Assistant Agents to Advice Users in Hybrid Structured 3D Virtual Environments[*]

Pablo Almajano
IIIA - CSIC, University of Barcelona
palmajano@iiia.csic.es

Maite Lopez-Sanchez
University of Barcelona
maite@maia.ub.es

Inmaculada Rodriguez
University of Barcelona
inma@maia.ub.es

Tomas Trescak
University of Western Sydney
t.trescak@uws.edu.au

Abstract

Hybrid structured 3D Virtual Environments model serious activities in immersive 3D spaces, where participants are human and SW agents, and their interactions are regulated by an OCMAS (Organization Centered Multi-Agent System). In this context, both OCMAS social model and the tasks that users need to accomplish can be rather complex, and thus, users may benefit from having an assistance service. Hence, we propose Personal Assistant agents (PA) which, based on knowledge about the OCMAS specification and current system state, provide the user with an advice (a plan) to achieve her goal. Additionally, we implement this service with PLAN-EA, an Extension of the A^* algorithm that generates plans for a user whose actions may depend on other users' actions. Thus, PAs provide plans that do not only include assisted user actions but other users' ones. We illustrate our approach by means of v-mWater –an online water market– and make a comparative analysis, with and without assistance, where efficiency –in terms of number of user actions– shows an improvement (7 vs 10.8), efficacy –percentage of completed tasks– also improves (93% vs 77%), and assistance's overall satisfaction is positive. **Keywords:** Personal Assistant Agents, Social Model, Structured 3D Virtual Environment, OCMAS Planning

[*]Work funded by Spanish CSD2007-0022, TIN2011-24220 and TIN2012-38876-C02-02 research projects.

1 Introduction

The blending of digital technologies, such as artificial intelligence, interactive systems, 3D interfaces and the Internet, is transforming the way people interact with computers. This blending is enabling new services for users, favoured by huge advances in the development of graphics and networking.

In particular, multi user online 3D Virtual Environments (VE) provide users with a collaborative space not only for entertainment and socialization but also for developing "serious" activities like learning, negotiating and shopping.

Most often, these serious activities require the execution of complex tasks, involving different subtasks, and highly regulated interactions. Nevertheless, standard 3D VE lack of mechanisms to structure (regulate) users' interactions. An OCMAS (Organization Centered Multi-Agent System) [1] constitutes a powerful tool that can serve this purpose. Specifically, the marriage of 3D VE and OCMAS allow the participation of both human and software agents in a so-called hybrid structured system. However, its intrinsic complexity constitutes a serious obstacle for the participant system comprehension, which is seldom aware of all its runtime property values.

In this context of hybrid structured VEs, we propose Personal Assistant agents (PA) which, based on knowledge about the OCMAS specification and current state, support participants to understand the social model underlying such a specification and to perform complex tasks in a seamless way. We describe the overall system as a Hybrid Assisted Structured 3D Virtual Environment.

Personal Assistants provide different general Assistance Services. In a previous work, we presented an *Information Service* for SW agents [2]. This paper focuses on the formalisation, operationalisation, and evaluation of the *Advice Service* for human users. Through this advice service, a PA provides the user with a plan (i.e., a set of actions compatible with the OCMAS specification) to accomplish her goal (task). PA embodiment is an "angel" which may accompany the user during the interactive experience and is available under request.

We illustrate and evaluate our approach by means of v-mWater, a regulated virtual environment for the trading of water rights. Our evaluation results assess that, whenever requested, this *Advice Service* provides comfortable and clear guidance that facilitates task accomplishment. In fact, its usage reduces both the percentage of task failures and the number of user performed actions, when compared to a previous system we evaluated without assistance [3]. Assuming we think in the same type of actions (i.e. the ones defined in the system specification), we consider the number of actions an efficiency measure: the less the number of actions to do the task, the more directly users reach the goal. Smaller number of actions also implies less cognitive load on the user and a reduction of the task perceived complexity.

2 Related Work

A number of works in the literature focus on (intelligent) virtual agents providing users with assistance. In the line of recommender systems, Dong et al. [4] propose a Reviewer's Assistant to give a product reviewer optional suggestions. This assistant is a web browser plug-in that employs different data mining strategies. Another recommender system [5] uses a Multi Agent System (MAS) populated by Personal Assistant (PA) agents. A rule driven PA provides the user with talks announcements. To do so, it exploits semantic web data and the user profile. In these works, assistants serve one (assisted) single-user not involved in any organization. As a remarkable difference, our PAs operate in a multi-user and structured social scenario, characterised by complex interleaved interactions.

Lujak et al. [6] develop EMA, a medical emergencies scenario modelled as an OCMAS which employs web services and a 2D User Interface (UI) to assist participants (e.g. patients, ambulances, medical professionals) in their coordination. RADAR [7] is another MAS with a Multi Task Coordination Assistant that observes experts and learns models to assist office workers cope with email overload. Its advice includes task ordering suggestions and warnings when the user's behaviour differs significantly from the expert's one. They propose a 2D task-management UI. Electric Elves [8] also developed specific Software Personal Assistants (*SPA*) but they were for project activities coordination and external meetings organisation in a real environment. Unlike our shared and collaborative 3D space with embodied Personal Assistants, these works feature 2D user interfaces where the PAs are non-embodied.

A recent work has presented a coalition planning assistant agent in a peacekeeping problem domain [9]. In this scenario the user moves in a $n \times n$ grid towards a predefined subset of goal cells, and norms represent prohibitions and obligations on visiting a cell. The assistant agent is able to discover the particular user's goal and execute actions (e.g. send escort request) to avoid user norm violations. As a difference, our PA provides users with a plan (actions) to achieve an indicated goal. Moreover, their agents execute independent actions to ensure norm compliance of user movements. Finally, assistance has been used to support users' physical navigation in 3D Virtual Environments, such as the Virtual Theatre [10], where a non-embodied agent uses environment knowledge to indicate the user places to visit; and the Virtual Museum [11], where a 3D character presents the user a precomputed guided tour to follow. Our PA is an interactive 3D character that is always available to the user and helps her/him to achieve a given task.

3 Personal Assistant

This section first introduces the knowledge that a Personal Assistant agent needs to provide an advice (plan) to the human user. Next, it describes the operationalisation of the *Advice Service*. It illustrates them with v-mWater, an example scenario in the agriculture domain that models an electronic market for trading water rights (i.e., the right to use water for irrigation).

3.1 Agent Knowledge: Social Model

Our structured 3D VE integrates a 3D Virtual World (VW) and an OCMAS. On the one hand, the 3D interface facilitates the interaction of human users. On the other hand, the OCMAS infrastructure (an Electronic Institution [12]) regulates real-time participants' interactions, where participants can be both human and software agents. A Personal Assistant creates a plan for the user employing, as knowledge, the *static* specification of the OCMAS social model (*Spec*) and the system current (*dynamic*) state (S_c). Particularly, it considers the Social Structure and Social Conventions ($SocStr$, $SocConv \in Spec$) and their runtime values ($RtSocStr$, $RtSocConv \in S_c$).

Social Structure ($SocStr$) defines the properties associated to user agents (AgP), the roles (Rol) that participants enact, and their relationships (Rel). Fig. 1 depicts an extract of v-mWater specification concerning seller participants. Its $SocStr$ (at the top) specifies two participant properties –their *location* (loc) and *goal*– and two participant roles — seller (*s*) and market facilitator (*mf*).

$$SocConv = \langle Activ, ActivRel, Prot\rangle \quad (1)$$
$$ActivRel = \langle Tra, Mov\rangle \quad (2)$$
$$mov = \langle mRol, ori, des, mT\rangle \quad (3)$$
$$prot = \langle ProtP, ProtSpec\rangle \quad (4)$$
$$ProtSpec = \langle ProtC, Nod, Illo\rangle \quad (5)$$
$$ProtC = \{\langle rolC, min, max\rangle\} \quad (6)$$
$$nod = \langle EnterRol, ExitRol\rangle \quad (7)$$
$$illo = \langle sRol, rRol, cM, ori, des\rangle \quad (8)$$

Social Conventions (see Eq. 1) stand for the "rules of the game". Participants (roles) gather in activities ($Activ$). There, they can perform tasks by following protocols ($Prot$) or move to other related activities ($ActivRel$). In our example of Fig. 1, seller participants can: enter/leave the system in the *Initial/Final* activity; ask for market information in the *Waiting&Info* activity; and register a water right in the *Registration*. Activities relationships ($ActivRel$) in Eq. 2 constraint the flow of roles among activities. Transitions (Tra) are intermediate locations meant for user synchronization. Movements ($mov \in Mov$, Eq. 3) are participant actions that change their location. They are defined by: i) its role ($mRol$); ii) the transition and activity it connects ($ori, des \in \{Tra \cup Activ\}$, ori being origin and des destination); and iii) its type ($mT \in \{$"new", "$enter$", "$exit$"$\}$, meaning create and enter a *new* activity, *enter*, and *exit* an existing activity). Fig. 1 represents a directed graph where nodes correspond to both activities (rectangles) and transitions (diamonds) and edges (labelled arrows) represent allowed movements between them.

Protocols structure user interactions within activities. Eq. 4 defines a protocol ($prot \in Prot$) as a set of properties ($ProtP$) and its specification ($ProtSpec$). $ProtSpec$ (Eq. 5) is a finite state machine (FSM), where nodes (Nod) correspond to the states and illocutions ($Illo$) to state transitions. Fig. 1 depicts protocols' nodes as circles and illocutions as labelled arrows. Those nodes where the protocol execution starts and ends are named *initial* and *final* respectively. Moreover, the protocol capacity ($ProtC$ in Eq. 6) defines a set of tuples that bound the number of participants that enact each role ($rolC$) : min sets the minimum number required to open the protocol, whereas max sets the maximum number of allowed participants in this protocol. We can also specify, for each protocol node ($nod \in Nod$) in Eq. 7, the list of roles that can join ($EnterRol$) and leave ($ExitRol$) the activity when it is at this protocol state. For example, in the *Registration* activity of Fig. 1, users playing seller role are allowed to enter (+*s*) and exit (-*s*) whenever its protocol state is node *n1*.

Finally, illocutions are messages that participants interchange. They act as transitions of the

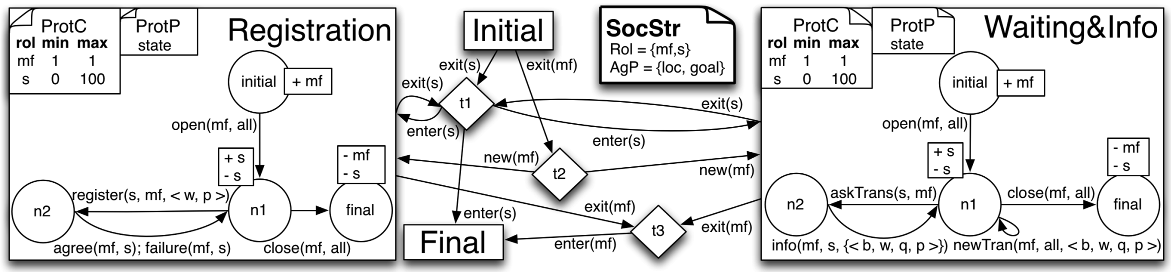


Figure 1: Extract of the static specification of v-mWater social model.

FSM, so they imply a change in the protocol execution state (node). Thus, for instance, any illocution uttered at an *initial* node opens the protocol. An illocution ($illo$ in Eq. 8) formalises a participant action by specifying: i) the role of the sender ($sRol$) or *all*, in case anybody can send it; ii) the role of the receiver ($rRol$) or *all*, if it is public and everybody will receive it; iii) the message content (cM), conforming an ontology; iv) the origin node (ori) where the message can be sent; and v) the destination node (des), the node reached once the illocution is successfully uttered. In Fig. 1, illocution *register(s, mf, ⟨ w, p ⟩)* in *Registration* specifies the request from a seller (*s*) to a market facilitator (*mf*) the registration of a water right (*w*) to a given price (*p*). It can be uttered when the protocol's state is node *n1* and, once performed, the protocol state will transit to *n2*.

Execution State (S_c) contains current runtime values of dynamic organisational elements, such as actual participants ($RtAg \in RtSocStr$) and running activities ($RtActiv \in RtSocConv$).

3.2 *Advice Service: Operationalisation*

Activities in the social model specify user interactions so that certain tasks can be performed (e.g., users can register water rights in the *Registration* activity). Whenever a user aims to perform one of such tasks, s/he can request an advice service to her/his Personal Assistant (PA). This PA will then respond with a plan ($pl = \{a_1, \ldots, a_m\}$) consisting of a sequence of m actions (movements and illocutions). The provided advice (plan) will conform to the social model specification and, if executed at the current system state, will lead the user to her/his goal of performing the task.

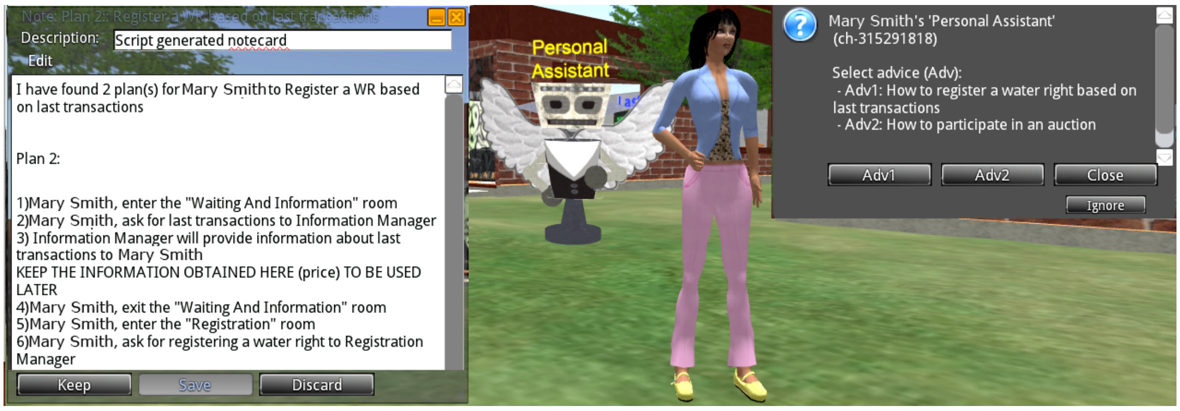
Eq. 9 formalises this advice service. When a user request (Req) is received, the PA uses social model specification ($Spec$) and current state (S_c) to provide a response (Res). Both Req and Res are messages which contain a sender, a receiver, and a content. In the former (see Eq. 10), the user asks for PA's advice to perform a $task$. In the latter (see Eq. 11), the PA answers the user with a plan (pl).

$$Advice : Req \times Spec \times S_c \rightarrow Res \quad (9)$$
$$Req = \langle rtAg, PA, task \rangle \quad (10)$$
$$Res = \langle PA, rtAg, pl \rangle \quad (11)$$

A PA is graphically represented as an Angel bot, an interactive non-player 3D character. Fig. 2 shows a snapshot of our 3D UI, with user Mary (female user avatar) and her PA (angel bot). PA always accompanies the user during the interactive experience so it is always available for her. Nevertheless, she can activate (show) her PA, interact with it whenever she needs help or deactivate (hide) it otherwise.

User-PA interaction has been implemented in the following way: i) A user performs the action *touch* to her/his PA in the 3D VW to start a help request. Then, the PA shows an *option dialogue* (on the right of Fig. 2) where the user can *select* one of the available Advice Services. ii) The PA sends a plan to the user as a *note card* (on the left of Fig. 2), which is a document where the plan, initially generated using a notation only readable by SW agents, is written in natural English language to facilitate user comprehension.

Based on the social model specification, the designer identifies those complex tasks that will be automatically assisted. Particularly, in v-mWater PAs help i) sellers to ask for information about market prices and to register water rights; and ii) buyers to participate in auctions.


Note: Plan 2:: Register a WR based on last transactions

Description: Script generated notecard

I have found 2 plan(s) for Mary Smith to Register a WR based on last transactions

Plan 2:

1)Mary Smith, enter the "Waiting And Information" room
2)Mary Smith, ask for last transactions to Information Manager
3) Information Manager will provide information about last transactions to Mary Smith
KEEP THE INFORMATION OBTAINED HERE (price) TO BE USED LATER
4)Mary Smith, exit the "Waiting And Information" room
5)Mary Smith, enter the "Registration" room
6)Mary Smith, ask for registering a water right to Registration Manager

Mary Smith's 'Personal Assistant' (ch-315291818)

Select advice (Adv):
- Adv1: How to register a water right based on last transactions
- Adv2: How to participate in an auction

Figure 2: Mary's avatar with her Personal Assistant in the 3D VW

Next section details the planning process, which is general for any social model specification.

4 OCMAS Planning

As previously stated, Personal Assistants (PA) compute plans based on both the static OCMAS social model specification ($Spec$) and current (dynamic) system state (S_c). This knowledge allows to explore the search space by expanding a directed planning tree to compute the path towards a goal state (i.e., the one reached once a user accomplishes a task). Nodes of the planning tree represent different system states whereas edges correspond to possible participant actions. The goal is expressed in terms of desired values for state runtime properties (e.g. the location of the user). Notice though, that, since we consider a multi-user scenario with action dependencies, we need to reckon with different plans executed by different participants.

Specifically, a PA computes a plan for a participant by invoking the recursive function PLAN-EA$(Spec, rtAg, PlS_c)$ with parameters: i) its corresponding run-time agent ($rtAg$), including its goal ($rtAg.goal$); ii) the social model specification ($Spec$); and iii) current planning state (PlS_c), a working copy of S_c. Considering our multi-user scenario, we propose to implement PLAN-EA as an Extension of A^* [13] (with an admissible heuristic) that handles action dependencies by providing plans that are extended with other users' plans. By action dependencies we mean actions whose success depend on other users's actions. For example, Mary needs an activity to be open before she can move into it or she cannot utter an illocution if there are no receivers. Thus, plans for opening activities or including receivers are associated to her plan, so she will be aware that she has to wait for other participants to execute these plans before she can successfully perform her planned action.

Algorithm 1 ASENT$(Spec, rol, rtActiv, PlS_c)$

```
$TmpPls \leftarrow \emptyset$
$RtAg \leftarrow$AGROLNACT$(Spec, rol, rtActiv, PlS_c)$
**for all** $rtAg \in RtAg$ **do**
    $rtAg.goal \leftarrow \{rtAg.loc == rtActiv\}$
    $pl \leftarrow$ PLAN-EA$(Spec, rtAg, PlS_c)$
    $TmpPls \leftarrow TmpPls \cup \{pl\}$
**end for**
**return** MINCOST$(TmpPls)$
```

On the one hand, to utter an illocution in an activity $rtActiv$ requires a sender, a receiver and, if applicable, the min number of participants from the protocol capacity (see Eq. 6). For each missing participant at $rtActiv$, we call ASENT$(Spec, rol, rtActiv, PlS_c)$ in Alg. 1, which returns an associated plan for entering the activity. It uses function AGROLNACT to select participants located outside $rtActiv$ and enacting role rol. Then, for each selected $rtAg$, it computes a plan to join $rtActiv$ by setting its $goal$ to be this location. ASENT returns the plan having the minimum cost.

On the other hand, to enter an activity $activ$ requires the activity to be: (1) created and (2) opened to the participant agent. Regarding (1), ASCREATE$(Spec, activ, PlS_c)$ in Alg. 2 computes the associated plan for creating $activ$. It first invokes CANCREATE to select agents ($RtAg$) currently enacting a creator role for

Algorithm 2 ASCREATE($Spec, activ, PlS_c$)

```
$TmpPls \leftarrow \emptyset$
$RtAg \leftarrow$ CANCREATE($Spec, activ, PlS_c$)
for all $rtAg \in RtAg$ do
    $rtAg.goal \leftarrow \{rtAg.loc == new(activ)\}$
    $pl \leftarrow$ PLAN-EA($Spec, rtAg, PlS_c$)
    $TmpPls \leftarrow TmpPls \cup \{pl\}$
end for
return MINCOST($TmpPls$)
```

Algorithm 3 ASOPEN($Spec, rol, rtActiv, PlS_c$)

```
$TmpPls \leftarrow \emptyset$
$Nod \leftarrow$ ENTERNODS($Spec, rtActiv, rol$)
$RtAg \leftarrow$AGACT($rtActiv, PlS_c$)
for all $nod \in Nod$ do
    for all $rtAg \in RtAg$ do
        $rtAg.goal \leftarrow \{rtActiv.state == nod\}$
        $pl \leftarrow$ PLAN-EA($Spec, rtAg, PlS_c$)
        $TmpPls \leftarrow TmpPls \cup \{pl\}$
    end for
end for
return MINCOST($TmpPls$)
```

$activ$. Second, for each $rtAg$, it sets the creation goal and calls PLAN-EA to have its plan. Third, it returns the minimum cost plan. As for (2), ASOPEN($Spec, rol, rtActiv, PlS_c$) in Alg. 3 calls: ENTERNODS to get all nodes (Nod) where role rol can enter; and AGACT to get all activity participants ($RtAg$). Then, it computes the minimum cost plan for changing the state of $rtActiv$ to be a node in Nod.

If associated plans were not found, the action a would be discarded for expansion. Otherwise, the successor function will return a planning state (PlS'_c) resulting from executing the associated plans at PlS_c. Notice that order, which is related to action dependencies, matter. Overall, the successor function of a state PlS_c returns a set of tuples, where each tuple is composed of: a possibly empty ordered set of associated plans; a *possible action*; and the resulting state. PLAN-EA then chooses to expand the successor node having minimum cost f, which is the sum of: i) the past path-cost g, computed as the number of actions to reach node PlS_s from the root node; and ii) an *admissible heuristic* $h(rtAg, PlS_s)$, which computes a lower bound of the actions required to reach the goal from PlS_s. It does so by relaxing the constraints imposed by the social model specification and runtime properties. Specifically, h only considers a subset of the action dependencies that are taken into account in the actual planning process. Thus, as we briefly explain below, most action dependencies handled by algorithms 1, 2, and 3 are also considered by h. Nevertheless, it is done in a far less costly way: applying admissible "rules of thumb" and without invoking PLAN-EA.

Initially, h checks whether the $rtAg$ is located at the same activity than the goal (so it can be reached by uttering illocutions). If it is the case, h selects from $Spec$ and PlS_s, all illocutions that are related to $rtAg$'s role in this activity. For each illocution, h aggregates the cost of uttering it (1) and, if applicable, a lower bound of the cost of adding required participants to the activity. Otherwise, if $rtAg$ is not located at the goal location, then $rtAg$ should: i) exit its current location; and ii) enter the goal location. h estimates the cost for both movements, considering also if current protocol states allow the participant to exit/enter the corresponding activities. Additionally, h checks if the action execution will reach the goal, since we can consider that, if it is not the case, at least an additional action will be required.

Finally, as for standard A^*, the search will continue until the goal is reached or no more nodes can be expanded. If found, PLAN-EA will return the plan (i.e., a sequence of actions from the root to the goal).

4.1 v-mWater planning example

Fig. 3 illustrates a planning process PLAN-EA($Spec, Mary, PlS_c$) that considers the static specification ($Spec$) from Fig. 1 and the runtime properties (S_c) depicted at the top of Fig. 3. In particular: Mary enacts a *seller* role, is located at transition $t1$, and her goal is to *register* a water right; the *Waiting&Info* activity is open ($n1$ in $Spec$); and *Registration* is not created. The root node PlS_0 (in solid black) is initialised to S_c and the successor function considers two seller actions for Mary. First, $a1$ =*enter(s, t1, Waiting&Info)* will directly lead to state PlS_1, because the current state of the activity protocol is $n1$, and sellers can enter at such node. Second, $a2$ =*enter(s, t1, Registration)* cannot be performed, since Mary needs to wait for a market facilitator to create and open it. As a conse-

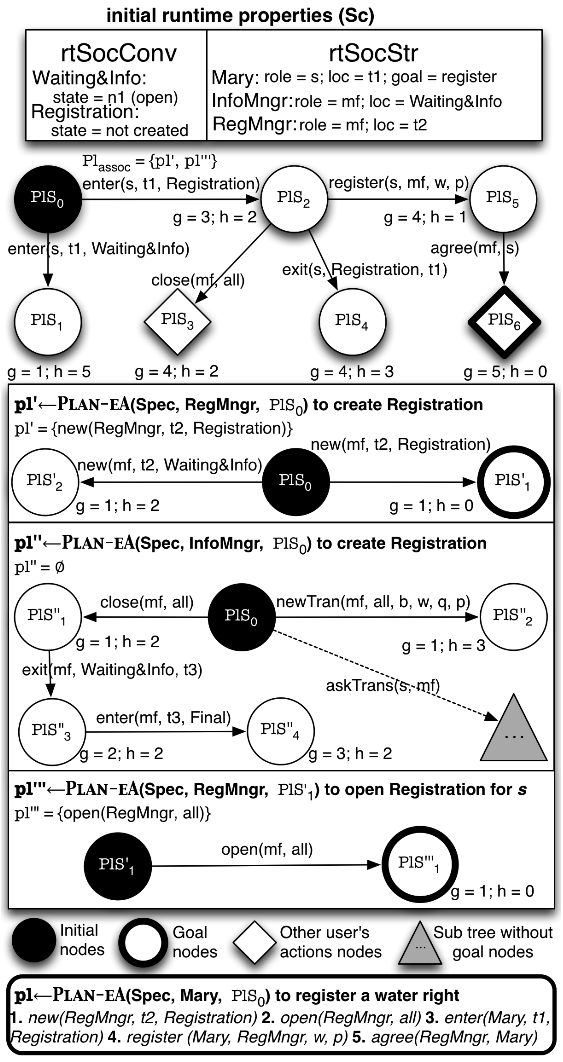

initial runtime properties (Sc)
rtSocConv
Waiting&Info: state = n1 (open)
Registration: state = not created
rtSocStr
Mary: role = s; loc = t1; goal = register
InfoMngr: role = mf; loc = Waiting&Info
RegMngr: role = mf; loc = t2
Pl_assoc = {pl', pl'''}
PlS0
enter(s, t1, Registration)
g = 3; h = 2
PlS2
register(s, mf, w, p)
g = 4; h = 1
PlS5
enter(s, t1, Waiting&Info)
close(mf, all)
exit(s, Registration, t1)
agree(mf, s)
PlS1
PlS3
PlS4
PlS6
g = 1; h = 5
g = 4; h = 2
g = 4; h = 3
g = 5; h = 0
pl'←PLAN-EA(Spec, RegMngr, PlS0) to create Registration
pl' = {new(RegMngr, t2, Registration)}
new(mf, t2, Waiting&Info)
new(mf, t2, Registration)
PlS'2
PlS'1
g = 1; h = 2
g = 1; h = 0
pl''←PLAN-EA(Spec, InfoMngr, PlS0) to create Registration
pl'' = ∅
close(mf, all)
newTran(mf, all, b, w, q, p)
PlS''1
PlS''2
g = 1; h = 2
g = 1; h = 3
exit(mf, Waiting&Info, t3)
askTrans(s, mf)
enter(mf, t3, Final)
PlS''3
PlS''4
g = 2; h = 2
g = 3; h = 2
...
pl'''←PLAN-EA(Spec, RegMngr, PlS'1) to open Registration for s
pl''' = {open(RegMngr, all)}
PlS'1
open(mf, all)
PlS'''1
g = 1; h = 0
Initial nodes
Goal nodes
Other user's actions nodes
Sub tree without goal nodes
pl←PLAN-EA(Spec, Mary, PlS0) to register a water right
1. new(RegMngr, t2, Registration) 2. open(RegMngr, all) 3. enter(Mary, t1, Registration) 4. register (Mary, RegMngr, w, p) 5. agree(RegMngr, Mary)

Figure 3: Example of PLAN-EA returning plan pl with associated plans pl' and pl'''.

quence, function ASCREATE in Alg. 2 looks for plans for InfoMngr and RegMngr and returns pl' in Fig. 3 as the plan for creating the *Registration* activity. Afterwards, function ASOPEN in Alg. 3 finds plan pl''' for RegMngr to open the activity. Together, associated plans pl' and pl''' transit current state to PlS_1''', where $a2$ can now lead to successor state PlS_2. Hereafter, since PlS_2 has lower cost [1] ($f = 5$) than the one [2] of PlS_1 ($f = 6$), the planning expands PlS_2 and continues until it reaches the goal node PlS_6 (diamond shape indicates another participant, mf, did the action) and returns plan pl in Fig. 3.

4.2 *Plan Presentation to Users*

Computed plans are sequences of actions meant for software agents. Personal Assistants facilitate the interaction with human users by presenting them in natural language, so that human user Mary can easily interpret the plan in Fig. 2.

Action translation requires the designer to specify a template. For example, the register illocution is described as "\$sender\$, ask for registering a water right to \$receiver\$", where \$sender\$ and \$receiver\$ are substituted by the name of the actual participants, as last sentence in Fig. 2 reads. Moreover, if the receiver is the user, then a template is also used to generate sentences such as "Information Manager will provide information about last transactions to Mary Smith" (see third sentence in Fig. 2).

5 System Evaluation

We evaluated the assistance service [3] by i) fulfilling a usability test that follows the Formative Evaluation methodology, and ii) comparing the obtained results with our previous study [3], where the users performed the same task in v-mWater but without assistance.

5.1 *Test Goals and Research Questions*

We conducted a user evaluation whose goal was twofold. First, we aimed to evaluate our assistance in terms of its i) *effectiveness*, if it helps the users to actually perform the task; ii) *efficiency*, if it reduces the effort (in terms of number of user's actions and cognitive load) required to conduct the task; and iii) *users' satisfaction*: their opinions, feelings and experience. Second, we also aim at identifying the *errors/problems users make/encounter* when using the assistance.

[1] $g(PlS_2) = 3$ since the associated plans add two actions to the one performed by Mary. $h(PlS_2) = 2$ because, although Mary is at the Registration, none of her possible actions in PlS_2 do lead to the goal state.

[2] $g(PlS_1) = 1$ since only one action ($enter$) has been executed. $h(PlS_1) = 5$ because Mary is not where the goal task is performed. Thus, she has, at least, to *exit* Waiting&Info and *enter* Registration which must first be *created* and *opened*. Once inside the activity, at least one *illocution* should be uttered, since the goal is not just to enter the activity.

[3] We encourage the reader to watch *http://youtu.be/VOQ9DavaqNA*

Having these goals in mind, we addressed the following research questions: **RQ1:** *Assistance helpfulness.* At what stage of task completion was the help requested? Was the provided advice useful for the user to complete the task? **RQ2:** *User-Personal Assistant (PA) interaction.* Is the assistance easy to request? How easy and pleasant is the interaction with the PA? **RQ3:** *Plan tracking.* What obstacles do users encounter when following the plan? Is it clearly explained? Is it detailed enough to complete the task successfully and in a seamless way? **RQ4:** *Task completion.* How many users do complete the task? How do they perceive it?

For this usability test, users were asked to perform a rather complex task: to register a water right in v-mWater at a price that depends on previous market transactions. It implies 4 subtasks: i) to *understand the task* (they are required to visit 2 rooms in a specific order); ii) to *get particular information about the market prices* at the *Waiting&Info* room (this subtask can be accomplished by asking the Information Manager bot or by reading the information panel); iii) to *come up with the required registration price*, which has to be 5€ higher than the price of the most recent transaction; and iv) to *register the water right* by interacting with the Registration bot at the *Registration* room. Whenever needed, users can ask for assistance to their Personal Assistant.

5.2 Participants and methodology

We recruited 14 participants for our experiment. Table 1 shows details on their age, gender, computer skills ('basic' stands for users of limited computer functionalities and 'advanced' for computer professionals such as programmers) and VE experience ('none'/'high' describe users who have never/often used a VE).

Since we are mostly interested in finding relevant qualitative and quantitative data, our usability test is summative and follows the Formative Evaluation. The tests took place at users' locations: 30% of the participants did the test at their home and the rest at their workplace, on a separate room. The equipment consisted in 1 portable computer. It had the overall system installed: OCMAS execution infrastructure [12], OpenSimulator VW server, and a VW client.

Name	Age	Gender	PC exp	VE exp
P1	23	Female	Advanced	None
P2	24	Male	Advanced	High
P3	26	Female	Advanced	High
P4	27	Male	Advanced	High
P5	27	Male	Advanced	High
P6	27	Male	Advanced	High
P7	29	Female	Advanced	None
P8	32	Male	Basic	None
P9	32	Male	Basic	None
P10	32	Male	Advanced	High
P11	41	Male	Advanced	High
P12	42	Male	Basic	High
P13	53	Male	Advanced	None
P14	66	Female	Basic	None

Table 1: List of participants' characteristics

It also recorded user interactions and sound. All participants were requested to perform the aforementioned task by telling them: *"act as a seller, and register a water right for a price which is 5€ higher than the price of the last transaction done"*.

A moderator guided the test along four different phases: **1)** *Pre-test interview*: the moderator welcomed the user, briefly introduced the test and asked the user about her/his experience with similar VEs. **2)** *Training*: The moderator taught the user to move in a 3D demo VE as well as to interact with objects, avatars, bots (whose 'special' appearance, i.e. bold and coloured skin, was made noticeable) and her/his PA. This training part was mostly fully guided, except at the end, when the user could freely roam and interact in the demo scenario. **3)** *Test*: The user performed the assigned task without receiving any guidance (unless s/he ran out of resources). Meanwhile, the moderator encouraged the user to think-aloud (i.e., to describe her/his actions and thoughts) while performing the test. **4)** *Post-test satisfaction survey*: the moderator gave the user a survey with qualitative (open-ended) and quantitative (close-ended) questions regarding v-mWater and the assistance provided.

5.3 Results and discussion

In this section we show and discuss results obtained after we analysed data gathered during the test, i.e desktop and voice recordings, moderator notes, users' comments, and post-test satisfaction surveys.

Table 2 summarizes the 8 questions in the post-test survey and Fig. 4 depicts users' an-

Question	Brief description
Q1	Info gathering (panel/bot)
Q2	Human-bot interaction
Q3	Bot visual distinction
Q4	Dialogue-based bot communication
Q5	Overall system opinion
Q6	*Assistance usefulness*
Q7	*PA interaction*
Q8	*Advice Understanding*
open Q	User's comments

Table 2: Post test questionnaire

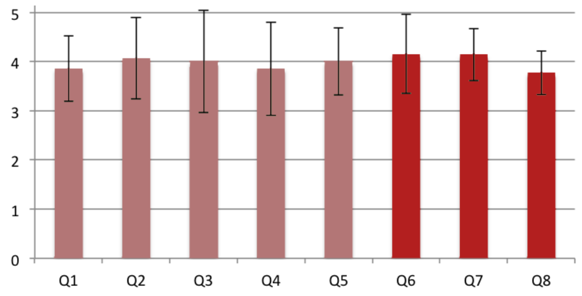


Figure 4: Post-test questionnaire average results and standard deviations.

swers. There, X axis shows questions and the Y axis shows average values of answers considering a 5-point Likert scale. This scale provides 5 different alternatives in terms of application successfulness ('very bad'/'bad'/'fair'/'good'/'very good'), where 'very bad' corresponds to 1, and 'very good' to 5.

We collected data on general issues about the 3D environment in questions Q1-Q5, results are similar to those we obtained in the evaluation without assistance [3]. It is worth noting that the new group of questions related to assistance (Q6-Q8) have scores over 3.8. We extracted a number of relevant aspects of assistance in v-mWater from questionnaire qualitative measures as well as from user debriefings with the evaluation team. Generally, users like the way that the assistance was provided and how it helps task accomplishment. The assistance clearly guided them to perform the task and corrected users behaviour when they deviated from the proper one. The overall opinion of the system was positive.

Usability criteria, such as effectiveness, efficiency, errors and satisfaction have been analysed answering the research questions introduced in section 5.1.

RQ1: *Assistance helpfulness.* The assistance was voluntarily requested by the 93% of the testers: i) at the beginning of the task (6 users), ii) in the middle of the task (4 users), to check if they were doing it properly and iii) when they were trying to register without getting price information (3 users). Therefore, the assistance was helpful to all users at some time during the task. This fact is reinforced by the answers to Q6, which has an average value of 4.2.

RQ2: *User-PA interaction.* Along the test, all users interacted with the PA in a seamless way: they managed to ask for the correct advice and recognised the received plan. Thus, user-PA interaction was satisfactory (as Q7 also indicates with a value over 4).

RQ3: *Plan tracking.* All users who requested the plan understood its structure and followed it without errors. In general, the advice was highly comprehensible. Related question Q8 has a value of 3.8.

RQ4: *Task Completion.* The difference in number of actions with assistance (average of 7, $\sigma = 2.3$) and without it (average of 10.8, $\sigma = 3.4$) has proven to be significant by a *one tailed* unequal variance t-test with a p-value of 0.004 (p-value < 0.05). In order to successfully complete the given task, users had to perform a minimum of 5 actions (see Mary's actions on the left of Fig. 2). If we analyse these averages with and without assistance respect to this minimum, they represent a 140% and 216% respectively, so that assistance provides a 76% reduction. Assuming we are considering the same type of actions (i.e. the ones defined in the social model), we take the number of actions as a measure of efficiency. Assisted users went more directly to the goal and, thus, had less cognitive load than non-assisted ones. Overall, assistance reduces perceived task complexity, and this fact is confirmed by the lower percentage of fails with assistance (7%) respect to those without it (23%).

6 Conclusion

We propose Personal Assistant agents (PA) for advising users to perform complex tasks in hybrid structured 3D Virtual Environments (VE). These VE are hybrid since participants can be human and SW, and structured because most interactions are regulated by an Organization Centered MAS (OCMAS). A PA provides its assisted user with a plan, a sequence of (OCMAS compliant) user actions, that will be extended with other participants' actions whenever needed to accomplish user's task. To do so, we propose an extension of A^*, namely PLAN-EA. Evaluation results indicate that assistance impacts positively in usability measures of efficiency, efficacy and satisfaction. A comparative analysis of the number of user actions with (7) and without (10.8) assistance showed a significant difference. Moreover, with assistance, 93% of users completed the task, compared to 77% of users without assistance. Finally, users liked the way assistance was provided and how it facilitated task completion.

We are currently studying PAs in the smart microgrids domain [14]. As future work, we plan to improve the way plans are presented to users, to make PAs proactive, and to extend their capabilities with both justification and estimation services.

References

[1] J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: An organizational view of multi-agent systems. In *Agent-Oriented Software Engineering IV*, volume 2935 of *Lecture Notes in Computer Science*, pages 214–230. Springer Berlin Heidelberg, 2004.

[2] P. Almajano, M. Lopez-Sanchez, and I. Rodriguez. An assistance infrastructure to inform agents for decision support in open mas. In *ITMAS'12, 93-106*, 2012.

[3] P. Almajano, E. Mayas, I. Rodriguez, M. Lopez-Sanchez, and A. Puig. Structuring interactions in a hybrid virtual environment: Infrastructure & usability. In *GRAPP'13*, pages 288–297, 2013.

[4] R. Dong, K. McCarthy, M. O'Mahony, M. Schaal, and B. Smyth. Towards an intelligent reviewer's assistant: recommending topics to help users to write better product reviews. In *IUI'12*, pages 159–168. ACM, 2012.

[5] S. Kumar, A. Kunjithapatham, M. Sheshagiri, T. Finin, Joshi A., Y. Peng, and R.S. Cost. A Personal Agent Application for the Semantic Web. In *AAAI 2002 Fall Symposium Series*, November 2002.

[6] M. Lujak and H. Billhardt. Coordinating emergency medical assistance. In *Agreement Technologies*, volume 8 of *Law, Governance and Technology Series*, pages 597–609. Springer Netherlands, 2013.

[7] A. Faulring, B. Myers, K. Mohnkern, B. Schmerl, A. Steinfeld, J. Zimmerman, A. Smailagic, J. Hansen, and D. Siewiorek. Agent-assisted task management that reduces email overload. In *IUI'10*, pages 61–70. ACM, 2010.

[8] H. Chalupsky, Y. Gil, C.A. Knoblock, K. Lerman, J. Oh, D.V. Pynadath, T.A. Russ, and M. Tambe. Electric elves: Applying agent technology to support human organizations. In *IAAI'01*, pages 51–58. AAAI Press, 2001.

[9] J. Oh, F. Meneguzzi, K. Sycara, and T.J. Norman. Prognostic normative reasoning. *Engineering Applications of Artificial Intelligence*, 26(2):863 – 872, 2013.

[10] E.M.A.G. van Dijk, H.J.A. op den Akker, A. Nijholt, and J. Zwiers. Navigation assistance in virtual worlds. *Informing Science, Special Series on Community Informatics*, 6:115–125, 2003.

[11] L. Chittaro, L. Ieronutti, and R. Ranon. Navigating 3d virtual environments by following embodied agents: a proposal and its informal evaluation on a virtual museum application. *PsychNology Journal*, 2(1):24–42, 2004.

[12] M. Esteva, B. Rosell, Rodríguez-Aguilar. J. A., and J. L. Arcos. AMELI: An agent-based middleware for electronic institutions. In *AAMAS'04*, pages 236–243, 2004.

[13] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Sci-*

ence and Cybernetics, IEEE Transactions on, 4(2):100–107, 1968.

[14] A. Bourazeri, J. Pitt, P. Almajano, I. Rodriguez, and M. Lopez-Sanchez. Meet the meter: Visualising smartgrids using self-organising electronic institutions and serious games. In *SASO'12, 145-150*, 2012.