Software-Defined Ubiquitous Operating System: A New Paradigm for Human–Cyber–Physical Systems

Yunfan Cao, Chaoyi Zhao, Hanzhi Liu, Jiayi Wang, Deyu Kong, Huiyan Wang, Ping Yu, Chun Cao, Chang Xu, Xiaoxing Ma, Yanyan Jiang
State Key Laboratory for Novel Software Technology at Nanjing University
School of Computer Science, Nanjing University
Nanjing, China

Abstract

The convergence of the Internet of Things (IoT) and Artificial Intelligence (AI) has ushered in an era of ubiquitous computing, creating immense opportunities for intelligent Human-Cyber-Physical Systems (HCPS). However, developing such applications is hindered by fundamental challenges: managing massively heterogeneous resources, decomposing complex and evolving requirements, and integrating probabilistic AI models with deterministic software logic. To address this, we propose a software-defined Ubiquitous Operating System (UOS), a novel paradigm designed for the unique demands of ubiquitous computing. Our UOS is built on three core principles: (1) a Software Twin to enable unified management of heterogeneous resources through abstraction; (2) a Reporting and Commanding architecture to hierarchically decompose application complexity; and (3) an AI-Native design to seamlessly integrate intelligence and manage uncertainty. We demonstrate its potential in three representative: an OS Intelligent Assistant, the "Tianwang" Urban-scale Monitoring System, and a national disease prevention system, demonstrating its capacity to significantly simplify the development of robust and intelligent ubiquitous applications.

Keywords: Ubiquitous Computing, Ubiquitous Operating System, Software Engineering

1 Introduction

The proliferation of the Internet of Things (IoT) is creating a new class of human-cyber-physical systems (HCPS) [6], from smart cities to intelligent healthcare [13]. However, their development is hindered by a new "software crisis" stemming from three principal challenges:

1. **Heterogeneous Resource Management:** Ubiquitous environments are characterized by vast, dynamic, and heterogeneous resources (e.g., sensors, actuators) for which traditional OS abstractions like files and processes are inadequate [3, 10].
2. **Complex Requirement Decomposition:** HCPS involve complex, high-level goals (e.g., "optimize city traffic") that are difficult to decompose into manageable tasks for distributed components.
3. **AI Integration and Uncertainty:** There is a huge gap between the probabilistic nature of AI models (e.g., LLMs) outputs and the precise formal semantics of programming languages. Systems must manage the inherent uncertainty from AI decisions.

Traditional OS abstractions struggle to address these challenges effectively. While some middleware and frameworks exist, they are usually limited to specific functions and lack a system-level perspective. Following the historical "20-year cycle" of OS evolution [2, 7], we argue that the era of ubiquitous computing necessitates a new OS paradigm: the Ubiquitous Operating System (UOS) [8].

This paper introduces a software-defined UOS paradigm designed to provide foundational support for intelligent HCPS applications. It is based on three pillars: using a Software Twin concept to abstract resources, a Reporting and Commanding architecture to manage complexity, and an AI-Native design to seamlessly integrate intelligence [12, 18].

2 The Ubiquitous Operating System

Building on these insights, we propose the UOS as an overarching framework for managing HCPS. Its architecture is organized around three core principles, as illustrated in Figure 1.

2.1 Fundamental Capability: Software Twin

To tackle resource heterogeneity, the UOS establishes a Software Twin which is a real-time, digital representation of the entire HCPS environment [4]. It creates a holistic and unified view of the system's state, encompassing all physical devices, software components, and human users. This approach shifts the programming paradigm from "building features to meet requirements" to "making decisions based on total system data", a vision shared by context-aware computing research [11, 16].

Instead of programming against low-level, device-specific APIs, developers program against this holistic and unified state. The UOS provides the underlying mechanisms (e.g., leveraging technologies like eBPF or JVM TI for non-intrusive instrumentation) to construct and maintain this twin. Applications can then query, filter, and react to patterns in this global state, effectively extracting useful information they need from the whole, thus simplifying development and enabling powerful and context-aware behaviors.

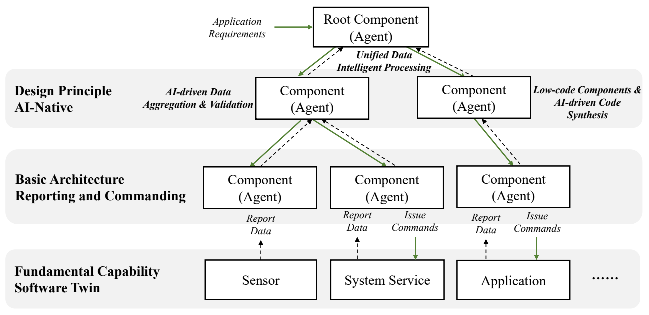

Figure 1. The architecture of the UOS. The Software Twin provides a unified view of underlying resources. The Reporting and Commanding architecture organizes the system into a hierarchy of agents. The AI-Native design principle governs data flow and enables intelligent processing at each level.

2.2 Basic Architecture: Reporting and Commanding

To manage application complexity, the UOS organizes applications into a hierarchical structure of cooperative components, which we term ubiquitous agents [14, 15]. This design is inspired by the effective organizational structures found in human society and is a departure from the flat, broadcast-based communication models (e.g., Android's Intents) that lead to quadratic complexity growth.

This structure defines a clear communication pattern:

- Reporting: Leaf agents, representing sensors or data sources, report their state upwards. Parent agents aggregate and abstract the data from their children, providing a summarized view to the next level up.
- Commanding: Higher-level agents decompose tasks and issue commands to their children. This flow continues down the tree until it reaches leaf agents controlling physical actuators or software APIs.

At the system support level, the UOS should be responsible for managing the streams of data and commands between agents. It provides a disciplined way to structure applications, manage data flow, and naturally decompose complex problems.

2.3 Design Principle: AI-Native

To overcome the challenge of uncertainty, the UOS is designed to be AI-Native. This principle standardizes inter-agent communication to make data flows inherently intelligible to AI models. We propose a universal data unit based on a four-element tuple: time (when), location (where), subject (who), and event (what).

This structured format enables LLMs to directly interpret system behavior, facilitates low-code development where models can synthesize component logic [5], and supports adaptive handling of uncertain or erroneous data [17]. It also allows for integrating advanced model management frameworks like Learnware [18] for scheduling the right model for each sub-task.

3 Application Cases

We demonstrate the UOS design's effectiveness through three HCPS scenarios.

OS Intelligent Assistant. Unlike current OS copilots limited to first-party apps [1, 9], a UOS-based assistant can handle complex commands like "Create a diary of my trip last week". A root agent decomposes this task, issuing commands to agents for photos, location, and calendar services. These agents query the Software Twin, report relevant data, and the root agent aggregates the results. Component reuse is maximized. For example, a single face recognition component can serve multiple applications like photo galleries and security systems.

"Tianwang" Urban-scale Monitoring System. In a UOS-based "Tianwang" system, the Software Twin unifies countless cameras and IoT sensors into a city-state view. A traffic management application can issue a high-level command like "reduce congestion on main street," which is decomposed down to components managing traffic lights and navigation services. These components collaborate using real-time data reported from sensor agents. The data and components are reusable for other applications like urban planning or emergency response.

Disease Prevention and Control. A UOS can power a national public health system. The Software Twin would integrate fragmented data (from hospitals, wearables, etc.) into a real-time population health view. The Reporting and Commanding Architecture enables an efficient data pipeline for health reporting and public alerts. The AI-Native design allows predictive models to detect outbreaks and dynamically generate software components for new public health protocols.

4 Conclusion

The ubiquitous computing era requires a new paradigm for constructing HCPS. We proposed the UOS to address the core challenges of heterogeneity, complexity, and AI integration in HCPS. Its three principles: Software Twin, Reporting and Commanding architecture, and AI-Native design, offer a coherent framework for building future intelligent and adaptive systems. Our future work will focus on tooling for migrating existing software to the UOS model and exploring solutions for security, privacy, and performance in this paradigm.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant No. 2022YFB4501800).

References

[1] Apple Inc. 2024. Apple intelligence. https://www.apple.com/hk/en/apple-intelligence Accessed: 2024-10-21.

[2] Gordon Bell. 2008. Bell's law for the birth and death of computer classes. *Commun. ACM* 51, 1 (2008), 86–94.

[3] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair. 2011. Distributed Systems: Concepts and Design. *Addison-Wesley Publishing Company* (2011).

[4] Ziqi Huang, Yang Shen, Jiayi Li, Marcel Fey, and Christian Brecher. 2021. A Survey on AI-driven digital twins in Industry 4.0: smart manufacturing and advanced robotics. *Sensors* 21, 19 (2021), 6340.

[5] Cong Li, Yanyan Jiang, and Chang Xu. 2022. Push-button synthesis of watch companions for android apps. In *Proceedings of the 44th International Conference on Software Engineering (ICSE)*. 1793–1804.

[6] Zhiming Liu and Ji Wang. 2020. Human-cyber-physical systems: concepts, challenges, and research opportunities. *Frontiers of Information Technology & Electronic Engineering* 21, 11 (2020), 1535–1553.

[7] Hong Mei, Donggang Cao, and Tao Xie. 2022. Ubiquitous Operating System: Toward the Blue Ocean of Human–Cyber–Physical Ternary Ubiquitous Computing. *Bulletin of Chinese Academy of Sciences* 37, 1 (2022), 30–37.

[8] Hong Mei and Yao Guo. 2018. Toward ubiquitous operating systems: a software-defined perspective. *Computer* 51, 1 (2018), 50–56.

[9] Microsoft. 2024. Windows Copilot. https://www.microsoft.com/en-us/microsoft-copilot/personal-ai-assistant Accessed: 2024-10-21.

[10] Rob Pike, Dave Presotto, Ken Thompson, Howard Trickey, et al. 1995. Plan 9 from Bell Labs. *Computing Systems* 8, 2 (1995), 221–254.

[11] Huiyan Wang, Chang Xu, Bingying Guo, Xiaoxing Ma, and Jian Lu. 2019. Generic adaptive scheduling for efficient context inconsistency detection. *IEEE Transactions on Software Engineering (TSE)* 47, 3 (2019), 464–497.

[12] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[13] Mark Weiser. 1999. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)* 3, 3 (1999), 3–11.

[14] Michael Wooldridge. 2009. An Introduction to Multiagent Systems. Second Edition. *Wiley Publishing* (2009).

[15] Michael Wooldridge and Nicholas R. Jennings. 1995. Intelligent agents: theory and practice. *The Knowledge Engineering Review* 10, 2 (1995), 115–152.

[16] Chang Xu, Shing Chi Cheung, Wing Kwong Chan, and Chunyang Ye. 2010. Partial constraint checking for context consistency in pervasive computing. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 19, 3 (2010), 1–61.

[17] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2024. Generalized out-of-distribution detection: a survey. *International Journal of Computer Vision (IJCV)* 132, 12 (2024), 5635–5662.

[18] Zhi-Hua Zhou. 2016. Learnware: on the future of machine learning. *Frontiers of Computer Science* 10, 4 (2016), 589–590.