ChatGPT

Self-Adaptive System Frameworks

Self-adaptive software systems use feedback loops to monitor and adjust their behavior under changing conditions. The classic **MAPE-K** loop (Monitor–Analyze–Plan–Execute with shared Knowledge) remains a foundation, but it is known to be sequential and reactive. Recent work highlights its limitations for highly dynamic environments, and new architectures have been proposed. For example, **AWARE** (Assess–Weigh–Act–Reflect–Enrich) replaces the linear MAPE-K loop with a *distributed, goal-driven* framework of AI agents. In AWARE, autonomous agents proactively **adapt and learn**, collaborating to improve decision-making and resilience, while dynamically integrating new agents and resources [1] [2]. Similarly, researchers have extended MAPE-K with meta-learning layers. In **lifelong self-adaptation**, a "lifelong ML" layer continually tracks the running system, detects emerging tasks, and updates adaptation models on the fly [3] [4]. This approach leverages transfer learning to adapt policies over time without human intervention. Other work combines **MAPE and control theory**: for instance, the DYNAMICO model decomposes adaptation into multiple loops (requirements, context, control) using discrete MAPE planners at high levels and continuous controllers at low levels [5]. In this view, the upper layers ensure goal satisfaction while embedded PID-like controllers maintain stability and fast reaction. These hybrid architectures trade off the interpretability and high-level reasoning of MAPE-K against the speed and stability of formal control.

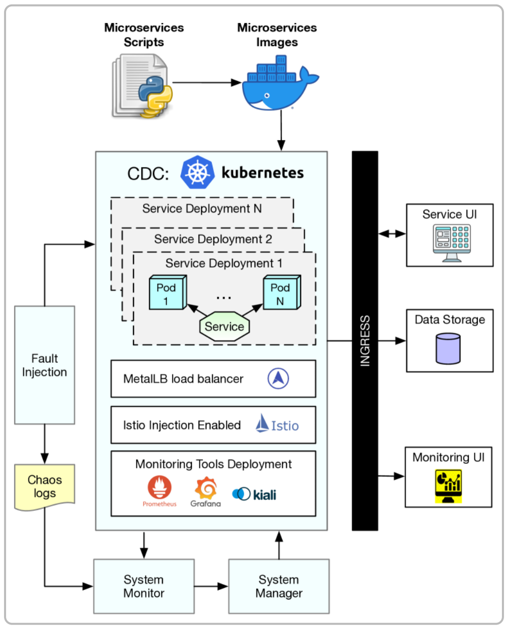

Figure: The CHESS framework (Chaos Engineering for Self-Adaptive Systems) illustrates a MAPE-K-style architecture. Microservices run in a Kubernetes cluster with monitoring tools (Prometheus, Grafana, etc.), while a System Monitor observes runtime behavior and records Chaos fault injections. A System Manager then enacts recovery rules. By triggering faults and watching how the system recovers, CHESS evaluates adaptation resilience [6] [7]*.*

Key SOTA frameworks and techniques in self-adaptation include:
- **MAPE-K Variants:** Extensions of the classical loop (e.g. human-in-the-loop, goal-model guided MAPE) that

improve planning or interaction. For example, MAPE-K–HMT augments the loop for human–machine teaming.
- **Distributed/Multi-loop Architectures:** AWARE uses multiple interacting agents in parallel (rather than one central loop) [1]. DYNAMICO's multi-loop decomposes goals, context, and control [5].
- **Learning-Enabled Adaptation:** Machine learning can reduce adaptation space or detect anomalies. Lifelong self-adaptation adds a continuous learning layer to handle concept drift [3] [4]. Other approaches learn controllers or prediction models at runtime.
- **Formal & Runtime Models:** Some work emphasizes formal guarantees. Hybrid models combine MAPE-K with formal controllers to verify properties at runtime. LLMs have even been used to **generate system models** (requirements or architecture) that could serve as internal representations [8]. For example, LLMs can translate natural-language requirements into domain-specific models, hinting at automated runtime-model synthesis [8].

Each framework introduces trade-offs. Centralized MAPE-K is simple but can be slow to adapt; distributed agent frameworks (like AWARE) scale better but add complexity. Adding ML layers enables *continuous evolution* but raises challenges in validation and trust. Researchers synthesize these by layering approaches: fast *control-theoretic loops* for immediate adjustments and higher-level *planning/learning loops* for strategic adaptation [5]. In all cases, **monitoring and feedback** remain core: the system senses its state and environment, analyzes deviations, plans mitigations, and executes changes (the essence of MAPE) [5] [6]. Formal verification is often done on abstract "runtime models," and one emerging idea is to treat a learned model (e.g., an LLM-based world model) as a kind of digital twin for analysis. Initial experiments show LLMs can simulate environmental scenarios (e.g. driving scenes) to test uncertainty [9], though aligning such simulations with reality remains an open challenge.

Agentic AI and Multi-Agent Systems

An **agent** is generally defined as a (possibly autonomous) computational entity with **sensory inputs and actuators**. It perceives its environment, pursues goals, interacts with others, and **can use tools or APIs** as part of its actions [10] [11]. Modern "agentic" AI often centers on large language models (LLMs) as the *cognitive core*. In an LLM-based agent, the model (e.g. GPT-4 or Gemini) provides knowledge and reasoning ability [12]. The agent may maintain memory of past interactions, process new inputs, and decide on actions, which can range from writing code to calling external functions [11]. For instance, one formal model describes an LLM-agent as a tuple that includes the LLM model, an objective (goal), memory, perception module, and an action mechanism (which explicitly allows "utilizing tools" and inter-agent communication) [12] [11]. Agents thus blend symbolic planning with data-driven intelligence, seeking goals in dynamic environments.

The latest SOTA agent frameworks exploit this flexibility. Researchers have demonstrated **LLM-enhanced multi-agent systems** where each agent (e.g. a chatbot or code generator) collaborates to solve complex tasks. For example, Nascimento *et al.* integrated GPT-4 into a Multi-Agent System using a MAPE-K backbone: agents use the LLM for advanced analysis and planning steps, enabling them to tackle more complex tasks and coordinate adaptively [13]. In practice, these LLM-based agents can decompose problems, debate solutions, and refine each other's outputs. A broad survey finds that multi-agent LLM systems improve robustness (agents cross-check to avoid hallucination) and scalability (new agents can be added for new expertise) [14] [15].

Recent taxonomies distinguish “AI Agents” (single-model systems extended with tool use and reasoning chains) from “Agentic AI” (networks of specialized agents) [16] [15]. AI Agents like AutoGPT enhanced LLMs with API calls, function chaining, and RAG (retrieval-augmented generation) to perform open-ended tasks [16]. Agentic AI systems go further: they comprise **multiple specialized agents** that coordinate, decompose goals dynamically, and maintain shared memory [15] [17]. For example, Agentic AI might use one LLM agent for planning, another for checking, and another for acting, all communicating to fulfill a high-level objective. This mirrors human teams: separate experts work in tandem. Such agentic systems promise greater autonomy and adaptability, but also raise new challenges in communication and trust.

Key trends in agentic technology relevant to self-adaptation include: - **Tool-Augmented LLMs:** Modern agents invoke external tools (web search, code engines, simulators) to augment decision-making [16].
- **Memory & Reflection:** Agents may keep logs or summaries of past reasoning and explicitly “rethink” after actions [18].
- **Multi-Agent Coordination:** Agents can form orchestrated workflows. Surveyed architectures support various models (centralized, decentralized, hierarchical) and coordination protocols [19].
- **Learning & Autonomy:** Beyond LLM’s knowledge, agents can be fine-tuned or trained via reinforcement to improve performance. For example, Agentic AI can incorporate RL loops to refine strategies.

Together, these agentic innovations suggest new design ideas for adaptation frameworks. One can imagine an adaptive system made of plug-in agents: each component (whether a microservice or an AI module) is wrapped as an agent with a standard I/O and tool interface. The *adaptation mechanism* itself could be an ensemble of LLM agents that monitor system logs, analyze anomalies in natural language, plan reconfiguration steps, and verify outcomes. The SOTA clearly shows that agents are not just abstract entities but can be realized with today’s LLM and multi-agent technology to build flexible, extensible adaptive systems.

Hybrid and Future Directions

Recent proposals advocate *hybrid frameworks* that layer classical adaptation control, AI-driven reasoning, and meta-adaptation. A promising architecture has: 1. **Fast Control Layer:** A low-level layer based on control theory or finite-state logic provides immediate reactions to perturbations. For example, Proportional-Integral-Derivative (PID) controllers or rule-based managers can quickly adjust system parameters (scaling servers, resetting components) to stabilize critical variables [5].
2. **Conceptual AI Layer:** Above that, an LLM-powered layer performs deeper reasoning. It can diagnose trends, refine goals, or interpret ambiguous sensor data. LLMs excel at handling uncertainty by leveraging knowledge and reasoning in language. For instance, an LLM could read the system’s logs, infer emerging issues (“the network latency pattern looks like X”), and suggest new adaptations. Early work uses LLMs to *generate and refine goal models* under the MAPE-K framework [20] [8].
3. **Meta-Learning Layer:** A topmost layer observes the adaptation process itself and tunes long-term policies or objectives. This is akin to “lifelong learning”: the system tracks its successes and failures over time and *updates its own goals*. The lifelong self-adaptation architecture [3] is an example, where a meta-layer identifies when the domain has shifted (e.g. new workload patterns) and retrains the adaptation model accordingly.
4. **Learned Internal Model:** Parallel to these, the system could maintain a **learned runtime model**. Instead of relying only on static design-time models, the system uses ML to build an internal representation of itself or its environment. For example, an LLM trained on historical data could act as a “world model” that

simulates how the system would behave under hypothetical changes [9]. This learned model can then feed into formal verification or planning: essentially a digital twin that evolves over time.

Key principles emerge from the state-of-the-art: **modularity**, **feedback**, and **adaptivity across levels**. Modular agent design lets us plug in new adaptation components without rearchitecting the whole system. Feedback loops (of various kinds) ensure continuous self-monitoring. And by combining symbolic/hard guarantees (control theory, formal models) with data-driven adaptation (ML, LLMs), the system can be both robust and flexible. For formal verification, one could leverage the learned model as a basis for checking safety: e.g. extract an abstract state machine from an LLM's knowledge to prove properties. Although this is still speculative, it aligns with the trend of using AI to **automate model generation** [8].

In summary, self-adaptive frameworks have evolved from simple loops to multi-layer architectures that integrate planning, control, and learning. At the same time, agentic AI offers a rich design space of intelligent, tool-using agents. A next-generation adaptation system will likely treat its components (including ML/LLM modules) as black-box agents with well-defined I/O, and orchestrate them in a layered feedback loop. The **core principle** is to balance *reactivity* (fast control actions) with *reasoning* (LLM-mediated planning) and *continual learning* (meta-level adaptation). By drawing on the SOTA in both self-adaptive systems (e.g. MAPE-K variants, control-theoretic loops) and agentic AI (LLM agents, multi-agent coordination), designers can craft hybrid frameworks that are software-agnostic yet capable of sophisticated goal-driven behavior [3] [17].

Sources: We surveyed recent peer-reviewed works and industry reports across SEAMS, ICSE, ACSOS, etc., focusing on the last three years. Key references include Weyns *et al.* on MAPE/Control integration [5], Gheibi *et al.* on lifelong learning for adaptation [3] [4], Nascimento *et al.* on GPT-4 in multi-agent MAPE-K systems [13], and recent surveys on LLMs in self-adaptation [20] [8]. These illustrate how state-of-the-art frameworks leverage layered feedback, learning, and agent collaboration. All cited works are drawn from the connected literature.

---

[1] [2] Breaking the Loop: AWARE is the New MAPE-K (FSE 2025 - Ideas, Visions and Reflections) - FSE 2025
https://conf.researchr.org/details/fse-2025/fse-2025-ideas-visions-and-reflections/22/Breaking-the-Loop-AWARE-is-the-New-MAPE-K

[3] [4] Lifelong Self-Adaptation: Self-Adaptation Meets Lifelong Machine Learning
https://people.cs.kuleuven.be/~danny.weyns/papers/2022SEAMSc.pdf

[5] arxiv.org
https://arxiv.org/pdf/2103.10847

[6] [7] [2303.07283] CHESS: A Framework for Evaluation of Self-adaptive Systems based on Chaos Engineering
https://ar5iv.org/pdf/2303.07283

[8] [9] [20] Exploring the Potential of Large Language Models in Self-adaptive Systems
https://arxiv.org/pdf/2401.07534

[10] [11] [12] [14] [18] [19] LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead
https://arxiv.org/html/2404.04834v4

13 arxiv.org
https://arxiv.org/pdf/2307.06187

15 16 17 AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges
https://arxiv.org/html/2505.10468v4