acquired to the server. The data is acquired when an RFID card is pointed over the antenna. In practice, the data is acquired when a user enters the room. Before this data is sent to the server, the program appends a numeric value that identifies the reader and subsequently the room. If there is no data from the reader (i.e. nobody enters the room) in the last five minutes, the program automatically generates a notification message and send it to the server. As shown in figure 6, this message is used by the server to request feedback of the users' keyboard and mouse activities from KAM and MAM respectively.

The server's program is also developed to continuously listen from and respond to data received from proxy computers. As illustrated in figure 6, the server initializes required software modules and interprets the data received to determine the venue. If the input is a notification message, the server checks if there is anyone in the venue. If there is someone, the server checks the status of their keyboard and mouse activities and determines their ongoing context. If the input is data from the reader, the server loads the corresponding information. This information includes name of the user who entered the venue, her office, role, social relations and other devices the user may own and their IP addresses. The server also checks the status of the keyboard and mouse activities of the users who are in the room. The server uses this information to determine the ongoing context of the users.

As shown figure 6, the server requests for feedback of the users' keyboard and mouse activities from KAM and MAM respectively when (i) a user enters in the room or (ii) after every five minutes from the time the first user entered the room. To address a similar problem, Sensay [16] used a ten-minute duration while Harter *et al*. [17] used a five-second duration. I used a five-minute duration because most of the Operating Systems consider a computer as idle when is inactive for five minutes. This feature is important to prevent the server from making decisions based on temporary changes within the environment.

The server, however, does more than the steps outlined in figure 6. When a card is frequently pointed on the reader, for instance, the server takes into account the first input and ignores the rest. If the existing user in the venue points her card after five or more minutes, the server infers that the user is leaving the room. Before deleting the user from the list of users who are in the room, however, the server monitors the status of her devices for next two minutes. If there is no interaction within that period, the server then deletes the user. Otherwise the server treats the detection as unreliable and ignores it. The best solution, however, would be to use two readers; one positioned near the door and the other a bit further. So when a user enters the room, her card will be detected first by the first reader and vice-versa.

Algorithm 1 Server Algorithm

```
1   Listen from the proxy computer
2   while (input from proxy computer is not null) do
3     Initialize the reasoning engine
4     Load the knowledge base
5     Read the system time
6     Open user activity logger
7     Open inference activity logger
8     Interpret data from the proxy computer
9       if (the input is from an RFID reader)
10        Get information about the user
11          if (room is empty)
12            go to 1
13          Else
14            while (the room is NOT empty) do
15              Check the status of user's computer
16                  if (computer is ON)
17                    Check user's keyboard activities
18                    Check user's mouse activities
19                  end if
20              Update the user's status
21              Log the user's status
22            end while
23            Insert facts about the room into the engine
24            Match the rules from the knowledge base
25            Fire the matched rule
26            Trigger application manager
27            Log the engine's decision
28          end if
29      elseif (the input is a notification message)
30            go to 11
31      end if
32      if (the user is not in the room)
33          Add the user in the list of the existing users
34          go to 15
35      end if
```

Figure 6. Pseudo code for the server

4. Application of KoDA

To evaluate a context-aware architecture, framework or middleware, researchers have been using imaginary or working context-aware applications. Kaenampornpan [18], Kofod-Petersen [19] and Henricksen [6], for instance, used scenarios to illustrate how their solutions can be used in a real-world. Biegel [5], Chen [20] and Dey [7] implemented and used working context-aware applications to demonstrate how their solutions can be used in a real-world. Using scenarios is a feasible approach since the focus of these solutions is not on implementing context-aware applications. Nonetheless, scenarios are far from reality to fully show the potentials of these solutions. Hence, this work adopts both approaches to evaluate KoDA.

4.1. Remotely Switching Devices ON/OFF

To illustrate how KoDA works, an application to remotely switch ON or OFF IP-based devices depending on a context was developed. As shown in figure 2, the application, through the application manager, are included in the right-hand side of inference rules and therefore can only be invoked if ongoing context of a user is determined. Upon recognizing a user in a venue, the server takes MAC and IP addresses of the user's devices. For illustration purposes, once the prototype recognizes an ongoing context, it triggers the application to remotely switch ON or OFF the devices.

Assume that this is an application to remotely change the alert mode of the users' mobile phones. If the recognized context is a meeting, for instance, KoDA would trigger this application to seamlessly change the settings of the user's phone from the ringing mode to the silence or vibrating mode. The users would not have to worry about where they enter and what settings their phones have. KoDA would make the use of mobile phone intuitive and thus contributing to the vision of UbiComp.

4.2. KoDA with Microsoft Cortana

Microsoft Cortana[4] is an intelligent personal assistant application for Microsoft Smartphones. It combines voice recognition and context-awareness to effortlessly assist a user. One of the boasting feature of Cortana is its ability to automatically transfer phone calls to voicemail when you do not want to be disrupted. This feature is useful, for instance, when you are briefing your boss about a product or giving a keynote speech in a conference. With Cortana installed in your Smartphone, you simply need turn it ON when you do not want to be disrupted and OFF when you are in your normal routines.

Nonetheless, Cortana cannot recognize ongoing context and hence depends on a user to turn the feature ON or OFF. Subsequently, this requires a user to continuously be aware of her social settings and settings of her Smartphone to effectively use this feature. So before meeting your boss for briefing, for instance, you need to remember about the meeting and turn the feature ON. Once you finish the meeting, you need to remember to turn the feature OFF. Using Cortana with KoDA removes the need of the users to continuously remember about their social settings and the settings of their devices. Hence there is a potential for increasing the users' productivity when KoDA is used.

5. Conclusion

This paper presents a Knowledge-driven Distributed Architecture (KoDA), describes implementation of a prototype and illustrates how the architecture can be applied and used in a real-world environment. Unlike the existing architectures, KoDA is designed based on a knowledge-intensive model hence enabling it to make its decisions based on a comprehensive model of the real-world environment. Like any other context-aware architectures, KoDA is a 3-layered architecture comprised with perception, inference and application layers. Through the perception layer, KoDA gathers facts of a venue and share with the inference layer. The inference layer uses the facts to determine ongoing context and communicate with application layer that triggers appropriate applications.

To illustrate capabilities of KoDA, a prototype has been developed. To illustrate application and usage of KoDA in a real-world, the prototype was set and used in a real-world environment. KoDA was able to recognize ongoing contexts of users in the environment and trigger an application to perform a task. On another occasion, a scenario of using Microsoft Cortana (i.e. an intelligent personal assistant) with KoDA has been used. The scenario shows that KoDA can work well with Cortana because the users will not have to think about their devices, social settings and whether settings of their devices conform or not. All these will be taken care by KoDA since it can dynamically recognize ongoing contexts and communicate with applications to respond accordingly.

Through its distributed nature, KoDA can be used to support multiple rooms in a workplace. KoDA can also support resource-constrained computing devices. Through its ability to continuously monitor, infer and respond to changes in an environment, KoDA can dynamically recognize ongoing contexts. Furthermore, KoDA is designed with a middleware that separates connectivity processes from interpretation processes. Complemented by a generic and knowledge rich context model, which allows a subset of context parameters to be implemented, KoDA is more scalable and flexible to new technologies as they emerge.

Experiments are crucial in a scientific research in Computer Science and are core to the evaluation of any UbiComp system. Therefore the next step of this work is to perform experiments to measure accuracy of KoDA on recognizing contexts. KoDA is capable of storing information about recognized contexts and evidences used to infer these contexts. This information can be used as another source of knowledge for KoDA. The majority of inference engines are incapable of learning and therefore currently such information is not useful. Therefore, a research on learning mechanisms for context-aware architectures is required to such information useful.

[4] https://www.microsoft.com/en-us/cortana/

6. References

[1] Evans, D. (2011). The internet of things how the next evolution of the internet is changing everything. Tech. rep., Cisco.

[2] Van Kasteren, T., Noulas, A., Englebienne, G. & Krose, B. (2008). Accurate activity recognition in a home setting. In Proceedings of the 10th international conference on Ubiquitous computing, 19, ACM.

[3] Liu, H. (2010). Biosignal controlled recommendation in entertainment systems. Technische Universiteit Eindhoven, Eindhoven, 1133.

[4] Kukkonen, J., Lagerspetz, E., Nurmi, P. & Andersson, M. (2009). Betelgeuse: A platform for gathering and processing situational data. Pervasive Computing, IEEE, 8, 4956.

[5] Biegel, G. (2005). A Programming Model for Mobile, Context-Aware Applications. Ph.D. thesis, University of Dublin, Trinity College.

[6] Henricksen, K. (2003). A Framework for Context-Aware Pervasive Computing Applications. Ph.D. thesis, School of Information Technology and Electrical Engineering, The University of Queensland.

[7] Dey, A.K. (2000). Providing Architectural Support for Building ContextAware Applications. Ph.D. thesis, Georgia Institute of Technology.

[8] Da, K., Roose, P., Dalmau, M., Nevado, J. & Karchoud, R. (2014). Kali2much: a context middleware for autonomic adaptation-driven platform. In Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT, 2530, ACM.

[9] Roalter, L., Kranz, M. & Moller, A. (2010). A middleware for intelligent environments and the internet of things. In Ubiquitous Intelligence and Computing, 267281, Springer.

[10] Ranganathan, A. & Campbell, R.H. (2003). A middleware for context aware agents in ubiquitous computing environments. In Middleware 2003, 143161, Springer.

[11] B. Schilit, N. Adams & R. Want (1994). Context-aware computing applications. In Mobile Computing Systems and Applications, 1994. WMCSA 1994. , 85-90.

[12] D. Lupiana (2015), A Knowledge-driven Distributed Architecture for Context-Aware Systems, Ph.D. thesis, School of Computing, Dublin Institute of Technology.

[13] Coutaz, J., Crowley, J.L., Dobson, S. & Garlan, D. (2005). Context is key. Commun. ACM, 48, 4953.

[14] Schmidt, A., Aidoo, K., Takaluoma, A., Tuomela, U., Van Laerhoven, K. & Van de Velde, W. (1999). Advanced interaction in context. In Handheld and ubiquitous computing, 89101, Springer.

[15] Forgy, C. (1979). On the Efficient Implementation of Production Systems. Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University.

[16] Siewiorek, D., Smailagic, A., Furukawa, J., Krause, A., Moraveji, N., Reiger, K., & Wong, F. L. (2003). Sensay: A context-aware mobile phone. In null (p. 248). IEEE.

[17] Harter, A., Hopper, A., Steggles, P., Ward, A. & Webster, P. (2002). The anatomy of a context-aware application. Wireless Networks, 8, 187197.

[18] M. Kaenampornpan (2009). A Context Model, Design Tool and Architecture for Context-Aware Systems Design. Ph.D. thesis, Department of Computer Science, University of Bath.

[19] A.Kofod-Petersen (2007). A Case-Based Approach to Realising Ambient Intelligence among Agents. Ph.D. thesis, Department of Computer and Information Science, Norwegian University of Science and Technology.

[20] H. Chen (2004). An Intelligent Broker Architecture for Pervasive Context Aware Systems. Ph.D. thesis, University of Maryland.

7. Acknowledgements

Many thanks to the Government of Tanzania, through the Institute of Finance Management, for funding this research. My sincere gratitude to Fredrick Mtenzi, Brendan O'Shea and Ciaran O'Driscoll for their invaluable contributions throughout this research.

