

A Software Environment for an Adaptive Human-Aware Software Agent Supporting Attention-Demanding Tasks¹

Tibor Bosse¹, Zulfiqar A. Memon^{1,2}, Rogier Oorburg³, Jan Treur¹, Muhammad Umair^{1,4}, Michael de Vos³

¹ VU University Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081, 1081 HV Amsterdam

Email: {tbosse, zamemon, treur, mumair}@few.vu.nl

URL: <http://www.few.vu.nl/~{tbosse, zamemon, treur, mumair}>

² Sukkur Institute of Business Administration (Sukkur IBA)
Air Port Road Sukkur, Sindh, Pakistan

³ Force Vision Lab, Barbara Strozziilaan 362a, 1083 HN Amsterdam, The Netherlands
Email: {rogier, michael}@forcevisionlab.nl

⁴ COMSATS Institute of Information Technology, Department of Computer Science Lahore, Pakistan

Abstract. This paper presents a software environment providing human-aware ambient support for a human performing a task that demands substantial amounts of attention. The agent obtains human attention-awareness in an adaptive manner by use of a dynamical model of human attention, gaze sensing by an eye-tracker, and information about features of the objects in the environment which is parameterised for characteristics of the human specified above. The agent uses a built-in adaptation model to adapt on the fly the values of these parameters to the personal characteristics of the human. The software agent has been implemented in a component-based manner within the Adobe® Flex® environment, thereby also integrating the Tobii® eye-tracker. It has been applied in a setup for a task where the human has to identify enemies and allies, and eliminate the enemies.

Keywords: Human-aware, ambient, attention, Adobe Flex, software agent, human-aware, adaptive

1 Introduction

The area of Ambient Intelligence or Pervasive Computing envisions a world in which humans are surrounded by intelligent software agents that are unobtrusively embedded in their direct environment, and support them in various tasks (e.g., [1, 2, 3]). One of the more ambitious challenges for the design of ambient or pervasive systems is to create an appropriate representation and awareness of a human's states; e.g., [1, 2, 3, 20]. Ambient agents can provide more dedicated support when they have a certain level of human-awareness. This may require awareness not only of personal characteristics such as preferences, but also of (dynamically changing) states of the human. Examples of such states are emotion and stress, fatigue and exhaustion, goals and intentions, and attention states. Such human-aware systems can be taken to perform a certain type of mindreading or to possess what in the psychological and philosophical literature is called a Theory of Mind; e.g., [9, 12, 13]. Like in the evolutionary human history, within different applications such mindreading may address different types of human states, such as intention, attention, belief or emotion states; e.g., see [12]. Acquiring awareness of such states is a nontrivial challenge. Sensors may be used by the agent to get information about the human's current state, such as face readers, eye-trackers or heart rate sensors, but sensor information can rarely be directly related in a one-to-one manner to the human's states that are of concern.

A more general situation is that such sensor information can be used in a more indirect manner in dynamical models that express temporal relationships between a number of variables including these human's states and the sensor information; e.g., [19]. As humans may show substantial individual differences in characteristics in cognitive functioning, such a dynamical model usually includes a number of parameters for a number of specific characteristics of the human. Therefore they only can be used in a dedicated manner when sufficiently accurate estimations can be made for the values of these parameters as representations for the characteristics of the human considered. For applications in software agents this implies that such an agent does not only need a dynamical model of the human's processes, but also an adaptation model describing how the parameter values of the former model can be adapted over time to the characteristics of the human.

¹ Parts of this article are based on work presented at the 12th International Conference on Principles of Practice in Multi-Agent Systems, PRIMA'09 [10], and the 10th International Conference on Computational Science, ICCS'10 [16].

This paper presents a software agent that makes use of such a parameterised dynamical model of the human, and in addition possesses an adaptation model that on the fly tunes the parameter values to the human characteristics. A case study was undertaken to test it for a human's attention states. Some literature on models for human attention can be found, for example, in [7, 15, 17, 22]. The human-attention-awareness for this case study uses three ingredients: (1) sensor information on the human's gaze, (2) information on features of objects in the environment, and (3) a dynamical (differential equations) model of the dynamics of attention levels over time integrating the instantaneous information from (1) and (2), and using partial persistency of attention over time. The agent's adaptation model was designed and implemented in a generic manner, but for the case study was applied to the dynamical model for attention, thereby addressing, among others, parameters for weights of different features of objects, the effect of distance between an object and gaze, and an attention persistency factor.

The software agent and the further ambient support software environment has been implemented within the Adobe® Flex® development environment according to a component-based design, with event-driven interaction between components based on ActionScript. For the gaze sensing it integrates the Tobii® eye-tracker. Before entering a task performance session, initially it allows to set values for a number of parameters.

In this paper in Section 2 an example of the type of application is presented. Section 3 addresses the assessment of the attention, whereas in Section 4 it is discussed how intervention actions are generated. In Section 5 some more details of the developed software are discussed. Section 6 presents experimental results for human attentional state model. Section 7 presents the background ideas of the parameter adaptation approach developed, and in Section 8 the overall adaptation model is described. In Section 9 some more details of the developed software and some results for the case study are discussed. In Section 10 a verification of the model is presented. Section 11 is a discussion.

2 The Type of Attention-Demanding Task Considered

To address the possibilities for human-aware ambient support of attention-demanding tasks a specific example environment has been developed. Main characteristics of the type of task considered in this environment are:

- it has elements of monitoring, inspection and analysis of a visually represented environment
- it has elements of deciding on and performing actions to cope with circumstances that require intervention
- part of the task concerns cognitive load
- (visual) attention plays an important role
- work load may vary over time and at times may become stressful and exhausting

This context has been given the form of a (simplified) simulated environment to perform such a type of task. More specifically, the idea is that a person has to (1) inspect visually displayed moving objects in the environment, and identify whether such an object is dangerous (enemy) or not (ally), and (2) for each of such objects, depending on the identification perform some actions. The person in charge (further called player) faces objects traversing the screen from the top to the bottom.

The player's task consists in classifying each object as ally or enemy, and shooting down the enemies while letting allies land safely. Identification of an object is a cognitive task, in a simplified form represented by an arithmetical calculation. In order to determine whether an object is an ally or an enemy, the player mouse-clicks on the object whereafter an arithmetical formula appears next to that object. A correct formula means that we are dealing with an ally while enemies show false formulas. To categorize an object as ally, the ← key can be pressed, for enemies the → key. A spoken voice will confirm the choice. Objects designated as allies turn greenish yellow while those identified as enemies go reddish orange. The player uses a cannon at the bottom of the screen to shoot down (hostile) objects. The picture in Figure 1 shows an identified ally to the left and an identified enemy to the right. Using scores and costs that are assigned it can be investigated if support measures affect the efficiency of the task execution. Scores are assigned according to Table 1. Costs are counted as +1 per fired missile; costs are calculated and shown independently of the score.

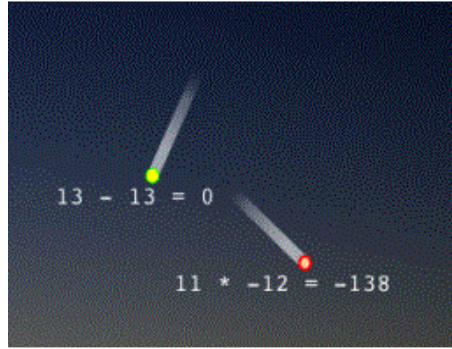


Figure 1. Partial screenshot of the task environment

Objects	ally	enemy
Landed	+1	-1
shot down	-1	+1

Table 1. Score table

3 Assessment of the Human's Attentional State

The assessment of the human's attentional state concerns three different elements. First, from the supply side it is discussed how to estimate the amount of attention the human is paying to each of the different objects. Next, from the demand side it is discussed how to estimate the amount of attention that is required for each of the objects (urgency). Finally, the two types of estimations are compared to determine a discrepancy assessment for each of the objects.

3.1 Attention Estimation

To perform a demanding task of the type considered, it is important to have sufficient visual attention. In such a situation, a player may be assisted by an ambient agent that keeps track of where his or her attention is, and provide support in case the attention is not where it should be. To this end, the software exploits a domain model of the state of attention of the player over time. Below the domain model for attentional processes of the player is described, as adopted from [7].

The player's attention at the current point in time is distributed over all objects on the screen (incoming unidentified objects, the object that displayed formulas, identified ally- and enemy-indications). Some objects get more attention, while other objects get less attention. This depends on two main aspects. One is the distance between the object and the player's gaze direction: the shorter the distance between the gaze target and the object, the higher the attention level of this object is, and vice versa. Another main aspect is an object's potential (capability) for attracting attention based on the object's characteristics (such as brightness and size, for example: a brighter and larger object attracts more attention). As a first step for each object on the screen its potential for attracting attention is determined. For example, when object O has two characteristics, brightness and size, which have numerical values V_1 and V_2 , respectively, then the value of the potential of the object O for attracting a player's attention at a given time point is calculated as $w_1 * V_1 + w_2 * V_2$ where w_1 and w_2 are parameters (weights) reflecting the relative importance of the characteristics brightness and size, respectively; w_1 and w_2 are both real numbers between 0 and 1. When multiple characteristics C_i for $i = 1, \dots, m$ are involved, the attention-attracting potential of an object O is expressed by a weighted sum

$$\sum_{i=1}^m w_i * V_i$$

where V_i is the value of characteristic C_i . The precise choice of the characteristics together with the values of the corresponding weights (the sum of which is set on 1 by normalisation) can be chosen from the following list:

- brightness
- colour
- size
- blinking of the object
- a circle around an object
- blinking of circle around object
- time to the ground
- direction of movement (angle)
- speed
- known as enemy or ally
- distance to the ground

The values for these variables are expressed by numbers in the interval $[0, 1]$, with 0 no attraction potential and 1 highest attraction potential. For a specific session, initially a selection of them can be made and weight factors assigned.

As a next step the player's gaze location is taken into account, in order to combine it with the potential of an object for attracting the player's attention; a location is indicated by a point on the screen, represented by a pair of coordinates (x, y) with respect to horizontal and vertical axes for the screen. To take gaze into account the potential of the object for attraction of the player's attention is divided by a function depending on the player's gaze location:

$$V(O) = \left(\sum_{i=1}^m w_i * V_i \right) / (1 + \alpha * d(O, G)^2)$$

Here $d(O, G)$ is the Euclidean distance between the object O and the player's gaze G . If the gaze is represented by coordinates (x_1, y_1) and the object's location coordinates are (x_2, y_2) then $d(O, G)$ can be determined as

$$\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$$

and hence $d(O, G)^2$ by $(x_1-x_2)^2 + (y_1-y_2)^2$. The parameter α (represented by a positive real number) affects how fast an object loses the player's attention as the gaze moves further away from the object. The higher α is, the lower the attention for objects distant from the player's focus.

It is assumed that a person can have a fixed total amount of attention A distributed over all available objects on the screen. Therefore the attention level $AV(O)$ of an object O expresses the amount of the player's attention directed at the object O as a proportion (percentage) of the player's total attention. If there are n objects in total on the screen and the attention value $V(O_j)$ of the object O_j is indicated by V_j ($1 \leq j \leq n$), then the attention level $AV(O_i)$ of the object O_i is determined in a normalised form as

$$AV(O_i) = (V_i / (V_1 + V_2 + \dots + V_n)) * A$$

As a gaze can change from one moment to the other, and it has a strong effect on the attention values, these attention values can show quite instable patterns. However, it can be assumed that attention persists over short time periods. To model this the attention values for object O can be modelled with persistence as $PAV(O)$ using the following difference equation:

$$PAV(O)_{t+\Delta t} = PAV(O)_t + \beta * (AV(O)_t - PAV(O)_t) \Delta t$$

Here β is an attention flexibility parameter with a positive value between 0 and 1 , and time step with $0 \leq \Delta t \leq 1$; a high value of β results in fast changes and a low value in a high persistence of the old value. Note that for $\Delta t = 1$ this can be rewritten into a weighted sum form

$$PAV(O)_{t+1} = \beta * AV(O)_t + (1-\beta) PAV(O)_t$$

which shows more explicitly how the parameter β expresses partial persistence of attention; for example, for the extreme value $\beta = 0$, the attention state would be fully persistent. Written in differential equation format the dynamical model is as follows:

$$dPAV(O)/dt = \beta * (AV(O) - PAV(O))$$

This represents a system of n differential equations for all of the n objects involved, which via $AV(O)$ integrates the gaze information and information about the object features over time.

3.2 Urgency Estimation

The task considered involves high attention demands, especially when many objects are coming in a short period of time. Some of these objects can be ignored, for example because they are allies. Other objects demand more attention, for example enemies that have a high speed and/or that are already close to the ground. To find out which objects should be given attention, it is estimated how critical objects are: the urgency of objects. This is done based on n urgency-indication factors by a weighted sum

$$UV(O) = \begin{cases} \sum_{i=1}^n w_i * U_i & \text{if } O \text{ is an enemy} \\ 0 & \text{if } O \text{ is an ally} \end{cases}$$

Here U_i is the value of the i -th urgency factor and w_i the weight of this factor (with total sum I). The factors that can be considered here are:

- time to the ground
- distance to the ground
- direction of movement (angle)
- speed

The values for these variables are expressed by numbers in the interval $[0, I]$, with 0 no urgency and I highest urgency. For a specific session, initially a selection of them can be made and weight factors assigned.

It has been assumed that like attention also urgency persists over shorter time periods. To model this the attention values for object O can be modelled with persistence as $PUV(O)$ using the following difference equation:

$$PUV(O)_{t+\Delta t} = PUV(O)_t + \gamma*(UV(O)_t - PUV(O)_t)\Delta t$$

Here γ is an urgency flexibility parameter with a positive value between 0 and I , and time step with $0 \leq \Delta t \leq I$; a high value of γ results in fast changes and a low value in a high persistence of the old value of the urgency. Note that for $\Delta t = I$ this can be rewritten into a weighted sum form

$$PUV(O)_{t+I} = \gamma*UV(O)_t + (1-\gamma) PUV(O)_t$$

which shows more explicitly how the parameter γ expresses partial persistence of urgency; for example, for the extreme value $\gamma = 0$, the urgency state would be fully persistent. Written in differential equation format the dynamical model is as follows:

$$dPUV(O)/dt = \gamma*(UV(O) - PUV(O))$$

Again, this represents a system of n differential equations for all of the n objects involved, which via $UV(O)$ integrates the gaze information and information about the object features over time.

3.3 Discrepancy Assessment

To determine whether there is enough attention for objects that demand attention some comparison has to be made. The attention levels that are estimated can be considered as offered attention utilisation, and the urgencies can be considered as demands. However, in principle these quantities are not expressed according to the same measure, which makes that they are not easily comparable. For example, the total sum of attention values for all objects is A , which may be set on I , whereas the total sum of urgencies can easily be much higher than I . Moreover, in general, it is not clear at forehand how high an attention level has to be in order to represent

‘enough attention’. Therefore some rescaling has been made in the comparison, in the following discrepancy assessment:

$$D(O) = w_u * U(O) - w_a * PAV(O)$$

This uses initially set weight factors for urgency and attention level to determine the discrepancy for an object O . The interpretation is

- $D(O) = 0$ sufficient attention for the object
- $D(O) < 0$ more than sufficient attention
- $D(O) > 0$ insufficient attention

The resulting discrepancy assessments are used as input to determine appropriate intervention actions.

<i>characteristics</i>	<i>parameter</i>	<i>symbol</i>	<i>range</i>
human characteristics	object feature weight factors	w_i	$0 \leq w_i \leq 1$ & $\sum w_i = 1$
	gaze distance effect rate	α	$0 \leq \alpha$
	attention flexibility rate	β	$0 < \beta \leq 1$
environmental characteristics	urgency aspect weight factors	r_i	$0 \leq r_i \leq 1$ & $\sum r_i = 1$
combined characteristics	comparison weight factors	s_i	$0 \leq s_i \leq 1$

Table 2. Overview of the parameters in the dynamical model

As shown, the overall dynamical model involves a number of parameters some of which relate to characteristics of the human and some others to characteristics of the environment, or to a combination. As a summary an overview is given in Table 2.

4 Selection Process for Intervention Actions

Within the process to determine intervention actions two main decisions made are based on the assessment information: (1) which objects to address, and (2) which intervention actions to apply on the addressed objects. Below the criteria are discussed on which these decisions are based. For a summary, see Table 3.

4.1 Object Focus Set

For the first decision two criteria are considered, based on two parameters th (a real number ≥ 0) and k (a natural number): (a) the discrepancy of the object is $> th$, and (b) it is among the k objects with highest discrepancy. Given certain values set for these parameters, the combination of these criteria defines a focus set of objects addressed. A special case used in the example simulations shown is $k = 1$ and $th = 0$. In this case the object with the highest positive discrepancy is selected.

4.2 Intervention Action Selection

Action selection does not involve only (a) the selection of a specific type of action (for example, to display a pointer to the object), but also (b) determination of the intensity value for that action (e.g., the size of the pointer). The actions that may be considered are affecting the following variables:

- brightness
- colour
- size
- blinking
- a circle around an object

- blinking of circle around an object

The values for these variables are expressed by numbers in the interval $[0, 1]$, with 0 no intensity and 1 highest intensity. For each object in the focus set an action and its intensity value is selected using *sensitivity factors*. A sensitivity factor S for a variable X addressed by a certain action with respect to discrepancy D defines how much (in the given situation) the discrepancy D will change upon a change in the variable, which is mathematically denoted by the partial derivative $\partial D / \partial X$.

To determine a sensitivity factor S (i.e., determining the partial derivative $\partial D / \partial X$) both analytical and approximation methods or a combination of them can be used. The attention model is defined by a differential equation and no direct formula is given as, for example, was the case in the adaptation approach described in [8]. Therefore here this partial derivative cannot be determined by analytic calculation. As an approximation method, a small arbitrary change ΔX in the value of variable X can be tried (for example a change of 4%, which for $X = 0.5$ makes $\Delta X = 0.02$), and based on the resulting change ΔD in the value of D (for example $\Delta D = -0.15$) found in predicted discrepancy, the sensitivity factor S can be estimated by

$$S = \Delta D / \Delta X$$

which for example provides $S = -0.15 / 0.02 = -7.5$. Note that the norm for the discrepancy D is 0 , so $\Delta D = D$ can be taken.

Given that for each action option the sensitivity factor is known, as a negative real number, it is used in the following manner. First it is multiplied by the remaining interval within $[0, 1]$ for the variable addressed by the action: this provides the maximal effect - $S*(1-X)$ on D that can be achieved by applying this action. An action is chosen where this value is maximal, while its potential impact on discrepancy is the highest. To approximately compensate the discrepancy the following value is taken for the intensity of the action:

$$\begin{aligned} \Delta X &= - D / S \\ X + \Delta X &= X - D / S \end{aligned}$$

So, when $D = 0.6$, $S = -3$ and X has value 0.3 this obtains $\Delta X = 0.6/3 = 0.2$, so the action is selected with value

$$X + \Delta X = 0.5$$

In case $X - D/S$ exceeds 1 , the maximal intensity 1 can be taken for the action. Table 3 shows an overview of the decisions made in intervention action selection.

	Indication	Decision parameters	Decision criteria
<i>Object focus set</i>	objects with high discrepancy	discrepancy D threshold th number of objects k	$D > th$ & the object is among the k objects with highest D
<i>Selected action</i>	actions with high impact on discrepancy	current intensity value X sensitivity S	action with highest maximal impact - $S*(1-X)$
<i>Action intensity</i>	approximate compensation of discrepancy	discrepancy D current intensity value X sensitivity S	action intensity $min(1, X - D/S)$

Table 3. Decision criteria for selection of intervention actions

5 The Software Environment in Adobe Flex

In this section some details of the ambient support environment for attention-demanding tasks are described. This software environment has been implemented within the Adobe® Flex® development environment. The software was implemented according to a component-based design. Between the different components event-driven interaction takes place, implemented using ActionScript. Moreover, for the measuring gaze a specific sensing component was included that connects the Tobii® eye-tracker that was used. An interface component

allows for initially setting of values for the parameters. Two other components represent the assessment and action selection processes described in Sections 3 and 4.

A number of sessions have been performed using this software environment to test the behaviour, and some data of one example session have been included in this section. Figure 2 shows the interface component where the user sets the values of the parameters used in the Attention estimation, Urgency estimation and Discrepancy assessment as explained in Section 3. Table 4 shows the values of urgency and attention estimation, and discrepancy of three objects during the session.

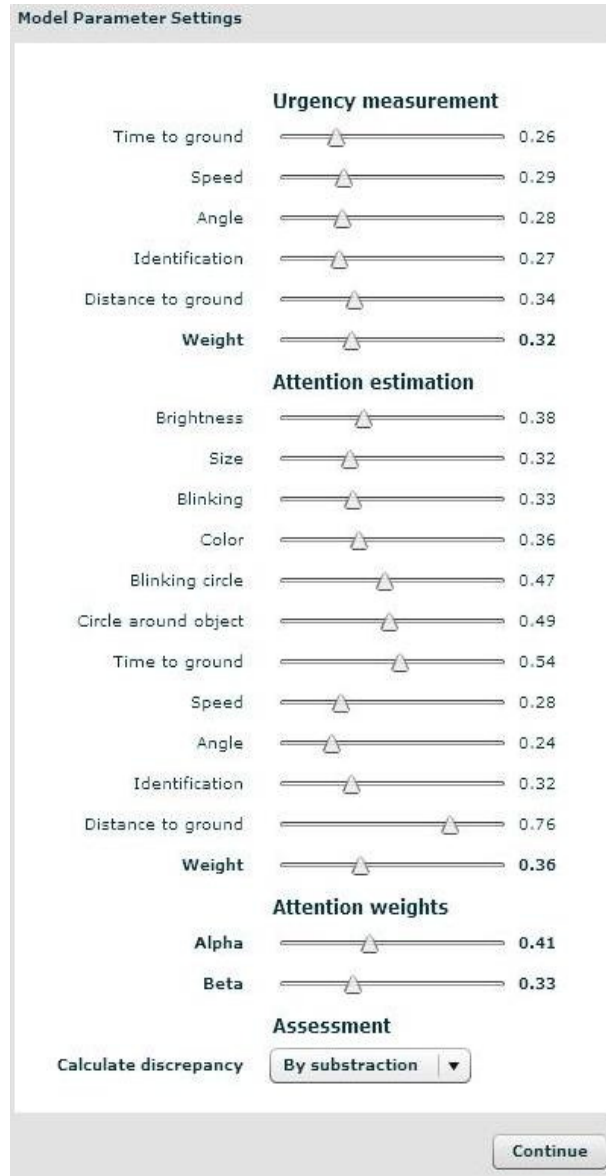


Figure 2. Interface to set parameter values

Example results of the session are shown in the snapshot displayed in Figure 3, where 3 objects are visible; i.e. object1 (leftmost object), object2 (middle object) and object3 (rightmost object) with different types of support provided to them depending upon the estimated urgency and attention levels. The gaze as given by the Tobii® eye-tracker is denoted by “ $\frac{4}{11}$ ”. As can be seen from the data shown in Table 4, object2 has a negative discrepancy value (i.e., sufficient attention is given), so in line with the decision criteria described in Table 3 in Section 4, no support is provided to it. In this situation support is provided to the object having the highest discrepancy, i.e., object1. Figure 3 also shows the current score and the cost in this stage of the session. As can

be seen, because of the support measures provided the cost is less as compared to the score earned by the human, which shows efficiency of the task execution with the given support.

	Urgency weight w1	Urgency value U(O)	Attention weight w2	Attention value PAV(O)	Discrepancy D(O)
object1(leftmost)	0.32	0.8	0.36	0.2	0.184
object2(middle)	0.32	0.6	0.36	0.8	-0.096
object3(rightmost)	0.32	0.4	0.36	0.3	0.02

Table 4. Values of parameters and variables for an example session



Figure 3. Snapshot of an example session

6 Simulation Experiments for the Ambient Attentional Support Agent

Based on the ambient support system described above, a number of simulations have been performed. A first example simulation trace for a non-attentive person included in this section is shown in Figure 4 as an illustration (one ally object) and Figure 5 (one enemy object).

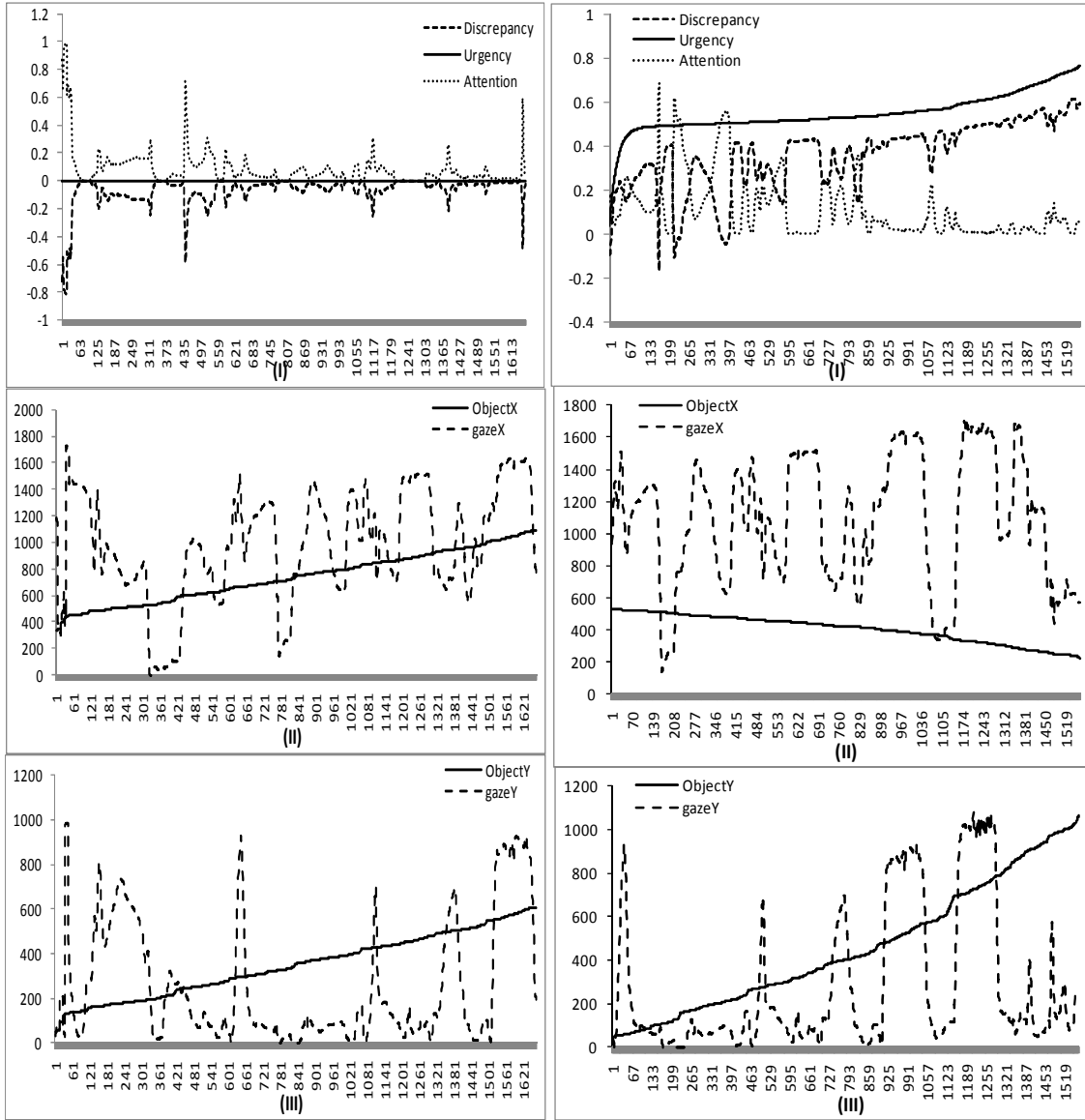


Figure 4. Simulation trace 1 – for an ally object

Figure 5. Simulation trace 1 – for an enemy object (non-attentive person)

$$\alpha=0.9, \beta=0.95, \gamma=0.1, w_1=w_2=0.95$$

In all of the figures shown, time is on the horizontal axis, whereas in Figure 4 (I) and Figure 5 (I), the vertical axis shows the values of the human’s attentional state elements, and in Figure 4 (II) & (III) and Figure 5 (II) & (III) it shows, respectively, x- and y-coordinates on the screen. The example trace in Figure 4 (I) shows the values of the elements of the human’s attentional state for an ally object. As can be noticed the urgency value of an ally object remains 0 for the whole simulation session, as has been described in Section 3.2. It further shows that the urgency value is independent of the human’s gaze. As described in Section 3.1, the attention element depends on the difference between the human’s gaze value and the current location of the object.

This can be seen in Figure 4 (I) and (II) & (III), where the attention value increases as the gaze of the human comes closer to the object and it decreases as human gaze goes away from it. Figure 4 (I) also shows the discrepancy element, which is the weighted difference between the attention and urgency elements. For this particular example simulation the weights for attention and urgency are same, i.e., 0.95, and as the urgency for this particular object is 0 (because of being an ally), therefore the discrepancy value is exactly opposite to the attention value. In other words, for this simulation experiment the discrepancy is dependent on the attention value only.

The example trace in Figure 5 (I) shows the values of the elements of the human’s attentional state for an enemy object. As can be noticed the urgency value of an enemy object gradually increases over the time until it reaches to the ground, as has been described in Section 3.2. For this example simulation the factors that have been considered are time and distance to ground along with object identification.

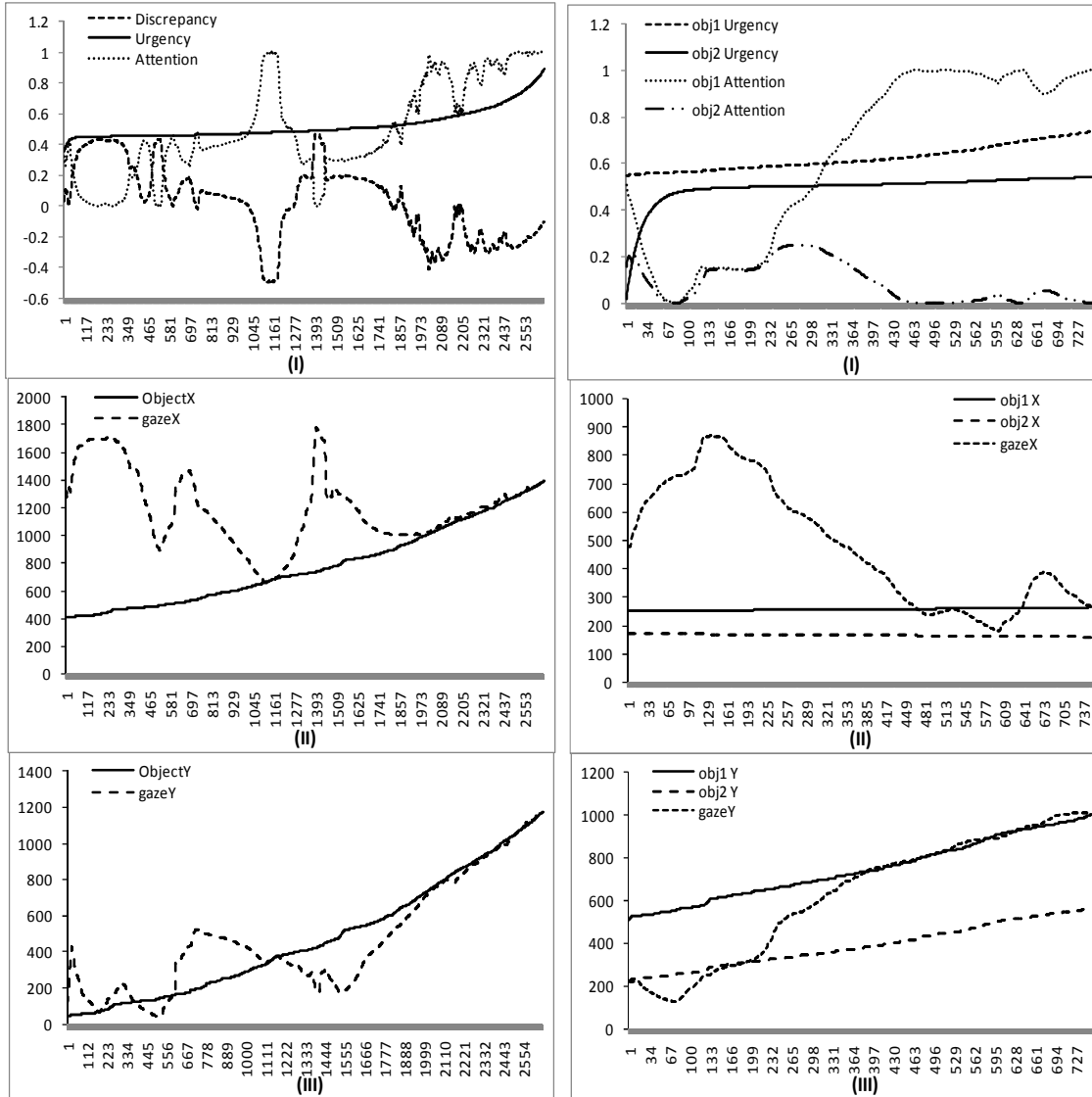


Figure 6. Simulation trace 2 – for an enemy object (attentive person)

$$\alpha=0.9, \beta=0.95, \gamma=0.1, w_1=w_2=0.95$$

Figure 7. Simulation trace 3 – Comparison of two enemy objects

It can also be noted that as the object’s Y coordinate increases, i.e., the object is coming closer to the ground (see Figure 5 (III)), the urgency value increases simultaneously. On the other hand, the value of the attention element shows a similar pattern as for the ally object described earlier, because it depends on the difference between the gaze and object coordinates. This can be seen in Figure 5 (I) and (II) & (III).

This example simulation trace depicts a scenario where the human does not pay attention to the object even though the urgency value of the object increases. This can be seen in Figure 5 (II) & (III) where the user gaze goes away from the object.

Figure 6 shows the simulation of an example scenario where the person pays attention to those objects for which the proposed software environment provides support, i.e., those objects that are closer to the ground and have not been paid attention to.

This can be seen in Figure 6 (I), (II) & (III), where the person's gaze comes closer to the object whose urgency is higher because of the ambient support provided by the software, as opposed to the pattern shows in Figure 5.

The simulation experiment trace shown in Figure 7 compares two enemy objects, where the person pays attention to the most urgent object as compared to the least urgent one. As can be noticed in Figure 7 (I), the person initially pays attention to obj1 but loses attention after a short duration. Later, as the proposed software environment provides support to the most urgent object, i.e., obj1, the person again starts to pay attention to it, and hence the attention for that object increases for the rest of the simulation trace. Notice that the attention value of obj2 is lower compared to obj1. It can also be validated by the generated pattern of movement of the human gaze and the object coordinates, as can be seen in Figure 7 (II) & (III).

7 The Adaptation Approach

When using a dynamical model to assess a human's states in the context of a task environment, the software agent has to maintain beliefs about characteristics of the human, used as parameters in the model. Examples of such parameters are the weight factors that are used for different human features, i.e., the w_i as used in the first formula in Section 3.1. As often it is not possible to determine accurate values for such parameters at forehand, this section describes a method by which the agent adapts its beliefs concerning human characteristics to the real characteristics. The main idea is as follows. The agent initially receives rough estimations of the values for these human characteristics, and maintains them as beliefs. Using the dynamical model with parameter values as represented by these initial beliefs, the agent predicts the human state, up to a certain time point. When at that time point, for example by observation, information is obtained that can be related to the real value of one or more state variables of the model, this can be used as input for the adaptation process. The agent then tries to minimize the difference between predicted and real value by adjusting the beliefs on the human characteristics (i.e., the parameter values which were initially assumed). This process of adaptation is kept going on until the difference is low enough, i.e., until the agent has a sufficiently accurate set of beliefs about the human's characteristics. To be able to make reasonable adjustments it is needed to obtain information on how a change in a parameter value affects the difference between predicted and real value of the variable that is considered; this is called the sensitivity of the variable value for the parameter value. In more detail the adaptation process is described as follows.

7.1 Sensitivities: how they are determined

Generally speaking the way of determination of the sensitivities is the same as we have already discussed in Section 4.2. The difference between these two applications is; formerly it was determined for *variables* whereas currently we are determining it for *parameters* related to some *variables*. Within this adaptation process sensitivities of state variables for changes in parameter values for human characteristics play an important role. The *sensitivity* S of variable X for parameter P is the number such that a change ΔP in the value of P of parameter P will lead to a change ΔX in X which is (approximately) proportional to ΔP with proportion factor S :

$$\Delta X = S \Delta P$$

This is an approximation which is more accurate when the Δ 's are taken small; in fact the sensitivity is the partial derivative $\partial X / \partial P$. Using the similar approximation method used in Section 4.2, the following can be done. A small change ΔP in the parameter is tried to make an additional prediction for X , and based on the resulting change ΔX found in the two predicted values for X , by

$$S_{X,P} = \Delta X / \Delta P$$

the sensitivity S can be estimated. The idea is that this is done for each of the parameters, one by one.

7.2 Sensitivities: how they are used

Given that a sensitivity $S_{X,P}$ of variable X for parameter P is known it can be used in the following manner. First it can be noticed that sensitivities for some parameters P can be 0 or almost 0. Apparently, such parameters do not have any serious effect on the outcome of the value of variable X . Therefore, changing them based on available values for X does not make much sense: a deviation in value of X cannot be attributed to them, due to

their low influence. Based on the estimation of a sensitivity which has distance at least τ_5 to 0 (where τ_5 is a small threshold value), in principle a better guess for the value of P can be determined by taking

$$\Delta P = -\lambda * \Delta X / S_{X,P}$$

where ΔX is the deviation found between observed and predicted value of X ; so, for example, when $\Delta X = 0.25$ and $\lambda = 0.3$, then for $S_{X,P} = 0.75$ this obtains $\Delta P = -0.3*0.25/0.75 = -0.1$. However, when the sensitivity $S_{X,P}$ is a bit smaller, it could be possible that the adjustment of the value of P based on the formula above would exceed the maximum or minimum value of its range. For example, when $\Delta X = 0.25$, $\lambda = 0.3$, and $S_{X,P} = 0.025$ it results in $\Delta P = -0.3*0.25/0.025 = -3$. To avoid such problems, a kind of threshold function can be applied that maps, for example for a parameter with values in the interval $[0, 1]$, the proposed adjustment on a $[0, 1]$ interval (which can still be multiplied by a factor for other intervals):

$$\begin{aligned} \Delta P &= \lambda * th(\sigma, \tau, \Delta X / S_{X,P}) * (I - W) && \text{when } \Delta X / S_{X,P} \geq 0 \\ \Delta P &= -\lambda * th(\sigma, \tau, -\Delta X / S_{X,P}) * W && \text{when } \Delta X / S_{X,P} < 0 \end{aligned}$$

Here the threshold function with steepness σ and threshold value τ is defined by

$$th(\sigma, \tau, V) = 1 / (1 + e^{-\sigma(V - \tau)})$$

or, to allow for lower steepness values by

$$\begin{aligned} th(\sigma, \tau, V) &= [1 / (1 + e^{-\sigma(V - \tau)}) - 1 / (1 + e^{\sigma\tau})] / [1 - 1 / (1 + e^{\sigma\tau})] \\ &= [1 / (1 + e^{-\sigma(V - \tau)}) - 1 / (1 + e^{\sigma\tau})] / [e^{\sigma\tau} / (1 + e^{\sigma\tau})] \\ &= [1 / (1 + e^{-\sigma(V - \tau)}) - 1 / (1 + e^{\sigma\tau})] * [1 + e^{-\sigma\tau}] \end{aligned}$$

When for more than one variable X information about its real value is obtained, the adjustment ΔP for parameter P is taken as the average of all calculated adjustments based on the different variables X such that the sensitivity $S_{X,P}$ is not close to 0.

8 The Overall Adaptation Process

Based on the adaptation approach explained in Section 7 above, the overall adaptation process was modelled as follows:

Initialisation

1. Take VF the *focus set of variables* X for which information about its real value can be obtained and take a time point t for which information about the real value of all X in F is to be obtained.
2. Take G the subset of parameters P for which adaptation is desired; the other parameters are kept constant.
3. Assume initial values for all of the parameters P .
4. Choose a value for adaptation speed λ .

Sensitivity Determination

5. By simulation determine predicted values V_X at time point t for each X in VF , using the assumed values of the parameters.
6. For each parameter P in G , by simulation determine predicted values CV_X at time point t for each X in VF , using only for P a value changed by some chosen ΔP and the unchanged assumed values for the other parameters.
7. For each parameter P in G and each variable X in VF , determine the sensitivity $S_{X,P}$ of X for P at time point t by dividing the difference between values for X found in 5. and 6. by ΔP :

$$S_{X,P} = (CV_X - V_X) / \Delta P$$

8. Take $VF(P)$ the *focus set of variables* X for P of those variable X for which the sensitivity $S_{X,P}$ has distance to 0 of at least the sensitivity threshold τ_s :

$$VF(P) = \{ X \in VF \mid |S_{X,P}| \geq \tau_s \}$$
9. Take PF the *focus set of parameters* for which the sensitivity for at least one $X \in VF$ has distance to 0 of at least the sensitivity threshold τ_s :

$$PF = \{ P \in G \mid \exists X \in VF |S_{X,P}| \geq \tau_s \} = \{ P \in G \mid VF(P) \neq \emptyset \}$$

Adjustment Determination

10. For each variable $X \in VF$ determine the deviation ΔX of the predicted value of X at t from information that is obtained about the real value of X at t .
11. For each parameter $P \in PF$ determine the change ΔP as

$$\Delta P = \lambda * th(\sigma, \tau, (\sum_{X \in VF(P)} \Delta X / S_{X,P}) / \#(VF(P))) * (1-W)$$

when $(\sum_{X \in VF(P)} \Delta X / S_{X,P}) \geq 0$

$$\Delta P = -\lambda * th(\sigma, \tau, -(\sum_{X \in VF(P)} \Delta X / S_{X,P}) / \#(VF(P))) * W$$

when $(\sum_{X \in VF(P)} \Delta X / S_{X,P}) < 0$
12. For each parameter $P \in PF$ adjust its value by ΔP .

By repeating this process a number of times both for each time point considered and for different time points, over time an approximation is obtained.

9 Further Details of the Implementation

To test the adaptation approach introduced above, again some simulation experiments have been performed. For this, a similar setup was used as in Section 6, but this time including parameter adaptation. Again, the main idea for an ambient agent supporting this task is that it has awareness about whether or not the human pays enough attention to urgent situations.

Using the same urgency model as before, the agent can determine how urgent the situations concerning certain objects are, and compare this to the amount of attention, thus obtaining the discrepancies for each of the objects (according to the model described in Section 3.3). These discrepancies play the role of the variables indicated by X in Section 8.

For the adaptation model, a crucial aspect is the way in which a deviation between the discrepancy value estimated by the agent and in reality can be determined. In this case, as an approximation the mouse clicks of the person have been used as an indication of discrepancy; see Table 5.

<i>enemy/ friend</i>	<i>mouse click</i>	<i>indicated discrepancy</i>	<i>current discrepancy</i>	<i>deviation based on current minus indicated discrepancy</i>
enemy	+	Negative	positive	current discrepancy + 0.2
enemy	+	Negative	negative	0
enemy	-	Positive	positive	0
enemy	-	Positive	negative	current discrepancy - 0.2
friend	+	Negative	positive	current discrepancy + 0.2
friend	+	Negative	negative	0
friend	-	Negative	positive	current discrepancy + 0.2
friend	-	Negative	negative	0

Table 5. Determination of deviations

For example, if at some time point a mouse click occurs at a certain position of an enemy, it is assumed that this indicates sufficient attention for that position at that point in time which implies an indication that the discrepancy should be negative. Similarly, when for certain duration no mouse click occurs at a friendly position, this is considered a sign for negative discrepancy. If the current discrepancy is positive and the indicated negative, the deviation is determined as the current discrepancy estimation plus 0.2.

Figure 8 shows for the example scenario how the three objects move compared to gaze positions. Object 1 (upper graphs), representing an enemy, moves almost vertically, and almost all the time the gaze position has

some distance to this object, except maybe around time points 800 and 950, here the gaze is a bit closer. In contrast, the positions of the friendly object 2, depicted in the second row of graphs, often coincide with the gaze positions. The lowest part shows that the positions of object 3 coincide with gaze positions only around time points 250 and 950. Tables 6 and 7 describe information about the settings in the example scenario. Note that the person considered performs far from perfect as the gaze is often at (friendly) object 2 and not at object 1 which is the enemy. Therefore an appropriate estimation of discrepancy between urgency and attention would show a rather positive value for object 1 and a rather negative value for object 2.

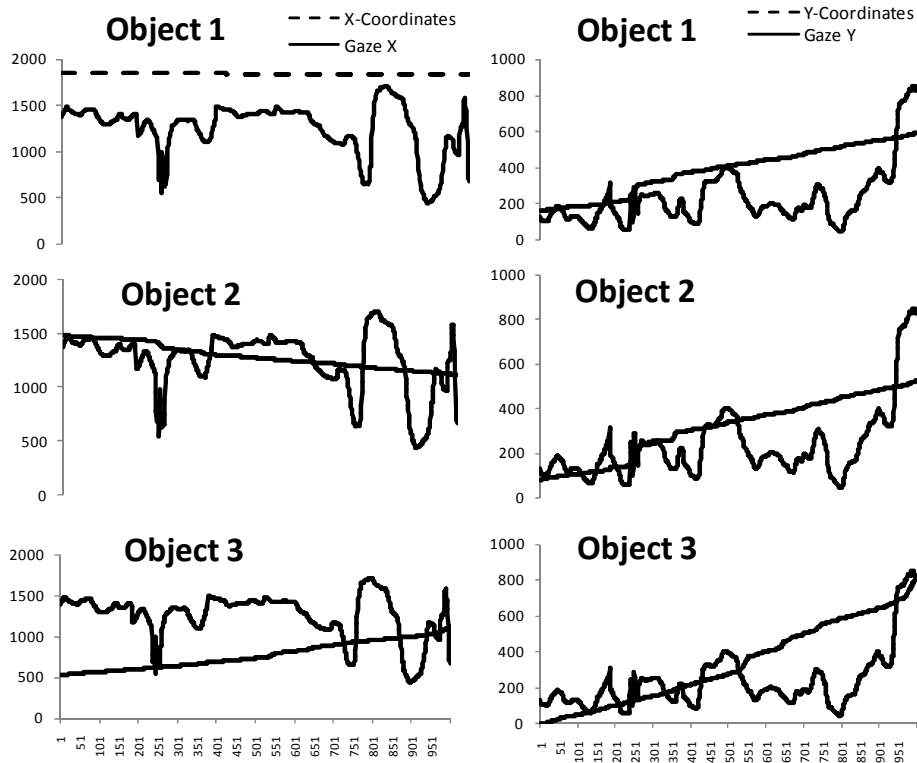


Figure 8. Positions of three objects over time compared to gaze positions in the example scenario: left hand sides show x-coordinates, right hand sides y-coordinates. It shows that from time point 0 to 200 the gaze is near object 2, around 250 near object 3, from 300 to 500 near object 2 again, around 800 most close to object 1, and after 900 near object 3 again.

	<i>Brightness</i>	<i>Size</i>	<i>Identification</i>
Object 1	0.9	0.2	enemy
Object 2	0.1	0.9	friend
Object 3	0.25	0.3	friend

Table 6. Some of the object features

α	β	<i>Identification weight</i>	<i>Urgency weight</i>	<i>Attention weight</i>	λ	<i>Urgency flexibility</i>
1	0.2	1	0.3	0.9	0.02	0.02

Table 7. Parameter values

In Figure 9 it is shown how the discrepancies develop over time for each of the objects. It shows that initially the discrepancies are not estimated well. For example, for object 1, the discrepancy starts negative. This incorrect estimation of the discrepancy can be explained by the parameter setting for weight factors for

brightness and size: the brightness weight factor is initially 0.9 , and object 1 happens to have a high brightness. The more or less converse situation is shown for object 2, which has a low brightness and high value for size. However, for the estimation of discrepancy the high size is not counted much as initially the weight factor parameter for size is only 0.2 .

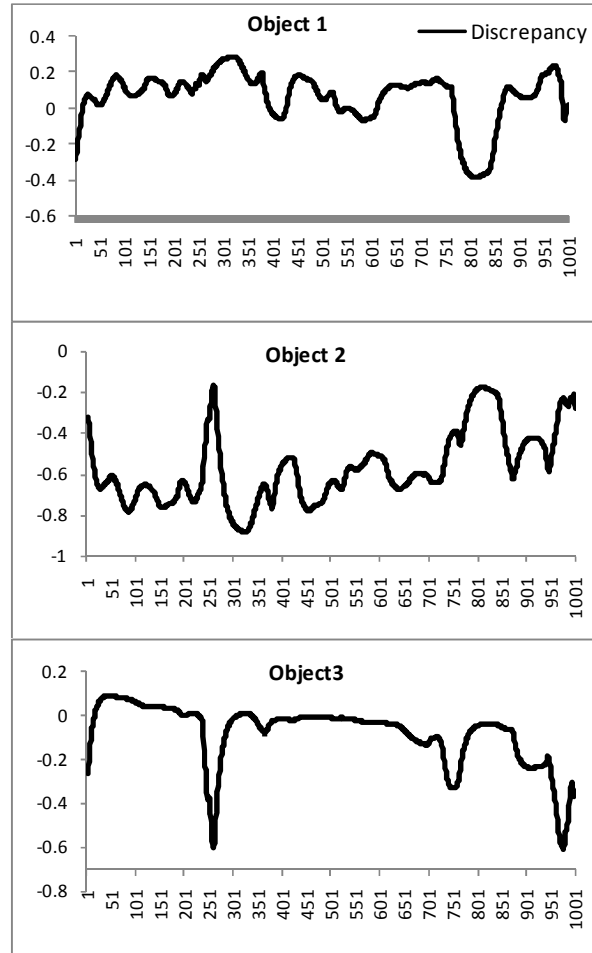


Figure 9. Estimated discrepancies between urgency and attention for the three objects

Due to these apparently inadequate parameter settings, the ambient software agent initially has a false awareness of the person's attention; for the agent it is as if the person pays enough attention to the enemy object but this is not the case in reality. By the adaptation process the initially inadequate parameter values for the brightness and size weight factors are changing to more appropriate values: from 0.2 vs 0.9 to around 0.8 vs 0.2 , as shown in Figure 10. As a result of this adaptation, the discrepancies shown in Figure 9 show a more faithful representation of the situation after, say time point 100 or 150: the discrepancy for object 1 is estimated as rather high positive, and for object 2 strongly negative. This gives the ambient software agent an appropriate awareness of the attention of this person who is almost all the time looking at the wrong (friendly) object, and does not seem to notice the enemy object.

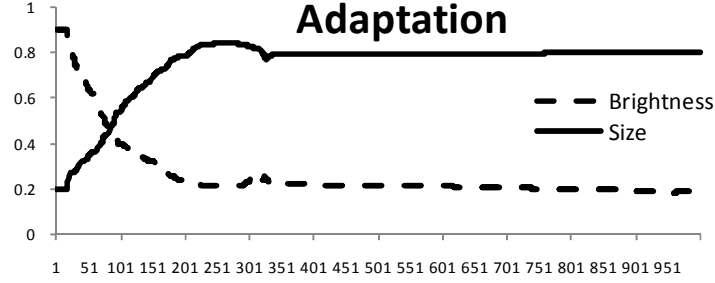


Figure 10. Adaptation of the parameter values for the brightness weight factor (from 0.9 to around 0.2) and size weight factor (from 0.2 to around 0.8)

10 Evaluation

In order to evaluate whether the agent functions as expected, an automated analysis of dynamic properties has been performed. In this analysis, a number of dynamic statements that were expected to hold for the agent have been formalised in the language TTL [4], and have been automatically verified against simulation traces (using Matlab). The predicate logical language TTL supports formal specification and analysis of dynamic properties, covering both qualitative and quantitative aspects. TTL is built on atoms referring to *states* of the world, *time points* and *traces*, i.e. trajectories of states over time. In addition, *dynamic properties* are temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, the state in γ at time point t is denoted by $\text{state}(\gamma, t)$. These states can be related to state properties via the formally defined satisfaction relation denoted by the infix predicate \models , comparable to the Holds-predicate in the Situation Calculus: $\text{state}(\gamma, t) \models p$ denotes that state property p holds in trace γ at time t . Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as \neg , \wedge , \vee , \Rightarrow , \forall , \exists . A first dynamic property that was expected to hold for the agent (expressed as P1 below) addresses the adaptation of parameter values:

P1 - Smaller range of adaptation over time

For all traces γ , and all parameters p

the difference between the highest and the lowest value of p is larger in the first part of γ than in the second part.

$$\begin{aligned}
 P1 \equiv & \forall \gamma: \text{TRACE} \forall p: \text{PARAMETER} \forall x1, x2, y1, y2: \text{REAL} \\
 & \max(x1, p, \gamma, 0, \text{end_time}/2) \ \& \ \min(x2, p, \gamma, 0, \text{end_time}/2) \ \& \\
 & \max(y1, p, \gamma, \text{end_time}/2, \text{end_time}) \ \& \ \min(y2, p, \gamma, \text{end_time}/2, \text{end_time}) \\
 \Rightarrow & \ x1 - x2 > y1 - y2
 \end{aligned}$$

Here, end_time denotes the last time point of the simulation trace, and \max and \min are defined as follows:

$$\begin{aligned}
 \max(x: \text{REAL}, p: \text{PARAMETER}, \gamma: \text{TRACE}, \text{tb}: \text{TIME}, \text{te}: \text{TIME}) \equiv \\
 \exists t: \text{TIME} \\
 & \text{state}(\gamma, t) \models \text{has_value}(p, x) \ \& \ \text{tb} \leq t \leq \text{te} \ \& \\
 & \neg [\exists t': \text{TIME} \exists x': \text{REAL} \ \text{state}(\gamma, t') \models \text{has_value}(p, x') \ \& \ \text{tb} \leq t' \leq \text{te} \ \& \ x' > x] \\
 \min(x: \text{REAL}, p: \text{PARAMETER}, \gamma: \text{TRACE}, \text{tb}: \text{TIME}, \text{te}: \text{TIME}) \equiv \\
 \exists t: \text{TIME} \\
 & \text{state}(\gamma, t) \models \text{has_value}(p, x) \ \& \ \text{tb} \leq t \leq \text{te} \ \& \\
 & \neg [\exists t': \text{TIME} \exists x': \text{REAL} \ \text{state}(\gamma, t') \models \text{has_value}(p, x') \ \& \ \text{tb} \leq t' \leq \text{te} \ \& \ x' < x]
 \end{aligned}$$

Automated checks pointed out that this property indeed holds for all generated traces. For example, for the first half of the trace shown in Figure 10, it turns out that the parameter ‘brightness’ varies between 0.18 and 0.90 (a difference of 0.72), whereas for the second half of the trace it only varies between 0.20 and 0.17 (a difference of 0.03). Similarly, the parameter ‘size’ varies between 0.20 and 0.84 in the first half of that trace (a difference of 0.64), and between 0.79 and 0.80 in the second half (a difference of 0.01). In addition to checking whether the parameter adaptation is performed correctly, it is interesting to check whether the accuracy of the model improves over time due to these parameter adaptations. To this end, one basically needs to check whether the estimated attention corresponds more to the actual attention over time. Since no information about the actual attention is available, the estimated attention is compared to the mouse clicks of the participant. In particular, the amount of *hits*, *misses*, *false alarms*, and *correct rejections* are counted, similar to signal detection approaches [13]. For the current purposes, these notions are defined as follows:

hit - The model estimates a *high* attention level for contact *c* at time point *t*, and indeed the participant *does click* on this contact within *d* time points

miss - The model estimates a *low* attention level for contact *c* at time point *t*, whilst the participant *does click* on this contact within *d* time points

false alarm - The model estimates a *high* attention level for contact *c* at time point *t*, whilst the participant *does not click* on this contact within *d* time points

correct rejection - The model estimates a *low* attention level for contact *c* at time point *t*, and indeed the participant *does not click* on this contact within *d* time points

These cases have been summed over all time points and all contacts. Within TTL, the following properties have been formalised to count the above cases (where *th* is a threshold to estimate high attention, and *D* is a constant to represent response time, which was taken 500 msec):

hit $\text{hit}(\gamma:\text{TRACE}, t:\text{TIME}, x:\text{CONTACT}) \equiv$
 $\exists t':\text{TIME} \exists i:\text{real}$
 $\text{state}(\gamma,t) \models \text{belief}(\text{has_value}(\text{av_for}(x), i)) \ \& \ i > \text{th} \ \&$
 $\text{state}(\gamma,t') \models \text{belief}(\text{clicked_on}(x)) \ \& \ t < t' \leq t+D$

miss $\text{miss}(\gamma:\text{TRACE}, t:\text{TIME}, x:\text{CONTACT}) \equiv$
 $\exists t':\text{TIME} \exists i:\text{real} \ \text{state}(\gamma,t) \models \text{belief}(\text{has_value}(\text{av_for}(x), i)) \ \& \ i \leq \text{th} \ \&$
 $\text{state}(\gamma,t') \models \text{belief}(\text{clicked_on}(x)) \ \& \ t < t' \leq t+D$

false alarm $\text{false_alarm}(\gamma:\text{TRACE}, t:\text{TIME}, x:\text{CONTACT}) \equiv$
 $\exists i:\text{real} \ \text{state}(\gamma,t) \models \text{belief}(\text{has_value}(\text{av_for}(x), i)) \ \& \ i > \text{th} \ \&$
 $\neg \exists t':\text{TIME} [\text{state}(\gamma,t') \models \text{belief}(\text{clicked_on}(x)) \ \& \ t < t' \leq t+D]$

correct rejection $\text{correct_rejection}(\gamma:\text{TRACE}, t:\text{TIME}, x:\text{CONTACT}) \equiv$
 $\exists i:\text{real} \ \text{state}(\gamma,t) \models \text{belief}(\text{has_value}(\text{av_for}(x), i)) \ \& \ i \leq \text{th} \ \&$
 $\neg \exists t':\text{TIME} [\text{state}(\gamma,t') \models \text{belief}(\text{clicked_on}(x)) \ \& \ t < t' \leq t+D]$

By counting these occurrences, one can calculate the sensitivity and specificity of the model as follows:

$$\text{sensitivity} = \text{hits}/(\text{hits}+\text{misses})$$

$$\text{specificity} = \text{correctrejections}/(\text{correctrejections}+\text{falsealarms})$$

In principle, an accurate model has a high sensitivity and a high specificity. However, these values also depend on the choice of the threshold *th*. An extremely low threshold (of 0) always results in a sensitivity of 1 and a specificity of 0, whereas an extremely high threshold (of 1) always results in a sensitivity of 0 and a specificity of 1. Therefore, the accuracy should be determined with the threshold as a variable. To this end, an ROC (Relative Operative Characteristic) analysis has been performed [11].

The results of this analysis are shown in Figure 11. This figure shows the ROC curves for an example situation in which our agent was applied. The blue curve indicates the start of the scenario, in which the parameters were not yet tuned (in particular, the first 12000 entries of the log file), whereas the red curve indicates the end of the scenario, in which the parameters were tuned (the last 12000 entries). The green curve indicates the results of a baseline agent which makes random estimations with respect to attention. Each curve contains 11 data points (representing 11 different values for *th*). As shown in Figure 11, the model produces much better results after parameter adaptation than before.

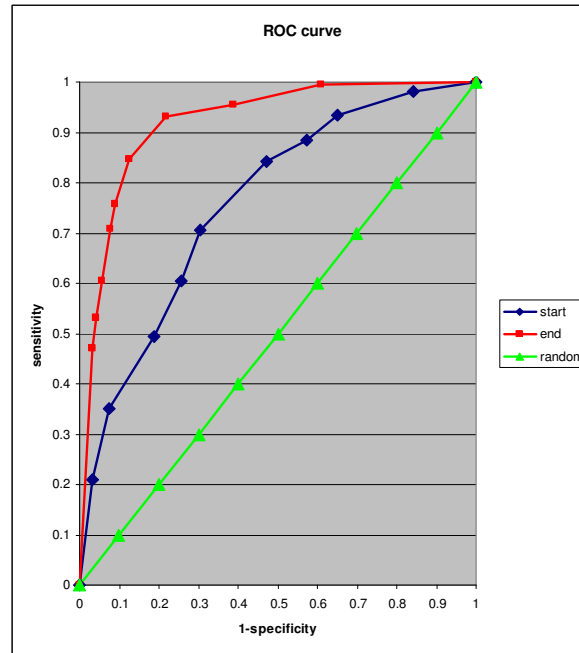


Figure 11. ROC curves for the model in two different stages, and a random estimation.

11 Discussion

To function in a knowledgeable manner, ambient or pervasive support systems (e.g., [1, 2, 20]) need to perform some form of mindreading (e.g., [9, 12, 13]) to obtain a model of the human(s) they are supporting. The software environment presented here focuses on mindreading concerning a human's attention states (e.g., [7, 15, 17, 19, 22]), based on information acquired by sensing of the gaze, features of the relevant objects in the environment, and a dynamical model based on differential equations integrating all this information. For the attention estimation the software environment adopts the model described in [7], which was also used in [5] and [6]. In contrast to [5] and [6], in the approach presented here the selection of intervention actions (as summarised in Table 3) is based on numerical approximation methods using sensitivity factors; these numerical methods are different from and more generally applicable than the analytical approach used in [5] and [6], where within the action generation process no differential equation format for (partial) persistency of attention levels is incorporated and no dynamic comparison between action options is made.

In addition to the variables for which values are calculated over time, typically such a dynamical model involves a number of parameters that represent characteristics of the human and the environment. As often individual differences between humans exist, a major challenge here is how the agent can acquire appropriate beliefs on values for parameters representing human characteristics that indeed reflect the specific person involved. Sometimes personal characteristics of a human can be determined at forehand by means of questionnaires and/or interviews. However, such methods do not guarantee appropriate outcomes, as what a person says he or she does is not always the same as what a person actually does. Therefore the option to estimate such parameters (e.g., [18], [21]) may be a better direction to explore. If such parameter estimation is performed by the agent at runtime, this results in an adaptive agent that over times gets more accurate beliefs on the characteristics of the human.

The software environment was implemented according to a component-based design within the Adobe® Flex® development environment, which makes it easy to adapt. Interaction between components was implemented using ActionScript, in an event-driven manner. A sensing component was included for the gaze sensing which connects to the Tobii® eye-tracker. Initially, a prototype implementation of the simulation was also carried out in Microsoft Excel®

The model was evaluated through a number of experiments, some of which were discussed in the paper. It was shown that after intervention actions, attention for urgent objects was higher. Moreover, by a formal

verification process it was shown for this case study that the adaptive software agent satisfies a number of expected properties.

The software agent with a capability to adapt to personal human characteristics as presented here may be the basis for applications in Ambient Intelligence or Pervasive Computing (e.g., [1], [2], [3]), where it is assumed that computing takes place in the background without addressing the human by an explicit interaction. When such agents can perform some forms of mindreading based on observations and dynamical models they possess, they can indeed act in the background without directly addressing the human. Some steps in this direction for the case of attention-reading were explored in [5] and [6]; however these approaches are not adaptive: parameter values need to be given initially. Another approach, not on attention-reading but on emotion-reading can be found in [8]. Although there parameter adaptation takes place, this is only for a simple case where only two parameters are adapted and where an analytical approach for parameter adjustment was used as the differential equation could be solved analytically to obtain an explicit formula for the sensitivity. As far as the authors know, no other approaches in the literature present attention-reading models that are integrated with mechanisms for parameter adaptation.

Despite the encouraging outcomes, care should be taken not to over-generalise the results. Although the verification of dynamic properties pointed out that the model performed better than a model making random estimations, one should keep in mind that these results were obtained in an experiment that involved only few participants in one specific scenario. In future research, more extensive experiments involving more participants will be conducted. Future work will address the combination of this model for attention with a model that estimates the human's work pressure and exhaustion and its effect on the attention.

References

1. Aarts, E.; Collier, R.; Loenen, E. van; Ruyter, B. de (eds.) (2003). Ambient Intelligence. Proc. of the First European Symposium, EUSAI 2003. Lecture Notes in Computer Science, vol. 2875. Springer Verlag, 2003, pp. 432.
2. Aarts, E., Harwig, R., and Schuurmans, M. (2001). Ambient Intelligence. In: P. Denning (ed.), *The Invisible Future*, McGraw Hill, New York, pp. 235-250.
3. Bosse, T., Hoogendoorn, M., Klein, M., and Treur, J., A Component-Based Ambient Agent Model for Assessment of Driving Behaviour. In: F.E. Sandnes et al. (eds), Proc. of the 5th International Conference on Ubiquitous Intelligence and Computing, UIC'08. Lecture Notes in Computer Science, vol. 5061. Springer Verlag, 2008, pp. 229-243.
4. Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J. (2009). Specification and Verification of Dynamics in Cognitive Agent Models. *International Journal of Cooperative Information Systems*, vol. 18, 2009, pp. 167-193
5. Bosse, T., Lambalgen, R. van, Maanen, P.P. van, and Treur, J., Attention Manipulation for Naval Tactical Picture Compilation. In: Baeza-Yates, R., Lang, J., Mitra, S., Parsons, S., Pasi, G. (eds.), *Proceedings of the 9th IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT'09*. IEEE Computer Society Press, 2009, pp. 450-457.
6. Bosse, T., Lambalgen, R. van, Maanen, P.-P. van, Treur, J., Automated Visual Attention Manipulation. In: L. Paletta, Tsotsos, J.K. (eds.), *Attention in Cognitive Systems*, Proc. of the Fifth International Workshop on Attention in Cognitive Systems, WAPCV'08. Lecture Notes in Artificial Intelligence, vol. 5395. Springer Verlag, 2009, pp. 257-272.
7. Bosse, T., Maanen, P.-P. van, Treur, J., Simulation and Formal Analysis of Visual Attention, *Web Intelligence and Agent Systems Journal*, vol. 7, 2009, pp. 89-105.
8. Bosse, T., Memon, Z.A., and Treur, J., Adaptive Estimation of Emotion Generation for an Ambient Agent Model. In: Aarts, E., Crowley, J.L., Ruyter, B. de, Gerhäuser, H., Pflaum, A., Schmidt, J., Wichert, R. (eds.), *Ambient Intelligence, Proceedings of the Second European Conference on Ambient Intelligence, AmI'08*. Lecture Notes in Computer Science, vol. 5355. Springer Verlag, 2008, pp. 141-156.
9. Bosse, T., Memon, Z.A., and Treur, J., A Two-Level BDI-Agent Model for Theory of Mind and its Use in Social Manipulation. In: *Proceedings of the AISB 2007 Workshop on Mindful Environments*, 2007, pp 335-342.
10. Bosse, T., Memon, Z.A., Treur, J., and Umair, M., An Adaptive Human-Aware Software Agent Supporting Attention-Demanding Tasks. In: Yang, J.-J.; Yokoo, M.; Ito, T.; Jin, Z.; Scerri, P. (eds.), *Proceedings of the 12th International Conference on Principles of Practice in Multi-Agent Systems, PRIMA'09*. Lecture Notes in Artificial Intelligence, vol. 5925. Springer Verlag, 2009, pp. 292-307
11. Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27, 861-874.

12. Gärdenfors, P. (2003). *How Homo Became Sapiens: On The Evolution Of Thinking*. Oxford University Press, 2003.
13. Goldman, A.I. (2006). *Simulating Minds: the Philosophy, Psychology and Neuroscience of Mindreading*. Oxford University Press.
14. Green, D.M., Swets J.A. (1966) *Signal Detection Theory and Psychophysics*. NY: Wiley.
15. Itti, L. and Koch, C., *Computational Modeling of Visual Attention*, *Nature Reviews Neuroscience*, vol. 2, no. 3, 2001, pp. 194-203.
16. Memon, Z.A., Oorburg, R., Treur, J., Umair, M., and Vos, M. de., *A Software Environment for a Human-Aware Ambient Agent Supporting Attention-Demanding Tasks*. In: *Proceedings of the Tenth International Conference on Computational Science, ICCS'10*. *Procedia Computer Science Series*, Elsevier, 2010, to appear.
17. Parkurst, D., Law, K., and Niebur, E. *Modeling the role of salience in the allocation of overt visual attention*. *Vision Research Journal*, vol. 42, 2002, 107-123.
18. Pearson, C.E., 1986. *Numerical Methods in Engineering and Science*. CRC Press.
19. Port, R.F., and Gelder, T. van (eds.), 1995. *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge, Mass.
20. Riva, G., F. Vatalaro, F. Davide, M. Alcañiz (eds.) (2005). *Ambient Intelligence*. IOS Press.
21. Sorenson, H.W. (1980). *Parameter estimation: principles and problems*. Marcel Dekker, Inc., New York
22. Turatto, M., and Galfano, G., *Color, form and luminance capture attention in visual search*. *Vision Research*, 40, 2000, 1639-1643.