Reasoning-Based Context-Aware Workflow Management in Wireless Sensor Network*

Endong Tong[1], Wenjia Niu[1], Hui Tang[1], Gang Li[2], and Zhijun Zhao[1]

[1] Institute of Acoustics, Chinese Academy of Science
High Performance Network Laboratory
Beijing, China, 100190
{Tonged,niuwj,tangh,zhaozhj}@hpnl.ac.cn
[2] School of Information Technology
Deakin University, 221 Burwood Highway
Vic 3125, Australia
gang.li@dekin.edu.au

Abstract. Workflow technology is regarded as the automation of an execution process in which the information or tasks are passed from one service to another, according to the predefined execution sequence. Recently, workflow technology has been successfully used in wireless sensor networks (*WSNs*) for service composition. Although workflows can dynamically change according to current context, there is still limited work to facilitate the atomic services reuse.

In this paper, we propose a reasoning-based context-aware workflow management(*Recow*) approach, in which a rule-based reasoning module is responsible for extract semantic information so that the lower sensor data will have a loose couple connection with the upper logic process. By using the semantic information as service I/O, we reconstruct atomic services, then they can be reused in workflow construction. Usually more than one rule will be matched and they can not be executed simultaneously in the rule matching process, so a conflict resolution algorithm is further proposed based on context-aware priority. Finally, two case studies demonstrate that our approach can effectively facilitate resource reuse and also indicate the performance of the *Recow* approach as well as the precision of the conflict resolution algorithm.

Keywords: Wireless Sensor Network, Workflow, Context-aware, Rule-based Reasoning, Resource Reuse.

1 Introduction

Composed of a large number of sensor nodes, wireless sensor networks (*WSNs*) are attractive for physical information gathering in large-scale data rich

* This research is supported by the National Natural Science Foundation of China (No. 60775035, 60802066, 61103158), the National S&T Major Project (No. 2009ZX03004-001, 2010ZX03004-002-01), the National Basic Research Program of China (No. 2007CB311004) the Deakin CRGS 2011 and the Securing CyberSpaces Research Cluster of Deakin University.

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 270–282, 2012.

environments. *WSNs* also add value to various applications such as surveillance [1], smart home [2] and precision agriculture [3].

However, in order to fully exploit sensor networks for such applications, energy-efficient and scalable solutions for *WSN* services are essential. To address this issue, recent work in *Web Services* [4] [5] has started to utilize the service-oriented architecture (*SOA*) to support resource-constrained *WSN* services by proposing novel protocol stacks. Accordingly, *WSN services* refer to those *Web Services* built on the data gathered from wireless sensors.

One essential characteristic of *SOA* is that it can break a big service into small atomic services which could be distributed over a network and composed into new services. For *WSN* services, *SOA* can bring the benefits of modularity, flexibility, loose-coupling and interoperability. As the key issue of *SOA*, service composition has attracted considerable research interest. Among them, *workflow technology*, known for its practicability and efficiency, has been largely used for service automation and management in industry [6].

Recently, there has been a trend in utilizing *context information* such as environment information into workflows, which forms the research area named *context-aware workflows* [7] [8]. Unlike traditional workflows which only serve static processes, context-aware workflow is more flexible and suitable for dynamic processes on a dynamic scenario.

However, for *WSN* services, the flexibility of context-aware workflow is usually hindered by two problems:

- First, there are a large number of sensors in *WSN*, so we should construct and manage large amounts of atomic services. Also, context frequently updates will further require the workflows to be adjusted accordingly.
- Second but just as important, *WSN* has resources limitations on energy, memory space, computation capabilities and so on. Due to the first problem, it will be a heavy process to construct and dynamically change workflows.

Due to these challenges, traditional context-aware workflows are impractical for *WSN*. Their mechanism doesn't facilitate the atomic service reuse and sharing, and it also frequently make decisions on which concrete workflow should be used to realize the specific task according to current context. Therefore, a mechanism which can limit the number of atomic service and construct much more flexible workflows is indispensable.

If we consider the example of a *temperature control* service where a temperature value is used to determine the status of air conditioner. In different environments, even though service conditions (such as $T > 58℃$, or $T' > 100℃$) are different, they have the same semantic information, namely, *the temperature is high.* In other words, the low-level information (such as the specific temperature value) can be abstracted while the upper-level semantic information (such as $< temperature, is, high >$) can be preserved. If we take this semantic information as input and build a new atomic service in place of existing atomic services whose inputs are concrete temperature data, the number of atomic service will be limited. We depart the reasoning from atomic services through a reasoning model so that atomic services can be reused in many temperature control

workflows due to the fact that the newly built service is loose coupling with sensor data.

The rest of this paper is organized as follows. In Section 2, we present a simple review of related works. The Reasoning-based Context-aware Workflow (*Recow*) management approach is proposed in Section 3, followed by a case study which shows the effectiveness of the proposed approach in Section 4. Finally, Section 5 concludes this paper.

2 Related Work

Since the birth of ubiquitous computing [9] in the 1990s, researchers have attempted to integrate context-awareness into traditional workflows. A variety of context-augmented workflows have been proposed. Wieland et al. [8] proposed a three layered system model which consists of a smart workflow layer (*SWL*), a context provisioning layer (*CPL*) and a context integration layer (*CIL*). The approach proposed by them means that when contexts change, the on-going workflow will be terminated and substituted by a new one.

However, for an environment where the context is frequently getting updated, the efficiency will be critical. In order to address this issue, Ardissano et al. [10] proposed the abstract workflow for context-aware workflow execution (*CAWE*). They utilized the *context manager service* (*CtxMgr WS*) to provide the contextual information and *context-aware workflow manager* (*CA-WF-Mgr*) to execute an abstract workflow as if it were a standard workflow. By using *CAWE*, each abstract service is associated with multiple concrete implementations corresponding to different contexts. Contexts were used to determine the service implementation at run time.

Choi et al. [11] also proposed a framework, in which *context workflow scenario parser* (*CWparser*) represents contexts, while *DItree Handler* decides which subtree to apply. They used contexts as conditions to form a set of service subtrees. Upon the context changes, the workflow will be dynamically reconstructed by adjusting the subtrees. In addition, other context-aware workflow management systems [12] [13] only consider using abstract workflows to bind the business process (the workflow definition level) and the services (the instance level), while the concrete workflow process is determined during execution.

Although automatic workflow construction can be implemented according to the current context, a majority of existing work only considers dynamic service execution, while the resource reuse among services has been largely overlooked. Hence, in this paper, we focus on the facilitating of service reusing/sharing in the context-aware workflows design.

3 The *Recow* Model

From the example of *temperature control* service, new atomic services characterized by semantic information (*the temperature is high*) can be reused by numbers

of workflows. Therefore, through abstracting semantic information, it is possible to achieve resource reuse/sharing in *WSN* workflows. In user-centric services, process logics will frequently change. This means it is more suitable to adopt rule-based reasoning to abstract semantic information because only the rules need to be updated when the process logics is changing.

In this paper, we propose a *Reasoning-based Context-aware Workflow management* (*Recow*) approach, using the framework as shown in Fig. 1.

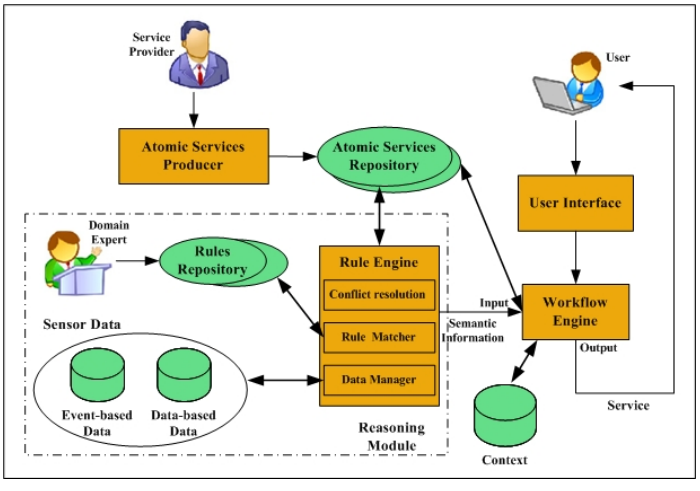

Fig. 1. *Recow* Framework

3.1 The *Recow* Model Overview

As shown in Fig. 1, the *Recow* model consists of three major modules:

- ***atomic services repository*** stores atomic services supplied by service providers for workflow composition.
- ***workflow engine*** which is responsible for building, monitoring and executing a workflow.
- ***reasoning module*** which is used to generate semantic information, as described inside the dotted box in Fig. 1. The captured *Sensor Data* will be used to select appropriate rules from the *Rules Repository* maintained by domain experts. The *Rules engine* is the core of the *reasoning module* and will be discussed further in Sec. 3.2.

The overall working process of *Recow* can be described as follows. Firstly, the service provider designs atomic services whose I/O are characterized with semantic information according to applications. Secondly, domain experts design the reasoning rules through a user-friendly interface. Thirdly, workflows will be

composed from atomic services which stored in the *atomic services repository* to implement working process. In real applications, the low-level sensor data will be imported into the *rule engine* to match designed rules as well as to extract the semantic information. This information is taken as inputs to trigger the workflows.

3.2 Reasoning Module

In *Recow*, we propose *reasoning module*. Firstly, it is possible. We can add the reasoning module between the upper-level business process and lower-level sensor data. Sensor data is initially loaded into the reasoning module, where we get corresponding semantic information. Secondly, it is necessary. Unlike traditional methods, *Recow* build new atomic services whose inputs are no longer low-level sensor data but semantic information. With these atomic services, workflows with improved flexibility and general ability can be composed.

In the reasoning module, the *rule engine* is the core of the reasoning approach, which originates in the expert system [14]. Different from other rule engines in context-aware applications, the rules in the *Recow* model are embedded with semantic information.

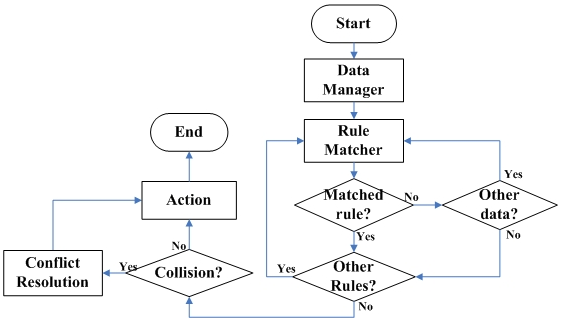

Fig. 2. Flow Chart of Rule-based Reasoning

As shown in Fig. 1, the *rule engine* consists of *data manager*, *rule matcher* and *conflict resolution*. *Data manager* determines which type of sensor data should be loaded into *rule matcher*; *rule matcher* handles the matching between sensor data and rules; *conflict resolution* determines which conflicting rule can be accepted. The rule-based reasoning process is described as a flowchart in Fig. 2. This process contains three components: extended rule representation with context-based priorities, sensor data management and rule matching, and conflict resolution.

Extended Rule Representation with Context-based Priorities. In real world applications, more than one rule could be matched to the same scenario.

Accordingly, a mechanism is needed to resolve the conflicts by deciding which rule will be activated.

The common approach to conflict resolution is based on the priority [15]. When more than one rule is matched at the same time, the rule with the highest priority will be activated. Traditional priority has only one static value. However, in *WSN*, an optimal rule should have different priorities due to different contexts. Accordingly in this paper, the priority part of each rule is extended to accommodate different contexts. Tab. 1 shows a sample rule which decides whether the temperature is comfortable or not.

Table 1. A Rule with Context-based Priority

rule name "temperature"
priority: 5(default),10(indoor),15(outdoor)
object: int temperature.value;
boolean temperature.comfortable;
if:
temperature.value>26(℃) **and**
temperature.value<30(℃);
then:
$temperature.comfortable \Leftarrow true$;
end

In this representation, each rule's *priority* on a specific context is determined by the matched priority. The *object* part of the rule specifies the variables involved in the rule.

In *Recow*, rules mainly come from domain experts. However, in real world, the rule extraction procedure is labor extensive. Therefore, a user-friendly graphical interface to simplify the construction of rules become necessary. For example, the outputs of rules corresponding with the inputs of atomic services, the *Recow* GUI provides drop-down menus for experts to select the sensor data types and semantic information for rules.

Sensor Data Management. Sensor data is usually updated frequently. In this part, we will discuss how sensor data will be controlled and loaded into the *rule engine*.

Rules in the *Recow* model will be constructed and registered into the *rule engine*. At the same time, the data sensed by sensors in *WSN* will be registered into the *data manager*, which is responsible for storing the mapping between the sensor data and the rule variable. For example, for a rule which checks the temperature, the rule variable should be mapped to the temperature data. Based on this mapping, the *data manager* can determine which sensor data will be loaded into the *rule engine*.

For each constructed rule, its rule variables are mapped with some sensor data, which can be roughly divided into two types: the *data-based sensor data* whose value will update frequently, such as the temperature data continuously

gathered by sensors; and the *event-driven sensor data*, whose value updates only when some specific event occurs, such as detection of a glass-break event. Corresponding to these two kinds of data, the *data manager* has two different treatments: the *data manager* will proactively obtain *data-based sensor data* in a fixed time interval, while the *event-driven sensor data* will not be captured until the event-based sensor detects a specific event occurring.

Rule Matching and Conflict Resolution. With respect to rule matching, many rule matching algorithms [16] have been proposed. Among them, *Rete*, an efficient forward inference rule matching algorithm, is one of the most popular matching algorithms.

The *Rete* algorithm starts with constructing a network of nodes, where each path from the root to leaf defines the *condition* part of a rule. When sensor data satisfies the condition of one node, it will propagate down to the next child node through the network until arriving at a leaf. Then the rules corresponding with the path propagated by the sensor data will be matched. Finally, the matched rules are added to the list of candidate firing rules.

In *WSN*, dynamic updating of sensor data will result in repeated rule matching, which costs more in terms of computing resources. We adopt the *Rete* algorithm [17] for its two characteristics. The first characteristic is state preservation, which can avoid duplicated computing on visited nodes. Secondly, data can pass through the network like a flow with a high processing speed. This makes it a suitable choice for large scale rules matching in *WSN* applications.

Nevertheless, conflicting rules remain a challenge when managing a large number of rules in the proposed *Recow* model. For example, let us assume that rule M is "*if air in a home is dry then turn on the humidifier*" and rule N is "*if there is nobody at home then turn off the humidifier*". When the context is "*no one at home and the air is dry*", both rules will be matched. If there is only one humidifier, conflict occurs.

In the *Recow*, we propose a conflict resolution algorithm based on context-based priorities. The pseudocode is provided in Alg. 1, in which *current.context* represents the current context, $Rule_i.action$ represents the *action* part of $Rule_i$ and *priority.context* represents the context with a specified priority in the extended rule.

The basic mechanism of our algorithm is that the rule with the highest priority will be activated. Due to the context-based priorities, the *priority* of each candidate rule will be checked against the current context, and the priority corresponding to the matched context is set as the priority of the rule. Finally, the rules with the highest matched context-aware priority will be activated.

4 Case Study

In this paper, we use two case studies to show the advantages of the *Recow* framework. In the first one, we will illustrate how the proposed *Recow* model

can facilitate the resource reuse and how context-based priority can be used to resolve conflicts. While in the second one, we will give statistical results to show the performance of *Recow*.

Algorithm 1. Context-aware Priority-based Conflict Resolution Algorithm

Require: Conflicted Rule Set $\{Rule_1, Rule_2, ...Rule_n\}$
Ensure: Output the action of selected rule
  $current.priority \Leftarrow 0$;
  $current.action \Leftarrow Rule_1.action$;
  $context\text{-}aware \Leftarrow false$;
  for $i = 1$ to n **do**
    read the rule priority
    if $\exists$ *a priority in* $Rule_i$
    ($priority.context == current.context$) and ($priority.value > current.priority$)
    then
      $current.priority \Leftarrow priority.value$;
      $current.action \Leftarrow Rule_i.action$;
      $context\text{-}aware \Leftarrow true$;
    end if
    if $context\text{-}aware == false$ **then**
      if $Rule_i.defaultpriority > current.priority$ **then**
        $current.priority \Leftarrow Rule_i.defaultpriority$;
        $current.action \Leftarrow Rule_i.action$;
      end if
    end if
  end for
  return *current.action*

4.1 Process Case Study

A husband (H) and his wife (W) have bought various types of sensors, including infrared sensors, sound detection sensors, etc., to build a smart home system for automatical intrusion detection. There will be various workflows to compose atomic services due to different combinations of sensors.

Let us suppose that a husband and wife build their own workflow in drag-drop way independently as shown in Fig. 3. Workflow H detects the infrared event when there is something around and the sound detection sensor checks whether sound intensity is higher than a threshold. Workflow W detects whether the sound volume is higher than a pre-determined threshold.

Traditional context-aware workflow management systems treat all workflows as concrete ones, from which the working workflow will be selected according to current context. While in the *Recow* model, we can construct a workflow C (as in Fig. 3), which contains a new "*intrusion detection*" atomic service whose input is characterized by semantic information, such as "*intrusion detected*" or "*intrusion not detected*". With the help of the *reasoning module*, semantic information can be extracted. By using the semantic information, such workflow

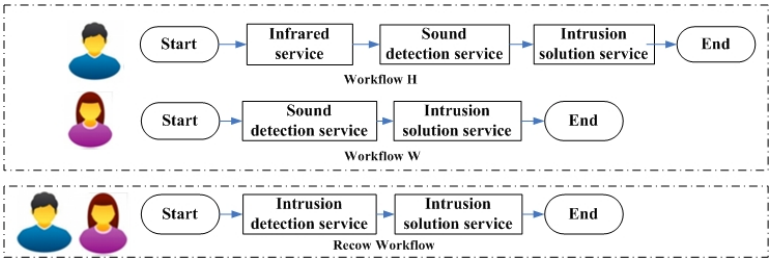

Fig. 3. *Recow* Workflow with Resource Reuse Compared with Original Workflows

can work as an abstract flow which can effectively load both the functions of workflow H and workflow W in the smart home.

In *Recow*, the *infrared service*, *sound detection service* and other similar atomic services can be substituted by the new *intrusion detect service*. Due to the new service's loose coupling with low-level sensor data, when the husband and his wife try to make their own workflow, the *intrusion detect service* can be reused and the same workflow C will be constructed even though different sensor combinations are in use. Therefore the *Recow* model reduces the number of atomic services needed, realizes the resource reuse and improves the reusability of workflow. Furthermore, with context changes, such as different sensor combinations for intrusion detection, workflow C can remain running without dynamically selecting the concrete workflow corresponding to the current context at execution time, which brings more flexibility than traditional context-aware workflows.

In order to get the upper-level semantic information, several rules should be designed initially(See Tab. 2). The default priority of rules depends on the rules' designer. Here, we suppose the wife's rules have higher default priorities than her husband's rules. Suppose the current context in the smart home is: "*infrared device detects some person is around and sound intensity is* 850". This context information will make rule H activated. Then *the reasoning module* can generate an output: "*intrusion happens*", which will be connected to the input of workflow C.

In addition to this, during Chinese New Year celebrations when crackers are released, and the environment is noisy, the system may give a false alarm. Suppose a rule E was constructed to resolve this problem, a new problem emerges: when an infrared sensor detects there is somebody around and sound intensity is 750, rule W and rule E will be matched at the same time but they will be in conflict(rule W outputs intrusion happens while rule E outputs intrusion does not happens). In traditional conflict resolution, only the default priority is used. So rule W will be activated because its default priority is higher, even though it might be a false alarm. In *Recow*, we propose a context-aware priority for conflict resolution. In rule E, the context-aware priority is at 10 during Chinese New Year celebrations, which also means the evening when sound intensity reaches 750, rule E will be activated because of its higher priority. Context-aware priority enables the smart home system to activate the most relevant rules corresponding to the current context.

Table 2. Intrusion Detection Rules

rule name "rule H" **priority**: 6(default) **object**: $(boolean)infrared.status \Leftarrow whethersome\ peopleis\ around\ or\ not$; $(int)(soundintensity) \Leftarrow theenvironment\ sound\ intensity$; **if**: $(infrared.status == true)$ and $((sound\ intensity) > 800)$; **then**: $intrusion.status \Leftarrow true$; **end**
rule name "rule W" **priority**: 7(default) **object**: $(int)(soundintensity) \Leftarrow theenvironment\ sound\ intensity$; **if**: $(sound\ intensity) > 700$; **then**: $intrusion.status \Leftarrow true$; **end**
rule name "rule E" **priority**: 5(default), 10(Chinese New Year) **object**: $(int)(soundintensity) \Leftarrow theenvironment\ sound\ intensity$; **if**: $(infrared.status == true)$ and $((sound\ intensity) < 900)$; **then**: $intrusion.status \Leftarrow false$; **end**

4.2 Performance Case Study

In this paper, a reasoning module is adopted to realize the resource reuse and this brings extra computation cost. In this case study, we will show the performance of the *Recow* approach with the reasoning module. Firstly, we suppose that more people make their rules or one people make more than one rule. So there will be not only above three rules in our intrusion service. We assume that the professional's rules are most suitable when possible conflicts exist. Then, we generate a large number of sensor data to match these rules. Fig. 4 shows the rule matching performance. X-coordinate indicates the number of sensor data used to match rules and Y-coordinate indicates the time consumed in rule matching algorithm.

Furthermore, we test the context-aware priority for the activation of the most suitable rule in *Recow*. We begin by giving several sensor data compositions as shown in Tab. 3. In Tab. 4, we will show the precision comparison of the traditional and context-aware priority used in the conflict resolution algorithm. In Tab. 4, the value "1" and "0" indicates whether the most suitable rule is

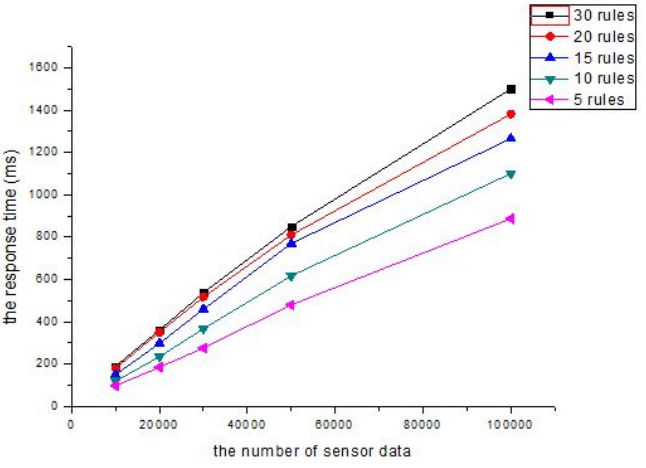

Fig. 4. The Performance of *Recow* Rule Matching

Table 3. The Context List

sensor data composition	description
*composition*1	$infrared.status == true, light == 930, sound\ intensity == 650$
*composition*2	$infrared.status == false, light == 1050, sound\ intensity == 850$
*composition*3	$infrared.status == true, light == 920, sound\ intensity == 750$
*composition*4	$infrared.status == true, light == 700, sound\ intensity == 670$
*composition*5	$infrared.status == true, light == 950, sound\ intensity == 780$

Table 4. The Precision Comparison of Traditional and Context-aware Priority Used in conflict resolution algorithm

	traditional	context-aware	current context
*composition*1	1	1	*Chinese New Year*
*composition*2	1	1	
*composition*3	0	1	*Chinese New Year*
*composition*4	1	1	
*composition*5	0	1	*Chinese New Year*
precision(%)	60%	100%	

activated or not. In this case study, we can see that the time efficiency of the reasoning module does not increase exponentially with the number of sensor data, and the precision of selecting the most suitable rules is also improved by using the proposed context-based priorities.

5 Conclusions

Workflow has been successfully used in wireless sensor networks (*WSNs*) for service composition. Recently, there has been a trend in utilizing context information into workflows. Now, research efforts on context-aware workflows have succeeded to realize the change in workflow according to their current context. However, in resource-limited *WSNs*, we also need to consider resource reuse.

In this paper to enable service resource reuse in context-aware workflows, we propose a reasoning-based context-aware workflow management (*Recow*) approach, in which the rule-based reasoning is exploited. We also present a corresponding resource conflict resolution algorithm. Compared with previous work, our main contributions can be summarized as follows: 1). Unlike traditional context-aware workflows, *Recow* can build new atomic services in place of existing ones. We characterize the new atomic service I/O with semantic information generated by a reasoning module, which effectively separates process logic and underlying sensor data. Hence, atomic services can be reused and furthermore, *Recow* can support context-aware workflow better. 2). We propose a context-based priority in the resource conflict resolution algorithm, through which we can select the most suitable rule for the current context.

The case studies presented in this paper demonstrate that *Recow* can effectively facilitate resource reuse and the extra computation time consumed in rule matching is acceptable. They also demonstrate the precision of the conflict resolution algorithm based on context-aware priority.

References

1. Wang, X., Wang, S., Bi, D.: Distributed visual-target-surveillance system in wireless sensor networks. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39(5), 1134–1146 (2009)
2. Suh, C., Ko, Y.B., C, C.: Collaborative workflow solution for distributed product development. In: 12th International Conference on Computer Supported Cooperative Work in Design, pp. 594–599. IEEE Press, Xi'an (2008)
3. de Lima, G.H.E.L., Neto, P.F.R.: WSN as a Tool for Supporting Agriculture in the Precision Irrigation. In: 6th International Conference on Networking and Services, pp. 137–142. IEEE Press, Cancun (2010)
4. Delicato, F.C., Pires, P.F., Pirmez, L., da Costa Carmo, L.F.R.: A Flexible Middleware System for Wireless Sensor Networks. In: Endler, M., Schmidt, D.C. (eds.) Middleware 2003. LNCS, vol. 2672, pp. 474–492. Springer, Heidelberg (2003)
5. Leguay, J., Lopez-Ramos, M., Jean-Marie, K., Conan, V.: An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks. In: 33rd IEEE Conference on Local Computer Networks, pp. 740–747. IEEE Press, Montreal (2008)
6. Tsai, M.J.: The workflow development for electronic manufacturing industry. In: 12th International Conference on Computer Supported Cooperative Work in Design, pp. 688–692. IEEE Press, Xi'an (2008)
7. Cho, Y., Choi, J., Choi, J.: A Context-Aware Workflow System for a Smart Home. In: 10th International Conference on Computer and Information Technology, pp. 95–100. IEEE Press, Dhaka (2007)

8. Wieland, M., Kaczmarczyk, P., Nicklas, D.: Context integration for smart workflows. In: 6th IEEE International Conference on Pervasive Computing and Communications, pp. 239–242. IEEE Press, Hong Kong (2008)
9. Weiser, M., C, C.: Ubiquitous Computing. Computer 26, 71–72 (2008)
10. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: A framework for the management of context-aware workflow systems. In: 3rd International Conference on Web Information Systems and Technologies, pp. 1–9. IEEE Press, Florian Lautenbacher (2007)
11. Choi, J., Cho, Y., Choi, J.: A Context-aware Workflow System Supporting a User's Dynamic Service Demands in Ubiquitous Environments. In: 1st International Conference on Multimedia and Ubiquitous Engineering, pp. 425–430. IEEE Press, Seoul (2007)
12. Chaari, T., Ejigu, D., Laforest, F., Scuturici, V.M.: A comprehensive approach to model and use context for adapting applications in pervasive environments. Journal of Systems and Software 80(12), 1973–1992 (2007)
13. Modafferi, S., Benatallah, B., Casati, F., Pernici, B.: A methodology for designing and managing context-aware workflows. Mobile Information Systems II, 91–106 (2005)
14. Jackson, P.: Introduction to expert systems. Addison-Wesley Longman Publishing Co., Inc., Boston (1998)
15. Thyagaraju, G.S., Math, M.M., Kulkarni, U.P., Yardi, A.R.: Conflict Resolving Algorithms to Resolve Conflict in Multi-user Context-Aware Environments. In: International Conference on Advance Computing Conference, pp. 202–208. IEEE Press, Patiala (2009)
16. Karp, R.M., Rabin, M.O.: Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development 31(2), 249–260 (2010)
17. Xiao, D., Zhong, X.: Improving Rete algorithm to enhance performance of rule engine systems. In: 2nd International Conference on Computer Design and Applications, pp. 572–575. IEEE Press, Qinhuangdao (2010)