Conflict and Interference Resolution for Physical Annotation Systems

Ahmad A. Alzahrani[1],[2], Seng W. Loke[1],Hongen Lu[1]

[1]Department of Computer Science and Computer Engineering,
La Trobe University, Melbourne, Australia
[2] Department of Computer Science, King Abdulaziz University, Saudi Arabia
{aaalzahrani@students., s.loke@, H.Lu@}latrobe.edu.au

***Abstract*. Physical Annotation (PA) systems have been widely used in recent years. They help users to annotate physical entities around them with digital data for different purposes such as education, tourism or personal memories. One of the main problems of the widespread uses of PA is the conflict and the interference between different annotations for the same entity. Therefore, this paper studies the conflict and interference issues in PA systems. The paper first provides a formal definition for PA conflict and interference, and then explains the causes and sources of them. Based on that, we propose detection techniques and then we provide policies to resolve conflict and interference in PAs. Finally, we provide an implementation of the techniques.**

Keywords - Physical Annotation; conflict and interference, pervasive computing, context-aware system.

I. INTRODUCTION

In our modern life, advanced development of telecommunication technologies allows users to annotate physical entities around them in digital form in a seamless integration between the real world and the digital world. Such a technology is generally called Physical Annotation (PA) systems, where users can annotate small entities such as a cup which often have a dynamic location that can be changed frequently, or can annotate large geographical areas such as a building which often have static locations. In recent years, PA systems have been used increasingly and can be used for different purposes – e.g., educational purposes where a teacher can annotate entities such as a printer or a class room with annotations to enrich the education process such as explaining things to his/her students. PAs can be used for tourism, health, telling stories or security purposes.

Generally, PA systems allow any user to annotate anything around him/her. However; things can get complicated by having a large number of annotations for the same entities as has been discussed in our previous work [1, 2]. One of the potential problems is the conflict and interference between annotated entities (targets) and their annotations which could cause many problems such as giving false information.

Therefore, in order to have useful PA systems that avoid conflict and interference, in this paper, we propose a dynamic conflict and interference management technique which is called CIRPA (short for Conflict and Interference Resolution for Physical Annotations) to detect and resolve conflicts and interference in PA systems. In this paper we extend our previous work [3] on a physical annotation system called *ALPHAsys*.

The rest of this paper is structured as follows: Section II provides our formal model for PAs which is called *ALPHA*. Then, we explain our location model that represents physical entities and physical annotations, which is called DPVW-model. In Section IV, we explain the causes and sources of conflict and interference in PA systems. Then in Section V, we provide techniques to discover and detect such a problem. Also we proposed some policies that can be used to resolve conflict and interference. In Section VI, we explain implementation called CIPRA, which is an extension on our previous work ALPHAsys. We discuss related work in Section VII, and conclude in Section VIII.

II. ALPHA: THE ADVANCED LOCATION MODEL FOR PHYSICAL ANNOTATION

In our previous work [1], we proposed a conceptual model of Physical Annotation systems; in this section, we briefly discuss the PA formal model and outline the main conceptual model called the ALPHA model which comprise three main parts: (A) the physical annotation (the link) which associates annotations to annotated targets; the link includes the context dependency, anchor properties, and so on; (B) the annotation part, which includes the content, type, and other attributes; and (C) the physical entity (the target) being annotated such as a location, or a small object.

A. The Link (a Physical Annotation)

We define a PA as the link or the bridge that connects the annotation component to the target component. The annotation part or the target part can be independent entities and can stand alone by themselves. However, the link is dependent on both the annotation and the target components. The following are the link's properties.

Definition A (physical annotation/link): A physical annotation (or link) is a 4-tuple of the following form:

$$pa = < paID, ann, t, cxD >$$

where $paID$ is the unique number to identify each PA, ann describes all properties which belong to the annotation (as given below in Definition B), t refers to the target, which is the annotated entity, i.e. $t \in T$ (the set of all possible targets as explained below in Definition C), and cxD is the annotation context dependency, i.e., cxD = {location, time, date, nearby person, nearby object, entity dependency} (in practice cxD are attributes with particular context values).

B. *The Annotation*

The annotation part comprises the following properties: the annotation ID, annotation type, the annotation content itself, users, groups, and author category.

Definition B (annotation): An annotation is a 6-tuple as the following form:

$$ann = < aID, type, ac, au, user, gr >$$

where aID is an annotation identifier, $type$ is the annotation type, i.e. $type \in TYPE$ such as education, tourism, health, ownership, security, entertainment, commercial or application-specific annotation types (e.g., price, ownership, comment, weight, and so on which are attributes relating to an application/use of the annotation), ac is the content media type (e.g., text, video), au is the annotation's author, $user$ is the annotation's intended user, gr is the annotation's intended group (i.e. annotations left for users in a group). Any one of these values (exceptaID) can be null or defaulted to a fixed value.

C. *The Target*

The target part describes the annotated entities (or *targets*) which exist physically in the real world. Before explaining the target properties, we would like to describe the target's different types. There are three different types of annotated entities, the first one is the single entity or what could be called an *atom* entity which always is one piece of physical object such as a cup, a person, or even a building, This type of annotated entity is the simplest type of entity, which also means they are easy to represent in a location model and easy to deal with in terms of creating, retrieving and querying annotations. The other type of the target is composed of more than one entity; this target type may be called a *collection* of entities, and is often more complex in terms of creating, retrieving, or querying their annotations. Also, a collection of entities often have similar properties in different ways which could be the colour, structure and so on. For example, the user may conceptually "make" a collection of cups in his/her room and give them a general annotation "my cups". So, a main feature of the collection is that it should be located within the same physical location (another example are books on the same bookshelf).

Another target type is the virtual group which refers to groups of entities physically existing in the real world, but generally not physically grouped together in one location, and may be far in distance from each other. The entities in a virtual group are grouped together via relationships between entities, or via sharing context, such as a virtual group that comprises all the user's possessions in the whole building in his/ her workplace, independent of where these objects are located in the building. This target type is associated with the idea of the annotation mapping property in our PA formal model, which means, for example, one annotation is linked to three different entities at three different locations in the location model. A virtual group is also associated with the context dependency property in the PA formal model. Context dependency means that one annotation is linked to a target entity; but this linking depends on different contexts or circumstances. For example, the annotation will be visible only if the annotated entity is in the presence of another specific entity, or person. So, a main feature of the virtual group is that, unlike a collection, it shouldn't need to have entities in close physical proximity or existing within the same small physical area (with respect to an a priori chosen location model). Figure 1 shows examples of a collection and a virtual group.

Definition C (target): A target is a 4-tuple of the form:

$$t = < t_{ID}, t_K, t_{LOC}, t_C >$$

where $t \in T$ is a target and T is a set of all possible targets (as defined with respect to a model that we explain later), T_ID is a set of all possible target IDs and t_ID $\in$ T_ID is a target identifier which can be a single value (or a set of identifiers t_ID $\in$ T_ID as explained below), T_K is the set of all target kinds, and $t_K \in T_K$ ={geographical_location, object, person}, t_K denotes a target kind (or a set of target kinds, i.e. or $t_K \subseteq T_K$), T_LOC is a set of all possible target locations which includes the geometric or/and symbolic locations (such a set might be predefined for a given system using a DPVW-model described later), t_LOC $\in T_{LOC}$ denotes a target location, and $t_C \in T_C = \{ta, b\}$ denotes target constraints, where ta denotes a target annotation type that must be of unary value (i.e., should only have one valid annotation of that type), so $ta \in TYPE$ which can be education, owner, tourism and so on. b denotes a Boolean value, which when true means that the constraint on the target is that it accepts (or user is allowed to leave) only one annotation of the annotation type ta at all times, whereas a false value indicates the target can accept more than one annotation for that particular annotation type ta, but only one annotation at a time is allowed to be retrieved in a particular context.

For example, a user can define a PC as an object as follows, t = (t102,"object", t504, {owner, true}), so the object was defined to prevent users from adding more than one annotation of the type "owner" for the PC. However, if the Boolean value is false, this means the PC can accept more than one annotation of type "owner", but only one can be valid (and is shown) at retrieval time (the method to decide which is valid/shown is discussed later). This could mean that the PC can be owned by different users, but in different times and context. Moreover, by having this constraint, that doesn't mean the target can't accept many annotations of some other annotation types.

III. DYNAMIC PHYSICAL AND VIRTUAL WORLD MODEL (DPVW-MODEL)

In our previous work, we created a location model to represent annotated entities. It is called DPVW-model. The model can represent different types of targets (i.e., atomic, collection and virtual group). Also, the DPVW-model can represent entities in indoor or outdoor locations. Figure 1 shows an example of the model that represents the Beth Gleeson building in Latrobe University. One of the advantages of this model is that it allows PA systems to retrieve annotations not only for the annotated entities, but also to retrieve the parents' annotations, i.e. such a model can serve as a mean to browse “surrounding” annotations. So, if you are in room 125, the PA system allows you to retrieve annotations for level two as well. In Figure 2, another DPVW-model shows the Doncaster shopping centre, we can use this model to annotate a whole section with one annotation to advertise a discount or a general message, such as annotating the TV section in JB Hi-Fi with a discount 10% which will be applied to all products in that section.

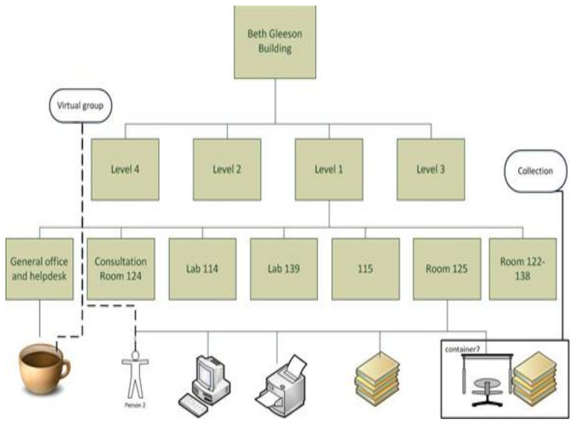

Figure 1: DPVW-model for BG building.

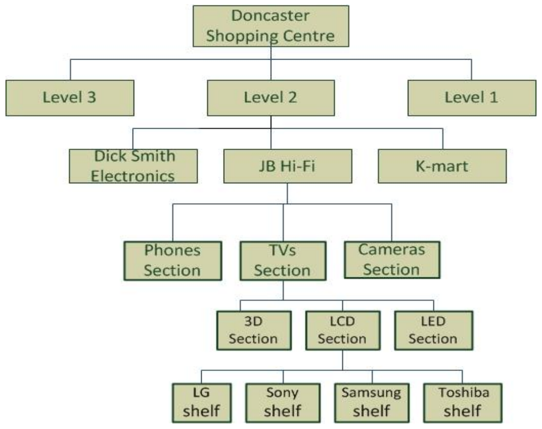

Figure 2: DPVW-model of Doncaster SC.

IV. CONFLICT AND INTERFERENCE IN PHYSICAL ANNOTATION SYSTEMS

As we noted before, there could be more than one annotation for one target, these annotations could be of different annotation types.

We may also have the same type of annotations for the same target but with different context dependencies as we discussed such as time, location and so on. Given a physical annotation $pa= <paID, ann, t, cxD>$, then a user u in context c can retrieve pa if (u = $ann.user$ or $u \in ann.gr$) & $ann.cxD.loc = c.loc$ & $ann.cxD.time = c.time$.

However, users may get multiple annotations of the same type for the same object which could cause interference or even a conflict between annotations especially if those annotations have conflicting data. Therefore, in this paper we denote interference or a conflict in a PA system as follows: a conflict and interference occurs in a PA system if more than one annotation has been retrieved at a time, and all of them have the same annotation type. This definition may not capture all possible kinds of conflict among annotations but provides a precise definition that is easily detectable, and also permits developers to enforce that certain annotations for a target must be singular (that is, annotations that belong to an annotation type specified in the target’s target constraint). In this paper, we differ between *conflict* and *interference* in PA as follows: both of them are detected when, in the same context (time, and so on), users retrieve more than one annotation, but a conflict has the additional condition that these annotations belong to the same annotation type ($AT_{multiple}$), and this annotation type is listed in the target’s target constraint. We also handle interference and conflict differently: an interference is handled by having the annotations combined, i.e. those annotations are allowed to exist together, whereas, for a conflict, only one, out of the many annotations retrieved of type $AT_{mutiple}$, is allowed to be used.

Also, before providing a formal definition of interference and conflict, we need to discuss the possible sources of conflict and interference.

There are different situations where the PA system has interference and conflict problems, and so we categorized the types of PA interferences and conflicts into two categories.

1) Target conflict and interference

This type of problem happens between targets themselves overlap, such as an overlap between two geographical areas (street and district). So if a user stands on the shared area between the two targets, he/she will get two annotations, one from each target. *Target conflict and interference* between annotations can be divided into *objects conflict and interference* and *regions conflict and interference* as follows.

Assume below that t1and t2 each refers to a collection of targets, i.e. a target container object, and e is a target object, the following points illustrate the interference and conflict arising from such object “overlap”:

- $e \in t1$, such as a cup inside a picnic basket; the cup has its annotations but the basket has annotations that apply to what’s inside it.
- $t1 \subseteq t2$, when the first container is effectively located inside another container, i.e. objects of one are found in the other.
- $t1 \cap t2$, where there are objects located in both containers at the same time, such as a cable on one end of it deemed to be located in the first container and the other end in the other container.

Now, for regions, assume t1 and t2 are regions and e is an object, the following points illustrate the annotation interference and conflict arising with regions overlap:

- e IN t2 an object inside a region such as a cup inside a room; and we have annotations for the cup (e.g., "belongs to John") which interfere with that in the room (e.g., "All objects in here belong to Mary").

- t1 PO t2, i.e. two target regions partially overlapping or intersecting such as a street and a geographical block (district), or a bridge between two buildings, as illustrated in Figure 3.
- t1 NTPP t2, i.e. a target is a non-tangential proper part of the other, such as higher level and lower level in a location model, or a room inside a building, as illustrated in Figure 2, the JB Hi-Fi shop is part of level 2 which is also part of the Doncaster shopping mall.

If either of these six situations occur, we say that that there is *interference among the targets*, inducing interference among annotations, i.e., we use the operator *"$\otimes$"* combining two targets, and thinking of $\otimes$ as either "∊", "⊆", "∩", "IN", "PO", or "NTPP", we write $|t1 \otimes t2|>0$ to mean that there is interference between objects or regions in the above ways.

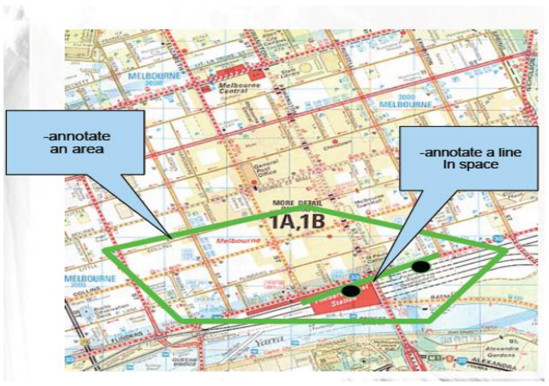

Figure 3: Examples of partial overlap between regions.

2) *Annotations interference and conflict*

This scenario can happen between annotations even if there is only one target, i.e., when it has more than one annotation, these annotations may vary in their meaning toward that target, so getting all these different annotations at the same time can make a false statement and causes a conflict between them.

For example, assume that there is a computer lab in a university. There are two annotations; the first one says "this lab is for subject A", and for the same time, the other annotation says, "this lab is for subject B". So it's obvious that the two annotations can't be true at the same time and this where a conflict between annotations can happen. The developer can represent this by having an annotation type called "activity" and then specifying this as a target constraint for the target (the computer lab) – this means that if there are more one annotations of the "activity" type, a conflict is detected, and must be resolved; this is convenient means of doing so without resorting to understanding the semantics of annotations. Note that, if an annotation of a type listed among the target's constraint with a false Boolean value, we may not simply stop the users from leaving more than one annotation of that type since the process of resolving the conflict can be useful and leaves room for greater users' expression and flexibility – we allow users to leave more than one annotations, i.e. allow such conflicts to occur, detect that, and then the system takes care of usefully resolving it – we discuss this further later.

Definition D (interference).

Interference happens between two annotations pa1 and pa2 in two cases; the first case if there is an intersection or overlap between two targets (where the targets may be objects or regions as we discussed earlier), or in the second case when one target has two annotations and both annotations have the same annotation type, annotation context time and annotation context locations. Also, both annotations' context locations are the same as the target's location (this applies when a target can have a dynamic location) and both annotation's user group or user are the same. More formally, let pa1, pa2 ∊ PA be physical annotations, then interference is determined by a Boolean function I on two annotations as follows:

$$I(pa1, pa2) = \begin{cases} true, if\ \ X\ is\ true \\ false, otherwise \end{cases}$$

where $X = \begin{cases} |\mathrm{pa1.t} \otimes \mathrm{pa2.t}| > 0, & \text{if}\ \mathrm{pa1.t} \neq \mathrm{pa2.t} \\ P & , otherwise \end{cases}$

$P = (pa1.cxd.loc = pa2.cxd.loc\ \&\ pa1.cxd.loc = pa1.t.loc\ \&\ pa1.cxd.time = pa2.cxd.time\ \&\ pa1.ann.type = pa2.ann.type\ \&\ (pa1.ann.user = pa2.ann.user\ or\ pa1.ann.group = pa2.ann.group))$.

Definition E (conflict).

Similar to interference, the only difference here is that the each involved target has constraints, so it accepts only one annotation of that annotation type at the same time.

Let pa1, pa2 ∊ PA be physical annotations, then conflict is determined by a Boolean function C on two annotations as follows:

$$C(pa1, pa2) = \begin{cases} true, if\ \ Y\ is\ true \\ false, otherwise \end{cases}$$

where $\mathrm{Y} = (X\ \ \& \ ((\boldsymbol{pa1.ann.type = pa2.ann.type})\ \&\ ((\boldsymbol{pa1.t.t_c.ta = pa1.ann.type})\ \boldsymbol{or}\ (\boldsymbol{pa2.t.t_c.ta = pa2.ann.type}))\ )$

(where X is as defined above for interference). Effectively, a conflict is an interference with an additional condition. So, the difference here is that the Y tests if the target constraint (T_C) (for the two interfering targets, or just that target if pa1 and pa2 have the same target) has a value that prevents having more than one annotation for a particular annotation type at the same time as discussed earlier.

V. Conflict and Interference Detection and Resolution for PA System (CIRPA Policy)

In this section we provide our proposed solution for detecting and resolving annotation conflict and interference in a PA system. It provides strategies for solving such a problem.

A) Detecting interference and conflict in a PA system

In this section, we explain how and when a PA system should detect and resolve conflicts or interferences. Two approaches can be employed:

1) Initial Conflict Prevention when leaving annotations (ICP): As the name implies, this approach aims to discover and prevent the conflict and interference before the problem occurs. This approach can be done on the server side when the admistrator set policies for the PA system. One way of doing this is by making constraints on targets by banning leaving more than one annotation of the same type for one target. So, the true value for the Boolean element in target constraint (T_C) prevents any user from leaving more than one annotation of that type.

An example for this approach is when an author of the annotation defines some targets in a university to have only one annotation of a particular annotation type at all time such as building name..

Therefore, no one can leave more than one annotation of type building name for those targets. Another example is in a shopping centre, where a PA admistrator can define the targets (which are products in this case) to have only one annotation of commercial type, so when the staff leaves an annotation on the product they can only have one annotation of commercial type. This step doesn't mean banning leaving more than one annotation for same target if they have different annotation types; so, for the product, there is only one annotation of type commercial, but we can have many annotations of type product information. The ICP approach has the advantage of preventing conflicts in the early stage which also will improve the efficiency of creating and retrieving annotations. However, the ICP approach may not be always the best solution for all targets especially when we want targets to have different annotations associated with different contexts.

2) Dynamic Conflict Detection when retrieving annotations (DCD): This approach is done during run-time when users retrieve annotations in the PA system. It allows leaving more than one annotation for a target, even if they are of the same annotation type. The PA administrator doesn't make any retrictions when leaving annotations on targets. Therefore, there could be many conflicts and interferences between annotations. The DCD approach aims to only check and solve the problem at the time of retrieving annotations by considering the current context of users and targets. So, each time users retrieve annotations, the system will first check if there is a conflict in the current context and then check the specifications for resolving it. This technique could consume some time and resources more than the ICP technique. However, many benefits will be available to the PA system such as having many annotations of one type which could work in different contexts, and allowing different ways of dealing with conflicting annotations (improving expressiveness).

For example, consider a scenario of having a meeting room in a university, and there are two annotations of type "meeting" for this room. The first one says "each Monday there is PhD students' meeting" so the context time is Monday from 1pm to 2pm. Another annotation says "the staff meeting is first day of each month". So the context date is first of each month from 12pm to 2pm.

The PA system allows having more than one annotation of the same annotation type for the meeting room. However, the conflict can happen when as specified in the room's target constraint that there should normally be only one annotation of that type; i.e., if one Monday comes on the first day of a month, this leads to having annotations saying this room is booked for PhD students and also booked for the department staff. At this stage the DCD approach will detect this conflict and then solve it based on the policy as explained in the next section.

B) Policy for resolution

In this section, we propose and discuss some strategies for resolving and handling the problem of detecting a conflict in a PA system.

The strategies below start from the more important one to the less important, so that the PA system must apply the first strategy to resolve the problem, but if unsuccessful (e.g., both targets and annotations have the same privileges for the first strategy), then the PA system should move on to the next one and so on. The strategies are as follows (given in order but the PA system administrator may change their order as he/she think it more suitable for the environment).

1) Privilege and higher role: PA systems allow having different types of power and privilege for annotations' authors so that when there is a conflict, the higher power and privilege author's annotation will get a higher priority and will be the only one valid during the conflict. As an example of different author privileges, consider a spot in Melbourne CBD, there are many annotations for this spot for different annotation types such as tourism, education and so on. Also, for tourism annotations, there are many annotations, some were created by the city council, others by "normal" people (but assuming all in the same user group). The council author has more privileges in the CBD than "normal" people, so that when the PA system detects a conflict, the annotations which were created by the council will get a higher priority than others. Also, in the same example, if police officers create annotations in the CBD, their annotations will get a higher priority than others even the city council. This strategy is also useful in an emergency situation, so that police officers may annotate the spot to be evacuated, and their annotation will override all other annotations and be the only ones available to PA users.

2) Ownership of target: the second strategy allows the PA system to give more advantage to the owner of targets over other users. So, if the first strategy hasn't resolved the interference or the conflict, the PA system will check the target's ownership. An example of this is when there are many annotations created by different users of type education, the target here is a movie poster in a cinema. When a conflict is detected, the PA system will check the owner of that poster, and give the owner's annotations a higher priority than others users' annotation (in this case the cinema staff is the owner).

3) Last-come first-adapt: as the name implies, this strategy gives the latest annotations a higher priority than older annotations. There are many cases when this strategy is the best solution when discovering interference or a conflict. An example of using this strategy is as follows: in a shopping store, assume there is a product "tablet note2", the first commercial annotation ann1 says "there is 10% discount during December"; however, on Boxing Day which is 26th of December, there is a new annotation ann2 about a further discount saying "15% off the price". As we can notice on the 26th of December there will be a conflict between the old annotation and the new one, hence, to avoid the conflict during the run-time, on 26th of December, the PA system will make the last annotation ann2 the only valid annotation on 26th of December without deleting the old annotation ann1, it will be hidden on that date only and then after the ann2 expires, ann1 will take higher priority and be active again.

4) Parent annotation: in our location model, DPVLW-model we allow the PA system to structure the annotated entities in a hierarchal structure, this allows the users to retrieve annotations for the parent nodes, such as a product in a section in a shopping store. In some cases there could be interference or a conflict between parents' nodes and children nodes, so this strategy simply gives the parent's annotations a higher priority than the children nodes. An example of this situation is as follows: assume if there is a Sony TV in the TV section in the DickSmith store, there is an annotation for the Sony TV saying "10% off the price", and at the same time, there is another annotation for the whole TV section saying "15% off the price". As a result, this strategy will avoid the child node's annotation and give priority to the section's annotation, so the discount will be 15% only.

5) Child annotation: this strategy is the opposite of parent annotation strategy, so the child node will get higher priority than the parent nodes in the DPVLW-model. Assume a user in a DVD store, and there is an annotation saying "discount 10% on all DVD". However, if the user is in kids' section, which is considered a child node of the whole store, with annotation saying "20% discount". Then, the child node's annotations will take priority over the parent's annotations.

6) Priority of the annotation (same user but in some context one of the annotations will be better): an author may make more than one annotation of the same annotation type for the same target. The author may give different importance levels for each annotation such as "urgent", "important", "normal", and "less important". So when there is a conflict, the more important annotation will be the only valid one. For example, assume a user annotates his/her desk with two annotations (defined to be of the same type listed in the target constraint), the first one saying "call your friends each Monday to check how is he/she doing" where the importance of the annotation is "normal" and another annotation says "the manager needs the financial report this Monday" where the importance of the annotation is "urgent". So, both annotations apply at the same time and causes a conflict, the urgent annotation will become the only one available.

7) Preferences priority (user's precedence): this strategy gives PA users the chance to prefer any type of annotations or authors, so that when an interference or a conflict occurs, this strategy will refer to his/her preferences, i.e., the priority is simply specified explicitly by the user.

After the PA system detects a conflict or interference and after referring to the policies, the system will deal with each one of the interference or the conflict individually. So, in the case of a conflict, only one annotation should be valid. In the interference case, multiple annotations could be retrieved because they could all make a consistent statement together; we can express this outcome as the following. Here are three outcomes from resolving PA interferences and conflicts, represented as the function Comp (note that the binary function can be repeatedly applied for more than two in conflict or interference):

- For interference, we have: Comp(pa1, pa2) = pa1+pa2, which means the contents of both annotations can be retrieved together at the same time (e.g., they are simply all displayed).
- For conflicts, we can have:
  - Comp(pa1, pa2) = pa1 or pa2, i.e., they are mutually exclusive, so we can have only one, as determined by the strategies above.
  - Comp(pa1, pa2) = pa1, where pa1 overrides pa2 at all times (as explicitly specified by the user), regardless of context dependencies.

VI. Implementation

In this section we describe our prototype PA system with ability to detect and resolve interference. CIRPA is an extension of our PA system called ALPHAsys [3].

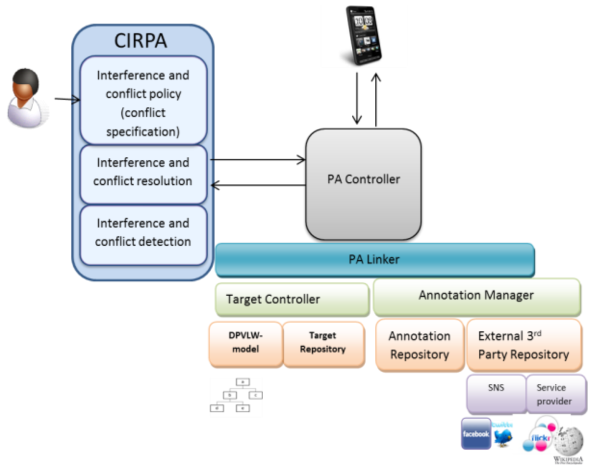
CIRPA
Interference and conflict policy (conflict specification)
Interference and conflict resolution
Interference and conflict detection
PA Controller
PA Linker
Target Controller
Annotation Manager
DPVLW-model
Target Repository
Annotation Repository
External 3rd Party Repository
SNS
Service provider
facebook
flickr

Figure 4:High level architecture for ALPHAsys & the CIRPA component

A) ALPHAsys

ALPHAsys is a Mobile Context-Aware System for Physical Annotations in the Physical World Model. As shown in Figure 4, the ALPHAsys architecture was designed based on our formal model, ALPHA, which has three main parts; annotation, target and the PA linker that link annotations to targets, basically enacting the physical annotations. Figure 5 shows in detail the process of creating and retrieving annotations and how components interact with each other. The main parts in the system are as follows.

The first part is the Annotation Manager; this layer contains the annotation content, stored in two repositories: Annotation Repository, which contains the system's own annotations written by users, and External 3rd party annotation repositories, where the annotation content can be also provided by third party providers, such as social network services. For example, Facebook can provide some useful annotations as well as Wikipedia, so this layer refers to such information services.

The second part is the target part which includes: *(1) Target Repository:* this contains information about the annotated entity, whether it is a small object, location or person. The repository includes also the entity's target type which will help to determine the possible associated annotations of this object. *(2) DPVW-model repository:* stores the DPVW-model (which includes the location model of spatial targets) and tracks the targets within the DPVW-model, e.g., their locations in the location model (e.g., within a room on a floor in a building), giving the user the option to browse the DPVW-model to retrieve the parent node's annotations as well. For example, if the user is in room A on the 2nd floor in building B, we can present this location model as a spatial tree to the user. The user may then like to retrieve the annotations for that room and the floor as well. So this will locate the entity in the location model hierarchy and give the user the option to retrieve the annotations belonging to the space containing the location, not just the annotations for the spot/location.

The third part in ALPHAsys architecture, as in the formal model, is the link part, which includes the PA Linker: the role of this layer is to map annotations to entities/targets. This tier includes conditions of the linking, and uses the PA link properties such as context dependencies and mapping. It also manages annotations linked to collections and virtual groups.

B) CIRPA

The CIRPA is an extension of our previous ALPHAsys; it is the interference and conflict manager: detecting conflicts, checking policies, and resolving interferences and conflicts. All this process is managed by the *PA Controller:* this component controls the PA access; after the system retrieves the annotation from the lower tiers, it manages the interference/collision (if any), and all possible situations that may affect annotations. *CIRPA* contains three parts.

1)CIRPA detector manager: it is responsible for detecting the interference and conflict between targets and/or annotations. It will check if two targets overlap or if one target has many annotations that conflict with each other. In this section if there is no conflict, it will send an approval to the PA controller which then sends users the required annotations, but if a conflict is detected, it will refer to the *CIRPA resolution manager*.

2)CIRPA resolution manager: after detecting a conflict, this component will work to resolve it and avoid the conflict. It refers to the *CIRPA policies manager* to get instructions to deal with the conflict.

3)CIRPA policies manager: this component contains the strategies and policies that were specified by the system administrator in order to direct the *CIRPA resolution manager*. The strategies, as we discussed in section 4.2, are taken in order. We implemented our system by using an Android phone platform 4.0.

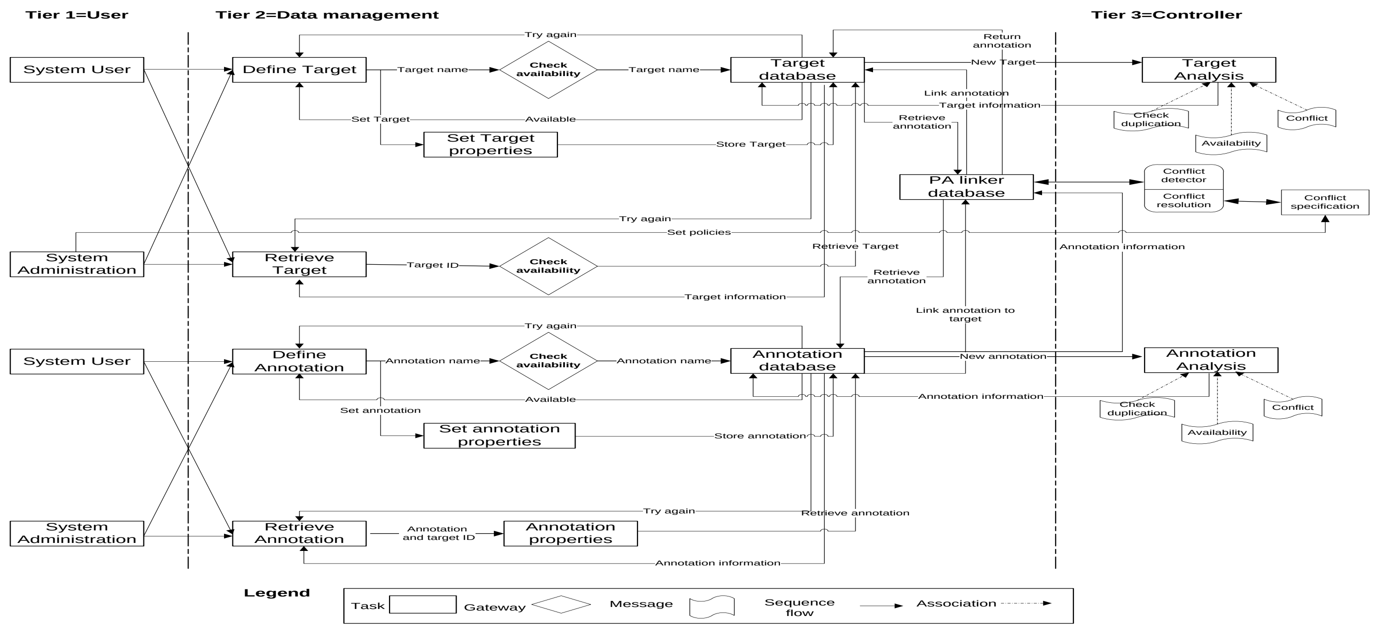

Figure 5: ALPHAsys architecture including CIRPA

VII. RELATED WORK

With the huge increase of using location based services and multiple user annotations, in general, conflict detection and resolution is one key point that challenges any system. Therefore, in the last decade, there is much research on conflict and resolution in pervasive computing environments, e.g., [4, 5]. Also [6-8] describes the hanging services framework that support ad hoc services.

There is much work on conflict techniques and policies [9-11]. All previous research are focused on pervasive computing in general. However, as the conflict and interference in PA systems are different from other pervasive systems, we found existing work not considering this problem. So, we presented our definitions and approach to cover interferences and conflict between annotations and/or targets in different contexts.

VIII. CONCLUSION AND FUTURE WORK.

In this paper, we have presented CIRPA, which is Conflict and Interference Resolution in Physical Annotation Systems. We first explain how a conflict and interference can occur when users create or retrieve annotations. We found that the interference between annotated targets could result in this problem. We also found that conflict can happen for one target when it has many annotations with the same type and context.

Then, we discussed two approaches to handle the conflict and interference in a PA system, which are proactive (or preventative) and dynamic conflict handling. Then, we provided strategies with examples which aim to enable PA systems to avoid and resolve conflicts and interference between annotations and targets.

While our approach has been presented pairwise mostly, n-way conflicts are resolved in pairs in a pairwise manner.

We also note that the above notions of interference and conflict provide a means, effectively, to filter annotations shown to the users.

We also note that other current augmented reality or physical annotation systems do not fully exploit a real-world model to position the annotations, i.e. using what we have called DPVW-models. We note that over the same physical environment, it is possible to build different DPVW-models, e.g., for a shopping street, we have a set of DPVW-models for tourists, another set for the police, and another set for local residents.

Our formal model aims to provide a precise notion of physical annotations (really as links between annotations and physical targets), and our formal notions of conflict and interference are just one conceptualization – there could be others. But we find this current conceptualization useful in resolving interfering or conflicting annotations without having to deal with understanding the actual semantics of the contents of annotations (which is generally difficult). Future work could look into Natural Language Processing and semantics analysis as another means to determine conflicting or interfering annotations (though this would be computationally very intensive).

Finally, we outlined our implementation and architecture of CIRPA within ALPHAsys. Although these techniques were designed for our PA system, it can be used for any PA system or mixed reality system that is used to annotate physical entities such as Wikitude, Junaio or Layar.

Future work will involve testing our approach with real users and evaluating the usefulness of our strategies in a larger application context, e.g., on a university campus, a shopping mall and in a shopping street.

We will also study the efficiency of our approach, with our evaluation and note any performance concerns.

REFERENCES

1. Alzahrani, A.A., S.W. Loke, and H. Lu. A Formal Model for Advanced Physical Annotations. in 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC),. 2011. Sydney, Australia.
2. Alzahrani, A.A., S.W. Loke, and H. Lu, A survey on internet-enabled physical annotation systems. International Journal of Pervasive Computing and Communications, 2011. 7(4): p. 293-315.
3. Alzahrani, A.A., S.W. Loke, and H. Lu, ALPHAsys: an Advanced Location-aware PHysical Annotation system from models to implementation. submited to a journal and still under reviw, 2013.
4. Tuttlies, V., G. Schiele, and C. Becker, COMITY - Conflict Avoidance in Pervasive Computing Environments On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, R. Meersman, Z. Tari, and P. Herrero, Editors. 2007, Springer Berlin / Heidelberg. p. 763-772.
5. Tuttlies, V., G. Schiele, and C. Becker. End-User Configuration for Pervasive Computing Environments. in Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on. 2009.
6. Syukur, E., S. Loke, and P. Stanski, A Policy Based Framework for Context Aware Ubiquitous Services, in Embedded and Ubiquitous Computing, L. Yang, et al., Editors. 2004, Springer Berlin Heidelberg. p. 346-355.
7. Syukur, E., S.W. Loke, and P. Stanski, Methods for Policy Conflict Detection and Resolution in Pervasive Computing Environments, in In: Policy Management for the Web Workshop in conjunction with the 14th International World Wide Web Conference2005: Chiba, Japan.
8. Jakob, H., C. Consel, and N. Loriant, Architecturing Conflict Handling of Pervasive Computing Resources, in Distributed Applications and Interoperable Systems, P. Felber and R. Rouvoy, Editors. 2011, Springer Berlin Heidelberg. p. 92-105.
9. Chae, H., T. Kim, D.-h. Lee, and H.P. In. Conflict Resolution Model Based on Weight in Situation Aware Collaboration System. in Future Trends of Distributed Computing Systems, 2007. FTDCS '07. 11th IEEE International Workshop on. 2007.
10. Dunlop, N., J. Indulska, and K. Raymond. Methods for conflict resolution in policy-based management systems. in Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International. 2003.
11. Insuk, P., L. Kyungmin, L. Dongman, J.H. Soon, and Y. Hee Yong. A Dynamic Context Conflict Resolution Scheme for Group-aware Ubiquitous Computing Environments. in Proceedings of the 1st International Workshop on Personalized Context Modeling and Management for UbiComp Applications (ubiPCMM'05). 2005.