Scientific Journal of King Faisal University:
Basic and Applied Sciences

Enhancing Interoperability in the Web of Things: A Reference Architecture Approach

Khalied M. Albarrak
Department of Management Information Systems, College of Business Administration, King Faisal University, Al-Ahsa, Saudi Arabia

LINK	RECEIVED	ACCEPTED	PUBLISHED ONLINE	ASSIGNED TO AN ISSUE
https://doi.org/10.37575/b/sci/240034	02/08/2024	18/12/2024	18/12/2024	01/06/2025

NO. OF WORDS	NO. OF PAGES	YEAR	VOLUME	ISSUE
5327	6	2025	26	1

ABSTRACT

The concept of the Internet of Things (IoT) has fundamentally redefined the connectivity landscape by integrating physical devices with the Internet for data exchange. The Web of Things (WoT) takes this integration a step further by embedding IoT devices within the web, thus facilitating their interoperability. However, the inherent complexity of WoT architecture requires a layer of abstraction to simplify development and integration. Patterns and reference architectures (RAs) provide such an abstraction, modeling solutions for recurrent problems within this domain. In this paper, we aim to develop an RA for the WoT ecosystem. We use architectural patterns to build the RA of the WoT. Here, we present the WoT gateway pattern, which provides a clear and adaptable framework for integrating diverse IoT devices, ensuring seamless communication and functionality within the broader web ecosystem. The purpose of this pattern is twofold: to ensure the interoperability of WoT implementations and to serve as a foundational guide for developers navigating the multifaceted challenges of WoT system design. The pattern also paves the way for building an RA for IoT environments.

KEYWORDS
Architectural patterns, Internet of Things, reference architecture, web development, web science, WoT

CITATION
Albarrak, K.M. (2025). Enhancing interoperability in the web of things: A reference architecture approach. *Scientific Journal of King Faisal University: Basic and Applied Sciences*, **26**(1), 7–12. DOI: 10.37575/b/sci/240034

1. Introduction

The Internet of Things (IoT) has revolutionized how we interact with the physical world through interconnected devices and systems. It encompasses a wide array of technologies that enable devices to collect, exchange, and act on data, often with minimal human intervention. However, as the IoT landscape expands, it has also become a complex web of proprietary systems with significant interoperability challenges. To address these issues, the Web of Things (WoT) has emerged as a unifying layer designed to enable interoperability across various IoT platforms and domains by leveraging web technologies.

The WoT seeks to create a cohesive ecosystem where IoT devices can communicate seamlessly, regardless of the diverse protocols and standards they may employ. Its integration into IoT architecture introduces a new level of standardization and interaction, allowing devices to effectively communicate, share, and act upon data. This approach is expected to mitigate existing IoT challenges, such as siloed ecosystems (Nedeltcheva and Shoikova, 2017) and lack of uniformity (Achirei *et al.*, 2020), by providing a common ground that fosters innovation and simplifies the development of IoT solutions. The application of WoT is particularly significant in ensuring that as the number of IoT devices grows, they can operate in harmony, creating more intelligent and responsive environments.

This paper introduces a pattern for WoT architecture as a foundational step toward building a reference architecture (RA) for the WoT ecosystem. The intended audience includes system architects and system designers. Patterns have been instrumental in modeling various virtualized environments (Alnaim *et al.*, 2019; Alwakeel *et al.*, 2019a; Syed and Fernandez, 2018), offering solutions to recurring design and architectural challenges. They address common problems such as flexibility (Alnaim *et al.*, 2019), heterogeneity (Fernandez and Hamid, 2015), and elasticity (Syed *et al.*, 2016). Furthermore, RAs have proven effective as tools for abstracting complex systems that often lack clear semantics.

Implementing an RA for WoT aids in understanding the system's functionality and identifying potential vulnerabilities. Over time, this RA could evolve into a security reference architecture (SRA) by incorporating misuse and security patterns addressing potential threats and vulnerabilities within the system. Despite this potential, a noticeable gap exists in the application of patterns to WoT. While IoT architectures provide general frameworks, WoT's unique challenges require more precise modeling.

The primary contribution of this paper is the development of a comprehensive RA for the WoT. This RA simplifies the integration and management of IoT devices across different platforms and protocols while enhancing interoperability and scalability. The WoT architecture defines an abstract architecture based on modular building blocks applicable across diverse application domains. This abstract architecture, as described by the World Wide Web Consortium (W3C), is not prescriptive but descriptive, emphasizing interoperability and complementing existing IoT standards rather than replacing them (Lagally *et al.*, 2023).

The remainder of the paper is structured as follows: Section 2 provides the background of WoT architecture, patterns, and RA. Section 3 presents a pattern for WoT. The paper concludes with insights and directions for future research.

2. Background

2.1. Web of Things

WoT architecture, developed by the W3C, is a conceptual framework designed to foster interoperability among disparate IoT devices and platforms (Lagally *et al.*, 2023). At its core, WoT aims to integrate IoT with the World Wide Web, establishing a uniform way to communicate across various protocols and data formats. The architecture leverages web standards to create a common interface, with the goal of making IoT devices as accessible and interactive as web pages and services. This approach enables developers to build cross-platform applications and facilitates efficient communication between devices, regardless of their underlying hardware or software specifications.

The WoT architecture comprises several key components (Lagally *et*

Corresponding Author: Khalied M. Albarrak 00966135895800, kalbarrak@kfu.edu.sa

al., 2023). At the forefront are WoT Thing Descriptions (TDs), which serve as the IoT equivalent of web pages. TDs provide a standardized, machine-readable format for describing the metadata, properties, interaction affordances, security requirements, and events associated with IoT devices. Another critical component is the WoT Binding Template, which extends TD functionality by defining how the abstract interactions described in TDs are implemented over various communication protocols. This ensures that devices not only communicate but do so in a consistent and comprehensible manner across diverse network environments.

Complementing these components are WoT Discovery mechanisms, which facilitate the dynamic detection and integration of devices within the IoT ecosystem. These mechanisms enable the automatic identification and configuration of new devices as they join the network, significantly simplifying the process of expanding and maintaining IoT systems. Together, these components form the backbone of the WoT framework, promoting a seamless, standardized, and scalable approach to IoT device interoperability and management.

2.2. Patterns

A pattern is essentially a solution to a recurring problem within a specific context (Buschmann *et al.*, 2001). Software patterns, such as design and architectural patterns, are instrumental in developing systems that are both flexible and extensible. Security patterns focus on constructing secure systems by detailing methods to manage threats, address vulnerabilities, and implement necessary security measures (Fernandez *et al.*, 2016). Similarly, misuse patterns offer insights into how attacks are executed from the attacker's perspective. They specify the conditions conducive to an attack, the required security measures, and methods for gathering forensic data post-incident (Alnaim, 2022).

In this paper, patterns are used to define the principal components of the WoT. Patterns serve as a robust mechanism for articulating comprehensive solutions that encompass not only software but also hardware and physical elements that collectively form ecosystems. These patterns are typically structured as templates with designated sections. In our approach, we adopt the POSA (Pattern-Oriented Software Architecture) template (Fernandez, 2013). The documentation of these patterns may include UML modeling techniques and formal language descriptions to ensure clarity and precision.

2.3. Reference Architecture

A RA serves as a conceptual blueprint for one or more domains, focusing on architectural aspects without addressing specific implementations (Angelov *et al.*, 2012; Avgeriou, 2003 and Cloutier *et al.*, 2010). An RA is designed to outline the core components of a system and their interactions, providing an architectural framework tailored to a specific domain. Key attributes that enhance the utility of RAs include configurability, extendibility, and reusability (Avgeriou, 2003). Beyond class and sequence diagrams, an RA may include a collection of use cases (UCs) and a set of roles (R) corresponding to its stakeholders or actors (Pankowska, 2015).

RAs can be categorized into various types, such as those for the technology domain, which details platforms and design artifacts (Angelov *et al.*, 2012); those for the application domain, which describe different types of applications; and those for the problem domain, which are similar to domain models but tailored for software solutions. Stakeholders of an RA may include groups, individuals, organizations, and systems that have a vested interest in the system and influence its design and development (Avgeriou, 2003). To enhance its security features, an RA can be transformed into an SRA by integrating security patterns that address and mitigate identified threats (Avgeriou, 2003 and Alnaim *et al.*, 2022).

3. Related Work

Numerous studies have been conducted to develop and refine the architecture of WoT. Guinard *et al.* (2010) detailed a resource-oriented architecture rooted in RESTful principles. Zhang, Cheng, and Ji (2012) introduced a Social WoT framework that merges RESTful web services with social networking elements. Guinard (2011) further advanced the field by proposing a four-layered WoT application architecture aimed at streamlining the development of applications involving smart devices. He also illustrated client–thing interaction through a sequence diagram, though his proposed architecture could benefit from a deeper exploration of its components and their interconnections. Mainetti *et al.* (2015) offered an architectural approach that includes mechanisms for discovering devices and virtualizing them outside their physical network. However, these architectures tend to be conceptual and lack detailed semantics.

Additionally, Manta-Caro *et al.* (2024) discussed the new opportunities and challenges brought by IoT and WoT, particularly in the field of information retrieval. They proposed architectural solutions to manage the vast data generated by interconnected devices. The use of architectural patterns has also become prevalent in various technological contexts, providing targeted solutions within specific ecosystems. Alnaim *et al.,* (2019) and Fernandez and Hamid (2015) applied patterns to model the network function virtualization architecture. Syed *et al.* (2016a) utilized patterns within Fog Computing to address particular challenges, while Hashizume *et al.* (2012) extensively employed patterns to tackle architectural issues in cloud environments.

4. A Pattern for Web of Things Gateway

- **Intent:** To enable interoperability and standardized interaction among diverse IoT devices and services in various application domains.
- **Context:** Within the IoT ecosystem, devices and services must communicate effectively regardless of their underlying implementations and across multiple networking protocols.
- **Example:** John's home contains various IoT devices, including a Bluetooth-enabled smart door lock, a Wi-Fi-connected TV, and smart lighting that operates on ZigBee. The diversity in communication protocols means each device requires its own specific application for control, complicating the management process. Currently, there is no unified standard that allows all these devices to be controlled seamlessly through a single application or platform.
- **Problem:** IoT devices vary widely in their implementations and communication protocols, creating a heterogeneous ecosystem. To facilitate smooth and effective interaction between these devices, it is crucial to ensure interoperability across this diverse landscape.

  The solution is influenced by the following forces:

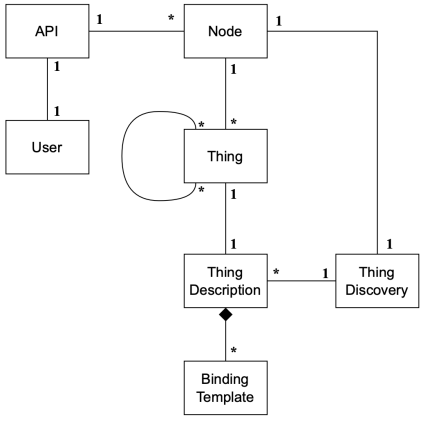
  1. **Compatibility:** Maintaining compatibility among an ever-growing number of IoT devices is a significant challenge. As new models and versions are released, they must communicate and function with older devices without requiring frequent upgrades or replacements.
  2. **Scalability:** As IoT ecosystems expand in terms of both the number of devices and the volume of data generated, scalability becomes a critical concern.
  3. **Interoperability:** IoT devices vary in their platforms and protocols. Ensuring seamless interoperability among heterogeneous devices remains a core challenge.
  4. **Discoverability:** In an IoT ecosystem, devices frequently join and leave the network. Automatic discovery and integration of new devices are essential for maintaining an up-to-date and responsive system.
  5. **Resource Constraints:** IoT devices often have limitations in processing power, memory, and energy, which complicates the implementation of complex communication protocols

and data processing tasks.

6. **Security:** IoT devices often collect and transmit sensitive data. Ensuring security and privacy is crucial, but the diversity of devices and protocols makes it difficult to implement uniform security measures.

- **Solution**: Utilize the WoT building blocks, which support describing network interfaces for IoT devices and services, define communication protocols, and facilitate the discovery, consumption, and exposure of IoT devices ("Things").
- **Structure**: Figure 1 presents a UML class diagram for WoT architecture. A "Thing" represents one or more physical IoT devices within a network responsible for collecting data and performing actions. Things can interact with other Things. A node device, which could be an edge, fog, or cloud server, manages the interactions of multiple Things. Each Thing is described by its own TD, which provides a machine-readable vocabulary for defining the physical device. TDs can be implemented over multiple protocols, each requiring a different Binding Template.
  The WoT Discovery mechanism manages TDs for various devices. When a device (Thing) is discovered, its TD is retrieved by the WoT Discovery component, allowing clients or services to understand the capabilities and interaction affordances of the discovered Thing. The WoT Discovery component also communicates with the Node to register new devices joining the network and to query devices based on specific criteria.
  In this case, the Node may act as a client to the discovery service when searching for new devices to manage or integrate. Users can send commands to Things via the API, enabling interaction and control within the IoT ecosystem.

Figure 1. Class diagram of the WoT pattern



- **Dynamics**: This section delves into the operational aspects of the WoT architecture to elucidate how the various elements of the WoT architecture interact and coordinate in practical scenarios through illustrative sequence diagrams. Two UCs are presented: one in a Smart Agriculture System and the other in a Health Monitoring System.

4.1. Use Case-1 (UC1): Unified Device Management

Summary: This scenario demonstrates how diverse Thing devices are unified under a single management platform by utilizing the Node, which aggregates access to all the Things' properties. Figure 2 presents a sequence diagram for this UC.

Actor: User

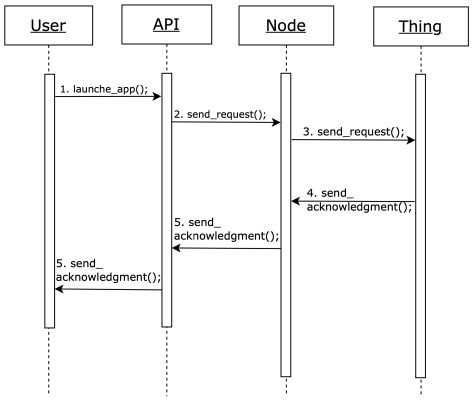
Precondition: The Node is pre-configured with all connected Things.

Description: Several IoT devices are installed in a user's premises and are ready to be utilized.

1. The user launches a unified application (e.g., a mobile application) designed to communicate with the Node, which serves as the central point for device management in the home.
2. The user sends a request to a Thing device to perform a task.
3. The Node forwards the formatted commands to the Thing devices.
4. The Thing devices acknowledge the completion of the commands and send confirmation to the Node.
5. The Node relays the acknowledgment to the user.

Postcondition: Requests sent to the diverse Thing devices are fulfilled using one unified control management platform.

Figure 2. Sequence diagram for the use case "unified device management"



4.2. Use Case-2 (UC2): Adjust Irrigation

Summary: This scenario illustrates how an IoT moisture sensor autonomously monitors soil moisture levels and activates an irrigation process through a smart ecosystem when necessary. Upon detecting low moisture, the sensor's data prompts either user intervention or an automated response, triggering an irrigation controller to hydrate the soil. In this case, the scenario highlights how a controller initiates an irrigation request based on predefined rules within the Node. Figure 3 provides the sequence diagram for this UC.

Actor: IoT sensor

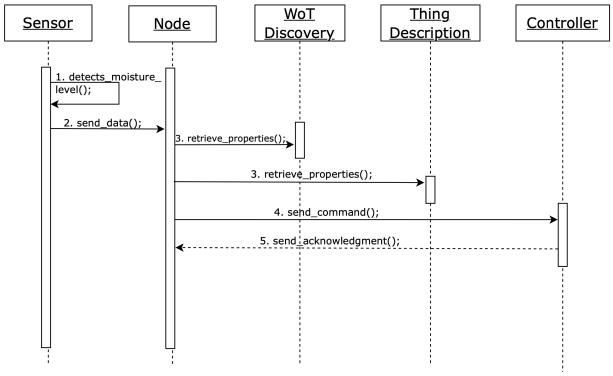
Precondition: The IoT sensor monitors soil moisture levels and detects that the soil moisture is below the predetermined threshold.

Description:

1. After measuring soil moisture, the sensor (a Thing) detects that the moisture level is below the predetermined threshold.
2. The sensor sends the measurement data to the Node.
3. The Node, utilizing WoT Discovery, identifies the sensor and retrieves its TD to interpret the data format and meaning.
4. The Node sends a command to the irrigation controller, activating the irrigation system.
5. The irrigation controller sends a confirmation or status update back to the Node.

Postcondition: The Node provides the user with reports about the soil moisture level.

Figure 3. Sequence diagram for use case "adjust irrigation"



4.3. Use Case-3 (UC3): Smart Emergency Response System

Summary: This UC illustrates a Smart Healthcare Monitoring System that leverages the WoT architecture to enhance patient care by continuously monitoring health metrics through connected medical devices in a healthcare facility.

Actors: IoT sensor, medical staff

Precondition: All IoT medical devices are operational and connected to the WoT gateway, with the system configured to monitor and respond to specific health metric thresholds.

Description:

1. IoT health sensors (e.g., heart rate monitors and blood pressure sensors) continuously collect patient data and transmit it to the WoT gateway.
2. The WoT gateway analyzes the incoming data against predefined health thresholds. If abnormalities are detected, it triggers alerts to medical staff and initiates necessary medical protocols.
3. Based on the analyzed data, automated adjustments to medical devices may be performed, such as modifying oxygen levels or administering medication through connected dispensers.
4. In critical conditions, the system automatically alerts emergency medical teams and provides detailed patient data to facilitate a rapid response.

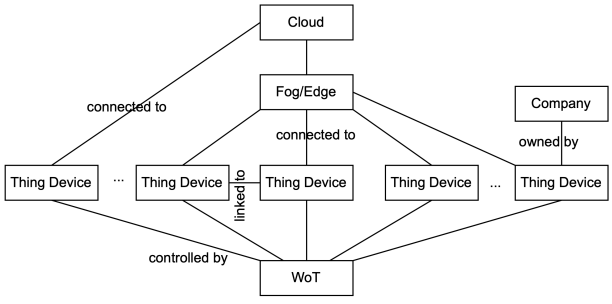
Postcondition: Appropriate medical responses are executed.

- **Implementation**

To understand the implementational aspects of WoT, we need to delve into its ecosystem and the possible scenarios where Thing devices are connected and implemented. Figure 4 shows a schema diagram of the WoT ecosystem. The ecosystem contains various heterogeneous IoT devices, some of which can be connected to each other, allowing for direct communication and collaboration.

The devices can be implemented through edge or fog computing to optimize data processing and decision-making closer to the source of data generation, while others are implemented directly in the cloud for broader data analysis and storage. Some devices are owned by companies, indicating their integration into business operations, while others are owned by individuals. All devices are connected and controlled by WoT (i.e., a web browser), ensuring standardized interactions and seamless interoperability across this diverse device environment.

Figure 4. Schema diagram of the WoT ecosystem



A notable example of WoT implementation is the Mozilla WebThing Gateway (Bolar, 2020), a smart IoT gateway that emphasizes interoperability and security. It plays a crucial role in connecting various IoT devices, allowing them to communicate and interact effectively in a standardized manner. The gateway serves as the central point of interaction for devices in the WoT network, resembling the Node class in our pattern.

- **Known Uses**

This section highlights real-world applications of the WoT, illustrating its practical utility and adaptability in addressing complex challenges across various domains. These examples provide insights into how the WoT pattern enhances functionality, interoperability, and efficiency in diverse IoT systems.

One example is Greenhouse Horticulture, which utilizes various sensors and facilities (such as heaters, CO2 generators, and sheet controllers) connected to a gateway and managed via the cloud. This system employs WoT architecture and WoT TDs to optimize environmental conditions, including temperature, humidity, and CO2 concentration, for plant growth. It demonstrates the application of WoT in agricultural technology (Matsukura and Kamiya, 2019).

Another example is in education, where an IoT remote lab enables students to interact remotely with various IoT devices as part of a practical course (Korkan, 2020). This setup includes robotic arms, conveyor belts, motorized sliders, and sensors for atmospheric data. The course allows students to build mashup applications, deepening their understanding of WoT technologies. These applications leverage the WoT TD and Scripting API, providing hands-on experience in controlling physical devices and verifying actions via video streams. This UC emphasizes the importance of standardized resource sharing in educational settings (Steinhorst and Korkan, 2020).

Lastly, the Eclipse Thingweb Project leverages W3C WoT standards to create IoT solutions that are both scalable and interoperable.

Key features of this implementation (Eclipse Foundation, ThingWeb) include:

1. Device Description: Utilizing standardized formats to describe device information, capabilities, and data schemas. This ensures that devices can be understood and managed consistently across different systems.
2. Device Integration: Providing connectivity for devices via various IoT protocols under a uniform interface. This approach enables diverse devices to communicate effectively, regardless of their underlying protocols.
3. Device Description Validation: Ensuring consistent metadata for devices across directories, which is crucial for maintaining the accuracy and reliability of device information.
4. Application Development: Offering a web browser-like runtime for developing portable IoT applications. This feature allows developers to create flexible, headless applications suited for various IoT scenarios.
5. Other Services: Including several libraries, tools, and services such as node-wot for building IoT devices, a playground for TD validation, and Online Things for simulating IoT devices. This project provides a comprehensive set of tools for developers to build interoperable IoT solutions while maintaining flexibility in their development choices.

- **Consequences**

This pattern provides the following advantages:

1. Compatibility: Utilizing TDs as a uniform interface allows devices with different models and versions to interact without direct compatibility issues.
2. Scalability: Employing the Node as a central point to manage connections and data flow between a vast number of Thing devices enables the system to scale effectively. By offloading device management and communication handling to Nodes, which are designed to efficiently process and route messages across the network, the architecture supports a growing ecosystem.
3. Interoperability: Leveraging Binding Templates alongside WoT TDs defines how devices communicate over various protocols. This setup ensures that devices can not only exchange data but also interpret and act on the information received, regardless of the underlying communication standards.
4. Discoverability: Implementing dynamic discovery mechanisms within the WoT Node allows it to automatically detect new devices and their services based on their TDs. This facilitates the seamless addition and integration of devices into the ecosystem, enhancing system responsiveness and user experience.
5. Security: Integrating security protocols directly into TDs and enforcing them through the Node ensures robust system protection. This includes specifying authentication mechanisms, data encryption methods, and access control policies within TDs, ensuring all interactions with and between devices adhere to predefined security standards.

- **Related Patterns**

1. Cloud Ecosystem Pattern (Syed and Fernandez, 2018): Illustrates the dynamic interaction between IoT devices and various components of ecosystem patterns within the cloud infrastructure.
2. Gateway Pattern for Integrated IoT Systems (Tekinerdogan and Köksal, 2018): Presents several gateway patterns integrated into IoT systems, including one focused on web services.
3. Design Patterns for the Industrial Internet of Things (IIoT) (Bloom *et al.,* 2018): Describes communication protocols for IIoT applications.
4. An Ontology Design Pattern for IoT Device Tagging System (Charpenay *et al.,* 2015).
5. A Pattern for Secure IoT Thing (Fernandez *et al.*, 2007): Presents a pattern for adding security to IoT devices.

5. Comparative Analysis

In this section, we present a detailed comparative analysis to examine the key components that play pivotal roles in the WoT architecture , specifically focusing on API, Node, Thing, TD, Thing Discovery, and Binding Template. This focused approach enables a direct comparison of how each architecture incorporates and supports these essential elements, highlighting areas where our architecture provides significant improvements or novel contributions.

As shown in Table 1, our architecture offers a more robust and flexible API, capable of supporting a diverse range of protocols and data formats, compared to the basic or RESTful-focused APIs in other architectures. The Node component, which is largely undefined in other works, is a key feature of our architecture. It enables sophisticated network operations and local data processing, which are crucial for real-time IoT applications.

Moreover, our architecture enhances the management and interoperability of Things by providing advanced features for autonomous operations and dynamic, detailed descriptions. This ensures seamless integration and operability within IoT ecosystems, representing a significant advancement over the simpler object management and static descriptions seen in the compared frameworks.

Additionally, the automated Thing Discovery mechanisms in our architecture support dynamic IoT environments, in contrast to the manual or socially enhanced methods in earlier models. Furthermore, the Binding Template in our architecture supports a variety of communication protocols, offering greater adaptability than the more limited implementations found in other studies.

Table 1. Comparison of the proposed architecture with the existing architecture

Architectural Component	Our Architecture	Guinard *et al.,* (2010)	Zhang *et al.,* (2012)	Guinard (2011)	Mainetti *et al.,* (2015)
API	API for device management and data interaction	RESTful API for basic device interaction	RESTful APIs integrated with social networking features	RESTful API focusing on device interaction	Limited API scope focused primarily on device discovery
Node	Node for managing network communication and processing	Not explicitly defined	Not explicitly defined	Not explicitly defined	Focus on virtualizing devices outside their physical network
Thing	Thing management with communication and control features	Basic smart Things integration	Enabled Things with limited control features	Basic Thing management	Basic Thing integration without extensive management features
Thing Description	Thing descriptions to facilitate interoperability and automation	Basic static description	Descriptions integrated with social profiles	Structured descriptions for application development	Not emphasized
Thing Discovery	Discovery mechanisms with support for dynamic environments	Discovery mechanisms based on RESTful services	Enhanced discovery through social interactions	Not emphasized	Advanced discovery mechanisms for virtualized environments
Binding Template	Templates supporting multiple protocols and data formats	Basic binding using web standards	Moderate binding capabilities with a focus on web integration	Not explicitly defined	Not emphasized

6. Conclusions and Future Works

This research presents an architectural pattern for the WoT, an area that has, until now, lacked precision in its modeling approach. The framework developed here not only aligns with the intricate requirements of the WoT environment but also enhances its interoperability and efficiency. The architecture proposed in this study is intended to serve as a foundational guide for future WoT architectures, providing a replicable and scalable approach for integrating the vast array of IoT devices into the web ecosystem. Additionally, the pattern lays the groundwork for creating a comprehensive RA for the IoT/WoT ecosystem.

Biography

Khalied M. Albarrak

Department of Management Information Systems College of Business Administration, King Faisal University, Al-Ahsa, Saudi Arabia, 00966135895800, kalbarrak@kfu.edu.sa

Dr. Albarrak, an Assistant Professor in the MIS department at King Faisal University, holds a B.Sc. in Computer Science (2009) from King Faisal University, an M.Sc. from King Abdullah University of Science and Technology (2012), and a Ph.D. from the University of Southampton (2019). His research areas include web sciences, open data, data analytics, artificial intelligence, e-governance, and digital transformation.

ORCID: 0000-0001-9224-5926

Acknowledgment

This study could not have been initiated or completed without the encouragement and continued support of King Faisal University.

References

Achirei, S.D., Zvoristeanu, O., Alexandrescu, A., Botezatu, N.A., Stan, A., Rotariu, C. and Caraiman, S. (2020). Smartcare: On the design of an IoT-based solution for assisted living. In: *International Conference on e-Health and Bioengineering (EHB),* **n/a**(n/a), 1–4. IEEE. DOI: 10.1109/EHB50910.2020.9280185

Alnaim, A.K. (2022). Misuse patterns from the threat of modification of non-control data in network function virtualization. *Future Internet*, **14**(7), 201. DOI: 10.3390/fi14070201

Alnaim, A.K., Alwakeel, A.M. and Fernandez, E.B. (2019). A Pattern for an NFV virtual machine environment. In: *13th Annual IEEE International Systems Conference*, **n/a**(n/a), 1–6. IEEE. DOI: 10.1109/syscon.2019.8836847

Alnaim, A.K., Alwakeel, A.M. and Fernandez, E.B. (2022). Towards a security reference architecture for NFV. *Sensors*, **22**(10), 3750. DOI: 10.3390/s22103750

Alwakeel, A.M., Alnaim, A.K. and Fernandez, E.B. (2019a). A Pattern for a virtual network function (VNF). In: *14th International Conference on Availability, Reliability and Security (ARES)*. 1–7. Canterbury, UK. DOI: 10.1145/3339252.3340519

Angelov, S., Grefen, P. and Greefhorst, D. (2012). A framework for analysis and design of software reference architectures. *Information and Software Technology*, **54**(4), 417–31. DOI: 10.1016/j.infsof.2011.11.009

Avgeriou, P. (2003). Describing, instantiating and evaluating a reference architecture: A case study. *Enterprise Architecture Journal*, **342**(n/a), 1–24.

Bloom, G., Alsulami, B., Nwafor, E. and Bertolotti, I.C. (2018). Design patterns for the industrial Internet of Things. In: *14th IEEE International Workshop on Factory Communication Systems (WFCS)*, **n/a**(n/a), 1–10. DOI: 10.1109/wfcs.2018.8402353

Bolar, T. (2020, August 18). *Web of things Over IOT and its Applications*. *InfoQ*. Available at: https://www.infoq.com/articles/web-of-things-iot-apps/ (accessed on 30/11/2024).

Buschmann, F., Meunier, R., Rohnert, H., Sornmerlad, P. and Stal, M. (2001). *Pattern-Oriented Software Architecture: A system of Patterns*. Volume 1. Wiley.

Charpenay, V., Kabisch, S., Anicic, D. and Kosch, H. (2015). An ontology design pattern for IoT device tagging systems. In: *5th International Conference on the Internet of Things (IOT)*, **n/a**(n/a), 138–45. DOI: 10.1109/iot.2015.7356558

Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E. and Bone, M. (2009). The concept of reference architectures. *Systems Engineering*, **13**(1), 14–27. *Portico*. DOI: 10.1002/sys.20129

Eclipse Foundation (n/a). *Eclipse ThingWeb*. Available at: https://thingweb.io/ (accessed on 30/11/2024).

Fernandez, E.B. and Hamid, B. (2015). A pattern for network functions virtualization. In: *20th European Conference on Pattern Languages of Programs*, **n/a**(n/a), 1–9. DOI: 10.1145/2855321.2855369

Fernandez, E., Pelaez, J. and Larrondo-Petrie, M. (2007). Attack patterns: A new forensic and design tool. *Advances in Digital Forensics III*, **n/a**(n/a), 345–57. DOI: 10.1007/978-0-387-73742-3_24

Fernandez, E.B. (2013). *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns*. John Wiley and Sons. DOI: 10.5555/2531565

Guinard, D. (2011). *A web of Things Application Architecture: Integrating the Real-World into the Web*. PhD Theiss, ETH Zurich. DOI: 10.3929/ethz-a-006713673

Guinard, D., Trifa, V. and Wilde, E. (2010). A resource-oriented architecture for the Web of Things. In: *Internet of Things (IOT)*, **n/a**(n/a), 1–8. DOI: 10.1109/iot.2010.5678452

Hashizume, K., Fernandez, E.B. and Larrondo-Petrie, M.M. (2012). A Pattern for Software-as-a-Service in Clouds. In: *ASE/IEEE International Conference on Biomedical Computing (BioMedCom)*, **n/a**(n/a),140–4. DOI: 10.1109/biomedcom.2012.29

Korkan, E. (2020). *Wot-Architecture Use Cases: Shared Devices*. Available at: https://github.com/w3c/wot-architecture/blob/main/USE-CASES/education.md (accessed on 30/11/2024).

Lagally, M., Matsukura, R., McCool, M., Toumura, K., Kajimoto, K., Kawaguchi, T. and Kovatsch, M. (2023). Web of things (WoT) architecture 1.1. *PR-wot-architecturell-202307111*.

Mainetti, L., Mighali, V. and Patrono, L. (2015). A software architecture enabling the web of things. *IEEE Internet of Things Journal*, **2**(6), 445–54. DOI: 10.1109/biomedcom.2012.29

Manta-Caro, C. and Fernández-Luna, J.M. (2024). IR.WoT: An architecture and vision for a unified web of things search engine. *Sensors*, **24**(11), 3302. DOI: 10.3390/s24113302

Matsukura, R. and Takuki, K. (2019). *Smart Agriculture: Greenhouse Horticulture*. Available at: https://github.com/w3c/wot-architecture/blob/main/USE-CASES/smart-agriculture.md (accessed on 30/11/2024).

Nedeltcheva, G.N. and Shoikova, E. (2017). Models for innovative IoT ecosystems. In: *International Conference on Big Data and Internet of Thing*. DOI: 10.1145/3175684.3175710

Pankowska, M. (2015). Stakeholder Oriented Enterprise Architecture Modelling. In: *12th International Conference on E-Business (ICETE15)*, **n/a**(n/a), 72–79. DOI: 10.5220/0005544700720079

Steinhorst, S. and Ege, K. (2020). *IoT Remote Lab*. Available at: (https://campus.tum.de/tumonline/ee/ui/ca2/app/desktop/#/slc.tm.cp/student/courses/950504601?$ctx=lang=en&$scrollTo=toc_overview&$scrollto=toc_overview (accessed on 30/11/2024).

Syed, M.H. and Fernandez, E.B. (2016). A pattern for a virtual machine environment. In: *The 23rd Conference on Pattern Languages of Programs*, **n/a**(n/a), 1–8. DOI: 10.5555/3158161.3158172

Syed, M.H. and Fernandez, E.B. (2018). A reference architecture for the container ecosystem. In: *The 13th International Conference on Availability, Reliability and Security*, **n/a**(n/a), 1–6. DOI: 10.1145/3230833.3232854

Syed, M.H., Fernandez, E.B. and Ilyas, M. (2016a). A Pattern for fog computing. In: *The 10th Travelling Conference on Pattern Languages of Programs*, **n/a**(n/a), 1–10. DOI: 10.1145/3022636.3022649

Tekinerdogan, B. and Köksal, Ö. (2018). Pattern based integration of internet of things systems. In: *Internet of Things–ICIOT 2018: Third International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, Springer International Publishing*, **3** (n/a). 19–33. DOI: 10.1007/978-3-319-94370-1_2

ThingWeb. (n/a). *Eclipse ThingWeb: Web of Things Components*. Available at: https://www.thingweb.io/ (accessed on 30/11/2024).

Zhang, C., Cheng, C. and Ji, Y. (2012). Architecture design for social web of things. In: *1st International Workshop on Context Discovery and Data Mining (ContextDD '12)*, **n/a**(n/a), 1–7. DOI: 10.1145/2346604.2346608