Intelligent Environments 2020
C.A. Iglesias et al. (Eds.)

doi:10.3233/AISE200044

Managing Preference Profiles in Multi-User Intelligent Environments

Juan Carlos AUGUSTO [a,1], Andrés MUÑOZ [b]
[a] *R.G. on Development of Intelligent Environments, Department of Computer Science, Middlesex University London, U.K.*
[b] *Polytechnic School, Universidad Católica de Murcia, Spain*

Abstract. Development of Intelligent Environments have been so far mostly ad-hoc and here we investigate one fundamental bottleneck which needs to be addressed to facilitate more effective developments in the future: handling of user preferences in a multiple user environment. This paper analyzes some cases which combine different approaches to manage users preferences combining services at higher and lower levels with user-led and environment-led approaches. We assess some practical pros and cons in each of these combinations as well as some more fundamental building blocks which developers need to reflect on from a scientific point of view before embarking on the engineering of these systems.

Keywords. Intelligent Environments, User Preferences, Multi-user Environments, System Design

1. Introduction

Intelligent Environments (IE) [1] should be designed to satisfy their users. Most environments are inhabited or used by several users, which leads to decision making dilemmas for the environment [2]. We are investigating ways of representing user preferences and reasoning with the partial orders they represent, in an attempt to automate systems more aligned with user's expectations [3].

Managing user preferences is not new and has been done before within closely related communities such as Pervasive Systems, Ubiquitous Systems and Ambient Intelligence (see for example [4,5]). However, these previous proposals tend to be focused on one user, or more focused on the environment than the humans or assume a "one solution fits all" approach, none of which we think are entirely satisfactory. We think there is scope for a deeper and more ambitious debate which our community should have to support the next evolutionary step in our area, which is reaching certain maturity for single users but still being very basic for the most common multiple-users scenarios.

In this paper we attempt to highlight some of the challenges awaiting us and provide a framework which facilitates discussions and debate about this topic. First we introduce the basic components, concepts and notation which facilitates the explanation of some the challenges we are concerned about. Then we explain some of the scenarios which

[1]R.G. on Development of Intelligent Environments, Dep. of Computer Science, Middlesex University London, U.K.; E-mail: J.Augusto@mdx.ac.uk

we believe need further study. Finally, we explore some of the conceptual tools which we think need considering from a more scientific perspective as a preparation to the engineering of such systems with a deeper understanding.

2. System Architecture

Consider an Intelligent Environment ξ, examples could be a smart home, a smart office, a smart hospital. Each of these have associated a number of users[2] $\Upsilon = \{U_1, \ldots, U_u\}$ where each U_i may have an associated scale creation process which associates scales with users: $S(U_i) \rightarrow S_{U_i}$. But, what sort of process would that be?

Sometimes these scales may be external so the scale creation process can be as simple as accepting it. Other times users can be guided in a process to create a personalized scale. See examples of generic scales in Fig. 1(a) below for visitor, adult and teenager, although these could be also for a specific person, say the adult could be John or Alice.

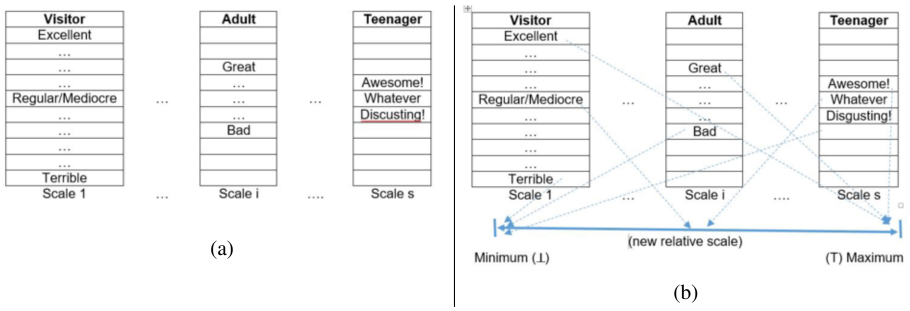


Figure 1. a) Examples of individual preference scales; b) Different individual scales being mapped into a system master/reference scale.

To facilitate discussion of various issues related to preferences scales management we will assume each scale S_{U_i} can be represented as an ordered list (we will assume here they are ordered in growing level of importance) of adjective labels $A_i = [a_1, \ldots, a_z]$ some of which may be synonymous (represented as a set of equally meaningful options), leading to a number E of meaningful adjectival elements represented, so $E \leq z$. For example, in Fig. 1(a) if we assume meaningful adjective labels were provided in each cell with dots, user visitor will have 11 adjective labels but only 10 values represented, because Regular and Mediocre are considered by that person to have the same meaning and value: $S_{visitor}$=[Terrible,..., {Regular, Mediocre},..., Excellent]. In some environments it may be that visitors are associated with a default preferences profile, whilst in others it may be the person has a historic profile stored and retrieved at arrival, or it may be generated at arrival based on information provided by the user or the system, so for example when John, a hypothetical user, is detected as a visitor in ξ he is allocated with the profile S_{John}=[Terrible,..., {Regular, Mediocre},...,Excellent]. For some user, say "x", it could be the case that U_x=[ ] because they do not have one or do not want to have a preferences profile in a given environment. Therefore, the system will not be able to consider those

[2]Typically human, although non-human entities like pets and robots can be considered users of an environment too and with rights to have preferences.

individuals preferences in its decision making, and as a result the system decisions may be less aligned with the user preferences and perhaps have a less satisfactory experience of that environment. Scales can be imported externally from the user or they can be created on demand as a new user arrives to an environment. In any case we will expect the information from a user can be linked with indices and even when stored internally with scales from other users these indices help us identify which elements of the merged scale belong to which user. As a simplifying assumption in these scenarios we can assume a U.N.A. (unique name assumption) mechanism can be worked out so that all labels in the system can be univocally attached to different individuals without confusion.

In Intelligent Environments there is a need to understand how a user will perceive the adequacy of the services being exposed to. Also as these environments are usually shared there is also a need to understand how the perceptions from different users relate to each other and how the system can manage these combinations of value scales. Hence one of the main topic concerning us in this study is how ξ can integrate the scales of its users (see Fig. 1(b)). The new environment "master scale" S_ξ should be able to accommodate and combine all scales from all users in that environment ξ.

The preference integration problem augments its complexity when we assume each ξ has a number of associated services the environment is supposed to manage in favour of the users. We can represent these services as a set $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_s\}$, which we assume non-empty given we are assuming an environment which can do at least something useful. As a result, each user may have a different scale for each service, represented as $S_{U_i}^{\sigma_j}$, that will need to be integrated with the rest of users scales for that service. So we have some interesting variables emerging here already, different users, different services, and different value scales about what users consider better or worse.

It could be that these user value scales have some metrics attached and in the ideal case if these metrics are the same or there are well known systems to transform one metric system into the other then allows us to immediately relate the elements of both scales. Say we have two users of an office, Alice and John, and $S_{Alice}^{temperature}$=[chilly, mild, hot], $S_{John}^{temperature}$=[cold, temperate, warm]. There is no way as such to determine whether what Alice has in mind when stating the environment is mild is the same than what John means when stating the the environment is temperate. Without this information we do not have a way to order them and the system does not know whether temperate is the same than mild or colder or hotter and by how much. As a result, we will apply the following process: we assume each adjective label has attached a value, so that if $A_i = [a_1, \ldots, a_z]$ is a list of adjective labels, $v(A)$ gives us a vector $[v(a_1), \ldots, v(a_z)]$, e.g. $S_{Alice}^{temperature}$=[(chilly, 20°C), (mild, 25°C), (hot, 30°C)] and $S_{John}^{temperature}$)=[(cold, 17°C), (temperate, 22°C), (warm, 26°C)]. Merging these two individual scales in one will result in $S_\xi^{temperature}$=[(John_cold, 17°C), (Alice_chilly, 20°C), (John_temperate, 22°C), (Alice_mild, 25°C), (John_warm, 26°C), (Alice_hot, 30°C)]. Should it have been $S_{John}^{temperature}$=[(cold, 41°F), (temperate, 77°F), (warm, 95°F)] then the merging would have resulted into $S_\xi^{temperature}$=[(John_cold, 5°C), (Alice_chilly, 20°C), (Alice_mild, 25°C), (John_temperate, 25°C), (Alice_hot, 30°C), (John_warm, 26°C)]. We will discuss towards the end of the article the formalities of operations such as merging of scales.

Sometimes these values $v(a_i)$ may not be available and the system will have to apply some default assimilation process, for example, if an incoming scale S has to be assimilated into a system global scale S_ξ, then it could be considered that the highest value of

S is assimilated with the highest value of S_ξ and internal values in between are proportionally spread out. Say $S_{\xi_1} = [\bot = b_1, b_2, b_3, b_4, b_5 = \top]$, and $S_{\xi_2}[\bot = l_1, l_2, l_3, l_4 = \top]$, then the merging could result on $S_\xi = [\{b_1, l_1\} b_2, l_2, b_3, l_3, b_4, \{b_5, l_4\}]$ and unknown values calculated relative to border neighbour known values, for example the $v(l_3) = (v(b_3) + v(b_4))/2$.

Therefore we expand our initial basic concept of a user scale into a list of pairs (adjective_label, value(adjective_label)): $S_{U_i} = [(a_1, v(a_1)), \ldots, (a_z, v(a_z))]$. As we expressed at the beginning, some scale levels may have more than one adjective label associated with the same meaning or relative value within the scale, for example having more than one word to refer to the same sensation of warmth. So the most complete description is:

$$S_{U_i} = [\{(a_1, v(a_1)), \ldots (a_1^n, v(a_1^n))\}, \ldots, \{(a_z, v(a_z)), \ldots, (a_z^m, v(a_z^m))\}]$$

Each system may consider what is the standard internal translation unit, could it be numerical or not. For simplicity, we assume $\mathbb{R}$ (actually a suitably finite subset of it will be enough) as the internal universal numerical scale all values can be ultimately compared through. Hence, when throughout this article we refer users scales: $S(U_i) \rightarrow S_{U_i}$, we refer to them as in the sense discussed in the paragraph above, that is with a value function attached in each of these components, even if this sometimes may return trivial values because the scale as not been provided with meaningful values.

3. Scales Integration Scenarios

There are various different ways of organizing preferences and their management and most likely there is no single way which is the best for all environments. Consider for example the variety of needs amongst environments with such diversity of privacy, security, population and objectives as smart homes, smart offices, smart hospitals, smart shopping, etc. However, it is in the interest of our community to explore, understand and find recipes which can work for at least certain subsets of those problems.

Two interesting conceptual dimensions which we think may help to conceptualize this problem are:

- Whether the user's scale is about his/her perception of the whole IE ξ or is about each specific service $\sigma_i \in \Sigma$ of ξ. Hence, we have two categories we are interested in exploring: ξ-type and σ-type.
- Whether the management system is such that it allows different external profiles to be accommodated in the system or not. We can call these *Human-led* or *System-led*, meaning in the former the system should adapt to the scale the user is bringing whilst in the later the user has to fit in her/his preferences into what the system offers.

We explore these four combinations (in no particular order) for a smart office scenario: ξ-type / Human-led (Fig. 2a); σ-type / Human-led (Fig. 2b); ξ-type / System-led (Fig. 2c); and σ-type / System-led (Fig. 2d).

While exploring the scenarios, we will take into account the four opposing forces involved in the design of IEs in accordance with the Intelligent Environment manifesto [1], namely Cost, Complexity, Services and Privacy as illustrated in Fig. 3a.

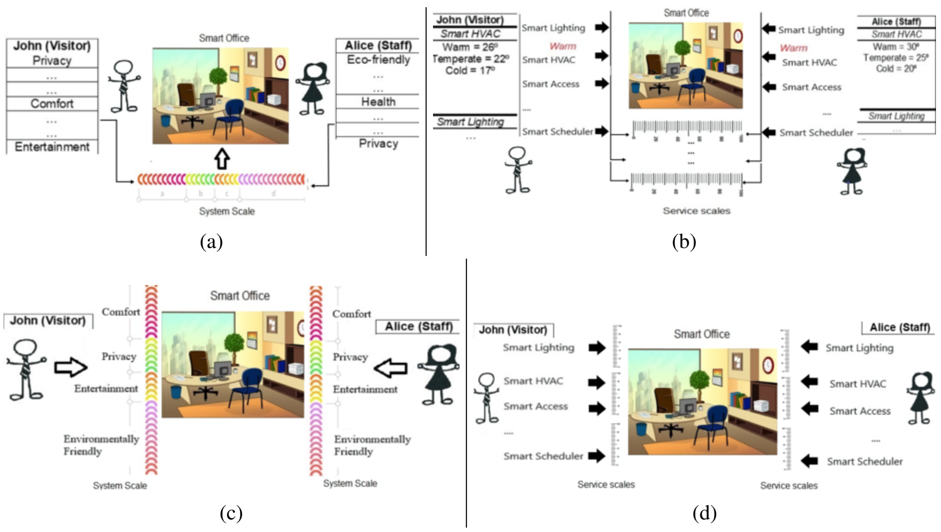


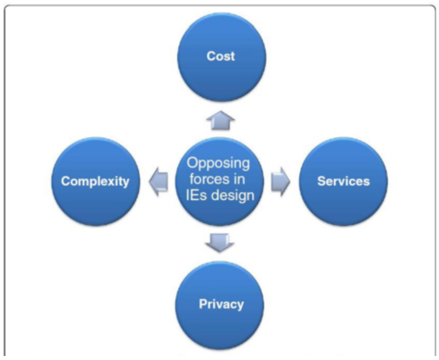
Figure 2. Four combinations of user preferences: a) ξ-type/Human-led scenario; b) σ-type/Human-led scenario; c) ξ-type/System-led scenario; d) σ-type/System-led scenario.

Let us suppose that there are two users interacting in this smart office scenario, Alice and John, who are co-working and sharing the same office. Alice is a staff member of the company whereas John is a visitor from an abroad branch of the same company. Let us explore how the definition of their preferences is performed in each scenario of Fig. 2.

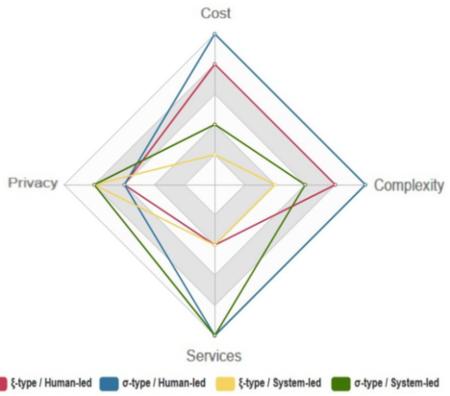
In the ξ-type / Human-led scenario (Fig. 2a), both users are prompted to express their own order of preferences for any IE. Thus, users are free to choose any valid preference language for the smart environment and to include any kind of possible preference, either it can be processed by the specific smart environment or not. Note that some preferences values may coincide for different users (e.g., "Privacy"). In this case, John indicates to the IE that the privacy issues in the office (i.e., a secure and non-shared Internet connection but also shading windows to avoid sideways glances) are the most important element for him, followed by the availability of comfort elements, and stating that the entertainments options in the office (e.g., notification about leisure activities with co-workers or automatic management of entertainment spaces in the office) is the least important element. On the other hand, Alice indicates to the IE that her main preference is to use the available services in an environmentally-friendly way (e.g., reducing the use of lighting or HVAC devices), which is preferred to the use of office elements devoted to create a healthy workplace, and finally the least important elements for her are the ones related to privacy issues. This scenario presents several challenges. First, there should exist a (preferably standard) language to express the users preference that is understood by the IE. Ontologies [6,8] may come here as a useful resource. Secondly, the users preferences should be translated to the IE preference scale. For example, in Fig. 2a a system scale is shown where each segment (a, b, c and d) represent a preference element in the system. Note that these preference elements will not necessarily be the same as the users preference ones (for example, it could be that without further guidance John's "Comfort" concept may be represented in the IE as the "Health" concept). As a result, a matching process is needed to solve this problem. Ontology matching or alignment [7] could be applied here. Finally, conflicts in IE will arise when different users' preference orders affect to the same elements in the IE. For example, John's privacy preference resulting in shading the windows will imply in using more lights, which conflicts with Alice's eco-friendly preference on reducing the use of artificial lights. A possible solution based on argumentation was provided by the authors elsewhere [2]. This scenario introduces a high complexity due to the large amount of external information to be integrated in the system and the merging operators involved in this task. This includes the capability of the system of "understand" the different languages in which the external user preferences could be expressed, augmenting the cost of the system for each language to consider. Besides that, the services offered may not be clear to the users as they perceive the system as a whole unit, so they cannot express their preference for a specific service. Finally, privacy matters should be taken into account as the users may own different preference profiles on different IEs (e.g., the preference profile for a smart home will be different to the preference profile for a smart office). As a result, it must be clear which preferences are being communicated to the new IE and which ones are to be kept out.

In the σ-type / Human-led scenario (Fig. 2b), the users are now prompted to define their preferences for each service available in the IE (e.g., smart lighting, smart HVAC, etc.). In this case, each user defines his/her preferences value for the possible states of each service. For example, in the case of the smart HVAC with three possible states: warm, temperate and cold, the users could assign a numerical temperature value to each state. Then, they could order these states for each service and send the information to the IE. The challenges found in this scenario are the following. First, the states of each service may be unknown for new users who are not previously registered in the IE. In fact, these users may import their preference profile from their previous IE and they may want to use it in the new IE. It is even possible that some services that were available in their former IE are not available in the new IE any more. Then, a matching process is needed again between the users preference information for each service and the actual services in the IE. This process may include the translation of the users preferred value (e.g., warm) to the actual value in the service (28°C). Other challenges are the language used to express the preferences for each service and the potential conflicts among users preference, as discussed in the ξ-type / Human-led scenario. Note that in this case, even though two users prefer the same value for a service, it may lead to a conflict. For example, in Fig. 2b both John and Alice prefer the HVAC to be activated as "warm", however their concept of "warm" is different since John considers it as 26°C whereas Alice does it as 30°C. This scenario introduces a higher complexity and cost than the previous one (ξ-type / Human-led), since the dependence of external information now increases for each new service included in the user preferences. However, the user perceives a better alternative to manage services. Again, privacy needs to be consider since different user profiles for the same service in different IEs may coexist.

In the ξ-type / System-led scenario (Fig. 2c), each user is presented with a predefined preference list that should be ordered by them. In this case, the preference list is the same for every user. For example, John and Alice are presented with a list including 4 preference elements, namely "Comfort", "Privacy", "Entertainment" and "Environmentally Friendly". In this system it should be decided how the preferences are ordered, taking into account if total orders are required or if partial orders are allowed, or if each preference element is given a weight or numerical value, etc. A challenge identified here is how the IE will behave as an integral unit to satisfy the most of the preference requirements of a user. Another challenge is related to conflicting preference in multi-user sce-



(a)



(b)

Figure 3. a) Opposing forces in IE development [1]; b) Relationships of the IE driving forces ξ-type/σ-type and Human-led/System-led scenarios.

narios, as discussed for the previous scenarios. The complexity and cost of this scenario are lower than in two previous scenarios, since no external information regarding user preference needs to be integrated, and for this reason too, privacy matters could be more easily managed. However, the system is more restricted with respect to the definition of preferences than in the previous scenarios, and therefore users are constrained to the language imposed by the system, reducing the user's perception about the management of the services offered.

Finally, in the σ-type / System-led scenario (Fig. 2d), each user is presented with a predefined scale for each service. In this case, users need to configure the value for each service within the parameters offered by the service. The most likely challenge here is dealing with conflicting values for each service for different users. Another challenge could be the situation of services for the same user "competing" among them, e.g., a user indicating a high preference on services offering comfort features but also a high preference on using eco-friendly services. This problem may be solved by the users themselves by stating an order among services, but it could be a rather complicated process when a large number of services are offered and no explicit connection among them has been foreseen. Therefore, an automatic way of solving this problem will be desirable. The opposing forces in the design of IEs for this scenario follow the same rationale than in the ξ-type / System-led scenario, with a small increase in complexity and cost due to the necessity of managing different services, whilst improving the user's perception on the services offered as they can now be managed separately. From the above scenarios we can see different forces at play and competing with each other, and it seems one principle is emerging for engineers to pose to the project funders. If we accept an IE as a system which should aim to maximize the perceived effect of context services by the environments users [3], then human-led systems requiring a large amount of information so the system can make more informed choices may be favoured. However, if there are budget restrictions then system-led options will be simpler to design and create, although the system may be less satisfactory. In all multi-user options there will be disagreements, just the more effort is gathered from user preferences the more information the system may have to consider exceptions instead of applying blanket decisions to all. Fig. 3b summarizes the mutually competing of the opposing forces in these scenarios. It is observed that the *Human-led* scenarios are expected to have the highest cost and complexity, due to the large amount of information to be integrated in them and the processes associated to this task. Likewise, the σ-*type* scenarios are expected to be more complex and costly than ξ-*type* ones, due to the management of services for the same user and also the management of the same service among different users. Regarding the Service dimension, it is obvious that the *System-led* scenarios are the ones offering a broader palette of services. Finally, it can be considered that *System-led* scenarios could preserve better the user privacy, as they do not need to be fed with user's profiles imported or created in other scenarios.

4. Preparing the Technical Ground

At a more technical level, we can consider the question of what conceptual tools developers will need should they try to make user preferences a more systematic and better organized part of their system creation. So far we have considered the following ones.

4.1. Elementary Operations

Here we define some simple tools which will help in the next subsections.

Given a scale $S = [e_1, \dots, e_l]$, length(S)=l is the number of elements in the scale. Remember a scale can have non-atomic elements with some of the elements consisting actually of a set of adjective labels which are considered to have the same overall meaning, some sort of synonymous as far as the scale concerns. So if $S = [\{(cold, 5)\}]$, $S' = [\{(cold, 5)\}, \{(hot, 25)\}]$ and $S'' = [\{(cold, 5)\}, \{(hot, 25), (warm, 25)\}]$ then $length(S) = 1$, $length(S') = 2$, and $length(S'') = 2$. As "preferential synonyms" are sets we can use the usual set operations such as cardinality to work out in how many ways a concept is described by a user. We can also use the usual list operations, for example to retrieve an element which is in the i-th position or an element which adjective label or value meet certain properties.

Also it will be often necessary to compare two elements from two different scales. Here several options open up based on how literal and strict the match between these two elements is expected to be. For example, if one element in the John's temperature preferences scale is $e_1 = (warm, 30°C)$ and the other element in Alice's temperature preferences scale is $e_2 = (warm, 30°C)$ then that is an exact match. If we compare $e_1 = \{(warm, 30°C)\}$ and $e_2 = \{(warm, 30°C), (hot, 30°C)\}$ the answer is not so obvious: the content of e_1 is in e_2, however strictly speaking e_1 is structurally different to e_2 as the later contains a "scale synonymous". Thus, a comparison which is 'modulo "scale synonymous" ' will be intelligent enough to return true, one which is too literal will return false. As a result, we can define two match operations:

Literal element match ($e_1 \doteq e_2$): $e_1 = \{(a_i, v(a_i)), \dots, (a_i^k, v(a_i^k))\} = e_2$

Similarity element match ($e_1 \simeq e_2$):]

$$e_1 = \{(a_i, v(a_i)), \dots, (a_i^{k_1}, v(a_i^{k_1}))\} \text{ and } e_2 = \{(a_j, v(a_j)), \dots, (a_j^{k_2}, v(a_j^{k_2}))\}$$

This can be "adjective label based": there is at least one $a_i^{k_m}$ and one $a_j^{k_n}$ such that a_i and a_j are the same adjective label. Or, it can be "adjective value based": with some $a_i^{k_m}$ and some $a_j^{k_n}$ such that $|v(a_i^{k_m}) - v(a_j^{k_n})| < \delta$ for some pre-specified δ, i.e., the values are "close enough" regarding a context. These types of matching ignore the possible presence of various synonyms for some elements in any of the scales.

4.2. Built-in Operations

Let be $S_1 = [\ldots\{(a_i, v(a_i))\ldots(a_i^{k_1}, v(a_i^{k_1}))\}\ldots]; S_2 = [\ldots\{(a_j, v(a_j))\ldots(a_j^{k_2}, v(a_j^{k_2}))\}\ldots]$ then the following *Logical Scale Operations* can be considered:

Compares=(S_1,S_2): a Boolean function returning true when both scales S_1 and S_2 are "equivalent" and false otherwise. We can again establish different ciriteria based on how strict the matching has to be at element level: $e_1 \doteq e_2$ or $e_1 \simeq e_2$ or other element comparison operator.
Inclusion(S_1,S_2): a Boolean function returning true when all elements of S_1 are in S_2 and false otherwise. This function again depends on how equality amongst elements is defined, whether literal or approximate.

Let be S and S' two arbitrary well-formed scales in the sense discussed above, then the following *Build Scale Operations* can be considered:

Add(e,S,S'): adds e at the appropriate place in list S resulting in S'.
Delete(e,S,S'): extracts e from S resulting in S'.
Replace(e,e',S,S'): replaces e by e' in S resulting in S'.
Transfer(S,S',S''): provided $length(S) = length(S')$ then for each i-th e of S the value of e overrides the value of the i-th element e' in S' and the resulting list is S''.

And the following *Scale Level Operations* can be considered:

Absorbing(incomingScale, S, S'): is an (adjective value) ordered merge, based on the values of each adjective label so that elements in "incomingScale" are added to S, resulting in S'.
Detaching(outgoingScale, S ,S'): for all elements in "outgoingScale" it effectively Delete(e,S,S').

4.3. System Properties

We can define an algebraic structure with the set of possible scales definable over a bounded fragment of $\mathbb{R}$ as the carrier and have associated the operations outlined above. Investigating this in more depth is a task we will not pursue here, although we recognize the value of investigating algebras with different properties which will affect the expectations from the developers in terms of the trade-off between feasibility of implementation and predictability of operations.

Algebraic structures have intrinsic properties derived from the initial assumptions imposed over them, however in our field we also have a growing number of consumer expectations which are external to the system and system engineer are increasingly under pressure to accommodate. For example, nowadays technology consumers are becoming more aware of their rights over their own data, and consequences of legislation such as GDPR give consumers more rights (and developers more obligation) on *traceability* of their data. This overall property over the set of scales in a system requires that data which is linked to a specific user should be traceable within the system in case a user requests his/her history to be removed. This can be achieved in practice in many ways depending on the way the databases are implemented. In general terms:

An environment ξ is *scale-traceable* if for every user U_i and scale S_{U_i}, it is possible to apply Detaching(S_{U_i}, S, S').

5. Conclusions and Future work

Creating intelligent environments is a challenging enterprise as our community has discovered in so many years of hard work before. However, some progress has been made and there is an increasing expectation that multiple user environments should and could be more feasible now. So we have analyzed above one important bottleneck, namely the lack of shareable resources for developers which want to tackle such systems. Most multiple users environments are created pretty much in an ad hoc manner.

Our approach above is a first look at this problem and it will not be hard to persuade us many more things need to be considered. As a first approach we have made some assumptions which in practice may require relaxing to be useful. For example, we assumed that given $S_{U1} = [\bot_1, \ldots, \top_1]$ and $S_{U2} = [\bot_2, \ldots, \top_2]$ then $\bot_1 \equiv \bot_2$ and $\top_1 \equiv \top_2$. How operations are affected when this assumption is removed and either or both of $\bot_1 \equiv \bot_2$ and $\top_1 \equiv \top_2$ do not hold, or we do not know, or is not accepted by one of the sides?

We have analyzed how these essential building blocks can be achieved in a system. ‘A priori’ we see various competing reasonable ways to develop these and we considered four of those which combine different possible and valid approaches to manage users preferences. These combine services at higher and lower levels with user-led and environment-led approaches. We did the exercise of exploring some of the practical pros and cons in each of these combinations to trigger some reflections on potential developers. We hope this will spark some discussion and sharing of tools which will help to treat development of multiple user environments in a more organized and professional way.

References

[1] J. C. Augusto, V. Callaghan, D. Cook, A. Kameas, and I. Satoh, “Intelligent Environments: a manifesto,” *Human-centric Computing and Information Sciences*, vol. 3, no. 1, pp. 1–12, 2013.

[2] C. Oguego, J. Augusto, A. Muñoz, and M. Springett, “Using argumentation to manage users preferences,” *Future Gener. Comput. Syst.*, vol. 81, no. C, p. 235-243, 2018.

[3] J. C. Augusto and A. Muñoz, “User preferences in intelligent environments,” *Applied Artificial Intelligence*, vol. 33, no. 12, pp. 1069–1091, 2019.

[4] E. Papadopoulou, S. Gallacher, N. K. Taylor, M. H. Williams, and F. R. Blackmun, “Context-aware user preferences in systems for pervasive computing and social networking,” in *1st Int. Conf. on Context-Aware Systems and Applications*, Vietnam, P. C. Vinh et al., Eds., vol.109. Springer, 2012, pp. 10–17.

[5] S. Gallacher, E. Papadopoulou, N. K. Taylor, and M. H. Williams, “Learning user preferences for adaptive pervasive environments: An incremental and temporal approach,” *TAAS*:(8)1, pp. 5:1–5:26, 2013.

[6] T. R. Gruber, “A translation approach to portable ontology specifications,” *Knowledge Acquisition*, vol. 5, pp. 199–220, 1993.

[7] V. Rajeswari, M. Kavitha, and D. K. Varughese, “A weighted graph-oriented ontology matching algorithm for enhancing ontology mapping and alignment in semantic web,” *Soft Computing* (23)18:8661-76, 2019.

[8] M. Martnez-Garca, A. Valls, and A. Moreno, “Inferring preferences in ontology-based recommender systems using WOWA,” *Journal of Intelligent Information Systems* (52)2:393-423, 2019.