Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 370–377

Procedia Computer Science

The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011)

CReMe: a Dynamic and Flexible Conflict Resolution Methodology for Competitive Ambient Systems[1]

Thais R. M. B. Silva[a,*], Linnyer B. Ruiz[b], Antonio Loureiro[c], Fabrício A. Silva[a]

[a] *Federal University of Viçosa – Florestal Campus – Florestal, Minas Gerais, Brazil*
[b] *State University of Maringá – Department of Informatics – Maringá, Paraná, Brazil*
[c] *Federal University of Minas Gerais - Department of Computer Sciente – Belo Horizonte, Minas Gerais, Brazil*

Abstract

Ambient Systems are available to their users all the time and everywhere. Most of them can be viewed as collective applications, i.e., shared by two or more users, which may lead to competition for resources and personal needs, generating conflicts of interests. The goal of this work is to present CReMe, a dynamic and flexible methodology to detect and solve conflicts for ambient systems. Two distinct case studies were evaluated to demonstrate CReMe's characteristics. The results showed that it was possible to accomplish a tradeoff between computational resources consumption and users' satisfaction.

Keywords: Ubiquitous Computing; Context-awareness; Collective Applications

1. Introduction

The substantial evolution of embedded hardware and software technologies observed in the last few decades has inspired the development of computational applications that are available to users all the time and everywhere [1]. These applications comprise what researches have been calling lately ambient systems. Their main endemic characteristics are ubiquity, pervasiveness, transparency, users, devices and environment resources volatility, and computational resources restriction. Ambient systems must be context-aware to be truly ubiquitous. Contexts are information about entities (i.e., people, places or objects) of interest for a given application. They can be used to adapt a system according to its users interests, allowing transparent and on-demand services provisioning and adequate resources consumption considering the current computational devices' capacity [2].

Collective ambient systems are composed of two or more users. They have to deal with contexts collected from all users and their computational devices, which are collective contexts, and from the shared physical environment. These information will be further used by the system to evaluate which service

[1]Thanks to the Foundation for Research Support of Minas Gerais (FAPEMIG).
[*]Corresponding author
Email addresses: `thais.braga@ufv.br` (Thais R. M. B. Silva), `linnyer@gmail.com` (Linnyer B. Ruiz), `loureiro@dcc.ufmg.br` (Antonio Loureiro), `fabricio.asilva@ufv.br` (Fabrício A. Silva)

should be made available to the users at each given time. Since users may present different profiles and/or the environment resources shared by them may be scarce, they may appear to the system as competitive peers. The ambient system can, therefore, reach an inconsistent state, also called collective conflict, since it cannot answer the individual and collective demands at the same time.

Users depend on the resolution of collective conflicts to proceed performing the system's services. However, this is not a trivial task. Whenever a user feels neglected, he/she starts to perceive the system's presence, which hinders its ubiquity aspect. Due to the limited resources availability, the computational devices used may not be able to execute sophisticated algorithms to solve the identified collective conflicts. Therefore, a clear tradeoff is established between the quality of the resolution service and its related resources consumption. Furthermore, the specific reasons why users became competitive may diverge from one system to another. This occurs due to the specific nature of each ambient system, as well as their volatility. The main goal of this work is to present the details of a novel collective conflicts resolution methodology called CReMe (*Conflict Resolution Methodology*). To validate the CReMe methodology, two distinct ambient systems were developed and evaluated.

The rest of this work is organized as follows: Section 2 introduces the related work found in literature. Section 3 describes the concrete concepts and models of the CReMe methodology. Two distinct competitive systems that use the CReMe methodology are presented in Sections 4 and 5, as well as results regarding their conflicts resolution performance. Finally, Section 6 shows the conclusions and some future work.

2. Related Work

Roy et al. [3] propose a framework to capture the correlation and interaction of movements for different inhabitants of a smart house. The work is focused on routes prediction and localization as a support for an eventual treatment for users divergence. Shin and Woo [4] propose a solution to solve users and applications collective conflicts for competitive smart houses. The former type of conflict occurs when users try to adapt the environment resources in different ways, whereas the latter indicates competition among applications that want to share these resources. Shin et al. [5] state that conflict situations should be solved using automatic and manual schemes, depending on the competition's nature. They proposed a solution that solves conflicts automatically whenever an intermediary value can be calculated and manually otherwise. Wang and Turner [6] describe a system prototype that uses policy based management for home care services. The authors have investigated the detection and resolution aspects among policies competition, which might occur whenever they are analysed and performed by the system. A sorted list of preferences is used to solve the conflicts. Capra et al. [7] introduce a middleware that uses micro-economy techniques to solve conflicts among different context-aware ambient systems. Applications are seen as competitors and the middleware tries to satisfy them all. The study performed by Rarau and Pusztai [8] defines that ambient systems have multiple faces, each of them related only to some system's functionalities activated on certain conditions. Conflicts, therefore, occur when users try to activate different faces at the same time. The author's resolution strategy is based on the use of priorities and requests lines.

3. CReMe: A Conflict Resolution Methodology

As opposed to the studies found in literature (see Section 2), this work presents a flexible and dynamic methodology called CReMe. It comprises a set of concepts and models, which can be instantiated to create concrete implementations of conflicts resolution schemes suitable for a specific competitive ambient system. Furthermore, this methodology expects changes on the systems' behavior throughout their lifetime, allowing adaptations on the current tradeoff between quality of the provided service (i.e., users' satisfaction towards the conflicts resolution results) and the related amount of computational resources needed to perform them.

In the work of Silva et al. [9] a preliminary proposal of the CReMe methodology design was presented. The key aspects of that work were to describe competitive ambient systems, their collective aspects, the conflicts that they may face due to users' profiles divergence and/or environment resources availability and the need of a resolution strategy that could be used by many different scenarios, according to the current needs of their users. The study also described a structural framework for the methodology, called *Conflict Engine*, as well as an abstract proposition to its operation, based on an algorithms repository, and a conceptual definition of the configuration file used to provide system's information to the methodology. The *Conflict Engine* framework organizes the methodology operations into modules, defining their interfaces and relationships. Four modules were proposed: Conflicts Detection, Conciliation, Support and Tasks Release. An abstract implementation to the Conciliation module was defined. It proposes the use of an algorithms set and some information provided by each ambient system to identify which would be the best way possible to solve the current conflicts. The information must be provided through a file called *Action Level*, which encompasses indications on how to detect conflicts, solve them using an algorithms repository and modify CReMe's behavior considering the intended tradeoff between satisfaction and resources consumption.

In order to establish the complete CReMe methodology approach and determine concrete elements that can be directly used by any context-aware competitive ambient system, new models and concepts were developed and are presented in the current work. Initially, two models were defined: application and architecture. The former determines that each ambient system must be composed of a set of well-defined services or tasks. The time interval that comprises the indication, selection and execution of a task by users is called a round. The latter defines a client-server organization, with servers rotation. The server is responsible for running all the *Conflict Engine* modules. The clients send messages to the server indicating which task they intend to perform at a given round. The server aggregates all indications, analyse them considering their competition, identify and solve potential conflicts, selects which tasks each user will perform and publish the results. The servers rotation scheme occurs at the beginning of each round. It considers that the system is composed of heterogeneous computational devices. Hence, only the most powerful elements are preselected as candidates. Before the round begins, one of the candidates that has not performed the server role yet is randomly selected. When all the candidates were already selected, this process is restarted.

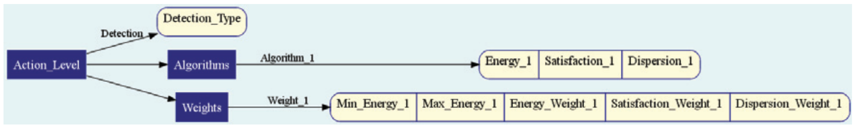


Fig. 1. *Action Level* XML schema.

The next step is to determine how the algorithms repository and the *Action Level* should be designed. These two elements comprise key aspects to the implementation of the flexibility and dynamic nature of the CReMe methodology. To build the repository, the ambient system designer must provide resolution algorithms that implement a predefined interface. Considering the *Action Level*, it has to be developed individually to each system as an XML (eXtensible Markup Language) file. A pre-defined XML schema has to be used by the systems' designers (Figure 1). First, it has a field that indicates a conflicts detection implementation. The ambient system designer has, therefore, to provide this implementation, which must complies to the Conflict Detection module interface, and then indicate its complete name and address to the corresponding *Action Level* field. Then, the file has an item that represents the list of resolution algorithms that could be used by the CReMe instance. The list may contain one or more algorithms, each of them represented by a specific XML element. This element is composed of the full algorithm name and address and some of its meta-data, which are information about the algorithm execution performance. Finally, the last part of the *Action Level* contains a list of weights, which are elements that determine the tradeoff between resources consumption and satisfaction intended by the system, at some energy availability interval.

At this point, all the information needed for the methodology to select the resolution algorithm is available. One question remain unanswered though: how to use these information to make such a decision? CReMe uses the Euclidean Distance Algorithm. Each algorithm's meta-data set is modeled as a point on a three-dimensional space. The weight element that corresponds to the current servers' energy availability is also modeled this way. The Conciliation module uses theses points as input to compute the distance between each algorithm and the current weight element. The algorithm that leads to the smallest distance value is selected, since it is the one that matches in the best way possible the system's current intended tradeoff between satisfaction and consumption. The Conciliation module can use one last resource, called *adjustable parameters*, to fine-tune the algorithm according to the system's demands. Some algorithms may present parameters that can be dynamically modified to allow less computational resources consumption or to improve users' satisfaction. Whenever this is the case, these parameters can be described through the *Action Level*, which indicates how to modify them and what are the related consequences.

4. A Collective Tourist Guide System

System Description. Computational tourist guides are supposed to obtain information on user's desires and characteristics, as well as from his/her current physical environment, and use them to indicate which tourist task should be performed. So far, the proposals found in literature consider only individual systems. However, it is highly habitual to have groups of people engaging on tourist rides together. In this case, the system may became competitive whenever the users' profiles diverge, which results on users indicating different tasks at a given time. In this work, a collective tourist guide ambient system was developed as a case study for the CReMe methodology. It was considered that each tourist task is performed by the entire group per system round. The system ends when all the tasks have already been performed by the group.

CReMe Instance. An instance of CReMe was developed to the tourist guide system. This was accomplished by the implementation of an algorithms repository and an *Action Level* file (Figure 2). Three algorithms were implemented: Random (one task is randomly selected), Majority (the most indicated task is selected) and Priorities (the most indicated task by high priority users is selected). They diverge on the resources consumption scale and users' satisfaction (collective and individual) level (see Figure 2). Notice that a conflict detection implementation called *Tasks* was also developed and indicated. It verifies whether the tourists indications are all identical or not. Finally, three weight elements were defined. They indicate that the system intends to begin saving energy only when the average residual value is below 60%.

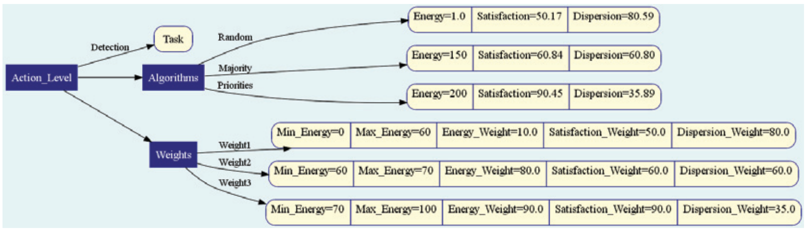
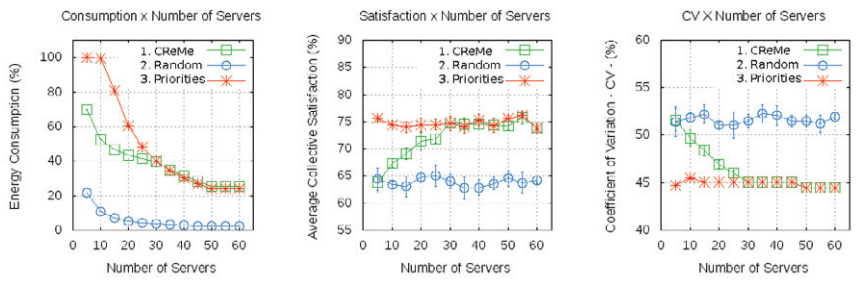

Action_Level
Detection
Task
Algorithms
Random
Majority
Priorities
Energy=1.0 | Satisfaction=50.17 | Dispersion=80.59
Energy=150 | Satisfaction=60.84 | Dispersion=60.80
Energy=200 | Satisfaction=90.45 | Dispersion=35.89
Weights
Weight1
Weight2
Weight3
Min_Energy=0 | Max_Energy=60 | Energy_Weight=10.0 | Satisfaction_Weight=50.0 | Dispersion_Weight=80.0
Min_Energy=60 | Max_Energy=70 | Energy_Weight=80.0 | Satisfaction_Weight=60.0 | Dispersion_Weight=60.0
Min_Energy=70 | Max_Energy=100 | Energy_Weight=90.0 | Satisfaction_Weight=90.0 | Dispersion_Weight=35.0

Fig. 2. Collective Tourist Guide System *Action Level*.

Simulation Parameters. The collective tourist guide system and its associated CReMe methodology instance were implemented through simulation to be properly evaluated. The Siafu simulator tool [10] was used due to its robustness, flexibility and adherence to the context-awareness field. Siafu is a free and open-source simulator (GNU GPL license) written in the Java programming language. It supports the development of multi-users context-aware applications, which was required by this work.

In the simulations, each tourist uses an *HTC* mobile device [11] that collects contextual data. Users indicate their preferred tasks at each round through their devices. Priorities ranging from 1 (higher) to 3 (lower) were randomly distributed among users. Besides, a certain users' percentage, called homogeneity level, indicates a subgroup of people that presents very similar profiles and, therefore, always indicate the same preferred task at each system round. Three scenarios were developed to the evaluation process: scenario 1 uses the CReMe methodology instance to solve collective conflicts; scenario 2 uses only the Random algorithm; scenario 3 uses only the Priorities algorithms. The goal is to compare the approach proposed by CReMe with other typical scenarios that use always the same algorithm. In particular, 5 simulations were performed per conducted test, each one using 80 tourists, a set of 50 tourist tasks and device's battery initial capacity set to 0.01 Joules.

Evaluation Parameters. The items evaluated per performed test were the energy consumption with processing, the collective users' satisfaction (calculated through a weighted average) and the individual users' satisfaction dispersion (calculated as the coefficient of variation over individual satisfaction values). An individual satisfaction value (from 0% to 100%) is attributed to each user at the end of a round, according to both indicated and selected tasks. At the end, the average value of those parameters are calculated.

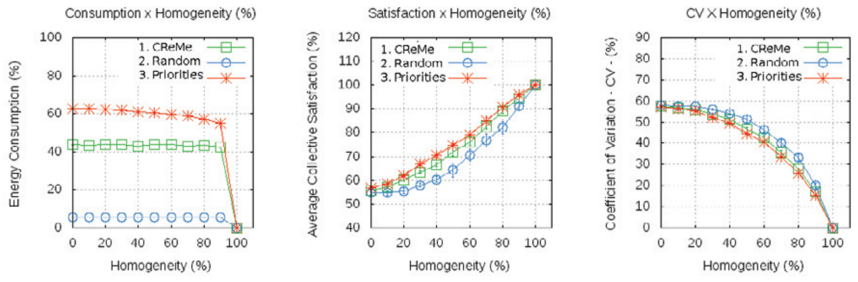


(a) Energy (b) Collective Satisfaction (c) Individual Satisfaction

Fig. 3. Number of Servers Evaluation - Collective Tourist Guide System

Results. The first test evaluated the number of server candidates (N) (Figure 3). The smaller the number of servers compared to the number of rounds with conflicts, the higher the number of times they will be activated. When the number of servers are equal to or greater than the number of conflicts, the average energy consumption stabilizes, once each server will perform the resolution a smaller amount of times. Therefore, for all scenarios, the energy consumption decreases while the number of servers increases. It is worth to highlight that scenario 1 already includes an extra energy consumption with the CReMe methodology algorithms selection process. Observing Figure 3a, it is verified that for N greater than or equal to 30, scenarios 1 and 3 present nearly identical energy consumption, since scenario 1 begins to use only the Majority or Priorities algorithms, which are similar. Scenarios 2 and 3 did not suffer any alteration regarding users' satisfaction. This is due to the fact that they always use the same algorithm. However, the decrease of energy consumption allowed an increase on collective satisfaction and decrease on the individual one for scenario 1 (Figures 3b and 3c). As stated earlier, with more energy available, the scenario could use more often the algorithms that provide higher satisfaction levels.

The value of N used for the other simulations was selected from the analysis of this test. When the collective tourist system uses 20 servers, it can experience a smooth degradation of their battery capacity, which simulates the most common occurrence for ambient systems, i.e., computational resources volatility. In this case, the CReMe methodology can be fully used, varying the resolution algorithms according to the tradeoff between consumption and satisfaction as stated by the system designer.



(a) Energy (b) Collective Satisfaction (c) Individual Satisfaction

Fig. 4. Homogeneity Level Evaluation - Collective Tourist Guide System

Another test evaluated the homogeneity level. The goal was to verify which impact this percentage value has on the evaluated parameters. Figure 4 depicts the related graphics. It shows that the homogeneity level almost does not has any impact on the servers energy consumption for all scenarios. The number of instructions processed by the scenarios does not suffer substantial changes with the homogeneity level increase. Considering the collective and individual satisfaction though, all scenarios presented an increase and decrease, respectively, while the homogeneity level increases. The higher the number of users indicating the same task, the higher the average satisfaction level among them. Scenario 1, as expected, presented the intermediary behavior configured in its *Action Level* file.

Considering the results obtained for the number of server candidates and the homogeneity level, new simulations were conducted and average values for energy consumption with processing, collective satisfaction and individual satisfaction were calculated. The system was configured with 20 servers and 50% for the homogeneity level. An intermediary pattern could be clearly seen for scenario 1. The energy consumption of this scenario was 43.58%, against 60.47% of scenario 3. Besides, its collective satisfaction average was 70.21%, while scenario 2 presented a 63.32% value. The individual satisfaction value obtained was very similar to the scenario 3 one, and both were around 5% smaller than the result for scenario 2.

5. A Large Scale Emergency Attendance System

System Description. An emergency attendance system helps medical rescue teams to identify and assist victims of a major disaster in an efficient and organized way. When the disasters have large proportions, victims will normally compete for rescue resources, such as doctors, nurses, ambulances, hospital beds, and so on. In this case, a context-aware competitive ambient system could identify and solve the collective conflicts that occur due to environment resources scarceness. A large scale emergency attendance system was developed as a second case study for the CReMe methodology. This kind of system was selected due to its clear differences to the collective tourist guide application. The goal is to show that it is possible to create a CReMe instance to another competitive ambient system in the same way it was done for the tourist application case, which demonstrates the CReMe's flexibility.

Particularly, we have considered the following application model for the emergency system: users, which are the disaster victims, have to be assisted throughout a three-step procedure. The first step is the local first-aid, the second is ambulance allocation and, finally, the third is the hospital admittance. The system provides a set of tasks related to each of these steps. At each round, users indicate one of the tasks related to their current attendance step. Whenever a user performs a task on the current step, he/she is automatically moved to the next one. The execution of a third-step task ends a victim's emergency attendance. The application as a whole ends when all users are completely assisted. Each task has an environment resource with limited capacity. Collective conflicts may be detected whenever users indications at a given round surpasses the environment resources capacities. Each user has a priority value assigned by the system, which is also a value from 1 to 3. In case of conflicts, the unattended users have to indicate the same task again during the next system round. Notice how different the emergency system is from the tourist one. Nevertheless, it is possible to build CReMe methodology instances for both of them.

CReMe Instance. The emergency system CReMe instance is also composed of an algorithms repository and an *Action Level* file. The algorithms implemented in this case were Sequential (the first indicated tasks are selected), Sequential-Priorities (uses the Sequential and Priorities algorithms interchangeably) and Priorities (the tasks indicated by high priority users are selected). They were also developed based on a predefined interface. Figure 5 depicts the system's *Action Level*. A conflicts detection algorithm, called *Demand*, which compares users' indications and environment resources capacities was implemented and used. Further, the three developed algorithms for the repository along with their meta-data were described. Finally, three weigh elements reflecting the desired tradeoff between resources and satisfaction were assigned. The same values from the tourist system were used due to comparison reasons. The Sequential-Priorities algorithm has one adjustable parameter that defines how many times each implementation will be used.

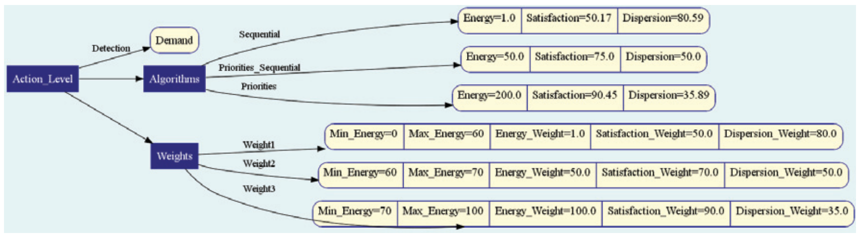
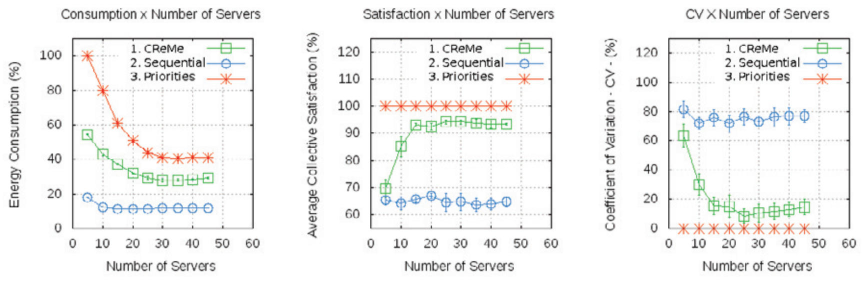

Action_Level
Detection
Demand
Algorithms
Sequential
Priorities_Sequential
Priorities
Energy=1.0 | Satisfaction=50.17 | Dispersion=80.59
Energy=50.0 | Satisfaction=75.0 | Dispersion=50.0
Energy=200.0 | Satisfaction=90.45 | Dispersion=35.89
Weights
Weight1
Weight2
Weight3
Min_Energy=0 | Max_Energy=60 | Energy_Weight=1.0 | Satisfaction_Weight=50.0 | Dispersion_Weight=80.0
Min_Energy=60 | Max_Energy=70 | Energy_Weight=50.0 | Satisfaction_Weight=70.0 | Dispersion_Weight=50.0
Min_Energy=70 | Max_Energy=100 | Energy_Weight=100.0 | Satisfaction_Weight=90.0 | Dispersion_Weight=35.0

Fig. 5. Emergency Attendance System *Action Level*.

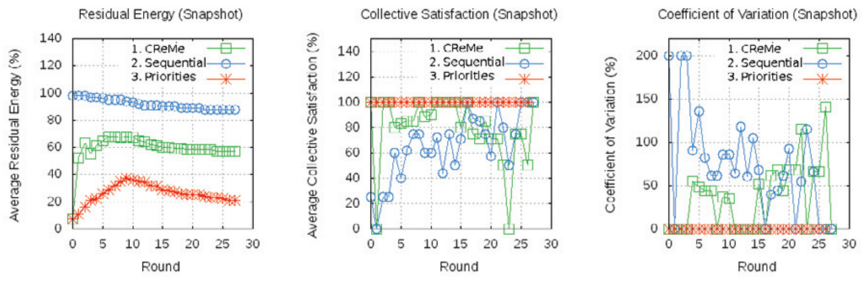
Simulation and Evaluation Parameters. An implementation of the emergency system, as well as its CReMe methodology instance, were also developed for the Siafu simulator. The same three scenarios were used. In this case, however, scenario 2 uses the Sequential algorithm and scenario 3 the Priorities one. The same evaluation parameters were also considered. The collective satisfaction computation, though, is slightly different: after the weighted average is calculated, it is compared to the maximum and minimum possible values, generating, this way, a relative result. The number of tasks and the initial devices' battery capacity were 15 (5 for each attendance step) and 0.02 Joules, respectively.



(a) Energy (b) Collective Satisfaction (c) Individual Satisfaction

Fig. 6. Number of Servers Evaluation - Emergency Attendance System

Results. The number of servers candidates (*N*) were also evaluated for the emergency attendance system (Figure 6). The same observations obtained for the collective tourist guide holds true. One particular difference can be observed in Figure 6a, in which the energy consumption for scenarios 1 and 3 are not alike as it was the case for the tourist system. This is due to the fact that scenario 1 uses the Sequential and Sequential-Priorities algorithms, which are considerable different from the Priorities one, used by scenario 3, in all aspects. The value of *N* selected to be used for other emergency system simulations was 10. The reason is the same as stated earlier for the collective tourist case.



(a) Energy (b) Collective Satisfaction (c) Individual Satisfaction

Fig. 7. Snapshots - Emergency Attendance System

Using 10 server candidates, a snapshot analysis was performed for the emergency attendance system (Figure 7). The main goal of this test is to evaluate the behavior of the system throughout its lifetime. The graphics clearly show the adaptation of scenario 1, considering the three weight elements configured in the *Action Level* file, as well as the corresponding three algorithms used. Conflicts tend to occur more frequently during the initial rounds. Considering the residual energy, until the tenth round the values increase for scenarios 1 and 3 because the number of devices acting as servers also increases until this round, but the amount and intensity of conflicts decrease. From the eleventh round on, the server candidates start to be repeated and then the residual energy decreases. On the last rounds the consumption remains constant since the number of conflicts is low and, sometimes, do not even occur. During the first half of the rounds, the results for scenario 1 (energy consumption, collective and individual satisfaction) are very similar to scenario 3, once it uses the Sequential-Priorities and Priorities algorithm interchangeably. However, during the second rounds half, scenario 1 begins using the Sequential algorithm, which causes its satisfaction levels to be similar to those presented by scenario 2. Simulations were also conducted for the emergency system to acquire average values for the evaluated parameters. The results express the same intermediary pattern for the scenario 1 behavior as obtained for the collective tourist guide system. In particular, the energy consumption of scenario 1 was 36% less than the value for scenario 3, considering the extra processing with the algorithms selection process of the former. Considering the collective and individual satisfactions, the values for scenario 1 were, respectively, 20% higher and 44% smaller than those for scenario 2.

6. Conclusion

This work presented in details a novel methodology, called CReMe, to detect and solve conflicts of interest that may occur among users of a competitive ambient system. So far, the literature has few studies on the subject, which are all based on specific and limited approaches. The CReMe methodology appears as a flexible and dynamic solution, composed of concrete models and concepts that can be directly instantiated to many different ambient systems. In order to demonstrate the methodology principles, two distinct competitive ambient systems were developed. Results showed that both of them could control the tradeoff between computational resources consumption and users' satisfaction towards the conflicts resolution as desired. Some aspects of the CReMe methodology can be further analysed as future work: implementation of different applications with particular characteristics such as smart homes and vehicles; Development of new resolution algorithms for the repository; Use of a decentralized architecture model and evaluation of alternative servers rotation scheme for the client-server one.

References

[1] J. Krumm, Ubiquitous Computing Fundamentals, Chapman & Hall/CRC, 2009.
[2] A. K. Dey, Understanding and using context, Personal Ubiquitous Computing 5 (1) (2001) 4–7, issn:1617-4909.
[3] N. Roy, A. Roy, S. K. Das, Context-aware resource management in multi-inhabitant smart homes: A nash h-learning based approach, in: PERCOM '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society, Washington, DC, USA, 2006, pp. 148–158, isbn:0-7695-2518-0.
[4] C. Shin, W. Woo, Conflict resolution method utilizing context history for context-aware applications, in: Proceedings of the 1st International Workshop on Exploiting Context Histories in Smart Environments (ECHISE'05), Munich, Germany, 2005.
[5] C. Shin, A. K. Dey, W. Woo, Mixed-initiative conflict resolution for context-aware applications, in: UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing, ACM, New York, NY, USA, 2008, pp. 262–271, isbn:978-1-60558-136-1.
[6] F. Wang, K. J. Turner, Policy conflicts in home care systems, in: Proc. 9th Int. Conf. on Feature Interactions in Software and Communications Systems, IOS Press, Amsterdam, Netherlands, 2008, pp. 54–65.
[7] L. Capra, W. Emmerich, C. Mascolo, Carisma: context-aware reflective middleware system for mobile applications, IEEE Transactions on Software Engineering 29 (10) (2003) 929–945.
[8] I. S. A.Rarau, K.Pusztai, Software framework for building context aware applications using multifacet items, in: Proceedings of the 2nd International Workshop on Software Aspects of Context(IWSAC'05), Santorini, Greece, 2005.
[9] T. R. M. B. Silva, L. B. Ruiz, A. A. Loureiro, Towards a conflict resolution approach for collective ubiquitous context-aware systems, in: The 12th International Conference on Information Integration and Web-based Applications and Services (iiWAS'10), Paris, France, 2010, pp. 433–440.
[10] M. Martin, P. Nurmi, A generic large scale simulator for ubiquitous computing, in: Annual International Conference on Mobile and Ubiquitous Systems, IEEE Computer Society, Los Alamitos, CA, USA, 2006, pp. 1–3, isbn:0-7803-9791-6.
[11] Google Nexus One Device, http://www.htc.com/www/product/nexusone/overview.html (May 2011).