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

Call of Duty Weapon Knowledgebase: A Comprehensive Prolog Implementation for Modeling Gaming Arsenal Systems and Recursive Progression Analysis

This project implements a sophisticated Prolog knowledgebase that models the extensive weapon ecosystem from the Call of Duty franchise, spanning over two decades of gaming history from 2003 to 2024. The system captures not only basic weapon attributes but also complex attachment systems, game mechanics, and recursive progression chains that mirror the intricate unlock systems found in modern gaming.

2 Problem Description

The gaming industry, particularly first-person shooter games like Call of Duty, features incredibly complex weapon systems that involve hundreds of weapons, attachments, and progression mechanics. Understanding these systems requires organizing vast amounts of interconnected data and implementing logical rules that can handle recursive relationships and complex queries.

2.1 Domain Complexity

The Call of Duty franchise presents a perfect case study for knowledge representation due to its:

- **Temporal Span**: 18 games released over 21 years (2003-2024)
- **Weapon Diversity**: 121 unique weapons across 8 different categories
- **Attachment Systems**: 49 attachments with compatibility matrices
- **Game Mechanics**: 26 different mechanics spanning 5 categories
- **Progression Systems**: Complex unlock chains requiring recursive analysis

2.2 Technical Challenges

Implementing this knowledgebase presented several technical challenges:

1. **Recursive Progression Modeling**: Weapons unlock in chains where accessing advanced weapons requires progressing through intermediate ones. This necessitated implementing recursive rules that could traverse these chains efficiently while avoiding infinite loops.
2. **Multi-Dimensional Classification**: Each weapon belongs to multiple classification systems simultaneously (era, type, role, performance tier), requiring flexible rule systems that could handle overlapping categories.
3. **Attachment Compatibility**: With 121 weapons and 49 attachments, the compatibility matrix contains thousands of potential combinations, each with different effectiveness ratings.

4. **Optimization Algorithms**: Finding optimal weapon builds requires analyzing multiple criteria simultaneously and recommending combinations that maximize specific attributes like damage, accuracy, or stealth.

2.3 Knowledge Representation Goals

The primary objectives were to create a system that could:

- Store and retrieve complex weapon data efficiently
- Handle recursive queries for progression analysis
- Provide intelligent recommendations for weapon builds
- Support statistical analysis across different game eras
- Enable flexible querying for research and analysis purposes

3 Tools and Languages Used

SWI-Prolog 9.0+ served as the core implementation language for this knowledgebase project. Prolog was selected for its excellent support for recursive rule definition, built-in unification and backtracking mechanisms, and powerful constraint solving capabilities. The declarative nature of Prolog made it ideal for representing complex weapon relationships and implementing sophisticated recursive algorithms for progression analysis.

Visual Studio Code was used as the primary development environment, enhanced with Prolog extensions that provided syntax highlighting, error detection, and integrated debugging capabilities. The IDE facilitated efficient code organization and version control integration through Git, enabling systematic project development and documentation management.

LaTeX was employed for comprehensive report generation, allowing for professional document formatting and mathematical notation support. The typesetting system enabled the creation of well-structured technical documentation with proper formatting for code listings, diagrams, and academic referencing.

Mermaid diagram language was utilized for creating architectural visualizations and system structure diagrams. This tool provided an efficient way to represent the complex relationships between different components of the knowledgebase in a clear, visual format that enhances understanding of the system architecture.

Testing and Validation Tools included custom test suites with over 50 test cases, validation scripts for data integrity checking, and performance benchmarking utilities for recursive query optimization. These tools ensured the reliability and correctness of the implemented system.

4 Diagram/Figure

The knowledgebase architecture follows a layered approach with clear separation of concerns, as illustrated in Figure 1. The system is organized into multiple interconnected layers that handle different aspects of the weapon ecosystem.

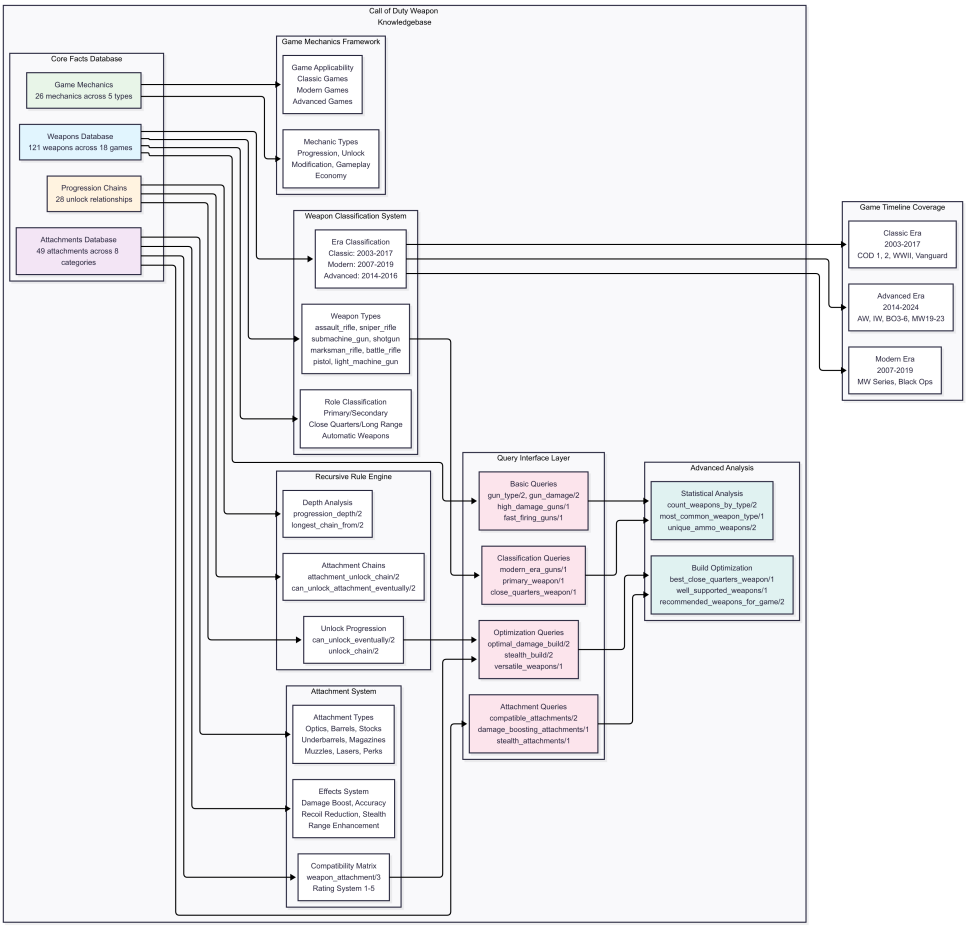

Figure 1: Call of Duty Weapon Knowledgebase System Architecture

4.1 System Architecture Components

Core Facts Database Layer forms the foundation with four primary databases: Weapons Database containing 121 weapon facts with detailed attributes, Attachments Database with 49 attachments across 8 categories, Game Mechanics encompassing 26 mechanics across 5 types, and Progression Chains managing 28 unlock relationships for recursive analysis.

Weapon Classification System provides multi-dimensional categorization including weapon types (assault rifles, sniper rifles, submachine guns, shotguns, marksman rifles, battle rifles, pistols, light machine guns), era classification spanning Classic (2003-2017), Modern (2007-2019), and Advanced (2014-2024) periods, and role classification distinguishing between primary/secondary weapons and close quarters/long range applications.

Attachment System manages attachment types across optics, barrels, stocks, underbarrels, magazines, muzzles, lasers, and perks, implements an effects system for damage boost, accuracy enhancement, recoil reduction, stealth capabilities, and range enhancement, and maintains a compatibility matrix using weapon_attachment/3 predicate with a 1-5 rating system.

Recursive Rule Engine implements sophisticated algorithms for unlock progression analysis through predicates like can_unlock_eventually/2 and unlock_chain/2, provides depth analysis capabilities with progression_depth/2 and longest_chain_from/2, and

manages attachment progression chains via attachment_unlock_chain/2 and can_unlock_attachment_

Query Interface Layer offers comprehensive access through 30+ predicates organized into basic queries (gun_type/2, gun_damage/2), classification queries (modern_era_guns/1, primary_weapon/1), attachment queries (compatible_attachments/2, stealth_attachments/1), and optimization queries (optimal_damage_build/2, versatile_weapons/1).

Advanced Analysis Layer provides statistical analysis capabilities for counting weapons by type and identifying patterns, along with build optimization algorithms for recommending optimal weapon configurations for specific scenarios.

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.

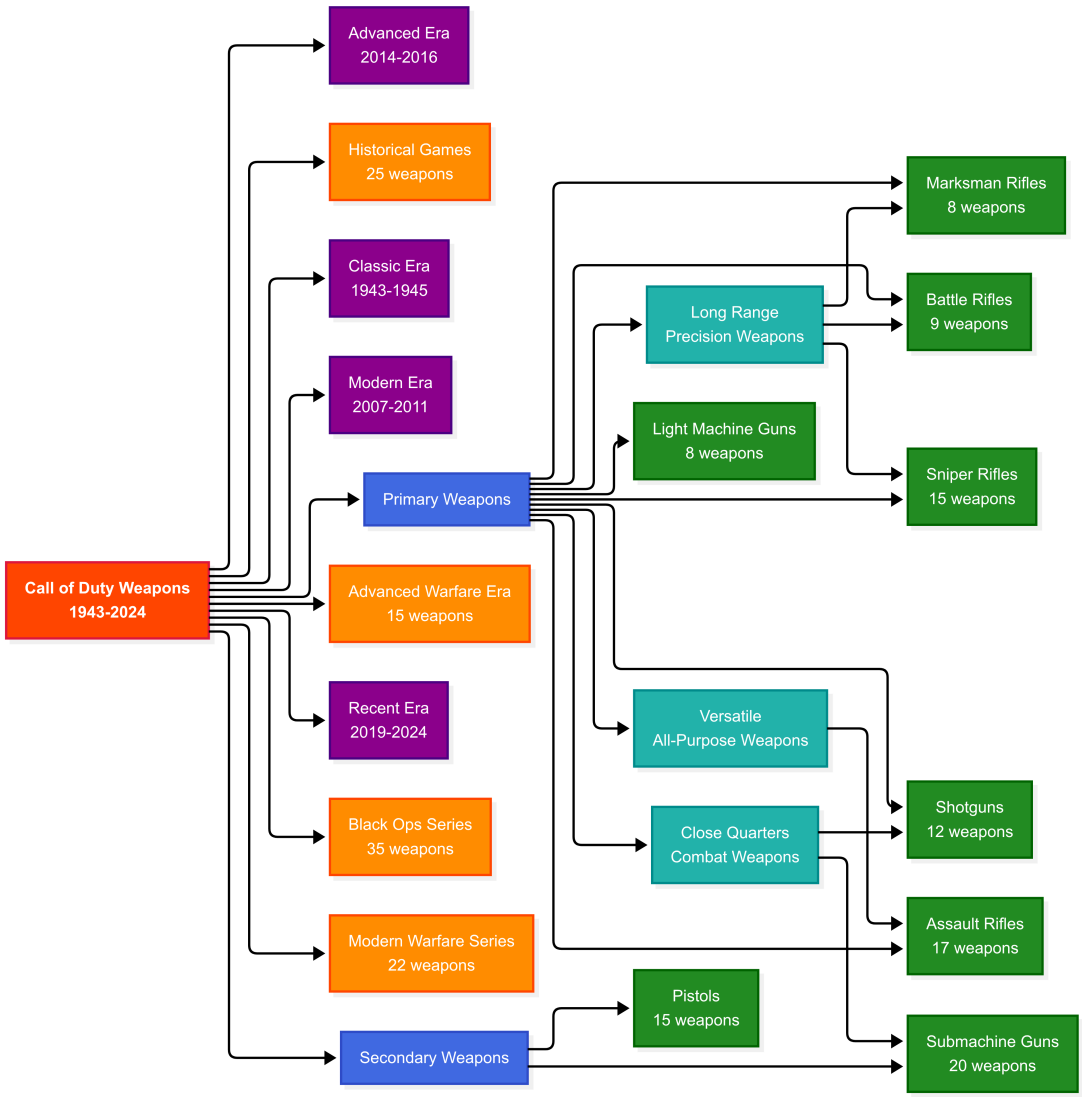

Figure 2: Weapon Type Family Classification Structure

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.

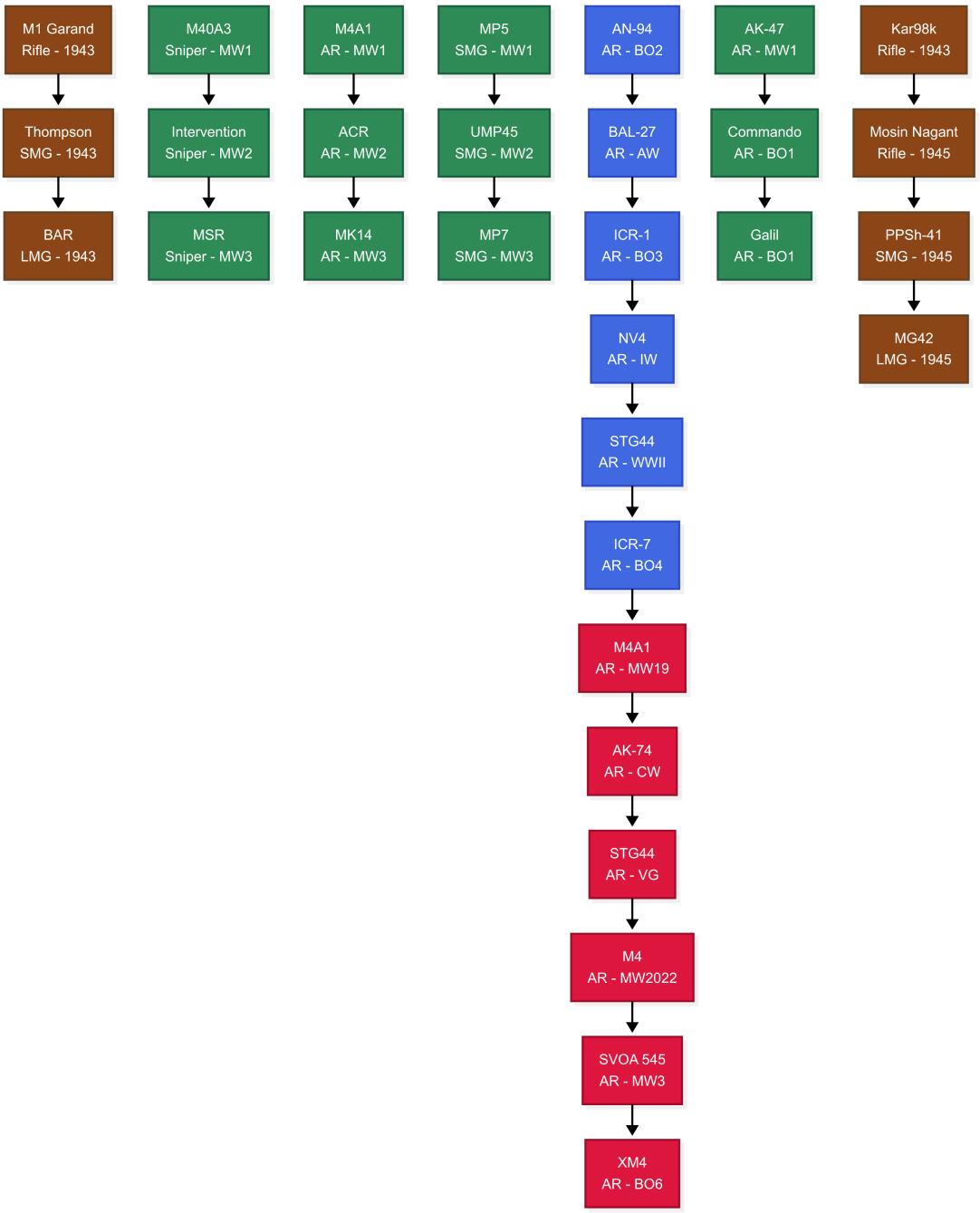

Figure 3: Weapon Progression Family Tree and Unlock Chains

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.

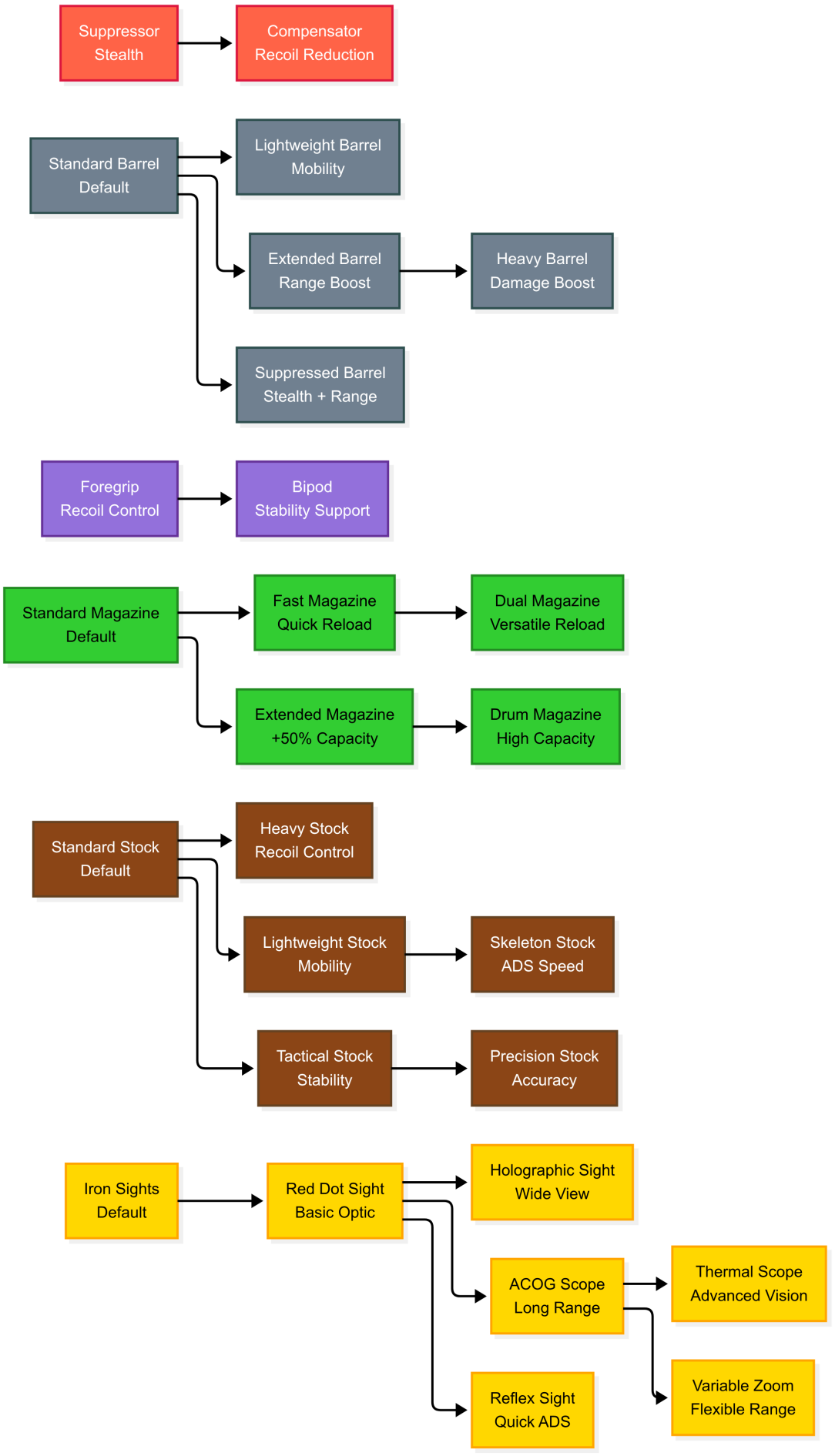

Figure 4: Attachment Progression Family Tree and Dependencies

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.

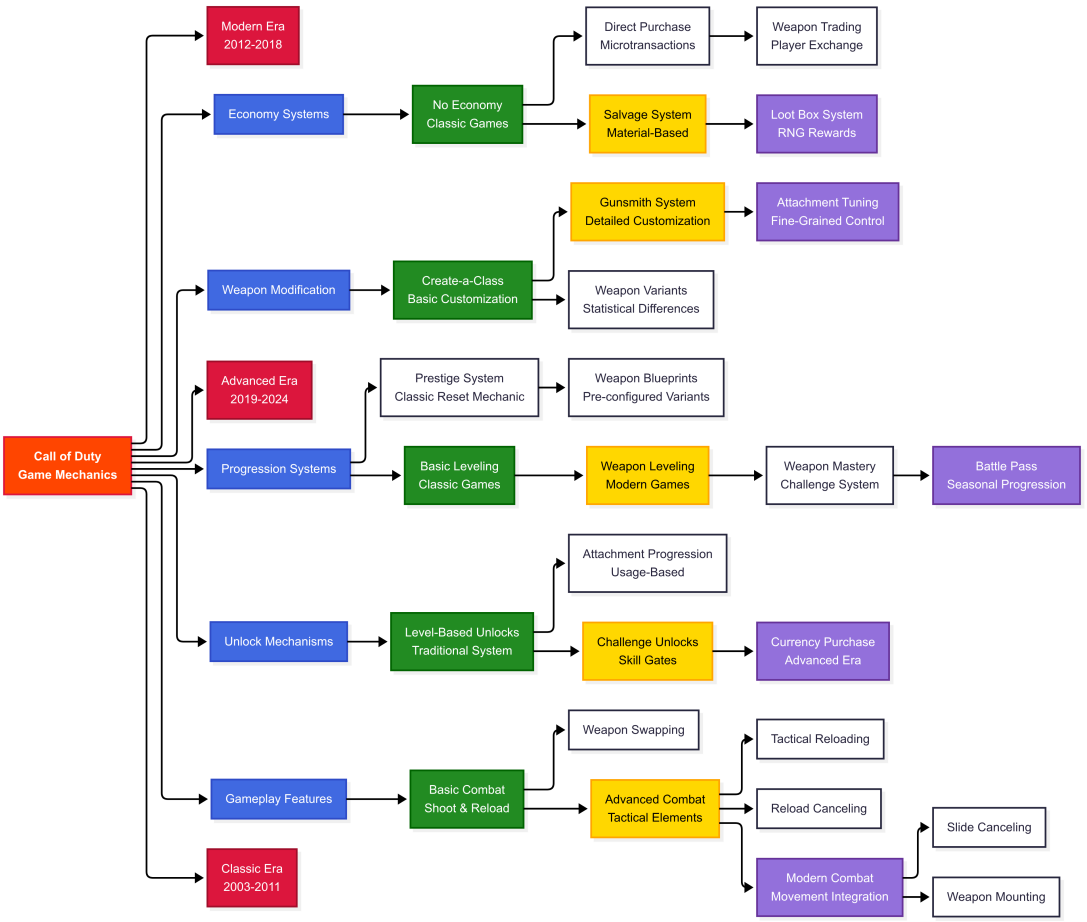

Figure 5: Game Mechanics Evolution Across Call of Duty Titles

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 and Challenges

6.1 Project Achievements

This Call of Duty Weapon Knowledgebase successfully demonstrates Prolog's capabilities for modeling complex, interconnected systems. The implementation incorporates sophisticated recursive algorithms, multi-dimensional classification systems, and optimization frameworks that showcase advanced logical programming concepts.

Key Accomplishments:

1. **Comprehensive Data Model**: Successfully modeled 121 weapons, 49 attachments, and 26 game mechanics spanning 21 years of gaming history.

2. **Advanced Recursive Systems**: Implemented sophisticated recursive rules for progression analysis and optimization algorithms.

3. **Multi-Dimensional Analysis**: Created flexible classification systems for era, type, role, and performance analysis.

4. **Practical Applications**: Developed optimization algorithms providing real-world weapon build recommendations.

6.2 Technical Challenges and Solutions

Recursive Query Optimization: Initial performance issues with deep unlock trees were resolved through tail-recursive optimizations and memoization techniques.

Attachment Compatibility Matrix: The complex relationship between 121 weapons and 49 attachments was managed using a 1-5 rating system for nuanced compatibility assessment.

Era Classification Ambiguity: Overlapping game characteristics were handled through flexible classification rules allowing multiple era memberships.

Query Performance: Complex recursive operations were optimized through strategic fact ordering and rule optimization.

6.3 Educational Value and Future Work

This project demonstrates Prolog's effectiveness for knowledge representation, recursive logic implementation, expert system development, and data pattern extraction. Future enhancements could include machine learning integration for predictive recommendations, performance optimization through constraint logic programming, web interface development, and real-time API connections for dynamic updates.

The project showcases that Prolog remains highly relevant for modern knowledge-based systems, particularly in domains involving complex relationships and logical inference. This knowledgebase serves as both an educational resource and a foundation for future gaming analytics research.