Preventing Attacks by Classifying User Models in a Collaborative Scenario*

César Andrés, Alberto Núñez, and Manuel Núñez

Universidad Complutense de Madrid, E28040 Madrid, Spain
c.andres@fdi.ucm.es, alberto.nunez@pdi.ucm.es, mn@sip.ucm.es

Abstract. There are several methods to assess the capability of a organization to prevent attacks in a potentially wrong collaborative scenario.

In this paper we explore a methodology based on considering some probabilistic information. We assume that we are provided with a *probabilistic user model*. This is a model denoting the probability that the entity interacting with the system takes each available choice.

We show how to build these models using the log files. Moreover, we define the meaning of a good, a bad and a suspicious behavior. Finally, we present a mechanism to share the information presented in each node of the collaborative system.

1 Introduction

Among those areas where the development of Computer Science has changed our society during the last years, the relevance of the collaboration among different information systems is remarkable [9,4,11,16]. There is a strong demand for access control of distributed shared resources in nets of *collaborative organizations* [8,13], where the classical notion of client server architecture is obsolete and useless. A net is composed of several organizations sharing services, employees, and resources among them. Therefore cross-organizational interoperability is a major challenge to nets of virtual organizations [5]. To be able to specify not only the functional aspect of a net but also those aspects that guarantee the interoperability policies is an industrial necessity [12,6].

Currently a perspective that has received little attention is the establishment of a *sound methodology* to determine the correctness of a net with respect to all the interoperability policies that are defined on it. Formal methods [14] provides us a compound of mathematical techniques that allows the automated design, specification, development and verification of software systems.

One of the advantages of using a formal representation of systems is that it allows to rigorously analyze their properties. In particular, it helps to establish the *correctness* of the system with respect to the specification or the fulfillment of a specific set of requirements, to check the semantic *equivalence* of two systems, to analyze the *preference* of a system to another one with respect to a given

* Research partially supported by the Spanish MCYT project TESIS (TIN2009-14312-C02-01).

criterion, to predict the possibility of *incorrect behaviors*, to establish the *performance* level of a system, etc. By using these techniques, the search of critical cases becomes a more systematic task, as it depends less on the pure intuition of programmers.

In a collaborative scenario sometimes we can consider that we are provided with some information that does not concern properties of the environment itself to increase the power of error detections of these methodologies [3]. For instance, let us assume that we are provided with a probabilistic model of the *user* that will interact with the system. This model defines the probability that the external environment (e.g., a human user, another system, a medium, etc) takes each available choice at each time. Let A be a finite set of *ways to interact* with the system and let us suppose that, according to the user model, the probability that the user interacts with the system according to any behavior belonging to A is p.

The main goals of this paper are: a) to define the user models in a collaborative scenario; b) to show how to build these models by using the log files; c) to provide a methodology to classify the interactions of real users with these virtual models in the sets of *good*, *bad* and *suspicious*. Moreover, taking into account that our systems are in a collaborative scenario, an additional contribution is to show how the information collected in different user models can be shared among the net of systems in order to use this knowledge to prevent attacks.

The rest of the paper is structure as follows. In Section 2 we present how to create the user models. In Section 3 we show how to spread the knowledge among the different nodes. Finally, in Section 4 we present our conclusions and some lines of future work.

2 User Models

Data mining is becoming an increasingly important tool to transform data into information. This technique is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection, and scientific discovery. Roughly speaking, data mining is the process of extracting hidden patterns from data sets. In Figure 1 we represent the process that we use in order to perform the extraction of the most relevant information related to the interaction of users with the system. It follows the scheme presented in [15]. First, we consider that we are provided with a set of data recorded from the interaction between different users and the system. Let us note that a bigger number of records in the database reflects that there exist more *kinds* of users represented in it. Next, a selection of data is made and preprocessed in order to check that there does not exist any incongruence, that is, the set of data represents traces of users that have been observed during their interaction with the system. We will focus on extracting sets of *relevant behaviors* from this set of data, that is, those sequences of interactions of the users with the system that appear more frequently. In order to determine these behaviors we use the usual techniques for obtaining frequent patterns from a database (see, for example, [1]). The task of discovering the frequency of each behavior in the database is performed by means of the

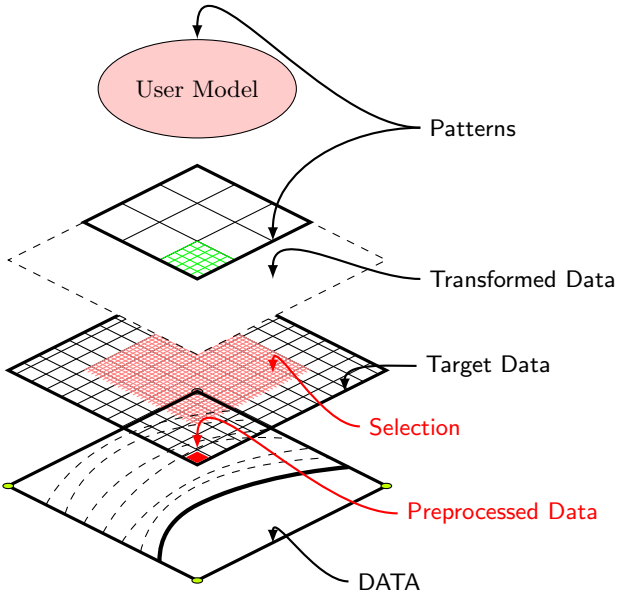

Fig. 1. The data mining process

applications of any of the existing tools. In our approach we use [7]. Once we have collected the information corresponding to the usual behaviors and their frequency, we can *convert* this information into a user model that we can use in our algorithm (that will be described later).

Next, let us describe the extraction of a probabilistic model to represent how users decide to interact with the system, using the processed data that we have obtained with the data mining process. Intuitively, a user model includes the probability that a user chooses each input in each situation. We use a particular case of *Probabilistic Machine* to represent user models, interested readers can be found the formal details in [2] . Let us note that the interaction of a user with a system ends whenever the user decides to stop it. Consequently, models denoting users must represent this event as well. Therefore, given a probabilistic machine representing a user model, we will require that the sum of the probabilities of all inputs associated to a state is lower than or equal to 1. The remainder up to 1 represents the probability of stopping the interaction at this state. Due to space limitations, we simply describe the intuitive idea to deal with the construction of the model.

Let us consider that we have three different interactions of the users with our organization. These are $\langle$`AccessFile`, 10 : 20 : 20123005$\rangle$, $\langle$`AccessFile`, 10 : 47 : 20123005/`UpdateFile`, 10 : 52 : 20123005$\rangle$ and $\langle$`ModifyFile`, 10 : 31 : 20123005$\rangle$.

Each iteration is a sequence of pairs action/time stamp. For instance the first one represents a user that performs `AccessFile` in a shared resource at 10 :

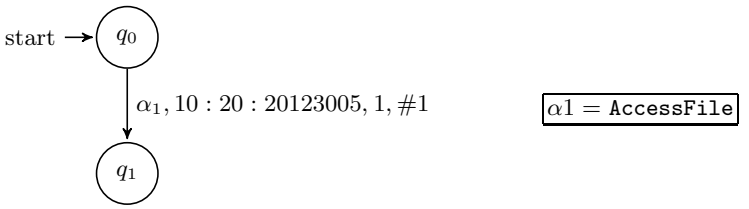

Fig. 2. Creation of user model (1/3)

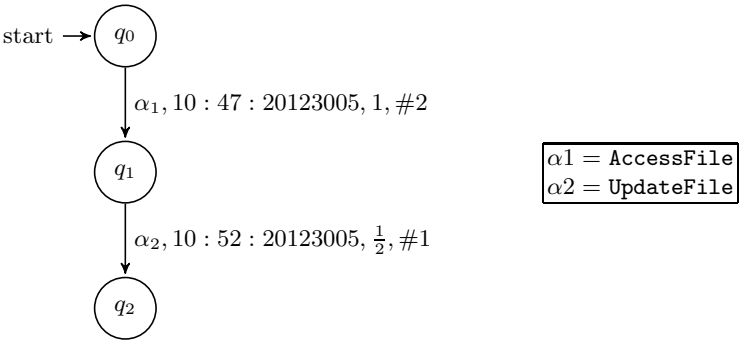

Fig. 3. Creation of user model (2/3)

20 : 20123005 (hour:minute-year-day-month). The user model that represents this interaction is presented in Figure 2.

The initial state of this model (M) is q_0, and it represents the probability of executing each interaction by the users. The transitions of the Figure 2 follow this pattern: action/time-stamp/probability/number-of-behaviors. For instance the transition $q_0 \xrightarrow{\texttt{AccessFile},10:20:20123005,1,\#1} q_1$ represents that any user that interacts with the organizations, with probability 1 (always) performs the action `AccessFile`. The time stamp is included in order to update the model in the future. The last parameter, by means #1 represents the number of behaviors that the transition represents. Finally, the meaning of not having outgoing transitions from q_1 means that the user stops at q_1. Now, let us consider the next behavior, that is $\langle \texttt{AccessFile}, 10:47:20123005/\texttt{UpdateFile}, 10:52:20123005\rangle$. After including this behavior in the previous model we obtain the model presented in Figure 3.

Let us remark the main differences of this automaton with respect to the previous one. First of all, the transition $q_0 \xrightarrow{\texttt{AccessFile},10:47:20123005,1,\#2} q_1$ has upgraded the time stamp and the number of behaviors that it represents. That is, we have collected the last time when this action was performed into the system. Next, a new transition has been created $q_1 \xrightarrow{\texttt{UpdateFile},10:52:20123005,\frac{1}{2},\#1} q_2$. Let us note that the probability of executing this transition is 0.5. The remainder to 1 means that with probability $1 - 0.5$ the user stops at state q_2. So, we do not loose the first behavior, and we are able to add a new behavior. Next, we

upgrade the previous behavior with the log $\langle$`ModifyFile`$, 10:31:20123005\rangle$. The final version of the user model is presented in Figure 4.

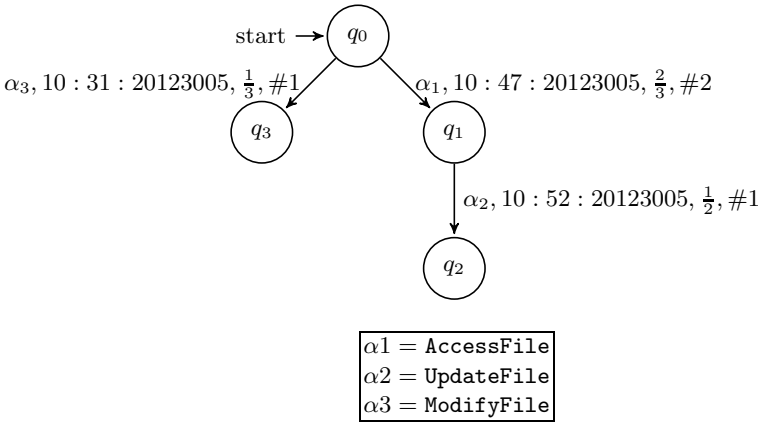

Fig. 4. Creation of user model (3/3)

There are two relevant parts in this automaton. On the one hand, a new transition $q_0 \xrightarrow{\texttt{ModifyFile},10:31:20123005,\frac{1}{3},\#1} q_3$ has been created. On the other hand, all the transitions outgoing from q_0 (in this case there is only one): $q_0 \xrightarrow{\texttt{AccessFile},10:52:20123005,\frac{2}{3},\#1} q_1$ have updated their associated probability value. Let us note that the new automaton behaves as follows. It starts at state q_0, and defines that all the users of this model perform with probability $\frac{1}{3}$ the action `ModifyFile`, and with probability $\frac{2}{3}$ the action `AccessFile`.

In addition, the previous process is incremental, and we consider that each organization during a period of time is able to build its own user model. The following step of our approach is to define the meaning of a *probabilist log*, the semantics of a *good/bad user behavior* and the notion of a *suspicious behavior*.

Next, we identify the *probabilistic traces* of user models. These traces will be used in order to provide a coverage degree of a possible user behavior with respect to another. These traces are sequences of pairs of action/probability. We say that $\alpha_1, p_1/\alpha_2, p_2/\ldots/\alpha_n, p_n$ is a probabilistic trace of the user models if there exists these transitions in the automaton:

$$q_0 \xrightarrow{\alpha_1,-,p_1,-} q_1 \xrightarrow{\alpha_2,-,p_2,-} s_2, \ldots, q_{n-1} \xrightarrow{\alpha_n,-,p_n,-} q_n$$

Following, the notion of a good/bad user behavior represents a possible error of the system. The idea is that if a sequence of actions performed by a user in an organization is not presented in the user model that it has generated, then, a warning message should be sent to the administrator in order to ask him if this is a good or a bad sequence. On the one hand, if ω is good then our approach dynamically will accept it and will update the model. On the other hand, the

model continues without any change. Let $\omega' = \langle \alpha_1, t_1, \ldots, \alpha_n, t_n \rangle$ be a user behavior ($t_1, \ldots, t_n$ represent time stamp). We say hat ω' is good if there exists the following probabilist trace of the automaton $\alpha_1, p_1, \ldots, \alpha_n, p_n$. However, if this probabilist trace does not exist then ω' is classified as bad.

Finally, we introduce the notion of suspicious behaviors. Basically, these are good interactions of the users with respect to the organization, but the probability of these actions do not match with those that appear in the model. For doing this, instead of comparing a sequence we need a set of sequences. To compute them, first we need to assign a probability value to each behavior. Let $\omega = \langle \alpha_1, t_1, \ldots, \alpha_n, t_n \rangle$ be a sequence of actions of a user. We say that the probability to happen ω is the probability to perform each action and to stop after the input n. That is, if ω is good then there exists a probabilist trace $\alpha_1, p_1, \ldots, \alpha_n, p_n$. This means that there exists the following transitions $q_0 \xrightarrow{\alpha_1,-,p_1,-} q_1, \ldots, q_{n-1} \xrightarrow{\alpha_n,-,p_n,-} q_n$. Finally, the probability of ω is $p_1 * \ldots * p_n * \texttt{stop}(q_n)$. Where $\texttt{stop}(q_n)$ is the remainder to 1 of the probability values associated to the transitions outgoing from this state.

For example, let us consider the user model presented in Figure 4 and the following four user behaviors (we remove the time stamps because they are not used in this approach) $\omega_1 = \langle \texttt{AccessFile}, - \rangle$, $\omega_2 = \langle \texttt{AccessFile}, -/\texttt{UpdateFile}, - \rangle$, $\omega_3 = \langle \texttt{ModifyFile}, - \rangle$, and $\omega_4 = \langle \texttt{ModifyFile}, -/\texttt{UpdateFile}, - \rangle$. The probability associated with them are:

sequence	*Operations*	*Value*
ω_1	$\frac{2}{3} * 0.5$	$\frac{1}{3}$
ω_2	$\frac{2}{3} * 0.5 * 1$	$\frac{1}{3}$
ω_3	$\frac{1}{3} * 1$	$\frac{1}{3}$
ω_4	$\frac{2}{3} * 0$	0

Let us remark that the probability values for ω_1, ω_2 and ω_3 were $\frac{1}{3}$. These values are the expected one because we generated this user model from ω_1, ω_2 and ω_3. The probability associated with ω_4 is 0. It means that this sequence was never performed before, while the user model was created, thus, it represents a *bad* error.

3 Propagating Knowledge

In this section we present the architecture of a collaborative scenario, and how the information about the users are shared in order to prevent some attacks or bad behaviors.

In Figure 5 there are represented three different organizations that might share resources (the circles). Each organization has its *local set of security policies* that defines the roles and the access for its employees. Moreover, each shared-resource has associated a user model (represented by a square). In particular, in

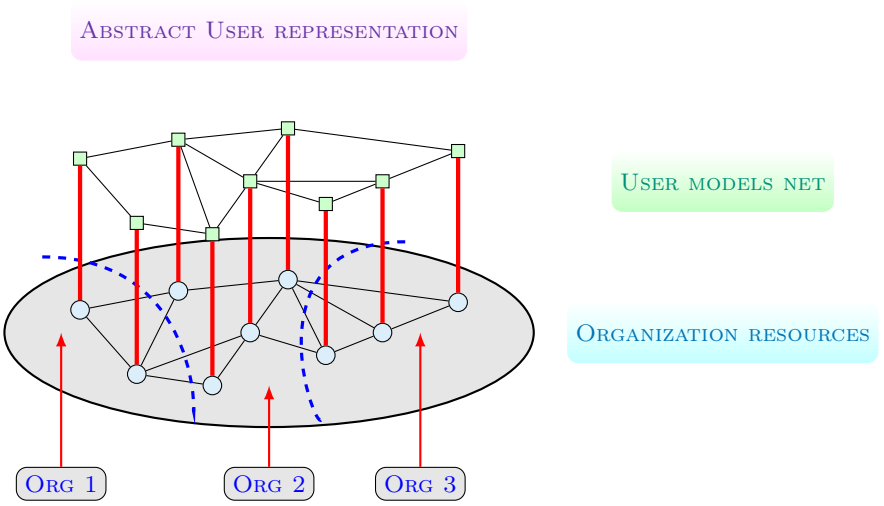

Fig. 5. Example of collaborative representation

this example the first organization has two shared resources, the second one four resources and the third one three resources. The lines between different resources denote a possible dependency of a resource with another resource. Sometime this link is inside the same organization but, sometimes they link shared resources of different organizations.

In addition, the user models, represented by squares, have two different kind of connections. On the one hand they are connected with the shared-resource from where this model was created. The second kind of link is made based on the *similarity* of the users. Previously, when the different behaviors of the users were collected into logs we considered that they only contained the action and the time stamp. But we can consider that there are recorded also some *metadata* in those files. Let us note that in the legal community, the term metadata has been defined as "a variety of data associated with electronic documents or files" [10]. The lines that links the user models in Figure 5 define different relationships among them. These relationships will be used in the algorithm to compare two different user models, and dynamically decide whether those models should be updated.

Let M_1 be a user model.We will say that M_1 is up-to-date with respect to their neighborhood if at least the 50% of its possible user behaviors appears in its associated neighbors. On the contrary, if M_1 is not up-to-date then its neighbors will start sending to him some of their more relevant behaviors in order to update M_1.

Let us introduce this notion with our current example. In Figure 6 it is presented the last user model that we created and two of its neighbors. Note that according to our definition, our user model M_1 should contain at least 50% of the behaviors of its neighbors. On the one hand, the user model represented on the

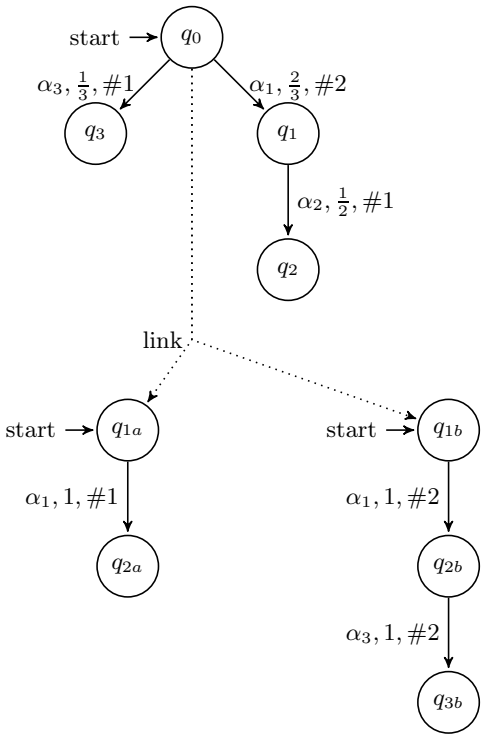

Fig. 6. User model net

left, by means M_{1a}, only represents one sequence of user interactions: $\langle \alpha_1, t_1 \rangle$. The probability to perform this sequence in M_1 is $\frac{1}{3}$. On the other hand, the user model represented on the right, by means M_{1b}, only contains $\omega = \langle \alpha_1, t_1/\alpha_2, t_2 \rangle$. The probability to perform ω in M_1 is 0, so at the end we have that the sum of representativity of the neighbors of M_1 in it is $\frac{1}{3}$. Thus, M_1 should be updated, that is, to spread the knowledge from their neighbors to M_1 as follows:

1. Select the set of behaviors of M_{1a} and M_{1b} that are not presented in M.
2. Sort these sequences with respect to their representativity.
3. Send one by one these sequences to M_1, and update the model until the representativity of M_{1a} and M_{1b} would be bigger than or equal to 50%.

In addition, according to previous updating algorithm, M_{1b} sends to M_1 the sequence $\langle \alpha_1, t_1/\alpha_2, t_2 \rangle$. M_1 receives this sequence and it is upgraded. In Figure 7 it is shown M_1 after the updating process. Now we can check that in this model the probability associated with $\langle \alpha_1, t_1 \rangle$ changed to $\frac{1}{4}$ however, the probability associated to $\langle \alpha_1, t_1/\alpha_2, t_2 \rangle$ is $\frac{1}{2}$, so at the end we have that the representativity of their neighbors is at least 50% of its behavior.

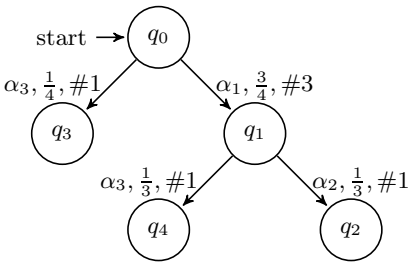

Fig. 7. Updating the model

4 Conclusions and Future Work

In this paper we have presented a methodology to check correct and incorrect behaviors in a collaborative scenario. By using data mining techniques we generate user models in order to guide this technique. Due to the fact that we are in a collaborative scenario, the knowledge about the users should be spread among the different nodes. Thus, we also present a methodology to upgrade the models with respect to its neighbors.

As future work we plan to extend this framework to study the inclusion of user models to check the security policies of the nets of virtual organizations.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: 19th ACM Int. Conf. on Management of Data, SIGMOD 1993, pp. 207–216. ACM Press (1993)
2. Andrés, C., Llana, L., Rodríguez, I.: Formally comparing user and implementer model-based testing methods. In: 4th Workshop on Advances in Model Based Testing, A-MOST 2008, pp. 1–10. IEEE Computer Society (2008)
3. Andrés, C., Merayo, M.G., Núñez, M.: Using a mining frequency patterns model to automate passive testing of real-time systems. In: 21st Int. Conf. on Software Engineering & Knowledge Engineering, SEKE 2009, pp. 426–431. Knowledge Systems Institute (2009)
4. Cao, J., Chen, J., Zhao, H., Li, M.: A policy-based authorization model for workflow-enabled dynamic process management. Journal of Network and Computer Applications 32(2), 412–422 (2009)
5. Coma, C., Cuppens-Boulahia, N., Cuppens, F., Cavalli, A.R.: Interoperability of context based system policies using O2O contract. In: Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems, SITIS 2008, pp. 137–144. IEEE (2008)
6. Elrakaiby, Y., Cuppens, F., Cuppens-Boulahia, N.: Formal enforcement and management of obligation policies. Data & Knowledge Engineering 71(1), 127–147 (2012)
7. Frank, E., Hall, M., Holmes, G., Kirkby, R., Pfahringer, B.: Weka - a machine learning workbench for data mining. In: Maimon, O., Rokach, L. (eds.) The Data Mining and Knowledge Discovery Handbook, pp. 1305–1314. Springer (2005)

8. Franke, U.J.: Managing virtual web organizations in the 21st century. IGI Publishing (2002)
9. Hoogendoorn, M., Jonker, C.M., Schut, M.C., Treur, J.: Modeling centralized organization of organizational change. Computational & Mathematical Organization Theory 13(2), 147–184 (2007)
10. Liao, S.-H., Huang, H.-C., Chen, Y.-N.: A Semantic Web Approach to Heterogeneous Metadata Integration. In: Pan, J.-S., Chen, S.-M., Nguyen, N.T. (eds.) ICCCI 2010, Part I. LNCS, vol. 6421, pp. 205–214. Springer, Heidelberg (2010)
11. Nguyen, T.-D., Islam, M. M., Al-Saffar, A., Park, J.-Y., Huh, E.-N.: Secure Collaborative Cloud Design for Global USN Services. In: Pan, J.-S., Chen, S.-M., Nguyen, N.T. (eds.) ICCCI 2010, Part I. LNCS, vol. 6421, pp. 178–187. Springer, Heidelberg (2010)
12. Salay, R., Mylopoulos, J.: The Model Role Level – A Vision. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 76–89. Springer, Heidelberg (2010)
13. Travica, B.: Virtual organization and electronic commerce. SIGMIS Database 36(3), 45–68 (2005)
14. Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J.: Formal methods: Practice and experience. ACM Computing Surveys 41, 19:1–19:36 (2009)
15. Wu, M.-T., Hong, T.-P., Lee, C.-N.: An Improved Ant Algorithm for Fuzzy Data Mining. In: Pan, J.-S., Chen, S.-M., Nguyen, N.T. (eds.) ICCCI 2010, Part II. LNCS, vol. 6422, pp. 344–351. Springer, Heidelberg (2010)
16. Zamfirescu, C.-B., Candea, C.: Planning in Collaborative Stigmergic Workspaces. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011, Part II. LNCS, vol. 6923, pp. 160–169. Springer, Heidelberg (2011)