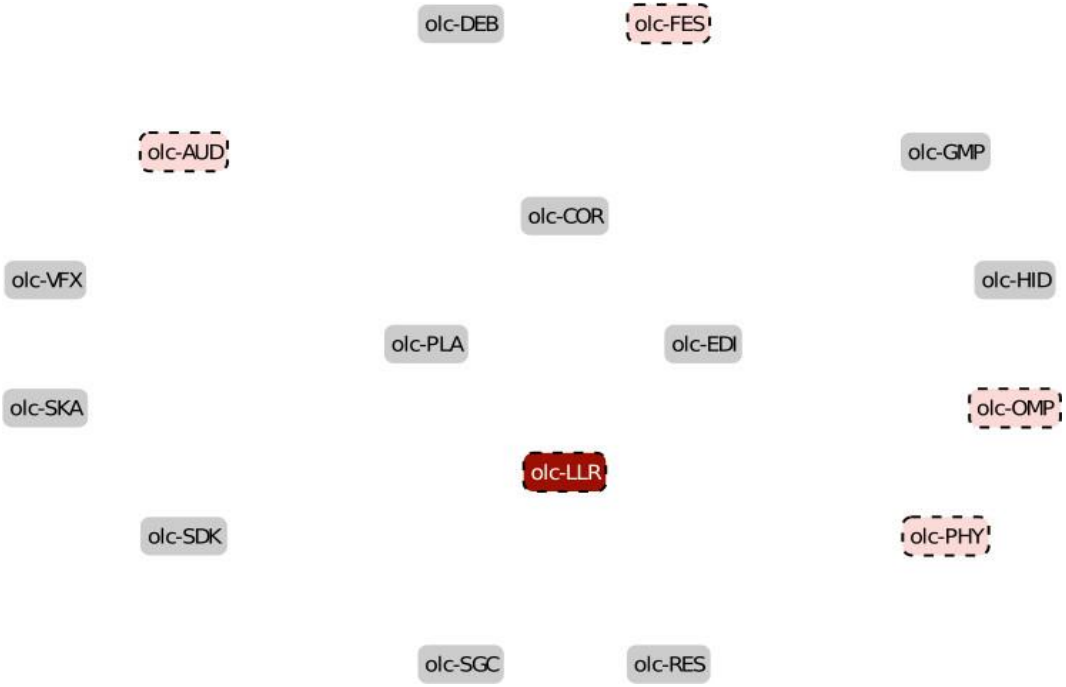
Cocos2d-x

RQ1: Which Subsystems Are More Often Coupled With One Another?



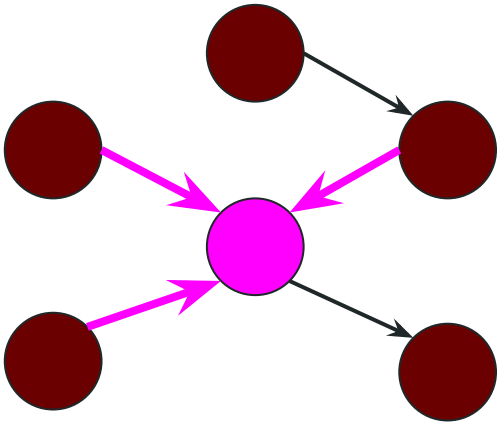
Piccolo

RQ1: Which Subsystems Are More Often Coupled With One Another?



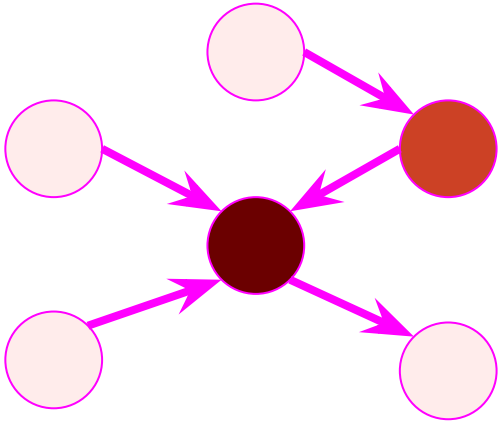
olcPixelGameEngine

RQ1: Which Subsystems Are More Often Coupled With One Another?



In-degree

Incoming edge count

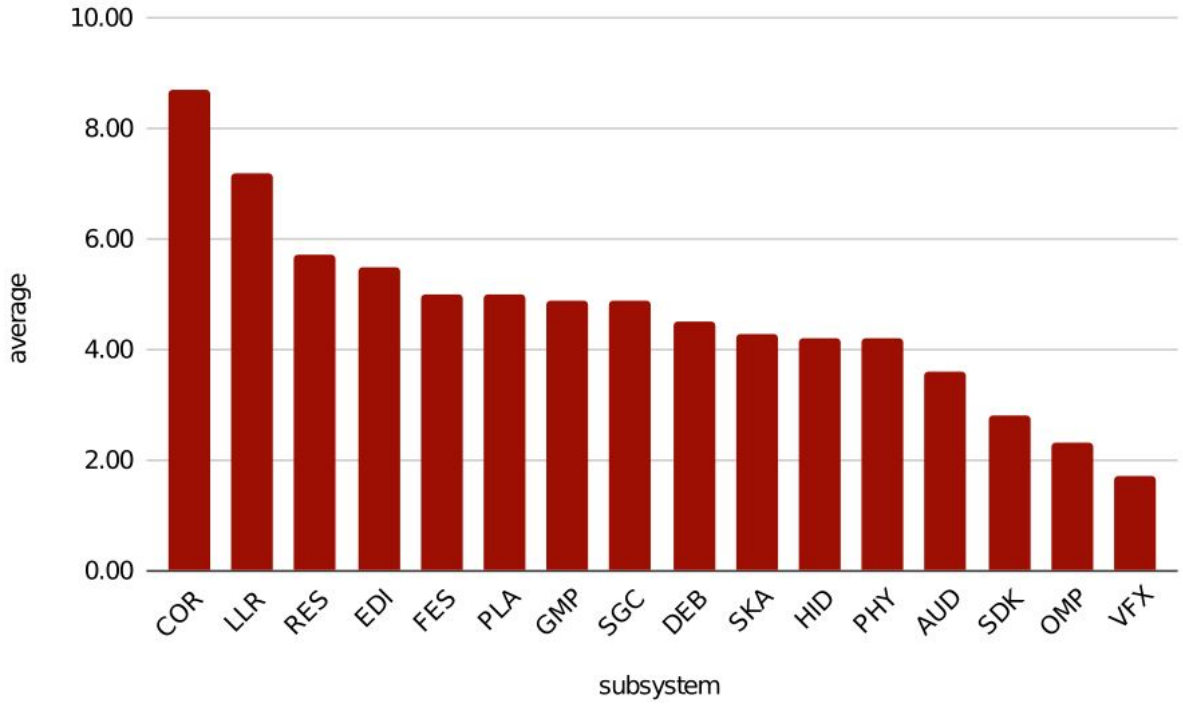


Betweenness Centrality

How frequently a node is positioned between other nodes

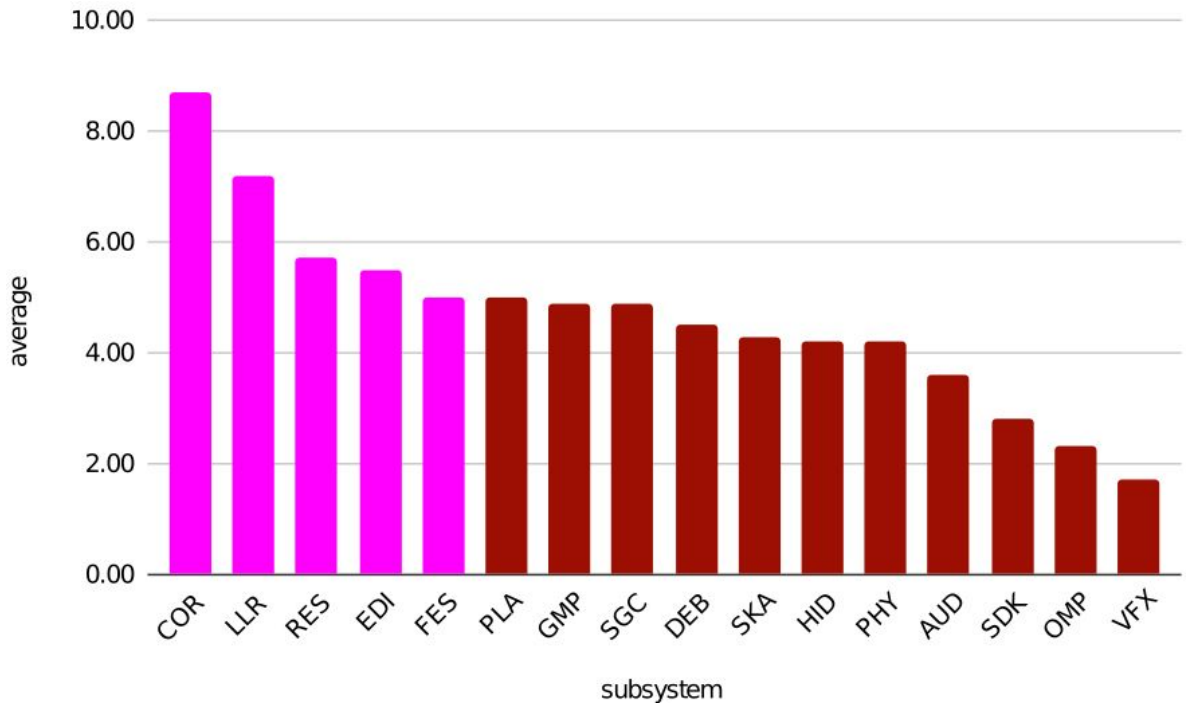
RQ1: Which Subsystems Are More Often Coupled With One Another?

Average In-Degree Overall



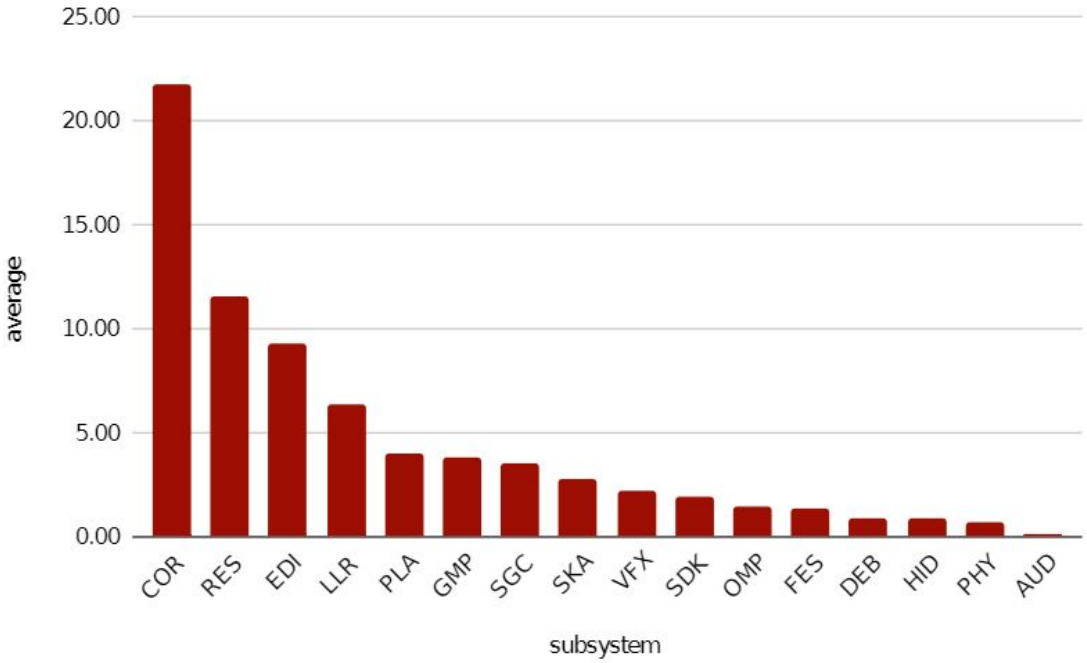
RQ1: Which Subsystems Are More Often Coupled With One Another?

Top Average
In-Degree Overall



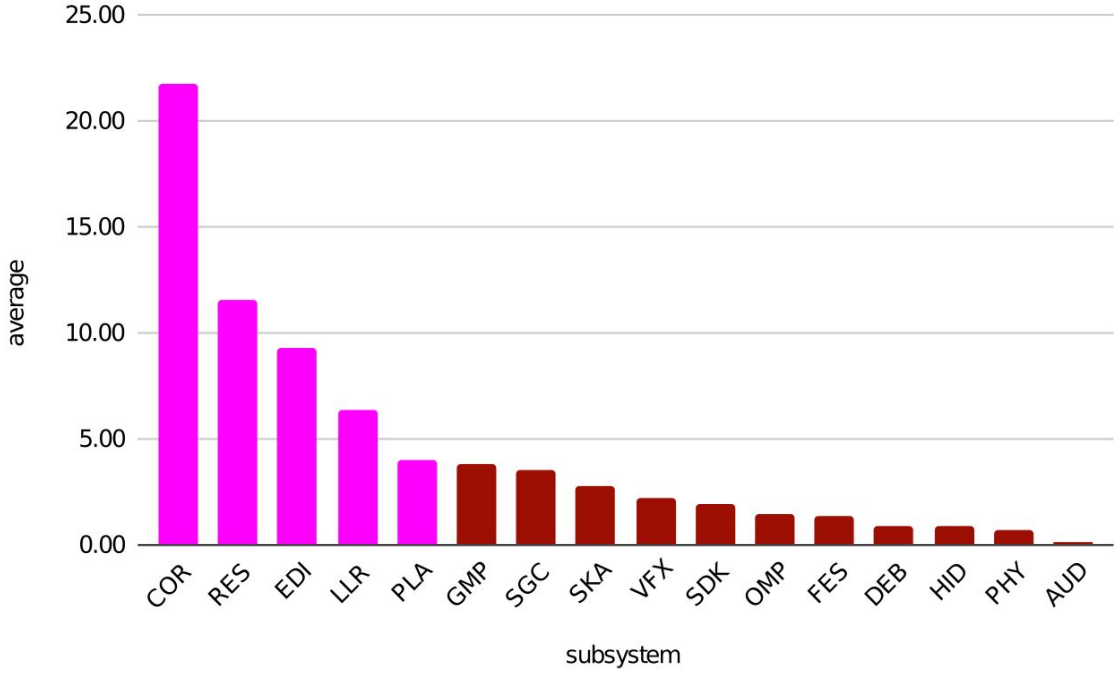
RQ1: Which Subsystems Are More Often Coupled With One Another?

Top Average Centrality Overall



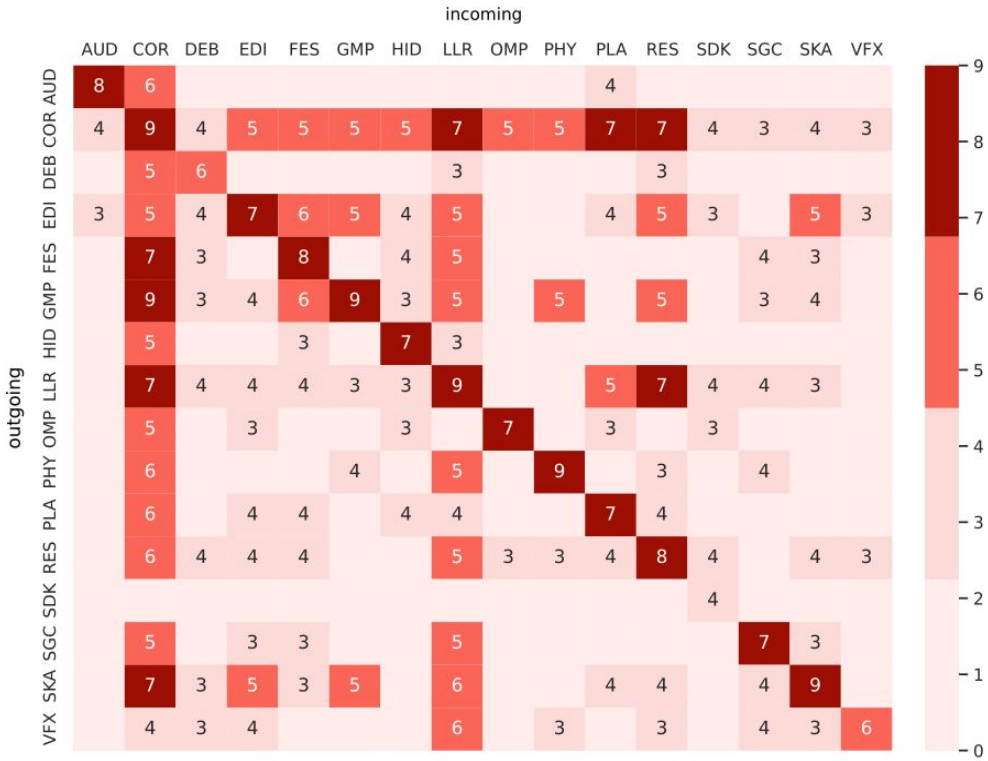
RQ1: Which Subsystems Are More Often Coupled With One Another?

Top Average Centrality Overall



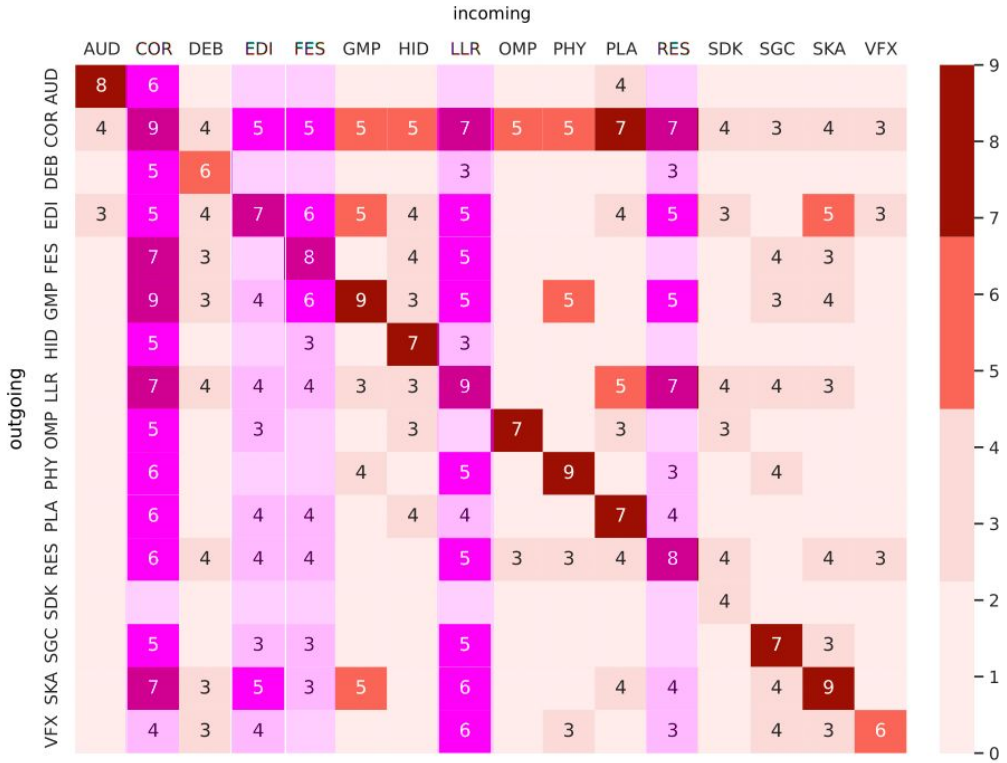
RQ2: Do Game Engines Share Subsystem Coupling Patterns?

In 10 game engines, how often subsystem A includes B?



RQ2: Do Game Engines Share Subsystem Coupling Patterns?

Top In-degree highlighted

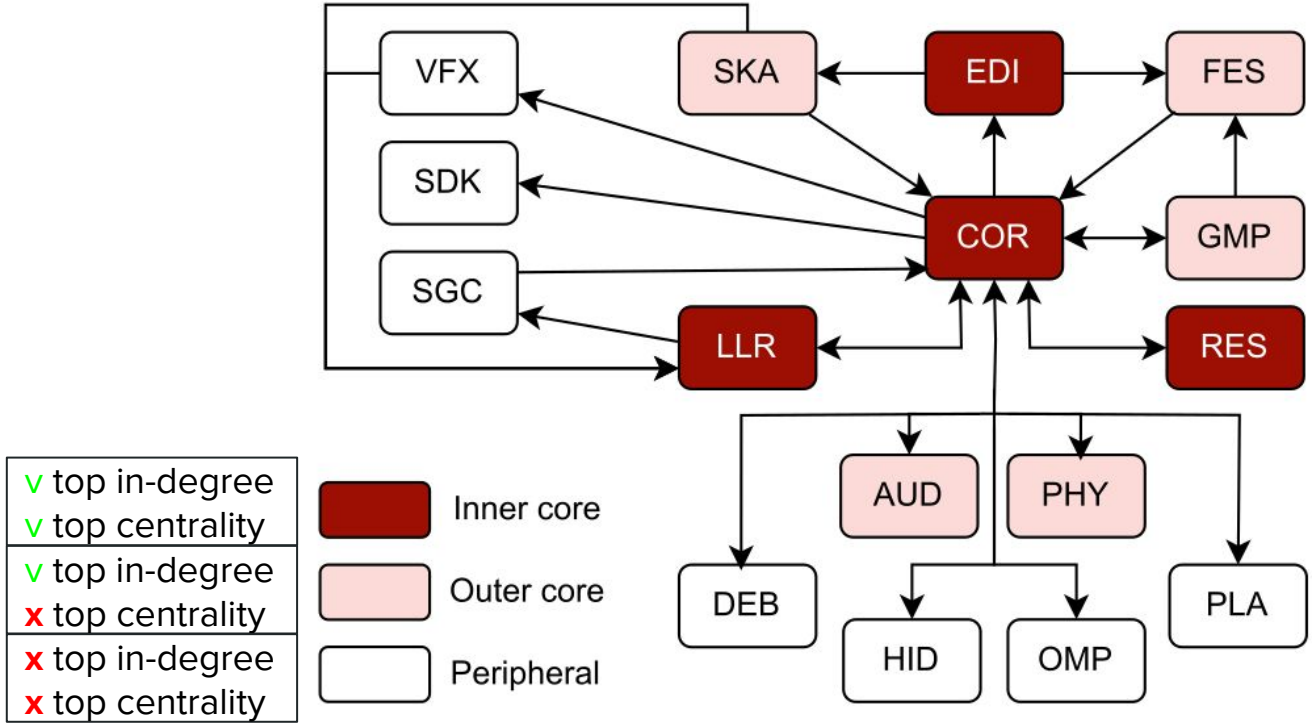


Emergent Game Engine Architecture

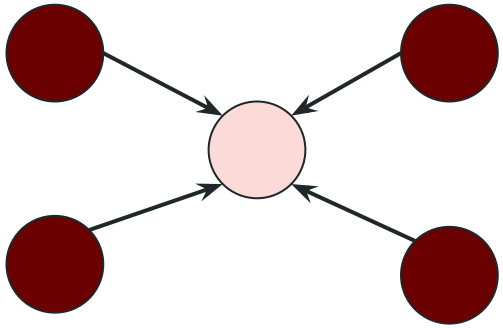
For each subsystem we take:

1. The most frequent incoming pair
2. The most frequent outgoing pair
3. All pairs go to a new graph
4. Graph nodes are color-coded based on in-degree and centrality
5. Emergent architecture!

Emergent Game Engine Architecture

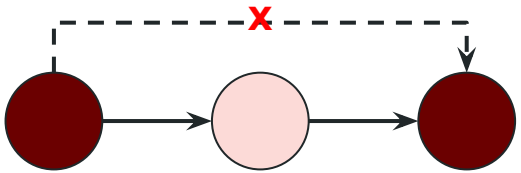


Subsystem Roles



Foundation

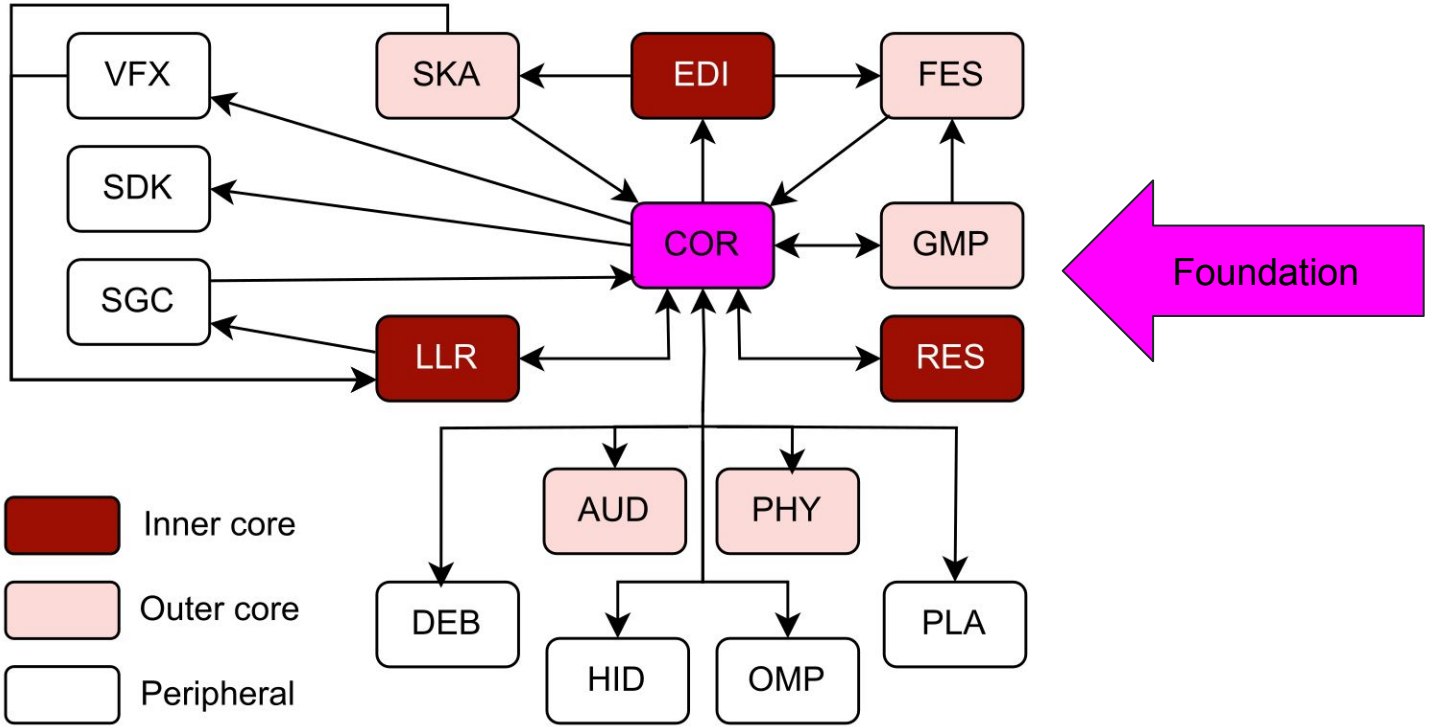
A subsystem frequently depended by others



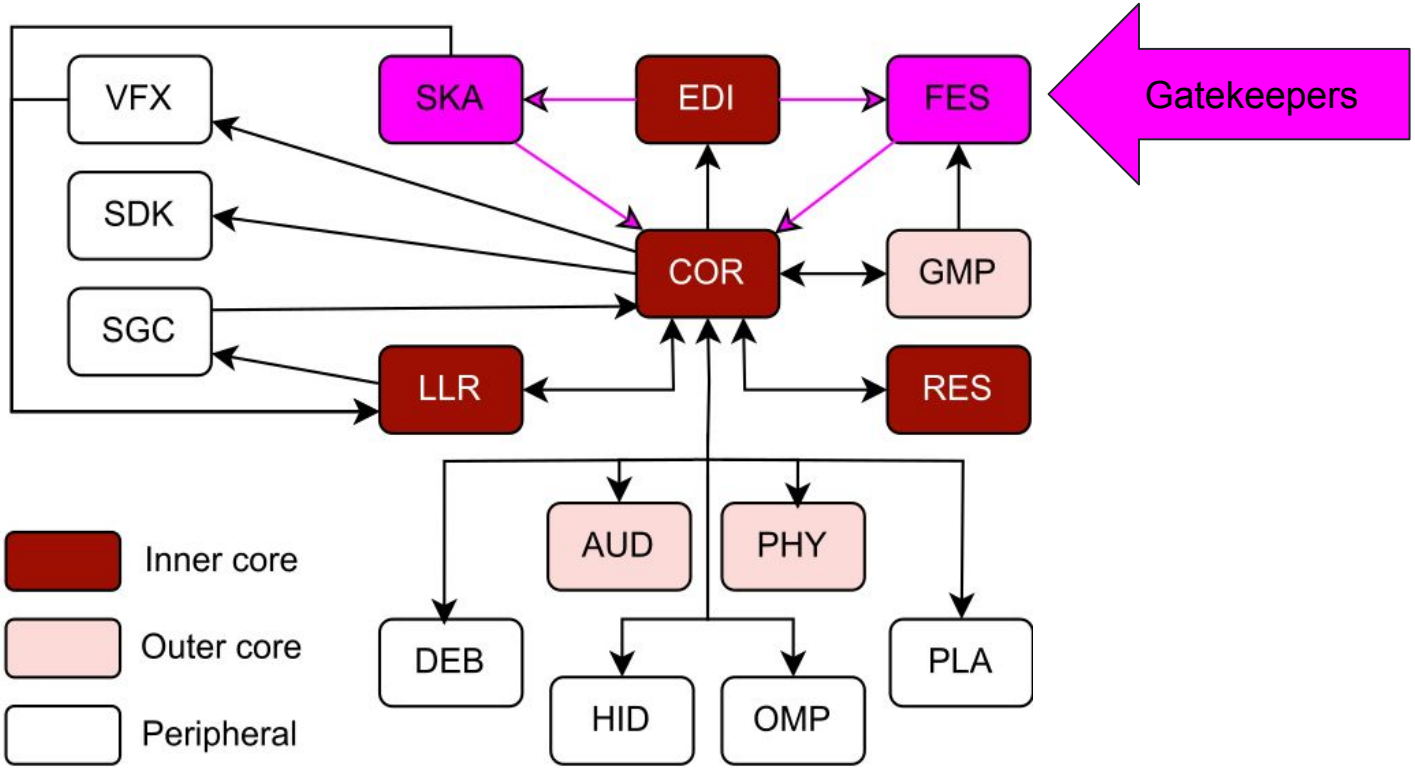
Gatekeeper

A subsystem acting as intermediate

RQ2: Do Game Engines Share Subsystem Coupling Patterns?



RQ2: Do Game Engines Share Subsystem Coupling Patterns?

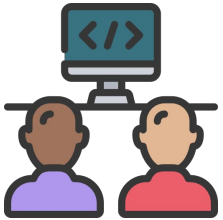


6. Evaluation

Evaluation Overview

Qualitative

To what extent do the game engines that we analysed with **SyDRA** match the software architecture literature?

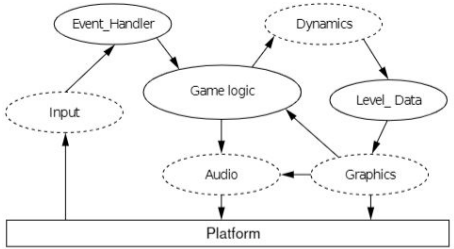


User Study

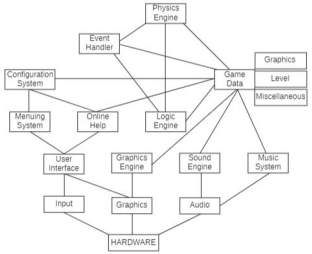
Does **SyDRA** help developers understand and maintain the architecture of game engines?

Qualitative Evaluation - Description

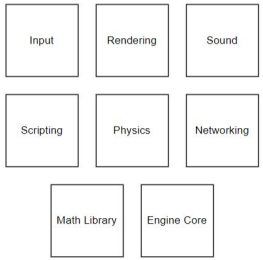
We compared Gregory with 4 other game engine architectures:



Bishop et al. (1998)



Rollings and Morris (2004)



Sherrod (2007)

Thorn (2010)

Qualitative Evaluation - Description

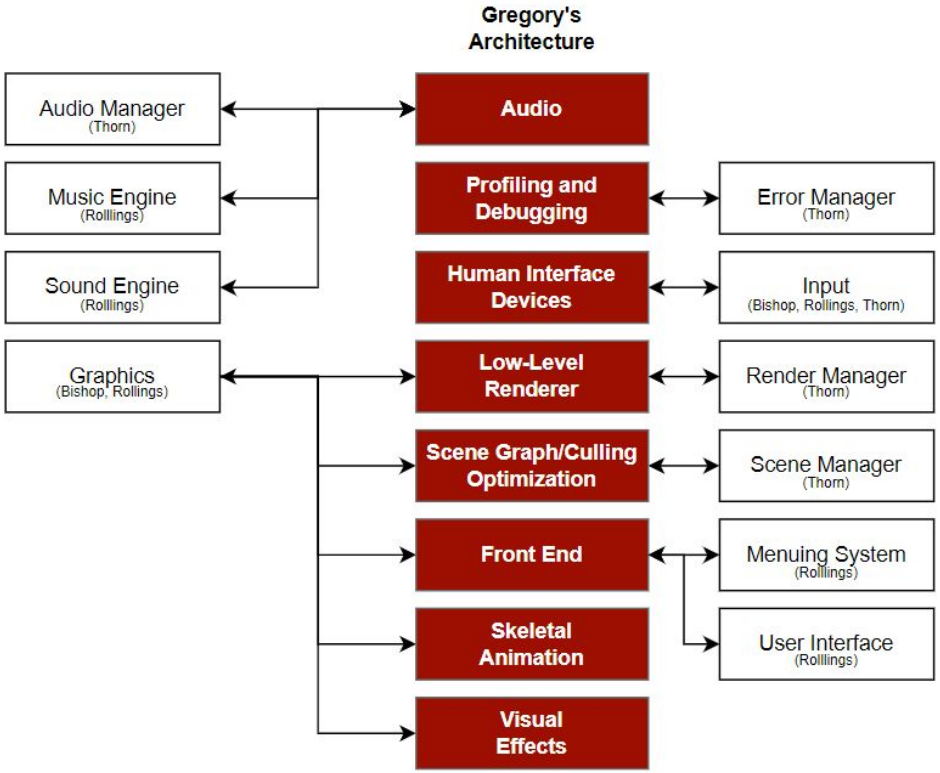
We compared the following aspects of subsystems:

Existence

Naming

Responsibilities

Qualitative Evaluation - Results



Qualitative Evaluation - Conclusion

- Two-way mapping
- Naming differs
- Subsystem responsibility descriptions differ

User Study - Description

Hypotheses

Tasks

Participants

Design

Procedures

Data Analysis

Adapted from Briand et al. "A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs". In IEEE Transactions on Software Engineering, 2001.

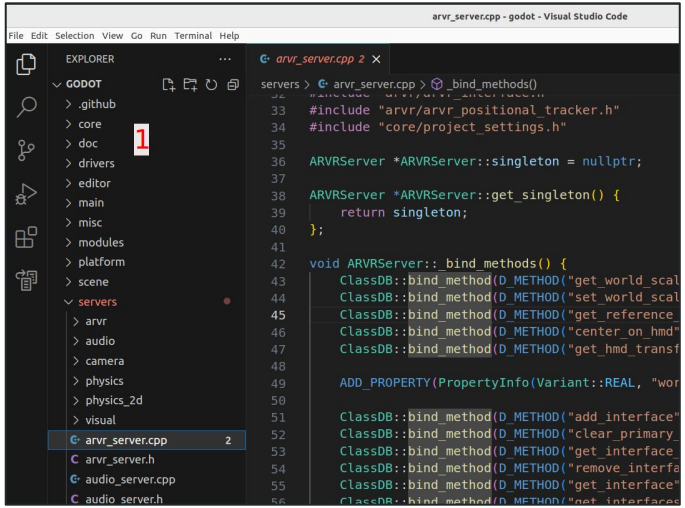
User Study - Hypotheses

- **H1:** Game engine architecture is significantly easier to understand with the use of SyDRA.
- **H2:** It is easier to perform impact analysis (locate changes) on game engines with the use of SyDRA.
- **H0 (Null Hypothesis):** The use of SyDRA provides no significant difference in the understandability and maintainability of game engine architecture.

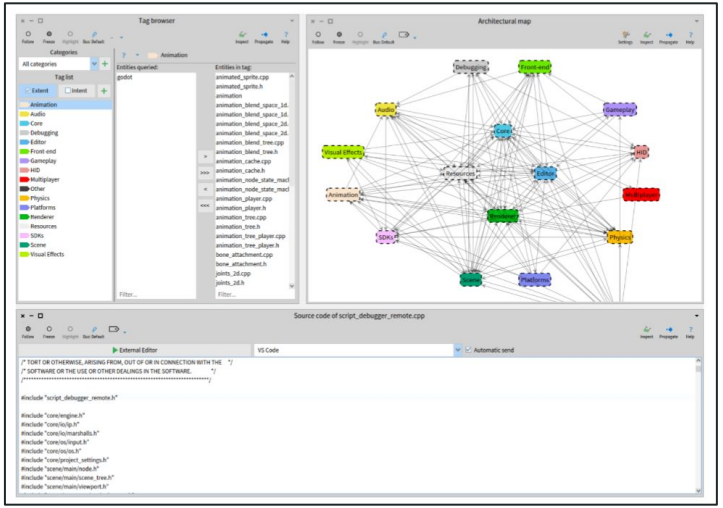
User Study - Participants

- 16 participants
- Required: age, experience with OO programming
- Most frequent demographics:
  - Men under 30 years old
  - From Canada and Brazil
  - Software developers outside the video game industry
  - 2 to 5 years of development experience
  - Unreal, Unity, or no experience at all

User Study - Procedures

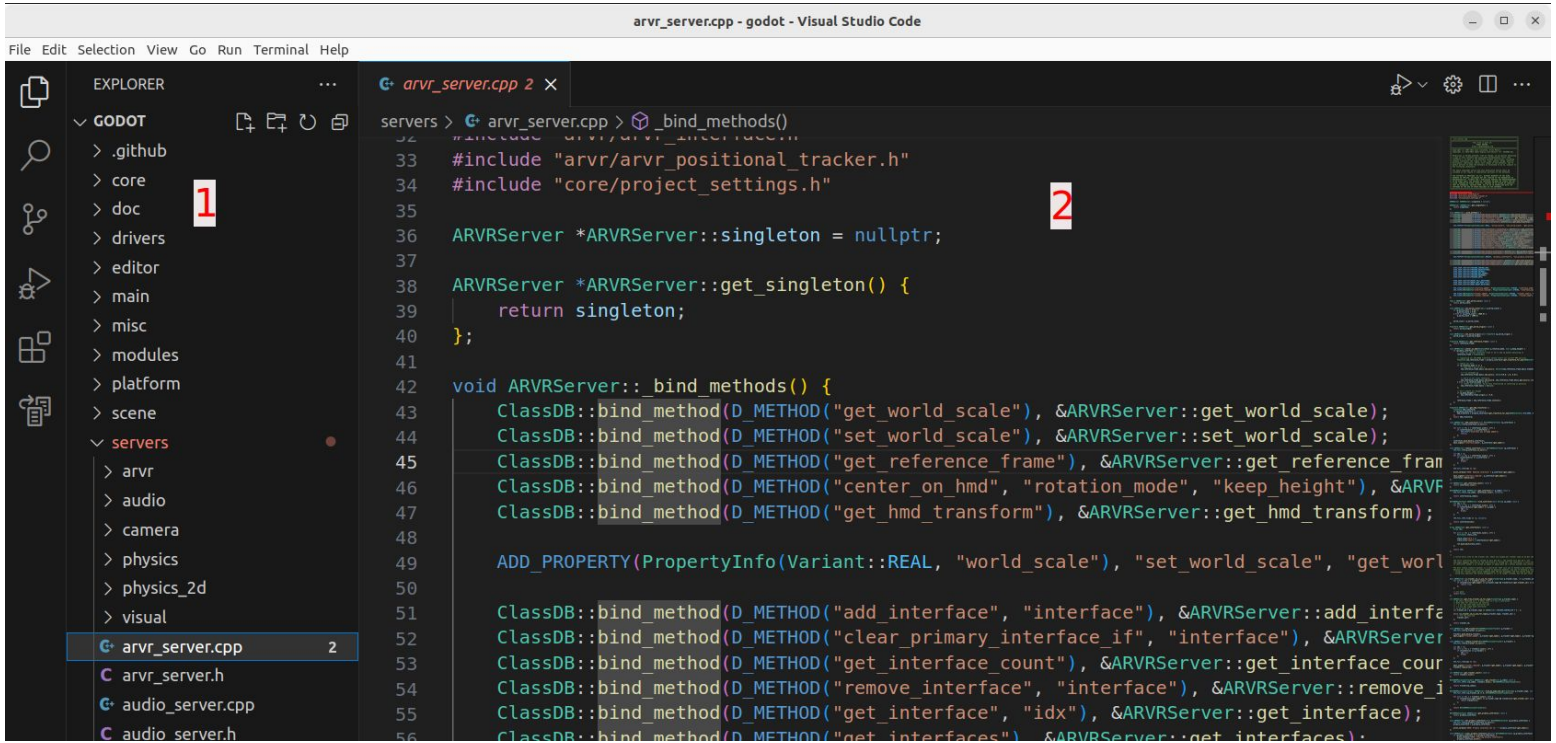


VS Code

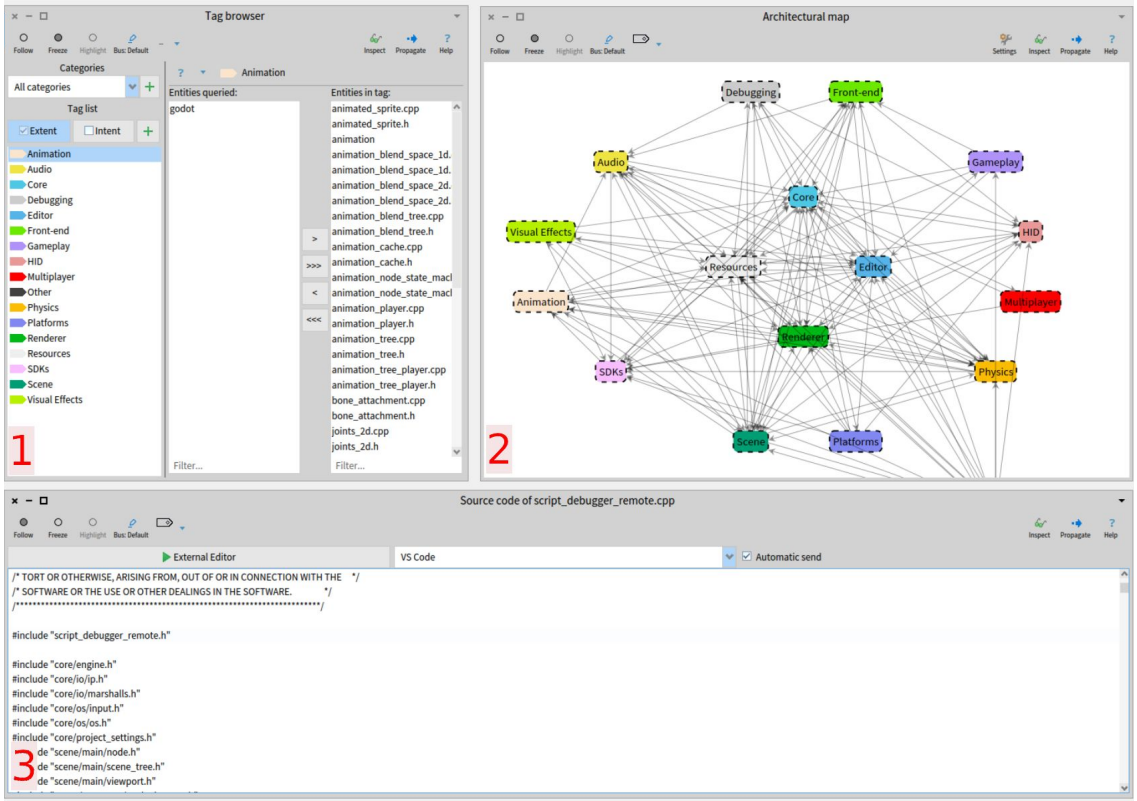


Moose

User Study - Procedures - VS Code



User Study - Procedures - Moose



User Study - Tasks

Architectural Understanding

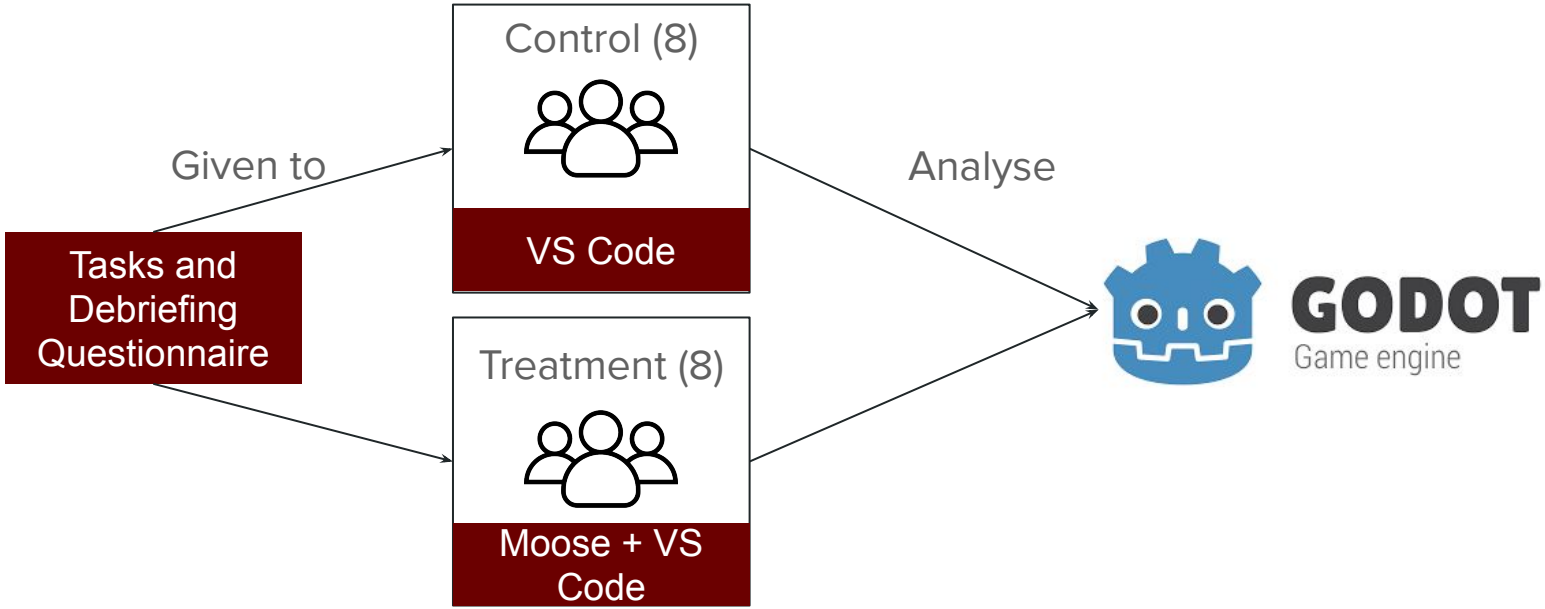
- "**Explain** the **functionalities** in this subsystem."
- "**Explain** what these files do and why they **depend** on each other."

Impact Analysis

- "If we were to **change** this file, which other files are affected?"
- "If we were to **remove** this file, which other files are affected?"

+ Debriefing questionnaire (NASA TLX)

User Study - Design



User Study - Data Analysis

- **UndTime:** Time spent on architectural understanding tasks (s).
- **UndCorr:** Correctness of architectural understanding tasks.
- **ModTime:** Time spent on impact analysis tasks (s).
- **ModCorr:** Participant Correct File Count / Actual Correct File Count.
- **ModComp:** Participant Total File Count / Actual Correct File Count.
- **ModRate:** Participant Correct File Count / ModTime.

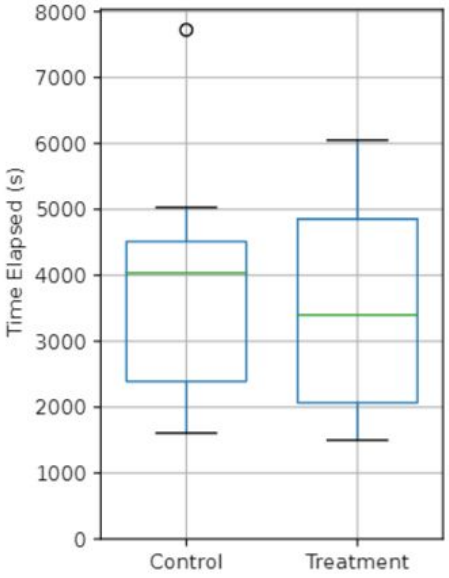
Adapted from Briand et al. "A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs". In IEEE Transactions on Software Engineering, 2001.

User Study - Data Analysis

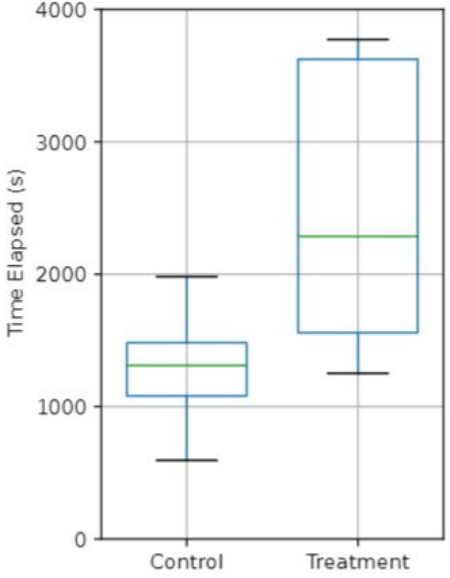
- Minimum number of participants (Two-sample t-test)
- Normality testing (Kolmogorov-Smirnov, Shapiro-Wilks' W)
- Statistical significance testing (Wilcoxon Matched Pairs)
- Descriptive statistics
- Effect size

Adapted from Briand et al. "A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs". In IEEE Transactions on Software Engineering, 2001.

User Study - Task Results

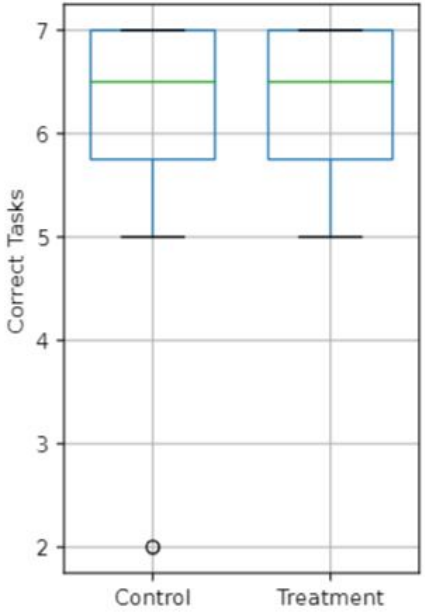


(a) *UndTime*

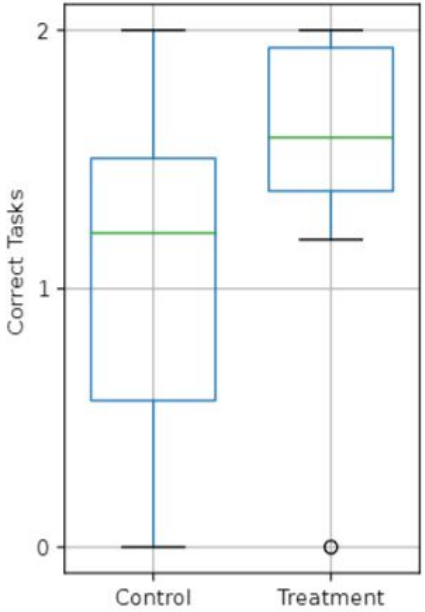


(b) *ModTime*

User Study - Task Results



(c) *UndCorr*



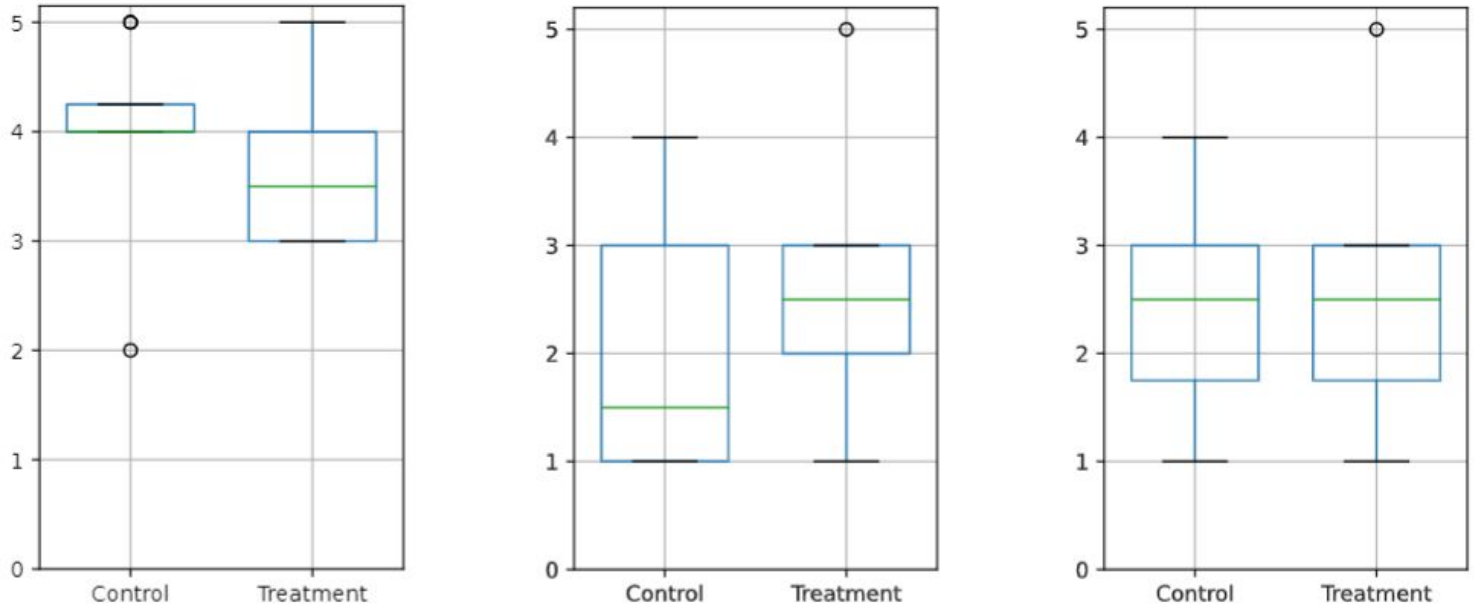
(d) *ModCorr*

User Study - Task Results

Table 5.8: Effect size for each dependent variable

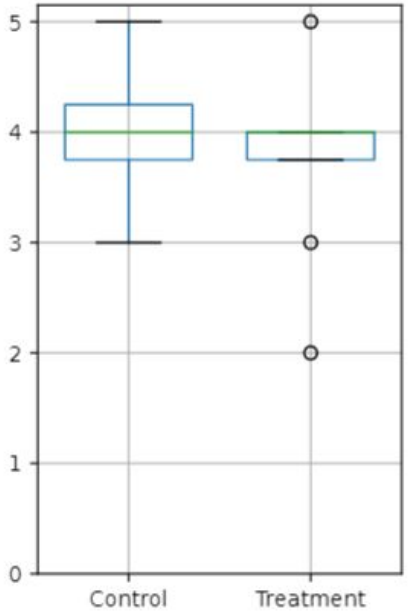
Variable	$\overline{X}$ Control	$\overline{X}$ Treatment	Effect Size	Z
UndTime	4,377.78	3,545.19	0.368	20.0
UndCorr	5.87	6.25	0.303	10.5
ModTime	1,282.73	2,482.08	1.71	5.0
ModComp	1.02	2.04	1.48	5.0
ModCorr	1.09	1.46	0.55	7.0
ModRate	0.00	0.00	0.23	12.0

User Study - Debriefing Results

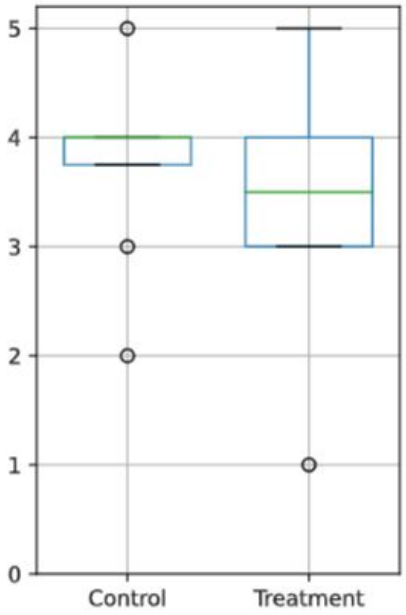


(a) Mental Demand (b) Physical Demand (c) Temporal Demand

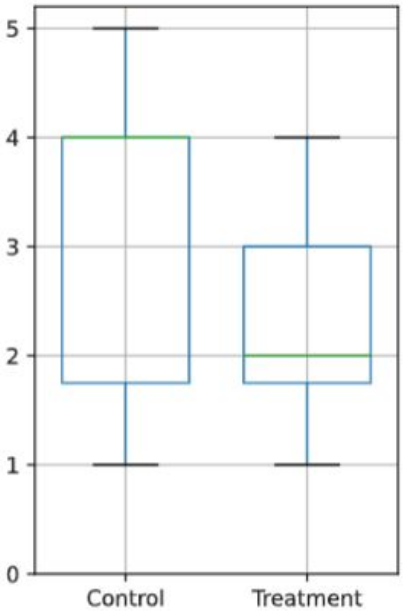
User Study - Debriefing Results



(d) Success



(e) Effort



(f) Frustration

User Study - Exit Interview Comments

"It's a painful question… What is a functionality? Both the terms and the scope of the question are way too abstract/large."

"The relationship lines [edges] always stand out from all other components, and the lack of a border on the windows when I expand a node is very confusing."

"Mostly I found it a bit difficult to read when expanding modules within modules, as everything starts to overlap."

User Study - Conclusion

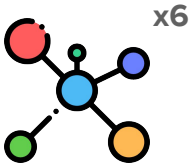
- We **accept H1** and **H2**, we **reject H0**
- Influence in time and correctness
- Impact Analysis Correctness
- Treatment: lower task load

7. Conclusion

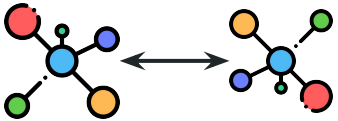
Conclusion

- Game engine **source code** can be used to create **architectural models**
- We show model **visualisations** can be used to analyse:

Division of responsibility



Frequent subsystem coupling

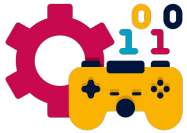


Analyse game engines isolatedly or as a group

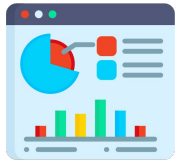
Limitations

We are aware of the impact of our choices of:

Reference architecture



Game engines and subsystems



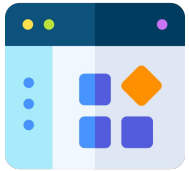
Models and Visualisations

Manual processes

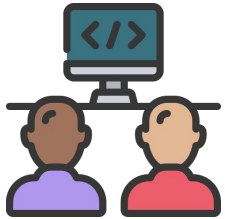
Future Work

Improvement and evaluation of the of the approach and implementation:

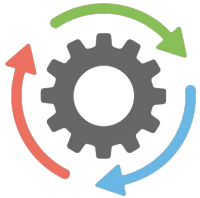
Include more architectures



Study other software families



Expanded user study



Automate subsystem detection

SyDRA: An Exploratory Approach to Game Engine Architecture Recovery

Gabriel C. Ullmann
Concordia University

Supervised By: **Dr. Yann-Gaël Guéhéneuc**
Concordia University

Dr. Fabio Petrillo
École de Technologie Supérieure