Journal of Engineering and Applied Science

RESEARCH Open Access

Realizing an excellent solution for detecting and solving conflicts between viewpoints of designers in self-adaptive systems

The-Can Do*

*Correspondence: dtcan@dut.udn.vn

Faculty of Mechanical Engineering, The University of Danang, University of Science and Technology, 54 Nguyen Luong Bang Street, Lien Chieu District, Danang City, Vietnam

Abstract

Recently, given the rise of emerging demands on intelligent technological systems, increasing attention has been given to proliferating and enhancing the adaptation capacities of self-adaptive systems (SAS), which are capable of adapting their behavior to changes in the context or system resources contributing to overcome the complexity of today's software-intensive systems. However, there are a significant number of challenges for developing adaptations on the SAS, such as simplifying the designer's task, improving responsiveness, and reducing the conflicts between its adaptations. This study focuses on simplifying the designers' tasks by utilizing independent designers' viewpoints on context modeling. We proposed several solutions for solving the conflicts among different viewpoints of designers in the layer of context-aware management (CAM). The validation results are promising and show that our method effectively provides a solution that supports the problem of using independent viewpoints in the context modeling process and improves the SAS adaptation capacity.

Keywords: Context-awareness, Context management, Context modeling, Self-adaptive system

Introduction

Under rapid technology development and continuous advancement in society, demands on information systems and related devices that consider properties such as flexibility, dependability, customizability, and adaptability have become increasingly vigorous [1]. The SAS, which is the primary component operating automatically in the framework of the system and interacting with the system on behalf of the user, plays an inevitable role in maintenance and configuration and is integrated into ubiquitous devices more than ever before [2, 3]. Therefore, several services at the inside of interactions between ubiquitous devices and the application layer are provided by the SAS. Furthermore, it allows the services or applications of the system to use the context information to change their behaviors according to the situations of the device and user.

The contextual information around users, such as connections and devices, is changed due to their various locations. Hence, the user's mobility in the design of the SAS should be considered. It means that the SAS, whose context model is required to cover all user

scenarios, must auto-adapt applications that can intelligently adjust to numerous environment changes [4]. However, the modeling context in the SAS based on predicting all user's situations and application scenarios can increase the number of required concepts to descibe during the context modeling process. Moreover, the designer has to combine knowledge of different expert domains to design a single big context model, making the designer's task highly complicated. It would be better if the single large context model is separated into multiple independent context models in which the designers utilize their knowledge to present the scenario of user and application in terms of their perspective. That work can help each designer's task become simpler than before. In addition, the use of independent models also makes it easier to develop or reconfigure the context model system in the SAS.

Moreover, the data obtained after the modeling process based on the perspectives of various experts have different data structures and types as a result of their implementation of dissimilar modeling techniques or tools. It is therefore essential that the data obtained needs to be standardized by standard formats for all views (e.g., the context-intermediate model (CIM)) [5]. The standardized data is then added to context-aware management (CAM) to help the SAS in context processing [6]. The CAM consists of two phases, i.e. the design time where the designer uses specific models to describe the views and the runtime for which all the special views are independently managed was used to manage all specific views and detect changes in context [7]. Each viewpoint in work at an observation cycle stores one state of adaptation that is defined by one predication and each state of the adaptation has its objective. Necessary implementation steps to fulfill the identified goals are defined by the objective of the state (OS). In particular, each OS with a defined action is specific and measurable. At one moment, if the OS of one viewpoint contradicts or is incompatible with the OS of the others in a SAS, then there will be a conflict between the viewpoints that are not easily determined since it exists in many aspects of the response. Wang et al. presented that "conflict is a natural disagreement between different attitudes, beliefs, values or needs" [5]. The conflict between viewpoints also is a conflict between the OSs. The conflict between OS occurs when there is an incompatibility in desired end states or preferred outcomes. Hence, it is worth designing solutions for addressing the conflicts.

The works [6, 8–10], and [11] indicated that the present SASs are capable of detecting and resolving conflicts between adaptation rules at runtime. However, further enhancing SAS's adaptive capacity requires a significant increase in the number of adaptive rules, resulting in determining solutions for the conflicts becoming more complex and consuming time. Moreover, since all user scenarios associated with context cannot be predicted, resolving the conflicts between viewpoints at design time is impossible. In addition, adding new perspectives is mainly dependent on the user's choice, causing the challenge and difficulty of using all of the viewpoints together in a system. For example, the CAM manages two viewpoints, and two objectives of the state in these viewpoints may remain conflicts in the same period and place, such as viewpoint 1 with increasing the light intensity and viewpoint 2 with decreasing the light intensity. It is not easy to solve this problem with the usual system. Thus, a mechanism is needed to find and solve the conflicts between the semantics of viewpoints on the CAM before providing adaptation rules for the deployment process of the SAS [6].

Numerous studies have been conducted in the SAS using "Who, What, When, Where, Why, and How" (5W1H) questions to handle the problems of capturing the adaptation requirements. Salehie et al. use the 5W1H questions to elicit adaptation information: Where

must the change be implemented? When does an adaptation need to be implemented? What needs to be adapted in each situation? Why is the adaptation necessary (identify the goals of the adaptation)? Who must execute the adaptation? And how are the adaptations implemented? [12]. Krupitzer et al. provided the taxonomy to answer the questions of adaptation [13]. They used the 5W1H question as the solution to describe their taxonomy. Answering each question provides an element of adaptation: "When?" is the time of adaptation, "Why?" provides the reason for adapting, "Where?" shows the location of adaptation, and "What?" presents the technique of adaptation, "Who?" is the nature of the system leads to an automatic type of adaptation) and "How?" provides an adaptation control solution.

Many works, such as the works on [14, 15], have used the 5W1H approach or a part of it to model contexts. Kim et al. use 5W1H questions to define the model for contextual information named ontological context-aware model based on 5W1H (CA5W1Honto). They used three elements "Concept, instance and context" to define each object of the contextual information. Moreover, the authors of the CA5W1HOnto model separated ontologies and context information in the form of two independent modules. Therefore, this approach can provide a high level of recyclability and expandability of the context model and some formalism to contextual information through the web ontology language-description logic (OWL-DL). They used six elements "goal, role, location, action, status, time" corresponding to six context modeling ontology elements (why, who, how, where, what, when) to define the context. Rathi et al. introduce a framework named "event and implied situation ontology" (ESO-5W1H) for adjudging the machine and human roles in their corresponding interaction [16]. The ESO-5W1H uses six classes: "Who", "When", "Where", "Why", "What" and "How" to build the basic decision process for the system's adaption. The element "Who" is used to present the role (Device, Signboard, etc.); "When" is used to define causal chains; "Where" is used to describe Location; "Why" is used to describe Goal; "What" is used to describe Status; and "How" is used to describe Action.

From the works mentioned above, we can see that the 5W1H approach can describe the adaptation information. In addition, the ontology-based on 5W1H questions in [15] and [16] shows that the 5W1H framework can also be used to build a domain context ontology for the context model. It provides the ability to concepts and relations analyzing and extracting within the discourse domain. In our approach, we propose using the OS to outline the "what, where, and how" of each state's implementation steps to achieve the primary goal of the viewpoint's state. So, we use also the 5W1H approach or a part of it to build our contextual ontology that is used to describe the OS. For this, we need to provide a standard definition of the OS, and then the designer will use the elements of contextual ontology to describe the OS from all viewpoints at design time.

In this study, we proposed a solution to add two new mechanisms to handle problems associated with detecting and solving conflicts between viewpoints on CAM. In our approach, CAM is independent of the SAS. While CAM can anticipate actions of perspectives to dismantle unnecessary adaptation rules, the SAS manages the adaptation rules without caring about their semantics of it. In particular, the SAS can tackle only the conflicts of access to components or devices. The SAS cannot operate with conflicts related to the semantics of each state in viewpoints. The details of the contextual ontology, objective of state, and conflict detection methodology are analyzed in the next section. Finally, a solution for solving the conflicts between designers' viewpoints is also proposed.

Methods

Contextual ontology and objective of state

In this section, our construction of a contextual ontology database (COD) that stores the essentials for describing the OS, based on the 5W1H conceptual modeling frameworks. In our previous research [17], we used four elements including "What, where, how and priority" to describe the OS. However, if we have two OS with the same composite time, the CAM cannot choose a suitable adaptation for the state. In this case, the CAM needs more information related to the atomic time of OS. So, we proposed an approach that requires five elements of the 5W1H: "what, where, when, who and how" [18–20]. In our methodology, "Why" is not used to describe the OS. The designers will explain their choice in the official language or human language (for better understanding by the user and the system). This means that the reason for adaptation corresponds to the designer's choice.

Figure 1 presents the domain concept in our contextual ontology (CO) which is composed by the union of several sets. Each set stores five aspects of 4W1H ontology (what-where-when-who and how). The combination of 4W1H is given by:

$$CO = C_{what} \cup C_{where} \cup C_{when} \cup C_{who} \cup C_{how} \tag{1}$$

$$\text{With} : C_n \cap C_m = \varnothing(\text{n\#m}\Lambda C_n, C_m \epsilon \{C_{\text{what}}, C_{\text{where}}, C_{\text{when}}, C_{\text{who}}, C_{\text{how}}\}) \tag{2}$$

where, C_{what} represents the class of concepts related to objects which are impacted during the adaptation process of SAS (such as Humidity, temperature, etc.). C_{where} shows the location of objects (Parking, Office, etc.) respectively. C_{when} is used to describe the time of adaptation need. C_{who} is used as an entity to indicate the priority level of adaptation [15]. C_{how} denotes the set of concepts about the action that needs to be performed to reach the adaptation's goal (Activate, deactivate, etc.).

Additionally, the designer can utilize the information obtained from the 4W1H database to demonstrate the OS in the design time. With {*O,L,T,P,A*} being a set of concept names, the term O can be defined as *O* [*o,l,t,p,a*]. Where:

o ∈ *O*, with *O* as a class of concepts related to objects on "What element" (i.e., temperature, light intensity).
l ∈ *L*, with *L* as a class of concepts related to places or locations on "Where elements" (i.e., Parking, Office).
t ∈ *T*, with *T* as a class of concepts related to times on "When elements" (i.e., daytime, nighttime).

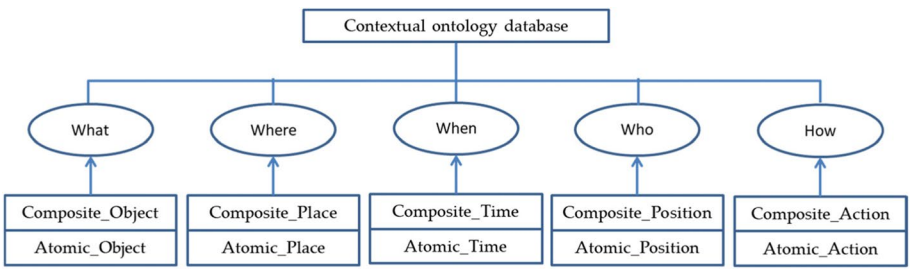


Fig. 1 Contextual ontology 4W1H

$p \in P$, with P as a class of concepts related to the position of the user on "Who elements" (i.e., manager, security).
$a \in A$, with A as a class of concepts related to actions on "How elements" (i.e., activate, turn of).

Detecting and solving methods

CAM can handle states of multiple viewpoints at once, causing the possibility of conflict between any two objectives of the state under different viewpoints. Thus, detecting and solving conflicts between OSs on CAM pleasure a leading role in improving the responsiveness of the SAS. This section demonstrates detecting conflicts between the OSs in different viewpoints, and a solution to solve these conflicts problem.

Conflict detection

Figure 2 illustrates the structure of the SAS in our approach [17]. In the SAS as shown in Fig. 2, if there are n viewpoints used on one CAM system, at one moment, the CAM will manage n states corresponding with N viewpoints. This means that if the system has N states corresponding with the N viewpoint, the CAM has to manage N OS at once. It is required an OS merging process to find and solve the conflicts between them. In addition, it provides also incompatibility situations between conflicting OSs.

In our approach, the five elements "what, where, when, who and how" of the states' objectives are employed to analyze situations of conflict between viewpoints. Of which, $\{o, l, t, p, a\}$ are five concepts corresponding to the information used to define OS (included elements of "what, when, where, who and how" class) of two OSs: $O_{1(e)}$ and $O_{2(e)}$. Figure 3 presents the association between each element of two OSs (O_1 and O_2).

In terms of element "What," with $O_i(o) = o_i$ ($i = 1, 2$, $o_i \in \{C_{What}\}$), there are four relation situations of $O_1(o)$ and $O_2(o)$ as shown in Fig. 3. As illustrated in Fig. 3(a), O1(o) and O2(o) have same object to be adjusted (i.e., $O_1(o) = O_2(o) =$ Light intensive), Fig. 3(b) show the totally different objects in O_1 and O_2 (i.e., $O_1(o) =$ Humidity, $O_2(o) =$ Devices). Meanwhile, in Fig. 3(c), O1(o) and O2(o) exist intersection relation (i.e., $O_1(o) = (200^0$C

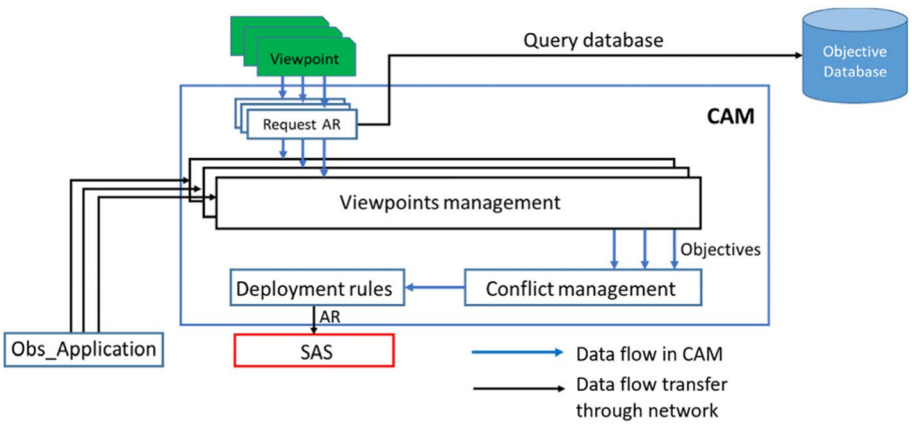


Fig. 2 Self-adaptive system (SAS) diagram

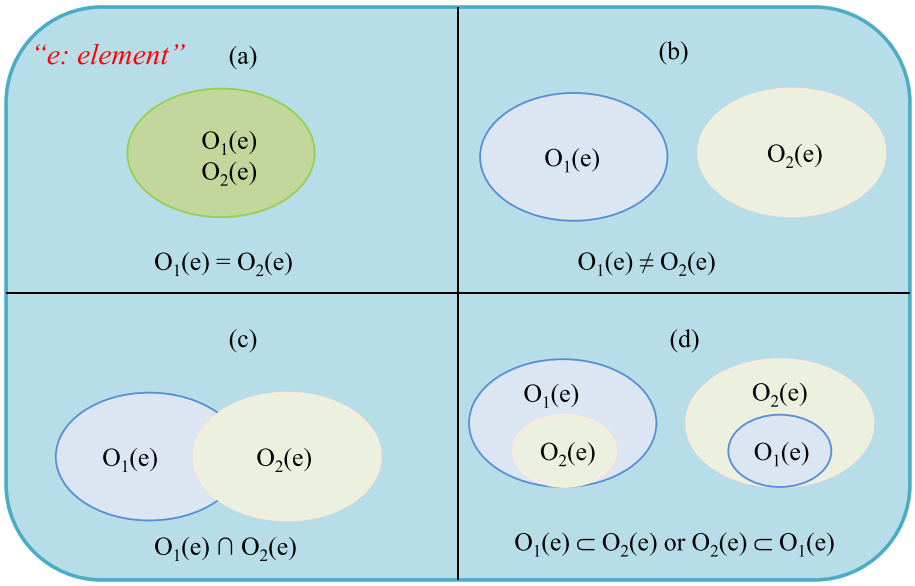


Fig. 3 The relationship between two elements $O_{1(e)}$ and $O_{2(e)}$ with different cases of **a** having the same object, **b** having different objects, **c** having an intersection relation, and **d** having a subspace in each object

to 350^{0}C) and $O_1(o)$ = (250^{0}C to 450^{0}C), and in Fig. 3(d), O2(o) is a subspace of $O_1(o)$ or $O_1(o)$ is a subspace of $O_2(o)$ (i.e., $O_1(o)$ = (150^{0}C to 450^{0}C) and $O_1(o)$ = (250^{0}C to 350^{0}C)).

Similarly, for the elements "Where" and "How", there are also four relation situations of *O(l)* and *O(a)* corresponding to their cases. ($O_1(l) = O_2(l)$ = Car parking) or (O_1(a) = O_2(a) = Deactivate) is an example of the situation O_1 and O_2 have the same location or same action, respectively. In some other cases, O_1 totally different to O_2 both location and action (i.e. $O_1(l)$ = Department and $O_2(l)$ = Office or O_1(a) = Deactivate and O_2(a) = Turn On). There is even the existence of an intersection relation between O_1 and O_2 of the elements "Where" and "How", for instance, O_1(l) = Second floor of department and O_2(l) = Corridor of department).

If two OSs have the same action, it is confirmed that they will never have conflict. In contrast, if two OSs have different actions, the CAM must check the relation of these actions to determine whether these actions conflict or not conflict. In the 4W1H contextual ontology database, the relationship (R) between section "*O(a)*" of two OSs can be easily constructed as presented in Fig. 4. In "How class" of contextual ontology, the relation of two actions Y and Z obtain two values a or $\overline{a}$. We have R (Y, Z) = a if the action Y of OS_1 does not involve action Z of OS_2 (i.e., decrease and deactivate) and is R(Y, Z) = $\overline{a}$ if the action Y of OS_1 is opposite or incompatible with the action Z of OS_2 (i.e., Increase and Decrease). The relationship between all elements of "How" class (including their subclass) is based on the rules:

If action Y conflicts with action Z:

$$\mathrm{R(Y,Z)} = \overline{a} \rightarrow \mathrm{R}\big((\forall y \epsilon \mathrm{Y}), (\forall z \epsilon \mathrm{Z})\big) = \overline{a} \tag{3}$$

If action Y does not involve action Z:

$$\mathrm{R(Y,Z)} = a \rightarrow \mathrm{R}\big((\forall y \epsilon \mathrm{Y}), (\forall z \epsilon \mathrm{Z})\big) = \mathrm{a} \tag{4}$$

Conflict case definition: Objective (O1) CONFLICT objective (O2) (Table 1), if:

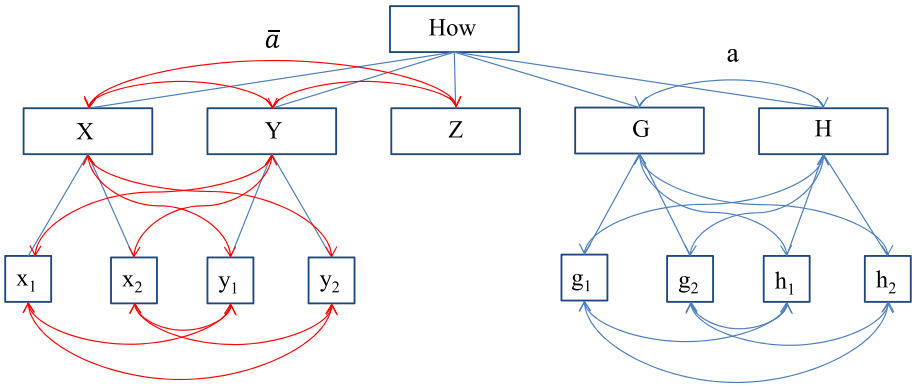


Fig. 4 The relation properties between elements in "How class" of 4W1H ontology with R(x,y) = a when the action x does not affect action y and R(x,y) = $\bar{h}$ when the action x is incompatible or opposite with action y [17]

Table 1 Objective of states conflict analysis results

No	$R_{1,2}(.)$	Conflict analysis "What, Where"	Conflict analysis "When"	Relation of $O_1(a)$, $O_2(a)$ "How"	OS conflict
1	olta	Same object, same place	Same time	Not affect	Not conflict
2	olt $\bar{a}$	Same object, same place	Same time	Incompatible opposite	Conflict
3	ol $\bar{t}$ a	Same object, different place	Different time	Not affect	Not conflict
…	…	…	…	…	…
16	$\overline{olta}$	Different object, different place	Different time	Incompatible opposite	Not conflict

- *Action of O_1 incompatible with or opposite to action of O_2*
- *These actions impact to same "object" in same "place" and same "time"*

Example 1: As the conflict case mentioned in the introduction, if we have two application scenarios:

- Viewpoint 1: In the office: if it is daytime and "light intensity < M = yes", then increase the light intensity.
- Viewpoint 2: In the office: if it is daytime and "experiment completed? = yes", then decrease the light intensity.

Randomly during the daytime, the "experiment completed? = yes" and "light intensity < M = yes", we have:

From the analysis result in Table 2, we can conclude that O_1 conflicts O_2.

Solving conflicts between designers' viewpoints

In this section, we present the solution for solving the conflicts between designers' viewpoints. If at least two viewpoints are applied, the CAM has to supervise at least two objectives of states at every moment. Each state of viewpoint has its own OS, hence, the

Table 2 Conflict situation analysis from example 1

Scenarios	Object (what)	Place (where)	When (time)	Action (how)
Viewpoint 1 (Security)	Light intensity	office	Daytime	increase
Viewpoint 2 (Employee)	Light intensity	office	Daytime	decrease
Relation (O_1 and O_2)	Same object	Same place	Same time	Opposite action

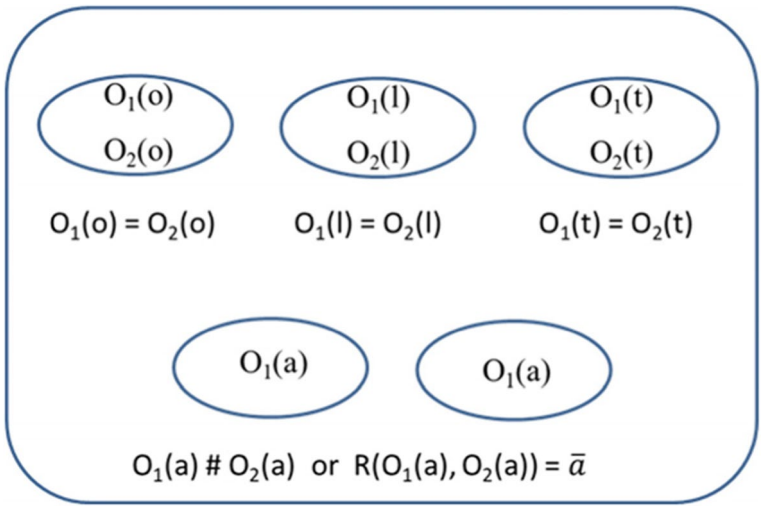


Fig. 5 The diagram of the relationships of three elements of two OSs when there is a conflict situation between two OSs

CAM must find out the conflict situation of any two OSs by checking each pair of all OSs. Two OSs exist in conflict when four elements of these OSs have a relation as shown in Fig. 5.

SAS will do the action in each state of viewpoint following the adaptation rules (AR) built by the experts at design time. However, each expert designer has different ways to describe his adaptation rules following his concern. So, the CAM cannot work on the rules level [21] to find and solve the conflict between OSs. Nevertheless, the CAM cannot adjust the action through adaptation rules in each state of viewpoint; it can choose the best adaption rule to suit the needs for adaptation at one time. However, the problem here is: "How to find what is the best action for adaptation?". In this case, all designers must provide additional information to support the CAM in choosing the adaptation rule of OS. Furthermore, one of the most used solutions is adding priority levels for all states of each viewpoint. The SAS can use priority levels from viewpoints as important indexes to select more valuable adaptation rules when the CAM detected the conflicts between OSs.

Many researchers have used "Priority" as an important index for evaluating or choosing adaptation rules on research domains such as security, health care, management, etc. [22–24]. While Spicker proposes to use five priority levels (Lexical ordering, special status, precedence, relative value, and importance,) in

choosing adaptation rules [25]. Spicker agreed that the "priority" of something shows that it is more important than others. As a result, many approaches have used a level of priority to provide well services to adapt the versatile user situations. In the above approaches, the designer or user points out conflict situations, and then the CAM use the priority of the OS as an important index to support the SAS in selecting services or applications if the CAM find out the conflict situations. In this approach, we find the motivation to use the "Priority level" to describe each OS's importance in viewpoints. Moreover, we believe that it is a well-suited solution to support the SAS in detecting and solving conflicts between OSs at run time. In our viewpoint, a SAS must provide the adaptation rules based on four levels of user: manager, security guards, employees, and guests. So, in our approach, we use four elements of "Who class" including "Manager, security, employee, and guest" corresponding with four objects of adapt, as shown in Table 3. The priority element of each OS shows the importance of the viewpoint's states. Following that, the SAS can select the best adaptation rules in conflict cases.

With the addition of the priority element in the OS description, each OS will be established by five elements corresponding with concepts from five classes (what, where, when, who and how) of the contextual ontology database, as shown in Fig. 6.

All OSs that conflict with at least one other OS will be classified into four subclass PL_1 to PL_4 corresponding with four important levels of OSs. Next, the conflict-solving step is done by the CAM, as shown in Fig. 7.

Table 3 Priority levels used on 4W1H ontology database

Level	Who	Priority
PL_1	Manager	Very important
PL_2	Security	Important
PL_3	Employee	Normal
PL_4	Guest or Device	Low

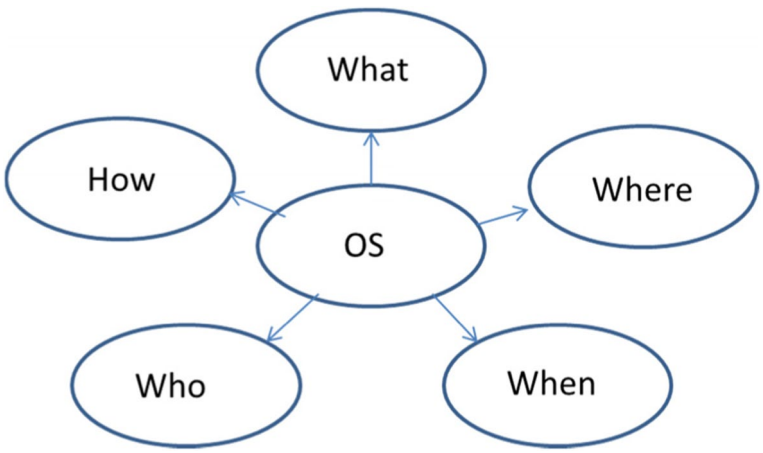


Fig. 6 The Objective of state description with 4W1H with five elements: what-where-when-who-how

At time t, for an example, the observation predicate cycle of the CAM can detect n-OS (*n is the total number of all OS at one time*). The CAM will classify them into four levels, from PL_1 to PL_4 (corresponding with four levels of priority), as shown in algorithm 1:

```
//algorithm (1)
int i = n;// constant
int e = 0, f = 0, g = 0, h = 0;
string[] PL1 = { };// PLi is priority level OS
string[] PL2 = { };
string[] PL3 = { };
string[] PL4 = { };
for (int j = 1; j <= i; j++) {
  switch (Oj[p]) {// get priority level of Oj
    case "Manager":   PL1[e] = "O" + j; e = e + 1; break;
    case "Security":   PL2[f] = "O" + j; f = f + 1; break;
    case "Employee":      PL3[g] = "O" + j; g = g + 1; break;
    case "Guest or device":         PL4[h] = "O" + j; h = h + 1; break;
                        }
 }
```

Furthermore, the CAM compares all OSs at the same priority level (PL_i) to find the same OSs in this priority level (two OSs is called the same if five elements them are the same ($O_i\ (o) = O_j\ (o)$, $O_i\ (l) = O_j\ (l)$, $O_i\ (t) = O_j\ (t)$), $O_i\ (p) = O_j\ (p)$, $O_i\ (a) = O_j\ (a)$). Suppose the CAM has class PL_i (i = 1 to 4) containing OSs with the same priority level. It can keep one of them and determine the number of viewpoints with identical OS in each class PL_i following algorithm 2:

```
//algorithm (2)
// applied for PL1
int [] a = new int[1];
int n = N^0_PL1; //N^0_PL1 is the number of OSs in PL1 class
for (int b=1; b < n; b++) {
    for (int c = b + 1; c <= n; c++) {
        if (PL1[b] == PL1[c]) { //(Ob[o,l,t,p,a] = Oc[o,l,t,p,a] and (Ob, Oc) ϵ PL1)
            a[b] = a[b] + 1; // determine the number of viewpoints with identical OS
            int index =c;
            for(int i=index; i<n-1; i++){
            PL1[i] = PL1[i+1];    //Delete PL1[c]
            --n;
                    }
               }
       }
}
```

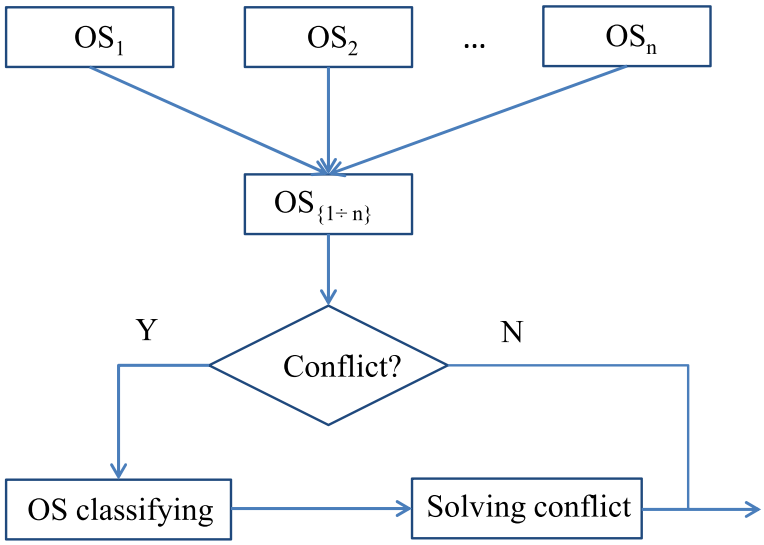


Fig. 7 Schematic of classifying and solving conflict between multiple viewpoints

Results and discussion

In the previous section, we proposed to use four priority levels in the "Who class" including "Manager, Security, Employee, Guest and Device" to point out the important level of each OS. Based on the priority level, the solution that solves conflicts between OSs can be expressed logically. This solution based on the proposed method can be illustrated in three following steps in some issues which can be happened in system.

Step I: Determine the number of designers' viewpoints which has the same OS in each priority level

- ***Issue 1*:** After the classifying OS step (use algorithm 1), suppose we have nine OSs where each OS conflicts with at least one other OS, and $O_2 = O_8$, and the number of OS in each class of priority is shown in Table 4.

Table 5 presents the result of the calculation process (use algorithm 2) to determine the number of designers' viewpoints which has the same OS in each priority level at runtime.

Step II: Find out the conflict between OSs in the same priority

The SAS examines the conflicts among OSs at the same level of priority (element "Who") from PL_1 to PL_4. In the same PL_i, the CAM compares the number of conflicts between each OS with others. After that, the CAM ignores the OS has the highest number of conflicts. It updates the OS list in each class and checks again without ignoring OS. This process repeats until it has no conflict between viewpoints in all classes. If any two OSs (O_i and O_j) have the same conflict number compared to others; the CAM must check the number of viewpoints associated with them. It keeps the OS with a higher number of viewpoints related to that OS and ignores the other. If two OSs have the priority level

Table 4 The priority of objective of the state

Who class	OS
PL1	O_2, O_3, O_6, O_8
PL2	O_5, O_4
PL3	O_1
PL4	O_7, O_9

Table 5 The result of merging the same OSs in each priority level

	Decision	Merging results
PL_1	Ignore O_8 because $O_2 = O_8$ Concatenate (2) with O_2 (a[2] = 2)	$O_2^{(2)}$, O_3, O_6
PL_2	O_4 # O_5	O_4, O_5
PL_3	-	O_1
PL_4	-	O_7, O_9

and the number of viewpoints associated with them equally, it must keep randomly one of them for responding.

- ***Issue 2:*** With the previous example in Issue 1, at PL_1 level, if $O_2 > < O_3$ (we propose to use the symbol " > <" to replace the conflict situation), and they do not conflict with any other OS. The result of step 2 shows in Table 6.

In the above example, the amount of conflict with other OS of O_2 and O_6 is equal. However, O_2 is used by more than one viewpoint, while O_6 is used by only one viewpoint. Therefore, the CAM keeps O_2 (ignore O_6), then it has to update the situation of OSs as shown in Table 7.

In case of two OSs have the same viewpoints associated with them and there is a conflict between them, the CAM will examine the conflict between these OSs with other OSs in the lower priority level. Next, the CAM ignores the OS having the most conflict with others. In a special case, if these two OSs have the same amount of conflict as other OS in the lower level, the CAM has to keep randomly one of these OSs for responding.

Assuming that the conflict situations between OSs in the above example can be: {$O_4 > < O_5$; $O_4 > < O_1$, $O_4 > < O_7$ and $O_5 > < O_1$}, the results of checking conflict in the PL_2 class can be illustrated in Table 8.

In the PL_2 class, because the conflict number of $O_4 = O_5 = 1$, they have the same number of viewpoints associated with them (k[4] = k[5] = 1). The CAM has to check the conflict situation between them with OSs at the lower priority level. And the result of checking and solving conflict is presented in Table 9.

Step III: Checking the conflicts with OSs in the lower priority level

If the CAM cannot find out the conflict between OSs at the same priority level, it will check the conflicts between Oss at the highest priority level with other OSs at other lower levels:

Table 6 Conflict and result of conflict-solving process

	O_2	O_3	O_6
O_2			
O_3			
O_6	><		

PL_2	Conflict number	Decision
O_2	1	Retain O_2 (a[2]) > a[6])
O_3	0	
O_6	1	Ignore O_6 (a[6])=1)

- If a conflict exists between OSs at the highest level and other OSs at the lower level, the CAM ignores the OS in the lower PL_i.
- If the CAM does not detect any conflict, it will check for the next lower level in the "Who class".

The CAM will continue checking and solving the conflict process until not no conflict is detected; it will provide a new set of OSs without conflict for the deployment rules process. With the example above, the result of the solving conflict process can be expressed as $O^* = O_2 + O_3 + O_5 + O_7 + O_9$. This is the most suitable solution for giving the determination of the system based on the viewpoints.

Based on some mentioned issues and the solving process, the conflicts between viewpoints at CAM before executing the adaptation rules for the SAS can be found and resolved by the proposed method in the paper. When we make an application using multiple viewpoints to model the context, the conflicts between the adaptation of the viewpoints can happen at run time. This disadvantage occurs because the viewpoints are designed by different designers. Each designer only cares about the purpose of their application scenario. So, when the CAM merges the viewpoints at runtime, the corresponding adaptation rules of each viewpoint may conflict with each other. These conflicts can not be resolved at the design time because the designer does not have the instance information of the components which are impacted by the adaptation of the SAS at runtime. The goal of each viewpoint state by OS can become the key to solving the problem. It can be used to find out the potential conflict between designers'

Table 7 The updated table of the OS list

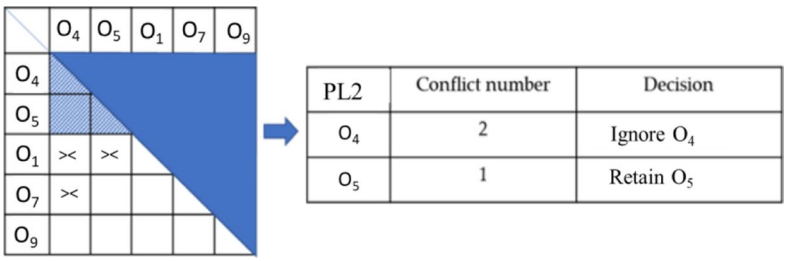
	OS	N_{pi}
PL1	$O_2^{(2)}, O_3$	2
PL2	O_4, O_5	2
PL3	O_1	1
PL4	O_7, O_9	2

Table 8 The conflict number of OSs in class PL_2

	O_4	O_5
O_4		
O_5	><	

PL2	Conflict number
O_4	1
O_5	1

Table 9 The result of solving conflict



viewpoints. The conflicts between viewpoints at a time are also the conflicts between the OS of viewpoints. Therefore, the resolution is introduced to detect and solve the conflicts between the OS of viewpoints. This solution can reduce the number of conflicts between viewpoints that need to be solved in the SAS. However, this solution also has its disadvantage. It requires the system to spend a lot of time checking, selecting, and updating the OS situation during the adaptation process.

Conclusions

When we make an application using multiple viewpoints to model the context, the conflicts between the adaptation of the viewpoints can occur at run time. Detecting and resolving conflicts in CAM plays an important role in improving the adaptive capacity of the SAS. However, detecting conflicts between viewpoints through adaptive rules is not possible because there is no information at device instances. In the proposed method, the goal of each viewpoint state by the Objective ontology database is described and used to find out the potential of conflict of designers' viewpoints. The contextual ontology database composes of five classes: what, where, when, who and how. All designers can use the contextual ontology database (4W1H) to present the objective of state for each viewpoint at design time. The CAM detects the conflict between viewpoints by comparing five elements "what, where, when, who and how" of OS at runtime.

In addition, in our research, we propose to add "Who class" (*on contextual ontology database*) as the priority element of each OS for the conflict resolution process. The CAM uses the priority of OS to classify all OS into four levels corresponding from PL1 to PL4. The priority level of OS is an essential index for choosing OS in case of a conflict. The result of detecting and solving conflict process is the list of OS without conflict and the CAM ready for deployment rules to SAS.

In the future, we want to build an observation mechanism to observe context automatically without triggering an observation signal from the CAM. It can operate independently with the CAM to provide useful contextual information for the system. This solution allows using the same observation mechanism for many CAM on the self-adaptive system.

Abbreviations

SAS	Self-adaptive system
CIM	Context-intermediate model
CAM	Context-aware management
OS	The objective of the state
5W1H	Questions to elicit adaptation information

Acknowledgements
This work was supported by The University of Danang, University of Science and Technology, code number of Project: T2021-02-02.

Authors' contributions
The author has contributed significantly to this work, read, wrote, and approved the manuscript; and agree to its submission to the ***Journal of Engineering and Applied Science.***

Funding
This study is funded by The University of Danang, University of Science and Technology.

Availability of data and materials
All data generated or analyzed during this study are included in this article.

Declarations

Competing interests
The author declare that he has no competing interests.

Received: 8 March 2022 Accepted: 12 August 2022
Published online: 29 August 2022

References

1. Vandana B, Purkayastha, Kumar A (2020) Issues and Challenges in Management related to Information Technology. Int J Comput Digit Syst. 9(4):703–713
2. T C Do, G Rey, JY Tigli, S Lavirotte, N, Le Thanh (2019) "From BPMN to Live Application: How the Context Can Drive an Auto-Adapted System", 2019 IEEE-RIVF International Conference on Computing and Communication Technologies, pp. 1–6
3. Macías-Escrivá FD, Haber R, Del Toro R, Hernandez V (2013) Self-adaptive systems: A survey of current approaches, research challenges and applications. Expert Syst Appl 40(18):7267–7279
4. Krupitzer, Christian et al (2015) A survey on engineering approaches for self-adaptive systems. Pervasive and Mobile Computing 17:184–206
5. Wang WM, Ting SL (2011) Development of a computational simulation model for conflict management in team building. Int Journal of Engineering Business Management 3:9–15
6. Sawyer P, Bencomo N, Whittle J, Letier E, Finkelstein A (2010) Requirements-aware systems: A research agenda for re for self-adaptive systems. In: 2010 18th IEEE International Requirements Engineering Conference. pp 95–103 (IEEE)
7. Continuum project, Continuum ANR, Programme VERSO, Continuum ANR-08-VERS-005, 12–2008/09–2012.
8. Anthony R J (2006) Generic support for policy-based self-adaptive systems. In: 17th International Workshop on Database and Expert Systems Applications (DEXA'06). pp 108–113 (IEEE)
9. Pradeep P, Kant K (2022) Conflict Detection and Resolution in IoT Systems: A Survey. IoT 3(1):191–218
10. Welsh K, Bencomo N, Sawyer P, Whittle J (2014) Self-explanation in adaptive systems based on runtime goal-based models. In: Transactions on Computational Collective Intelligence XVI. Springer, Berlin, Heidelberg, pp 122–145
11. Lemos R D, Garlan D, Ghezzi C, Giese H, Andersson J, Litoiu M, Zambonelli F (2017) Software engineering for self-adaptive systems: Research challenges in the provision of assurances. In: Software Engineering for Self-Adaptive Systems III. Assurances. Springer, Cham, pp 3–30
12. Salehie M, Tahvildari L (2009) Self-adaptive software: Landscape and research challenges. ACM transactions on autonomous and adaptive systems (TSAS) 4(2):1–42
13. Krupitzer C, Roth F M, VanSyckel S, Schiele G, Becker C (2015) A survey on engineering approaches for self-adaptive systems. Pervasive Mobile Computing 17:184–206
14. G Rey (2005) "Contexte en Interaction Homme-Machine : le contexteur", PhD Thesis, Universite Joseph-Fourier-Grenoble I.
15. Benaddi H, Laaz N, El Kettani E, Hannad Y (2022) Ontology Model for Public Services in Morocco Based on 5W1H Approach: PSOM-eGovMa. Procedia Computer Science 198:429–434
16. S Rathi, A Alam (2018) "ESO-5W1H Framework: Ontological model for SITL paradigm". HumL@ ISWC 2018, Vol-2169, pp. 51–63. [2nd International Workshop on Augmenting Intelligence with HumansintheLoop co-located with 17th International Semantic Web Conference]
17. Do The Can (2019) A context manager for solving conflicts between designer's viewpoints. Ubiquitous Computing. COMUE Université Côte d'Azur
18. Oh Y, Shin C, Jang S, Woo W (2005) ubi-UCAM 2.0: A unified context-aware application model for ubiquitous computing environments. In the 1st Korea/Japan Joint workshop on Ubiquitous Computing and Network Systems.
19. Hong D, Shin C, Oh S, Woo W (2006) A new paradigm for user interaction in ubiquitous computing environment. ISUVR 2006:41–44
20. Kim JD, Son J, Baik DK (2012) CA 5W1H onto: ontological context-aware model based on 5W1H. Int J Distrib Sens Netw 8(3):247346

21. M Yagita, F Ishikawa, S Honiden (2015) "An application conflict detection and resolution system for smart homes". Proceedings of the First International Workshop on Software Engineering for Smart Cyber-Physical Systems, pp. 33–39, IEEE Press
22. Wan Z. (2009). Priority Based Adaptive Video Transmission Over Ad Hoc Networks. In 2009 International Conference on Multimedia Information Networking and Security (Vol. 1, pp. 277–281). IEEE.
23. Renners L, Heine F, Kleiner C, Rodosek G D (2018) A Feedback-Based Evaluation Approach for the Continuous Adjustment of Incident Prioritization. In: 2018 1st International Conference on Data Intelligence and Security (ICDIS). pp 176–183
24. Wang H, Mehta R, Supakkul S, Chung L (2011) Rule-based context-aware adaptation using a goal-oriented ontology. In: Proceedings of the 2011 international workshop on Situation activity & goal awareness. pp 67–76
25. Spicker P (2009) "What is a priority?" J Health Serv Res Policy 112–116.

Publisher's Note