Journal of Theoretical and Applied Information Technology
31st December 2016. Vol.94. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A SOFTWARE-HARDWARE OPTIMIZER MODEL FOR OPTIMIZED DESIGN OF THINGS IN AGENTS OF THINGS

[1]ANAS M MZAHM, [2]MOHD SHARIFUDDIN AHMAD,
[3]ALICIA Y. C. TANG, [4]AZHANA AHMAD
[1]College of Graduate Studies, Universiti Tenaga Nasional, Malaysia, 2016
[2,3,4]College of Computer Science and Information Technology,
Universiti Tenaga Nasional, Malaysia, 2016
E-mail: [1]anas1982mm@outlook.com, {[2]sharif, [3]aliciat, [4]azhana}@uniten.edu.my

ABSTRACT

The machines, or 'things' in the Internet of Things (IoT) lack self-reasoning capability, which limits their potential to provide value-added services for humans. Consequently, we introduce the concept of Agents of Things (AoT) as an extension to the IoT, in which the things are embedded with self-reasoning intelligent software agents to provide value-added services for humans. Two crucial issues in designing intelligent things are to determine what value-added services they should offer and the subsequent level of reasoning abilities that are required for these services. Consequently, we need to find an optimum match between the hardware capabilities of the things and their corresponding software agents' reasoning abilities to deliver value-added services on top of performing their basic IoT functions.

In this paper, we present the results of a software analysis represented by a software spectrum and a hardware analysis represented by a hardware spectrum. We then link these two spectra to form a structured hardware-software optimizer for a thing's design model, which we called the Structured Hardware-Software Optimizer or SHOM. We demonstrate the use of SHOM in designing optimized things in a simulated traffic scenario in manifesting the AoT concept.
Keywords: *Internet of Things; Agents of Things; Hardware Analysis; Software Analysis; Structured Hardware-Software Optimizer; Software Hardware Optimizer Model; Value-added Services; Optimum Things;*

1. INTRODUCTION

The Internet has progressed over the years with the use of new technologies such as the Web 2.0 [1-3] and the Semantic Web [4, 5]. Recently, the Internet is further expanded with a new concept of interconnected 'things' known as the Internet of Things (IoT) [6], in which devices or things are connected to the Internet to provide connectivity and communication between the cyber world and the real world. Glass [7] and Jermyn et al. [8] believe that the ability to establish a machine-to-machine (M2M) interaction allows devices to be connected and communicated with each other without a user's intervention. This characteristic opens the door to many research in building useful systems that could help humans in many aspects in their daily lives. However, Tan and Wang [9] argue that the IoT is constrained by the lack of intelligence and self-reasoning on their environments. Therefore, to augment the IoT, we propose the Agents of Things (AoT) concept and discussed a complete introduction of it in [10].

The benefits of developing applications utilizing the concept of Agents of Things would support and enhance the vision of the Internet of Things. Creating things to be intelligent entities, enable researchers to develop intelligent applications that have great impact on the society, for example, an application similar to that we proposed in a traffic system in [11].

The society will experience the impact of such applications, when the benefits translate to saving human lives, warning other drivers about accidents and minimizing the actions and time taken by the authority to respond to an accident on the road. Moreover, the benefits of continuously monitoring driver's speeds and giving them a fair warning before fining for over speeding will create an atmosphere of fair, justice and equality among the drivers and motorists on the road. On the other hand, this could indirectly manifest the efficiency of the authority and government, when they will use less resources, such as police forces, ambulances, rescue services, et cetera and timely actions to solve traffic system emergencies.

The AoT concept could be expected to advance the research field to a new unprecedented level, by enhancing machine-to-machine interactions. However, to make the AoT concept a reality, we need a model that determines an optimum reasoning level in the things, i.e. we need to find an optimal match of software and hardware for each thing. In our previous work, we conducted analyses on the available types of hardware [12] and software [13]. We use the results from these two analyses to form a structured software-hardware optimizer [14], which plays an important role in constructing the Software-Hardware Optimizer Model (SHOM).

This paper presents the work-in-progress of our research in Agents of Things. We organize the paper as follows: Section 2 reviews related work in this area. Section 3 discusses the results of the hardware analysis, software analyses and the structured hardware–software optimizer (SH-SO). Section 4 introduces the Software-Hardware Optimizer Model (SHOM). Section 5 demonstrates the application and testing of SHOM in an AoT traffic scenario. Section 6 discusses the SHOM and Section 7 concludes the paper.

2. RELATED WORK

Researchers who work on intelligent systems that can improve and solve many issues in daily life are courting on the huge potential of the IoT concept [15-19]. They inspire us to take a different approach to the IoT concept by analyzing and identifying the inherent constraints in the IoT. We attempt to alleviate these constraints by proposing an extension to the IoT concept, which we call the Agents of Things (AoT) [10, 11].

A review of the literature on similar work does not reveal significant results on the related work. While there are a few studies made on the optimization of hardware and software in computing, there is a lack of universal studies on the hardware and software to find an optimum combination between them. However, there are many studies that are focused on software optimization to suit a certain type of hardware, such as finding the optimum software for embedded systems in studies by Tabbara et al. and Tang et al. [20, 21].

There are also studies that attempt to optimize a hardware in running a certain type of software, such as Kolasa and Dlugosz [22] who investigate the optimum architecture to run self-organizing neural networks. Likewise, Lysecky [23] studies a configurable architecture to run various types of software.

We focus our effort to review some research on IoT models that constitute some factors that are related to our application scenarios, specifically in hardware and software integration, and especially, those that present or introduce a model for a specific application.

The IoT applications on business issues are important to discuss. Some researchers, like Yu et al. [24], use the IoT to improve a business model by reducing information asymmetry and improving the communication channel between consumers and producers of green agriculture products. Others, like Jia et al. [25], create a new IoT business model based on the IoT infrastructure itself. These models are telecom model, Internet model and vertical industry model, each of which has its own characteristics that fit different types of applications.

Researchers have also investigated on the IoT infrastructure and shown how it improves or incorporates new applications. For example, Zhang and Meng [26] attempt to improve the IoT itself by proposing a multi-dimensional ontology model to manage resources and representation of IoT devices' attributes, which improve the ability of inquiry and perceptiveness. Another researcher, Zuerner [27], proposes a model based on the Greenfield approach, which categorizes IoT hardware devices into a number of flags that consist of many options to represent different categories. Using this approach, he attempts to achieve a number of goals namely, to catalogue hardware devices with a certain order or classification, and to display some important information with labelled flag such as, applications, storage capacity or energy requirements. He also attempts to improve the transparency for social acceptance by specifying the applications and promoting self-autonomy via managing and controlling the labelled flags storage capacities.

Research has also been conducted on power consumption of IoT hardware devices, such as sensors and RFIDs. Abedin et al. [28] propose a model for green IoT systems, which improve the energy efficiency of IoT devices and extending the life span of the IoT network. They use an algorithm to evaluate the need for certain devices and reschedule their operations for more appropriate time to save power, which promotes green IoT systems. Correspondingly, Kang et al. [29] propose a monitoring system for future smart homes. They propose a model built from three levels of context, which is collecting or data acquisition, processing and generating information and context making. This model claims to improve the context awareness of smart home monitoring.

We review another IoT research about a warning system for environment crisis management proposed by Poslad et al. [30]. They introduce an IoT EWS system model built from combining a lightweight and heavyweight semantics with W3C web ontology to rescale a valuable data for exchange and process. This is done to adaptably manage and control information and communication resources in a crisis zone.

3. SIGNIFICANCE OF HARDWARE AND SOFTWARE ANALYSES

We incorporate the results from the hardware and software analyses [12, 13] as an important aspect of the AoT design to determine the association of hardware and software in designing optimized things that are used in the AoT concept. Consequently, these things are endowed with optimum reasoning level for use in building AoT systems.

3.1. Hardware Analysis Spectrum

The hardware analysis that we conducted previously [12] classifies a wide range of hardware devices and their specifications. The results of the analysis are summarized as a hardware spectrum, as shown in Figure 1. This spectrum includes the devices' specifications such as size, computing capability and cost. We structure the hardware spectrum to focus on important specification and make the spectrum expressive and convenient to form a structured hardware-software optimizer.

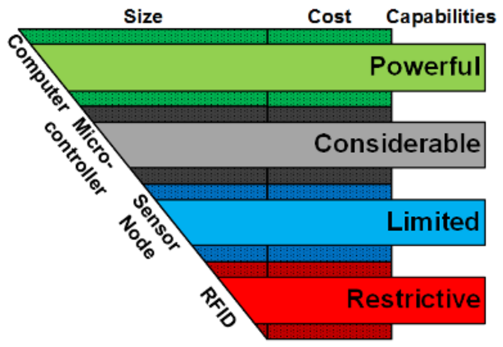

Figure 1: Hardware Spectrum

From Figure 1, at the bottom of the spectrum, we find the RFID devices, which is specified by their small sizes, very low costs and restrictive computing capabilities. An RFID device can only run a simple program that is specifically designed for it. Above the RFIDs are the sensor devices. These devices are distinguished by their small sizes, low costs and limited computing abilities. However, it is powerful enough to execute simple software with interchangeable capability, i.e. it can run different types of software sequentially but not concurrently.

Next, one step above to the middle of the spectrum, are the microcontrollers. These devices are notable by their moderate sizes, reasonable costs and considerable computing capabilities. They manifest the characteristics of the lower and higher levels' devices. The moderate size with powerful performance ability to run sophisticated software make them contenders to the computers. However, due to their unique architectures, they are able to run several simple programs or one sophisticated program at the same time. Finally, at the top of the spectrum, we find the computers. These devices are well-known for their large size, high cost and very powerful computing capabilities. These devices are capable of concurrently executing several simple and sophisticated programs.

3.2. Software Analysis Spectrum

The software analysis we conducted previously [13] re-classifies a wide range of software based on their types and programming abilities. The results of this analysis are summarized in the software spectrum as shown in Figure 2. This spectrum represents the capabilities, such as the program sophistication level and the corresponding type of software that this level supports, such as application software, system software, and et cetera.

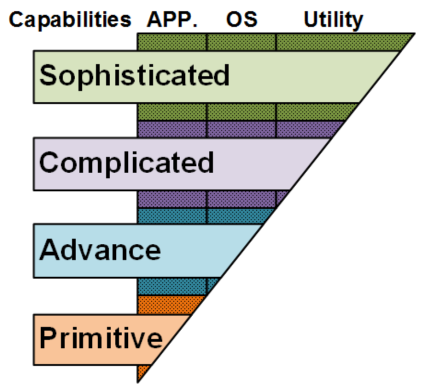

Figure 2: Software Spectrum

This spectrum holds the information about the software. It is structured vertically to show the software main types, such as application software, operating systems and utility software. However, it is more focused on the software types and abilities, e.g. an application software that forms a complicated and sophisticated program and uses to solve a user's needs. This ability represents an important factor to find the corresponding match or optimum device in the hardware spectrum.

Starting from the bottom of the inverted triangle, a software at this level is represented by a simple or primitive logical operations, such as Boolean logic. It is also formed by application software without an operating system or other supporting software. An example of a software at this level is a thermostat in an air-conditioner or an electric water heater. Such a software can also be found in low level sensors, such as motion, light or radiation sensors [31]. Moving up to the next level, a software at this level is more advanced from the previous level. It is represented by advanced logical operations, such as multiple selection, loops and functions. Moreover, it is formed as an application software running under an operating system software. Good examples are the software in home appliances, such as washing machine and television set [32]. Another good example is the software of sensor nodes in sensor networks. The sensor's software represents the operating system that provides the basic operations to control the communication protocols and the internal operations of the sensor [33, 34]. However, the application software is usually a standalone software provided by a user/designer to meet his/her requirements. This software ranges from simple functions to reactive software agents to alter the behavior of the sensor [35-37].

In the next level, the software is more advanced and powerful represented by a complex algorithm. It can be a single algorithm that includes a combination of functions, objects or abstraction of logic. In this level, a software contains all the software classification types, such as application, operating system and utility software. An example is any software coded to run on single chip devices, such as Arduino microcontroller or Roseberry Pi microcomputer [38-40]. Both of these devices come with internal operating system and supported software to identify and connect external devices, such as a printer and a monitor. The software application is usually a user-defined app, such as image recognition and editor, robot controller and programming code editor [41, 42].

Finally, at the top of the inverted triangle, the software is very powerful, represented by multiple sophisticated algorithms. It can include a countless combination of functions, objects and advanced logical abstraction or artificial intelligence. The software at this level constitute all the software classification types, such as application, operating system and utility software. Examples of software at this level are computer and mobile applications, such as graphic editors, movie and video editors, video players and text editors [43].

3.3. Structured Hardware-Software Optimizer

The combination of these two spectra forms a structured hardware-software optimizer for the AoT thing's design model. This structured optimizer plays an important role in finding the optimum combination of the hardware and software to design an AoT thing. Figure 3 shows the structured hardware-software optimizer.

The main working principle of the structured optimizer is centered on the requirement to find the hardware and software capabilities. Basically, any thing's design must start from a basic point, which is predefined by a user depending on the hardware and software required by the thing. The hardware software integration produces the function the thing must perform in the AoT system. We design the structured optimizer to determine the software capabilities as output given the hardware as an input or vice versa.

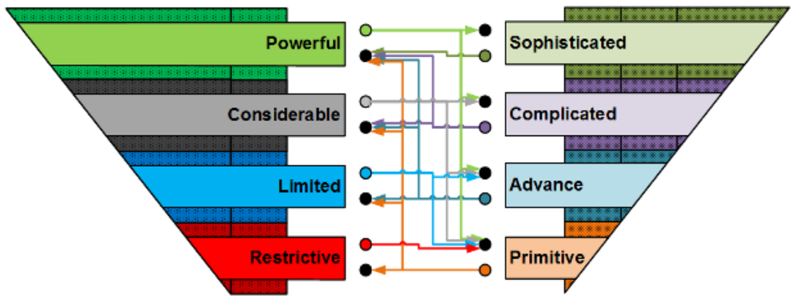

Figure 3: The Structured Optimizer for the AoT Model

To clarify this working principle, consider the structured optimizer of Figure 3. It is formed by associating two parts represented by the hardware and software spectra. Each spectrum is divided into four levels, each represents a category of hardware and software capability. The hardware part is divided into four hardware capabilities. First, powerful category that include hardware devices such as computers, smart mobile phones and tablets. Second, considerable category that include hardware devices such as the cellular mobile phones and microcontrollers. Third, limited category that include hardware devices such as mobile sensor nodes. Finally, we have restrictive category, which includes hardware devices such as the basic sensors and RFIDs.

Correspondingly, the software part is similarly divided into four capabilities. Starting from the sophisticated capability category which represents sophisticated software, such as multiple BDI software agents, real-time systems and dynamic artificial intelligent programs. Second, the complicated category that represented by complicated algorithms, such as reactive software agents and multiple object-oriented entities. Third, the advanced category which is represented by software with complicated logic, such as simple object-oriented entity or multiple functions programs. Finally, the primitive category, represented by limited software capabilities, such as a simple logic or selection program.

Assume that we are designing a system for which we know the hardware device (thing) to use but we want to find the software capabilities that run on the hardware. Therefore, if the device is from the first category, then it can execute all the software programs in the software spectrum. If the device is from the second category, it can run the complicated software category and those below it. If the device is from the third category, then it can run the software programs from the advanced and primitive software categories. Finally, if the device belongs to the fourth category in the hardware spectrum, then it can run the software programs from the primitive category only. It is important to mention, that when a thing's hardware belongs to the powerful, considerable or limited category, there is a few rules that need to be applied to find the optimum software without wasting hardware or software resources. These rules are represented by these questions:

- ❖ Does the thing require multiple advance artificial intelligence or object entities?
- ❖ Does the thing require simple artificial intelligence abilities or object entities?
- ❖ Does the thing require using multiple sub-programs, such as functions and procedures?

Now, let us assume the opposite scenario, in which we know the essential software functionality but we need to find the optimum hardware capability to run the software. From Figure 3, if the program belongs to the first category in the software spectrum, then it can be run on the first hardware category only (restrictive). If the software belongs to the second category, then it can be run on the powerful and considerable categories in the hardware spectrum. However, if the program belongs to the advanced category, then it can be run by the limited hardware category and the categories above it. Finally, if the program belongs to the fourth category, then it can be run by all the hardware categories in the hardware spectrum. When we have a program that can run on multiple hardware categories, a few rules need to be applied to find the optimum hardware without wasting any hardware or software resources. These rules are represented by the following questions:

- ❖ Does the software need to be executed multiple times with different forms at the same time?
- ❖ Does the software need to be executed multiple times with different forms one at a time?

❖ Does the software need to be executed multiple times with the same form one at a time?

We can explain and clarify the working principle of these rules in any given scenarios. For an example, let us assume that we want to design a system for a road accident monitor (RAM), which we described in [11]. The system monitors the road for traffic accidents. When an accident occurs, the system alerts other vehicles heading towards the accident area. It then contacts the emergency services to facilitate the accident situation.

This system consists of two main devices (things), which is the RAM that monitors the vehicles and the device in each car. Knowing the software capability of each device as "Complicated" (i.e., specifically software agents as used in the scenario), we trace the corresponding hardware capability by using the structured optimizer flowchart, as shown in Figure 4.

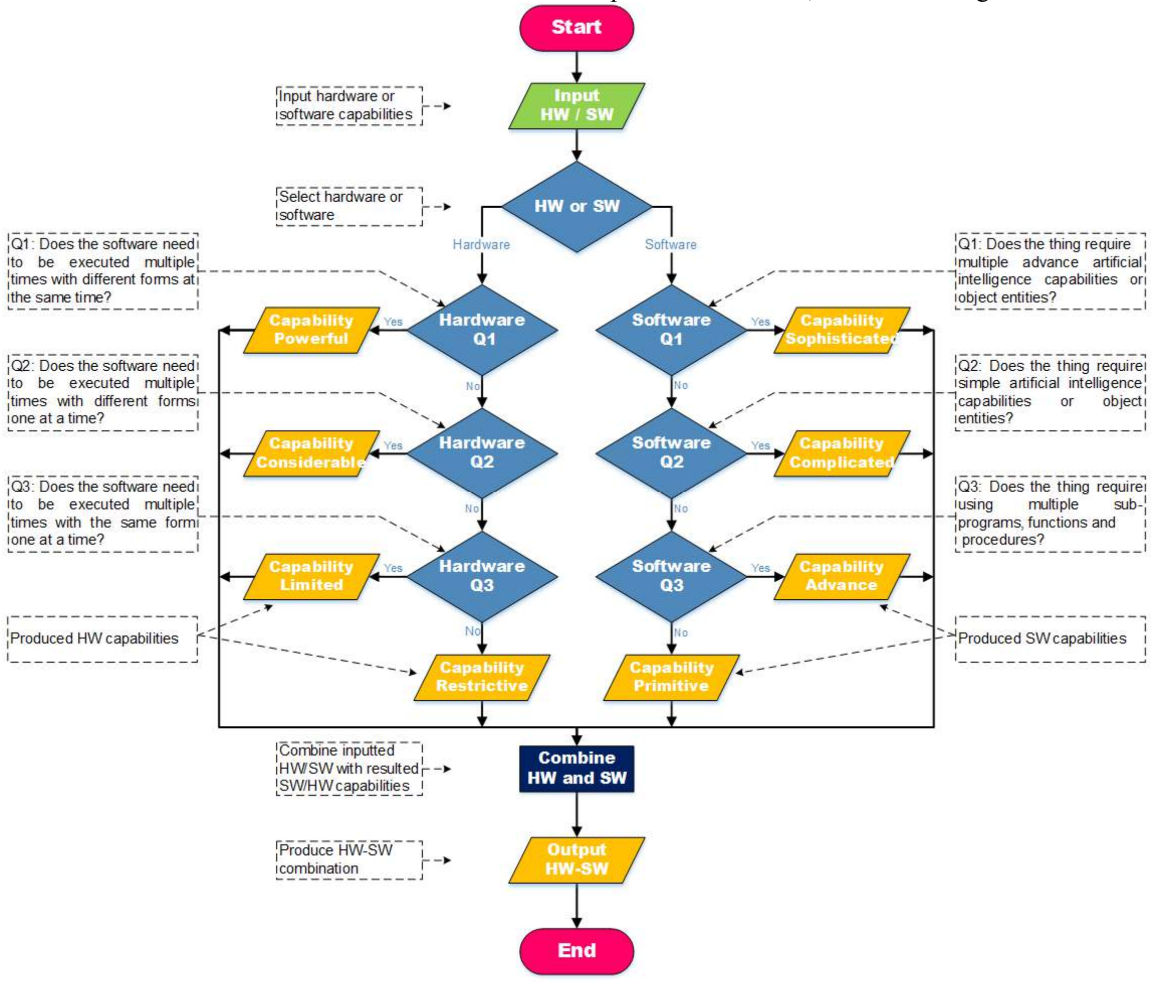
Start
Input hardware or software capabilities
Input HW / SW
Select hardware or software
HW or SW
Hardware
Software
Q1: Does the software need to be executed multiple times with different forms at the same time?
Q2: Does the software need to be executed multiple times with different forms one at a time?
Q3: Does the software need to be executed multiple times with the same form one at a time?
Q1: Does the thing require multiple advance artificial intelligence capabilities or object entities?
Q2: Does the thing require simple artificial intelligence capabilities or object entities?
Q3: Does the thing require using multiple sub-programs, functions and procedures?
Hardware Q1
Hardware Q2
Hardware Q3
Software Q1
Software Q2
Software Q3
Yes
No
Capability Powerful
Capability Considerable
Capability Limited
Capability Restrictive
Capability Sophisticated
Capability Complicated
Capability Advance
Capability Primitive
Produced HW capabilities
Produced SW capabilities
Combine inputted HW/SW with resulted SW/HW capabilities
Combine HW and SW
Produce HW-SW combination
Output HW-SW
End

Figure 4: The Structured Hardware-Software Optimizer Flowchart

Tracing the flowchart, the first question is "Looking for Hardware or Software capabilities?" Moving to the "Hardware" route, the trace meets the second question "Does the software need to be executed multiple times with different forms at the same time?" A "No" route moves the trace to the next question, "Does the software need to be executed multiple times with different forms one at time?" Since a software agent is characterized by such implementation, the resulting route should be "Yes", hence, determining the optimum hardware capability for our complicated software as "Considerable". We then look up the corresponding hardware specification based on this capability.

The combination of the hardware and software spectra offers a framework in designing an optimum thing for the AoT system. The SH-SO determines a perfect match between a thing's hardware and its software capabilities. A near perfect hardware-software match provides the configuration in determining an optimum design for things that commensurate with a correspondingly optimum reasoning level to be used in building AoT systems.

4. SOFTWARE-HARDWARE OPTIMIZER MODEL

The Software-Hardware Optimizer Model (SHOM) is the final step to complete a thing's design for the AoT concept. It is used to create optimized things that form the components of AoT systems. The main purpose of this model is to find the right hardware specification for a given software or vice versa. Figure 5 shows the general illustration and internal components of SHOM.

The SHOM accepts inputs such as hardware and software specifications and reasoning capabilities from the outside world, perform internal processes and procedures on these inputs to produce an output represented by "optimized things". The SHOM internal components are formed from two main parts, which is the controller and the structured hardware-software optimizer.

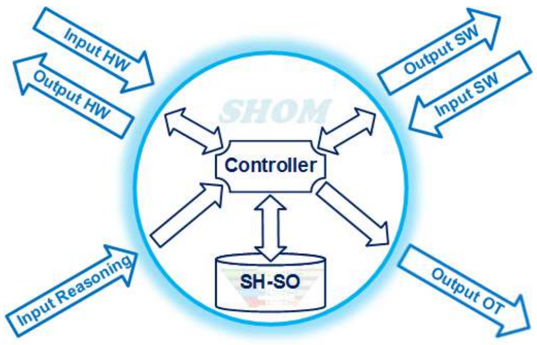

Figure 5: SHOM Illustration and Internal Components

The controller is responsible for all the internal operation inside the model, such as receiving/ delivering the inputs/outputs, evaluating and consulting the structured hardware-software optimizer. On the other hand, the structured hardware-software optimizer (SH-SO) is responsible for finding the optimum match for the inputs to produce outputs.

The internal operation of SHOM starts with the controller receiving the inputs of hardware or software and reasoning level. Then, it consults the SH-SO for the optimum match for the given inputs (i.e. the SH-SO finds the perfect software match for a hardware input or finds the perfect hardware match for a software input). At this point, the SH-SO returns the result represented by a hardware-software combination to the controller, which it evaluates and compares with the reasoning input. The controller then checks and evaluates if the hardware-software combination produced by the SH-SO is powerful enough to handle the reasoning capability received from the Value-added Service Analyzer (VaSA). If it is adequate to execute the suggested reasoning capability, the controller is considered to have produced an "optimized thing" as a consequence of the fusion of the reasoning capability with the hardware-software combination. However, in case the hardware-software combination is not adequate to execute the suggested reasoning capability, then, the controller returns the hardware-software combination result to the SH-SO to improve the hardware or software aspect of the hardware-software combination. Only the hardware aspect is improved because it is the only material thing that can be altered or changed. Finally, when the improved result is returned from SH-SO the controller, it proceeds with the fusion process and produce the optimized thing. To clarify the internal process of SHOM, we use the following flowchart in Figure 5.

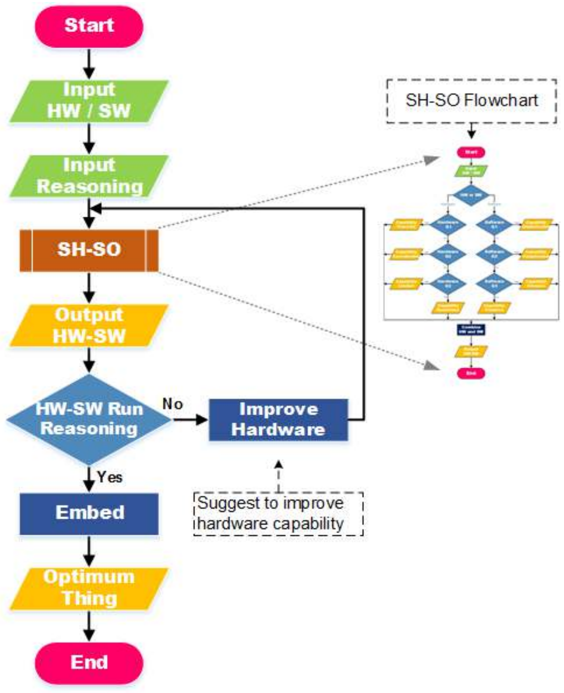

Figure 6: SHOM Operation Flowchart

From Figures 5 and 6, we can see that this model is flexible enough to produce optimally specified things for AoT and IoT systems. The model produces designs of things for IoT systems without the reasoning capability input, i.e. at the step for checking the reasoning capability, "if it can be run by HW-SW combination?" respond with a Yes and proceed normally.

5. APPLYING SHOM ON AOT ROAD ACCIDENT MONITORING (RAM) SCENARIO

The SHOM model is a very flexible tool that is able to produce optimized things for AoT systems. It finds the optimum match for any hardware or software and add to it the reasoning capability produced by the Value-added Service Analyzer. These optimized things are used to build AoT systems that solve real life issues. An example of these issues is the previously designed AoT road accident monitoring scenario [11]. In this scenario we apply the AoT concept on a traffic system to monitor the road for any traffic accidents.

In this system, called the Road Accident Monitor (RAM), when an accident happens, the system warns other vehicles heading towards the accident area. It then contacts the emergency services to facilitate the accident situation. Figure 7 shows the main topography of this system.

Before we apply the SHOM model on the RAM system, it is important to notice the topography of the RAM system. Due to the nature of this system, it is not designed to cover a vast area, but to cover a limited area, where the accident rate is high, like road junctions. On the other hand, when an accident occurs, the number of vehicles involved in the accident are usually small. Therefore, there is no need to use a high range device to perform the RAM delegated operations.

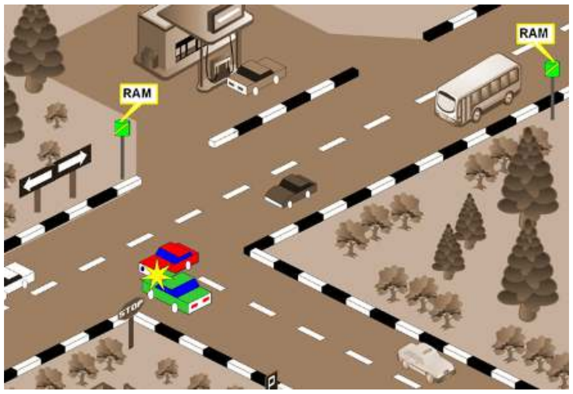

Figure 7: RAM System Topography

Now, let us apply the SHOM model on this scenario when we know the hardware capability and we need to find the optimum software capability to form the HW-SW combination for the RAM system. SHOM consults the SH-SO for a perfect match. The process starts as follow:

Q 1 What is the user input: Hardware/Software?
Ans.: Hardware.

Q 2 What is the hardware capability?
Ans.: Considerable.

Q 3 Does the thing require multiple advance artificial intelligence or object entities?
Ans.: Skipped.

Q 4 Does the thing require simple artificial intelligence abilities or object entities?
Ans.: Yes.

Q 5 Does the thing require using multiple sub-programs, such as functions and procedures?
Ans.: Skipped.

At this level, we have the optimum software capability result which is "Complicated". The HW-SW combination is formed from "Considerable" hardware and "Complicated" software. Now, SHOM checks if this HW-SW combination can run or execute the reasoning capability. If we assume that the reasoning capability is "Medium".

Q 6 Does the reasoning capability equal or less than software capability?
Ans.: Yes, Equal.

The result of applying SHOM on RAM system produced an optimum thing formed from Considerable hardware, Complicated software and Medium reasoning. This optimum thing looks like a micro-controller programmed with hybrid software agents, such as rational BDI agent. This is the final result when we try to find the software capabilities for RAM system. However, can we get the same result if we apply SHOM to find the hardware capabilities for the RAM? The process starts as follow:

Q 1 What is the user input: Hardware/Software?
Ans.: Software.

Q 2 What is the software ability?
Ans.: Complicated.

Q 3 Does the software need to be executed multiple times with different forms at the same time?
Ans.: No.

Q 4 Does the software need to be executed multiple times with different forms one at a time?
Ans.: Yes.

Q 5 Does the software need to be executed multiple times with the same form one at a time?
Ans.: Skipped.

The result shows that the optimum hardware capability is "Considerable", i.e. the HW-SW combination is formed from "Considerable" hardware and "Complicated" software. Now, SHOM check if this HW-SW combination can run or execute the reasoning capability. If we assume that the reasoning capability is "Low".

Q 6 Does the reasoning capability equal or less than software capability?
Ans.: Yes, Equal.

The result of applying SHOM on the RAM system produces an optimized thing formed from Considerable hardware, Complicated software and Medium reasoning. Therefore, the result after we reverse the inputs from hardware to software shows "Considerable" capability, which is the same hardware capability that we used when we initially try to find the software ability.

By applying SHOM on the traffic scenario, we demonstrate that SHOM works well when we try to find the optimum software or hardware for the corresponding hardware or software input. Moreover, it works in this scenario when we use simple compact device for limited number of communications. Consequently, it can be used to find optimum things for IoT system or other real world systems.

6. DISCUSSION

The AoT is a revolutionary concept in some aspects. It is an inherently intelligent system due to the reasoning action performed by software agents embedded in the things. The generality of the concept is gained by its ability to be customized and reshaped to suit any related issues and generate proper solutions. The central core of this ability is due to SHOM.

The SHOM is a very effective tool capable of finding an optimized hardware and software combination for any system design fused with a reasoning capability to solve any issues as demonstrated by the RAM things in a traffic system scenario. The SHOM model determines the optimum level of hardware or software given the corresponding software or hardware input. Likewise, the model is able to determine the general category or level of the hardware computing capability or software representation capability.

The significance of SHOM is enhanced by the need for agents embedded in the things to be endowed with a reasoning ability. Consequently, a Value-added Service Analyzer is required to improve agents' actions in serving other value-added services to humans. For example, in the RAM system, the agent in the RAM monitors the vehicle numbers on the road and if an alert about insurance state is placed for a particular vehicle number, the agent responds by informing the relevant authority.

The limitation in the SHOM model is represented by two main issues, which is limited hardware and software specifications and granularity. First, this model does not illustrate detailed specifications about the hardware and software found from the matching process. It is purposely designed to determine a specific HW/SW level with its description but without further technical details. The main reason is due to the HW/SW short life span. If we specify our model with detailed HW/SW specifications based on the current technology, it will be obsolete in a year or less, which will render our model obsolete too. For example, the computer specifications that we used ten years ago, which sits on the top level of the HW spectrum, is now a specification of a low end mobile device in the market, which is at the second level of the HW spectrum. Therefore, we opt to make SHOM a generic and flexible model that can be applicable for any era leaving the HW/SW details to the users' considerations.

The second issue is represented by the granularity of the HW/SW spectra levels. The idea behind this issue is based on how coarse or fine the levels' granularity, i.e. the number of the HW/SW spectra levels that could be conceived. Our studies of identifying the HW/SW spectra levels are based of the current technology with which the corresponding level devices and software are designed. We could have more HW/SW spectra levels (not just four of each) if we consider the finer details of the HW/SW in the studies. In our scope, the SHOM model limits the optimality of the working things in the real world. We reckon that the finer the HW/SW levels, the higher the optimality of the designed things.

7. CONCLUSION AND FUTURE WORK

In this paper, we introduce a Software-Hardware Optimizer Model (SHOM), which helps in the design of optimized things that form the components of AoT systems. We review the software and hardware analyses to focus on the important factors that improve the process of finding a suitable hardware-software match from inputs. The combination of these two analyses produces a structured hardware-software optimizer (SH-SO) for SHOM. Then, we represent rules in a flowchart that the SH-SO uses to find the optimum match for the inputs which results in the HW-SW combination. SHOM then tests this (HW-SW) for its ability to run or execute the reasoning capability to produce "Optimized Things".

We present a scenario of a traffic system in which we apply the AoT concept and test the SHOM to demonstrate its efficiency and capability to produce the optimized things for the system. The result shows that the model is able to find the optimum match for both types of inputs, software or hardware. Similarly, it is able to be adaptive in finding the correct match whatever the system design is, as we notice the differences in the design of the traffic system example. Finally, we discuss our findings and how far the model proves its capability and shed some light on the limitations of the SHOM model.

For the future work, we shall investigate a Model of Things that include SHOM and the Value-added Service Analyzer in one general model. Moreover, we shall discuss the internal operations, outcome and the importance of producing an optimum reasoning capability and the relations between these combined models. On the other hand, the SHOM model can be improved by upgrading the structured hardware-software optimizer and including more details about hardware and software specifications. Finally, a practical demonstration of the AoT traffic system will be established to validate the concept that uses the RAM and an agent-endowed vehicle.

ACKNOWLEDGEMENT

This work is sponsored by Ministry of Higher Education (MOHE), Malaysia under the Fundamental Research Grant Scheme (FRGS). Under the grant number 20140116FRGS.

REFRENCES:

[1] T. O'reilly, "What is Web 2.0: Design patterns and business models for the next generation of software," *Communications & strategies,* p. 17, 2007.

[2] E. Rahimi, J. van den Berg, and W. Veen, "Facilitating student-driven constructing of learning environments using Web 2.0 personal learning environments," *Computers & Education,* Vol. 81, pp. 235-246, 2015.

[3] A. P. Ribeiro, H. Barranha, and R. Pereira, "Towards the metaphorical transformation of urban space: Digital Art and the City after Web 2.0," in *1st International Symposium Global Cities and Cosmopolitan Dreams*, 2015.

[4] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American,* Vol. 284, pp. 28-37, 2001.

[5] A. Hogan, P. Hitzler, and K. Janowicz, "Linked dataset description papers at the semantic web journal: A critical assessment," *Semantic Web,* Vol. 7, pp. 105-116, 2016.

[6] A. Zaslavsky and D. Georgakopoulos, "Internet of Things: Challenges and State-of-the-Art Solutions in Internet-Scale Sensor Information Management and Mobile Analytics," in *Mobile Data Management (MDM), 2015 16th IEEE International Conference on*, 2015, pp. 3-6.

[7] R. Glass, "The Impact of Disruptive Technology: The Internet of Things," 2015.

[8] J. Jermyn, R. P. Jover, I. Murynets, M. Istomin, and S. Stolfo, "Scalability of Machine to Machine systems and the Internet of Things on LTE mobile networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, 2015, pp. 1-9.

[9] L. Tan and N. Wang, "Future Internet: The Internet of Things," in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, 2010, pp. V5-376-V5-380.

[10] A. M. Mzahm, M. S. Ahmad, and A. Y. Tang, "Agents of Things (AoT): An intelligent operational concept of the Internet of Things (IoT)," in *Intelligent Systems Design and Applications (ISDA), 2013 13th International Conference on*, 2013, pp. 159-164.

[11] A. M. Mzahm, M. S. Ahmad, and A. Y. Tang, "Enhancing the Internet of Things (IoT) via the Concept of Agent of Things (AoT)," *Journal of Network and Innovative Computing,* vol. 2, pp. 101-110, 2014.

[12] A. M. Mzahm, M. S. Ahmad, and A. Y. Tang, "Computing hardware analysis for Agents of Things (AoT) applications," in *Information Technology and Multimedia (ICIMU), 2014 International Conference on*, 2014, pp. 223-228.

[13] A. M. Mzahm, M. S. Ahmad, A. Y. Tang, and A. Ahmad, "Software Analysis for Agents of Things (AoT) Applications," 2015.

[14] A. M. Mzahm, M. S. Ahmad, A. Y. Tang, and A. Ahmad, "Towards a Design Model for Things in Agents of Things," in *Proceedings of the International Conference on Internet of things and Cloud Computing*, 2016, p. 41.

[15] J. Bughin, M. Chui, and J. Manyika, "An executive's guide to the Internet of Things," *McKinsey Quarterly, McKinsey&Company,* 2015.

[16] M. Parashar, M. Abdelbaky, M. Zou, A. R. Zamani, and J. Diaz-Montes, "Realizing the Potential of IoT Using Software-Defined Ecosystems," in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 1149-1158.

[17] A. Hakiri, P. Berthou, A. Gokhale, and S. Abdellatif, "Publish/subscribe-enabled software defined networking for efficient and scalable IoT communications," *IEEE Communications Magazine,* Vol. 53, pp. 48-54, 2015.

[18] W. Ramirez, V. B. C. Souza, E. Marin-Tordera, and S. Sanchez, "Exploring potential implementations of PCE in IoT world," *Optical Switching and Networking,* 2015.

[19] L. Atzori, A. Iera, and G. Morabito, "Social Internet of Things: Turning Smart Objects into Social Objects to Boost the IoT," *Newsletter,* vol. 2016, 2016.

[20] B. Tabbara, A. Tabbara, and A. Sangiovanni-Vincentelli, "Hardware and software representation, optimization, and co-synthesis for embedded systems," *a:= a,* Vol. 1, p. S2, 2000.

[21] J. W. Tang, Y. W. Hau, and M. Marsono, "Hardware/software partitioning of embedded System-on-Chip applications," in *Very Large Scale Integration (VLSI-SoC), 2015 IFIP/IEEE International Conference on*, 2015, pp. 331-336.

[22] M. Kolasa and R. Dlugosz, "An advanced software model for optimization of self-organizing neural networks oriented on implementation in hardware," in *Mixed Design of Integrated Circuits & Systems (MIXDES), 2015 22nd International Conference*, 2015, pp. 266-271.

[23] R. Lysecky and F. Vahid, "A configurable logic architecture for dynamic hardware/software partitioning," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 2004, pp. 480-485.

[24] L. Yu, L. Xuemei, Z. Jian, and X. Yuning, "Research on the innovation of strategic business model in green agricultural products based on Internet of things (IOT)," in *e-Business and Information System Security (EBISS), 2010 2nd International Conference on*, 2010, pp. 1-3.

[25] X. Jia, J. Wang, and Q. He, "IoT business models and extended technical requirements," in *IET International Conference on Communication Technology and Application (ICCTA 2011)*, 2011, pp. 622-625.

[26] H. Zhang and C. Meng, "A multi-dimensional ontology-based IoT resource model," in *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on*, 2014, pp. 124-127.

[27] H. Zuerner, "The Internet of Things as greenfield model: A categorization attempt for labeling smart devices," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, 2014, pp. 5-9.

[28] S. F. Abedin, M. Alam, G. Rabiul, R. Haw, and C. S. Hong, "A system model for energy efficient green-IoT network," in *Information Networking (ICOIN), 2015 International Conference on*, 2015, pp. 177-182.

[29] B. Kang, S. Park, T. Lee, and S. Park, "IoT-based monitoring system using tri-level context making model for smart home services," in *Consumer Electronics (ICCE), 2015 IEEE International Conference on*, 2015, pp. 198-199.

[30] S. Poslad, S. Middleton, F. Chaves-Salamanca, R. Tao, O. Necmioglu, and U. Bugel, "A Semantic IoT Early Warning System for Natural Environment Crisis Management," 2015.

[31] L. Ciabattoni, G. Cimini, F. Ferracuti, and G. Ippoliti, "Humidex based multi room thermal comfort regulation via fuzzy logic," in *Consumer Electronics (ISCE), 2015 IEEE International Symposium on*, 2015, pp. 1-2.

[32] D. M. Berry, "The Philosophy of Software: Code and Mediation in the Digital Age," 2011.

[33] X.-m. Cao, "The Research on Wireless Sensor Network for Mechanical Vibration Monitoring," in *2015 International Conference on Intelligent Systems Research and Mechatronics Engineering*, 2015.

[34] M. Tancreti, V. Sundaram, S. Bagchi, and P. Eugster, "TARDIS: software-only system-level record and replay in wireless sensor networks," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, 2015, pp. 286-297.

[35] K. Kim and H. Myung, "Sensor Node for Remote Monitoring of Waterborne Disease-Causing Bacteria," *Sensors,* Vol. 15, pp. 10569-10579, 2015.

[36] A. R. Deshmukh, R. R. Sonawane, A. S. Shedwad, P. D. Humane, and R. Satao, "Fast Detection of Replica Node in Mobile Sensor Network," 2015.

[37] L. Gao, H. Yin, Y. Wei, and L. Wang, "Data Collection Methods Based on Mobile Sink Node," in *2015 International Conference on Advances in Mechanical Engineering and Industrial Informatics*, 2015.

[38] A. K. Dennis, *Raspberry Pi Home Automation with Arduino*: Packt Publishing, 2013.

[39] N. Agrawal and S. Singhal, "Smart drip irrigation system using raspberry pi and arduino," in *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, 2015, pp. 928-932.

[40] A. K. Dennis, *Raspberry Pi home automation with Arduino*: Packt Publishing Ltd, 2015.

[41] P. Teja, V. Kushal, and K. Srinivasan, "Photosensitive security system for theft detection and control using GSM technology," in *Signal Processing And Communication Engineering Systems (SPACES), 2015 International Conference on*, 2015, pp. 122-125.

[42] K. Propp, A. Fotouhi, and D. J. Auger, "Low-cost programmable battery dischargers and application in battery model identification," in *Computer Science and Electronic Engineering Conference (CEEC), 2015 7th*, 2015, pp. 225-230.

[43] G. B. Shelly and M. E. Vermaat, "Discovering Computers-Fundamentals 2011 Edition," 2010.