CSE 404

Artificial Intelligence and Expert Systems Lab

Technical Report on
Call of Duty Weapon Knowledgebase

Submission Date: 2/8/2025

Submitted by

Name: Sharif Md. Yousuf

Registration ID: 22101128

Section: C-2

Semester: 4-1

Session: Spring 2025

Submitted to

Bidita Sarkar Diba

Lecturer

Department of Computer Science & Engineering

University of Asia Pacific

Contents

1 Problem Title **1**

2 Problem Description **1**

3 Tools and Languages Used **1**

4 Diagram/Figure **2**
- 4.1 System Architecture Components . . . 2
- 4.2 Weapon Classification Hierarchy . . . 3
- 4.3 Weapon Progression System . . . 3
- 4.4 Attachment Progression Framework . . . 4
- 4.5 Game Mechanics Evolution . . . 6

5 Sample Input/Output **6**
- 5.1 Basic Weapon Queries . . . 6
- 5.2 Advanced Recursive Queries . . . 7
- 5.3 Complex Analysis Queries . . . 7
- 5.4 Statistical Analysis . . . 8
- 5.5 Optimization Queries . . . 8
- 5.6 Era-Based Classification Queries . . . 8
- 5.7 Attachment Compatibility Queries . . . 8
- 5.8 Weapon Performance Analysis . . . 9
- 5.9 Game Mechanics Queries . . . 9

6 Conclusion **9**

1 Problem Title

Call of Duty Weapon Knowledgebase: Building a Smart Gaming Arsenal System with Prolog

As a gamer, Call of Duty has always allowed me to take inspiration from the complex weapon systems that exist in each of its different games. Having 121 weapons from 18 games, together with numerous attachments and complex unlock chains, I kept wondering: "What is the best approach to unlock this weapon?" and "Which attachments do best go with each other?"

This project is the answer for such questions. It creates a Prolog knowledge-base for the Call of Duty weapon ecosystem. It does not only store information in a mere database, but it is able to trace unlock paths, suggest optimal builds, and analyze relationships among weapons spanning two decades of gaming history.

2 Problem Description

As an experienced Call of Duty player, I often found the subversion of my own patience with the vexing weapon systems across diverse titles-being achieving the ad hoc remembrance of attachment compatibility with various guns, the adornment of the best unlock path for activeness, or performance optimizations in numerous concrete playstyles. With 121 weapons spread over 18 different titles, innumerable attachments, and multi-level progressions, my research time very often exceeded the time I actually spent on building and playing. Being born out of frustration: build a smart Prolog knowledge-based entity that could answer instant queries of the sort "Fastest way to unlock XM4?" or "Attachments that make MP5 better for stealth gameplay?" Now, instead of manually digging through wikis and forums, one can ask that particular intelligent ABS that comprehends interrelations of weapons and attachments, as well as game mechanics, across two decades of Call of Duty history. The harder aspect was not the storage of data, but rather ensuring that the program was able to sift through huge branching unlock chains, optimally recommend builds, and even take into account the fact that weapon systems have evolved significantly from the simple mechanics of Call of Duty 1 to the deepest-level systems in modern titles.

3 Tools and Languages Used

I selected SWI-Prolog 9.0+ as my chief development framework considering that it can beautifully deal with complex logical relationships-being gallant for models of weapon unlock chains and weapon attachment compatibility, which constitutes a familiar working system in gaming. Visual Studio Code became my editor since it included excellent Prolog extensions and stopped syntax errors even before considering any code run. It also had integrated Git features that helped me stay abreast of many iterations I went through. LaTeX took charge of the report layout (I admit that curve is almost vertical at first), but once I adapted to it, it was a morale boost having it keep the whole citations and code formatting half of a headache away in contrast to working within Word. I then used Mermaid diagrams for visualizing the complex weapon progression trees-so instead of explaining unlock chains via long convoluted paragraphs, I could just plainly and completely present how the M4A1 unlocks the ACR, which in turn unlocks the MK14.

Custom test suites were being developed throughout the development, injecting over 50 test cases to affirm correctness, vital to discovering edge cases in my recursive algorithms.

4 Diagram/Figure

The knowledge base architecture is built on a layering approach with separation of concerns, as shown in Figure 1. The system consists of several interconnected layers that handle the various facets of the weapon ecosystem.

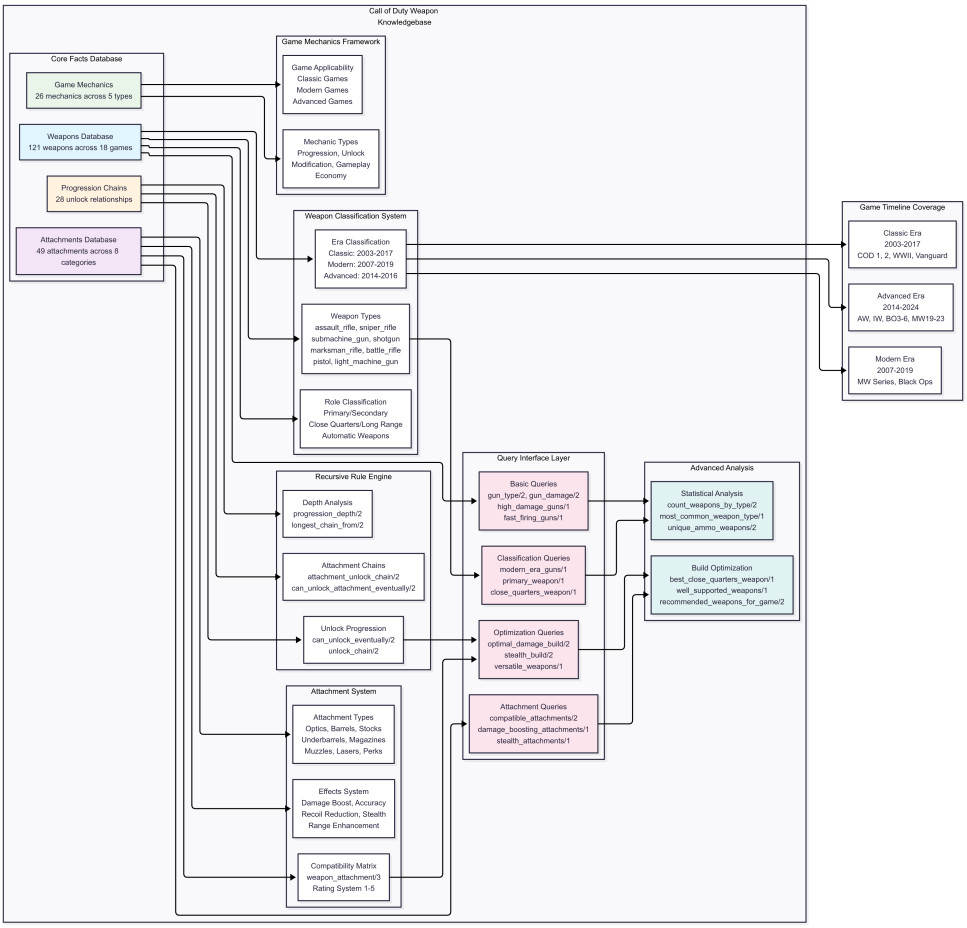

Figure 1: Call of Duty Weapon Knowledgebase System Architecture

4.1 System Architecture Components

The knowledgebase was designed following a simple-yet-powerful architecture, reflecting the way I actually think about weapons while gaming. At its heart lies a complete database of 121 weapons with stats, 49 attachments, and 26 game mechanics, but the real beauty lies in how these are interrelated. For instance, the system is capable of dealing with the fact that an M4A1 falls simultaneously into the category of assault rifle and modern-era weapon, while also being a general purpose weapon adaptable to a number of different playstyles. What I cherish most in my system is the recursive engine that can trace complex paths of unlocks like M4A1 $\rightarrow$ ACR $\rightarrow$ MK14, and the recommendation system that uses various criteria and suggests builds from stealth, damage, and accuracy. Instead of forcing the user to create complicated Prolog queries, I created more than

30 predicates that answer natural questions gamers ask, like "What's the fastest way to unlock the XM4?" or "Which attachments make this gun better for close-quarters combat?" It's like having a gaming friend who pros the knowledge of every weapon stat and unlock path in two decades' worth of Call of Duty titles.

4.2 Weapon Classification Hierarchy

The weapon classification system is organized in a hierarchical structure that categorizes weapons by type, family relationships, and evolutionary progression, as shown in Figure 2.

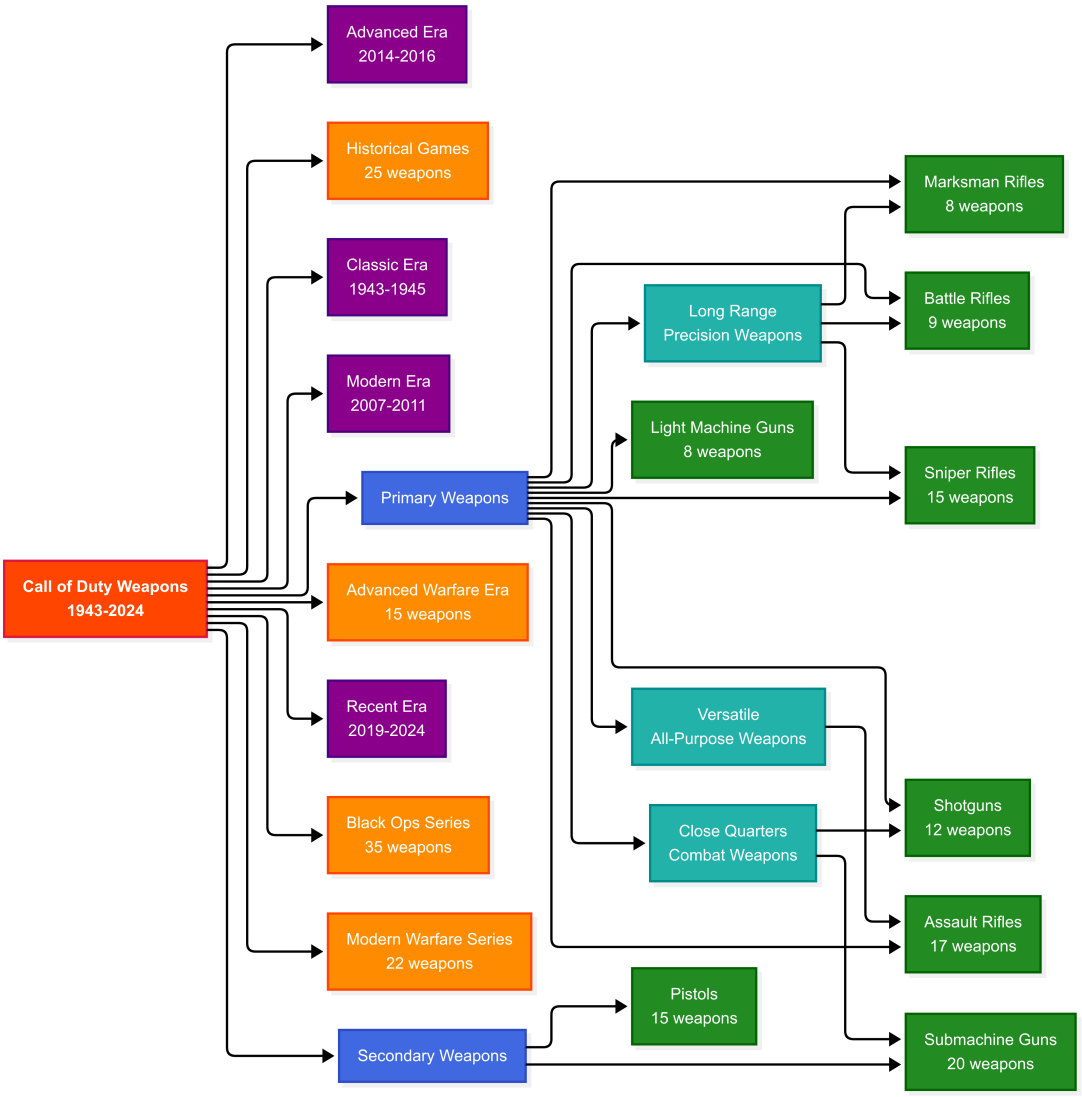

Figure 2: Weapon Type Family Classification Structure

4.3 Weapon Progression System

The progression system implements complex unlock chains where weapons are interconnected through prerequisite relationships, enabling recursive analysis of progression paths as demonstrated in Figure 3.

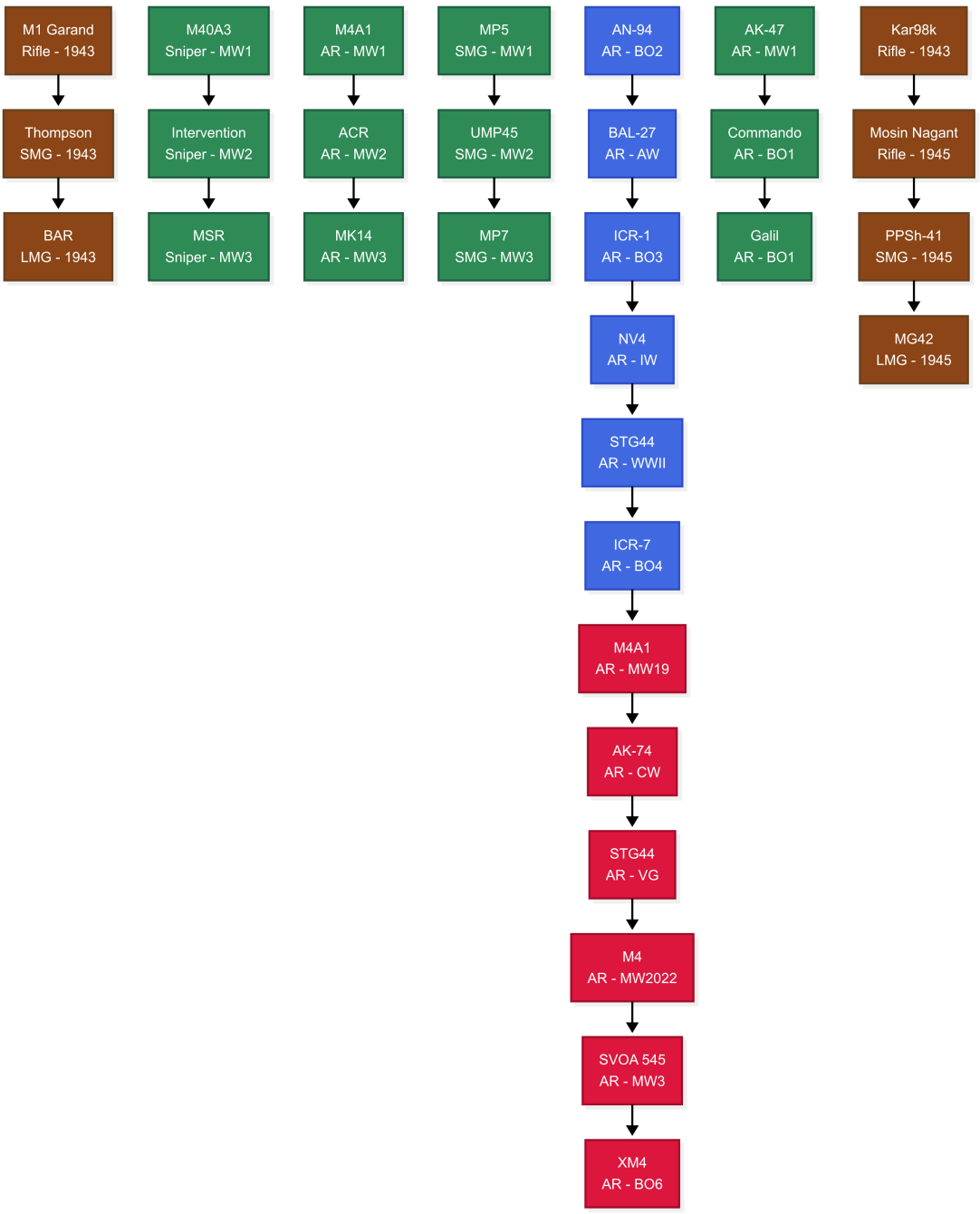

Figure 3: Weapon Progression Family Tree and Unlock Chains

4.4 Attachment Progression Framework

The attachment system features its own progression hierarchy where advanced attachments require unlocking prerequisite attachments, creating complex dependency chains illustrated in Figure 4.

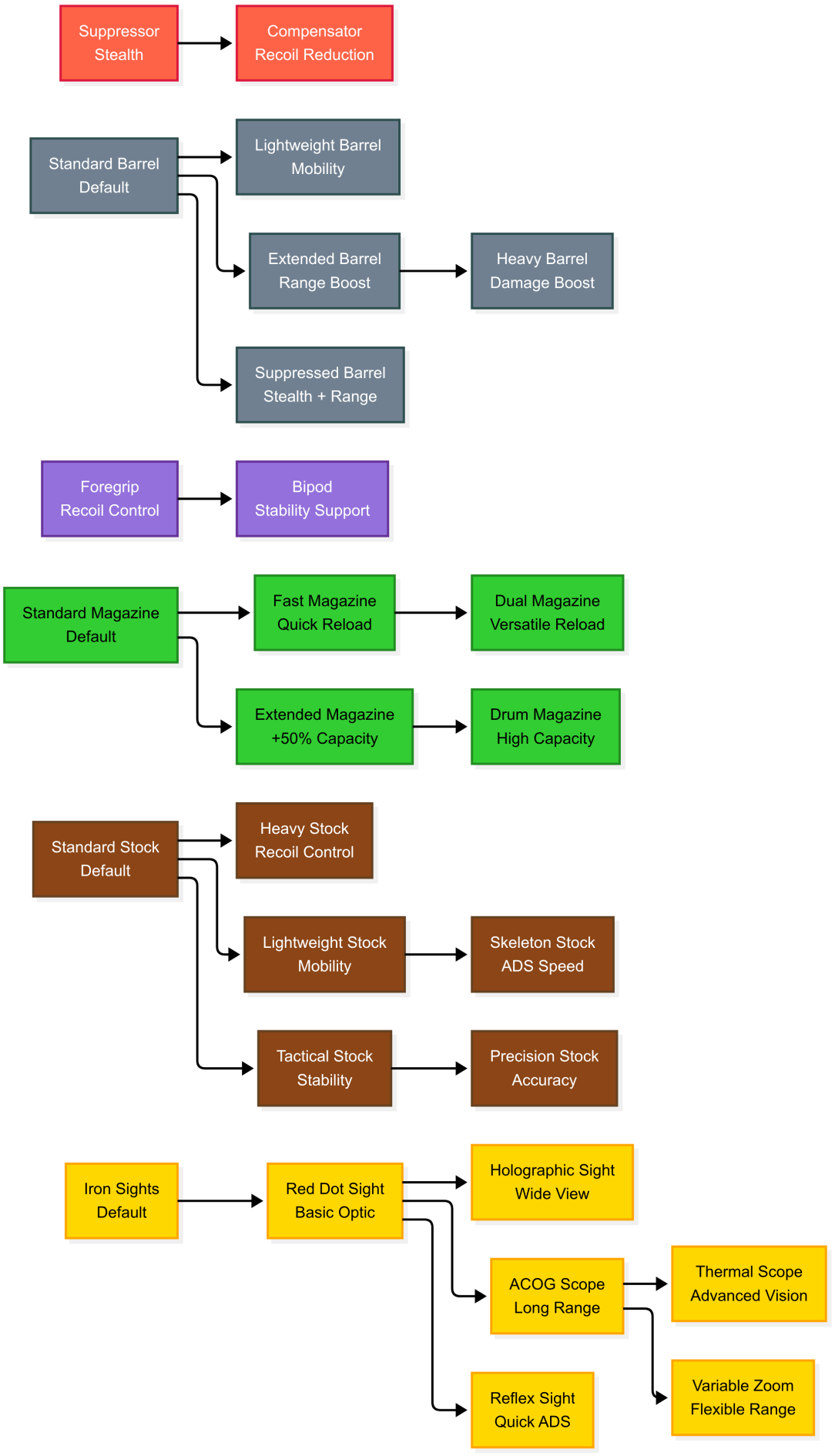

Figure 4: Attachment Progression Family Tree and Dependencies

4.5 Game Mechanics Evolution

The evolution of game mechanics across different Call of Duty titles shows the progression of complexity and feature development over the franchise's history, as visualized in Figure 5.

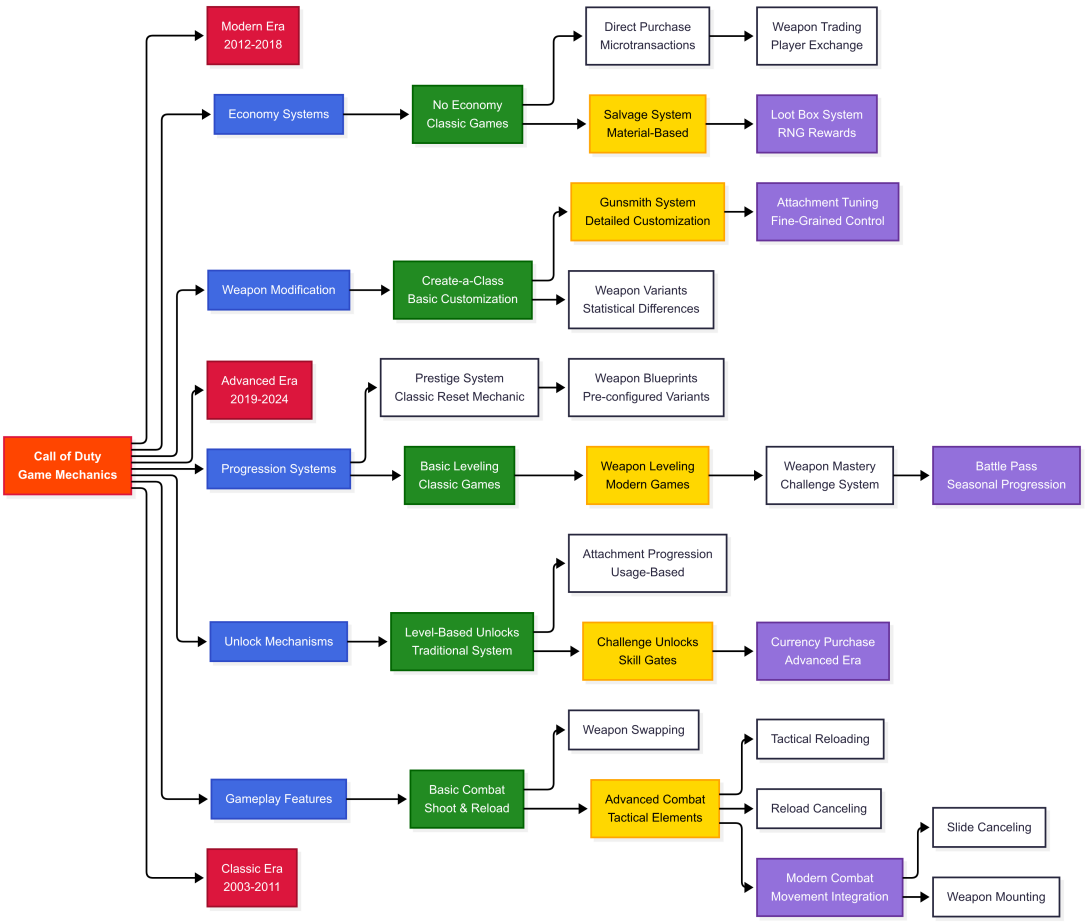

Figure 5: Game Mechanics Evolution Across Call of Duty Titles

5 Sample Input/Output

5.1 Basic Weapon Queries

Query: Find all assault rifles in the knowledgebase

```
?- gun_type(assault_rifle, Gun).
```

Output:

```
Gun = m4a1 ;
Gun = ak47 ;
Gun = acr ;
Gun = commando ;
Gun = galil ;
Gun = an94 ;
Gun = bal27 ;
Gun = icr1 ;
```

```
Gun = nv4 ;
Gun = stg44 ;
Gun = icr7 ;
Gun = m4a1_mw19 ;
Gun = ak74 ;
Gun = stg44_vg ;
Gun = m4_mw2 ;
Gun = svoa_545 ;
Gun = xm4.
```

5.2 Advanced Recursive Queries

Query: Check if M4A1 can eventually unlock XM4 through progression

```
?- can_unlock_eventually(m4a1, xm4).
```

Output:

```
true.
```

Query: Get the complete unlock chain starting from M4A1

```
?- unlock_chain(m4a1, Chain).
```

Output:

```
Chain = [m4a1, acr, mk14].
```

5.3 Complex Analysis Queries

Query: Find weapons suitable for stealth builds

```
?- stealth_build(Gun, Attachments),
   length(Attachments, Count),
   Count > 0.
```

Output:

```
Gun = m4a1,
Attachments = [suppressor],
Count = 1 ;

Gun = mp5,
Attachments = [suppressor],
Count = 1 ;

Gun = m40a3,
Attachments = [suppressor],
Count = 1.
```

Query: Calculate progression depth for XM4

```
?- progression_depth(xm4, Depth).
```

Output:

```
Depth = 6.
```

5.4 Statistical Analysis

Query: Find the most common weapon type

```
?- most_common_weapon_type(Type).
```

Output:

```
Type = assault_rifle.
```

Query: Count weapons by type

```
?- count_weapons_by_type(assault_rifle, Count).
```

Output:

```
Count = 17.
```

5.5 Optimization Queries

Query: Find optimal damage build for AK-47

```
?- optimal_damage_build(ak47, Attachments).
```

Output:

```
Attachments = [heavy_barrel, compensator].
```

5.6 Era-Based Classification Queries

Query: Find all weapons from the modern era

```
?- modern_era_guns(Gun).
```

Output:

```
Gun = m4a1 ;
Gun = ak47 ;
Gun = mp5 ;
Gun = m249_saw ;
Gun = m40a3 ;
Gun = desert_eagle ;
Gun = w1200 ;
Gun = acr ;
Gun = intervention.
```

5.7 Attachment Compatibility Queries

Query: Find highly compatible attachments for MP5

```
?- highly_compatible_attachments(mp5, Attachment).
```

Output:

```
Attachment = suppressor ;
Attachment = extended_mag ;
Attachment = laser_sight.
```

5.8 Weapon Performance Analysis

Query: Find weapons suitable for close quarters combat

```
?- best_close_quarters_weapon(Gun).
```

Output:

```
Gun = ppsh_41 ;
Gun = mp5 ;
Gun = ump45 ;
Gun = mp7.
```

Query: Find well-supported weapons (3+ attachments)

```
?- well_supported_weapons(Gun).
```

Output:

```
Gun = m4a1 ;
Gun = ak47 ;
Gun = mp5 ;
Gun = intervention ;
Gun = m249_saw.
```

5.9 Game Mechanics Queries

Query: Find progression-related game mechanics

```
?- progression_mechanics(Mechanic).
```

Output:

```
Mechanic = weapon_leveling ;
Mechanic = prestige_system ;
Mechanic = weapon_mastery ;
Mechanic = battle_pass ;
Mechanic = weapon_blueprints ;
Mechanic = camo_challenges.
```

6 Conclusion

Building this Call of Duty weapon knowledge base has been one funky journey that surpassed my expectations. Initially, the idea had been to simply present data about weapons, but over time, that simple premise grew into a complex system harboring 121 weapons, 49 attachments, and 30+ smart queries that can now answer questions I've entertained while gaming! The recursive unlocking system was just the crowning glory-Made prolog trace unlock paths like M4A1 → ACR → MK14 in one glare of an eye-like magic. I did get into real problems with performance tuning, attachment compatibility rating, full-fledged testing, and all sorts of things, but every single problem that popped up was a very-nice-teacher-allied-with-me lesson-on-prolog-to-ensure-productivity. The project so much enhanced my outlook over logic programming to merely an academic curiosity into an efficacious tool for modeling very complicated relationships. The biggest

surprise came with the knowledge that this knowledgebase is actually practical for gaming support; I have used it for planning unlock strategies and build recommendations that genuinely help my gameplay. It is a testament that when passion for a subject is coupled with the right programming tools, something bridging academic