UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

UNIVERSITY OF MODENA AND REGGIO EMILIA

"Enzo Ferrari" Department of Engineering

Master's Degree in Artificial Intelligence Engineering (LM-32)

Cognitive-Aware Adaptive HMI for Electric Vehicles: A Multi-Agent AI Architecture for Real-Time Context-Sensitive Interaction

Supervisor:
Prof. Simone Calderara

Candidate:
Silvia Giovanardi
Student ID 196455

ACADEMIC YEAR 2024/2025

Contents

List of Figures

List of Tables

List of Code

1. Introduction

Context and Motivation

The rapid evolution of intelligent driver assistance systems has significantly increased complexity and transformed the role of human–machine interaction (HMI) in modern vehicles. Advanced driver assistance systems (ADAS), driver monitoring technologies, environmental sensors and connected services continuously generate information that must be communicated to the driver in an effective and timely manner. While this technological evolution enables higher levels of safety and comfort, their increasing number and sophistication introduce new challenges related to information management and the risk of cognitive overload.

Traditional human–machine interfaces in the automotive sector are largely based on static interaction paradigms, in which information content and presentation strategies are predefined and rarely related to the actual driving context. Such approaches can prove inadequate in complex or dynamic situations, where excessive or untimely information can distract the driver and negatively impact safety. This problem is particularly relevant in electric and hybrid vehicles, where additional system states, energy management information and advanced automation functions further increase interaction requirements.

These considerations motivate the need for adaptive HMI solutions, that can regulate not only what information is presented, but also when and how it is communicated, taking into account both environmental conditions and the driver's cognitive state. A context-sensitive and cognitively informed interaction paradigm represents a fundamental step towards safer and more intuitive human–vehicle cooperation.

The EPIGNOSIS Project

This thesis was developed in the context of the EPIGNOSIS project, an industrial research initiative focused on the development of advanced driver assistance solutions for new-generation hybrid and electric vehicles. More specifically, the work originates from the internship activity

carried out at RE:LAB within the framework of the EPIGNOSIS project, where the proposed adaptive HMI architecture was progressively designed, implemented and analyzed as part of the broader development of AI-based assistance solutions for intelligent vehicles.

The project explores innovative approaches to vehicle safety, performance optimization and user interaction, with a particular focus on open and modular architectures for active safety systems.

Within EPIGNOSIS, particular attention is devoted to the validation of intelligent vehicle functionalities through vehicle-in-the-loop methodologies, which enable realistic testing of the complex interactions between software components, vehicle subsystems and human drivers. In this context, the human–machine interface represents the main channel of communication between the vehicle's intelligence and the user.

The work presented in this thesis contributes to the project by addressing the design and implementation of an adaptive in-vehicle assistance system based on artificial intelligence techniques, aimed at dynamically supporting the driver while preserving safety and usability. The proposed solution seeks to improve driver interaction by adapting interface behavior and assistance strategies according to contextual and cognitive conditions, in line with the project's principles of modularity, flexibility and intelligent system integration.

In particular, the HMI is not treated as a simple passive display layer, but as an active and context-sensitive component of the vehicle's intelligence.

Problem Statement

Despite significant advances in ADAS systems and vehicle-integrated artificial intelligence, current HMI solutions often do not systematically integrate cognitive considerations into their decision-making processes. Many adaptive interfaces rely on limited contextual parameters or predefined rules, without explicitly modeling the driver's cognitive load or accounting for the dynamic nature of real-world driving scenarios.

Furthermore, recent advances in artificial intelligence, particularly large language models, offer powerful capabilities for contextual reasoning and natural language interaction. However, their direct application in automotive environments raises critical challenges related to control,

predictability, explainability and real-time performance. Monolithic conversational systems are often unsuitable for safety-critical contexts, where decisions must be constrained, interpretable and tightly integrated with vehicle logic.

The central problem addressed in this thesis is therefore the design of an HMI architecture capable of leveraging advanced AI techniques while remaining compatible with automotive constraints. This includes the need for modularity, real-time responsiveness, cognitive awareness and secure integration with vehicle systems.

The main challenge is to enable the onboard HMI to dynamically adapt its behavior to the driver's cognitive load, driving context and vehicle status in real time, while ensuring safety, usability and compliance with automotive requirements. This adaptation must occur transparently across multiple dimensions of the driving experience, while maintaining the reliability and predictability required in safety-critical automotive systems.

Several technical challenges arise from this general problem. First, the driver's cognitive load must be estimated by integrating heterogeneous data sources, including driver monitoring systems (DMS), vehicle status information from the vehicle control unit (VCU) and advanced driver assistance systems (ADAS). Second, complex decisions based on multiple criteria must be orchestrated across functional domains such as safety, comfort and interaction management, without relying on rigid rule-based logic that struggles to handle real-world variability. Third, real-time responsiveness must be preserved when integrating large cloud-based language models, while keeping end-to-end latency within acceptable thresholds for the automotive industry. Finally, the architecture must ensure modularity and scalability in the coordination of specialized agents, while preserving maintainability and predictability of behavior.

Original Contributions

This thesis introduces an adaptive and cognitive human–machine interface (HMI) designed around a distributed multi-agent architecture. While traditional adaptive interfaces are based on static rules or limited contextual adaptation, the proposed system breaks down the HMI decision-making process into a series of specialized agents, each of which is responsible for a specific functional domain such as safety, comfort, planning or interaction management. These agents operate

under the coordination of a central orchestrator, enabling separation of responsibilities, scalable, modular and context-sensitive behavior.

Large language models are integrated, not as monolithic conversational agents, but as components of constrained reasoning within selected agents, supporting high-level interpretation and decision-making synthesis while preserving predictability, explainability and compatibility with real-time automotive constraints. The outputs generated by LLMs are structured and transformed into explicit system actions and interface adaptations.

A key contribution of this work is the explicit integration of driver cognitive load estimation into the HMI decision-making process. Based on the estimated cognitive state, the system dynamically adjusts the priority of information, interaction mode and timing. This approach directly addresses the risk of driver distraction and promotes safer and more comfortable driving behavior in complex scenarios.

Finally, the proposed architecture emphasizes transparency and interpretability of the system's decisions. By structuring the reasoning process through modular agents and explicitly linking decisions to contextual and cognitive factors, the system generates understandable explanations for its adaptive behavior, fostering driver confidence in AI-based assistance systems.

The ultimate goal of this thesis is to demonstrate how a cognitive multi-agent HMI architecture that integrates constrained LLM-based reasoning can improve safety, usability and driving comfort in modern vehicles. The proposed approach aims to move beyond static and reactive interfaces towards cooperative and adaptive human–machine interaction tailored to both the driver's needs and real-time driving conditions.

2. State of the Art

2.1 Cognitive Load Theory

Theoretical foundations

Cognitive Load Theory (CLT), introduced by Sweller [48], provides a formal framework for understanding the structural limitations of human working memory during complex task execution. The theory assumes that working memory has a severely limited processing capacity, commonly estimated at approximately 4±1 information chunks, whereas long-term memory stores structured schemas that enable more efficient information processing by reducing cognitive effort through pattern recognition and prior knowledge integration.

In dynamic and safety-critical environments such as driving, these structural limitations become particularly relevant. Vehicle operation requires continuous perceptual monitoring, situation assessment, motor coordination and time-constrained decision-making. Secondary interactions with the Human–Machine Interface (HMI) compete for the same finite cognitive resources required for primary vehicle control. When total demand exceeds available cognitive capacity, performance degradation may occur, potentially compromising safety.

CLT distinguishes three components of cognitive load [50, 40]:

- **Intrinsic Cognitive Load**, determined by task complexity and element interactivity. In driving contexts, intrinsic load varies according to traffic density, environmental conditions, vehicle speed and scenario criticality.

- **Extraneous Cognitive Load**, generated by suboptimal information presentation and interface design. Poor layout organization, redundant notifications, excessive visual clutter or modality-task mismatches increase cognitive burden without improving task effectiveness [49].

- **Germane Cognitive Load**, associated with schema construction and learning processes. In assisted and automated driving systems, this component relates to the driver's understanding

of ADAS behavior, system boundaries and automation logic.

While educational applications aim to optimize germane load to enhance learning, automotive systems prioritize minimizing extraneous load and maintaining intrinsic load within manageable thresholds to preserve stable task performance.

Cognitive Load in Driving Contexts

Cognitive load must be distinguished from related constructs such as mental workload and distraction. Mental workload describes the relationship between task demands and available cognitive resources [53], whereas cognitive load refers specifically to processing demands imposed on working memory [48]. Distraction occurs when attention is diverted from the primary driving task due to visual, manual or cognitive interference [35].

Wickens' Multiple Resource Theory further refines this perspective by suggesting that interference depends on modality overlap. For instance, simultaneous visual demands, such as monitoring traffic while interacting with a touchscreen, produce greater interference than tasks distributed across different perceptual channels [53]. This principle has direct implications for HMI design and multimodal interaction strategies.

Empirical studies consistently demonstrate that elevated cognitive load increases reaction times in a non-linear manner, reduces hazard detection capability and elevates crash risk even in the absence of overt visual distraction [10, 47]. These findings confirm that cognitive load represents a central safety variable in automotive interaction design.

Measurement and Practical Limitations

Since cognitive load is a latent and continuous construct, it cannot be directly observed [40]. Its estimation relies on indirect measurement techniques that can be broadly categorized as subjective or objective.

Subjective instruments such as the NASA Task Load Index (NASA-TLX) [18] and the Rating Scale Mental Effort (RSME) [58] are widely validated and frequently adopted in experimental studies. However, their retrospective and self-reported nature limits their applicability in real-time adaptive systems.

Objective approaches rely on physiological and behavioral proxies. Frequently investigated indicators include:

- pupil dilation, associated with cognitive effort and locus coeruleus activation [2];
- Heart Rate Variability (HRV), particularly reductions in parasympathetic activity under increased mental demand [29];
- blink rate and PERCLOS as indicators of attentional load and fatigue [9];
- vehicle-based metrics such as lane deviation and steering entropy [34].

These signals are inherently noisy and influenced by confounding factors such as stress, lighting conditions and fatigue. Recent research suggests that multimodal fusion combined with machine learning techniques improves robustness compared to single-signal estimation approaches [57].

Nevertheless, most current automotive systems do not integrate continuous cognitive load estimation into coordinated HMI adaptation strategies.

Implications for Automotive HMI

The safety implications of cognitive load have informed the development of automotive standards and regulatory guidelines. ISO 15005 and ISO 15008 define visual behavior requirements and glance duration constraints [22, 23], while the NHTSA Visual-Manual Distraction Guidelines provide quantitative criteria for evaluating infotainment-related distraction [35]. UNECE regulations further address driver monitoring and human–machine interaction in partially automated vehicles.

Despite their relevance, these frameworks typically assume static and population-averaged workload models. They rarely account for real-time inter-individual variability or dynamically adjust interaction strategies based on estimated cognitive state. Consequently, cognitive load management is still predominantly reactive and rule-based, instead of being continuously adaptive and grounded in contextual reasoning.

These structural limitations motivate the investigation of AI-driven adaptive architectures capable of multimodal inference, dynamic workload estimation and coordinated interaction management, forming the conceptual foundation for the approach proposed in this thesis.

2.2 Automotive HMI and Adaptive Interfaces

Human–Machine Interfaces (HMIs) in the automotive domain mediate interaction between the driver and increasingly complex vehicle systems. Unlike conventional human–computer interaction environments, automotive HMIs operate in safety-critical, real-time conditions, where interface design directly affects driving performance, situational awareness and accident risk.

Design Principles in Automotive HMI

Automotive interface design is guided by principles aimed at minimizing distraction while preserving informational sufficiency. Among the most relevant are:

- **Glanceability**, ensuring that visual information can be acquired within short glance durations;
- **Functional prioritization**, structuring content according to safety relevance;
- **Modality appropriateness**, distributing interaction across visual, auditory and haptic channels to reduce perceptual competition.

Empirical findings, including the *NHTSA Visual-Manual Distraction Guidelines* [35] and naturalistic driving studies such as [26], demonstrate that prolonged visual engagement with secondary tasks significantly increases crash risk. In particular, glance durations exceeding two seconds are associated with substantial safety degradation.

Wickens' Multiple Resource Theory [53] further suggests that interference depends on competition within the same perceptual modality. Visual–visual conflicts, such as interacting with a touchscreen while monitoring traffic, generate greater performance degradation than cross-modal interactions. These findings inform modality allocation strategies in modern automotive HMI design.

Safety Constraints and Regulatory Frameworks

Automotive HMIs are subject to stringent regulatory and safety constraints. Standards such as ISO 15005 and ISO 15008 define ergonomic requirements for dialogue management and visual

presentation, while ISO 26262 governs functional safety in electrical and electronic systems. Regulatory guidelines impose limits on interaction time, information density and task complexity.

These frameworks are primarily based on population-level assumptions and conservative thresholds. While effective in reducing risk, they do not incorporate dynamic modeling of individual cognitive states. As vehicle systems become more intelligent and interactive, this static approach reveals structural limitations.

Limitations of Static Interface Architectures

Traditional automotive HMIs rely on predefined layouts, fixed interaction flows and deterministic rule-based logic. Although some context-awareness has been introduced, such as prioritizing safety alerts during critical maneuvers or suppressing notifications at high speed, adaptation typically remains local and subsystem-specific.

Infotainment, navigation, driver assistance and comfort modules often operate independently, optimizing local objectives without global coordination. As a result:

- interaction demands may conflict across subsystems;
- cognitive load is not estimated or regulated at a system-wide level;
- prioritization decisions remain static or narrowly contextual.

This fragmentation becomes increasingly problematic as vehicles integrate advanced driver assistance systems and higher levels of automation. Interfaces must communicate automation status, manage takeover requests and support situational awareness during transitions of control. Static architectures struggle to balance informational sufficiency with cognitive load minimization under these dynamic conditions.

The Role and Limits of Adaptivity in Automotive HMI

Adaptive interfaces offer a promising response to these challenges. In the automotive domain, adaptation may involve:

- modulation of information density based on driving complexity;

- suppression or deferral of non-critical notifications;
- modality switching according to workload conditions;
- personalization of feedback and interaction timing.

However, current commercial implementations typically rely on deterministic rule-based mechanisms and limited contextual triggers. Driver monitoring signals, when available, are often integrated locally rather than through a unified decision layer capable of arbitrating between competing interaction demands.

Moreover, in safety-critical environments, adaptive mechanisms must remain predictable, explainable and certifiable. This creates a tension between flexibility and validation: highly dynamic adaptation increases responsiveness but complicates safety assurance and trust calibration.

Consequently, while adaptivity is increasingly recognized as necessary for managing interaction complexity in modern vehicles, existing systems remain constrained by fragmented architectures and limited cognitive modeling capabilities. Addressing these limitations requires structured approaches capable of integrating multimodal driver monitoring, contextual awareness and coordinated decision-making across heterogeneous subsystems.

2.3 Existing Intelligent Automotive Systems

Recent advances in automotive technology have led to the integration of increasingly intelligent features within in-vehicle Human–Machine Interfaces (HMIs). Commercial systems and research prototypes incorporate adaptive infotainment modules, digital cockpits, conversational assistants and driver monitoring solutions. Despite their technological sophistication, these systems reveal structural limitations in terms of cognitive modeling, systemic coordination and high-level orchestration.

Adaptive Infotainment and Digital Cockpits

Modern digital cockpit platforms integrate infotainment, navigation, connectivity services and Advanced Driver Assistance Systems (ADAS) within unified display environments. Examples

include platforms such as Mercedes-Benz MBUX and BMW iDrive, which provide personalized profiles, predictive suggestions and multimodal interaction capabilities [31, 3].

These systems support user preference learning, adaptive layout configurations and context-triggered recommendations. However, adaptation typically remains limited to personalization and feature-level optimization. Decision logic is often rule-based or derived from usage statistics, without explicit modeling of driver cognitive load or global arbitration across subsystems.

While digital cockpit architectures centralize hardware and software stacks, interaction management remains functionally partitioned. Safety alerts, infotainment content and navigation prompts are frequently generated by independent modules, reducing opportunities for coordinated reasoning.

Conversational In-Vehicle Assistants

Conversational systems have become central components of modern automotive interfaces. Platforms such as MBUX Voice Assistant and other AI-enhanced in-car assistants leverage natural language processing to reduce manual interaction and visual distraction [44].

These assistants improve accessibility and support multimodal interaction. Nevertheless, their primary objective is task execution and dialogue continuity rather than cognitive state regulation. Context-awareness is typically limited to command interpretation and historical usage patterns. Real-time integration of environmental complexity, ADAS state or driver workload into conversational decision-making remains limited.

Furthermore, conversational modules are often implemented as service layers rather than as orchestrators of interaction across heterogeneous subsystems.

Driver Monitoring and In-Cabin Intelligence

Driver Monitoring Systems (DMS) employ computer vision and machine learning to estimate gaze direction, attention level, drowsiness and in some cases emotional state [15]. These technologies are increasingly mandated in regulatory frameworks for higher levels of vehicle automation.

While DMS provide valuable signals for safety warnings and attention management, they are commonly implemented as isolated subsystems. Integration with infotainment adaptation,

notification scheduling or modality allocation is typically limited to threshold-based triggers. Continuous cognitive load estimation rarely informs system-wide interaction prioritization.

As a result, driver state awareness is often decoupled from broader interface orchestration mechanisms.

Optimization-Based and AI-Driven Adaptive Prototypes

Academic research has proposed adaptive HMI frameworks based on multi-objective optimization, workload estimation and contextual modeling [56]. These approaches dynamically regulate information presentation by considering driver, vehicle and environmental variables.

Although such prototypes demonstrate the feasibility of context-driven adaptation, they frequently rely on predefined objective functions or offline calibration procedures. Scalability to complex, heterogeneous automotive ecosystems remains limited. Moreover, many solutions focus on specific interface domains rather than on unified cross-domain arbitration.

Recent research also explores AI-based recommender systems for infotainment personalization and interaction scheduling. However, these systems generally operate at the application level and lack coordination with safety-critical modules.

Structural Limitations of Current Intelligent Systems

Across commercial and research systems, several recurring limitations can be identified:

- **Fragmented adaptation**: Adaptive logic is distributed across independent modules without a centralized reasoning entity.
- **Limited cognitive modeling**: Driver monitoring signals are rarely integrated into continuous cognitive load estimation frameworks.
- **Predominantly rule-based mechanisms**: Many systems rely on deterministic triggers rather than adaptive AI reasoning.
- **Absence of global orchestration**: No unified decision layer coordinates competing interaction demands under dynamic contextual constraints.

- **Explainability and certification challenges**: AI-driven components are typically decoupled from safety-critical decision pathways.

Consequently, existing intelligent automotive systems implement only partial adaptive capabilities, such as personalization, voice interaction or alert modulation, but lack a coherent AI-driven orchestration architecture capable of integrating multimodal driver monitoring, contextual awareness and cross-domain prioritization in real time. This architectural fragmentation motivates the multi-agent adaptive framework proposed in this thesis.

Table 2.1: Comparative analysis of existing intelligent automotive systems

System Category	**Type of Adaptivity**	**Cognitive Modeling**	**Architecture**	**Main Limitations**
Adaptive Infotainment & Digital Cockpits	Preference-based, rule-driven personalization	No explicit real-time cognitive load estimation	Partially centralized platform, functionally modular	Local optimization; lack of cross-domain prioritization; limited safety-aware reasoning
Conversational In-Vehicle Assistants	Intent-based NLP; contextual dialogue management	Limited context awareness; no continuous workload modeling	Service-layer integration within broader HMI stack	Focus on command execution; absence of global arbitration logic
Driver Monitoring Systems (DMS)	Threshold-based alerts; attention detection	Yes (attention/drowsiness), but not integrated into system-wide reasoning	Typically isolated safety subsystem	Weak integration with infotainment and interaction scheduling; reactive rather than proactive
Optimization-Based Adaptive HMI (Research)	Multi-objective optimization; workload-aware scheduling	Scenario-dependent workload modeling	Centralized experimental prototypes	Limited scalability; handcrafted objective functions; lack of real-time AI orchestration
AI-Based Infotainment Recommender Systems	Machine learning personalization	No integration with driving task complexity	Application-level AI modules	Domain-specific adaptation; absence of safety-critical coordination

Table 2.1 summarizes the main characteristics of current intelligent automotive systems and

highlights the absence of unified orchestration mechanisms capable of combining continuous cognitive state estimation with cross-domain interaction management.

Table 2.2: Comparative capability analysis highlighting the architectural and cognitive gap addressed by the proposed multi-agent framework

Capability	**Digital Cockpits**	**Voice Assistants**	**DMS**	**Research A-HMI**	**Proposed Framework**
Central Orchestration Layer	Partial	No	No	Limited	Yes
Continuous Cognitive Load Estimation	No	No	Partial	Scenario-based	Yes
Cross-Domain Arbitration	Limited	No	No	Limited	Yes
Multi-Agent Architecture	No	No	No	No	Yes
LLM-Based Reasoning	No	Emerging	No	No	Yes
Safety-Aware Adaptive Decision Logic	Rule-based	No	Threshold-based	Optimization-based	AI-driven + constrained
Explainability Support	Limited	Limited	Limited	Experimental	Designed-in

The comparative analysis reported in Table 2.2 further emphasizes that no existing solution simultaneously integrates continuous cognitive modeling, multi-domain interaction arbitration and AI-based reasoning within a single architectural framework.

This structural limitation motivates the multi-agent orchestration approach presented in the following chapter.

2.4 Gap Analysis

The review of the state of the art in cognitive modeling and adaptive HMI shows significant progress within each domain. However, when these research areas are considered collectively, a clear architectural fragmentation becomes apparent.

The main limitation of existing approaches is not the lack of enabling technologies, but rather the absence of a unified integration framework that combines cognitive state estimation, adaptive decision-making and safety-aware reasoning.

Fragmented Cognitive Integration

Cognitive Load Theory provides a well-established framework for understanding human performance limitations in complex tasks [48]. Modern driver monitoring systems increasingly rely on physiological and behavioral indicators to estimate workload and attention levels.

Despite these advances, cognitive state estimation in current automotive systems remains weakly integrated into high-level system logic. Workload metrics are typically used for isolated adaptations, such as suppressing notifications or issuing alerts, rather than being embedded as core variables within a global decision architecture. Estimation mechanisms are often subsystem-specific and insufficiently connected to cross-domain prioritization processes.

As a result, cognitive modeling rarely functions as a central organizing principle for adaptive interface behavior.

Limited Adaptation and Orchestration Mechanisms

Existing intelligent automotive systems typically implement rule-based adaptation strategies. While deterministic logic ensures predictability and facilitates validation, it struggles to scale in high-dimensional, context-rich driving environments. Vehicles increasingly integrate driver assistance systems, automation supervision, infotainment and connected services, creating scenarios with competing objectives and dynamic situational variability.

In these conditions, adaptation logic often remains fragmented across independent modules. This fragmentation can lead to inconsistent prioritization, redundant alerts and a lack of globally optimized behavior. The challenge extends beyond modularity: it involves coordinating adaptations under strict safety constraints while maintaining responsiveness and usability.

Underutilization of Contextual and Semantic Reasoning

Current systems increasingly collect heterogeneous contextual signals, from driver state to environmental data, yet the integration of these signals into meaningful, safety-aware decisions remains limited. Advanced semantic reasoning, in the form of pattern recognition, situation assessment or predictive modeling, is often confined to isolated functions such as navigation or infotainment recommendations.

This separation creates a gap between available cognitive and contextual information and its use in real-time, safety-critical decision-making. High-level reasoning is seldom embedded as a core component in adaptive HMI orchestration, limiting the system's ability to anticipate driver needs or proactively manage workload.

Absence of a Safety-Aware Integrative Architecture

The combined analysis of cognitive modeling, adaptation mechanisms and contextual reasoning reveals a broader architectural deficiency: no established framework simultaneously integrates

- real-time cognitive state estimation,
- coordinated adaptation across subsystems,
- context-aware decision-making under safety constraints.

While each of these elements appears independently in the literature, their coordinated integration within a unified, HMI-oriented architecture is largely unexplored. In particular, there is no consolidated model in which cognitive state estimation directly informs adaptive decision logic, while remaining fully compliant with automotive safety standards.

Synthesis and Research Motivation

The identified gap is therefore architectural rather than technological. The central challenge is to reconcile:

- human cognitive limitations with adaptive, context-aware behavior,
- modular system components with coordinated global prioritization,
- high-level reasoning with deterministic, safety-compliant operation.

Addressing this gap requires a structured framework capable of embedding cognitive modeling and context-aware adaptation into a unified, safety-aware HMI architecture. The adaptive HMI architecture proposed in this thesis is designed to fill this gap, providing an integrated approach that bridges cognitive theory, context sensing and adaptive interface orchestration within modern automotive environments.

3. EPIGNOSIS Baseline System Architecture

3.1 Project Overview

General Objectives

The EPIGNOSIS project (acronym from ancient Greek meaning "precise and accurate knowledge") is part of European research into intelligent transport systems and safe mobility. It addresses challenges in Human-Machine Interaction (HMI) within the automotive sector [11].

The project aims to develop advanced technological solutions capable of monitoring both the vehicle environment and the driver, supporting safer and more efficient driving. Specifically, EPIGNOSIS integrates three main components:

- Driver Monitoring Systems (DMS) for assessing the driver's mental and physical state;
- Vehicle-environment perception modules for analyzing external conditions and driving dynamics;
- Human-machine interfaces (HMI) to provide interaction based on operational context.

Project Pillars

A distinctive feature of EPIGNOSIS compared to other projects in the automotive sector is its cognitive approach to interaction: the system is not simply configured as a set of sensors and actuators that respond to isolated stimuli, but rather as an intelligent architecture that constructs and maintains a dynamic and coherent representation of the overall driving situation.

The project is based on three fundamental pillars:

1. Safety: reducing accidents through better driver attention management and timely assistance;
2. Comfort: improving the driving experience through intuitive and non-intrusive interfaces;
3. Efficiency: optimizing energy management, especially for electric and hybrid vehicles [11].

3.2 Work Package 5 – Adaptive HMI

Scope and Role of RE:LAB

The EPIGNOSIS project is structured into several Work Packages, each focusing on specific technological and application aspects.

Work Package 5 (WP5) focuses on the design and implementation of the adaptive Human-Machine Interface (HMI). RE:LAB is responsible for this WP, which provides the baseline HMI system of the project [12].

The main objectives of WP5 are:

- definition of functional and non-functional requirements for the adaptive HMI;
- design of the baseline Adaptivity Engine architecture for context interpretation, task generation and interaction management;
- integration with on-board systems including Driver Monitoring Systems (DMS), Advanced Driver Assistance Systems (ADAS), Vehicle Control Unit (VCU) and context perception modules;
- validation through representative use cases provided by the project [12].

Link with Previous Research

The development of WP5 builds upon the findings of previous studies on Human-Machine Interaction and cognitive load estimation in automotive contexts, as discussed in Chapter 2 [42, 7, 53]. The baseline HMI addresses gaps in real-time monitoring, adaptive task management and multimodal interaction, providing a structured system for evaluating driver support strategies.

3.3 System Requirements

Functional Requirements

The functional requirements define the capabilities that the adaptive HMI system must provide to support driver assistance within the baseline EPIGNOSIS architecture.

First, the system must be able to **acquire heterogeneous data** in a multimodal way, coming from at least five different sources, including Driver Monitoring System (DMS), the ADAS systems, the Vehicle Control Unit (VCU), the Digital Vehicle Environment (DVE) and the Forward Collision Warning (FCW). In addition, the system must be robustly designed so that it can handle any temporary loss of one or more information sources without compromising essential functionalities.

The acquired data must be merged and **interpreted in context**, in order to build a coherent and up-to-date representation of the state of the vehicle, the environment and the driver. This interpretation must take into account not only individual events, but also their temporal and causal dependencies, enabling the system to detect abnormal or potentially critical conditions.

An additional key requirement concerns the **estimation of the driver's cognitive load** in real time, using indicators such as biometric signals, behavioral parameters and driving data. The cognitive load must then be classified into at least three distinct levels, for example low, medium and high, in order to support effective adaptive decision-making [15, 57, 53].

Based on contextual information and cognitive load, the system must **generate interaction tasks**. These tasks may be proactive, anticipating possible needs or critical situations, or reactive, in response to events or actions of the driver. Each task must be described by a set of attributes, including priority, suggested communication modality, information content and time constraints.

Dynamic priority management is a central aspect of the system. Task priorities must be assigned according to safety, urgency and relevance. In particular, priorities must also be dynamically updatable as the operational context evolves.

The system must be able to optimally **select the modes and channels of communication**: it must support at least five different modes, including visual, acoustic, haptic, vocal and combined. The choice of the most appropriate mode must take into account the estimated cognitive load, the

urgency of the message and the environmental conditions [22, 23, 7].

Finally, the system must maintain persistent user profiles in which the driver's preferences and behavioral patterns are stored, allowing adaptation based on implicit feedback and enabling explicit preference modification by the driver [38, 5].

Non-functional Requirements

Non-functional requirements define performance, reliability, safety and architectural constraints of the baseline system:

- **Performance and Real-time Constraints:** system latency must comply with automotive real-time requirements and the graphical interface must operate at a minimum of 30 frames per second.
- **Reliability:** fault-tolerance mechanisms must handle sensor failures without compromising the overall system.
- **Scalability:** the architecture must support the addition of new tasks or output channels with minimal modifications and accommodate platforms with varying computational resources.
- **Modularity:** components must be independently developed, tested and replaceable, with clearly defined and documented interfaces.
- **Functional Safety:** the system must comply with ISO 26262, guaranteeing at least ASIL B for critical functions, and implement monitoring mechanisms such as watchdog timers and event logging [24].
- **Privacy and Data Protection:** the system must comply with GDPR, process biometric data locally, avoid transmitting personal data externally without consent and allow deletion of stored personal data at any time [14].

3.4 Baseline Adaptivity Engine Architecture

3.4.1 Overview

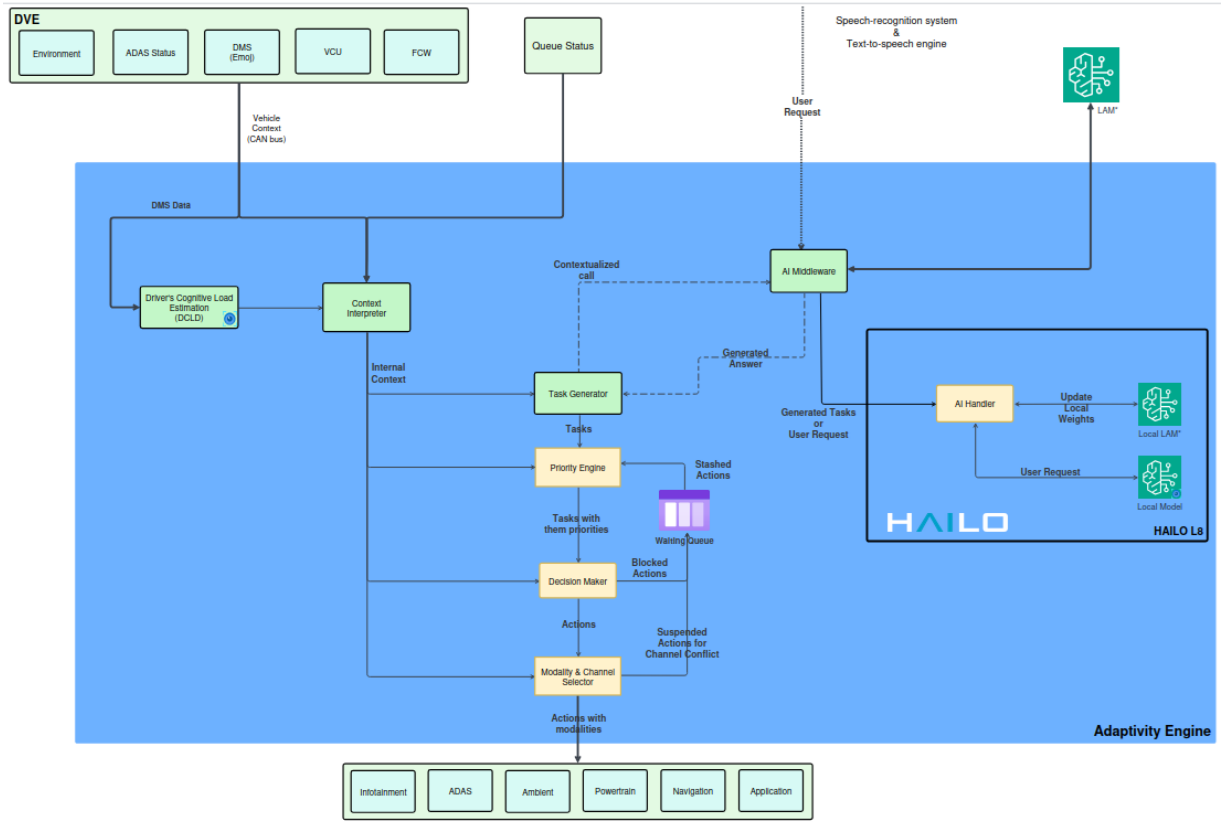

Figure 3.1: Baseline cognitive pipeline defined in EPIGNOSIS D5.2.2 [12].

The Baseline Adaptivity Engine represents the reference architecture defined at project level for the adaptive HMI [12]. It constitutes the logical core responsible for transforming contextual and driver-related data into structured interaction actions delivered through appropriate communication channels.

The architecture follows a cognitive pipeline organized into sequential processing stages, spanning from context acquisition to decision and modality selection. It is designed to ensure:

- **Modularity**: each component operates as an independent module with clearly defined interfaces.

- **Scalability**: additional functionalities can be integrated without altering the overall structure.
- **Real-time performance**: processing latency is compatible with automotive timing constraints.
- **Robustness**: the system tolerates partial subsystem failures without compromising global functionality.

3.4.2 Context Interpreter

The Context Interpreter collects and synchronizes data originating from on-board vehicle subsystems, including:

- Driver Vehicle Environment (DVE)
- Advanced Driver Assistance Systems (ADAS)
- Driver Monitoring System (DMS)
- Vehicle Control Unit (VCU)
- Forward Collision Warning (FCW)

The module performs data fusion and generates a structured *Context State*, representing a coherent snapshot of the driver, vehicle and environmental conditions. The output is provided in a normalized format to downstream modules.

3.4.3 Driver Cognitive Load Estimator

The Driver Cognitive Load Estimator computes a real-time index representing the driver's current cognitive demand.

The estimation is based on:

- Biometric and physiological indicators (e.g., heart rate, skin conductance) when available.
- Behavioral observations: eye movements, posture, facial micro-expression.

- Driving performance metrics: speed variability, steering oscillations, reaction times.

The result is a normalized scalar value used to modulate the quantity, modality and timing of interaction tasks. Threshold-based logic enables classification into predefined cognitive load levels [15, 57, 9].

3.4.4 Task Generator

The Task Generator produces interaction tasks based on the Context State and cognitive load estimation.

Tasks can be categorized as:

- **Reactive**: triggered by external events or driver's actions.
- **Proactive**: generated to anticipate foreseeable situations.
- **Safety-related**: associated with critical driving conditions and require immediate attention.

Each task is characterized by priority, suggested communication mode, content and timing. The combination of these features makes it possible to manage interactions with the user in the best possible way, avoiding overstimulation that could affect cognitive load.

3.4.5 Priority Engine

The Priority Engine assigns and updates priority levels for generated tasks.

Priority assignment is based on:

- Safety relevance
- Urgency
- Contextual severity
- Predefined weighting policies

The engine ensures that critical safety tasks override lower-priority interactions and that task ordering dynamically reflects the evolving driving context.

3.4.6 Decision Maker

The Decision Maker selects the final interaction action to be executed, based on priorities and predefined decision rules.

The selection process considers:

- Available communication alternatives
- Interaction intrusiveness
- System constraints
- Driver profile preferences

Decision rules are deterministic and policy-driven, ensuring compliance with safety requirements and predictable system behavior.

3.4.7 Modality and Channel Selector

This module determines the most appropriate output channel(s) for delivering the selected task.

- **Central display**: for complex information that requires sustained attention
- **Head-Up Display (HUD)**: for critical information to be shown in the driver's field of view
- **Ambient lighting**: for discreet feedback and emotional communication
- **Haptic feedback**: for warnings that require immediate attention
- **Acoustic feedback**: for audible alarms and notifications
- **Voice assistant**: for dialog-based interactions

The selection of the modalities is extended by introducing temporal and cognitive optimization, which takes into account not only the most effective channel, but also the best moment to convey the information, thereby reducing intrusiveness.

3.4.8 Waiting Queue

The Waiting Queue manages the deferral of non-urgent tasks that cannot be executed immediately, releasing them when the context allows.

Tasks are ordered according to:

- Updated priority level
- Temporal validity window
- Context suitability

The queue releases tasks when contextual and cognitive conditions permit safe presentation, preventing information overload and ensuring balanced interaction timing.

3.5 Interaction Flows

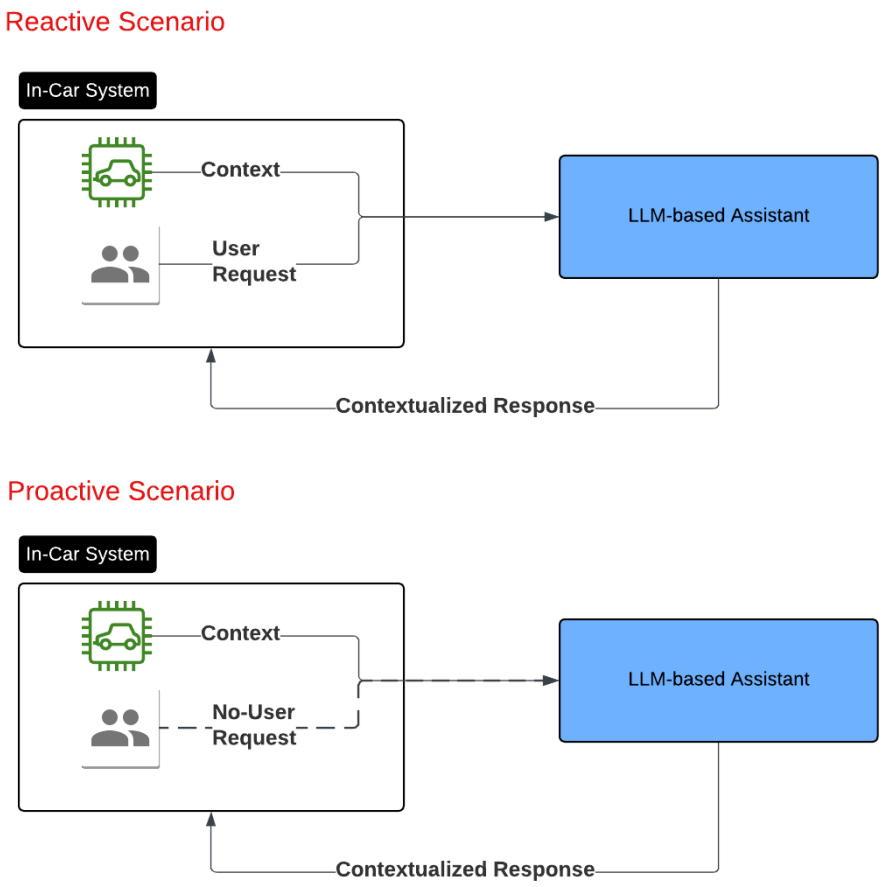

Figure 3.2: Comparison between reactive and proactive interaction flows in the baseline Adaptivity Engine.

The interaction logic of the baseline Adaptivity Engine is structured around two complementary operational flows: a Reactive Flow and a Proactive Flow. These flows define how the system generates, prioritizes and delivers interaction tasks under different driving conditions.

3.5.1 Reactive Flow

The reactive component is triggered when explicit events or risk conditions are detected by the vehicle subsystems, in particular DMS, ADAS and FCW. Such events include, for example, the

activation of collision-risk alerts, lane departure warnings, high level of driver distraction or a confirmed drowsiness condition.

Interaction is also triggered by an explicit driver request. When the user issues a voice command, the system activates to process the request and provide an appropriate response.

However, unlike a simple voice assistant, the system does not merely interpret the request in isolation. Before delivering the response to the user, a set of contextual information is taken into account in order to modulate it appropriately, including vehicle parameters, environmental conditions and driver state.

This information is integrated with the user's request and sent to the LLM agent. This allows the agent to process the request in a holistic way, considering also the context in which it was made. For example, if the user says "I'm hot", the agent might not only lower the cabin temperature but also suggest slightly opening the windows if weather conditions allow it.

Once an event is detected, the following sequence is executed:

1. The Context Interpreter updates the Context State.
2. The Cognitive Load Estimator evaluates the current driver load.
3. The Task Generator creates a reactive task.
4. The Priority Engine assigns urgency.
5. The Decision Maker selects the final action.
6. The Modality and Channel Selector determines the communication channel.

The reactive logic is characterized by short response times, low communicative ambiguity and limited reliance on complex reasoning mechanisms. The primary objective is to ensure driver safety, minimizing additional cognitive load and maintaining full compliance with automotive requirements.

3.5.2 Proactive Flow

The proactive mode represents an innovative aspect of the system. This flow is oriented toward anticipating the driver's needs and optimizing the driving experience over the medium term. It

operates in the absence of immediate critical events and is based on continuous monitoring of contextual and driver-related variables.

In this mode, the onboard system continuously monitors trends in cognitive load, evaluates environmental stability and detects emerging non-critical conditions. If predefined thresholds or contextual patterns are met, the Task Generator produces proactive interaction tasks. These tasks may include information suggestions, comfort-related adjustments and planning or assistance prompts.

The Waiting Queue plays a central role in this flow, ensuring that proactive tasks are delivered only when contextual and cognitive conditions allow safe interaction.

The Proactive Flow is therefore characterized by:

- Lower urgency
- Context-dependent timing
- Adaptive scheduling
- Controlled interaction frequency

3.5.3 Integration of the Two Flows

The Reactive and Proactive Flows do not operate as mutually exclusive modes, but rather as the extremes of an adaptive spectrum. When a critical event occurs, reactive tasks override proactive ones, non-urgent tasks are postponed or suppressed by the system and communication priority shifts toward safety management. Conversely, under stable conditions, the proactive component helps preventing overload situations that could trigger future reactive events.

This dual-flow architecture ensures a balance between safety-critical responsiveness and adaptive interaction management, maintaining controlled cognitive demand and consistent system behavior.

3.6 Integration with Vehicle Systems

The Adaptivity Engine integrates with the main vehicle subsystems through structured data interfaces and real-time data streams. The integration layer abstracts heterogeneous sources (e.g., CAN bus, perception modules, external sensors) into standardized data structures used by the internal processing pipeline.

This approach ensures consistent data representation, decision traceability, modular interfacing between subsystems and temporal synchronization of signals.

3.6.1 Driver Monitoring System (DMS)

The Driver Monitoring System provides information regarding the driver's psychophysical state through camera-based sensing and vision algorithms. The acquired data include:

- Attention level indicators
- Drowsiness detection flags
- Distraction estimation
- Emotional state classification
- Head pose and posture parameters

Data are typically sampled at fixed frequency (e.g., 10 Hz) and timestamped to allow temporal alignment with vehicle dynamics. Within the Adaptivity Engine, DMS outputs contribute to the construction of the Context State and are used by the Cognitive Load Estimator to determine interaction constraints.

3.6.2 Vehicle Control Unit (VCU)

The Vehicle Control Unit provides a comprehensive view of the vehicle's dynamic and operational state via standardized CAN messages. Relevant parameters include:

- **Vehicle dynamics**: speed, acceleration, steering angle

- **Powertrain status**: battery level, energy consumption, motor temperature
- **System status**: brakes, suspension, tires

These signals are used to contextualize driver's state and to limit or adapt interaction strategies during complex or safety-critical maneuvers.

3.6.3 ADAS

Advanced Driver Assistance Systems provide safety-related information and system status indicators.

Monitored ADAS modules include:

- Automatic Emergency Braking System (AEBS)
- Lane Keeping Assist (LKA / ELKS)
- Intelligent Speed Assistance (ISA)
- Attention warning systems

The integration layer extracts activation states, warning levels and intervention flags. These variables are propagated to the Context State and influence task prioritization in safety-critical situations.

3.6.4 Driver-Vehicle-Environment (DVE)

The DVE module aggregates contextual information related to the external driving scene, derived from external sensors, digital maps or contextual services. Considered parameters include:

- Road type and curvature
- Traffic density
- Weather and visibility conditions
- Presence of intersections or pedestrian crossings

This information makes it possible to estimate the complexity of the driving scene and to modulate the behavior of the HMI proactively. For example, in conditions of heavy traffic or reduced visibility, the system can decrease the number of secondary notifications and prioritize minimal interactions.

3.6.5 Forward Collision Warning (FCW)

The Forward Collision Warning subsystem provides critical information related to imminent collision risks. The main variables include the warning activation level, the time-to-collision (when available), the distance from the obstacle and any automatic braking activation status.

FCW events are treated as high-priority inputs within the Adaptivity Engine and impose strict constraints on interaction timing and modality selection.

3.7 Baseline Use Cases

The EPIGNOSIS project defines a structured portfolio of automotive use cases targeting safety enhancement, driver support and operational efficiency. These use cases are formally documented in the official project deliverables and represent the functional baseline of the system architecture and provide the operational scenarios within which the proposed architecture is evaluated [12].

Each use case specifies:

- a well-defined operational scenario;
- a set of input signals derived from vehicle, environmental and driver monitoring sensors;
- a reasoning process based on predefined logic;
- corresponding output actions delivered through the Human–Machine Interface (HMI) or vehicle subsystems.

The baseline use cases demonstrate how contextual information can be leveraged to provide proactive and reactive assistance to the driver under different driving conditions.

3.7.1 Representative Scenarios

The official EPIGNOSIS scenarios can be grouped into several representative domains [12]:

- **Driver Monitoring and Safety Support**: detection of fatigue, stress or reduced attention, with activation of warnings or supportive interventions.
- **Cognitive and Emotional State Awareness**: estimation of driver state using physiological and behavioral indicators to inform assistance strategies.
- **Energy and Efficiency Management**: optimization of battery usage and trip planning through context-aware recommendations.
- **Context-Dependent Notifications**: adaptive presentation of information depending on traffic conditions, route characteristics or mission constraints.
- **Decision Support Scenarios**: aggregation of heterogeneous data sources to assist the driver in making informed choices regarding routing, charging or schedule management.

In the baseline configuration, these scenarios are implemented as distinct functional modules, each designed around specific triggers and operational rules. Interactions are primarily event-driven and structured around clearly identifiable conditions that activate predefined system responses.

The Human–Machine Interface serves as the communication layer through which system outputs, such as alerts, recommendations or informational summaries, are delivered to the driver. The interaction logic is therefore closely tied to the internal decision mechanisms defined for each use case.

Together, these baseline use cases define the official functional scope of the EPIGNOSIS system and establish the architectural reference against which its structural characteristics and limitations can be analyzed in the following sections.

3.8 Architectural Characteristics and Limitations

The baseline EPIGNOSIS architecture reflects a modular and scenario-driven design approach. Each use case is implemented as a functionally bounded component, characterized by clearly

defined inputs, deterministic processing logic and predefined output actions.

This structure ensures traceability and ease of validation within well-scoped operational domains. However, when analyzed from a systemic and interaction-level perspective, several architectural characteristics emerge that may constrain scalability, coherence and long-term adaptability.

Rule-based Decision Logic

Baseline use cases rely predominantly on rule-based decision mechanisms. System behavior is governed by predefined conditional structures that map specific input configurations to corresponding output actions.

This approach offers transparency and straightforward validation, particularly in safety-critical domains. Nevertheless, rule-based logic tends to grow in complexity as the number of contextual variables increases. The addition of new scenarios often requires extending rule sets, potentially leading to combinatorial expansion and reduced maintainability [42, 7].

Moreover, rule-based systems are inherently reactive: they respond to recognized patterns but do not inherently reason about broader situational context beyond encoded conditions [20].

Threshold-based Cognitive Classification

Driver state estimation within the baseline framework is typically structured around threshold-based classification. Continuous physiological or behavioral signals are discretized into categorical states (e.g., normal, medium risk, high risk) according to predefined limits [57, 15].

While this discretization simplifies decision-making and supports clear intervention triggers, it introduces sharp transitions between states. Minor variations around threshold boundaries may produce disproportionate changes in system behavior.

Additionally, this approach does not inherently model gradual cognitive evolution or uncertainty, potentially limiting sensitivity to nuanced changes in driver condition [53, 29].

Static Priority Management

In the baseline configuration, interaction priorities are generally predefined per use case. Each scenario establishes its own hierarchy of alerts, notifications or interventions according to internal logic.

Although effective within isolated contexts, static priority assignment may lead to conflicts when multiple use cases are simultaneously active. Without a dynamic mechanism for arbitration, competing interactions risk overlapping or generating inconsistent communication patterns [42, 54, 17].

This characteristic reflects a design centered on local optimization rather than system-wide interaction governance.

Limited Cross-Use-Case Coordination

Baseline use cases are designed as relatively independent modules. While they may share data sources, their decision processes are largely self-contained.

Such modularity supports development clarity and functional separation. However, limited cross-use-case coordination can reduce narrative continuity across interactions. Sequential or overlapping scenarios may generate responses that are coherent locally but not necessarily aligned at the system level.

As the number of supported scenarios increases, the absence of a unified coordination layer may affect consistency and overall interaction stability.

Lack of Predictive Modeling

The baseline architecture primarily operates in an event-driven and reactive manner. Interventions are triggered once predefined conditions are satisfied.

Although this ensures timely responses to detected states, it does not inherently incorporate forward-looking evaluation of possible future trajectories. Anticipatory adaptation, based on projected evolution of driver state, environmental complexity or mission constraints, is not a structural component of the baseline framework.

Consequently, the system behavior is mainly shaped by current-state recognition rather than predictive reasoning about future developments.

3.9 Transition Towards an Intelligent Orchestrated Architecture

The architectural characteristics described above do not represent design flaws, but rather reflect the constraints of a modular, rule-based and scenario-driven framework. Such an approach ensures reliability and clarity within bounded operational domains. However, as interaction complexity increases and use cases expand, the need for a more integrated decision structure becomes evident.

Specifically, the coexistence of multiple heterogeneous scenarios, dynamic driver states and safety-critical constraints calls for mechanisms capable of coordinating decisions beyond isolated functional modules. A system-level perspective becomes necessary to manage interaction timing, priority arbitration and contextual coherence across the entire driver–vehicle–environment ecosystem.

Furthermore, supporting explainable and context-sensitive decision processes requires reasoning capabilities that can synthesize distributed information sources while preserving transparency and safety constraints. This motivates the exploration of an architectural evolution toward a distributed and orchestrated reasoning framework.

Chapter 4 introduces a multi-agent architecture designed to address these structural requirements, enabling coordinated decision processes, distributed reasoning and interaction governance at the system level.

4. Intelligent Orchestrated Multi-Agent Architecture

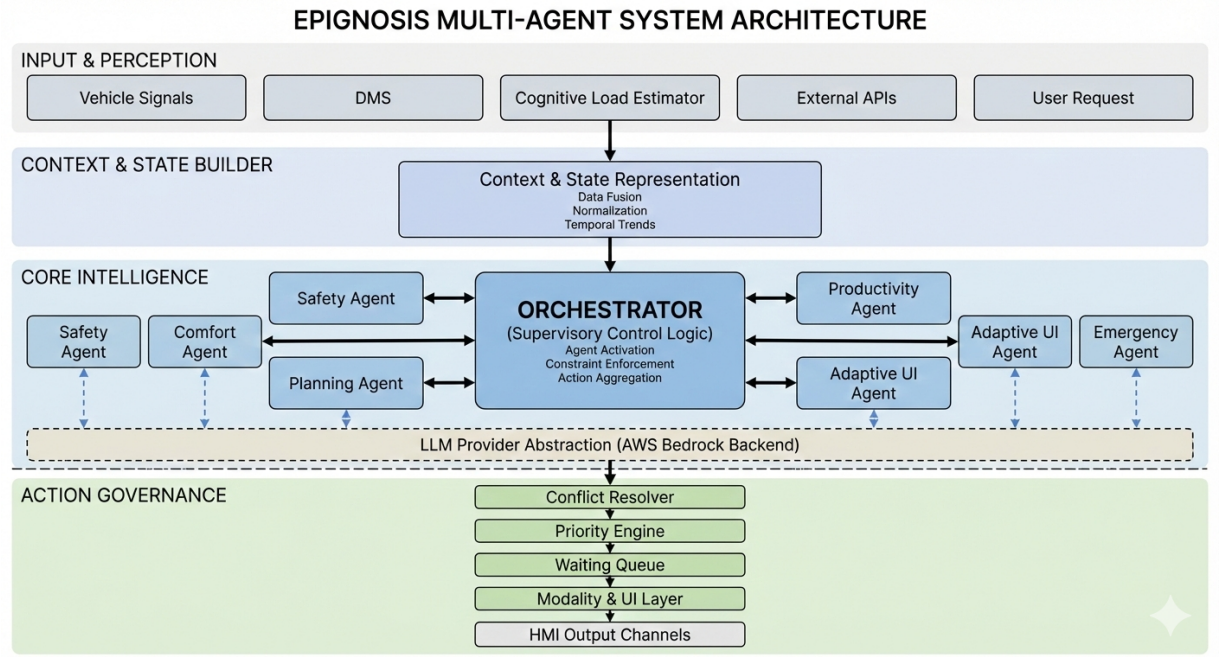

Figure 4.1: Orchestrator-Centric Multi-Agent Architecture for Cognitive-Aware Adaptive HMI

Unlike the baseline pipeline architecture, the proposed system adopts an orchestrator-centric multi-agent structure. Intelligence is decomposed into context-aware specialized agents that are dynamically activated and coordinated by a supervisory orchestrator. This structure allows for modular, scalable and context-sensitive decision-making, moving beyond the rigid, sequential logic of traditional adaptive systems.

Figure 4.1 illustrates the proposed orchestrator-centric multi-agent architecture, highlighting how cognitive awareness, adaptive decision logic and AI-driven modules (including LLM-based reasoning) are integrated into the orchestration layer, in contrast with the simpler baseline depicted in D5.1 [11].

4.1 From Reactive Adaptation to Intelligent Orchestration

Traditional driver assistance and adaptive vehicle systems operate primarily on reactive principles: they respond to sensor inputs and predefined thresholds without anticipating future states or coordinating across multiple subsystems [42, 7, 17]. While this is effective for basic safety and comfort functions, such reactive adaptation is inherently limited in handling complex, dynamic driving scenarios that require anticipatory decision-making, context-aware responses and seamless interaction among multiple vehicle functions.

The proposed Intelligent Orchestrated Multi-Agent Architecture addresses these limitations by introducing a cognitive orchestration layer capable of:

1. Proactive Decision-Making: Agents continuously analyze the Driver-Vehicle-Environment (DVE) state and may generate anticipatory assistance actions or recommendations, allowing pre-emptive interventions before critical situations emerge.

2. Dynamic Coordination: Multiple agents communicate and negotiate priorities, ensuring consistent and optimal system-level behavior.

3. Contextual Awareness: Semantic interpretation of sensor and subsystem data allows the architecture to adapt its strategies to changing environmental and driver conditions.

4. Learning and Adaptivity: Continuous evaluation of outcomes enables agents to refine their decision models over time, improving performance in recurring or evolving scenarios.

This shift from reactive adaptation to intelligent orchestration enables a vehicle to move beyond isolated subsystem responses toward a harmonized, system-wide cognitive behavior, laying the foundation for advanced autonomous support and highly personalized driver assistance.

While rule-based architectures are effective for well-defined scenarios, the growing number of contextual variables and interaction tasks in adaptive HMI systems can lead to combinatorial growth in rule complexity. As the number of possible contextual states increases, maintaining and validating large rule sets becomes progressively more difficult, potentially affecting scalability and long-term maintainability [54, 7].

The proposed multi-agent architecture addresses this limitation by introducing a coordinated reasoning layer capable of synthesizing candidate actions from heterogeneous contextual inputs while preserving deterministic governance through validation, prioritization and safety constraints.

4.2 Architectural Design Principles

The Intelligent Orchestrated Multi-Agent Architecture is grounded on a set of foundational design principles derived from the specific constraints of automotive HMI, where safety, cognitive ergonomics, real-time responsiveness and system predictability are mandatory requirements.

Rather than optimizing a single performance objective, the architecture balances adaptability, robustness, explainability and scalability. These principles collectively define how intelligence is distributed, coordinated, constrained and validated within the system, providing the conceptual framework that informs all architectural and implementative choices presented in this chapter.

Cognitive Awareness as a Primary Design Constraint

Cognitive load is treated as a primary architectural constraint rather than a secondary optimization metric. All adaptive behaviors, interaction flows and decision policies are explicitly conditioned on the estimated cognitive state of the driver within the DVE model.

By embedding cognitive awareness at the architectural level, the system avoids static interaction strategies and enables context-sensitive adaptation aligned with human cognitive limitations [48, 53, 57, 42]. In high workload conditions, the architecture prioritizes simplification, postponement of non-critical interactions and reduction of information density. Conversely, under low workload conditions, richer interaction, proactive assistance and advisory functions can be safely enabled.

This principle ensures that adaptivity enhances situational awareness without generating cognitive overload.

Distributed Intelligence Through Specialized Agents

The architecture adopts a distributed intelligence paradigm based on domain-specialized agents. Each agent is responsible for a clearly defined functional scope (e.g., safety monitoring, comfort

management, planning support, interaction generation), promoting separation of concerns and reducing systemic complexity [54, 17].

Agents operate autonomously within their domain while remaining context-aware. Their activation is conditional and situation-dependent, allowing the system to scale its reasoning effort according to contextual demands. This prevents unnecessary computation and reduces the risk of redundant or conflicting system behaviors.

Large Language Models (LLMs), when employed, are embedded within these agents as domain-specific reasoning components. Within each agent, LLMs support contextual interpretation, decision synthesis and the generation of candidate actions or assistance suggestions based on the current DVE state.

Rather than acting as centralized decision-makers, they operate as reasoning modules inside specialized agents.

Orchestrated Decision-Making and Global Consistency

Although intelligence is distributed across multiple agents, global coherence is enforced through a supervisory orchestrator. The orchestrator does not perform semantic reasoning directly; instead, it manages agent activation, priority assignment, conflict resolution and enforcement of system-level constraints.

This separation ensures a controlled balance between local agent autonomy and global system consistency [54, 20]. Agents may propose actions or recommendations, but final decisions are validated and prioritized at the orchestration level.

Such proposals may originate either from explicit user requests (reactive interaction) or from proactive context monitoring, where agents autonomously analyze driver state, vehicle telemetry and environmental signals to generate assistance suggestions. This guarantees alignment with safety requirements, cognitive feasibility and interaction policies.

The orchestrator therefore acts as a governance layer, preserving systemic stability while enabling adaptive flexibility.

Constraint-Driven and Controlled LLM Reasoning

LLM-based reasoning is explicitly constrained through structured prompting strategies, formal output schemas, deterministic validation layers and rule-based post-processing. Language models are employed as contextual reasoning engines capable of synthesizing contextual information and generating candidate actions, recommendations, or interaction strategies based on the current system state.

These candidate actions are not executed autonomously by the language model itself. Instead, they are converted into structured internal representations and passed through deterministic governance layers including validation, priority management, conflict resolution and rule-based filtering.

Only actions that satisfy system constraints and orchestration policies can be executed or presented to the driver.

Such constraint-driven integration is essential in automotive contexts, where reliability, fail-safe behavior and certification compatibility are critical requirements [24].

Real-Time Compatibility and Non-Blocking Execution

All architectural components are designed to comply with real-time constraints typical of in-vehicle systems. The architecture adopts asynchronous and non-blocking execution models to prevent AI-driven reasoning processes, external API interactions, or inter-agent communication from interfering with safety-critical operations.

When latency requirements cannot be satisfied, the system degrades gracefully by postponing non-essential tasks or reverting to deterministic fallback strategies. This guarantees that adaptive intelligence enhances the driving experience without compromising responsiveness or safety [24].

Progressive Disclosure and Interaction Minimalism

Interaction strategies follow the principle of progressive disclosure, whereby information and controls are presented incrementally based on urgency, contextual relevance and cognitive availability.

In demanding driving situations, the system limits interaction to essential safety-related content. As cognitive load decreases, additional features, recommendations or explanatory content may be introduced. This principle directly informs adaptive interface generation, modality selection, timing policies and content prioritization [22, 23, 53, 35].

The objective is to minimize unnecessary interaction while preserving transparency and usability.

Explainability and Traceability by Design

Explainability is treated as an architectural requirement rather than a post-hoc feature. Each adaptive decision is associated with structured metadata capturing contextual inputs, agent-level reasoning, priority justifications and orchestration outcomes [36, 19].

This traceability supports debugging, validation, post-event analysis and future certification processes. It also enables transparent reconstruction of how cognitive state, environmental context and system priorities influenced interaction decisions.

Embedding explainability at design time ensures accountability and long-term maintainability of AI-driven adaptive systems.

Scalability and Extensibility

The modular multi-agent structure enables incremental extension through the addition of new agents, reasoning strategies, interaction modalities or sensing capabilities without requiring architectural redesign.

Clear interface definitions and orchestration-based coordination allow the system to evolve while preserving structural integrity and operational guarantees. This scalability supports experimentation with emerging AI techniques and adaptation to different vehicle platforms or deployment contexts.

As a result, the architecture is designed not only for current functionality but also for long-term evolution.

4.3 System Overview and High-Level Architecture

This section presents a structured overview of the proposed Intelligent Orchestrated Multi-Agent Architecture, describing its main functional blocks, the separation between data and control flows and the overall cognitive-adaptive processing pipeline.

Rather than focusing on individual components in isolation, this overview clarifies how contextual information, cognitive state estimation, agent-level reasoning and orchestration mechanisms interact to produce coherent and safety-aligned adaptive behavior.

4.3.1 Global Functional Blocks

At a high level, the architecture is organized into five macro-layers:

1. **Context Acquisition and Interpretation Layer**
2. **Cognitive Modeling Layer**
3. **Multi-Agent Reasoning Layer**
4. **Orchestration and Governance Layer**
5. **Interaction and Execution Layer**

The *Context Acquisition and Interpretation Layer* aggregates heterogeneous inputs from vehicle subsystems, environmental sensors, driver monitoring systems and external data sources. These signals are normalized and semantically structured by the **Context Interpreter**, producing a unified contextual representation.

The *Cognitive Modeling Layer* includes the **Cognitive Load Estimator**, which evaluates the driver's mental workload. Its output is not treated as advisory information but as a structural constraint influencing all downstream decisions.

The *Multi-Agent Reasoning Layer* contains the specialized agents (Safety, Comfort, Planning, Productivity, Adaptive UI, etc.). Each agent evaluates the contextual state from its domain perspective and may propose candidate actions.

The *Orchestration and Governance Layer* constitutes the core supervisory mechanism. It includes:

- the Orchestrator,
- the Request Splitter,
- the Priority Engine,
- the Assistant Evaluation module,
- the Conflict Resolver,
- the Waiting Queue.

This layer coordinates task decomposition, action ranking, supervisory evaluation, conflict filtering and deferred execution, thereby guaranteeing compliance with safety and cognitive constraints while preserving global consistency.

Finally, the *Interaction and Execution Layer* translates validated decisions into UI updates, voice responses, ambient adjustments or other vehicle-level adaptations.

4.3.2 Data Flow and Control Flow Separation

A fundamental architectural principle is the strict separation between data flow and control flow.

Data flow refers to the propagation of contextual information, cognitive state estimates and agent-generated proposals. It follows a predominantly bottom-up structure:

- Sensors and subsystems generate raw signals.
- The Context Interpreter structures and normalizes them.
- The Cognitive Load Estimator produces workload constraints.
- Specialized agents process this structured state and generate candidate actions.

Control flow, on the other hand, follows a top-down structure and has a supervisory role. It is governed by the Orchestrator and Governance modules:

- The Orchestrator decides which agents are activated and when task decomposition is required.
- The Request Splitter decomposes complex requests into structured sub-tasks.
- The Priority Engine ranks proposed actions.
- The Assistant Evaluation module validates candidate actions before execution.
- The Conflict Resolver ensures mutual compatibility among approved actions.
- The Waiting Queue defers contextually valid but temporally inappropriate actions.

This separation prevents probabilistic reasoning components (e.g., LLM-based agents) from directly influencing execution pathways without passing through deterministic decomposition, prioritization, evaluation and conflict-management layers.

4.3.3 Cognitive–Adaptive Processing Pipeline

The overall cognitive–adaptive pipeline can be conceptualized as a constrained reasoning loop composed of the following stages:

1. **Context Fusion**: heterogeneous signals are fused into a normalized situational model.
2. **Cognitive State Estimation**: driver workload is estimated and transformed into operational constraints.
3. **Agent-Level Reasoning**: specialized agents generate structured action proposals based on contextual relevance.
4. **Request Decomposition and Prioritization**: complex requests are decomposed, when necessary, and candidate actions are ranked according to urgency, safety relevance and contextual utility.
5. **Assistant Evaluation**: candidate actions are validated with respect to safety, appropriateness, clarity and actionability.

6. **Conflict Resolution**: approved actions are filtered to remove incompatible or redundant proposals.

7. **Execution or Deferral**: actions are either executed through adaptive UI and vehicle interfaces or deferred via the Waiting Queue.

Crucially, cognitive load operates as a gating mechanism throughout the pipeline. It influences agent activation thresholds, priority assignment, progressive disclosure strategies and deferral decisions.

This transforms the architecture from a reactive stimulus–response system into a cognitively regulated adaptive framework, where every interaction is evaluated not only for contextual correctness but also for cognitive feasibility.

The subsequent sections detail each functional block of this architecture, beginning with the Orchestrator as the core supervisory logic and proceeding through context modeling, agent execution, governance mechanisms and LLM abstraction.

4.4 Context Modeling and Cognitive State Estimation

The effectiveness of an orchestrated multi-agent adaptive HMI critically depends on the system's ability to accurately interpret the current operational context and estimate the cognitive state of the driver. Rather than treating context as a collection of raw signals, the proposed architecture constructs a structured, semantically normalized representation of the Driver–Vehicle–Environment state.

Context modeling and cognitive estimation therefore constitute the foundational layer of the architecture: downstream reasoning, agent activation and decision governance mechanisms operate on this structured representation and are constrained by it.

4.4.1 Context Interpreter

The Context Interpreter is responsible for transforming heterogeneous input signals into a unified situational model. Inputs may include:

- vehicle dynamics data (speed, acceleration, steering angle),

- ADAS states and warnings,
- driver monitoring indicators (gaze direction, blink rate, posture),
- infotainment and interaction events,
- environmental data (traffic density, weather, road type),
- external services (navigation, calendar, traffic APIs).

Raw signals are first normalized into structured features and then mapped into higher-level semantic descriptors (e.g., *urban dense traffic*, *high-speed highway cruising*, *complex intersection*, *driver distraction detected*).

This abstraction layer prevents downstream agents from operating on fragmented or sensor-specific data. Instead, agents consume a standardized context object that encapsulates:

- situational complexity,
- environmental criticality,
- driver attention indicators,
- active vehicle subsystems,
- ongoing interaction states.

The Context Interpreter therefore acts as a semantic fusion engine, ensuring coherence and reducing ambiguity before any decision-making process is triggered.

4.4.2 Cognitive Load Estimation as a Hard Constraint

Cognitive load estimation is not treated as advisory metadata but as a structural constraint embedded in the decision loop [48, 53, 57, 15].

The Cognitive Load Estimator evaluates the driver's mental workload using a combination of behavioral indicators, interaction patterns, vehicle dynamics and contextual complexity. The resulting workload score is mapped to discrete operational bands (e.g., low, medium, high, critical).

Crucially, this output functions as a gating variable across the architecture:

- it conditions agent activation thresholds,
- it directly influences the Priority Engine,
- it determines whether actions are executed immediately or deferred,
- it constrains UI density and interaction richness.

In high-load conditions, non-essential proposals are automatically suppressed or transferred to the Waiting Queue. In low-load states, proactive assistance and richer interaction modalities may be enabled.

By embedding cognitive load as a hard architectural constraint, the system ensures that adaptivity remains aligned with human cognitive capacity rather than purely contextual opportunity.

4.4.3 Predictive Modeling Integration

Beyond reactive estimation, the architecture integrates predictive modeling to anticipate short-term cognitive and contextual evolution.

A recurrent neural architecture (LSTM-based model) is employed to capture temporal dependencies in driver behavior and environmental dynamics. Unlike static threshold-based approaches, the predictive component estimates near-future workload trends and situational escalation probabilities [25].

This enables:

- proactive suppression of non-critical interactions before workload peaks,
- early prioritization of safety-relevant information,
- smoother transitions between interaction modes.

The predictive output does not override deterministic constraints; instead, it refines them. Forecasted workload increases may temporarily tighten interaction policies even before critical thresholds are reached.

This anticipatory capability represents a significant advancement over purely reactive HMI adaptation strategies.

4.4.4 Interaction Bandwidth Modeling

Interaction Bandwidth Modeling translates cognitive load and contextual complexity into an operational estimate of available interaction capacity.

Rather than assuming binary availability (interruptible vs non-interruptible), the architecture models interaction as a limited bandwidth resource that fluctuates over time. Available bandwidth is derived from:

- estimated cognitive load,
- driving task criticality,
- environmental uncertainty,
- ongoing system interaction.

This bandwidth parameter directly influences:

- the number of concurrent UI elements displayed,
- modality selection (visual, auditory, haptic),
- verbosity of voice responses,
- frequency of proactive suggestions.

By formalizing interaction capacity as a bounded resource, the architecture moves from simple threshold-based filtering to continuous regulation of interaction density [53, 22, 23].

Together, the Context Interpreter, Cognitive Load Estimator, predictive modeling layer and interaction bandwidth model form a tightly integrated cognitive modeling subsystem. This subsystem provides the structured and constrained foundation upon which the orchestrator and specialized agents operate in the subsequent layers of the architecture.

4.5 Orchestrator-Centric Multi-Agent Execution

Architectural Principles of Centralized Orchestration

The proposed architecture adopts a centralized orchestration paradigm in which a supervisory entity, referred to as the *Orchestrator*, governs the execution of a distributed set of specialized agents. Unlike fully decentralized multi-agent systems based on peer negotiation, this approach ensures deterministic behavior, priority coherence and safety compliance within the highly constrained automotive domain [54, 17].

The Orchestrator operates as the system-level decision authority. It receives structured inputs from the Context Modeling layer (Chapter 4.4), including environmental conditions, vehicle state, driver profile, predicted events and real-time cognitive load estimation. Based on this information, it evaluates agent proposals, enforces constraints and determines the final system action.

Formally, the orchestration process can be abstracted as:

$$A^* = \mathcal{O}(C, L_c, P, \mathcal{A})$$

where:

- C represents the contextual state vector,
- L_c denotes the estimated cognitive load,
- P is the priority policy set,
- $\mathcal{A}$ is the set of candidate agent actions,
- A^* is the selected system action.

This formulation highlights that agent autonomy is bounded by system-level constraints and that the final decision authority remains centralized.

Supervisory Role and Decision Authority

The Orchestrator fulfills three primary supervisory functions:

1. **Constraint Enforcement**: ensuring compliance with safety, regulatory and cognitive constraints.
2. **Priority Arbitration**: resolving conflicts among competing agent proposals.
3. **Global Consistency Preservation**: maintaining coherence across multimodal adaptations (visual, auditory, haptic, environmental).

Safety-related agents are granted structural priority. No comfort or productivity optimization may override safety-critical interventions. Cognitive load operates as a hard constraint: if the estimated load exceeds a predefined threshold, non-essential adaptations are suppressed or postponed.

Agent Autonomy and Constraint Boundaries

Each agent operates autonomously within its functional domain, generating adaptation proposals based on localized objectives. However, autonomy is bounded by:

- Cognitive load constraints
- Safety precedence rules
- Interaction bandwidth limitations
- System-level optimization policies

Agents do not directly execute actions; instead, they submit structured proposals to the Orchestrator. This ensures explainability, traceability and deterministic arbitration.

Reactive and Proactive Task Routing

The orchestration mechanism supports both reactive and proactive execution modes:

- **Reactive routing**: triggered by real-time events (e.g., sudden traffic conditions, driver distraction spikes).

- **Proactive routing**: triggered by predictive modeling outputs (e.g., anticipated congestion, upcoming complex intersections).

Proactive execution leverages predictive inputs from the modeling layer, enabling anticipatory adaptation while respecting cognitive constraints.

Conflict Resolution and Priority Arbitration

Conflicts may arise when multiple agents propose incompatible actions. The arbitration mechanism follows a hierarchical decision logic:

1. Safety-critical actions override all others.
2. Cognitive overload suppression is applied when thresholds are exceeded.
3. Planning and navigation-related tasks take precedence over experiential enhancements.
4. Comfort and ambient optimizations are executed only when bandwidth allows.

This structured arbitration ensures predictable behavior under high system load.

4.5.1 Specialized Agent Taxonomy

The multi-agent layer is composed of domain-specialized agents organized into functional categories [54, 17].

Safety-Critical Agents

Safety Agent Monitors risk-related contextual variables (speed, proximity, hazardous conditions) and proposes interventions aimed at risk mitigation. This agent holds structural priority within the arbitration hierarchy.

Emergency Agent Activated under abnormal or critical system states. It overrides all non-essential adaptations and enforces minimal-interaction protocols.

Experience Optimization Agents

Comfort Agent Optimizes environmental parameters such as climate, seating adjustments and ride smoothness according to driver profile and contextual conditions.

Ambient Modulation Agent Manages lighting, soundscapes and atmospheric cues to maintain an optimal cognitive-emotional balance without exceeding interaction bandwidth limits.

Planning and Predictive Agents

Planning Agent Handles route management, task scheduling and contextual foresight integration.

Holistic Decision Agent Synthesizes cross-domain signals to propose system-level optimizations that consider multiple objectives simultaneously.

Interaction and Interface Agents

Voice Interaction Agent Manages speech-based communication, adapting verbosity and confirmation strategies according to cognitive state.

Adaptive UI Agent Dynamically modifies interface density, layout complexity and notification frequency based on cognitive load and situational demands.

System-Level Optimization Agents

Productivity Agent Enables task-oriented functions (calls, messages, scheduling) when cognitive bandwidth allows.

Adaptive Optimization Agent Continuously refines adaptation strategies using feedback loops and performance metrics.

This orchestrator-centric architecture ensures that distributed intelligence remains aligned with system-level safety, cognitive sustainability and experiential coherence. The result is a structured multi-agent system capable of deterministic arbitration, proactive adaptation and context-aware interaction modulation within the constraints of an automotive environment.

4.6 Decision Governance Framework

The Decision Governance Framework represents the core mechanism ensuring that multi-agent outputs remain coherent, safe and cognitively aligned. It governs how complex requests are decomposed, prioritized, evaluated, filtered for conflicts and, when necessary, deferred for later execution, integrating predictive cognitive modeling, safety constraints and agent proposals.

The overall governance process is illustrated in Figure 4.2.

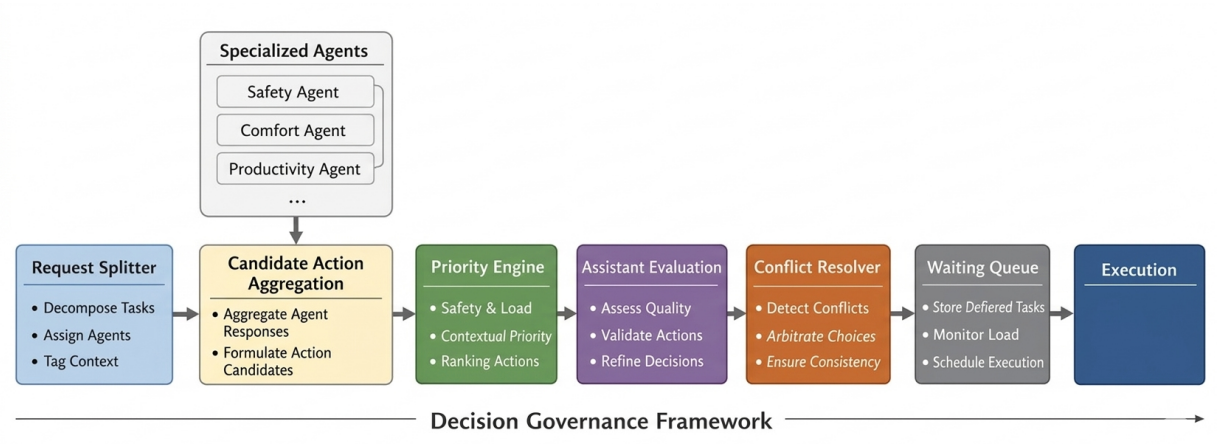

Figure 4.2: Decision Governance Framework: structured pipeline from request decomposition and candidate-action aggregation to validated execution, including prioritization, assistant evaluation, conflict resolution and deferred scheduling under cognitive constraints.

4.6.1 Request Decomposition

Complex user requests or system-driven objectives are first processed by the **Request Splitter**. Its function is to transform high-level or compound requests into a set of structured, actionable tasks that can be independently assessed by specialized agents.

Key aspects:

- Semantic parsing of natural language or system commands.
- Assignment of tasks to relevant agents according to domain specialization.
- Tagging of each sub-action with context, predicted cognitive load and preliminary priority.

This decomposition ensures that each agent works on manageable, contextually relevant tasks, enabling modular reasoning and facilitating traceability and explainability.

4.6.2 Priority Engine

The **Priority Engine** determines the execution order of actions generated by agents. Prioritization is guided by multiple factors:

- **Safety-criticality**: safety-related interventions always take precedence.
- **Cognitive Load**: actions that could overload the driver are deferred or simplified.
- **Contextual Relevance**: urgency or predicted impact of the action within the current driving scenario.
- **System Constraints**: execution timing, resource availability and interaction bandwidth.

The engine produces a ranked list of candidate actions, ensuring that the most critical and cognitively feasible operations are executed first. It also communicates with the **Waiting Queue** for deferred actions, maintaining global consistency.

4.6.3 Assistant Evaluation Layer

After prioritization, candidate actions are passed to the **Assistant Evaluation Layer**. This stage assesses the quality, safety and contextual appropriateness of each action before it can proceed further in the governance pipeline. Evaluation criteria include alignment with the driver's cognitive state, relevance to the current scenario, compliance with safety constraints and conformance with overall system objectives. Only actions that pass this validation stage are approved for subsequent processing, while unsuitable actions are rejected or reformulated through controlled fallback logic.

- Alignment with driver cognitive state.
- Relevance to the current scenario and environmental constraints.
- Compliance with safety constraints and system-level operational policies.

- Conformance with system-wide objectives (comfort, productivity, planning).

This layer contributes to system robustness by filtering unsuitable actions before execution and by supporting a more controlled and explainable governance process [36].

4.6.4 Conflict Resolution Mechanism

Once candidate actions have passed evaluation, the **Conflict Resolver** enforces deterministic arbitration among the approved proposals. Its role is to remove incompatible, redundant or mutually disruptive actions and to preserve global coherence across channels, priorities and cognitive constraints. In this way, conflict resolution operates on a filtered set of already validated actions rather than on the full raw candidate pool.

The main objectives of this mechanism are:

- Preserve system safety and prevent contradictory actions.
- Maintain cognitive feasibility by avoiding simultaneous high-load interventions.
- Ensure explainability by documenting reasoning behind conflict resolution.

Conflicts are resolved according to a hierarchical policy:

1. Safety overrides all other objectives.
2. Critical cognitive load constraints dictate deferral of non-essential actions.
3. Multi-domain conflicts are resolved using weighted priority scores derived from agent type, context urgency and predicted outcomes.

4.6.5 Waiting Queue and Deferred Execution

The **Waiting Queue** handles actions that are valid but cannot be executed immediately due to cognitive or contextual constraints. Its responsibilities include:

- Storing deferred actions with metadata for future execution.
- Monitoring predicted cognitive load trends to determine optimal execution windows.

- Integrating progressive disclosure principles to avoid overwhelming the driver.
- Ensuring that critical actions are never blocked and are escalated according to priority rules.

By combining deferred execution with predictive cognitive modeling, the architecture maintains a balance between responsiveness and safety, ensuring that adaptive interventions occur at the right time and under the right conditions [53, 42].

Together, the Request Splitter, Priority Engine, Conflict Resolver, Assistant Evaluation Layer and Waiting Queue form a comprehensive governance framework. This framework guarantees that agent-driven intelligence translates into safe, explainable and contextually appropriate system actions [36].

4.7 Structured LLM Integration

The integration of Large Language Models (LLMs) within the proposed architecture enhances contextual reasoning and adaptive interaction capabilities while preserving determinism, safety and system-level control [37, 21].

Unlike generic conversational deployments, LLMs in this system do not operate as autonomous decision-makers. Instead, they function as structured cognitive modules embedded within a tightly governed execution pipeline.

LLMs as Bounded Cognitive Modules

In the proposed architecture, LLMs are treated as *bounded cognitive modules*. Their role is limited to high-level semantic reasoning, contextual synthesis and the generation of candidate assistance actions within specific agents.

These candidate actions represent proposed system interventions that are subsequently evaluated, prioritized and validated by deterministic orchestration components before any execution may occur.

This bounded design enforces three fundamental constraints:

- **No direct actuation authority**: LLM outputs are never directly executed. Instead, they

represent candidate actions or interaction suggestions that must pass deterministic validation and prioritization layers before being considered for execution.

- **Domain confinement**: each LLM instance operates within a well-defined functional scope (e.g., planning, interaction, productivity).
- **Structured output requirement**: responses must conform to predefined schemas.

This approach prevents uncontrolled generative behavior and ensures that language models enhance, rather than replace, deterministic system components [37, 21].

Standardized Context Blocks

In order to guarantee coherence and reproducibility, all LLM calls are constructed using standardized context blocks. These blocks encapsulate:

- Current environmental context (traffic, route, hazards).
- Vehicle state and operational constraints.
- Driver profile and cognitive load estimation.
- System-level priorities and safety policies.

By structuring the input context explicitly, the system reduces ambiguity and improves reasoning stability [37].

Formally, each LLM invocation can be represented as:

$$R = \mathcal{M}(S_c, S_v, L_c, P_s)$$

where:

- S_c is the contextual state,
- S_v represents vehicle state variables,
- L_c denotes cognitive load estimation,

- P_s encodes system policies and constraints,
- R is the structured response.

This structured input-output interface ensures compatibility with downstream validation layers.

Provider Abstraction and Deployment Flexibility

The architecture incorporates a provider abstraction layer that decouples agent logic from specific LLM vendors or deployment environments.

This abstraction enables:

- Seamless switching between cloud-based and on-premise models.
- Support for multiple model families with consistent interfaces.
- Progressive deployment strategies (e.g., hybrid local-cloud inference).

By isolating provider-specific APIs behind a standardized interface, the system preserves long-term maintainability and technological flexibility. This is particularly relevant in automotive contexts, where regulatory, latency or data privacy constraints may require different deployment configurations.

Safety Constraints and Output Validation

All LLM outputs undergo a multi-layer validation process within the Decision Governance Framework before execution. This validation stage operates on candidate actions generated by the agents and ensures that only cognitively appropriate, policy-compliant and contextually valid actions can proceed through the governance pipeline.

Validation includes:

- **Schema validation**: ensuring structural compliance with expected formats.
- **Policy compliance checks**: rejecting outputs that violate safety or regulatory rules.

- **Cognitive feasibility assessment**: verifying alignment with current cognitive load constraints.
- **Preliminary compatibility checks**: flagging potential inconsistencies that may require downstream conflict resolution.

If validation fails, fallback mechanisms are activated. These may include deterministic rule-based responses, simplified interaction strategies or action deferral via the Waiting Queue.

This layered control mechanism effectively transforms inherently probabilistic language models into predictable, governable and certifiable system components.

Through bounded reasoning, structured context injection, provider abstraction and rigorous validation, the architecture achieves a controlled and engineering-compliant integration of generative AI within a safety-critical adaptive HMI system.

4.8 Architectural Comparison with the EPIGNOSIS Baseline

This section provides a structured comparison between the original EPIGNOSIS baseline architecture and the proposed orchestrator-centric multi-agent system [11, 12]. The objective is not only to highlight technological differences, but to clarify the evolution in control philosophy, cognitive modeling, AI integration and governance mechanisms.

Control Model Evolution

The EPIGNOSIS baseline architecture follows a predominantly sequential pipeline model. Functional modules process inputs in predefined stages, with limited cross-module coordination and implicit prioritization logic.

In contrast, the proposed architecture adopts an orchestrator-centric control paradigm. Rather than relying on static processing chains, it introduces a supervisory decision authority that dynamically coordinates specialized agents. This transition represents a shift:

- from linear execution to dynamic orchestration,
- from implicit precedence rules to explicit arbitration policies,

- from module-driven logic to system-level governance.

The result is a more adaptive and context-aware control structure capable of handling concurrent objectives and conflicting proposals.

Cognitive Modeling Advancements

In the baseline system, cognitive load is treated primarily as a threshold-based classification variable. Interaction strategies are simplified once predefined limits are exceeded, but cognitive estimation does not structurally constrain all decision processes.

The proposed architecture elevates cognitive load to a hard architectural constraint. All adaptation proposals are conditioned on real-time cognitive estimation and governance mechanisms (priority engine, waiting queue, conflict resolver) explicitly incorporate cognitive feasibility checks.

This evolution enables:

- predictive cognitive-aware adaptation,
- progressive disclosure strategies,
- prevention of overload through structural suppression of non-essential actions.

Conflict Handling and Governance Improvements

In the baseline architecture, conflict handling is largely implicit. Functional modules are assumed to operate within predefined boundaries and conflict resolution emerges from static priority rules.

The proposed architecture introduces an explicit Decision Governance Framework comprising:

- Request decomposition,
- Priority ranking,
- Assistant evaluation layer,
- Deterministic conflict resolution,

- Deferred execution via waiting queue.

This explicit governance layer improves explainability, robustness and safety compliance by ensuring that all competing actions are systematically evaluated before execution.

AI Integration Strategy

Within the baseline system, AI components primarily support task-specific functionalities. Their integration is modular and often optional, without structural control over decision authority.

In the proposed architecture, AI, particularly Large Language Models, is embedded as a structured reasoning layer within specialized agents. However, AI remains bounded and governed:

- No direct actuation authority,
- Structured input-output schemas,
- Multi-layer validation before execution.

This strategy transforms AI from a peripheral enhancement into an integrated yet controlled cognitive engine aligned with safety-critical requirements.

System-Level Adaptive HMI Unification

The baseline architecture treats adaptive HMI functions as partially distributed capabilities across modules.

The proposed system unifies adaptive interaction within a coherent orchestrated framework, where:

- Cognitive modeling,
- Agent-level reasoning,
- Governance mechanisms,
- Structured LLM integration,

converge into a single adaptive decision pipeline.

This unification enables consistent multimodal adaptation across visual, auditory and environmental channels, maintaining global coherence even under high system load.

To conclude, Table 4.1 summarizes the main architectural differences between the baseline EPIGNOSIS Adaptivity Engine and the proposed orchestrator-centric system. The comparison highlights how the introduction of coordinated agent reasoning, centralized governance and cognitive-aware constraints extends the capabilities of the original architecture.

Dimension	**EPIGNOSIS Baseline**	**Proposed Architecture**
Control Model	Sequential pipeline	Orchestrator-centric governance
AI Usage	Task-level support	Agent-level bounded reasoning
Conflict Handling	Implicit priority rules	Explicit conflict resolution framework
Cognitive Load	Threshold-based modulation	Hard architectural constraint
LLM Integration	Optional and modular	Structured abstraction with validation layer
Decision Governance	Limited and distributed	Centralized, multi-layer governance framework
HMI Adaptation	Partially distributed	System-level unified adaptive pipeline

Table 4.1: Architectural comparison between EPIGNOSIS baseline and proposed orchestrator-centric system

5. Implementation

5.1 Implementation Goals and Technology Stack

This chapter presents the main implementation choices adopted for the prototype, including the software stack, runtime organization, model integration strategy and supporting modules used to realize the proposed architecture.

From an implementation perspective, the system was designed around four main goals. First, it had to support **modularity**, so that each functional capability could be encapsulated in a dedicated component and independently extended or replaced. Second, it had to preserve **safe orchestration**, ensuring that critical conditions, such as emergency or high-risk driver states, could override standard adaptive flows and trigger immediate responses. Third, it had to enable **bounded AI integration**, where LLM-based reasoning could be exploited for contextual interpretation and action generation without directly controlling execution. Finally, it had to satisfy **near-real-time responsiveness**, which is essential in adaptive in-vehicle systems operating under continuously changing contextual conditions.

To satisfy these requirements, the prototype was implemented in **Python 3.11+**, selected for its flexibility in rapid prototyping, its extensive ecosystem for artificial intelligence and machine learning and its suitability for integrating heterogeneous services within a unified control layer. Python also supports a clean modular organization, which proved particularly useful for structuring the project into agents, orchestration modules, service components, external API connectors, machine learning utilities, simulation tools and evaluation scripts. This organization reflects the main design principle of separating context interpretation, reasoning, validation and action dispatch into distinct but interoperable units.

At the center of the implementation lies the **orchestrator**, which coordinates the full decision pipeline. Around this central layer, a set of specialized agents manage different domains, including safety, emergency handling, comfort, ambient regulation, planning, productivity, adaptive user interface generation, voice interaction and system optimization. These agents are implemented as

specialized runtime modules coordinated by the orchestrator according to contextual triggers and system priorities.

The surrounding service layer provides the operational mechanisms required by both the orchestrator and the agents. These include context analysis, cognitive load estimation, user request decomposition, action prioritization, conflict resolution, deferred queue management, LLM output evaluation, adaptive interface selection and external API integration. This service-oriented decomposition improves maintainability and traceability, since each major decision stage corresponds to a clearly identifiable software module.

A key implementation choice concerns the integration of **Large Language Models**. The system was designed to support both direct model access and provider-based inference through an abstraction layer, so that model invocation remains decoupled from higher-level orchestration logic. This makes it possible to preserve the same reasoning pipeline even if the underlying LLM backend changes. In the implemented prototype, language models are integrated through a bounded reasoning layer: their outputs are structured, validated and filtered before they can influence executable actions [37, 21]. This design is consistent with the safety-oriented principles discussed in the previous chapter [24, 36].

The software stack also includes optional integrations for multimodal and contextual enrichment. Speech-related functions can rely on external speech-to-text and text-to-speech services, while contextual augmentation can be supported through dedicated connectors for weather, maps, calendar and other mobility-related information sources. These integrations are accessed through dedicated modules, so that data retrieval remains separated from the internal orchestration logic and does not affect the modular structure of the system.

Table 5.1: Main technologies adopted in the implementation

Technology	**Role**	**Motivation**
Python 3.11+	Core implementation language	Rapid prototyping, modular design and broad AI/ML ecosystem support
`asyncio`	Asynchronous orchestration	Non-blocking coordination of concurrent services and external calls
LLM provider abstraction	Language model access layer	Decouples orchestration logic from the underlying inference backend
External APIs	Context enrichment	Provides weather, navigation, calendar and mobility-related information
Speech services	Voice interaction	Supports speech-to-text and text-to-speech capabilities
Machine learning modules	Predictive support	Enables cognitive trend estimation and proactive adaptation mechanisms
Simulation environment	Scenario execution	Supports controlled testing of contextual and behavioral conditions
Evaluation scripts	Experimental validation	Enables metric collection, comparison and analysis of system behavior

For adaptive and predictive behavior, the implementation also includes a machine learning layer. This layer supports functionalities such as cognitive trend prediction, behavioral pattern recognition

and proactive assistance mechanisms. More specifically, the predictive layer was structured around four complementary components. A `CognitiveTrendPredictor` is intended to estimate the short-term evolution of cognitive load from recent temporal observations, while a `RouteLearner` captures recurring mobility patterns such as time-dependent destination preferences. In parallel, a `BehaviorPatternRecognizer` is used to identify repeated stress conditions or preferred intervention windows, and a `PredictiveOrchestrator` exposes decision-oriented functions that support deferred execution and proactive timing selection.

From an implementation perspective, the predictive layer is hybrid rather than uniformly learning-driven. A dedicated LSTM-based sequence model was concretely implemented for cognitive-load forecasting, together with model-management utilities and an offline training pipeline operating on multivariate temporal sequences. In particular, the cognitive model processes sequences of 20 timesteps with 15 normalized features, including vehicle dynamics, emotional cues, head-pose information, driver-state indicators and contextual variables such as traffic density and weather severity.

At runtime, however, the integration remains deliberately conservative. The `CognitiveTrendPredictor` attempts to use the trained LSTM model only when model weights are available and sufficient feature history has been accumulated; otherwise, it reverts to lightweight statistical prediction based on recent observations and short-horizon trend extrapolation. This fallback-supported design preserves operational continuity while allowing more advanced sequence-based forecasting to be progressively incorporated without making the orchestration pipeline dependent on full training maturity.

The same distinction applies to the other predictive modules. Route and behavior prediction are already structured in an LSTM-ready form at architectural level and corresponding sequence models are defined in the machine-learning layer. However, in the current operational prototype, `RouteLearner` still relies on frequency-based destination estimation, while `BehaviorPatternRecognizer` identifies recurring patterns through accumulated observations and rule-based aggregation. As a result, the predictive subsystem should be interpreted as partially operational and progressively extensible: sequence learning is concretely supported, but the deployed runtime behavior remains safeguarded by deterministic fallback logic whenever learned components are unavailable or insufficiently mature.

Responsive execution is supported through asynchronous orchestration based on Python's `asyncio` library. This allows independent tasks, such as external API calls, proactive routing and candidate-action evaluation, to be coordinated concurrently without blocking the main control flow. Asynchronous execution is therefore used as an implementation mechanism to improve responsiveness while preserving centralized decision governance.

To support controlled experimentation and reproducible validation, the project also includes a dedicated **simulation and evaluation environment**. This environment makes it possible to execute predefined scenarios, compare alternative configurations, collect performance metrics and analyze system behavior under different workload conditions. The presence of this additional infrastructure is important because it shows that the implementation was not limited to a conceptual prototype, but was developed together with tools for systematic testing and assessment.

```
project/
|-- agents/
|-- core/
|-- services/
|-- ml/
|-- simulator/
|-- experiments/
|-- orchestrator.py
```

Figure 5.1: Simplified logical organization of the software project

The resulting technology stack reflects the dual nature of the proposed system. Overall, the adopted technology stack supports both adaptive AI-based reasoning and the explicit orchestration mechanisms required to keep such reasoning operationally controlled within the prototype.

5.2 Orchestrator Implementation

The orchestrator is the central runtime component of the implemented system. Its role is to coordinate the full decision pipeline, from input processing to final output generation, by

integrating contextual interpretation, cognitive-state estimation, agent-level reasoning, action filtering, deferred scheduling and command conversion within a single control flow.

From an implementation perspective, the orchestrator operates on a structured processing cycle. At each iteration, it receives heterogeneous inputs from the Driver Monitoring System (DMS), the Vehicle Control Unit (VCU), Advanced Driver Assistance Systems (ADAS) and the surrounding environment. These inputs are first collected and normalized into a shared internal representation, so that downstream modules operate on a consistent state rather than on source-specific raw data.

A preliminary stage of the pipeline consists in constructing a structured external-data envelope. This object aggregates the information received from ADAS, environmental sensing and forward-collision monitoring into a unified container. When location-related information is available, the envelope can be enriched through external contextual services, such as weather, traffic, charging-station availability, route information or calendar data. This enrichment step is optional and does not affect the execution of the internal pipeline when external services are unavailable.

The normalized inputs are passed to the context interpretation layer, which produces a structured `ContextState` used throughout the remaining stages of the pipeline. Cognitive-state estimation is then requested through the `CognitiveLoadManager`. The resulting value is stored in the contextual state together with the corresponding cognitive reserve and is subsequently used by prioritization, queue management and adaptation modules [53, 42].

Code 5.1: Simplified orchestrator processing cycle

```
async def process(...):
    envelope = build_external_data_envelope(...)

    if location and api_manager:
        api_data = await enrich_with_apis(location, destination)
        envelope = merge_api_data(envelope, api_data)

    context = context_interpreter.analyze(dms_data, vcu_data, envelope)

    if emergency_condition:
        return await emergency_fast_path(...)

    cognitive_load = await cognitive_manager.get_cognitive_load(...)
    context.cognitive_load = cognitive_load
```

```
    context.cognitive_reserve = 1.0 - cognitive_load

    request_actions, proactive_actions = await asyncio.gather(
        handle_user_request(...),
        proactive_router.route(...)
    )

    all_actions = request_actions + proactive_actions
    ranked_actions = await priority_engine.rank_actions(
                all_actions, context, ...
        )
        approved_actions = await evaluate_actions(
                ranked_actions, context, dms_data
        )
        resolved_actions = conflict_resolver.resolve(
                approved_actions, context
        )
        executable, queued = queue_management(
                resolved_actions, context
        )

    outputs = await convert_to_system_outputs(executable, ...)
    return outputs
```

The orchestrator includes an explicit emergency branch for highly critical situations. When impact conditions or severe collision risk are detected, the standard processing flow is bypassed and the request is forwarded directly to emergency handling. In this branch, non-essential stages such as standard ranking, deferred scheduling and secondary filtering are skipped in order to reduce latency and preserve deterministic response behavior under critical conditions [24].

In non-emergency conditions, candidate actions are generated through two complementary paths. The first path is reactive and processes explicit user requests by decomposing them into sub-tasks and routing them to the relevant domain agents. The second path is proactive and operates directly on the contextual state, allowing the system to generate candidate interventions even in the absence of explicit commands. This implementation supports both request-driven interaction and context-driven adaptation within the same orchestration cycle.

In the implemented prototype, the proactive branch is not treated as a generic background mechanism, but is mediated by a dedicated routing component. This `ProactiveAgentRouter` analyzes the current contextual state and selectively forwards it only to the agents whose activation

conditions are relevant in the current situation. In this way, proactive behavior is not obtained by broadcasting the full context to all agents indiscriminately, but by applying a first selection layer that reduces unnecessary processing and limits the generation of low-value candidate actions.

This routing strategy has two implementation advantages. First, it improves scalability, because the number of active reasoning paths remains bounded even when the agent ecosystem grows. Second, it increases behavioral coherence, since proactive interventions emerge only when the contextual configuration makes them plausible and useful. The proactive branch therefore acts as a controlled anticipatory mechanism, not as an always-on parallel assistant competing with the main orchestration logic.

To make this controlled activation logic more explicit, the main specialized agents implemented in the prototype are summarized below in terms of functional role, activation conditions and typical outputs.

5.2.1 Specialized Agents: Functional Roles and Activation Logic

Beyond the general orchestration logic, the proposed architecture includes a set of specialized agents designed to cover distinct but complementary dimensions of in-vehicle assistance. Their role is not merely to produce isolated suggestions, but to generate context-sensitive actions that can later be filtered, prioritized, coordinated, deferred or suppressed by the upper control layers. In this sense, agent specialization constitutes the functional basis on which the orchestration pipeline operates.

A first central component is the *Safety Agent*, which represents the most safety-critical branch of the architecture. Its behavior is explicitly trigger-based and reacts to conditions such as forward-collision warnings, driver drowsiness, excessive cognitive load and distraction. In particular, the implementation checks structured thresholds derived from driver monitoring and vehicle state and produces immediate actions when dangerous conditions are detected. Collision-related situations generate urgent multimodal warnings; drowsiness activates direct alerts and may include suggestions for nearby rest areas; high cognitive load can also trigger recommendations for more interventionist ADAS behavior and simplified interface modes. This agent therefore embodies the system's safety-first principle by ensuring that critical conditions are handled with maximum priority and minimal interpretative delay.

A second important module is the *Planning Agent*, which covers proactive planning functions related to battery management and vehicle maintenance. Its logic is activated when relevant planning conditions emerge, such as low battery charge, abnormal tire pressure or maintenance requirements signaled by contextual data. In low-battery situations, the agent analyzes available charging stations and formulates a charging plan that balances urgency, route compatibility, charging power, station availability and the driver's state. In maintenance-related situations, it proposes service appointments by considering the severity of the issue together with available calendar windows and route compatibility. This agent is therefore responsible for translating vehicle status and contextual constraints into anticipatory planning actions that reduce future inconvenience and operational risk.

The *Adaptive UI Agent* addresses the interface adaptation layer of the system and is particularly relevant to the cognitive-aware nature of the proposed HMI. Rather than producing a single fixed interface, the agent builds a structured UI adaptation context from cognitive load, cognitive reserve, attention level, emotion, driving complexity, vehicle speed, active tasks, weather, traffic and safety alerts. On this basis, it generates a layout configuration that specifies complexity mode, widget selection, position, size, detail level, visibility and global interaction settings. The implementation also includes validation and conflict-resolution mechanisms to ensure interface consistency: mandatory widgets are preserved, conflicting widgets are removed, invalid positions are resolved and changes are only applied when sufficiently meaningful. As a result, interface adaptation is treated not as a cosmetic variation, but as a governed decision process aimed at minimizing distraction and preserving information salience under changing cognitive conditions.

The *Ambient Modulation Agent* extends adaptation beyond the visual interface and addresses the modulation of the vehicle environment itself. Its operation is based on pattern extraction over repeated driving situations, combining temporal features, road type, speed, emotional indicators, stress level and scenario descriptors. When a sufficient number of similar historical patterns is available, the agent attempts to predict the likely emotional state in the near future and proactively configures environmental variables such as lighting, music, temperature and seat-massage settings. Importantly, this behavior is enabled only in intermediate cognitive-load conditions, which reflects a design choice: ambient adaptation should support regulation and comfort without interfering with highly critical or highly disengaged situations. This makes the agent a complementary

regulation layer whose purpose is to stabilize or improve the driver's experiential state before discomfort or tension fully emerge.

The *Holistic Decision Agent* is designed for situations in which multiple critical factors coexist and cannot be managed adequately through isolated single-domain responses. Its activation depends on the simultaneous presence of at least two relevant complexity factors, such as low battery, urgent meetings, high stress, heavy traffic or adverse weather. In those cases, the agent synthesizes driver state, driving context, external API data, vehicle information and time constraints into a structured multi-option decision output. Rather than issuing a single imperative suggestion, it constructs alternative timelines, each associated with estimated arrival time, battery at arrival, stress implications, punctuality expectations, pros, cons and success probability. The agent then recommends the most balanced option and provides an explicit rationale. This behavior is particularly important because it operationalizes the idea of cognitive support in complex trade-off scenarios, where the goal is not only to react, but to help the driver compare alternatives under uncertainty.

Finally, the *Adaptive Optimization Agent* introduces an opportunistic layer aimed at exploiting favorable temporal or cognitive conditions rather than only mitigating critical ones. Its implementation supports two distinct modes. The first is a *flow-state* mode, in which the system detects cognitively favorable states based on load, engagement and driving conditions, and then proposes high-value tasks that can take advantage of that moment of optimal concentration. The second is a *temporal arbitrage* mode, in which the agent identifies short predicted temporal gaps and allocates small pending tasks to those windows in order to maximize value while minimizing disruption. This agent is intentionally assigned a low-priority role, since its function is not essential intervention but intelligent opportunity management. Nevertheless, it is conceptually important because it shows that the architecture is not limited to defensive adaptation; it can also support productive and efficient interaction when situational conditions are favorable.

Taken together, these specialized agents show that the proposed system is not based on a monolithic assistant with generic behavior, but on a differentiated set of context-aware decision modules. However, their outputs do not directly translate into executable system behavior. They become operational only when reintegrated into the broader orchestration pipeline described in this chapter, where urgency, conflict resolution, waiting logic and cognitive suitability determine

whether, when and how each action is ultimately delivered to the driver. This separation between specialized action generation and centralized governance is one of the main architectural strengths of the proposed multi-agent approach.

The candidate actions generated by the reactive and proactive branches are then merged and forwarded to the prioritization module. At this stage, they are ranked according to urgency, safety relevance, contextual appropriateness, cognitive compatibility and expected utility. Ranking is necessary because different agents may produce overlapping, redundant or competing actions within the same cycle.

After prioritization, the resulting candidates are passed to the evaluation layer. This stage verifies whether each action satisfies structural, quality and safety constraints before it can be considered for execution. Actions that pass evaluation are approved for further processing, while actions that fail validation are either discarded or reprocessed according to the logic of the corresponding service module. This stage is particularly relevant for actions generated through LLM-based reasoning, since it enforces bounded integration of model outputs within the execution pipeline.

The approved actions are then processed by the conflict-resolution module. This stage removes incompatible or redundant decisions and ensures that the final set of actions is globally coherent. Conflict filtering is necessary because multiple agents may target the same output channel, recommend semantically equivalent interventions or generate mutually incompatible adaptations.

A subsequent stage manages deferred execution. Actions that are not urgent can be passed to a waiting queue instead of being executed immediately. Queue management is used primarily to postpone non-critical interventions when the driver is under high cognitive load or when the current context is not suitable for interaction. In the implemented system, this mechanism is integrated with predictive logic, so that postponement depends not only on the current state but also on the estimated suitability of future execution windows.

In the final stage of the pipeline, the selected actions are converted into concrete `SystemOutput` objects. This conversion maps abstract agent-level recommendations into channel-specific commands for modules such as infotainment, navigation, ADAS, climate control, ambient lighting, haptic feedback, application interaction, battery management and adaptive user interfaces. The conversion layer separates decision generation from actuator-specific implementation and therefore

improves both modularity and extensibility.

The runtime implementation also adopts asynchronous coordination to improve responsiveness. Independent operations, such as external-data retrieval, proactive action generation and action evaluation, can be executed concurrently through Python's `asyncio` framework. This reduces blocking time in the main control loop while preserving centralized orchestration and deterministic governance of the decision flow.

The orchestrator therefore implements the operational logic of the proposed architecture as a centralized execution pipeline. Its implementation integrates multimodal input processing, cognitive-state estimation, bounded LLM-based reasoning, action validation, conflict filtering, deferred scheduling and multi-channel output generation within a single coordinated runtime structure.

5.3 Cognitive Load Estimation and Management

Cognitive load estimation is implemented as a dedicated subsystem of the runtime pipeline rather than as an auxiliary computation embedded into individual agents. This design choice allows the system to expose a single and coherent estimate of the driver's mental workload to all downstream decision stages, including prioritization, adaptive interface selection and deferred execution [48, 53, 57]

From an implementation perspective, the subsystem is organized into two layers. The first layer is the `CognitiveLoadEstimatorAI`, which computes the cognitive load value from multimodal inputs. The second layer is the `CognitiveLoadManager`, which acts as the production entry point and provides centralized access, time-based caching, history tracking and simple trend estimation. This separation keeps the estimation logic independent from runtime optimization concerns and makes the overall subsystem easier to maintain and extend.

The estimation process starts from a multifactor baseline model implemented in the `CognitiveLoadEstimatorAI`. The baseline score is computed as a weighted combination of seven components: driving-task load, driver-state load, ADAS-related load, environmental load, attention dynamics, maneuver complexity and thermal load. The final baseline value is normalized in the range $[0, 1]$, where higher values represent greater mental demand. This implementation

makes it possible to combine psychophysiological, vehicular, environmental and assistance-related factors within a single scalar measure [57, 15].

The selection of these input families is consistent with prior work on driver mental workload, multimodal workload estimation and driver monitoring, which suggests that reliable workload-aware support should not rely on a single signal source but rather integrate task-related, driver-related and contextual factors [48, 53, 57, 15]. At the same time, the numerical parametrization adopted in the prototype should be interpreted as design-driven and heuristic rather than as the result of statistical calibration on annotated ground-truth data. In other words, the literature informed the choice of the contributing variables, whereas the exact weighting scheme was defined at implementation level to obtain a coherent and operational workload signal for downstream orchestration.

The task-load component models the cognitive demand associated with the current driving situation. It includes speed-dependent contributions, traffic conditions, road curvature, intersection proximity, pedestrian crossings and vehicle operating mode. Traffic information can be derived either from onboard contextual data or from external API enrichment. When external traffic information is available, it is preferred over local estimates in order to use a more accurate representation of congestion and route conditions. This component is literature-consistent, since driving workload is known to increase with task complexity, traffic demand and concurrent attentional requirements [53, 55, 57].

The driver-state component captures the psychophysiological condition of the driver using DMS signals. In the implemented model, anger, fear, sadness, drowsiness, distraction, engagement level and affective valence all contribute to the estimated workload. This component is intended to capture the fact that mental demand is not determined solely by the external task, but also by the internal state of the driver [15, 9, 29].

The ADAS-related component models the cognitive impact of warnings and interventions generated by assistance systems. The current implementation considers forward-collision warning status, AEBS activation, ELKS state, ISA warnings, excessive speed relative to the speed limit, ADAS attention warnings and the severity of forward-collision events. This allows the estimation layer to represent the increase in mental demand associated with active safety interventions and concurrent warning conditions.

The environmental component captures the contribution of weather, visibility, road type and time of day. The implementation uses a unified weather-and-visibility function that can rely either on external API weather data or on local DVE measurements. When API data are available, they are used as the preferred source. This provides a consistent mechanism for incorporating external context into the cognitive-load estimate without modifying the structure of the calculation pipeline.

A further component models attention dynamics through head-pose variation and gaze behavior. The implementation considers both absolute head movement and temporal variation with respect to the previous observation, as well as gaze magnitude and gaze displacement over time. This enables the estimator to capture both rapid scanning behavior and abnormal fixation patterns, which may be associated with increased cognitive demand, distraction or reduced situational stability [53, 15].

The remaining two components account for maneuver complexity and thermal conditions. Maneuver complexity is derived from indicators such as gear status, turn signals and hazard lights, while thermal load depends on cabin temperature, outside temperature and large temperature gaps between cabin and external conditions. Although these factors have lower weights than the main components, they provide additional contextual sensitivity and improve the behavioral realism of the estimate.

These components should be interpreted as complementary contextual modifiers rather than as primary workload predictors. In particular, maneuver-related signals and thermal conditions were introduced to increase the behavioral realism and context sensitivity of the estimator, even though their role is more heuristic and implementation-driven than that of the main task-, driver- and attention-related components.

The following table summarizes the weighting scheme used by the baseline estimator.

Component	Weight	Main input categories
Task load	0.25	Speed, traffic, road curvature, intersections, pedestrian crossings, vehicle mode
Driver-state load	0.25	Emotion, drowsiness, distraction, engagement, valence
ADAS-related load	0.15	FCW, AEBS, ELKS, ISA, ADAS attention warnings
Environmental load	0.15	Weather, visibility, road type, day/night conditions
Attention dynamics	0.10	Head pose, gaze amplitude, gaze variation over time
Maneuver complexity	0.05	Gear status, turn indicators, hazard lights
Thermal load	0.05	Cabin temperature, external temperature, thermal mismatch

Table 5.2: Baseline cognitive-load components and weights

The adopted weighting scheme reflects a deliberate architectural prioritization. Higher weights were assigned to driving-task load and driver-state load because these dimensions are the most directly associated with the driver's immediate cognitive demand during vehicle operation. Intermediate weights were assigned to ADAS-related and environmental factors, since they strongly affect workload but typically do so through contextual amplification rather than as primary standalone sources. Lower weights were assigned to attention dynamics, maneuver complexity and thermal load, which were treated as corrective and complementary contributors intended to refine the estimate without dominating it in ordinary conditions. This choice was therefore guided by domain knowledge and implementation goals, rather than by automated parameter fitting or regression-based optimization.

In addition to the baseline computation, the estimator supports an optional AI-based refinement stage. When enabled through configuration and when an LLM provider is available, the system sends a structured prompt containing the baseline estimate together with DMS, VCU, DVE, ADAS, FCW and external API information to the language model. The model is instructed to return a JSON object containing a refined score, a short explanation and a list of key contributing factors. The final cognitive-load estimate is then computed as a weighted combination of the rule-based baseline and the LLM-refined value, with coefficients 0.3 and 0.7 respectively. If

refinement fails for any reason, the system falls back to the baseline score [37, 21].

The 0.3 / 0.7 combination between baseline and AI-refined estimate should also be interpreted as a design choice rather than as a statistically optimized parameter setting. The baseline component was preserved as a deterministic anchor, ensuring continuity, interpretability and fallback robustness even when the LLM-based refinement is unavailable or fails. A larger weight was assigned to the refined component because the purpose of the AI stage is precisely to capture higher-order and non-linear interactions among heterogeneous factors that are difficult to encode through a purely additive rule-based model, such as the joint effect of traffic congestion, adverse weather, distraction and active safety warnings. The resulting formulation therefore reflects an architectural balance between deterministic stability and contextual expressiveness.

This hybrid design has two implementation advantages. First, it preserves deterministic availability through the rule-based baseline, which remains computable even when no LLM backend is accessible. Second, it allows the system to account for higher-order interactions among factors that are difficult to encode in a purely additive formula, such as the joint effect of adverse weather, traffic congestion, distraction and active safety warnings.

In production, the estimator is not accessed directly by the orchestrator. Instead, the runtime pipeline uses the `CognitiveLoadManager`, which acts as the single entry point for cognitive-load queries. The manager introduces a cache with a default time-to-live of two seconds and uses lightweight hashes of selected DMS and VCU fields to detect whether recalculation is necessary. If the cache is still valid and the relevant input hashes have not changed, the previously computed value is reused. This reduces redundant computation during consecutive orchestration cycles and improves runtime responsiveness.

Besides caching, the manager stores a bounded history of the most recent estimates. In the current implementation, this history is used to derive simple trend information, classified as `increasing`, `decreasing` or `stable`, and to compute a short-horizon prediction through lightweight statistical extrapolation when enough samples are available. This choice is consistent with the current prototype stage, in which operational prediction remains fallback-supported even though the broader predictive layer already includes dedicated LSTM model definitions and an offline training pipeline for more advanced sequence-based forecasting.

Code 5.2: Simplified cognitive-load computation flow

```
if cache_is_valid(dms_data, vcu_data):
    return cached_value

baseline = calculate_baseline(
    dms_data, vcu_data, context, dve_data, adas_data, fcw_data, api_data
)

if use_ai and llm_available:
    refined = ai_refinement(...)
    cognitive_load = 0.3 * baseline + 0.7 * refined
else:
    cognitive_load = baseline

update_cache(cognitive_load)
update_history(cognitive_load, context.scenario)

return cognitive_load
```

Within the orchestrator, cognitive-load management is integrated immediately after context interpretation and before reactive or proactive action generation. Once the value has been obtained from the manager, it is stored in the contextual state as `context.cognitive_load`, while the complementary quantity `context.cognitive_reserve` is computed as $1.0 - $ `cognitive_load`. This makes both values directly available to the priority engine, queue management logic, adaptive interface selection and other modules that need to reason about the driver's current processing capacity.

The implementation also includes a specific **emergency exception**. In critical collision-related situations, the orchestrator bypasses the normal cognitive-load computation and directly activates the emergency branch. In this case, the contextual state is forced to maximum load and minimum reserve in order to represent the criticality of the situation while minimizing latency. This exception is consistent with the safety-first execution model of the system and avoids delaying emergency handling with non-essential computation.

The resulting subsystem treats cognitive load as a centralized runtime variable combining multifactor estimation, optional LLM refinement, cache-based optimization and lightweight temporal reasoning. This design makes cognitive load a first-class operational variable of the system rather than a passive annotation and enables its direct use in orchestration, prioritization

and adaptive interaction management.

From a methodological perspective, the cognitive-load estimation adopted in this prototype was not validated through a dedicated human-subject study. In particular, it was not possible within the scope of the present work to collect workload annotations from test participants or to conduct a direct validation campaign based on subjective instruments such as NASA-TLX, nor to compare the estimator against physiological ground-truth measures. As a consequence, the estimated cognitive-load value should not be interpreted as a clinically or psychophysiologically validated measure of the driver's true mental state, but rather as an operational system-level signal designed to support adaptive orchestration, prioritization and interaction filtering within the proposed framework.

Nevertheless, the role of the estimator is not left entirely unassessed. Its contribution is evaluated indirectly at system level in Chapter 6, where the impact of the cognitive-load module and of its AI-enhanced refinement is analyzed through comparative experiments and ablation results. In this sense, the validation provided in this thesis should be interpreted as internal, architectural and comparative rather than as a direct ground-truth validation of the estimator as an autonomous measurement model. This distinction is important for correctly positioning the contribution of the present work: the goal is not to claim a definitive psychometric validation of cognitive-load measurement, but to show how a structured workload signal can improve the adaptive behavior of an orchestrated HMI architecture.

5.4 LLM Integration, Prompt Structuring and Execution Safeguards

Large language models are integrated into the prototype as bounded reasoning components that support context-sensitive interpretation, candidate-action generation, task decomposition and adaptive content synthesis. Their role is not to directly control execution, but to produce structured intermediate outputs that remain subject to explicit validation, sanitization, prioritization and conflict filtering before entering the final decision pipeline [37, 21].

From an implementation perspective, access to language models is encapsulated behind

a provider abstraction layer. The common interface is defined by the abstract `LLMProvider` class, which exposes a unified `generate(...)` method together with provider metadata. This allows the remaining components of the system to invoke language models without depending on backend-specific APIs or transport details. In the current implementation, the abstraction supports at least two backends: direct Anthropic access and AWS Bedrock-based inference. Provider selection is handled through a dedicated factory layer, which instantiates the appropriate backend from configuration or runtime settings. This design keeps the orchestration pipeline independent from the specific deployment target of the language model and improves portability across alternative deployment environments.

The implementation adopts a factory-based instantiation strategy through `LLMFactory`, which reads the selected backend from the runtime settings and creates the corresponding provider object. This choice makes the orchestration logic independent from provider-specific configuration details and allows the same reasoning pipeline to be preserved even when the underlying inference backend changes. As a result, model access becomes a replaceable infrastructure concern rather than a hard-coded dependency embedded in the decision flow.

In the current prototype, AWS Bedrock is used as one of the main provider options for model access. The `BedrockProvider` initializes a `boto3` client for the `bedrock-runtime` service, stores the selected region and model identifier and exposes an asynchronous `generate` method compatible with the common provider interface. The request body is constructed according to the Bedrock message API, including the Anthropic protocol version, the message sequence, the generation budget and the sampling temperature, with optional support for a separate system prompt. The returned response is then parsed to extract the generated textual content, which is passed back to the calling service in normalized form. This implementation detail shows that provider integration is not limited to a configuration-level abstraction, but is concretely realized through a dedicated execution layer aligned with the rest of the architecture.

A complementary low-level wrapper, implemented as `BedrockClient`, reproduces the same Bedrock invocation pattern through a message-oriented interface inspired by the structure of external SDK clients. This component encapsulates the raw Bedrock response into lightweight objects exposing content, model information, stop reason and usage metadata. Although the

higher-level provider abstraction remains the main entry point used by the orchestrator, this additional wrapper confirms the design choice of isolating provider-specific communication logic from the rest of the system and of normalizing outputs before they reach downstream decision modules.

At runtime, agent-level LLM invocation is standardized through the `BaseAgent` class. This class provides two reusable helper methods: `_call_llm(...)` for generic text generation and `_call_llm_json(...)` for structured JSON-oriented outputs. The first method performs provider-aware inference and removes Markdown code fences when present. The second extends this mechanism with JSON parsing and optional response sanitization. As a result, individual agents do not implement provider-specific logic directly, but rely on a shared inference layer that standardizes model access, response cleaning and error handling.

A similar abstraction principle was adopted for speech interaction. Voice-based operation is coordinated through a dedicated controller that manages the full speech pipeline from speech-to-text acquisition to language-model interpretation and text-to-speech rendering. Also in this case, provider-specific dependencies are encapsulated behind dedicated modules, so that different STT and TTS services can be used without changing the higher-level orchestration logic.

This choice is relevant for two reasons. On the one hand, it preserves portability across alternative deployment environments and cloud providers. On the other hand, it keeps voice interaction aligned with the same bounded-integration philosophy adopted for language models, since spoken requests are first transformed into structured internal representations before they can affect runtime behavior.

Code 5.3: Simplified LLM invocation and validation flow

```
response_text = await agent._call_llm_json(
    prompt=task_prompt,
    system_prompt=system_prompt,
    auto_sanitize=True
)

if not response_text:
    return fallback_action()

parsed = sanitize_ai_response(response_text, agent_name)
```

```
evaluation = await assistant_evaluation.evaluate(
    action=build_action(parsed),
    context=context,
    user_state=user_state
)

if evaluation["approved"]:
    return build_action(parsed)
else:
    return await assistant_evaluation.suggest_modification(...)
```

Prompt design follows a task-specific strategy. Instead of relying on a single generic prompt template, each agent defines prompts tailored to its operational role and expected output schema. This means that prompt engineering is not treated as a generic conversational technique, but as a software mechanism embedded in the service modules that rely on LLM reasoning. Each prompt specifies the role of the model, the contextual variables that must be considered, the relevant decision criteria and, most importantly, the exact structure of the output to be produced.

Representative excerpts of the schema-constrained prompts adopted by the main LLM-supported modules are reported in Appendix B. These excerpts are included to document the bounded prompting strategy used in the prototype and to make explicit the role of output contracts, contextual grounding and deterministic validation constraints in the overall implementation.

For example, the `ComfortAgent` uses different prompts for traffic stress, understimulation and moderate overload, with explicit constraints on admissible `action_type` values, target channels and required JSON fields. The `SafetyAgent` follows the same approach for scenarios such as collision warning, drowsiness and high cognitive load, again constraining both the intervention type and the response structure. The `HolisticDecisionAgent` uses a prompt designed to generate multi-timeline decision support, including alternative futures, probabilistic outcomes and an explicit recommendation. Similarly, the `RequestSplitter` relies on an LLM prompt to decompose a free-form request into atomic tasks associated with target agents, dependencies, priorities and execution strategy. In this case, the prompt also requires the generation of a clarification task when the request is ambiguous.

A recurrent design pattern in these prompts is the use of explicit output contracts. Prompts typically require the model to return only valid JSON, without Markdown wrappers, free text

or explanatory content outside the target object. They also constrain valid field names, action types, channels and semantic ranges. This reduces response ambiguity and improves downstream parsability. A dedicated schema-guidance block is reused across multiple agents to enumerate the admissible `action_type` and channel values accepted by the system. Embedding these constraints directly in the prompt reduces the probability of backend-specific output drift and improves compatibility with the internal action model [37].

The adaptive user-interface module provides the most explicit example of this strategy. In `AdaptiveUIAgent`, the system prompt includes a strict JSON schema, a valid reference example, a serialized widget catalog, conflict-prevention rules, cognitive-load-dependent design principles and output constraints such as mandatory widget types, admissible visibility values and bounded priority levels. The generated response is parsed into a `UILayoutConfig` object and then processed by additional validation logic that resolves position conflicts, removes incompatible widgets and checks widget-level consistency. This module is representative of the broader integration strategy adopted in the prototype: the LLM proposes a structured configuration, while final system behavior remains governed by deterministic validation logic.

The same structured prompting strategy is also used in downstream evaluative modules. In the `PriorityEngine`, the model does not generate executable actions directly, but estimates the *cognitive fit* of an already proposed action by considering message content, channels, agent identity, current cognitive load, cognitive reserve and scenario constraints. The LLM output is again restricted to a compact JSON object containing a numerical score and a short justification, which is then integrated with deterministic components such as safety relevance, urgency, user value and context alignment to produce the final priority score. This shows how generative reasoning is embedded within a broader computational pipeline rather than replacing explicit control logic.

Because model outputs are not assumed to be reliable by default, the implementation includes a dedicated sanitization layer. In `agent_utils.py`, action types and channels returned by the model are normalized against predefined valid sets. The module provides exact mappings, fuzzy matching and keyword-based fallbacks to convert non-canonical values into valid internal representations. It also includes helpers for cleaning JSON responses, extracting the first valid

JSON object from noisy text and removing trailing commas before parsing. This layer makes the runtime pipeline tolerant to small deviations in LLM formatting while preserving a well-defined internal vocabulary for actions and channels.

A further control stage is provided by the `AssistantEvaluation` module, which performs quality assessment before execution. Each generated action is evaluated with respect to safety, appropriateness, clarity, actionability and user value. The evaluator uses a structured prompt to return an assessment containing approval status, a global quality score, dimension-level scores, issues, suggestions and a recommended decision. A quality threshold is enforced and the prompt explicitly requires rejection in clearly unsafe situations. If JSON parsing fails, the module applies a fallback evaluation instead of blocking the pipeline. The evaluator also supports post-hoc revision through a second prompt that reformulates rejected actions while preserving their original intent. In this way, LLM-generated outputs are not passed directly to execution, but remain subject to an explicit quality-control stage [36, 21].

The final safeguard layer is provided by the execution governance modules coordinated by the orchestrator. The orchestrator instantiates the request splitter, priority engine, assistant evaluation, conflict resolver and waiting queue as separate services connected to the same shared LLM provider and uses them as intermediate control stages between context interpretation and action execution. This means that LLM-based reasoning can contribute to request decomposition, cognitive prioritization and action assessment, but no generated output is executed without first passing through explicit orchestration logic. The architectural role of the language model is therefore assistive and deliberative, while authority over activation, filtering, ordering and timing remains in deterministic supervisory modules.

Conflict management introduces an additional protection mechanism. The `ConflictResolver` orders actions according to severity, enforces the precedence of safety-related interventions, removes redundant outputs produced by the same agent and limits the number of simultaneous actions under very high cognitive load. Lower-priority actions are not necessarily discarded, but may instead be deferred so that they can be reconsidered under more suitable conditions. This prevents the language model from indirectly increasing interaction overload through the generation of multiple concurrent suggestions.

Deferral is handled by the `WaitingQueue`, which stores pending tasks together with execution windows, retry counters, required cognitive reserve and optimal cognitive-load ranges. A task is proposed for execution only when temporal constraints and cognitive conditions are jointly satisfied; expired tasks are discarded, while failed tasks can be retried with bounded backoff. This mechanism is particularly coherent with the cognitive-adaptive philosophy of the system because it transforms action timing into a controlled implementation concern rather than leaving it to immediate LLM output. In this way, the architecture separates the generation of a candidate action from the decision of when that action should actually reach the driver.

Taken together, structured prompting, backend abstraction, response sanitization, post-generation evaluation and execution-time safeguards implement the bounded reasoning model adopted throughout the prototype. Candidate outputs produced by the language model remain intermediate artifacts until they have been transformed into valid internal representations and checked against execution-oriented constraints. This is consistent with the role assigned to LLMs in the overall architecture: they extend the contextual reasoning capabilities of the system without bypassing deterministic governance [36, 37].

Table 5.3: Main LLM-supported modules and generated outputs

Module	**LLM role**	**Expected structured output**
RequestSplitter	Request decomposition	List of atomic tasks with target agent, priority, dependencies and execution strategy
SafetyAgent	Safety message generation	JSON object with message, channels and scenario-specific action details
ComfortAgent	Comfort/stress intervention generation	JSON object with message, channels and intervention-specific fields
HolisticDecisionAgent	Multi-option decision support	JSON object with alternative timelines, recommendation and reasoning
AdaptiveUIAgent	UI layout synthesis	`UILayoutConfig` JSON with widgets, priorities, global settings and reasoning
PriorityEngine	Cognitive-fit estimation for candidate actions	JSON object with score and concise justification
AssistantEvaluation	Quality assessment of generated actions	JSON object with approval, scores, issues, suggestions and recommended action

As a result, the implemented LLM layer operates as a controlled inference subsystem embedded within the orchestration pipeline. Its behavior is constrained by backend abstraction, task-oriented prompt design, structured outputs, deterministic sanitization and downstream governance, making LLM-based reasoning compatible with the safety-aware and execution-bounded design principles of the proposed system.

5.5 Real-Time Constraints and Performance Optimization

In an in-vehicle intelligent assistant, implementation quality is determined not only by the correctness of the generated decisions, but also by the ability to produce them within time constraints compatible with the current driving context. For this reason, the proposed prototype was designed with *near-real-time responsiveness* as an explicit implementation objective. In practice, this means that the runtime architecture must preserve modularity and bounded AI integration while keeping the decision cycle sufficiently reactive under continuously changing contextual conditions [24, 22].

From an implementation perspective, responsiveness is achieved through a combination of architectural and runtime-level strategies rather than through a single optimization technique. The most relevant mechanisms are asynchronous orchestration, lightweight state reuse, emergency fast-path handling, fallback logic for non-critical failures and selective pruning or postponement of actions that are not suitable for immediate execution. Taken together, these mechanisms reduce blocking behavior, avoid unnecessary recomputation and keep the execution pipeline aligned with safety-oriented timing requirements.

5.5.1 Asynchronous Orchestration and Non-Blocking Execution

The first optimization layer concerns the orchestration strategy itself. The runtime pipeline is implemented around an asynchronous control flow in which independent operations can be coordinated concurrently through Python's `asyncio` framework. This choice is already reflected in the technology stack of the project, where asynchronous orchestration is explicitly introduced as a mechanism for non-blocking coordination of concurrent services and external calls. The goal is not to distribute control across independent controllers, but to preserve centralized governance while avoiding unnecessary waiting time in the main decision loop.

At orchestrator level, asynchronous execution is used in at least two relevant parts of the runtime flow. First, optional context enrichment through external APIs is triggered asynchronously when location-related information is available. Second, after context interpretation and cognitive-load estimation, the reactive branch handling explicit user requests and the proactive branch generating context-driven interventions are executed concurrently through `asyncio.gather(...)`. This

allows request-driven and proactive reasoning to proceed in parallel before their outputs are merged and passed to the prioritization stage. Such coordination reduces blocking time without relaxing the centralized supervision of the pipeline, since synchronization still occurs before ranking, evaluation and execution filtering.

The use of asynchronous coordination is particularly important because the runtime pipeline combines multiple heterogeneous operations, including context analysis, external-data retrieval, cognitive-state estimation, candidate-action generation, action evaluation and output conversion. Some of these stages are independent enough to benefit from concurrent scheduling, whereas others must remain sequential in order to preserve determinism and policy enforcement. The implementation therefore uses concurrency selectively: it accelerates compatible branches while keeping the governance stages—prioritization, evaluation, conflict resolution and queue management—under explicit orchestration control.

5.5.2 Caching and Lightweight State Reuse

A second optimization strategy concerns the treatment of cognitive load as a frequently queried runtime variable. Since cognitive-state estimation is needed early in the orchestration cycle and its result is consumed by multiple downstream modules, recomputing it at every iteration would introduce avoidable overhead. For this reason, the production pipeline does not access the estimator directly, but routes all queries through the `CognitiveLoadManager`, which acts as the runtime entry point for caching and lightweight state reuse.

The manager introduces a cache with a default time-to-live of two seconds and uses lightweight hashes of selected DMS and VCU fields to determine whether recalculation is necessary. If the cache is still valid and the relevant hash values have not changed, the previously computed estimate is reused instead of invoking the full estimation flow again. This optimization is particularly effective in short consecutive orchestration cycles, where the contextual state may remain sufficiently stable for the cached value to remain representative. In such cases, the system reduces redundant computation while preserving a coherent estimate of the driver's current workload.

The cognitive-load subsystem also stores a bounded history of recent estimates. In the current implementation, this history is used to derive lightweight trend information—for example

`increasing`, `decreasing` or `stable`—and to support simple short-horizon extrapolation when enough samples are available. Although this mechanism is intentionally lightweight, it contributes to runtime efficiency in two ways. First, it avoids the need for heavier predictive processing in the core orchestration loop. Second, it provides a minimal temporal signal that can support queue management and proactive adaptation without introducing a separate high-cost forecasting stage into every cycle.

An additional optimization effect comes from the internal structure of the estimator itself. The baseline cognitive-load score remains deterministically computable through a rule-based multifactor model, while the optional AI-based refinement stage is executed only when enabled and when an LLM provider is available. If refinement fails, the system falls back to the baseline estimate rather than blocking the pipeline. This hybrid design supports responsiveness because it avoids turning LLM availability into a hard dependency for the production of a cognitive-load value. In other words, estimation quality can improve when refinement succeeds, but runtime availability is preserved even when the AI stage is disabled or unavailable.

5.5.3 Emergency Fast Paths, Fallback Logic and Runtime Pruning

A third group of optimizations concerns the reduction of unnecessary work under critical or unfavorable conditions. The most explicit example is the emergency branch implemented in the orchestrator. When highly critical situations such as impact conditions or severe collision risk are detected, the standard processing flow is bypassed and the request is forwarded directly to emergency handling. In this branch, non-essential stages such as standard ranking, deferred scheduling and secondary filtering are skipped in order to reduce latency and preserve deterministic behavior under high-risk conditions [24]. This fast-path design reflects a safety-first runtime policy: when the situation is critical, responsiveness is improved not by accelerating the full pipeline, but by temporarily removing stages that are useful in normal conditions but not essential in emergency handling.

A similar principle is applied inside the cognitive-load subsystem. In critical collision-related situations, the orchestrator can bypass the normal cognitive-load computation and force the contextual state to maximum load and minimum reserve. This avoids delaying emergency handling with estimation steps that would add little operational value in an already critical condition. The

implementation therefore treats fast-path simplification as an explicit optimization mechanism: reducing computational effort in the most time-sensitive situations can be more effective than attempting to preserve the complete decision process under all circumstances.

Fallback logic is also used as a performance-preserving strategy in non-emergency situations. In the cognitive-load subsystem, failure of the AI-based refinement stage does not interrupt the pipeline, since the system falls back to the deterministic baseline estimate. Similarly, in the broader architecture, core functionality is designed to remain available even when predictive components or learned modules are not fully deployed. This is an important implementation choice because it prevents auxiliary intelligence layers from becoming single points of failure or major sources of runtime instability. From a performance perspective, fallback logic protects responsiveness by ensuring that the system can continue producing bounded decisions even when some advanced components are unavailable.

Beyond fast paths and fallbacks, the implementation also optimizes runtime behavior by pruning or postponing actions that are not suitable for immediate execution. After prioritization and evaluation, approved actions are passed to the conflict-resolution stage, which removes incompatible or redundant decisions and ensures global coherence of the final output set. The `ConflictResolver` also limits the number of simultaneous actions under very high cognitive load, giving precedence to safety-related interventions and preventing low-value concurrency from increasing interaction overhead. This mechanism reduces execution complexity at runtime because the system does not attempt to deliver every generated candidate action immediately. Instead, it actively minimizes output competition and suppresses combinations that would degrade usability or overload the driver [53, 22].

When an action is not urgent but remains potentially useful, the system can avoid immediate execution through deferred scheduling. The waiting-queue stage stores pending tasks together with execution windows, retry counters, required cognitive reserve and suitable cognitive-load ranges. Actions are proposed for execution only when temporal and cognitive conditions are jointly appropriate; expired tasks are discarded, while some tasks may be retried with bounded backoff. This mechanism is not only a behavioral adaptation strategy but also a runtime optimization device: by postponing non-critical interventions, the system reduces immediate processing pressure in overloaded situations and distributes interaction more efficiently over time [42, 53].

More generally, this pruning-and-deferral strategy changes the meaning of performance optimization in the proposed architecture. The goal is not simply to execute more actions more quickly, but to ensure that the *right* subset of actions is executed at the right moment. In a safety-aware in-vehicle assistant, reducing the number of concurrent outputs can be as important as reducing computational latency, because excessive parallel interaction may itself become a form of performance degradation from the user's perspective. The implementation therefore optimizes both machine-level responsiveness and interaction-level suitability through the same governance logic.

5.5.4 Summary of Runtime Optimization Strategies

Overall, the runtime performance of the prototype is supported by a layered optimization strategy. Asynchronous orchestration reduces blocking time across compatible branches; cache-based state reuse prevents redundant cognitive-load computation; emergency fast paths remove non-essential stages under critical conditions; fallback logic preserves functional continuity when advanced components fail; and conflict filtering together with deferred execution limits interaction overload while keeping the pipeline operationally manageable. Rather than relying on a single low-level optimization, the implementation combines architectural, algorithmic and policy-driven mechanisms to satisfy the responsiveness requirements of an adaptive in-vehicle system.

This design is consistent with the overall philosophy of the proposed architecture. Performance is not obtained by removing bounded AI reasoning or adaptive behavior, but by embedding them inside a runtime structure that explicitly controls when computation should be parallelized, when it should be reused, when it should be simplified and when actions should be deferred. As a result, the system remains both context-sensitive and operationally tractable, providing the basis for the experimental evaluation discussed in the next chapter.

5.6 Prototype HMI Scenarios and Interface Examples

To complement the component-level implementation discussion presented in the previous sections, this final section introduces a set of representative Human–Machine Interface (HMI) scenarios derived from the implemented prototype. The purpose of these examples is not to provide a

complete industrial design study of the interface, but to illustrate how the internal orchestration logic becomes visible at interaction level through adaptive layouts, proactive recommendations and cognitively aware multimodal support.

In the proposed architecture, the HMI is not treated as a static display layer that simply renders predefined outputs. Instead, it acts as the visible endpoint of a decision pipeline in which contextual interpretation, cognitive-load estimation, agent-level reasoning, prioritization, conflict filtering and execution safeguards determine not only *what* should be communicated, but also *when*, *how* and with which degree of complexity. For this reason, interface-level examples are relevant within the implementation chapter: they provide a concrete view of the operational consequences of the orchestration mechanisms described so far.

These visual examples should therefore be interpreted as implementation-oriented illustrations of the interaction logic rather than as the outcome of a dedicated usability-validation campaign with end users. Each example reflects one or more implementation properties of the prototype, such as adaptive interface complexity, proactive assistance, multimodal comfort regulation or bounded AI-supported decision synthesis. In this sense, the HMI becomes an observable projection of the underlying coordination logic.

5.6.1 Adaptive Interface Complexity

One of the most direct effects of the proposed architecture concerns the regulation of interface complexity according to contextual and cognitive conditions. Since the orchestrator computes the current workload state and exposes it to downstream adaptive modules, the final HMI does not need to preserve a fixed density of information across all situations. Instead, interaction complexity can be selectively reduced, reorganized or enriched depending on the estimated cognitive reserve, driving criticality and current task demands.

Under favorable conditions, the interface can expose richer contextual content, such as navigation previews, recommendation cards, infotainment suggestions or secondary assistance widgets. Under higher cognitive load, the same interface can be simplified by reducing visual competition, highlighting only the most relevant elements and suppressing non-essential content [22, 53, 42]. This behavior is coherent with the execution model adopted in the prototype, where

candidate actions are filtered not only for semantic relevance but also for suitability with respect to driver state and current operational context.

At implementation level, this adaptive behavior is consistent with the role assigned to the adaptive interface layer and with the broader prioritization and conflict-resolution stages described earlier in the chapter. The resulting interface is therefore not merely personalized in a generic sense, but actively shaped by the same governance logic that regulates message timing, action selection and multimodal output coordination.

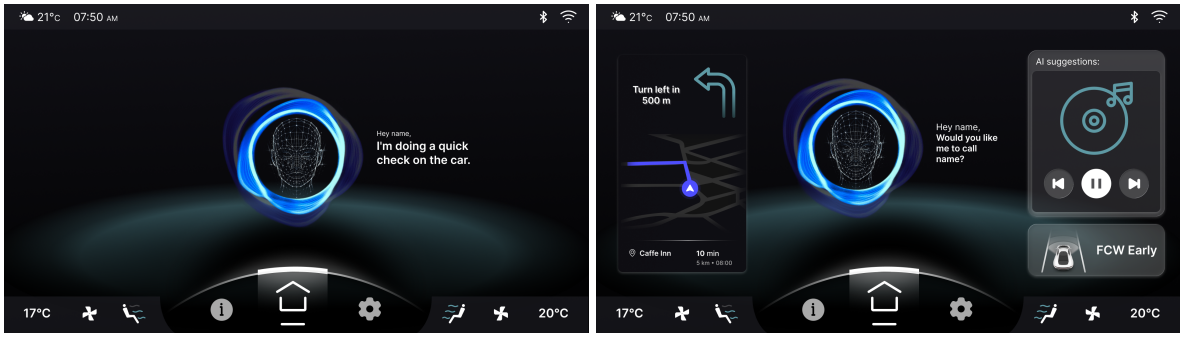

Figure 5.2: Representative interface layouts showing different levels of visual complexity and contextual information density under different operating conditions.

5.6.2 Proactive Assistance and Context-Aware Recommendations

A second relevant implementation outcome concerns proactive assistance. In the proposed system, not all outputs are triggered by explicit driver commands. After contextual interpretation and cognitive-state estimation, the orchestrator may activate proactive reasoning branches that generate candidate interventions based on environmental signals, route conditions, scheduling information, comfort needs or predicted near-term utility. These candidate actions are then evaluated, prioritized and either executed immediately, simplified or deferred.

This mechanism can be reflected at HMI level through route suggestions, calendar-aware reminders, mobility recommendations or context-sensitive decision proposals. For example, the system may suggest a navigation target, anticipate a timing constraint or present a bounded set of alternatives when it detects that an intervention could be helpful but is not yet strictly required. In such cases, the interface does not merely display system status; it externalizes the result of internal context fusion and bounded decision support [7, 42].

From an implementation perspective, these scenarios are particularly significant because they show that proactive support is not generated as unconstrained conversational behavior. Rather, it emerges from the orchestrated interaction between contextual services, LLM-supported reasoning modules, action-ranking logic and execution safeguards. The visual output therefore remains aligned with the same controlled decision framework that governs the rest of the architecture.

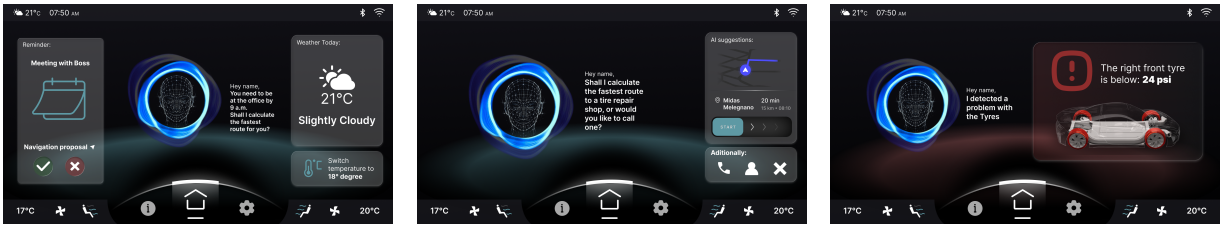

Figure 5.3: Examples of proactive and context-aware assistance, including route suggestions, calendar-informed support, maintenance-aware recommendations and safety-oriented transitions.

5.6.3 Multimodal Comfort and Ambient Adaptation

A further class of examples concerns multimodal comfort-oriented adaptation. The proposed prototype does not treat driver support as limited to warning generation or task-oriented recommendation. When the situation allows it, the system can also produce lower-criticality interventions oriented toward comfort regulation, stress mitigation and environmental coherence. These may involve coordinated suggestions related to climate, ambient lighting, audio content or relaxation-oriented interface modes.

Such scenarios are important because they highlight a central design property of the architecture: adaptation is not confined to a single communication channel. Once a candidate intervention has been generated and validated, its delivery can involve multiple synchronized modalities, provided that this remains compatible with the current cognitive and situational constraints [53, 22]. This makes the system more expressive than a purely notification-based assistant, while still preserving bounded execution through prioritization and conflict filtering.

At interface level, this multimodal behavior can appear as coordinated visual, textual and environmental recommendations that support the driver's condition without introducing unnecessary urgency. In implementation terms, these outputs reflect the cooperation between

comfort-oriented agents, adaptive presentation logic and the waiting-queue strategy used to postpone low-priority suggestions until a suitable moment for interaction becomes available.

Figure 5.4: Illustrative multimodal comfort-oriented outputs combining interface suggestions, ambient adaptation and infotainment-related support.

5.6.4 Discussion

Taken together, these examples clarify an important implementation point: the HMI of the proposed system is not a separate frontend added after the decision process, but the final manifestation of the same orchestration pipeline discussed throughout this chapter. Interface adaptation, proactive suggestions and multimodal comfort support are all downstream consequences of contextual interpretation, cognitive-state awareness, bounded LLM reasoning and execution-time governance.

For this reason, the visual examples presented here should be read as implementation evidence of architectural behavior rather than as standalone design artifacts. They make explicit how the prototype translates internal decision policies into interaction-level outputs and show that the proposed framework affects not only reasoning and action selection, but also the final form of communication experienced by the driver.

Additional interface variations and supplementary mockups are reported in Appendix A.

6. Experimental Evaluation

6.1 Evaluation Objectives

The experimental evaluation was designed to assess the proposed adaptive HMI framework from a system-level perspective. Rather than relying on a traditional user study with human participants, the present chapter examines the behavior of the implemented architecture under a controlled set of simulated driving scenarios. This choice reflects the actual nature of the developed prototype and makes it possible to evaluate the system in a reproducible way across heterogeneous contextual conditions. The goal is therefore not to validate isolated interface impressions or subjective user preferences, but to determine whether the proposed framework, considered as a complete adaptive system, is capable of generating useful, context-sensitive and operationally coherent interface adaptations under diverse driving conditions.

The first objective is comparative. The evaluation aims to determine whether the proposed architecture provides a measurable advantage over simpler adaptation strategies. In this context, the system should not be interpreted as a purely LLM-based interface generator. Instead, it is conceived as a hybrid orchestrated framework in which language-model reasoning is embedded within a broader governance structure including cognitive-state estimation, action prioritization, supervisory evaluation, conflict resolution and deferred execution mechanisms. Within this architecture, LLMs contribute to contextual interpretation and candidate generation, whereas the final system behavior remains regulated by deterministic and supervisory components. The comparative study is therefore intended to verify whether this hybrid configuration produces more useful and contextually appropriate adaptive behavior than simpler alternatives, in particular a deterministic rule-based strategy and a lower static reference.

A second objective concerns the quality of the adaptive behavior produced by the framework. More specifically, the evaluation seeks to determine whether the system is able to regulate interface complexity and information density in a manner that remains coherent with the contextual demands represented in the scenarios, including variations in urgency, traffic intensity, driver condition, weather severity and safety-critical events. In this sense, the purpose of the chapter is not to

validate the absolute correctness of cognitive-load estimation as an isolated prediction task. Rather, it is to assess whether the architecture makes effective use of contextual and cognitive-related signals in order to generate interface adaptations that are appropriate, robust and consistent with the operational situation.

The third objective is architectural. Beyond comparing the full framework against external alternatives, the evaluation also investigates the internal contribution of the main components through an ablation study. Selected modules are removed or simplified, including AI-enhanced cognitive load refinement, the quality evaluation layer and AI-based priority refinement. This part of the evaluation is essential to verify that the observed behavior of the framework does not depend on a single isolated mechanism, but instead emerges from the coordinated interaction of multiple components within the orchestration layer. In this way, the analysis highlights which modules are most relevant for preserving appropriateness, supervisory acceptance, safety control and context-sensitive adaptation.

Taken together, these objectives motivate a twofold experimental strategy. On the one hand, the proposed approach is evaluated externally through comparison with simpler adaptation paradigms. On the other hand, it is evaluated internally through ablation, in order to isolate the contribution of its main architectural components. The chapter is thus intended to provide a quantitative and reproducible basis for discussing the strengths, limitations and practical value of the proposed framework. In this sense, the evaluation does not aim to demonstrate that the system always outperforms every simpler solution under every condition, but to determine in a controlled and methodologically transparent way whether the proposed architecture offers a meaningful contribution to cognitive-aware adaptive HMI design.

6.2 Metrics

The evaluation methodology was defined to measure the proposed framework along four complementary dimensions: adaptation quality, cognitive-awareness, execution behavior and comparative significance. Since the experimental campaign is based on controlled scenarios and automated analysis rather than on a full human-subject protocol, the adopted metrics focus on system-level outputs that can be computed consistently across approaches and experimental

conditions. In particular, the metric design reflects the structure of the implemented evaluation pipeline, where baseline comparison, ablation study, metric collection and statistical analysis are executed as distinct but connected stages.

The selected metrics were not intended to capture only generic system performance. Instead, they were chosen to evaluate whether the proposed architecture produces adaptations that are appropriate for the current driving context, aligned with the estimated cognitive state of the driver, operationally feasible under execution constraints and measurably distinguishable from simpler alternatives. For this reason, the evaluation combines direct output metrics, component-specific metrics and statistical comparison indicators.

6.2.1 Appropriateness and Quality Metrics

The first group of metrics focuses on the quality and suitability of the adaptive output generated by the system. Among them, the central indicator is the *appropriateness score*, which is used to assess how well the produced interface configuration matches the needs of the current driving scenario. In the present evaluation, appropriateness is not intended as a universally validated human-centered usability measure, but as a controlled scenario-based comparison metric designed to assess whether the output of the adaptation module matches the most relevant structural expectations of the interface. More specifically, the metric was constructed by considering two key properties of the generated layout as the most important aspects of the agent output: the selected *complexity type* (e.g., minimal, standard, rich) and the *number of widgets* displayed on the screen. These two elements were considered particularly important because they directly capture the system's ability to regulate visual and informational density, which is one of the main objectives of a cognitive-aware adaptive HMI.

To support this evaluation, each experimental scenario was associated in advance with a reference complexity level and with a reference number of widgets. These values were established during the scenario design phase and were used as scenario-level ground truth for comparison. Their role was not to provide an externally validated measure of correctness, but rather to define a coherent benchmark against which different outputs could be assessed in a controlled and reproducible way. On this basis, the appropriateness score rewards outputs that correctly identify the expected complexity level, assigns an intermediate value when the generated interface remains

within an *acceptable complexity* range, and assigns no reward when the selected complexity is clearly inconsistent with the scenario requirements. In parallel, the metric also evaluates the number of displayed widgets, assigning the highest score when the output matches the expected value and progressively lower values when the deviation remains limited. In safety-critical scenarios, an additional penalty is applied whenever the generated layout is not minimal, in order to reflect the stronger need for visual simplification under demanding driving conditions.

Formally, the appropriateness score is computed as:

$$A = \min\left(1,\ \max\left(0,\ S_{\text{complexity}} + S_{\text{widgets}} - P_{\text{safety}}\right)\right)$$

where:

$$S_{\text{complexity}} = \begin{cases} 0.6 & \text{if the generated complexity matches the expected complexity,} \\ 0.3 & \text{if the generated complexity remains within an acceptable complexity range,} \\ 0 & \text{otherwise,} \end{cases}$$

$$S_{\text{widgets}} = \begin{cases} 0.4 & \text{if the number of widgets exactly matches the scenario reference,} \\ 0.3 & \text{if the deviation is equal to 1,} \\ 0.2 & \text{if the deviation is equal to 2,} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$P_{\text{safety}} = \begin{cases} 0.2 & \text{if the scenario is safety-critical and the generated layout is not minimal,} \\ 0 & \text{otherwise.} \end{cases}$$

This formulation ensures that the score remains in the interval $[0, 1]$ while explicitly reflecting the two structural properties considered central for the benchmark, namely complexity selection and widget-count calibration.

The resulting formulation combines complexity alignment and widget-count alignment into a single score. A higher weight is assigned to the correctness of the selected complexity type, while the number of widgets acts as a complementary indicator of interface density. This weighting

strategy reflects a project-specific methodological choice: the metric was deliberately designed to evaluate those output properties that were considered most relevant for the adaptive behavior expected from the system. For this reason, the appropriateness score should be interpreted as an internal comparative metric tailored to the present benchmark, rather than as a general-purpose measure of interface quality.

A second quality-related indicator is the *quality score* associated with the evaluation layer that filters candidate actions before execution. This score summarizes whether a generated action is acceptable from the perspective of safety, contextual appropriateness, clarity and executability. The internal evaluation logic explicitly checks criteria such as safety, appropriateness, clarity, tone and actionability, and associates each candidate action with both a numerical quality score and an approval decision. This makes it possible to evaluate not only whether an adaptation is generated, but also whether it satisfies the supervisory constraints imposed by the overall architecture [36].

Closely related to this quantity is the *approval rate*, defined as the proportion of candidate actions that pass the quality and safety gate without being rejected. In the ablation study, this metric is particularly useful because it highlights the effect of removing the evaluation layer or simplifying the decision pipeline. The combination of quality score and approval rate therefore provides a more informative view than a simple success/failure distinction, since it reflects both the average quality of produced outputs and the fraction of outputs considered acceptable under the established governance rules.

Finally, the broader experimental analysis also considers the *success rate*, defined as the proportion of cases in which the system is able to generate a valid adaptive output. While this metric is discussed in more detail among the execution indicators, it is also relevant here because it complements the previous quality-oriented measures: a system may successfully complete all scenarios from an operational perspective, yet still differ substantially in appropriateness, approval rate or overall output quality. Success rate is therefore interpreted together with the other evaluation metrics rather than as a stand-alone indicator.

6.2.2 Cognitive-Aware Adaptation Metrics

A second category of metrics concerns the extent to which the framework remains sensitive to the driver's cognitive condition and incorporates this information into the adaptive process. Since

cognitive-load awareness is one of the conceptual foundations of the proposed architecture, the evaluation must account for whether the generated behavior reflects workload-related constraints rather than generic context adaptation alone [48, 53, 57]. However, in the present experimental setting, this dimension is not assessed through the direct validation of cognitive-load estimation accuracy as an isolated prediction task. The synthetic scenarios include predefined reference values that are useful for driving the benchmark design, but they do not constitute an empirical ground truth comparable to measurements collected from real users. For this reason, the evaluation does not rely on a cognitive-load accuracy metric as a primary indicator of system validity.

Instead, cognitive-aware adaptation is assessed indirectly through the observable effects that cognitive-related information has on system behavior. In particular, this dimension emerges in the way the framework regulates interface complexity, constrains information density and adapts generated outputs to situations characterized by overload, urgency, distraction or safety-critical conditions. In this sense, the cognitive dimension of the evaluation is reflected less by the numerical agreement between estimated and predefined workload values, and more by the operational consequences of that estimate on the final adaptive output. The purpose of this metric group is therefore to verify whether cognitive-state awareness contributes meaningfully to the appropriateness and contextual coherence of the generated adaptations.

Within this perspective, a complementary indicator is represented by *reasoning quality*, which is used whenever the evaluated configuration produces an explicit textual justification for its adaptive choice. This metric does not attempt to measure whether the reasoning is objectively correct in a formal or human-validated sense. Rather, it assesses whether the explanation produced by the system addresses the main dimensions that are considered relevant in the benchmark, namely the cognitive condition of the driver, the rationale for the selected interface complexity, the presence of safety-related considerations and the ability to express these elements in a concise but sufficiently informative way. Reasoning quality is therefore interpreted as an auxiliary indicator of explainability and contextual adequacy, especially in the hybrid configuration where language-model reasoning contributes to layout generation and action selection [37, 21].

For this reason, the cognitive-aware metrics adopted in the chapter should not be interpreted as psychometric or predictive validation measures. Their role is instead to capture whether the proposed architecture uses cognitive-related signals in a way that materially influences the

adaptation process and supports outputs that remain coherent with the demands of the surrounding driving situation. In the context of this thesis, cognitive awareness is therefore evaluated at the level of adaptive behavior and generated justification, rather than through the direct accuracy of an isolated internal estimator.

6.2.3 Latency and Execution Metrics

The third group of metrics addresses execution behavior and real-time feasibility. In automotive HMI adaptation, correctness alone is not sufficient: the system must also produce its decisions within timing conditions compatible with the surrounding context [24, 22]. For this reason, the evaluation records *response time in milliseconds* for each scenario and adaptive approach. These latency values are then aggregated through descriptive statistics such as mean, median, 95th percentile and 99th percentile.

Latency is especially relevant in the comparison between deterministic and LLM-supported processing. In the present chapter, the main quantitative latency analysis focuses on the two adaptive strategies considered in the baseline comparison, namely the rule-based and hybrid approaches. The static configuration is retained only as a descriptive lower reference and is not included in the main timing comparison. As a consequence, latency is not treated merely as an implementation detail, but as a central evaluation dimension that informs the discussion on practical deployability and on the trade-off between richer contextual reasoning and strict real-time constraints.

A complementary execution metric is the *success rate*, defined as the proportion of cases in which the system is able to generate a valid layout or executable adaptive output. In the metric collection logic, this is derived from whether the expected adaptive artifact is successfully produced. This indicator is useful because it distinguishes between high-quality outputs and simple operational completion: a system may complete all scenarios, but still differ substantially in appropriateness, cognitive fit or timing. Success rate is therefore interpreted together with the other quality and latency measures rather than in isolation.

6.2.4 Comparative and Statistical Metrics

The last category of metrics supports the comparative interpretation of the results. Because the goal of the chapter is not merely to present isolated descriptive values, the evaluation includes statistical indicators to determine whether the observed differences between approaches are substantial or may instead be attributable to variation across scenarios. In the revised baseline comparison, the statistical analysis is centered on the appropriateness scores obtained by the two adaptive approaches retained in the main experiment, namely the rule-based and hybrid configurations.

More specifically, the comparative analysis reports mean, standard deviation, sample size and confidence intervals for each group, together with p-values and Cohen's d for the pairwise difference. These measures make it possible to distinguish between a merely visible difference in average score and a difference that is also statistically and practically meaningful [8]. Since the main baseline comparison involves only two adaptive approaches, the statistical procedure is formulated as a direct pairwise comparison rather than as a multi-group variance analysis.

In the ablation study, the comparative logic is different. Rather than comparing alternative adaptation paradigms, the analysis examines the effect of selectively removing specific architectural components. For this reason, the ablation results are interpreted through a common set of output-level indicators, namely appropriateness, approval rate and success rate, complemented when necessary by a component-specific auxiliary indicator. In particular, safety risk is introduced for the configuration without quality evaluation in order to capture the increased likelihood that contextually inappropriate or insufficiently conservative outputs may pass through the system when the supervisory filter is removed.

This design choice is methodologically important because it avoids treating heterogeneous quantities as if they were directly interchangeable. Instead of comparing cognitive accuracy, flexibility, safety filtering and prioritization quality under a single undifferentiated score, the revised framework evaluates all reduced configurations through a harmonized set of output-level metrics and uses auxiliary indicators only where they are functionally justified. In this way, the statistical and comparative analysis remains aligned with the actual role of the removed components and with the broader system-level objectives of the chapter.

6.3 Experimental Setup

The experimental setup was designed to evaluate the proposed framework under controlled and reproducible conditions. Since the validation was conducted through a system-level experimental pipeline rather than through an in-vehicle deployment or a full human-subject study, the setup focuses on the execution environment of the implemented software, the structure of the generated scenarios, the definition of the compared approaches and the sequence of processing steps used to obtain the final results. In this way, the evaluation remains closely aligned with the actual implementation of the framework and with the experimental artifacts produced by the codebase.

6.3.1 Hardware and Execution Environment

The evaluation was executed as an offline software-based experiment in which the adaptive HMI framework was tested on a collection of generated driving scenarios. The available experimental materials do not describe a dedicated physical driving simulator or a specialized hardware bench as the primary validation platform. Instead, the experimental setup corresponds to a computational execution environment in which scenarios are processed sequentially through the evaluation pipeline and the resulting outputs are stored as structured analysis artifacts. For this reason, the present chapter characterizes the setup mainly in terms of execution workflow and software orchestration rather than in terms of cockpit instrumentation or simulator hardware.

From an architectural point of view, the evaluated system operates on structured contextual representations derived from multiple vehicle-related and driver-related inputs, including DMS indicators, VCU variables and contextual state descriptors. These data sources are abstracted into internal representations that can be consumed by the adaptive components without dependence on a specific physical sensor stack during the experimental phase. This design choice makes the evaluation reproducible and allows the same scenarios to be replayed across all compared approaches under identical initial conditions.

6.3.2 Software Stack

The experimental setup is built around a Python-based evaluation framework composed of dedicated modules for scenario generation, comparative testing, ablation analysis, statistical processing, metric collection and visualization. The full execution flow is coordinated by the main evaluation script, which invokes each experimental phase in sequence and exports the corresponding results into persistent files. More specifically, the implemented pipeline includes the following main elements: a `ScenarioGenerator` for the construction of test scenarios, an `AdaptiveUIExperiment` for baseline comparison, an `AblationStudy` module for component removal tests, a `StatisticalAnalyzer` for inferential analysis, a `MetricsCollector` for structured aggregation of metrics and an `ExperimentVisualizer` for graph generation.

At system level, these experimental modules are connected to the adaptive architecture described in the previous chapter. In particular, the evaluation interacts with core services such as cognitive load estimation, priority computation, quality and safety evaluation, conflict filtering and adaptive UI generation. The software setup is therefore not an isolated benchmark harness detached from the implemented architecture, but an experimental layer built around the actual orchestration logic of the system. This is important because it ensures that the reported results reflect the behavior of the proposed framework and not that of an artificial proxy model.

6.3.3 Scenario Set and Test Case Design

The scenario set used in the evaluation was generated programmatically in order to provide both coverage and repeatability. The full evaluation script constructs the experimental dataset through three complementary steps: generation of a basic scenario set, generation of edge cases and generation of a larger realistic distribution of scenarios. The resulting scenarios are then exported to a structured JSON file and reused across the subsequent experimental stages. This design makes it possible to expose the evaluated approaches to both ordinary and critical conditions while keeping the input space fixed during the comparison.

The generated test cases include heterogeneous contextual factors relevant to adaptive HMI behavior. Among the represented variables are cognitive load level, vehicle speed, traffic condition, weather severity, drowsiness, distraction, emotional state, battery level, urgency score, safety

score and expected interaction complexity. The scenario set also includes explicitly critical cases, such as drowsiness detection, high-distraction conditions, low-battery situations and extreme high-load situations characterized by heavy traffic, severe weather and elevated safety risk. This variety is essential because the proposed framework is intended to regulate interaction complexity in relation to both cognitive demand and operational safety.

The scenarios are not used merely as generic test inputs. Each one functions as a compact representation of a driving context in which the framework must decide whether to simplify the interface, preserve a standard interaction layout or support richer adaptive behavior. In this sense, the scenario design directly supports the main goals of the evaluation, namely the assessment of contextual appropriateness, cognitive coherence and robustness under heterogeneous operating conditions.

6.3.4 Compared Approaches

The comparative setup considers three adaptation paradigms. The first is a *static* approach, which represents a non-adaptive or minimally adaptive interface configuration and is introduced primarily as a descriptive lower reference. The second is a *rule-based* approach, in which interface adaptations are produced through predefined deterministic logic. The third is the proposed *hybrid orchestrated approach with LLM support*, which integrates language-model reasoning into a broader architecture governed by cognitive load estimation, prioritization mechanisms and quality and safety supervision.

Although the static configuration is useful for conceptually representing the absence of adaptive behavior, it is not included in the main quantitative comparison reported in the following analyses. The reason is methodological: since the static interface remains unchanged across scenarios, its performance is inherently tied to accidental alignment with a subset of conditions rather than to an actual adaptation capability. As a consequence, it may appear fully appropriate in scenarios that happen to match its fixed configuration, while performing very poorly in all others. This behavior makes the resulting scores strongly dependent on the scenario distribution and less informative for the purposes of the present evaluation.

For this reason, the main comparative analysis focuses on the two genuinely adaptive strategies, namely the deterministic rule-based approach and the proposed hybrid orchestrated framework.

This choice makes it possible to assess whether the introduction of LLM-supported reasoning and orchestration mechanisms provides a meaningful advantage over a simpler adaptive baseline, without overemphasizing the artificially polarized behavior of a non-adaptive reference. The static approach is therefore retained only as a conceptual point of reference, while the quantitative results and graphical comparisons concentrate on the two approaches that are capable of modifying their behavior across scenarios.

6.4 Evaluation Procedure

The evaluation procedure was defined to ensure that all compared approaches were tested under identical conditions and that the resulting outputs could be analyzed in a systematic and reproducible way. Since the experimental campaign was not organized as a traditional participant-based study, the procedure does not follow the structure of briefing, task execution and post-task questionnaires. Instead, it is based on a software-controlled workflow in which the same scenario set is processed through multiple evaluation stages, each targeting a different aspect of the proposed framework. This choice is consistent with the overall objective of the chapter, namely the validation of the implemented architecture through controlled comparative analysis, component-level inspection and quantitative assessment.

At a general level, the procedure consists of four main phases. First, a structured set of scenarios is generated or loaded in order to define the evaluation input space. Second, the compared adaptation strategies are executed on the same scenarios to obtain baseline comparison results. Third, the proposed framework is subjected to an ablation study in which selected internal modules are removed or simplified. Finally, the outputs produced by these stages are aggregated, analyzed statistically and transformed into summary artifacts and plots. This organization makes it possible to separate comparative evaluation, architectural inspection and quantitative validation while preserving a unified experimental workflow.

6.4.1 Scenario Generation

The first phase of the procedure concerns the construction of the scenario set used throughout the evaluation. Rather than relying on manually selected examples only, the implemented workflow

generates scenarios programmatically through a combination of basic templates, edge cases and more realistic distributions of contextual conditions. This choice ensures both variability and repeatability: the test set contains heterogeneous situations, but once generated it can be reused unchanged across all experimental stages.

The generated scenarios include a wide range of contextual variables relevant to adaptive HMI behavior. These variables describe traffic intensity, speed, weather conditions, cognitive load, distraction, drowsiness, urgency, battery level, safety score and interaction complexity. In addition to ordinary driving situations, the scenario generator explicitly introduces critical cases intended to stress the system under demanding conditions. These include, for example, high cognitive load states, severe weather, low-battery conditions and situations characterized by increased safety risk. As a result, the scenario generation phase is not only a preparatory step, but a methodological element that defines the coverage and realism of the entire evaluation.

Once generated, the scenarios are stored in a structured format and become the common input reference for all subsequent analyses. This guarantees that any observed difference between approaches is attributable to the adaptation logic itself rather than to variation in the tested situations.

6.4.2 Baseline Comparison Procedure

After the scenario set has been defined, the evaluation proceeds with the baseline comparison stage. In this phase, each scenario is processed by the adaptive approaches considered in the main comparative study, namely the rule-based baseline and the proposed hybrid orchestrated framework with LLM support. The static configuration is retained only as a descriptive reference representing the absence of adaptation, but it is not included in the main quantitative comparison because its fixed behavior does not provide an informative benchmark for evaluating adaptive performance across heterogeneous scenarios.

For each scenario, the outputs generated by the two adaptive approaches are evaluated according to the same metric framework, so that the comparison is performed under identical contextual conditions. The purpose of this stage is not merely to verify that the proposed system is capable of producing an output, but to determine whether the hybrid orchestrated architecture leads to more appropriate, context-sensitive and qualitatively robust interface adaptations than a

simpler deterministic alternative.

For this reason, the baseline comparison records not only the final adaptation outcome, but also the main indicators that are directly available in the comparative pipeline, including appropriateness as the primary evaluation metric, response time as an execution-related indicator and complementary measures such as reasoning quality, when available. This phase constitutes the core comparative experiment of the chapter, since it provides the main empirical basis for assessing whether the integration of supervised language-model reasoning within the orchestration layer yields a tangible advantage over a purely deterministic adaptive strategy.

6.4.3 Ablation Study Procedure

The third phase of the evaluation consists of an ablation study designed to isolate the contribution of the main architectural components. While the baseline comparison examines whether the proposed framework provides an advantage over a simpler adaptive alternative, the ablation study investigates the internal structure of the system by selectively removing or simplifying specific modules and then re-evaluating the resulting configurations [32]. The purpose of this phase is therefore not to compare different adaptation paradigms, but to determine which components materially contribute to the final behavior of the proposed architecture.

In the revised ablation setting, the complete system is compared with three reduced variants: a configuration without AI-enhanced cognitive load refinement, a configuration without the quality evaluation layer and a configuration without AI-based priority refinement. These variants were retained because they correspond to actual architectural mechanisms whose removal produces interpretable changes in the behavior of the system. By contrast, the previously considered template-based alternative is not included in the main quantitative ablation analysis, since in the implemented pipeline it acts primarily as an illustrative simplified generation mode rather than as a fully comparable adaptive configuration.

To preserve methodological consistency, all ablated variants are evaluated through a common set of output-level indicators. In particular, the analysis focuses on appropriateness, approval rate and success rate, which together make it possible to assess whether the system still generates outputs, whether those outputs remain acceptable under supervisory constraints and whether the resulting adaptations stay aligned with the expected scenario-level configuration. This choice

avoids comparing heterogeneous quantities as if they were directly interchangeable and provides a more coherent basis for architectural interpretation across the different reduced variants.

When necessary, the common metric set is complemented by a component-specific auxiliary indicator. In particular, the configuration without quality evaluation is also examined through an internal safety risk indicator, introduced to estimate the extent to which contextually inappropriate or insufficiently conservative outputs may pass through the system when the supervisory validation layer is removed. In the present work, this indicator should be interpreted as a rule-based internal proxy rather than as an externally validated safety measure. More specifically, it reflects the presence of incoherences such as the failure to suppress or strongly constrain non-essential outputs in safety-critical situations.

The logic of the ablation is therefore straightforward. When AI-enhanced cognitive refinement is removed, the system is expected to preserve operational continuity while becoming less precise in regulating interface complexity and information density. When the quality evaluation layer is removed, the system may still produce outputs successfully, but it becomes more exposed to the passage of unsafe or weakly justified actions. When AI-based priority refinement is removed, candidate actions can still be generated, but their ordering and final selection become less context-sensitive. Taken together, these controlled reductions make it possible to determine whether the effectiveness of the proposed framework emerges from the coordinated interaction of multiple modules rather than from any single mechanism considered in isolation.

6.4.4 Statistical Analysis Procedure

Once the comparative and ablation results have been collected, the evaluation proceeds with statistical analysis. The goal of this phase is to determine whether the observed differences between approaches are sufficiently stable to support a rigorous interpretation rather than a purely descriptive one. The statistical procedure is mainly applied to the comparative results obtained from the baseline evaluation, where the adaptation strategies are compared on the basis of their appropriateness scores.

The implemented analysis combines descriptive and inferential statistics. For each evaluated approach, the results are summarized through measures such as mean, standard deviation, sample size and confidence intervals, and are further examined through direct between-group hypothesis

testing and effect size estimation [8]. Since the main baseline comparison involves only two adaptive approaches, the statistical procedure is formulated as a pairwise comparison rather than as a multi-group variance analysis.

This stage is essential because the thesis aims to support its claims with measurable evidence rather than with isolated examples. The statistical analysis therefore transforms raw experimental outputs into interpretable comparative findings and provides the quantitative foundation for the discussion presented in the final sections of the chapter.

6.4.5 Artifact Generation and Result Consolidation

The final phase of the procedure concerns the consolidation of the produced outputs into reusable analysis artifacts. Once scenario-level results, ablation outputs and statistical summaries have been obtained, the evaluation pipeline exports both raw and aggregated data into persistent files. These artifacts include structured JSON summaries, raw metric collections, textual statistical reports and graphical visualizations.

This phase has two purposes. First, it improves reproducibility by preserving the full experimental trace in a structured form. Second, it facilitates the interpretation of the results by transforming numerical outputs into tables, summaries and plots that can be discussed more clearly in the thesis. In this sense, the evaluation procedure does not terminate with the execution of the compared systems, but extends to the organization of the generated evidence into a form suitable for scientific reporting.

Overall, the adopted procedure provides a coherent and reproducible methodology for validating the proposed framework. By combining controlled scenario generation, multi-approach comparison, component-level ablation, statistical assessment and artifact generation, the evaluation is able to examine the system from complementary perspectives and to support the discussion of its strengths and limitations with structured evidence.

6.5 Data Collection and Artifacts Analysis

The evaluation process was designed not only to execute the compared approaches, but also to preserve the resulting evidence in a structured and reusable form. For this reason, the experimental

pipeline produces a set of persistent artifacts that document the different stages of the analysis, from scenario generation to comparative evaluation, ablation testing, statistical processing and graphical reporting. This design supports both reproducibility and traceability, since the final discussion of the results can be grounded in exported data rather than in transient execution outputs alone.

The first category of collected artifacts concerns the scenario space used in the experiments. Once the scenario generation phase is completed, the resulting set of test conditions is exported to a dedicated JSON file. This artifact acts as the common reference input for the entire evaluation, ensuring that all compared approaches and ablated variants are assessed on the same contextual configurations. In methodological terms, this is important because it guarantees consistency across runs and allows the experimental evidence to be interpreted with respect to a fixed and documented input set.

A second group of artifacts stores the outputs of the two main experimental branches. The baseline comparison phase produces a structured file containing the scenario-level and aggregate results of the rule-based and LLM-supported approaches. In parallel, the ablation phase exports a dedicated file in which the performance of the complete architecture is reported together with the results obtained after removing or simplifying selected components. These two artifacts form the empirical core of the evaluation, since they preserve the comparative evidence used later in the results and discussion sections.

The third category concerns the statistical and metric aggregation outputs. After the comparative results have been produced, the pipeline extracts the relevant appropriateness scores and processes them through a statistical analysis stage. The corresponding outputs are exported both as a machine-readable JSON file and as a textual report, so that inferential results can be inspected in different forms. In addition, the metrics collector stores experiment-level information in two complementary formats: a summary file containing aggregated descriptors such as mean, standard deviation, minimum, maximum and count for each metric, and a raw-data file preserving the individual metric values together with timestamps and associated metadata. This distinction is useful because the summary supports concise reporting, whereas the raw collection preserves the full experimental trace.

The generated artifacts are not limited to numerical files. The visualization stage transforms

the collected results into graphical and tabular outputs that support both interpretation and presentation. In the revised reporting structure, the baseline branch includes an appropriateness comparison plot across the two adaptive approaches, together with descriptive and inferential summary tables. The evaluation workflow also supports latency-oriented reporting and a revised ablation summary based on harmonized output-level indicators. These artifacts are particularly useful in the context of the thesis because they make the comparative trends more immediately interpretable and complement the textual discussion with a more direct representation of the observed differences.

Overall, the data collection strategy was conceived as an integral part of the evaluation methodology rather than as a secondary implementation detail. By exporting scenarios, comparative outputs, ablation results, statistical summaries, metric collections and visual artifacts, the experimental pipeline produces a coherent set of evidence that supports both reproducibility and scientific reporting. The following section builds on these artifacts to present and interpret the obtained results in a structured way.

6.6 Results

This section presents the main findings obtained from the comparative evaluation, the statistical analysis and the ablation study. The reported results should be interpreted in light of the experimental design introduced in the previous sections: the evaluation does not aim to validate the framework through subjective user perception, but to assess its behavior under controlled scenario-based conditions by comparing alternative adaptation strategies and by isolating the contribution of individual architectural components. For this reason, the discussion focuses on appropriateness, cognitive-aware adaptation, execution behavior and component-level impact.

6.6.1 Descriptive Statistics

Under the revised appropriateness formulation adopted in this work, the baseline comparison shows a clear difference between the two adaptive approaches considered in the main experiment. The deterministic rule-based strategy obtained a mean appropriateness score of 0.771, whereas the proposed hybrid orchestrated framework with LLM support reached a higher average value

of 0.886. This result indicates that, within the considered benchmark, the hybrid configuration achieved a stronger overall alignment with the scenario-level adaptation targets than the simpler deterministic baseline.

The observed variability is also informative. The rule-based approach exhibited a standard deviation of 0.095, while the hybrid configuration showed a slightly lower standard deviation of 0.090. This suggests that the advantage of the hybrid system was not produced by a small number of isolated peak cases, but rather by a more consistent tendency to remain closer to the expected interface configuration across the evaluated scenarios. In particular, the hybrid approach more frequently matched the expected complexity level or remained within an acceptable range while also producing a widget configuration closer to the scenario target.

From an interpretive perspective, this pattern is coherent with the architectural role of the two approaches. The rule-based strategy remains competitive in more regular and easily codifiable situations, where deterministic mappings between context and interface adaptation are sufficient to produce an adequate result. However, the hybrid framework appears better suited to scenarios in which adaptation requires a finer balance between complexity type and information density. In such cases, the integration of language-model reasoning within the orchestration layer supports outputs that more closely follow the expected scenario-level configuration.

Overall, these descriptive results suggest that, within the revised appropriateness benchmark adopted in this thesis, the proposed hybrid architecture yields on average a more appropriate adaptive behavior than the deterministic baseline. This should be interpreted as comparative evidence internal to the present evaluation framework, not as a universal claim of superiority across all driving conditions or all dimensions of HMI performance.

Figure 6.1 summarizes the score distributions for the two adaptive approaches and offers a compact visual representation of their relative spread.

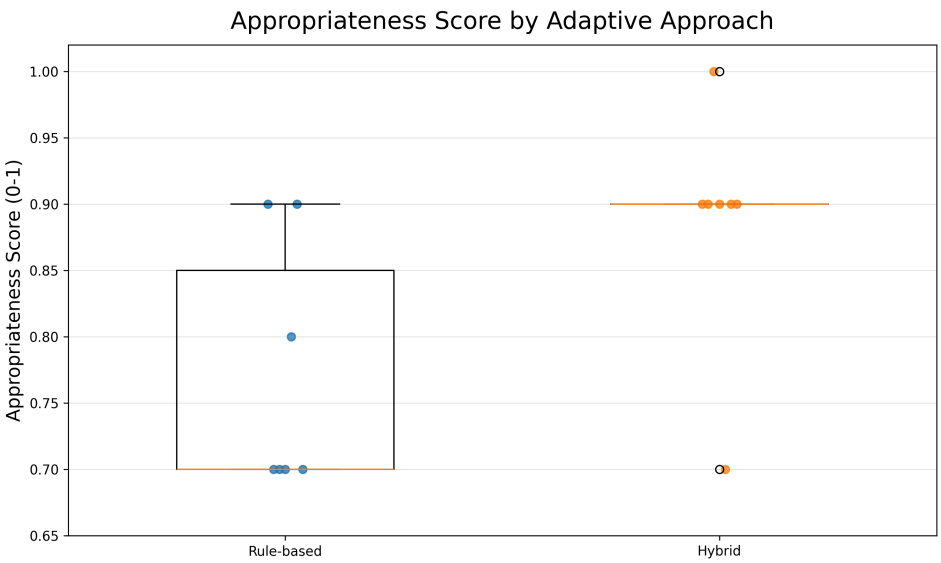

Figure 6.1: Distribution of appropriateness scores for the rule-based and hybrid adaptive approaches.

Table 6.1: Descriptive statistics for the main baseline comparison.

Approach	**Mean**	**Std**	**Median**	**Min**	**Max**	**n**
Rule-based	0.771	0.095	0.700	0.700	0.900	7
Hybrid	**0.886**	0.090	0.900	0.700	1.000	7

At scenario level, the rule-based approach performed best in relatively regular conditions such as `highway_cruise_optimal` and `parking_maneuver`, where it reached appropriateness values of 0.9, and remained intermediate in `low_battery_anxiety`, with a score of 0.8. However, its behavior became less aligned with the scenario targets in more demanding situations such as `urban_heavy_traffic`, `drowsy_driver_critical`, `angry_driver_stress` and `collision_warning`, where the appropriateness score dropped to 0.7. The hybrid approach showed a more favorable overall profile. It reached 1.0 in `drowsy_driver_critical`, maintained values of 0.9 in scenarios such as `highway_cruise_optimal`, `urban_heavy_traffic`, `angry_driver_stress`, `parking_maneuver` and `collision_warning`, and decreased to 0.7 only in `low_battery_anxiety`. This pattern suggests that the proposed architecture is particularly effective in high-demand or safety-relevant situations, while still remaining sensitive to

contextual framing in scenarios where the expected balance between complexity and informational density is less straightforward.

The graphical distribution confirms the numerical evidence. Although both approaches display a moderate degree of variability across scenarios, the hybrid configuration is consistently shifted toward higher appropriateness values. This visual pattern is coherent with the descriptive statistics reported above and reinforces the interpretation of the hybrid framework as the strongest adaptive solution in the baseline comparison.

6.6.2 Comparative Analysis of the Evaluated Approaches

The descriptive comparison is further supported by inferential statistical analysis. Since the main baseline experiment involves only two adaptive approaches, the statistical procedure was formulated as a direct pairwise comparison between the rule-based configuration and the proposed hybrid framework. The resulting test revealed a statistically significant difference between the two groups, with $t(12) = 2.309$ and $p = 0.040$. This result suggests that the observed improvement of the hybrid approach is unlikely to be explained solely by random variation across the evaluated scenarios.

The magnitude of the difference is reinforced by the corresponding effect size. The comparison yielded a Cohen's d equal to 1.234, which corresponds to a large effect according to standard interpretive conventions [8]. From the perspective of the benchmark adopted in this chapter, this indicates that the advantage of the hybrid framework is not only statistically detectable, but also substantial in practical terms.

Additional support is provided by the confidence intervals associated with the two means. The rule-based approach achieved a mean appropriateness of 0.771 with a 95% confidence interval ranging from 0.683 to 0.859, whereas the hybrid framework reached a mean of 0.886 with a 95% confidence interval between 0.803 and 0.969. Although the intervals are not completely distant from one another, the overall comparison remains consistently in favor of the hybrid configuration, which maintains the higher mean value and the more favorable distribution of scores.

Table 6.2: Inferential statistics for the comparison between the rule-based and hybrid approaches.

Comparison	t	dof	p-value	Cohen's d	Interpretation
Rule-based vs Hybrid	2.309	12	0.040	1.234	Large effect

Taken together, these findings indicate that the integration of supervised language-model reasoning within the orchestration layer yields a measurable advantage over a purely deterministic adaptive strategy. This advantage should not be interpreted as evidence that the hybrid system is uniformly optimal in every individual scenario. Rather, it indicates that, under the revised appropriateness framework adopted in this work, the proposed architecture is able to generate interface adaptations that are on average more consistent with the expected balance between layout complexity and information density.

6.6.3 Latency and Real-Time Considerations

The execution results reveal a clear trade-off between adaptive richness and timing behavior. Within the main comparative analysis, the rule-based strategy is associated with low and stable response times, operating at a constant latency of 50 ms across the evaluated scenarios. By contrast, the hybrid LLM-supported configuration exhibits substantially higher execution times, with observed values ranging approximately from 5641 ms to 7496 ms in the collected outputs. Representative values include about 7264 ms for `highway_cruise_optimal`, 5944 ms for `drowsy_driver_critical`, 6272 ms for `low_battery_anxiety` and 6652 ms for `collision_warning`. These results indicate that the proposed hybrid architecture introduces a significant computational overhead with respect to deterministic adaptation logic.

This finding is highly relevant for the interpretation of the framework. From the perspective of adaptive quality, the hybrid configuration achieves better average appropriateness under the revised evaluation scheme, indicating a stronger alignment with the expected balance between interface complexity and information density. However, this advantage is obtained at the cost of markedly higher latency. The rule-based baseline, by contrast, remains considerably more efficient in timing terms, which makes it more naturally compatible with situations requiring predictable and near-immediate response behavior. The comparison therefore does not point to a

single universally dominant strategy, but rather to a structural trade-off between richer contextual adaptation and stricter real-time responsiveness.

From the perspective of practical deployability, this trade-off has important implications. In safety-critical automotive conditions, correctness alone is not sufficient if the corresponding decision is generated too late to be operationally useful [24, 22]. For this reason, the current results suggest that the LLM-supported branch should not be interpreted as a universally applicable low-latency decision layer. Instead, it is better understood as a higher-level adaptive mechanism that can improve contextual reasoning and layout selection when timing constraints remain compatible with its computational cost. In practical terms, this reinforces the architectural need for fallback logic, deterministic shortcuts and selective activation of AI-enhanced processing only when the surrounding context allows it.

These latency findings therefore complement, rather than contradict, the appropriateness results. The hybrid framework appears stronger in terms of adaptive quality, whereas the rule-based strategy remains clearly superior in responsiveness and timing stability. Taken together, these results support the overall architectural rationale of the thesis: deterministic logic remains essential for strict timing guarantees, while AI-enhanced reasoning becomes most valuable when flexibility, contextual synthesis and more refined adaptation behavior justify its additional execution cost.

6.6.4 Component Impact: Ablation Study

The ablation study provides a complementary perspective by examining the internal contribution of the main architectural components. When all modules are active, the complete system achieves a success rate of 1.0, an average quality score of 0.816 and an approval rate of 0.810. These values indicate that the full architecture is able to generate outputs consistently while keeping most of them above the supervisory acceptance threshold imposed by the evaluation layer. In the revised interpretation adopted in this chapter, the full configuration also acts as the reference point for output appropriateness, since it corresponds to the strongest integrated version of the proposed framework.

The first reduced variant removes AI-enhanced cognitive load refinement. In this case, the system is assumed to preserve operational continuity and therefore success rate remains unchanged,

but the quality of the final adaptation decreases because the regulation of layout complexity and information density becomes less sensitive to the driver's estimated condition. For this reason, the effect of the ablation is interpreted primarily through a reduction in appropriateness rather than through a pseudo-accuracy measure of cognitive estimation. In practical terms, the architectural meaning of this result is that cognitive refinement does not merely produce an internal score: it materially affects the system's ability to select an interface configuration that remains coherent with the scenario-level balance between expected complexity and widget density. In the revised harmonized analysis, this configuration reaches an appropriateness score of 0.786 and an approval rate of 0.778, while preserving a success rate of 1.0. This confirms that the removal of cognitive refinement does not interrupt the operational pipeline, but reduces the precision with which the final adaptation is calibrated to the scenario context. This behavior is coherent with the architectural role of cognitive estimation in the implemented pipeline, where workload is not treated as a passive annotation but as a control variable that influences prioritization, adaptive interface regulation and deferred execution.

A different degradation emerges when the quality evaluation layer is removed. In this case, the system can still generate outputs and may even appear less restrictive from an approval perspective, precisely because the supervisory filter is no longer active. However, this does not correspond to an improvement in system quality. On the contrary, the exported analysis reports a safety risk increase of 0.0714, indicating that the absence of the evaluation layer makes the framework more exposed to contextually inappropriate or insufficiently conservative outputs. In the present evaluation, this safety risk indicator is interpreted as an internal rule-based proxy: it captures situations in which, under safety-critical conditions, outputs that should have been filtered, minimized or suppressed are allowed to pass through the pipeline. This supports the architectural role attributed to the quality evaluation module in the implementation chapter: its function is not limited to assigning cosmetic scores, but includes the active prevention of unsafe or operationally incoherent actions [36]. This interpretation is also coherent with the harmonized output-level indicators, where the configuration maintains an appropriateness score of 0.857 and a success rate of 1.0, but reaches an approval rate of 1.0 only because the supervisory validation mechanism is no longer active.

The removal of AI-based priority refinement also produces a measurable degradation. In

the revised harmonized analysis, this reduced variant reaches an appropriateness score of 0.757, corresponding to a decrease of 0.129 with respect to the full system configuration. This result shows that the final adaptive behavior becomes less well aligned with the scenario requirements when the prioritization stage is simplified. The finding suggests that candidate generation alone is not sufficient to ensure strong system behavior. The ordering and selection of candidate actions contributes materially to the quality of the final output, because it determines which adaptation is ultimately exposed to the user under the current contextual conditions. In quantitative terms, the configuration also reaches an approval rate of 0.762 while preserving a success rate of 1.0. This confirms that the main effect of removing AI-based prioritization is not operational failure, but a weaker alignment between the final selected adaptation and the contextual demands of the scenario.

Notably, this reduced configuration also falls slightly below the rule-based baseline reported in the main comparative study, whose mean appropriateness is 0.771. Although the difference is limited, it suggests that removing AI-based priority refinement is sufficient to eliminate part of the comparative advantage originally provided by the full hybrid architecture. In other words, the system does not lose its ability to generate outputs, but it loses part of the context-sensitive ranking capacity that allows the final adaptation to remain better aligned with the scenario requirements than a simpler deterministic strategy.

The priority engine should therefore be interpreted as a core coordination mechanism rather than as a secondary implementation detail.

For methodological reasons, the template-based alternative is not included in the main quantitative interpretation of the ablation results. In the implemented pipeline, this variant mainly acts as a simplified illustrative generation mode and its reported flexibility values are not directly comparable to the common output-level metrics adopted for the other reduced configurations. It can therefore be mentioned as a qualitative reference for the loss of expressive flexibility and explicit reasoning, but it is not used here as a primary source of quantitative evidence.

Table 6.3 summarizes the harmonized output-level indicators adopted for the revised ablation interpretation. In line with the metric framework introduced earlier in the chapter, the comparison is centered on appropriateness, approval rate and success rate, while safety risk is reported only for the configuration without quality evaluation, where it captures a functionally relevant degradation

that would not be visible through the common indicators alone.

Table 6.3: Revised ablation study results across the evaluated configurations.

Configuration	Appropriateness	Approval Rate	Success Rate	Safety Risk
Full System	**0.886**	0.810	1.000	0.000
No AI Cognitive Load	0.786	0.778	1.000	0.014
No Quality Evaluation	0.857	1.000[†]	1.000	**0.071**
No Priority AI	0.757	0.762	1.000	0.029

[†]The approval rate of the *No Quality Evaluation* configuration is structurally inflated because the supervisory validation layer is removed. It should therefore not be interpreted as an improvement in output quality.

Figure 6.2 complements Table 6.3 by providing a compact visual summary of the harmonized ablation results. In particular, it highlights the relative decrease in appropriateness and approval rate produced by the removal of selected components, while also showing that safety-related degradation becomes especially visible when the quality evaluation layer is removed.

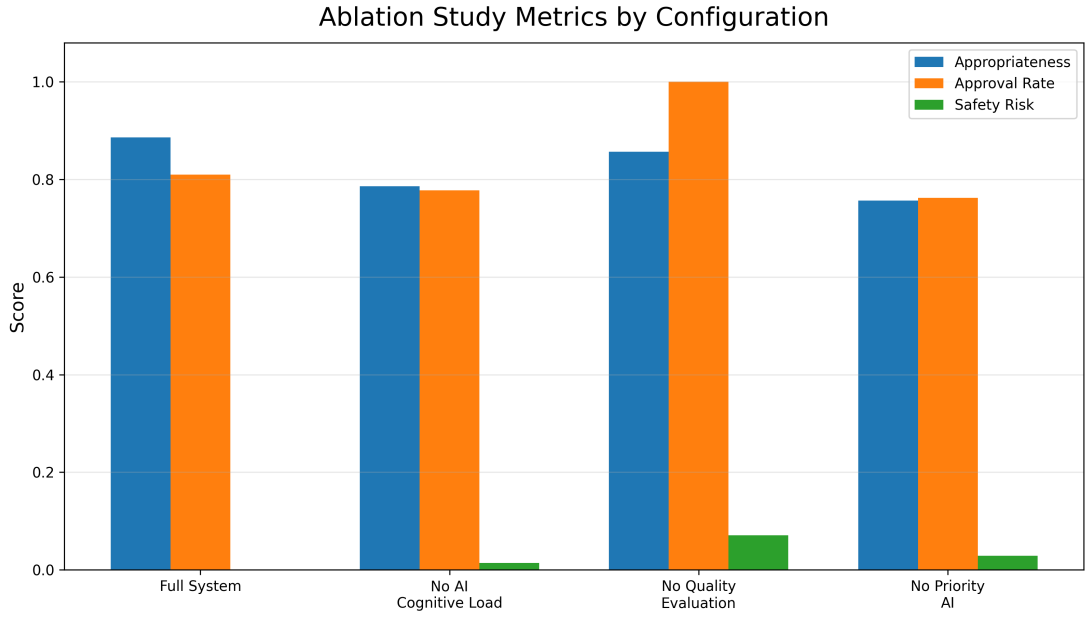

Figure 6.2: Harmonized ablation-study metrics across the evaluated configurations.

The visual comparison reinforces the tabular interpretation. The full system remains the

strongest configuration in terms of overall balance, while the removal of AI-enhanced cognitive refinement and AI-based priority computation leads to a visible decrease in appropriateness. By contrast, the absence of the quality evaluation layer does not primarily reduce output generation, but increases the exposure to unsafe or insufficiently conservative actions, as reflected by the rise in safety risk.

Overall, the revised ablation findings confirm that the proposed architecture should be interpreted as a coordinated system rather than as a single-model solution. Cognitive refinement, supervisory quality evaluation and AI-based priority computation all contribute to final behavior, but they do so in different ways. Some modules mainly preserve appropriateness and cognitive coherence, whereas others primarily protect safety and control the final acceptability of generated actions. The ablation study is therefore important not because it proves that every component contributes equally, but because it shows that the strongest system behavior emerges from the interaction of these mechanisms within a supervised orchestration pipeline.

7. Discussion

7.1 Main Contributions

This thesis contributes to the development of adaptive human–machine interfaces for intelligent electric vehicles by proposing a cognitive-aware architecture that moves beyond both static interaction logic and purely deterministic adaptation. Rather than treating the HMI as a passive presentation layer, the proposed framework conceives interaction as a governed decision process in which actions are selected, filtered and timed according to contextual relevance, cognitive sustainability and operational constraints. In this sense, the work reframes adaptive automotive HMI as a problem of coordinated decision-making rather than simple interface customization.

A first major contribution lies in the definition of an orchestrator-centric multi-agent architecture for context-sensitive interaction. The proposed system distributes reasoning across specialized agents while preserving centralized supervision through an orchestration layer responsible for request decomposition, coordination, arbitration and execution control. This design makes it possible to combine modularity and domain specialization with system-level coherence, allowing heterogeneous functions such as safety support, comfort adaptation, route assistance, voice interaction and interface management to operate within a unified decision structure. From an architectural perspective, this is relevant because it demonstrates how adaptive in-vehicle intelligence can be organized as a controlled ecosystem of cooperating modules rather than as a monolithic assistant.

A second important contribution concerns the elevation of cognitive load from a descriptive variable to a primary architectural constraint. In the proposed framework, cognitive-state estimation is not treated as a passive annotation of driver condition, but as an operational signal that directly affects prioritization, modality selection, interaction timing and the suppression or postponement of non-essential actions. This choice is particularly significant because it embeds cognitive feasibility into the internal logic of adaptation rather than adding it as an external evaluation criterion. The ablation study reinforces this interpretation: when AI-enhanced cognitive refinement is removed, the system preserves operational continuity, but the appropriateness of the

final adaptation decreases from the full-system value of 0.886 to 0.786, together with a reduction in approval rate. This result shows that cognitive refinement materially contributes to the system's ability to calibrate interface complexity and information density in a manner that remains coherent with scenario demands.

A third contribution lies in the introduction of a structured decision-governance layer for adaptive HMI behavior. The integration of request decomposition, dynamic prioritization, conflict resolution, supervisory evaluation and deferred scheduling provides a formal mechanism for regulating candidate actions before execution. This aspect is particularly relevant in the automotive domain, where the quality of interaction depends not only on what the system suggests, but also on when and under which conditions the suggestion is delivered. The experimental evidence supports the centrality of this governance layer. The ablation analysis shows that removing the quality evaluation stage increases safety risk, whereas removing AI-based priority refinement lowers appropriateness to 0.757, slightly below the rule-based baseline. These findings indicate that the effectiveness of the framework does not arise from isolated action generation alone, but from the controlled interaction of multiple supervisory mechanisms that determine whether an adaptive decision is contextually appropriate, safe and timely.

Another central contribution of this thesis is the bounded integration of large language models within an automotive-oriented execution pipeline. In the proposed architecture, LLMs are employed as constrained reasoning components embedded within selected agents, where they support tasks such as contextual interpretation, decomposition of user requests, generation of candidate actions and adaptive content synthesis. They do not, however, act as autonomous controllers. Their outputs remain subject to deterministic prioritization, validation and safety-aware filtering. This design is important because it shows how generative AI can enhance semantic flexibility and contextual reasoning while remaining compatible with the requirements of traceability, bounded execution and architectural control that characterize safety-sensitive domains. The results of Chapter 6 are consistent with this interpretation: under the revised appropriateness framework, the hybrid configuration achieved a higher mean appropriateness score than the deterministic rule-based baseline (0.886 vs. 0.771), with a statistically significant difference and a large effect size. At the same time, this adaptive advantage is accompanied by a substantial latency cost, confirming that LLM support is most valuable when used selectively

within a supervised architecture rather than as a universal low-latency decision mechanism.

From an implementation perspective, the thesis also contributes a working prototype that operationalizes the proposed principles in a modular software pipeline. The system supports both reactive and proactive behavior, integrates multimodal contextual inputs with cognitive-state assessment and coordinated agent reasoning, and maps validated decisions into concrete outputs such as visual adaptations, voice responses and environmental adjustments. This makes the work relevant not only at a conceptual level, but also as an executable proof of architectural feasibility. The evaluation confirms that the prototype is capable of producing context-aware adaptive behavior under controlled scenario-based conditions and that its strongest configuration emerges from the coordinated combination of cognitive refinement, priority management and supervisory filtering.

Finally, the thesis contributes an evaluation perspective in which adaptive automotive HMI is assessed not only in terms of functional completion, but also in relation to appropriateness, cognitive sensitivity of the adaptive behavior, execution behavior and architectural robustness. By combining baseline comparison, latency analysis, component-level ablation and structured artifact generation, the evaluation framework makes it possible to examine the system from complementary perspectives. The resulting contribution is therefore twofold: on the one hand, the thesis proposes a concrete cognitive-aware adaptive architecture for intelligent vehicles; on the other, it provides a methodological basis for analyzing how such architectures should be interpreted when adaptive quality, timing constraints and internal component interactions must all be considered together.

7.2 Limitations

Despite the architectural and experimental contributions discussed above, the proposed framework also presents a number of limitations that should be stated explicitly. These limitations do not undermine the value of the work; rather, they define the current scope of the prototype and clarify which aspects still require further development before the approach can be considered mature for real-world deployment. In the present thesis, the main constraints concern three complementary dimensions: technical implementation, methodological evaluation and system-specific design

assumptions.

7.2.1 Technical Limitations

A first technical limitation concerns the prototype nature of the implementation. Although the system was designed to simulate a realistic adaptive HMI pipeline, it was not deployed on an actual vehicle platform and was not integrated with production-grade automotive middleware. As a consequence, several properties that are crucial in embedded automotive environments, including hardware constraints, deterministic execution guarantees, communication overhead and long-term runtime robustness, were only approximated within an offline software-based setting. For this reason, the reported results should be interpreted primarily as evidence of architectural feasibility rather than as proof of deployment readiness.

A second technical limitation is related to the nature of the contextual inputs used by the framework. The proposed architecture assumes the availability of coherent signals describing vehicle state, driver condition and environmental context. In real driving environments, however, these inputs may be noisy, delayed, partially missing or mutually inconsistent. The current prototype does not yet address in a complete way the full range of issues associated with sensor uncertainty, asynchronous data streams, fault tolerance and degraded upstream information quality. Consequently, the quality of adaptation observed in the experiments may depend on input conditions that are more controlled than those encountered in operational settings.

A further technical limitation concerns the integration of LLM-based reasoning within a time-sensitive domain. The architecture constrains generative AI through prioritization, validation and supervisory filtering, but selected stages still rely on probabilistic reasoning components whose outputs are less deterministic than purely rule-based logic [37, 21]. In the revised experimental results, this limitation does not appear as a loss of average adaptive quality, since the hybrid configuration achieves a higher appropriateness score than the deterministic baseline. It emerges instead in terms of execution behavior and deployability: the hybrid path produces a substantial latency overhead, with response times in the multi-second range, whereas the rule-based configuration remains stable at approximately 50 ms. This means that the architecture is currently stronger as a controlled adaptive reasoning prototype than as a fully real-time solution for the most time-critical interaction conditions.

7.2.2 Methodological Limitations

From a methodological perspective, the main limitation concerns the scope of the evaluation. The framework was assessed through scenario-based experimentation rather than through controlled studies involving real drivers, simulator sessions or on-road testing. This strategy is appropriate for validating architectural behavior, comparing alternative configurations and isolating component-level contributions, but it does not allow definitive conclusions about real user experience, perceived usability, trust, distraction reduction or long-term acceptance. The evidence produced in this thesis is therefore primarily system-centered rather than user-centered.

A second methodological limitation concerns the cognitive dimension of the framework. Cognitive load is treated as a central decision variable and materially influences prioritization, modality selection and suppression of non-essential interactions. However, this cognitive layer was evaluated indirectly through its effect on the quality of the final adaptive output rather than through direct human-subject validation. No participant-based protocol, physiological measurement campaign or questionnaire-based assessment such as NASA-TLX was conducted in parallel with the system evaluation. As a consequence, the thesis demonstrates the architectural usefulness of cognitive awareness more clearly than its empirically validated effect on actual driver workload and human performance.

A third methodological limitation concerns the size and structure of the benchmark itself. The comparative and ablation results are informative and internally coherent, but they are derived from a limited set of controlled scenarios selected to represent meaningful contextual conditions rather than from a large and statistically diverse experimental population. For this reason, the inferential findings should be interpreted as evidence of a promising and non-trivial trend, not as a definitive estimate of general performance under all possible driving situations. In the same way, the ablation study should be understood as a targeted component analysis rather than as an exhaustive sensitivity analysis over every possible architectural variation.

A final methodological limitation concerns the nature of the evaluation metrics. Appropriateness, approval rate, success rate and safety risk make it possible to compare architectures in a structured and reproducible way, but they remain engineered evaluation constructs rather than direct measures of human benefit. In particular, the appropriateness metric captures alignment

with scenario-level adaptation targets defined within the thesis, not a direct behavioral measure of driver comprehension, reaction quality or reduced distraction. This does not invalidate the metric; it clarifies that the present evaluation measures how well the system matches a designed adaptive benchmark, not yet how strongly it improves human driving outcomes in empirical user studies.

7.2.3 System-Specific Limitations

Beyond technical and methodological constraints, the proposed framework also presents limitations that derive from its own design choices. One important limitation is the dependence on explicit orchestration logic. The architecture gains coherence, controllability and interpretability through centralized supervision, but this same characteristic increases the number of coordination rules, thresholds and interaction policies that must be specified, tuned and maintained. As the number of agents, scenarios and adaptation pathways grows, preserving transparency and scalability in the orchestration layer may become progressively more difficult.

A second system-specific limitation concerns the bounded nature of personalization. The framework adapts timing, modality, priority and interface intensity according to contextual and cognitive signals, but it does not yet implement deep personalization based on persistent user models, longitudinal preferences or evolving behavioral histories. In other words, the system supports context-sensitive adaptation, but not yet fully individualized adaptation learned over extended usage. This means that the current architecture is better described as cognitively regulated and context-aware than as deeply personalized in a long-term human-centered sense.

A further limitation lies in the reliance on design-time assumptions for some internal decision mechanisms. Cognitive-load regulation, priority refinement and output evaluation depend on thresholds, weighting choices and operational heuristics introduced to obtain a coherent and testable adaptive framework. These choices are reasonable within the scope of the prototype and are supported indirectly by the observed experimental behavior, but they should not yet be interpreted as universally validated human-factors constants. Their present role is to operationalize the architecture in a controlled and reproducible way, not to claim definitive calibration for all drivers, vehicles or contexts.

Finally, the current system was designed primarily to demonstrate controlled adaptive behavior rather than to optimize every performance dimension simultaneously. This is particularly visible

in the trade-off between adaptive quality and responsiveness. The full hybrid architecture emerges as the strongest configuration in terms of appropriateness and overall orchestration quality, yet this advantage is achieved with a timing cost that remains substantial for strict real-time automotive use. The present framework should therefore be interpreted as a strong proof of concept for governed cognitive-aware adaptation, but not yet as a finalized low-latency in-vehicle assistant ready for unrestricted deployment in safety-critical conditions.

7.3 Design Trade-offs

The proposed architecture was developed under the assumption that adaptive automotive HMI cannot maximize all desirable properties simultaneously. In intelligent in-vehicle interaction, flexibility, contextual richness and personalization must be balanced against safety, predictability, maintainability and timing constraints. For this reason, the system was intentionally designed not as a solution optimized along a single dimension, but as a controlled compromise among competing requirements. The experimental results discussed in Chapter 6 confirm that this balance is not merely theoretical: the framework achieves stronger adaptive quality in its full hybrid form, but this advantage depends on supervisory governance mechanisms and is accompanied by non-negligible costs in execution behavior and implementation complexity. The main design trade-offs concern the relationship between AI flexibility and safety, personalization and standardization, architectural complexity and maintainability, and real-time responsiveness and adaptive quality.

7.3.1 AI Flexibility vs Safety

One of the central trade-offs of the proposed framework concerns the use of AI-based reasoning in a safety-sensitive environment. On the one hand, LLM-supported agents increase the semantic flexibility of the system, making it possible to interpret heterogeneous contexts, decompose user requests, generate richer candidate actions and support more nuanced forms of adaptation than those obtainable through deterministic mappings alone. This is especially relevant in scenarios where the expected balance between interface complexity, widget density and contextual urgency cannot be reduced to simple fixed rules.

On the other hand, the use of probabilistic reasoning introduces uncertainty, because generative

outputs are inherently less predictable than deterministic logic [37, 21]. In an automotive setting, such uncertainty cannot be accepted as an unconstrained decision principle. For this reason, the architecture does not allow AI modules to directly determine execution. Their outputs are always mediated by prioritization, validation, conflict resolution and supervisory filtering, so that generative flexibility remains bounded within an explicitly governed pipeline.

The results of Chapter 6 strongly support this design choice. The hybrid configuration achieves a higher average appropriateness than the rule-based baseline, showing that AI-enhanced reasoning can improve the overall contextual alignment of the final adaptation. However, this does not imply that unrestricted AI behavior should be preferred in every condition. The ablation study shows that when AI-based priority refinement is removed, appropriateness falls to 0.757, slightly below the rule-based baseline, while the removal of the quality evaluation layer increases safety risk even when operational completion remains unaffected. These findings indicate that the value of AI in the proposed framework does not lie in autonomous generative freedom, but in its integration within a constrained decision-governance structure. The architecture therefore accepts AI flexibility only to the extent that it remains compatible with safety-oriented supervision [24, 36].

7.3.2 Personalization vs Standardization

A second trade-off concerns the balance between context-sensitive adaptation and standardized interaction behavior. The framework was designed to tailor interface timing, modality and intensity according to driver state, cognitive load and situational urgency, since fixed and uniform responses are often inadequate in dynamic driving conditions [42, 7]. In this sense, the system clearly moves beyond static interface logic and supports a more human-centered view of adaptation, in which the same information should not always be delivered in the same way or at the same moment.

At the same time, full personalization cannot be pursued without limits in a safety-critical domain. Excessive behavioral variability would reduce predictability, complicate validation and make the assistant more difficult for the driver to understand and trust over time [22]. For this reason, personalization in the proposed architecture is intentionally bounded. The system adapts what to present, how to present it and whether to delay or suppress it, but it does so within common governance policies, shared execution constraints and interpretable design rules.

The experimental evidence is coherent with this compromise. The hybrid configuration shows a stronger ability to align with scenario-specific adaptation targets, especially when the contextual balance between complexity and information density becomes less straightforward. At the same time, this increased adaptivity is not associated with uncontrolled instability: the reported variability of the hybrid approach is not higher than that of the rule-based baseline, but slightly lower in standard-deviation terms. This suggests that the framework does not obtain stronger adaptation by becoming arbitrarily variable; rather, it produces more context-sensitive behavior while remaining within a governed and comparatively stable decision space. The resulting design is therefore not one of unrestricted personalization, but of controlled contextual differentiation: the system becomes adaptive enough to remain relevant, while preserving the regularity needed for interpretability and evaluation.

7.3.3 Complexity vs Maintainability

A further trade-off concerns the relationship between functional richness and architectural maintainability. The proposed system derives much of its strength from the coordinated interaction of multiple components: specialized agents, centralized orchestration, cognitive-state regulation, prioritization, conflict management, quality evaluation and deferred scheduling. This modular organization improves expressiveness, separates responsibilities and makes the internal logic of adaptation more explicit [54, 17]. It also supports extensibility, since additional agents or policies can in principle be integrated without redesigning the entire framework from scratch.

However, this same richness increases structural complexity. As more agents, scoring rules, thresholds, orchestration policies and interaction paths are introduced, the system becomes more demanding to design, document, calibrate and maintain. The challenge is not only computational, but also conceptual: preserving clarity about why a given adaptive output was selected becomes more difficult as the number of interacting decision layers grows. In this respect, the architecture deliberately accepts a more complex internal organization in exchange for stronger control over adaptation quality.

The ablation study provides concrete evidence of this trade-off. Different components contribute in different ways to the final behavior of the system and their removal produces distinct forms of degradation. Removing AI-enhanced cognitive refinement lowers appropriateness

and approval rate, showing that cognitive regulation materially supports the calibration of the final interface adaptation. Removing AI-based prioritization produces an even stronger drop in appropriateness, confirming that candidate generation alone is insufficient when coordination and selection become weaker. Removing the quality evaluation layer does not reduce execution success, but increases safety risk, demonstrating that supervisory filtering plays a specific and non-redundant protective role. These results show that the architecture is not complex for its own sake: its complexity reflects the fact that adaptive HMI quality emerges from the coordinated contribution of multiple specialized mechanisms. The cost of this design is reduced simplicity; the benefit is a more governable and extensible adaptive system.

7.3.4 Real-Time Responsiveness vs Adaptive Quality

The final major trade-off concerns the relationship between adaptive quality and real-time responsiveness. Richer contextual reasoning can improve the appropriateness of the final output, but it also requires additional computation. In automotive HMI, this creates a fundamental tension, because even a well-formed adaptive decision may lose practical value if it is delivered too late to be operationally useful [24, 22]. For this reason, performance in this domain cannot be evaluated only in terms of output quality; timing behavior is itself a core part of system adequacy.

The results of Chapter 6 make this trade-off particularly clear. The hybrid configuration achieves the strongest average appropriateness in the main comparison, indicating that the integration of supervised language-model reasoning supports a more refined alignment with scenario-level adaptation targets. At the same time, this advantage is obtained with a substantial latency cost, with execution times in the multi-second range. By contrast, the rule-based baseline remains stable at approximately 50 ms across the evaluated scenarios, making it considerably more compatible with situations requiring immediate and predictable response behavior. The comparison therefore does not identify a single universally dominant strategy. Instead, it reveals a structural tension between richer adaptive reasoning and stricter timing guarantees.

This tension has direct architectural implications. It suggests that AI-enhanced processing should not be interpreted as a universal replacement for deterministic adaptation, but rather as a higher-level mechanism to be activated selectively when the contextual benefit of richer reasoning justifies its computational cost. Conversely, deterministic logic remains essential whenever timing

constraints become dominant. The proposed framework reflects this compromise by combining deeper reasoning with runtime control strategies such as fallback logic, bounded execution and deferred handling of non-urgent actions. In this sense, the architecture favors situational balance rather than uniformly maximizing either speed or sophistication. The design choice is therefore not between intelligence and efficiency in the abstract, but between different operational priorities that must be weighted according to the contextual demands of the driving environment.

7.4 Practical Implications

The practical relevance of this thesis lies in the possibility of rethinking in-vehicle interaction not as a static presentation layer, but as a governed adaptive process. The proposed architecture shows that cognitive-aware adaptation can be implemented through the coordinated integration of contextual sensing, specialized reasoning modules and centralized decision control. From an engineering perspective, this suggests that future automotive HMI systems may benefit from architectures in which intelligent assistance is not introduced as an isolated feature, but embedded within a structured supervisory framework capable of regulating when, how and under which conditions adaptive behavior should be delivered [42, 54].

A first practical implication concerns the design of driver assistance interfaces. By treating cognitive load as an operational constraint, the framework supports the idea that in-vehicle interaction should be regulated not only according to what information is available, but also according to when and how that information can be presented without overloading the driver [53, 22]. In practical terms, this means that adaptive HMI design should move beyond fixed display logic and should instead consider interface complexity, information density and interaction timing as dynamically governable variables. The results of Chapter 6 support this direction: within the revised appropriateness benchmark, the hybrid configuration achieved stronger average alignment with the expected scenario-level adaptation targets than the simpler rule-based baseline. This indicates that more advanced and supervised forms of adaptation may offer concrete benefits when the contextual balance between urgency, complexity and information density becomes less straightforward.

A second implication concerns the role of AI in automotive software systems. The thesis

shows that large language models can provide useful support for contextual interpretation, candidate generation and richer adaptive reasoning, but only when they remain embedded within bounded execution pipelines [37, 21]. The comparative and latency analyses make this point particularly clear. On the one hand, the hybrid architecture achieves the strongest appropriateness results in the main comparison, suggesting that AI-enhanced reasoning can improve contextual alignment. On the other hand, this benefit is associated with a substantial execution-time overhead, whereas the deterministic rule-based strategy remains far more compatible with strict low-latency requirements. From a practical deployment perspective, this suggests that generative AI should not be treated as a universal replacement for deterministic logic in all in-vehicle functions. Rather, it is better understood as a selective reasoning layer to be activated when the expected contextual benefit justifies its computational cost, while deterministic fallbacks remain essential in the most time-sensitive situations [24, 36].

A third practical implication concerns the architectural design of future adaptive vehicle assistants. The ablation study shows that the effectiveness of the framework does not depend on one isolated intelligent component, but on the coordinated contribution of multiple modules with distinct roles. Cognitive refinement improves the calibration of the final adaptation, priority refinement affects the contextual suitability of what is ultimately presented and supervisory evaluation contributes directly to safety-oriented filtering. This has an important practical consequence: future systems should not be designed by adding AI components opportunistically on top of otherwise unchanged interfaces. Instead, adaptive intelligence should be organized as a governed pipeline in which reasoning, prioritization, validation and execution control are explicitly coordinated.

Finally, the proposed framework also has practical value as a methodological reference for future prototyping and experimental research. Its modular structure, explicit management of trade-offs and combination of reactive and proactive behavior make it a useful basis for further work on adaptive HMI, multimodal assistance and human-centered vehicle intelligence. In this sense, the thesis contributes not only a prototype solution, but also a concrete architectural direction for how intelligent in-vehicle interaction can be developed in a way that remains adaptive, structured and compatible with the constraints of safety-sensitive environments.

7.5 Ethical Considerations

The development of cognitive-aware adaptive HMI for intelligent vehicles raises ethical questions that go beyond technical performance. In automotive environments, interaction systems do not merely provide information: they influence attention allocation, shape decision timing and may affect how the driver interprets urgency, relevance and safety. For this reason, the ethical value of an adaptive assistant cannot be reduced to whether it functions correctly at a computational level. It must also be assessed in terms of responsibility, transparency, bounded autonomy, fairness and respect for the driver as the primary decision-maker within the vehicle.

A first ethical issue concerns the relationship between adaptation and driver autonomy. The purpose of a cognitive-aware interface is to support the driver by regulating complexity, timing and modality according to situational demands. However, a system that decides what to show, what to postpone and what to suppress inevitably exerts influence over the driver's informational environment. This creates an ethical obligation to ensure that adaptation remains supportive rather than manipulative. In the proposed framework, this principle is addressed by treating the assistant as a bounded decision-support system rather than as an autonomous controller. The architecture does not replace the driver's agency; instead, it attempts to reduce overload and improve contextual relevance while preserving the human as the final locus of responsibility and action.

A second ethical issue concerns safety and bounded AI autonomy. In safety-sensitive domains, the use of generative AI cannot be justified solely on the basis of flexibility or contextual richness, because probabilistic reasoning may produce outputs that are difficult to predict or explain [37, 21]. For this reason, the ethical acceptability of AI in the proposed framework depends on the fact that LLM-based reasoning is never allowed to operate as an unconstrained decision authority. Its outputs remain subject to prioritization, validation and supervisory filtering, in line with a risk-aware approach to trustworthy AI [36, 24]. This design choice is not only technically prudent, but ethically necessary: it ensures that increased intelligence does not come at the cost of uncontrolled behavior in a context where interaction errors may have safety-relevant consequences.

A third ethical consideration concerns transparency and interpretability. Adaptive systems are difficult to trust if users cannot understand, at least at a high level, why the interface behaves differently across situations. This issue becomes even more important when adaptation is driven by internal estimates of cognitive state, urgency or contextual suitability. In the proposed architecture, transparency is supported not by full exposure of every internal computation, but by the use of explicit orchestration logic, structured evaluation stages and interpretable governance mechanisms. From an ethical perspective, this is important because it improves traceability and makes the behavior of the system easier to justify, inspect and revise. In other words, the framework aims to make adaptation governable rather than opaque, which is a necessary condition for accountability in intelligent automotive systems.

A fourth ethical issue relates to privacy and the treatment of driver-related data. A cognitive-aware HMI may rely on signals associated with driver condition, behavior or inferred workload, and these signals can be sensitive even when they are not explicitly medical in nature. This creates a clear ethical responsibility to minimize data collection, restrict processing to what is operationally necessary and avoid turning adaptive assistance into a form of excessive surveillance. Although the present thesis does not implement a full privacy-preserving infrastructure, the architecture is conceptually compatible with a data-minimization approach, since cognitive information is used as a functional regulation signal for adaptation rather than as a basis for user profiling or secondary exploitation. This distinction is ethically relevant: the legitimacy of cognitive-aware adaptation depends in part on whether driver-related data are used to protect attention and safety, rather than to maximize extraction, monitoring or behavioral control.

A fifth consideration concerns fairness and generalization. Adaptive systems may appear beneficial on average while still performing unevenly across users, driving styles or contextual conditions. This issue is particularly relevant when some internal parameters, thresholds or heuristics are defined at design time rather than learned from broad and diverse populations. In the present work, the architecture demonstrates promising behavior under controlled scenarios, but it has not yet been validated across heterogeneous real-world driver groups. Ethically, this means that the current system should not be interpreted as universally fair or equally suitable for all users. The responsible position is to recognize that personalization and cognitive adaptation may introduce uneven effects unless future work explicitly evaluates variability across individuals,

contexts and usage patterns.

Finally, ethical responsibility also concerns how claims about such systems are framed. The results of Chapter 6 show that the full hybrid architecture improves adaptive appropriateness relative to the deterministic baseline, but they also show that this advantage is accompanied by a substantial latency cost. This has an ethical implication for deployment: a system should not be presented as categorically superior simply because it is more intelligent or more adaptive in principle. In safety-sensitive environments, a responsible evaluation must acknowledge both benefits and constraints, including the fact that richer reasoning may not always be preferable when strict timing requirements dominate. For this reason, the contribution of this thesis should be interpreted as a governed proof of concept for ethical cognitive-aware adaptation, not as a final claim that AI-driven vehicle interaction is ready to replace deterministic logic in all operational conditions.

Taken together, these considerations suggest that the ethical value of the proposed framework lies less in the mere use of AI than in the way AI is bounded, supervised and subordinated to safety-oriented governance. The architecture is ethically defensible to the extent that it supports the driver without displacing responsibility, enhances contextual relevance without sacrificing control and incorporates adaptive intelligence without normalizing opacity or unrestricted autonomy. In this sense, the thesis supports a view of intelligent automotive HMI in which ethical acceptability is inseparable from architectural discipline.

8. Future Work

The architecture presented in this thesis shows how cognitive-aware adaptation, multi-agent orchestration and bounded LLM-based reasoning can be combined to support context-sensitive human–machine interaction in electric vehicles. The results discussed in the previous chapters indicate that this direction is promising, especially because the proposed system relies on the coordinated contribution of perception, prioritization, supervision and controlled response generation rather than on a single intelligent component. At the same time, the current implementation should still be regarded as a research prototype rather than as a deployment-ready solution. For this reason, several future developments can be identified to improve robustness, broaden contextual intelligence and extend the applicability of the framework beyond the current experimental setting.

8.1 Technical Improvements

A first line of future work concerns the technical evolution of the architecture. Although the proposed system already supports adaptive behavior through a modular orchestration pipeline, several components could be strengthened to improve deployment realism, contextual precision and long-term adaptability. Four directions appear especially relevant: the adoption of on-device language models, the refinement of multimodal fusion strategies, the introduction of privacy-preserving distributed learning mechanisms and the evolution of priority computation toward more adaptive decision policies. These developments would preserve the conceptual foundations of the framework while improving its robustness, scalability and suitability for real-world automotive scenarios.

8.1.1 On-Device LLMs

One of the most relevant directions for future work is the progressive transition from externally hosted language models to on-device or edge-deployed LLMs. In the current prototype, language models are integrated through structured prompting, schema-constrained generation and explicit

validation layers. This design already limits the risks typically associated with unconstrained generative systems and makes LLM support compatible with the safety-oriented requirements of the system. However, it still introduces a dependency on external inference services, which may affect latency stability, connectivity robustness and data-governance requirements.

Moving part of the reasoning pipeline onto the vehicle would provide several advantages. First, it would reduce the variability caused by network communication and remote inference, making response timing more predictable in conditions where stable interaction is essential. Second, it would improve privacy protection, since sensitive contextual information related to driver state, behavioral patterns and vehicle conditions could remain within the local computational environment. Third, local deployment would make the architecture more realistic for automotive use, where continuous cloud availability cannot always be assumed and where low-latency support is often preferable to more powerful but less predictable remote computation [37].

At the same time, the adoption of on-device LLMs should not be treated as a simple replacement of one model with another. Compact local models usually provide lower reasoning depth and may be less reliable in tasks that require precise contextual synthesis or strict adherence to structured outputs. For this reason, future work should investigate hybrid inference strategies rather than full substitution. A promising solution would be a tiered reasoning architecture in which a lightweight local model handles time-sensitive interpretation and first-level suggestion generation, while more advanced remote models are invoked only for non-urgent or cognitively appropriate situations. In this way, the system could preserve the flexibility enabled by language models while improving autonomy, timing consistency and deployment feasibility.

More broadly, research on on-device LLMs would help consolidate the role of language models as bounded reasoning components within the adaptive pipeline. Bringing this capability closer to the vehicle would therefore represent not only a technical optimization, but also a more mature form of AI-assisted in-vehicle interaction.

8.1.2 Multimodal Fusion

A second major direction for future work concerns richer multimodal fusion. The current architecture already combines heterogeneous inputs from different subsystems, including driver monitoring, vehicle signals, environmental conditions, ADAS-related events and external contex-

tual information. This integration is one of the strengths of the proposed framework, because it allows adaptation to emerge from a broader understanding of the driving situation rather than from isolated variables. Nevertheless, the present implementation still relies on a partially modular fusion logic, in which some interactions among signals are simplified through staged aggregation and rule-based interpretation.

Future developments could improve this aspect by introducing more expressive fusion mechanisms able to model temporal dependencies, cross-modal interactions and uncertainty more effectively. This is particularly important because many cognitively relevant situations do not arise from a single dominant signal, but from the combination of several moderate indicators. For example, unstable gaze behavior, increasing traffic density, repeated ADAS warnings and poor weather conditions may each be manageable when considered separately, but together they may indicate a qualitatively different workload condition. A more advanced fusion layer would be better suited to capturing these compound states and distinguishing them from simpler contextual variations.

Such an improvement would benefit not only workload estimation, but also the quality of the adaptation itself. A cognitively aware HMI should not merely determine whether the driver is under low or high load; it should also identify which dimensions of the context are generating that state. The difference is substantial. A driver experiencing perceptual overload due to dense traffic and environmental complexity may require suppression of secondary outputs and strong interface simplification, whereas a driver in a monotonous low-stimulation scenario may benefit from a different form of support, such as mild engagement strategies or anticipatory prompts. More refined multimodal fusion would therefore make the system not only more precise in regulating adaptive behavior, but also more semantically aware in the way it selects and shapes adaptive responses.

In this sense, future work should aim at transforming multimodal fusion from a predominantly analytic layer into a more explanatory one. The goal should not be limited to obtaining a better scalar estimate of cognitive demand, but should include the construction of a richer contextual representation describing why a certain driver state has emerged and which adaptation dimensions should consequently be adjusted.

8.1.3 Federated Learning

Another relevant direction for future work concerns the introduction of federated learning mechanisms to support personalization and continuous model improvement without relying on centralized collections of sensitive user data. In the current prototype, adaptation is already designed to be context-sensitive and potentially extensible toward user-aware behavior, since the orchestration logic can regulate candidate actions according to contextual and cognitive conditions, even though it does not yet implement deep personalization based on persistent user models or longitudinal interaction histories. However, this capability is still limited by the fact that the system does not yet include a structured learning framework able to refine its internal models over time across multiple users or vehicles.

Federated learning would provide a promising way to address this limitation. Instead of transferring raw behavioral or contextual data to a central server, each vehicle could locally update selected models on the basis of its own interaction history, while only aggregated model updates would be shared externally [30]. This would be especially valuable in a domain such as intelligent automotive HMI, where personalization can improve user experience, but where privacy, data minimization and deployment trust remain essential constraints. Through this approach, the architecture could progressively learn driver-specific preferences, recurring mobility habits and typical reactions to adaptive interventions, while maintaining stronger control over personal information.

The value of this direction lies in the fact that driver populations are inherently heterogeneous. Users differ in their tolerance to interruptions, in their preferred balance between proactivity and minimalism, in their sensitivity to multimodal outputs and in the way they respond to stress or information density. A fixed global model can only approximate this variability. By contrast, a federated learning strategy would make it possible to combine two complementary objectives: preserving local adaptation to individual patterns and still benefiting from fleet-level knowledge extracted from distributed usage. In this way, the framework could evolve from a system that is merely configurable into one that becomes progressively more personalized through real interaction data.

At the same time, the introduction of federated learning would require careful governance. In

the context of this thesis, personalization cannot be interpreted as unconstrained optimization of user preferences, because adaptive behavior must always remain compatible with safety-oriented interaction principles. Future work should therefore investigate federated learning not only as a technical enhancement, but as a controlled personalization strategy embedded within explicit safety boundaries.

8.1.4 Reinforcement Learning for Priority Engine

A further important extension concerns the evolution of the priority engine toward more adaptive learning-based policies. In the current architecture, action ranking is guided by an explicit combination of criteria such as urgency, safety relevance, contextual appropriateness, cognitive compatibility and expected utility for the driver. This design has the advantage of being transparent and relatively interpretable, which is particularly important in a safety-aware system. At the same time, it remains largely based on predefined weighting logic and does not yet learn systematically from the long-term outcomes of previous decisions.

Future work could explore the integration of reinforcement learning techniques in order to refine prioritization on the basis of observed interaction effects. Rather than relying only on manually designed scoring schemes, the system could progressively learn which categories of interventions tend to be accepted, ignored, postponed or overridden under recurring contextual conditions. Over time, this would allow the priority engine to model not only the immediate theoretical relevance of an action, but also its likely effectiveness in practice. For example, the architecture could learn that some proactive suggestions are useful only in specific cognitive-load ranges, that certain timing patterns improve acceptance, or that some classes of recommendations become counterproductive when repeatedly proposed within a short interaction window.

The main benefit of this direction is that prioritization could become more sensitive to sequential and long-term interaction dynamics. In adaptive automotive interfaces, the value of a decision cannot always be assessed at the instant it is made. Some actions may produce delayed benefits, while others may appear individually reasonable but collectively generate unnecessary burden, distraction or habituation. A learning-based priority policy could therefore improve the balance between responsiveness and restraint by incorporating feedback from cumulative system behavior rather than from isolated decision points alone.

Nevertheless, reinforcement learning must be approached with particular caution in this domain. An in-vehicle assistant cannot be allowed to explore arbitrary behaviors in order to discover what works best, because experimentation itself may introduce unsafe or cognitively inappropriate interactions. For this reason, future work should focus on constrained variants of reinforcement learning, such as offline reinforcement learning or policy refinement over curated historical traces, in which learning occurs under strong supervision and remains subordinate to explicit rule-based safety filters [28]. In this configuration, the learned policy would not replace the current governance logic, but would improve the calibration of priorities within fixed operational boundaries.

8.2 Extended Use Cases

Beyond technical refinement, an important line of future development concerns the extension of the proposed framework to broader application scenarios. The architecture introduced in this thesis has been designed and evaluated within the domain of electric-vehicle human–machine interaction, with particular attention to cognitive awareness, adaptive assistance and orchestrated decision-making. However, the same architectural principles may support a wider range of services and operational contexts, especially when the vehicle is considered not as an isolated platform but as part of a larger connected ecosystem.

In this perspective, future work should investigate how the current system can evolve from an in-vehicle adaptive assistant into a more comprehensive intelligence layer capable of mediating between driver, vehicle, infrastructure and external services. This extension should not be reduced to the addition of new information sources. Rather, it concerns the ability of the architecture to incorporate richer contextual signals while preserving the same governance principles that characterize the present framework. Three directions appear especially relevant in this respect: Vehicle-to-Everything communication, predictive maintenance support and integration with smart-city services.

8.2.1 V2X Integration

A first natural extension of the proposed framework concerns the integration of Vehicle-to-Everything communication. In the current prototype, adaptation decisions are derived primarily from internal vehicle signals, driver-monitoring information, environmental conditions and selected external contextual services. This already enables a relatively rich model of the driving situation. Nevertheless, the vehicle is still treated mainly as an autonomous sensing and decision node, while future mobility systems will increasingly rely on cooperative information exchange with surrounding vehicles, roadside infrastructure and connected transport platforms.

Integrating V2X capabilities would allow the architecture to reason on events and conditions that may not yet be directly observable through onboard sensors alone [13]. Examples include hazards beyond the driver's immediate field of perception, dynamic traffic coordination messages, infrastructure alerts, intersection-phase information, warnings transmitted by nearby vehicles and availability updates related to charging or road access conditions. Access to this kind of information would strengthen the anticipatory dimension of the system, enabling earlier and potentially more useful adaptive interventions.

From the perspective of human–machine interaction, however, the main challenge is not simply the availability of additional data, but its governed integration into the interaction loop. In a connected driving environment, the volume of potentially relevant information may increase significantly and naive transmission of such data to the driver would risk producing exactly the kind of overload that the proposed architecture is designed to avoid. For this reason, V2X integration should be understood as a test of the framework's scalability rather than as a mere functional addition. The orchestrator would become even more important as a filtering and mediation layer, responsible for deciding which cooperative signals deserve immediate attention, which should be reformulated into lower-burden suggestions and which should be suppressed or deferred altogether.

In this sense, V2X integration would extend the current architecture in a way that is fully aligned with its original rationale. The value of connected information would lie in improving contextual foresight without compromising cognitive sustainability.

8.2.2 Predictive Maintenance with AI

A second promising extension concerns the integration of predictive maintenance capabilities into the adaptive assistance framework. At present, the system mainly focuses on supporting the driving experience in relation to contextual, cognitive and interaction-level factors such as workload, route feasibility, charging needs and the urgency of ongoing events. Although some vehicle-status information already contributes to decision-making, the architecture does not yet exploit longer-term technical patterns related to component degradation or maintenance risk.

Future work could enrich this dimension by incorporating machine-learning models able to detect early signs of anomalies or progressive deterioration in relevant subsystems, such as battery health, braking efficiency, tire conditions, thermal regulation or sensor reliability [6]. This would make it possible to move from a predominantly situational assistant toward a more preventive and longitudinal support system. Rather than reacting only when a warning has already become explicit, the architecture could identify developing conditions and translate them into context-sensitive maintenance recommendations before critical failures or severe performance reductions emerge.

The contribution of such an extension would be particularly significant in electric vehicles, where energy management, battery condition and system efficiency play a central role in both usability and user trust. A cognitively aware HMI could, for example, recognize that a combination of charging history, temperature trends and consumption anomalies suggests progressive battery stress and could therefore propose a maintenance check or a modification of charging behavior at an appropriate moment. Similarly, recurring deviations in tire-pressure dynamics or sensor reliability could be transformed into anticipatory alerts that are not delivered as isolated technical messages, but integrated into the broader interaction strategy of the assistant.

The relevance of predictive maintenance for the present framework does not lie only in technical forecasting accuracy, but also in the possibility of governing maintenance-related communication through the same adaptive principles already applied to other forms of support. Integrating it into the orchestrated pipeline would allow the system to communicate technical risk in a more intelligent and less disruptive way, linking diagnosis, timing and actionable support within a unified assistance logic.

8.2.3 Smart City Integration

A third important direction concerns the interaction between the proposed architecture and smart-city ecosystems. As electric mobility becomes increasingly embedded in digitally connected urban environments, vehicles are no longer influenced only by road conditions and internal status, but also by a wide range of city-level services and constraints, including dynamic traffic policies, parking infrastructures, charging-network availability, local environmental regulations and multimodal transport coordination. Future work should therefore explore how the framework can incorporate these broader urban signals into its adaptive decision-making process.

Such an extension would be valuable because many of the decisions that affect driver workload and interaction quality are not generated solely within the vehicle. Urban congestion, restricted-access areas, charging-station occupancy, route disruptions and parking uncertainty can all increase mental demand and decision pressure, especially in electric mobility scenarios where operational planning is often more tightly coupled with infrastructure conditions. A more advanced version of the proposed assistant could use smart-city data to anticipate these pressures and provide support before they become immediate sources of stress.

For instance, the architecture could exploit city-scale information to recommend route adjustments, more suitable charging stops, alternative parking solutions or timing adaptations based on expected congestion or service availability. Importantly, these suggestions should not be conceived as isolated convenience features. Within the logic of this thesis, they would represent another form of context-sensitive assistance governed by cognitive appropriateness. A route alternative that is useful in principle may still be a poor intervention if proposed during a high-load maneuver; similarly, detailed parking information may be valuable before entering a dense urban area but counterproductive if delivered too late or with excessive complexity.

The integration with smart-city systems would therefore reinforce the role of the HMI as an intelligent mediation layer between driver, vehicle and environment. Rather than limiting adaptation to immediate in-cabin conditions, the framework could evolve toward a broader model of situated support in which urban infrastructure becomes an integral part of the contextual space interpreted by the orchestrator.

8.3 Long-Term Studies

Although the experimental evaluation presented in this thesis provides a meaningful first assessment of the proposed architecture, an important direction for future work concerns its validation over extended periods of use and under more ecologically realistic conditions. The current results are sufficient to support the technical plausibility of the framework and to highlight the contribution of its main components, but they do not yet capture the full temporal dynamics that characterize repeated interaction between a driver and an adaptive in-vehicle assistant.

This limitation is particularly relevant because the value of a cognitive-aware HMI cannot be evaluated only through short-term performance indicators. In systems of this kind, effectiveness depends not only on whether a single intervention is appropriate at a given moment, but also on how the overall interaction strategy evolves over time. Repeated exposure may change how users perceive suggestions, how quickly they understand the system's behavior and the extent to which they trust or disregard different forms of assistance. For this reason, future validation should move beyond scenario-based assessment and include longitudinal studies capable of observing adaptation as an ongoing relationship rather than as an isolated event [27].

A first reason for conducting long-term studies concerns habituation. An adaptive behavior that appears useful and well-calibrated in a short experimental session may become less effective after repeated exposure, either because the driver becomes accustomed to certain forms of intervention or because the same adaptive patterns lose salience over time. This is especially important in a system that includes proactive assistance, deferred suggestions and context-sensitive modulation of outputs. What is initially perceived as supportive may later be interpreted as predictable, redundant or insufficiently informative unless the system is able to preserve variation and contextual relevance across repeated usage.

A second important aspect concerns trust calibration. The proposed framework is designed to balance flexibility and control, partly through bounded LLM support and partly through explicit orchestration and evaluation layers. However, the way users interpret this balance cannot be fully assessed in short-term experiments. Trust in an adaptive assistant develops progressively: users need time to understand when the system is reliable, when it tends to remain silent, when it intervenes proactively and how its suggestions relate to contextual conditions.

Longitudinal evaluation would therefore be necessary to determine whether the architecture promotes appropriate trust, excessive reliance or, conversely, systematic underuse of potentially valuable support.

A third motivation relates to personalization. Several potential strengths of the proposed architecture emerge more clearly when interaction is observed across repeated sessions, since driver-specific patterns, preferences and behavioral regularities become more visible only over time. This is true both for explicit preferences and for more implicit tendencies, such as tolerance to interruption, preferred levels of proactivity or recurring reactions under stress. Without longitudinal observation, it is difficult to establish whether adaptive mechanisms genuinely improve the interaction for individual users or whether they remain limited to short-term context sensitivity without meaningful long-term personalization.

Long-term studies would also be essential for identifying unintended effects. In a real deployment scenario, even a well-designed adaptive system may gradually introduce issues that are not immediately visible in controlled tests, such as alert fatigue, reduced responsiveness to repeated categories of recommendations, excessive dependence on assistance or mismatch between subjective user expectations and actual system behavior. These effects are particularly important in safety-related domains, where the long-term quality of interaction matters as much as immediate decision accuracy. A future research agenda should therefore consider not only whether the system performs well in isolated scenarios, but also whether it remains beneficial, acceptable and cognitively sustainable after prolonged use.

From a methodological perspective, this suggests the need for evaluation protocols that combine objective and subjective measures over time. Future studies could examine variables such as action acceptance rate, override frequency, deferral effectiveness, intervention timing quality and stability of workload-related indicators, while also collecting user-centered measures related to trust, perceived usefulness, intrusiveness and mental effort. The combination of these dimensions would provide a more realistic understanding of how the architecture behaves outside controlled proof-of-concept conditions.

Ultimately, long-term studies are necessary to move the contribution of this thesis from initial architectural validation toward stronger evidence of sustained practical relevance. If the framework is intended to support real drivers in dynamic and cognitively variable environments,

then its success must be assessed not only in terms of immediate technical coherence, but also in terms of durable interaction quality, behavioral acceptance and stability over time.

8.4 Generalization

A final direction for future work concerns the generalization of the proposed framework beyond the specific prototype and application setting explored in this thesis. Although the architecture has been developed in the context of electric-vehicle human–machine interaction, its main contribution is not limited to a single use case or technological configuration. The broader value of the framework lies in the combination of orchestrated multi-agent coordination, cognitive-aware decision governance and bounded AI support for adaptive interaction. These principles may remain relevant even when the surrounding domain, available data sources or target services change significantly.

Within the automotive field, a first form of generalization concerns the extension of the architecture to vehicle categories and mobility scenarios different from the one considered here. The current prototype has been designed with electric vehicles in mind, particularly because EV driving introduces specific interaction needs related to energy management, charging planning and infrastructure dependence. However, many of the architectural mechanisms proposed in this thesis are not inherently tied to electrification. Cognitive-load-aware prioritization, conflict resolution among candidate actions, deferred execution of non-urgent interventions and explainable adaptive assistance are equally meaningful in conventional vehicles, hybrid vehicles, shared-mobility platforms and commercial fleets. In these cases, the domain-specific agents and contextual variables would change, but the underlying logic of selective, cognitively compatible and orchestrated support would remain applicable.

A second and even more significant level of generalization concerns different driving automation settings. As vehicles evolve toward higher levels of assisted and partially automated driving, the challenge of managing the interaction between human attention, system state and contextual complexity becomes even more critical. In such scenarios, the framework proposed in this thesis could support functions such as takeover preparation, explanation of automation state transitions, prioritization of alerts under degraded conditions and adaptive recovery of driver

attention. Here again, the specific content of the interaction would differ from the electric-vehicle case, but the broader architectural problem would remain similar: the system must decide not only what information is available, but what should be communicated, when, with what urgency and under which cognitive conditions.

Beyond the automotive domain, the framework may also offer a more general design pattern for adaptive AI systems operating in safety-relevant environments. Many application areas involve the same fundamental tension addressed in this thesis: the need to exploit increasingly powerful AI-based reasoning mechanisms without allowing flexibility to undermine predictability, interpretability or human cognitive compatibility. Domains such as aviation, healthcare support systems, industrial supervision, control-room assistance or advanced operator interfaces all present variations of this challenge. In each case, intelligent support must be context-sensitive, selective and accountable rather than merely informative or highly generative.

From this perspective, the broader contribution of the thesis is methodological as much as technical. It suggests that adaptive intelligence in complex human-centered systems should not be organized around a single monolithic model, but around a governed architecture in which perception, prioritization, evaluation and response generation remain functionally differentiated. This separation is important because it allows AI flexibility to be embedded within explicit decision structures instead of being treated as an autonomous source of action. Such an approach may prove valuable in many domains where assistance needs to remain both adaptive and bounded.

For this generalization to be made more explicit, future work should identify more clearly which parts of the framework are domain-specific and which can be considered domain-agnostic. Some components are evidently tied to the electric-vehicle context, such as charging-related reasoning, selected mobility services or certain vehicle-state interpretations. Others, however, are potentially reusable at a much broader level, including cognitive-state integration, action arbitration, deferred execution under overload conditions, supervision of generated suggestions and controlled multimodal adaptation. Formalizing this distinction would make the architecture easier to transfer, compare and adapt across different scenarios.

In this sense, generalization should not be understood as a claim that the current prototype already solves interaction problems in multiple domains. Rather, it should be interpreted as the possibility of extending the architectural logic developed in this thesis into a broader class of

AI-assisted systems in which contextual intelligence must be balanced with safety, interpretability and human-centered control. This perspective gives the work a broader relevance: not only as a contribution to electric-vehicle HMI, but also as a step toward more accountable, cognitively aware and transferable interactive AI architectures.

9. Conclusions

9.1 Summary of Contributions

This thesis contributes to adaptive automotive HMI through a coherent set of connected advances. Architecturally, it proposes an orchestrator-centric multi-agent framework that replaces fragmented adaptive logic with supervised coordination across specialized agents. Conceptually, it elevates cognitive load from a descriptive variable to a primary operational constraint affecting prioritization, timing, modality selection and deferred execution. Methodologically, it introduces a governed pipeline in which request decomposition, prioritization, evaluation, conflict resolution and waiting-queue logic regulate candidate actions before execution. Technically, it demonstrates that bounded LLM-based reasoning can be integrated into a safety-relevant architecture without granting direct actuation authority to generative components. Finally, it provides a modular prototype and a reproducible evaluation workflow combining comparative analysis, latency assessment and ablation study.

Taken together, these contributions show that adaptive in-vehicle interaction can be framed not merely as interface customization, but as a problem of controlled cognitive-aware orchestration. Within the scope of the adopted benchmark, the proposed framework demonstrates stronger adaptive appropriateness than the deterministic baseline, while also making explicit the corresponding trade-off in latency and deployability. The contribution of the thesis therefore lies not in claiming an unrestricted AI assistant for vehicles, but in showing how richer contextual reasoning can be embedded within bounded, transparent and safety-oriented execution structures.

9.2 Achievement of Objectives

The objectives defined at the beginning of the thesis can be considered substantially achieved.

The first objective was to design an HMI architecture capable of overcoming the fragmentation and rigidity of traditional adaptive systems. This objective was achieved through the introduction of an orchestrated multi-agent framework in which specialized modules operate under a central

coordination layer. Compared with sequential and weakly integrated adaptation pipelines, the proposed architecture provides a more expressive and governable structure for handling heterogeneous inputs, concurrent objectives and context-dependent interaction policies. The second objective was to incorporate driver cognitive load into the operational core of the HMI. This goal was achieved by embedding cognitive-state estimation directly into the decision pipeline as a variable that constrains action prioritization, timing and modality selection. As a result, the interface is not merely context-aware in a generic sense, but explicitly regulated according to the driver's current interaction capacity. This point is particularly important because one of the central gaps identified in the literature was precisely the weak integration of cognitive modeling into high-level system logic.

The third objective was to explore the use of advanced AI techniques, including LLM-based reasoning, while remaining compatible with automotive constraints. This objective was addressed by adopting a bounded integration strategy in which language models support selected reasoning tasks but remain subordinated to deterministic governance mechanisms. The thesis therefore shows that AI-based semantic reasoning can be incorporated into an automotive HMI without assigning unconstrained decision authority to generative models.

The fourth objective was to verify through implementation and evaluation whether the proposed architecture could produce more appropriate adaptive behavior than simpler alternatives. The experimental chapter supports this objective in two complementary ways. First, the comparative evaluation shows that the proposed framework produces context-aware behavior under heterogeneous simulated scenarios and, within the revised appropriateness framework adopted in this thesis, achieves stronger average alignment with the expected adaptation targets than the simpler deterministic rule-based baseline. Second, the ablation study demonstrates that the overall system behavior depends on the coordinated contribution of cognitive refinement, prioritization, quality supervision and controlled generation. The results therefore do not support a simplistic interpretation in which one isolated AI component explains system performance; rather, they confirm the architectural value of orchestration and governance.

Taken together, these elements indicate that the thesis achieved its primary goals at three levels: conceptual, through the formulation of a coherent architectural answer to the identified research gap; technical, through the implementation of a functioning prototype; and methodological,

through the construction of a reproducible evaluation workflow able to support the discussion with measurable evidence.

9.3 Impact on EPIGNOSIS Project

Within the broader context of the EPIGNOSIS project, this thesis contributes by extending the original adaptive HMI perspective toward a more explicit model of intelligent orchestration. The baseline project architecture already provided an important foundation for context acquisition, task generation and multimodal interaction management. However, the work developed in this thesis advances that foundation by introducing a supervisory orchestration layer, cognitively informed decision logic and a structured strategy for integrating AI-based reasoning into the adaptive HMI pipeline.

The impact on the project is therefore both architectural and methodological. From an architectural point of view, the thesis transforms the HMI from a mainly functional adaptation engine into an active coordination layer capable of arbitrating information, regulating intervention timing and maintaining coherence across multiple decision sources. From a methodological point of view, it provides EPIGNOSIS with a concrete prototype and an evaluation-oriented framework that can support future comparison, refinement and validation activities.

Another significant contribution to EPIGNOSIS lies in the conceptual repositioning of the HMI. In the proposed view, the interface is not a passive output layer attached to vehicle intelligence, but an active and context-sensitive component of that intelligence. This perspective is fully aligned with the project's cognitive approach to interaction and with its broader goals of safety, comfort and efficiency, particularly in the context of electric and hybrid vehicles, where the amount of contextual information and interaction complexity is steadily increasing.

In this sense, the thesis does not simply add one more adaptive mechanism to the EPIGNOSIS ecosystem. It offers a possible direction for the evolution of the project toward a more governed, cognitively aware and AI-enabled interaction architecture, capable of supporting future developments in advanced driver assistance and intelligent vehicle interfaces.

9.4 Broader Impact

Beyond the specific EPIGNOSIS context, the work presented in this thesis has broader implications for the design of intelligent interactive systems in safety-relevant domains.

From an automotive perspective, the thesis reinforces the idea that future HMIs should not be evaluated only in terms of usability, responsiveness or feature richness, but also in terms of their ability to regulate cognitive demand. In vehicles characterized by increasing levels of sensing, connectivity and automation, the main challenge is no longer simply to provide more information, but to deliver the appropriate amount of information in a cognitively sustainable way. In this respect, cognitive-aware adaptation may become a central design principle for next-generation in-vehicle interfaces.

From an AI systems perspective, the thesis also contributes to a broader discussion on how generative models should be integrated into constrained environments. The proposed architecture suggests that the most promising direction is not unrestricted conversational autonomy, but supervised integration within explicit decision structures. In such a setting, AI supports interpretation and flexibility, while prioritization, validation and execution control remain governed by transparent architectural rules. This principle is potentially relevant well beyond the automotive domain.

A further broader implication concerns human-AI cooperation. The framework developed in this thesis does not position the driver as a passive target of automated decisions, nor the AI as an autonomous controller. Instead, it supports a cooperative model in which system intelligence remains adaptive but bounded, and in which assistance is shaped by contextual demands and human cognitive limitations. This perspective may prove valuable in other human-centered domains where contextual intelligence must remain compatible with safety, interpretability and controlled interaction.

9.5 Final Remarks

This thesis started from a central observation: in modern vehicles, the problem is no longer only how to increase computational intelligence, but how to make that intelligence interact

with the driver in a safe, timely and cognitively sustainable manner. Addressing this challenge required combining insights from cognitive theory, adaptive HMI design, multi-agent systems and constrained AI reasoning within a single architectural proposal.

The resulting framework does not claim to solve every challenge of intelligent in-vehicle interaction. Rather, it offers a coherent and technically grounded step toward a new class of automotive HMIs that are not merely reactive, but context-sensitive, cognitively aware and explicitly orchestrated. Its main contribution lies in showing that these properties can be combined within a governable architecture that remains compatible with the logic and constraints of safety-relevant systems.

The broader significance of the work is therefore not limited to the specific prototype developed in this thesis. More generally, it lies in the argument that adaptive intelligence in human-centered systems should be organized through bounded and transparent coordination mechanisms, rather than delegated to isolated subsystems or monolithic AI components. In the automotive domain, this means moving beyond static interaction policies and fragmented smart features toward a more principled model of human–vehicle cooperation.

Beyond its architectural contribution, the implemented prototype also demonstrates that cognitive-aware adaptive HMI can be realized as a concrete software system rather than as a purely conceptual framework. The project integrates orchestrated agent reasoning, centralized cognitive-load management, bounded LLM support, asynchronous execution, contextual API enrichment, provider-agnostic speech services and a predictive layer prepared for data-driven extension. Taken together, these elements show that the proposed approach is not limited to a theoretical redesign of the interaction pipeline, but already takes the form of a modular and extensible prototype suitable for systematic experimentation.

In conclusion, this thesis shows that cognitive-aware orchestration is a credible design direction for the future of intelligent automotive interfaces. Especially in electric and software-defined vehicles, where interaction is becoming an increasingly strategic component of the driving experience, the ability to coordinate information, timing and modality according to both context and cognitive state is likely to become not an optional refinement, but a fundamental requirement of next-generation HMI design.

Bibliography

[1] Dario Amodei et al. “Concrete Problems in AI Safety”. In: *arXiv preprint arXiv:1606.06565* (2016). URL: https://arxiv.org/abs/1606.06565.

[2] Jackson Beatty. “Task-Evoked Pupillary Responses, Processing Load, and the Structure of Processing Resources”. In: *Psychological Bulletin* 91.2 (1982), pp. 276–292. DOI: 10.1037/0033-2909.91.2.276.

[3] BMW Group. *BMW iDrive System Overview*. 2023. URL: https://www.bmw.com.

[4] Rishi Bommasani et al. “On the Opportunities and Risks of Foundation Models”. In: *arXiv preprint arXiv:2108.07258* (2021). URL: https://arxiv.org/abs/2108.07258.

[5] Peter Brusilovsky. “Adaptive Hypermedia”. In: *User Modeling and User-Adapted Interaction* 11.1-2 (2001), pp. 87–110. DOI: 10.1023/A:1011143116306.

[6] T. P. Carvalho et al. “A systematic literature review of machine learning methods applied to predictive maintenance”. In: *Computers & Industrial Engineering* 137 (2019), p. 106024. DOI: 10.1016/j.cie.2019.106024.

[7] Fabian Clemens, Antonio Krüger, and Albrecht Schmidt. “Context-Aware User Interfaces for Automotive Applications”. In: *AutomotiveUI*. 2019. DOI: 10.1145/3342197.3344523.

[8] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed. Lawrence Erlbaum Associates, 1988.

[9] David F. Dinges and Melissa M. Mallis. *PERCLOS: A Valid Psychophysiological Measure of Alertness*. Tech. rep. Federal Highway Administration, 1998. URL: https://rosap.ntl.bts.gov/view/dot/14174.

[10] Johan Engström. “Understanding Attention Selection in Driving”. PhD thesis. Linköping University, 2005. URL: https://liu.diva-portal.org/smash/record.jsf?pid=diva2:21777.

[11] EPIGNOSIS Consortium. *D5.1 – Report di Presentazione del Modello di Context Monitoring per l’HMI Adattativa*. Internal project deliverable. 2024.

[12] EPIGNOSIS Consortium. *D5.2.2 – Report sull'Architettura dei Flussi HMI e Risultati del Primo Ciclo di Prototipazione (low-fi)*. Internal project deliverable. 2025.

[13] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications*. European Telecommunications Standards Institute. 2020.

[14] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 (General Data Protection Regulation)*. 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[15] Lex Fridman et al. "Driver Monitoring Systems: State of the Art and Research Challenges". In: *IEEE Transactions on Intelligent Transportation Systems* 22.5 (2021), pp. 2990–3007. DOI: 10.1109/TITS.2020.2987927.

[16] Krzysztof Z. Gajos, Daniel S. Weld, and Jacob O. Wobbrock. "SUPPLE: Automatically Generating User Interfaces". In: *IUI*. 2006. DOI: 10.1145/1111449.1111471.

[17] A. Guillet et al. "A Multi-Agent System for Cooperative ADAS". In: *IEEE Transactions on Intelligent Transportation Systems* (2017). DOI: 10.1109/TITS.2016.2620483.

[18] Sandra G. Hart and Lowell E. Staveland. "Development of NASA-TLX". In: *Human Mental Workload*. 1988, pp. 139–183. DOI: 10.1016/S0166-4115(08)62386-9.

[19] High-Level Expert Group on Artificial Intelligence. *Ethics Guidelines for Trustworthy AI*. 2019. URL: https://op.europa.eu/en/publication-detail/-/publication/d3988569-0434-11ea-8c1f-01aa75ed71a1.

[20] Eric Horvitz. "Principles of Mixed-Initiative User Interfaces". In: *CHI*. 1999. DOI: 10.1145/302979.303030.

[21] Xiaowei Huang et al. "A Survey of Safety and Trustworthiness of Large Language Models through the Lens of Verification and Validation". In: *arXiv preprint arXiv:2305.11391* (2023). DOI: 10.48550/arXiv.2305.11391. URL: https://arxiv.org/abs/2305.11391.

[22] International Organization for Standardization. *ISO 15005: Road Vehicles – Ergonomic Aspects of Transport Information and Control Systems*. 2017. URL: https://www.iso.org/standard/63185.html.

[23] International Organization for Standardization. *ISO 15008:2017 Road Vehicles — Visual Presentation Requirements*. 2017. URL: https://www.iso.org/standard/62087.html.

[24] International Organization for Standardization. *ISO 26262: Road Vehicles – Functional Safety*. 2018. URL: https://www.iso.org/standard/68383.html.

[25] Mehshan Ahmed Khan et al. "Functional near-infrared spectroscopy (fNIRS) and Eye Tracking for Cognitive Load Classification in a Driving Simulator Using Deep Learning". In: *arXiv preprint arXiv:2408.06349* (2024). DOI: 10.48550/arXiv.2408.06349. URL: https://arxiv.org/abs/2408.06349.

[26] Sheila G. Klauer et al. *The Impact of Driver Inattention on Near-Crash/Crash Risk*. Tech. rep. NHTSA, 2006. URL: https://rosap.ntl.bts.gov/view/dot/1646.

[27] John D. Lee and Katrina A. See. "Trust in Automation". In: *Human Factors* 46.1 (2004), pp. 50–80. DOI: 10.1518/hfes.46.1.50.30392.

[28] Sergey Levine et al. "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems". In: *arXiv preprint arXiv:2005.01643* (2020). DOI: 10.48550/arXiv.2005.01643.

[29] Antonio Luque-Casado et al. "Heart Rate Variability and Cognitive Workload". In: *Psychophysiology* 56.9 (2019). DOI: 10.1111/psyp.13304.

[30] Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. 2017, pp. 1273–1282.

[31] Mercedes-Benz Group AG. *Mercedes-Benz User Experience (MBUX)*. Official product documentation. 2023. URL: https://www.mercedes-benz.com.

[32] Richard Meyes et al. "Ablation Studies in Artificial Neural Networks". In: *Proceedings of the 2019 IEEE International Conference on Industrial Technology*. 2019, pp. 1–6. DOI: 10.1109/ICIT.2019.8755058.

[33] Dimitris Mourtzis et al. "The Future of the Human–Machine Interface in Society 5.0". In: *Future Internet* 15.5 (2023). DOI: 10.3390/fi15050162.

[34] Osamu Nakayama et al. *Development of a Steering Entropy Method for Evaluating Driver Workload*. Tech. rep. 1999-01-0892. SAE International, 1999. DOI: 10.4271/1999-01-0892.

[35] National Highway Traffic Safety Administration. *Visual-Manual NHTSA Driver Distraction Guidelines for In-Vehicle Electronic Devices*. Federal Register, Vol. 78, No. 81. 2013. URL: https://www.federalregister.gov/documents/2013/04/26/2013-09883/visual-manual-nhtsa-driver-distraction-guidelines-for-in-vehicle-electronic-devices.

[36] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. Tech. rep. AI 100-1. NIST, 2023. DOI: 10.6028/NIST.AI.100-1. URL: https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf.

[37] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*. Tech. rep. AI 600-1. NIST, 2024. DOI: 10.6028/NIST.AI.600-1. URL: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf.

[38] Reinhard Oppermann. "Adaptable and Adaptive User Interfaces". In: *Human Factors in Information Technology* (1994). URL: https://dl.acm.org/doi/10.5555/647909.

[39] Fred Paas. "Training Strategies for Attaining Transfer of Problem-Solving Skill". In: *Journal of Educational Psychology* 84.4 (1992), pp. 429–434. DOI: 10.1037/0022-0663.84.4.429.

[40] Fred Paas and Jeroen J. G. Van Merriënboer. "Instructional Control of Cognitive Load". In: *Educational Psychology Review* 6.4 (1994), pp. 351–371. DOI: 10.1007/BF02213420.

[41] Miguel A. Recarte and Luis M. Nunes. "Mental Workload While Driving". In: *Journal of Experimental Psychology: Applied* 14.2 (2008), pp. 119–137. DOI: 10.1037/1076-898X.14.2.119.

[42] Andreas Riener and Alois Ferscha. "Adaptive Driver Assistance Systems". In: *IEEE Pervasive Computing* 12.1 (2013), pp. 30–39. DOI: 10.1109/MPRV.2012.83.

[43] SAE International. *Taxonomy and Definitions for Driving Automation Systems (J3016)*. 2021. URL: https://www.sae.org/standards/content/j3016_202104.

[44] J. Smith and T. Keller. “Conversational AI in Automotive Systems”. In: *IEEE Intelligent Transportation Systems Magazine* 14.3 (2022), pp. 45–57. DOI: 10.1109/MITS.2021.3101234.

[45] Selim Solmaz et al. “Vehicle-in-the-Loop Methodology”. In: *IEEE Intelligent Vehicles Symposium*. 2020. DOI: 10.1109/IV47402.2020.9304811.

[46] Constantine Stephanidis. “Toward an Information Society for All”. In: *Interacting with Computers* 13.2 (2001), pp. 107–134. DOI: 10.1016/S0953-5438(00)00032-6.

[47] David L. Strayer et al. “Cognitive Distraction While Multitasking in the Automobile”. In: *Psychology of Learning and Motivation* 54 (2011), pp. 29–58. DOI: 10.1016/B978-0-12-385527-5.00002-4.

[48] John Sweller. “Cognitive Load During Problem Solving: Effects on Learning”. In: *Cognitive Science* 12.2 (1988), pp. 257–285. DOI: 10.1207/s15516709cog1202_4.

[49] John Sweller. “Cognitive Load Theory and the Format of Instruction”. In: *Cognition and Instruction* 8.4 (1991), pp. 293–332. DOI: 10.1207/s1532690xci0804_2.

[50] John Sweller, Jeroen J. G. van Merrienboer, and Fred Paas. “Cognitive Architecture and Instructional Design”. In: *Educational Psychology Review* 10.3 (1998), pp. 251–296. DOI: 10.1023/A:1022193728205.

[51] Ashish Vaswani et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems* 30 (2017). URL: https://arxiv.org/abs/1706.03762.

[52] Jason Wei et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”. In: *NeurIPS*. 2022. URL: https://arxiv.org/abs/2201.11903.

[53] Christopher D. Wickens. “Multiple Resources and Mental Workload”. In: *Human Factors* 50.3 (2008), pp. 449–455. DOI: 10.1518/001872008X288394.

[54] Michael Wooldridge and Nicholas R. Jennings. “Intelligent Agents: Theory and Practice”. In: *The Knowledge Engineering Review* 10.2 (1995), pp. 115–152. DOI: 10.1017/S0269888900008122.

[55] Kristie L. Young et al. “Driver Distraction: A Review of the Literature”. In: *Accident Analysis & Prevention* 39.2 (2007), pp. 278–289. DOI: 10.1016/j.aap.2006.07.005.

[56] Y. Zhang, X. Li, and H. Chen. “Adaptive Human–Machine Interface Based on Multi-Objective Optimization”. In: *Applied Sciences* 13.19 (2023). DOI: 10.3390/app131910687.

[57] Yanchao Zhang et al. “Driver Workload Estimation Using Physiological Measures and Machine Learning”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.8 (2017), pp. 2070–2085. DOI: 10.1109/TITS.2016.2633200.

[58] Frieda R. H. Zijlstra. “Efficiency in Work Behaviour: A Design Approach for Modern Tools”. PhD thesis. Delft University of Technology, 1993. URL: https://repository.tudelft.nl/.

A. Additional HMI Mockups

This appendix collects additional interface mockups related to the adaptive HMI prototype. These figures are provided as supplementary visual material and complement the representative examples discussed in Chapter 5. While they are not all individually analyzed in the main text, they further illustrate the range of interface variations, proactive suggestions and multimodal interaction strategies supported by the proposed system.

A.1 Additional Layout and Widget Variants

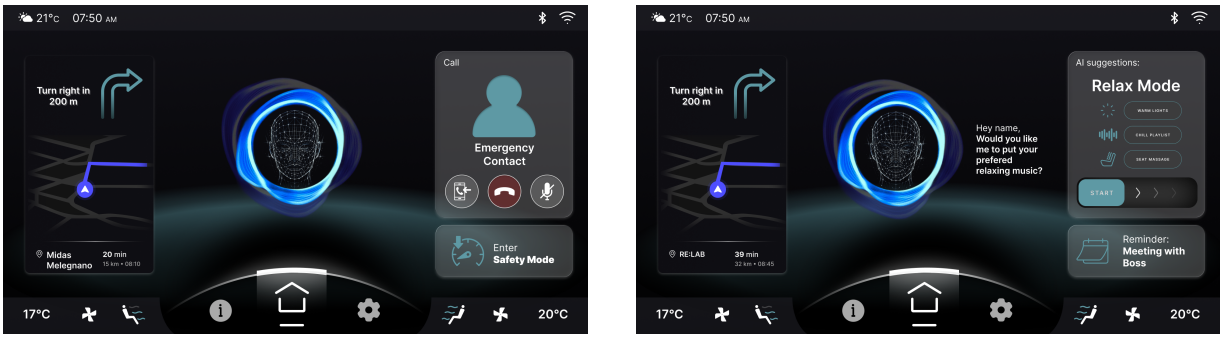

Figure A.1: Additional layout variants and widget combinations illustrating alternative interface compositions of the prototype.

A.2 Additional Safety, Monitoring and Decision-Support Screens

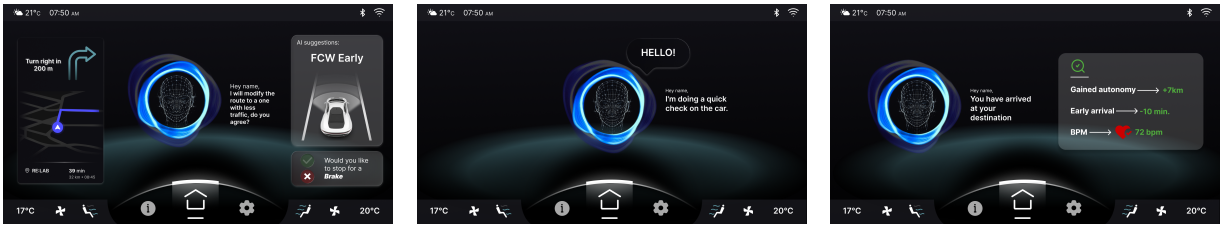

Figure A.2: Additional examples of safety-oriented intervention, vehicle-status monitoring and post-action summary presentation.

A.3 Additional Environmental and Comfort-Oriented Screens

Figure A.3: Supplementary environmental and comfort-related screens showing weather-aware support and alternative home-layout organization.

B. Representative Prompt Excerpts

This appendix reports representative prompt excerpts used by selected LLM-supported modules of the prototype. The goal is not to reproduce the full runtime strings verbatim, but to document the actual prompting strategy adopted in the implementation through compact and readable excerpts consistent with the language of the thesis.

Across the architecture, prompts are not used as generic conversational templates. They are embedded as bounded software components that specify the role of the model, the contextual variables to consider, the admissible decision criteria and, most importantly, the structure of the output expected by downstream modules. A recurring pattern is the use of explicit output contracts requiring valid JSON, constrained field names, bounded semantic values and the absence of free text outside the target object.

The excerpts reported below are therefore intended to complement the discussion provided in Chapter 5 by making concrete how language-model reasoning is integrated into the prototype under deterministic constraints, validation logic and execution-time governance.

B.1 Request Decomposition Prompt

The RequestSplitter uses an LLM prompt to transform a free-form user request into a set of atomic tasks that can be assigned to specialized agents. The prompt grounds the decomposition in the current driving context and explicitly constrains the output structure so that the result can be consumed directly by the orchestration layer.

A representative excerpt is reported below:

```
Role: EPIGNOSIS Request Splitter for an in-vehicle AI system.

Input:
- User request: "<free-form user request>"
- Context: speed, emotional state, cognitive load, scenario
```

```
Available agents:
- safety
- comfort
- productivity
- planning
- ambient_modulation
- adaptive_optimization

Task:
Decompose the request into atomic and measurable tasks.
Each task must be assigned to one target agent and must be independently
executable, unless explicit dependencies are required.

Return ONLY valid JSON in the following form:
{
"tasks": [
{
"task_id": "task_1",
"task_type": "...",
"description": "...",
"priority": "CRITICAL|HIGH|MEDIUM|LOW",
"target_agent": "...",
"dependencies": ["task_x"],
"estimated_duration_sec": 60,
"requires_user_confirmation": true
}
],
"execution_strategy": "parallel|sequential|conditional",
"reasoning": "..."
}

Critical rule: if the request is ambiguous or underspecified, generate
```

```
a task of type "clarification_needed".
```

This prompt illustrates three characteristic design choices of the implemented system: contextual grounding, explicit task-level structure and prompt-level enforcement of ambiguity handling.

B.2 Safety and Comfort Intervention Prompts

Several agent-level prompts generate bounded interventions rather than open-ended responses. In these modules, the LLM is asked to reason within a predefined operational role and to produce a structured action compatible with the internal action model.

Safety-oriented intervention

The SafetyAgent uses different prompts depending on the triggering condition, including collision warning, drowsiness detection and high cognitive load. A representative drowsiness-oriented excerpt is the following:

```
 Role: EPIGNOSIS vehicle safety agent.

Critical situation: drowsiness detected.

Context:
- cognitive load
- cognitive reserve
- scenario
- safety score
- additional driver monitoring data
- nearby rest areas, if available

Task:
Generate a safety action for drowsiness that includes:
1. immediate multimodal alert,
```

```
2. concrete suggestion such as a rest stop,
3. firm but not alarming wording.

Return ONLY valid JSON:
{
"message": "...",
"action_details": "...",
"channels": ["voice", "cluster", "ambient_light", "haptic"],
"reasoning": "..."
}
```

Comfort-oriented intervention

The ComfortAgent follows a similar pattern, but with a comfort-management objective. In the traffic-stress branch, for example, the prompt asks the model to produce an intervention that reduces stress without increasing frustration.

```
 Role: EPIGNOSIS comfort assistant.

Situation: traffic stress detected.

Context:
- emotional state and anger level
- cognitive load
- vehicle speed and congestion indicators

Available interventions:
- calming music
- short guided breathing
- ambient adjustments
- smoother alternative route, if available

Task:
```

```
Propose a multimodal intervention that reduces stress without sounding
patronizing.

Return ONLY valid JSON:
{
"message": "...",
"actions": [...],
"alternative_route": {"available": true, "extra_time_minutes": X},
"channels": ["voice", "ambient_light"],
"reasoning": "..."
}
```

These prompts show how the prototype uses LLMs to generate bounded agent proposals rather than unrestricted conversational responses. In both cases, the output is constrained by predefined channels, intervention types and scenario-dependent behavioral expectations.

B.3 Holistic Decision-Support Prompt

The HolisticDecisionAgent is designed for complex situations in which multiple critical factors are active at the same time, such as low battery, urgent appointments, heavy traffic, stress and adverse weather. Instead of generating a single direct action, the prompt asks the model to construct alternative future trajectories and to provide an explicit recommendation.

A representative excerpt is the following:

```
 Role: EPIGNOSIS holistic decision-support assistant.

Complex multi-factor situation detected.

Consider simultaneously:
- driver emotional state
- cognitive load and reserve
- traffic and weather data
```

```
- upcoming meetings
- battery autonomy
- driving complexity and safety risk

Task:
Generate three alternative timelines:
A) continue current route,
B) recommended balanced solution,
C) risky but faster alternative.

Return ONLY valid JSON:
{
"situation_summary": "...",
"timelines": [
{
"id": "timeline_a",
"name": "...",
"description": "...",
"eta_minutes": X,
"battery_at_arrival": Y,
"stress_level": "low|medium|high|critical",
"meeting_punctuality": "on_time|slight_delay|late",
"consequences": [...],
"success_probability": 0-100,
"pros": [...],
"cons": [...]
}
],
"recommended_timeline": "timeline_b",
"recommendation_reasoning": "...",
"message": "...",
"channels": ["voice", "cluster"]
```

```
}
```

This prompt is representative of the decision-support role assigned to the LLM in the proposed architecture: the model is used to synthesize structured alternatives and comparative reasoning, while the final system behavior remains embedded in the orchestration pipeline.

B.4 Adaptive UI Layout Prompt

The AdaptiveUIAgent provides the most explicit example of schema-constrained prompting in the prototype. Here the LLM does not generate a textual recommendation, but a structured user-interface layout that is later validated and normalized before being applied.

A representative excerpt is reported below:

```
Role: expert automotive UX/UI designer.

Critical rules:
1. Output ONLY valid JSON.
2. Start with { and end with }.
3. Follow the exact layout schema.
4. Use only admissible widget types.
5. Use only admissible visibility values.
6. Provide reasoning as a string.

Design principles:
- Low cognitive load: richer interface and more contextual information.
- Medium cognitive load: balanced interface.
- High cognitive load: minimal interface, essential information only.
- Critical load: safety-critical widgets only, voice priority.

Safety rules:
- Always preserve safety-relevant widgets.
- Never hide safety alerts.
- Reduce interactive widgets under distraction or drowsiness.
```

```
Conflict rules:
- navigation_map and poi_suggestions cannot coexist.
- driving_tips and poi_suggestions cannot coexist.
- If navigation is active, prioritize the navigation map.
- If cognitive load is high, remove non-essential suggestive widgets.

Return a UILayoutConfig JSON object with:
- layout_id
- complexity_mode
- cognitive_load_range
- widgets
- global_settings
- reasoning
```

The runtime version of this prompt also includes a strict schema description, a valid reference example and a serialized widget catalog. This makes the AdaptiveUIAgent the clearest instance of bounded LLM generation followed by deterministic post-processing and validation.

B.5 Priority Refinement and Quality Evaluation Prompts

Not all prompts in the prototype are used to generate executable actions directly. Some are used to estimate bounded decision variables or to supervise candidate actions before execution.

Priority refinement

In the PriorityEngine, the LLM is used to estimate the cognitive fit of an already proposed action. The output is a compact structured assessment that is then combined with deterministic terms such as safety relevance, urgency, user value and context alignment.

```
Task: evaluate the cognitive fit of a candidate action.

Input:
```

```
- action type
- message
- channels
- originating agent
- cognitive load
- cognitive reserve
- scenario
- safety score

Question:
How cognitively appropriate is this action right now?

Consider:
- whether the action requires additional attention,
- whether it reduces workload,
- whether it matches the current scenario.

Return ONLY valid JSON:
{
"cognitive_fit_score": 0.0-1.0,
"reasoning": "brief explanation"
}
```

Quality evaluation

The AssistantEvaluation module uses a second structured prompt to assess whether an action should be approved, rejected or modified before execution.

```
 Role: EPIGNOSIS Quality Control Inspector.

Evaluate the candidate action with respect to:
- safety
- appropriateness
```

```
- clarity
- actionability
- user value

Return ONLY a single valid JSON object:
{
"approved": true/false,
"quality_score": 0.0-1.0,
"dimension_scores": {...},
"issues": [...],
"suggestions": [...],
"reasoning": "...",
"recommended_action": "approve|reject|modify"
}

Critical rule: if safety < 0.6, the action must be rejected.
```

Together, these prompts illustrate that the LLM is also used as a bounded evaluative component inside the orchestration layer, rather than only as a generative one.