Consolidating a MAPE-K-Based Architecture for Dynamic Difficulty Adjustment: A Bottom-Up Approach

Carlos Henrique R. Souza, Luciana de O. Berretta, Sérgio T. de Carvalho

Instituto de Informática – Universidade Federal de Goiás (INF/UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brasil

carlos_henrique_rorato@discente.ufg.br, {luciana.berretta,sergiocarvalho}@ufg.br

Abstract. *Game developers are increasingly adopting Dynamic Difficulty Adjustment (DDA) mechanisms to tailor gameplay experiences for individual players. However, existing solutions often lack modularity and generalization, limiting their applicability across diverse games, genres, and mechanics. This paper presents the consolidation of a modular DDA architecture based on the MAPE-K loop, designed to support hybrid adaptation strategies. The architecture was developed incrementally by integrating previously evaluated components, including both rule-based and AI-driven mechanisms. While earlier studies validated these components in isolation, we now unify them into a single, cohesive architectural solution.*

1. Introduction

The demand for adaptive gameplay in digital games has grown significantly in recent years [Guo et al. 2024, Seyderhelm and Blackmore 2021]. To fulfill this demand, researchers have proposed a range of techniques under the umbrella of Dynamic Difficulty Adjustment (DDA) [Guo et al. 2024, Mortazavi et al. 2024]. DDA refers to the automatic adaptation of game parameters in response to player performance, to maintain a balance between challenge and ability, thus preventing frustration and abandonment [Guo et al. 2024, Zohaib 2018].

Researchers have explored numerous approaches to implementing DDA mechanisms [Guo et al. 2024, Seyderhelm and Blackmore 2021], which generally fall into two broad categories. The first includes *heuristic-based* techniques (parameter manipulation), which adjust game variables according to the player's performance. The second group comprises *AI-based* methods, employing reinforcement learning, genetic algorithms, or other artificial intelligence techniques to analyze player/game data and guide adaptive behavior.

Despite recent advancements, designing general-purpose DDA mechanisms that work across different genres, mechanics, and gameplay contexts remains a significant challenge [Mi and Gao 2023, Mi and Gao 2022, Sepulveda et al. 2020]. Many existing solutions are tightly coupled to specific games or scenarios, hindering reuse and broader applicability. One promising direction to address this limitation is the integration of multiple DDA strategies [Seyderhelm and Blackmore 2021, Zohaib 2018]. We argue that modular architectures are particularly well-suited to this goal, offering the flexibility required to combine, extend, and tailor adaptation strategies, but a consolidated architectural model addressing this need remains absent in the literature [Mi and Gao 2023, Guo et al. 2024, Souza et al. 2025].

In the context of software development, we can understand digital games with DDA mechanisms as self-adaptive systems [Fredericks et al. 2022], i.e., systems that autonomously monitor internal and external states, analyze contextual data, plan reconfiguration actions, and execute those actions to maintain or optimize performance in dynamic

environments [Weyns 2021]. Within this paradigm, the MAPE-K loop, composed of the Monitor, Analyzer, Planner, Executor, and a shared Knowledge Base, has emerged as a modular architectural model [Krupitzer et al. 2020, Porter et al. 2020, Lam et al. 2022]. Its conceptual clarity and extensibility make it particularly attractive for managing adaptation in complex interactive systems such as games.

In this context, this paper presents the consolidation of a modular architecture for Dynamic Difficulty Adjustment grounded in the MAPE-K loop, an integration not yet presented in previous works. Rather than emerging from a single design effort, this architecture is the result of a bottom-up integration of multiple previous studies and implementations. These include mechanisms for rule-based adaptation (e.g., performance metrics, symptom detection), experiments involving the MAPE-K loop for DDA, as well as AI-based inference using large language models (LLMs) and dynamic player modeling [Gallotta et al. 2024]. Our objective is to provide an architecture for a generalizable DDA mechanism capable of supporting hybrid DDA strategies across diverse games, genres, and mechanics.

2. Architecture and Components

The MAPE-K loop serves as the architectural foundation, organizing the system's components according to their canonical roles and responsibilities. We based the model's development on established design patterns for self-adaptive systems, particularly those proposed by Abuseta and Swesi [Abuseta and Swesi 2015, Krupitzer et al. 2020], which include SAS Monitor, SAS Analyzer, SAS Plan, and SAS Execute. These patterns define the internal logic and communication flow for each segment of the loop. To coordinate component interactions, we implemented the *Observer* pattern, which enables dynamic event propagation throughout the loop [Abuseta and Swesi 2015]. In this configuration, the `Monitor` subscribes to updates from the `Sensors` and notifies the `Analyzer` of relevant changes. The `Analyzer`, in turn, acts as an observable entity for the `Planner`, which then updates the `Executor`. Finally, the `Effectors` observe the `Executor` and apply the required adaptations within the system.

The core MAPE-K loop in the proposed architecture operates through a structured sequence of interactions among its components. The process begins with the acquisition of system data via `ContextElements`, which links the incoming values to predefined `Thresholds` that represent acceptable ranges. Upon capturing the data, the `Sensor` notifies the `Monitor`, which logs the reading in the `SystemStateLog` and checks for any threshold violations. If the system exceeds a threshold, the `Monitor` forwards the corresponding log to the `Analyzer`. The `Analyzer` then evaluates the performance based on the received log and queries the `SymptomRepository` for matching `Symptoms`, each defined by a condition (such as a `Threshold`) and a descriptive text. Upon identifying a match, the system generates an `AdaptationRequest` and notifies the `Planner`. The `Planner`, in turn, consults the `PolicyEngine`, which holds a set of `Rules` that pair symptoms with corresponding `Actions`. When applicable rules exist, the `Planner` formulates a `ChangePlan` containing the necessary actions. Finally, the `Planner` sends the plan to the `Executor`, which then orchestrates the execution of the actions by instructing the relevant `Effectors` to implement the adaptive changes within the system.

Figure 1 illustrates the implementation of the MAPE-K loop in the proposed architecture. The `Sensor` components monitor both the game environment and the player's state (manually extracted from code-level object attributes). On the game side, they collect difficulty-related attributes, while on the player side, they synthesize data into a structured

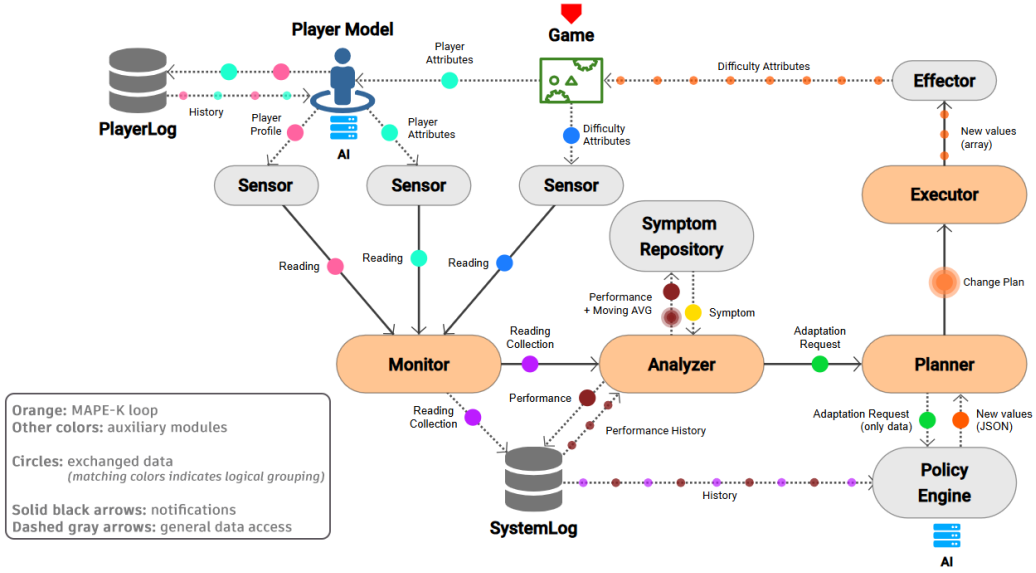

Figure 1. Architecture overview

`PlayerModel`. This component integrates current in-game data with historical records retrieved from the `PlayerLog`. In this context, the `PlayerModel` may employ AI techniques, such as large language models (LLMs), to infer a dynamic player profile based on interaction history. The system sends all readings to the `Monitor`, which logs the data and checks for violations of predefined thresholds. When it detects potential issues, it triggers further analysis to determine whether adaptation is needed.

The `Analyzer` receives the collected readings and computes the player's performance using both the current data and a moving average. It then queries the `SymptomRepository` to identify any symptoms that justify adaptation. When it detects a relevant symptom, it creates an `AdaptationRequest` and sends it to the `Planner`, which delegates the decision-making process to the `PolicyEngine`. Unlike traditional implementations that rely on fixed rule sets, the `PolicyEngine` in this architecture employs AI, such as LLMs, to dynamically generate adaptation responses. This inference considers not only the current `AdaptationRequest` but also the historical sequence of readings retrieved from the `SystemLog`. The AI outputs a structured `JSON` object containing the new parameter values. The `Planner` packages these values into a `ChangePlan` and forwards them to the `Executor`, which processes and dispatches the corresponding actions. Finally, the `Effectors` apply the changes directly to the game by updating its difficulty attributes. This conceptualization aimed to enable the architecture to handle hybrid strategies, fostering reusability, extensibility, and responsiveness.

3. Discussion and Final Remarks

The architecture presented in this paper is the result of a progressive consolidation of multiple isolated developments and experiments carried out throughout previous research. Over time, specific components of the system been implemented and evaluated independently [Souza et al. 2024b, Souza et al. 2025, Souza et al. 2024a]. In this sense, the architectural design points to potential for building DDA mechanisms in games.

As future work, we plan to conduct a systematic evaluation of the architecture as a whole, considering both its functional adequacy and its impact on game variables across different game genres. Through these steps, we aim to contribute to the advancement of engineering practices for adaptive gameplay systems.

References

Abuseta, Y. and Swesi, K. (2015). Design patterns for self adaptive systems engineering. *International Journal of Software Engineering & Applications*, 6(4):11–28.

Fredericks, E. M., DeVries, B., and Moore, J. M. (2022). Towards self-adaptive game logic. In *Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*, pages 24–29, Pennsylvania. ACM.

Gallotta, R., Todd, G., Zammit, M., Earle, S., Liapis, A., Togelius, J., and Yannakakis, G. N. (2024). Large language models and games: A survey and roadmap.

Guo, Z., Thawonmas, R., and Ren, X. (2024). Rethinking dynamic difficulty adjustment for video game design. *Entertainment Computing*, 50:100663.

Krupitzer, C., Temizer, T., Prantl, T., and Raibulet, C. (2020). An overview of design patterns for self-adaptive systems in the context of the internet of things. *IEEE Access*, 8:187384–187399.

Lam, A. N., Haugen, O., and Delsing, J. (2022). Dynamical orchestration and configuration services in industrial IoT systems: An autonomic approach. *IEEE Open Journal of the Industrial Electronics Society*, 3:128–145.

Mi, Q. and Gao, T. (2022). Improved belgian AI algorithm for dynamic management in action role-playing games. *Applied Sciences*, 12(22):11860.

Mi, Q. and Gao, T. (2023). *General Dynamic Difficulty Adjustment System for Major Game Genres*, page 189–200. Springer Nature Switzerland.

Mortazavi, F., Moradi, H., and Vahabie, A.-H. (2024). Dynamic difficulty adjustment approaches in video games: a systematic literature review. *Multimedia Tools and Applications*, 83(35):83227–83274.

Porter, B., Filho, R. R., and Dean, P. (2020). A survey of methodology in self-adaptive systems research. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 168–177.

Sepulveda, G. K., Besoain, F., and Barriga, N. A. (2020). Exploring dynamic difficulty adjustment in videogames. In *2019 IEEE CHILECON*, pages 1 – 6, Chile. IEEE.

Seyderhelm, A. J. A. and Blackmore, K. (2021). Systematic review of dynamic difficulty adaption for serious games: The importance of diverse approaches. *SSRN Electronic Journal*, 1(1):1–45.

Souza, C., Oliveira, S., Berretta, L., and Carvalho, S. (2024a). Large language models and dynamic difficulty adjustment: An integration perspective. In *Proceedings of the 23rd SBGames*, pages 31–36, Porto Alegre, RS, Brasil. SBC.

Souza, C. H. R., de Oliveira, S. S., Berretta, L. O., and Carvalho, S. T. (2025). Extending a MAPE-K loop-based framework for dynamic difficulty adjustment in single-player games. *Entertainment Computing*, 52:100842.

Souza, C. H. R., De Oliveira, S. S., Berretta, L. O., and de Carvalho, S. T. (2024b). DDA-MAPEKit: A framework for dynamic difficulty adjustment based on MAPE-K loop. In *Proceedings of the 22nd SBGames*, SBGames'23, page 1–10, New York, NY, USA. ACM.

Weyns, D. (2021). *An Introduction to Self-Adaptive Systems*. Wiley, USA.

Zohaib, M. (2018). Dynamic difficulty adjustment (DDA) in computer games: A review. *Advances in Human-Computer Interaction*, 2018:1–12.