ARCHITECTURAL ASPECT-AWARE DESIGN FOR IOT APPLICATIONS: CONCEPTUAL PROPOSAL

Anas M. R. AlSobeh and Aws A. Magableh

Department of Computer Information System, Yarmouk University, Irbid, Jordan

ABSTRACT

Aspect-Oriented Programming (AOP) provides new constructs and concepts to handle secondary requirements in applications. Secondary requirements, i.e. crosscutting concerns, of the Internet of things (IoT) applications is inherited from the nature of the complexity of interactions, and implementation crosscutting concerns over core IoT architecture. Realizing the full potential of the IoT application requires a new abstraction design technique. This paper proposes an abstract class element toward a design approach to providing better means better separation of concerns. The proposed approach is accompanied by gathering relevant contextual properties pertaining to the environment of IoT interactions. A new architectural aspect-aware definition is proposed for tracking the logic of interaction characteristics on the IoT components being designed.

KEYWORDS

Aspect-Oriented Programming, Aspect Orientation, Modularization, Behavior-Interaction-Priority Model, BIP Components, Internet of things, IoT, crosscutting concerns, Aspects

1. INTRODUCTION

Internet of Things (IoT) is a modern technology evolved throughout the years. IoT has turned into the most crucial and prevalent system that empowers everybody to make, create, offer, utilize the data to produce information. IoT is the network of physical things, cars, water systems, home appliances and other items embedded with electronics, software, sensors with different types, and networks which enables these objects to connect and exchange data [3]. IoT targets at connecting of smart devices facilitating interactions among things and people. With IoT real word and digital words are interacting through various kinds of technologies such as internet protocols, sensors types and communication [4]. Having all these issues in mind, there will be so many crosscutting concerns that spread across different components of IoT architecture [15]. Managing and dealing with these crosscutting concerns in IoT environments is not an easy task, due to lack of management tools that ensure the performance, robustness, dependability, and security of IoT systems [2].

To address such issues, we need to introduce a dynamic management approach which provide better separation of concerns. Aspect Orientation (AO) is a suitable technique that introduce a modularization concept to encapsulate common crosscutting concerns into lossy coupled abstraction [12]. However, Aspect-Oriented Programming (AOP) is limited to a set of specific programming constructs such as constructors, methods, and proprieties. Consequently, less attention has been paid on developing AO to be used for model based design flow. Behavior-Interaction-Priority (BIP) model has been designed to provide a formal description language for essential real-time interactions (e.g. IoT systems) between system's components based on a set of priority rules for the transition of the behavior [7].

In this paper, we present and discuss the modelling of a combination model-based design of both BIP and AOP (named BIP-AOP) to overcome the challenges of modularizing IoT-related crosscutting concerns. The proposed model presents BIP-AOP as an IoT-aware interaction design through introducing a set of intercepted invoke/execution points. These points represent IoT-related context information, which are encapsulated into a high-level abstract aspect. Our abstract aspect can be extended to create an efficient runtime crosscutting module. Such a module is implemented into runtime advices that might be woven into an IoT-based system dynamically. BIP-AOP consists of three-layer architecture: IoT layer, AOP layer and Application layer. IoT layer represents systems interactions between components; AOP layer represents a mediator on the IoT infrastructure to glue the IoT-related components with needed application concerns; and Application layer allows developer to customize the IoT behaviors without prior implementation knowledge. All those layers are mapped into BIP components to address the design of an event-based interaction components as a solution for pulling all relevant context information from IoT components.

The paper is organized as follows: section 2 provides an overview of the AOP and BIP components and a set of important related works, section 3 illustrates the motivated case study that inspired us to propose our model. Section 4 discusses our proposed model, finally conclusion and future work have been presented in section 5.

2. Background And Literature Review

Crosscutting concerns in network systems using IoT face challenges stemmed in nature of the interactions between nodes and resources. This makes the dynamic control and management at run-time is a fundamentally complex problem. Managing interpretability and complexity are one of the key essential ways of controlling the run-time interactions between connected nodes at different IoT layers architecture [1]. This section explains a little bit of background and essential major related studies and works that focused on managing IoT application complexity and portability either using AO or non-AO methods.

2.1. Nutshell background

The interactions amongst the different abstraction layers of the IoT-based application architecture impact the overall system's complexity and portability. IoT-based applications are overlapping networks of heterogeneous objects. Thus, the Representational State Transfer (REST) design is an architectural style that enables application-layer interoperability and reuse. Additionally, AOP is used to decompose systems, but the nodes may be tautly coupled in a design IoT-centric component, which is more often complex. Here, we use the most IoT systems support for the design REST-based applications. Understanding the below concepts are vital to understanding the novelty of the idea.

2.1.1. Aspect-Oriented Programming (AOP)

Generally, the relationships among user requirements and program components intersect or crosscut in distributed and extended systems. In other words, a requirement may have to be implemented through several components; while on the other hand, a single component could cover more than one requirement or different parts of more than one requirement. Thus, a component may provide core functionality and at the same time include code for several other system requirements. An approach that tries to overcome this programming difficulty is called aspect-oriented software development (AOSD). These aspects encapsulate the functionalities that crosscut other functionalities in different parts of a system (Ex IoT systems). In AOSD, an executable AO program is created by automatically combining or ‘weaving’ together objects, methods and aspects to create a program that is not only easier to maintain, but also to reuse [5].

Aspect-oriented programming (AOP) is one of the most promising methods that developers can use to produce encapsulated objects that do not have any unnecessary additional functionality. This type of programming enables the developer to divide crosscutting concerns (i.e., an activity is also known as the separation of concerns (SoCs) into single logic, i.e., aspects. These aspects are the modular units of crosscutting concerns. In addition, as mentioned above, new behaviour can be added to a cloud application without the need to alter or interfere with the base source code. There are three key components in AOP: joinpoints, pointcuts, and advices [8] [9].

2.1.2. Internet of Things (IoT)

Internet of Things has been defined as a paradigm consisting of a variety of uniquely identifiable day to day things communicating with one another to form a large scale dynamic network. The exponential growth in semiconductor domain has resulted in an explosion of usage patterns of cost-effective sensor based processor system. These systems when get empowered with advanced communication technologies (e.g., Bluetooth Low Energy, LoRA, ZigBee, Insteon, 3G, 4G, 5G etc.) converges into an emerging form of technological domain-Internet of Things or in short IoT. IoT aims to offer, a massive scale, heterogeneous, interoperable, and context-aware, and simplified application development cum deployment capabilities to the enterprises and end-users. The Internet of Things (IoT) envisions a world in which everyday objects collaborate using the Internet in order to provide integrated services for users. This vision defines the IoT as a dynamic global network requiring global self-managing capabilities, based on standard and interoperable communication protocols.

2.1.3. BIP Component Framework

BIP stands for (Behaviour-Interaction-Priority) [6] is a formal framework for building complex systems by coordinating the behaviour of a set of atomic components. Behaviour is defined as a transition system extended with data and functions. The definition of coordination between components is layered: in the first layer lie the component interactions, while the second layer involves dynamic priorities between interactions [10].

2.2. Related Literature Review

Our literature review is focusing on investigating and reviewing the existing works those talks about utilizing AO in IoT systems, AO and BIP model in IoT and other approached were used to maintain portability of IoT applications. As stated in [4], there is the decent amount of works have been done to maintain service discover and service quality using different approaches, in [4] they have proposed AO to extend and enable IoT application to be more portable. They have stated that IoT derives various challenges from the Internet in the context of scalability, heterogeneity, undefined topology and data point information, incomplete metadata, and conflicts in user preferences. They have investigated the ability to develop an AO intermediate layer to inject the needed context related functionality. Their proposition works as a layered interface between IoT applications hardware and data gathered at software.

Few more works such as [1] have proposed a model-based design flow for networked systems with nodes running an Internet of Things (IoT) operating system. The design flow specifically targets web service applications of REST style and it is based on a formal modelling language, the BIP component model. Figure 1 explains their approach to managing IoT applications by using the BIP model. However, in [7] defined a method to modularize crosscutting concerns in the BIP component-based design. The authors have defied the BIP using AOP in a formal method and have given a mathematical representation for the AO and BIP concepts. However, this work has not proposed a solution for possibly modularization the IoT system concepts and related common crosscutting concerns.

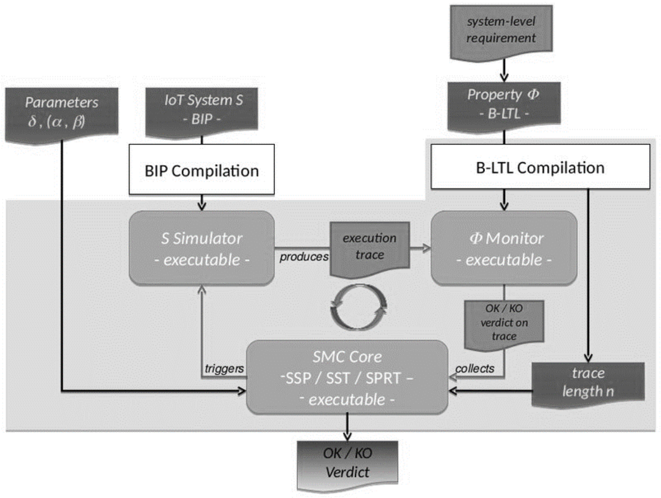

Figure 1. Model-based design flow for IoT [1]

Aspect-oriented programming is being used to manage systems (such as component-based systems) where IoT is considered as one of them. Some of the decent amount of works have been proposed at this space [11] [12].

3. CASE STUDY

IoT environments have a significant potential to provide for monitoring of water services to promote the possibility of tracing water flow. Such environments are typically equipped with many heterogeneous sensors that monitor both water and environmental parameters. The best example of it is water level display in the tank. IoT water system is used by water-level sensors to determine the level or amount of water that flow in an open or closed water system. Sensors usually detect the specific battery energy levels, if the sensing state is low then the brightness of dashboard light is reduced. It is integrated into the single device to get an alarm or trigger. These sensors measure water levels within a specified range and continuously make a notification of the water the level.

Figure 2 illustrates basic IoT water system components that represent the dashboard that consist of data input and sound components. The energy management component is responsible for keeping track of power level for turning off or sleeping the dashboard and responsible for controlling on the level of dashboard brightness (LED) by observing the percentage of power volume.

Energy management is a common crosscutting concern and often poorly modularized in traditional design approach. Battery-level applications such as remote controls, dashboard, sound and sensor network can be preserved: (a-Low energy state) by reducing sound volume by 33% turn off dashboard after 10 seconds of no interaction; (b-medium energy state) by reducing sound volume by 25%, and reducing display brightness to 50% after 5 seconds of no interaction; and (c-high energy state) by reducing sound volume and brightness 0%. Implementation energy management component as an aspect-oriented extension that supports decomposition and integration of it with core IoT water system including that management as crosscutting. From secondary requirement perspective, focus on the energy management concerns that will manifest as interaction in the IoT system, can be encapsulated in loosely coupled aspect module.

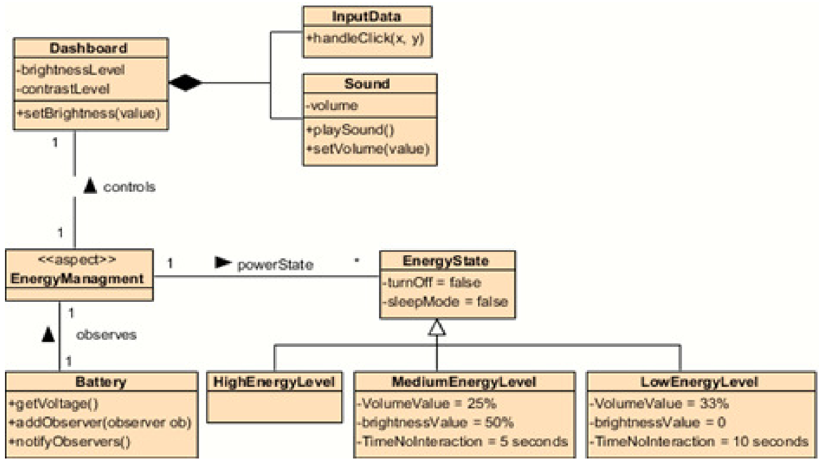

Figure 2. Basic IoT Components for Smart Water

Traditionally, *EnergyManagment* implementation is scattered across multiple classes (Dashboard and Sound) while calculation of energy state is tangled across multiple components. Indeed, the main difficulty is not the code complexity only, but it is mainly associated with interactions IoT nodes and component state they effect, which might refer to these interactions as "spaghetti bowl" [13] [14], as shown in Figure 3.

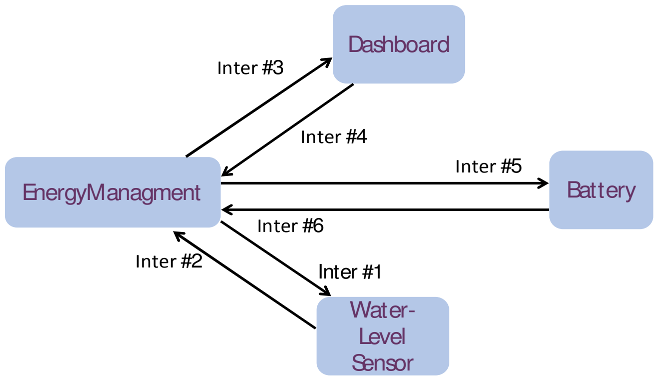

Figure 3.IoT Interaction Model

BIP is a model-driven engineering approach can help in reducing the complexity of IoT systems designs through its support a clear separation between architecture and interaction to allow for compositional design and analysis of systems [7]. However, BIP does not address the design of crosscutting concerns that do not manifest as code in the system but is complementary to existing techniques for capturing such requirements. In general, the code of such management interaction cuts across the system components, so their modularization significantly reduces their complexity, where the state of practice is to use BIP constraints for some behavior characteristics, BIP-AOP is an extension on top of such techniques to demonstrate its use to modularization and reuse. See next sections for more details.

4. Proposed BIP-IoT System Design Using AOP

During a comprehensive analysis of an IoT application, it emerged that concerns related to IoT architecture and behaviour were most significant. Inspiring BIP with the AOP technique was conceived to design and support the modularization during the development of IoT applications

[7]. Our design has a formal semantics and makes a clear separation between IoT components to allow for decomposition IoT design into maintainable and reuse modules. Figure 4 shows context-aware aspects that is utilized to capture the behaviour and interaction concerns of IoT components. It presents components, interactions, priorities, and their composition. An atomic-context aspect is the basic low-level computation modules, which encapsulate IoT-related context information. It is implemented as an aspect and their behaviours defined as a glue-aspect that is extended with high-level abstractions. Transitions are represented as arrows associated with context-related IoT components to transfer data and messages between components.

Figure 4 demonstrates artifacts of the proposed aspect layer- model design, they are possible to generate the BIP-AOP components to be reused in the IoT application. The IoT components are instantiated from the IoT design definition including the IoT's application mapping onto AOP system components. Atomic-context components obtained by pulling a set of interactions among low-level components occur when execution of one component modifies the behaviour of another one. Such interactions may cause by modification or extension aspects to any state that is accessible by high-level glue-aspect components, including the state of the core IoT application, communications, protocols, and network.

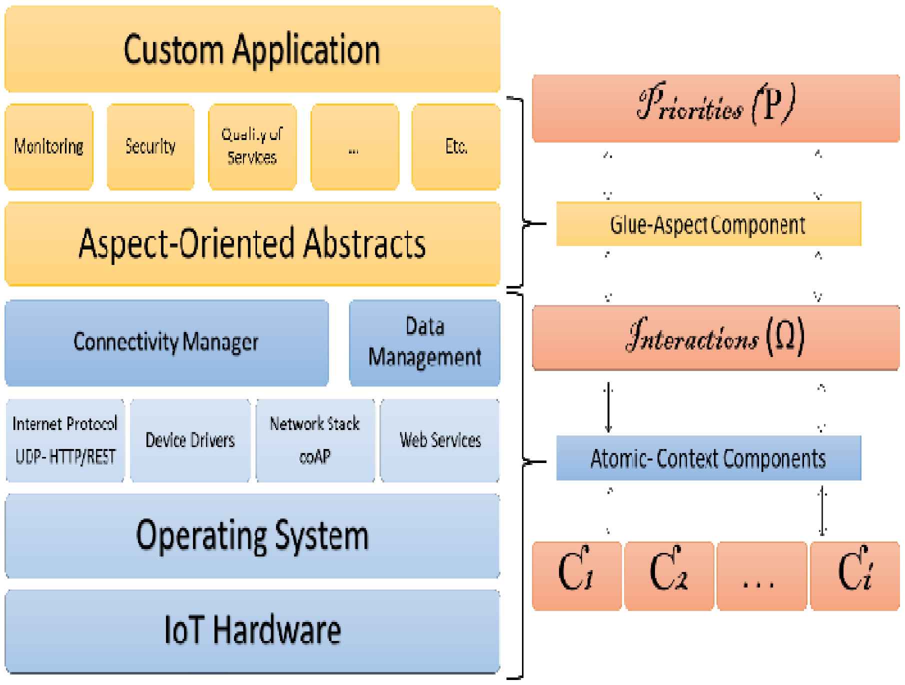

Figure 4. Standard IoT Design-based Model with AOP

4.1. The BIP model of the IoT Application

The overall our approach involves converting the application design definition for the IoT application and its protocol, e.g., REST, into BIP components as shown in Figure 5. Description of structure allows constructs application analysis findings back to the IoT design definition. The BIP components implemented by the BIP-AOP architecture are instantiated from the IoT framework through determining the components' context parameterization and their characteristics. These are encapsulated in the atomic-context component which is formalized as an observer for tracing the state space of secondary requirements, i.e., crosscutting concerns, with the BIP components as mentioned. Validation of properties derived from functional and nonfunctional IoT requirements takes place with by state-space exploration with BIP components according to the specified mapping onto system's component.

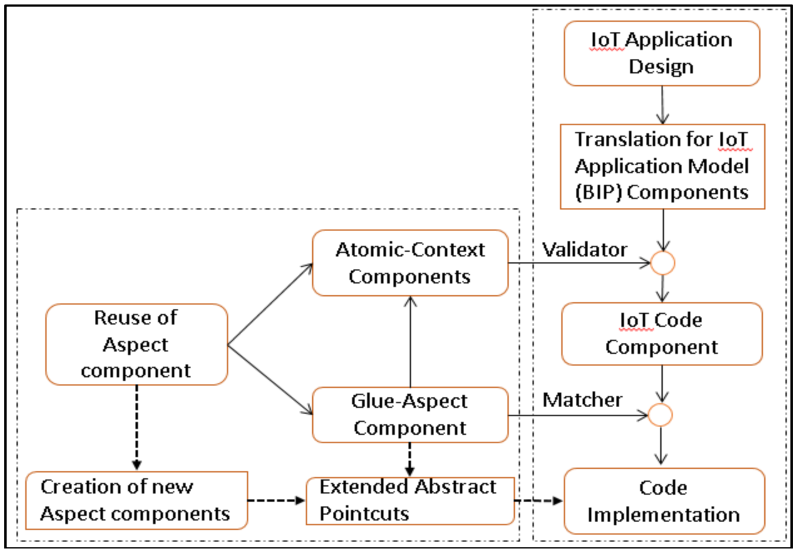
IoT Application Design
Translation for IoT Application Model (BIP) Components
Atomic-Context Components
Validator
Reuse of Aspect component
IoT Code Component
Glue-Aspect Component
Matcher
Creation of new Aspect components
Extended Abstract Pointcuts
Code Implementation

Figure 5. Architectural BIP-AOP for IoT Activities

The runtime properties IoT requirements take place by invoking the generated code on the components and within IoT environment. All IoT constraints are pulled and weaved to/into the core code modules. In AOP, for instance, abstract aspect used to impose constraints on the matching of pointcuts and application of advice. Glue-Aspect components can be framed as the abstractions directly representing the architecture of IoT event-based interactions as detailed below.

4.2. Initial Context-Aware Aspects

Context primitives allow the IoT application developer to identify scattered and tangled over multiple codes using existing standards, technologies, and protocols for encapsulating IoT crosscutting concerns and leverage wherever possible [16]. BIP encourages the development of a smaller set of defacto standards for IoT concerns (e.g., implement policy and practice to ensure the monitoring concerns of IoT interactions of distributed components), an interaction is a software implementation based on behaviour function(s) that transforms groups of data into context data, and priority is a selection process, we select more sequence of execution flow of advices out of many other advices.

In context-aware IoT applications, Definitions 1&2 identify the semantic properties for support of the different type of information that allows applications to adapt their behaviour in response to interaction in the IoT environment using AOP. These properties are encapsulated in the aspect-oriented abstract layer.

Definition 1 ($Advice$): An advice A encapsulates a set of context data C_a is defined by a pointcut$P$ and a set of joinpoint$the\ J_p\ such\ that \forall p \in \boldsymbol{P}: J_p \subseteq\ \boldsymbol{C_a}$.

Definition 2 ($Atomic\ Context\ Aspect\ Ap$): An abstract aspectis a tuple$\langle C_a | P | J_p \rangle$, where Ap is a transition state consists of relevant context proprieties, which is possibly receiving the new valuation of the J_p holds the application of computation $Advice$ with the set possible properties in $C_a\ of$.

Develop scalable approaches for separation of resource-constrained IoT nodes. We adapt some fundamental IoT-related context concepts into BIP components. IoT nodes interactions require that applications deal with the inherent unreliability of communications and processes. We have identified six primary context concerns that require support by developers. They are distributed, node discovery, limited connectivity, location, proximity, and quality of service. This context information can be exploited to address the semantic heterogeneities of data exchanged by nodes interactions, e.g., Atomic-context data of sensors, s1 $\rightarrow$ d1 means that sensor 1 has produced a piece of data that is numbered 1. Likewise, s2 $\rightarrow$ d2 means that sensor 2 has produced a piece of data that is numbered 2. Sensors will likely be heterogeneous, from different manufacturers, and collect data, with varying levels of data integrity. Usually, sensors are geographically located. Sensors may have an owner(s) who will have a control over the collected data, who can access it, and when. Implementing such input leads to scattered across multiple modules which cause spaghetti bowl as discussed in section 3.

Interaction Ωserves as the glue-aspect component that encapsulates crosscutting concerns through their atomic-context components. An interaction involves one or more abstract aspects of different atomic-context properties and extends advice that realizes data management between the IoT application components.

Definition 3 (*Interaction*): An interaction $\Omega$is enabled iff its C_a holds and all its pointcuts are invoked. An invoked interaction is called from the complete list of possible interactions that based on the states of the atomic-context components.

The BIP approach defines the invoked interactions and executes its $Advice$, which defined in the application context with an atomic-context aspect that is extended by implementing the logic that interferes with the execution of an IoT-based component used a special kind of inner class. Hook atomic-context components invoke their corresponding pointcuts given the new value received by the designated joinpoints. In the following, we consider an atomic-context component $C_a$with behavior is filtered the invoked higher-level abstractions dynamically and decrease non-determinism.

Definition 4 (*Priority*): Priorityρ is a set of advices to be applied once a pointcut$p$ has been matched to be found simply by specifying the precedence ordering to the abstract aspects, which contain the $advice$ in the glue-aspect components.

Apart from these changes, the weaving semantics for regular aspects does not have to be modified for IoT-based aspects. Priority ρ over C_a is used to define the event that should be performed preceding or succeeding a function execution. BIP-AOP model components are elaborated on AOP processes. It extends by defining glue-aspects for capturing multiple concerns and specifying a precedence order on the set of interactions Ω, which is defined the set of transitions satisfying Definition 4.

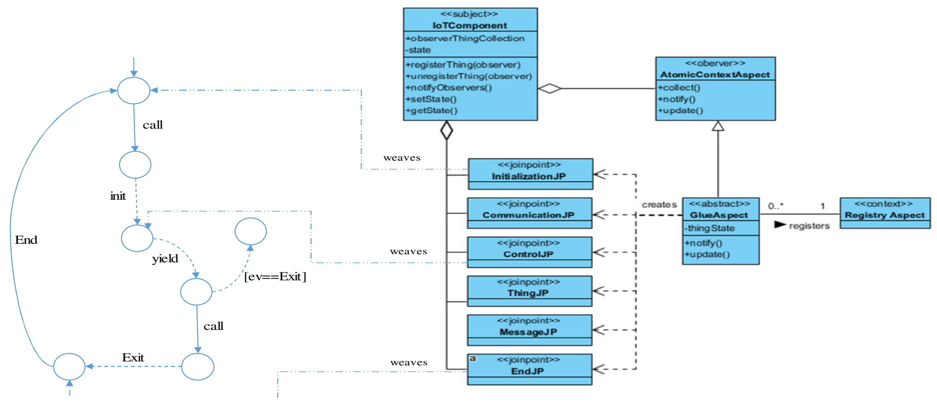

Figure 6. BIP-AOP Architectural IoT Pattern Design

4.3. BIP-AOP Model-Based Architecture

Figure 6 illustrates the architectural BIP-AOP design model in IoT domain. It traces state of every distributed IoT components based on using observing state approach. In case, the state of interaction changes, IoT components will be notified at different levels in the system. We define many BIP components to provide the understanding of using different IoT concepts (thing, protocol, node, communication, etc.) in an IoT context as discussedearlier. *AtomicContextAaspect* is an aspect component acts as an observer into a spectrum of the lower-level model to be a controller of the IoT components interactions. In other words, it represents the aspect that implements the advice function whose task is to handle all the crosscutting concern logic (e.g., monitoring, security, management, etc.). It relies on AOP inter-type declarations to introduce context properties and methods to IoT components and APIs. We extend this aspect with the proposed high-level customization aspect called *GlueAspect*, to weave custom logic or amend the normal workflow of interaction in the system. The idea behind defining such aspect is to decouple the implementation by offering pointcuts, advices, inter-type declarations were defined in *AtomicContextAspect* to exclude components dependency. *GlueAspect* involves a set of joinpoints such as initialization, call and execution for the *InitializationJP, MessageJP, CommunicationJP, EndJP, ControlJP, ThingJP* and so on. *GlueAspect* offers pointcuts to pick up those joinpoints and then use advice to inject the logic of the crosscutting concerns. Once running the IoT application, the state of the one thing object has been changed by the BIP event handler (such as call, initiation, exit, etc.) all other components objects depending on it will be notified and take appropriate action as shown in Figure 6. The *RegistryAspect* then takes responsibility to register and monitor all IoT components interactions states.

5. Conclusions

This paper has proposed an integrated model of using AOP and BIP to provide a better modularization of cross-cutting concerns in of IoT applications. The proposed model demonstrates crosscutting issues in IoT application and highlights how AOP addresses them. Specifically, the purpose of this paper is to provide a roadmap for using BIP model components with AOP to deal effectively with crosscuts in IoT application design, systems interaction, and integration. In the future, we will work on conducting a primary experiment to showcasing the efficiency and effectiveness of using our model in encapsulating, modularizing and separating crosscutting concerns obliviously.

REFERENCES

[1] Lekidis, A., Stachtiari, E., Katsaros, P., Bozga, M., &Georgiadis, C. K. (2018). Model□based design of IoT systems with the BIP component framework. Software: Practice and Experience, 48(6), 1167-1194.

[2] Tselentis, G., &Galis, A. (Eds.). (2010). *Towards the future Internet: emerging trends from European research*. IOS press.

[3] Borgia, E (2014), "The Internet of Things vision: Key features, applications and open issues", *Computer Communications,*Vol.54, pp.1-31.

[4] Balakrishnan, S. M., &Sangaiah, A. K. (2015). Aspect oriented middleware for Internet of things: a state-of-the art survey of service discovery approaches. *Int. J. Intell. Eng. Syst*, *8*(4), 16-28.

[5] Shanmughaneethi, V., Praveen, R. Y., &Swamynathan, S. (2012). CIVD: detection of command injection vulnerabilities in web services through aspect–oriented programming. International Journal of Computer Applications in Technology, 44(4), 312-320.

[6] Basu, A., Bensalem, B., Bozga, M., Combaz, J., Jaber, M., Nguyen, T. H., &Sifakis, J. (2011). Rigorous component-based system design using the BIP framework. *IEEE software*, *28*(3), 41-48.

[7] El-Hokayem, A., Falcone, Y., &Jaber, M. (2016, July). Modularizing crosscutting concerns in component-based systems. In *International Conference on Software Engineering and Formal Methods* (pp. 367-385). Springer, Cham.

[8] Djoko, S. D., Douence, R., &Fradet, P. (2012). Aspects preserving properties. *Science of Computer Programming*, *77*(3), 393-422.

[9] Katz, S. (2006). Aspect categories and classes of temporal properties. In *Transactions on aspect-oriented software development I* (pp. 106-134). Springer, Berlin, Heidelberg.

[10] Verimag: BIP Tools, http://www-verimag.imag.fr/BIP-Tools,93.html

[11] Nazarpour, H., Falcone, Y., Jaber, M., Bensalem, S., &Bozga, M. (2017). Monitoring Distributed Component-Based Systems. *arXiv preprint arXiv:1705.05242*.

[12] Ma, K., Sun, R., & Abraham, A. (2013). Toward a Module-centralized and Aspect-oriented Monitoring Framework in Clouds. *J. UCS*, *19*(15), 2241-2265.

[13] Abbes, M., Khomh, F., Gueheneuc, Y. G., &Antoniol, G. (2011, March). An empirical study of the impact of two antipatterns, blob and spaghetti code, on program comprehension. In *Software maintenance and reengineering (CSMR), 2011 15th European conference on* (pp. 181-190). IEEE.

[14] Mikkonen, T., &Taivalsaari, A. (2007). Web Applications: Spaghetti code for the 21st century.

[15] Bilal, M. (2017). A Review of Internet of Things Architecture, Technologies and Analysis Smartphone-based Attacks Against 3D printers. *arXiv preprint arXiv:1708.04560*.

[16] Perera, C., Zaslavsky, A., Christen, P., Compton, M., &Georgakopoulos, D. (2013, June). Context-aware sensor search, selection and ranking model for internet of things middleware. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on* (Vol. 1, pp. 314-322). IEEE.

[17] AlSobeh, A., & Clyde, S. Unified Conceptual Model for Joinpoints in Distributed Transactions. In *ICSE* (Vol. 14, pp. 8-15).

Authors

Dr.AnasAlSobeh, Dr.AlSobeh received his B.Sc. and M.Sc. degrees in computer information systems from Yarmouk University in 2007 and 2010, respectively. He also received PhD degree in computer science from Utah State University/USA with honour in Dec. 2015. He joined Yarmouk University academic staff in 2016. He is currently an assistant professor of computer information systems (CIS). His research interests include web technology, e-learning systems, software engineering, distributed systems, cloud systems, Internet of things (IoT) and data modelling. He has many scientific publications. He also has European-funded projects. He is also an active member of credit mobility projects to exchange academic members.

Aws A. Magableh is an Assistant Professor at the Faculty of Information Technology and Computer Science in Yarmouk University, Jordan. He obtained his PhD from the National University of Malaysia (UKM) in 2015, he obtained his master in Software Engineering from University Malaysia (UM) in 2008, and Bachelor's in Software Engineering from the Hashemite University in 2006. His research interests include Web technology, Web Services, Cloud Computing, Aspect-Orientation, Software Engineering, System design and modelling. Aws is very passionate about Learning & Development (L&D) and I have been immersed in the training industries with Nokia, Microsoft and Huawei for the past 10 years focusing on building skillsets