Intelligent Prompting System to Assist Stroke Survivors

Jean-Baptiste, Emilie; Russell, Martin; Howe, Joanne; Rotshtein, Pia

DOI:
10.3233/AIS-170461

Document Version
Peer reviewed version

Citation for published version (Harvard):
Jean-Baptiste, E, Russell, M, Howe, J & Rotshtein, P 2017, 'Intelligent Prompting System to Assist Stroke Survivors', *Journal of Ambient Intelligence and Smart Environments*, vol. 9, no. 6, pp. 707-723. https://doi.org/10.3233/AIS-170461

Intelligent Prompting System to Assist Stroke Survivors

Emilie M. D. JEAN-BAPTISTE [a,1], Joe HOWE [b], Pia ROTSHTEIN [b] and Martin RUSSELL [a]

[a] *School of Electrical, Electronic and System Engineering, University of Birmingham, Birmingham, UK. E-mails:{emj198@alumni.bham.ac.uk, m.j.russell@bham.ac.uk}*
[b] *School of Psychology, University of Birmingham, Birmingham, UK. E-mails: {howej, p.rotshtein}@bham.ac.uk*

Abstract. Stroke survivors often have difficulties performing Activities of Daily Living (ADL). When trying to complete a task, they tend to rely on caregivers who give them cues when necessary. However, this reliance on caregivers' support may affect their ability to live independently. In our study, we have developed CogWatch - an assistive system designed to provide guidance to stroke survivors during tea-making. An evaluation of the system was carried out on twelve patients. The latter were asked to complete different ADLs with and without CogWatch's assistance. Results showed that patients succeed at the tasks more often when assisted by the system than without guidance. It is anticipated that this system could be installed in the home environment and provide early stage rehabilitation.

Keywords. Stroke survivors, Activities of daily living, Action planning, MDP, Assistive technology

1. Introduction

There are more than 100,000 new stroke cases every year in the UK, with over half of all stroke survivors depending on others to carry out Activities of Daily Living (ADL) due to loss of physical and cognitive functions [1]. The loss of independence has a direct impact on the patients, relatives, carers and society as a whole. The cost of stroke to society is estimated to be £8.9 billion a year, with about half linked to indirect costs of on-going support [2]. Difficulty in completing ADL due to cognitive deficits is estimated to affect 46% of stroke survivors [3]. Errors during ADL relate to defective use of real tools and objects [4], the inability to correctly select appropriate tools for a task [5], or the inability to complete sequences of actions [6]. These impairments are associated with loss of action knowledge [7], attention and executive function deficits [8], and/or loss of object knowledge [9]. The aim of the current project is to develop a technology that can support stroke survivors during everyday tasks, such as making a cup of tea. Technology is widely used in physiotherapy and its efficacy is well researched [10]. In physiotherapy, it is used for:

[1] Corresponding author. E-mail: emj198@alumni.bham.ac.uk

1. Assessing patient's functionality and improvement [11],
2. Administering rehabilitation tailored to the patient's needs [12],
3. Providing on-going assistance and feedback [13].

When systems are able to monitor patients, they also have the potential to reduce the number of consultations with specialists, assess disease progression and evaluate medication effects [14]. In the arena of home-based monitoring of chronic diseases several studies have been carried out, in particular [15, 16]. In [15], the authors proposed a monitoring device able to provide self-assessments and motor tests. Cued and un-cued tests were designed to collect data on upper limb conditions. The data were then analyzed to evaluate user's compliance and the usability of the device. Based on similar methods, another home-based assessment tool was presented by Cunningham et al. in [16]. Their novel approach ensured that users did not need to wear any sensorized objects to be monitored. The computer-based device was designed to collect data on hand and finger movements. Their results showed that the data collected could distinguish between the states where the users' medication was working at its best and when it had worn off.

Monitoring home-based systems can be enhanced with Artificial Intelligence (AI) methods and automatic planning techniques. In such a case, these systems do not only collect data on patients' state but also provide reminders or guidance in order to help them during ADL. The increased interest in this field led to the development of systems such as Autominder [17], Guide [18] and COACH [19]. In [17], the authors described a cognitive orthotic system - the Autominder. The latter is designed to provide personalized reminders of ADL to older adults. To achieve its goal, the system tries to maintain a correct representation of the user's daily plan, monitor its execution, and plan reminders accordingly. After being specified, the user's plan is updated through the day. High quality reminders are then generated by an intelligent planning system based on a Planning-by-Rewriting approach [20]. Implemented as a scheduling aid, the Autominder reminds users what activities should be done through the day. However, it does not help them to correctly go through each of these activities. Another example is COACH [19], which has been designed to provide instructional cueing in order to guide users during one specific task: hand-washing. The system uses a Markov Decision Process (MDP) based planning system to provide prompts when needed. To fulfill this goal, it implements a hand-tracking module that needs to capture enough information about the user's environment and behavior during the task. However, in [21], the authors highlighted the fact that COACH did not track user progress well due to the poor performance of the tracking module. Moreover, the process of generating an accurate model of a task for COACH is extremely complex, time consuming and difficult to generalize to other activities [22].

Similarly to COACH, Guide [18] is an assistive system for cognition that focuses on one specific task and provides verbal prompts. In [18], authors described how Guide provides guidance to amputees with cognitive impairments when putting on their prosthetic limbs. To do so, the system interacts with users through speech. It asks a series of questions to users about their task completion and provides prompts based on their answers. Thus unlike COACH, Guide relies on users' verbal input to infer their current state. While this can overcome challenges related to the accuracy of automatic detection of actions, it cannot be used with users who have speech impairments (i.e., aphasia). Such impairments are a common problem of stroke survivors [23, 24] with high comorbidity of praxis (difficulty in complex manual movement) and aphasia [25]. In such a case, it is reasonable to consider that some users may be reticent to have to verbally interact with

an assistive system in order to make the later be aware of their task completion so they can receive its guidance accordingly [26].
In this paper, we introduce a novel MDP-based assistive system technology: CogWatch. Its aim is to provide meaningful instructional cues to stroke survivors during tea-making. Compared to other systems previously described, the model on which CogWatch is based has been designed to be easily adaptable to different ADLs and can automatically detect when users need assistance. Indeed, the planning module implemented in CogWatch is flexible, and allows the system to provide guidance during other tasks, such as teeth-brushing. Note that in this paper, results will only be given on the system's ability to successfully guide users during tea-making.
For CogWatch to fulfill its goals, it monitors the users in a sensorized environment, tracks their behavior, and infers their potential need for guidance in order to retrieve adequate prompts. A prompt is considered to be adequate when (1) it is a valid continuation of what the user has achieved so far (i.e., the user state), and (2) it is retrieved because an incorrect pattern in the user's behavior (i.e., user's error) is detected by the system. Hence, the assistive system implements an Artificial Intelligent Planning System (AIPS) that is composed of at least two modules: (1) the Action Policy Module (APM), which ensures that cues are a valid continuation of the user state, and (2) an Error Recognition Module (ERM), whose aim is to detect potential user errors during the task. This means that CogWatch needs to be able to plan, potentially under uncertainty, to retrieve the right strategies to the user, and to do so at the right time. Planning is the core of many studies focusing on automated sequential decision-making. Many researchers, in different fields, have addressed this problem by using a Markov Decision Process (MDP) framework [27, 28] as an underlying model for planning or recommendation. Even though assumptions can vary between fields, the MDP framework is powerful enough to embrace many different applications: [29, 30, 31, 32, 33].

This paper will focus on the structure of the APM and ERM. We propose a novel MDP state representation for ADL management, and report the ability of CogWatch to guide a simulated user ($SimU$) and twelve real patients during tea-making. The paper is structured as follows. First an overview of CogWatch is given, then the technique used for its APM and ERM to fulfill their goals is explained. The viability of our framework is also analyzed, by comparing users' performance with and without the help of the system's outputs. Finally, the limitations of the framework and future plans are discussed.

2. Assistive System Architechture

Figure 1 depicts the main modules composing CogWatch. It works as follows. First, the user chooses the type of tea (e.g., "Black tea with sugar") he/she would like to receive training for. Immersed in an instrumented environment, the user moves sensorized objects at his/her disposal to perform an action (denoted by a_u in the figure). When moved, these objects generate data that are passed to a module called the Action Recognition System (ARS), whose aim is to recognize the user action. Processing the data in real-time, the ARS outputs an observation o that it communicates to the AIPS. The latter is in charge of selecting the prompts a^*_m composed of: (1) the best next action a^*_s the system considers the user should follow to successfully continue the task, and (2) its best understanding of whether the user has just made an error or not e^*_s; and if an error has

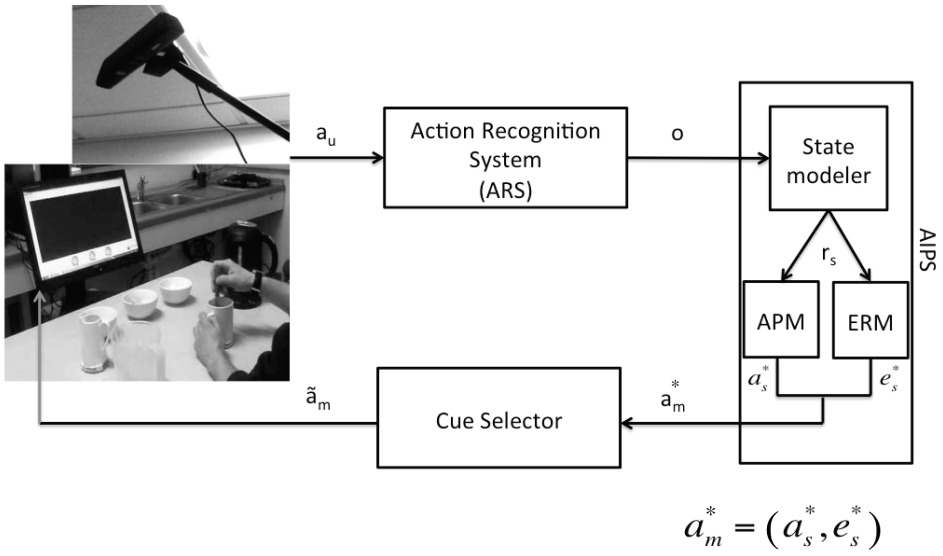

Figure 1. CogWatch architecture

been made, what was the type of this error. Each AIPS prompt a_m^* is then passed to the Cue Selector, which displays, when necessary, the AIPS output in a form of a cue that the user can easily understand $\tilde{a}_m$ (e.g., still images, video, recorded message).

Figure 2 shows the graphical interface connected to the Cue Selector. It has been designed for the user to easily interact with CogWatch. Through this interface, the user has access to:

- A history of completed actions, provided by the AIPS and displayed in the progress bar,
- A visual reminder of the type of tea currently selected (i.e., task selection),
- A “task completion button" that can be pressed if the user considers that the task is finished,
- A “help button" that can be pressed if the user needs to receive the AIPS prompt (it will cause a cue to be generated based on the best prompt a_m^*),
- A “repeat button" that can be pressed if the user needs to repeat the last prompts,
- A screen which displays a video of the prompt if necessary.

After each action, the user can continue his/her interaction with the system and enter in new cycles until the task is completed. For the AIPS to guide the user properly, it needs to learn the optimal action the user should perform at any stage of the task. To do so, it acts like an $agent$, and interacts with its environment by taking actions and receiving costs. In the long run, its aim is to take actions that lead to the lowest expected long-term cost. As one can see in Figure 1, when the AIPS receives an observation from the ARS, one of its modules called the *State Modeler* uses this information to re-create a representation r_s of the system’s environment. In CogWatch, the system’s environment is what the user has achieved so far (i.e., the user state). The representation of the user’s state is then passed to the APM and the ERM which need to decide what is the optimal prompt to be

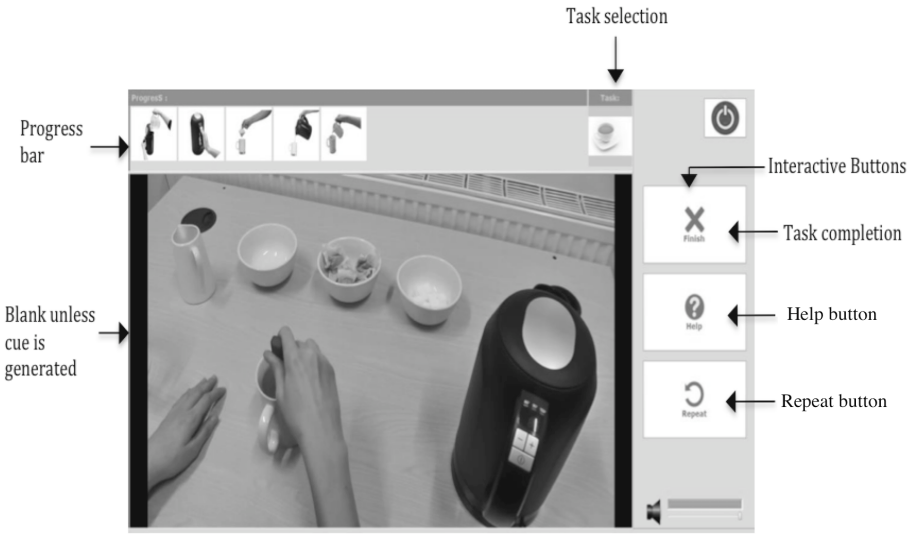

Figure 2. CogWatch patient's screen

sent to the user. It is important to highlight that both modules may have an incomplete understanding of the user state. In such a case, they may be forced to base their outputs on a probabilistic estimation of this state. Various reasons can explain why an $agent$ does not necessarily have access to a complete understanding of the user's state:

1. The sensors used to gather information about the environment can be noisy,
2. The ARS can make errors and output wrong information to the AIPS,
3. The effect of the $agent$ actions on the environment can be unpredictable (e.g., user's compliance to prompts),
4. Many events beyond the $agent$'s control can arise, such as exogenous events.

Note that in the case of the prototype presented in this paper, each time an ARS observation did not correspond to the real user action, a clinician had the possibility to modify this observation in real-time via a specific touch-screen. In other words, the ARS observations could be corrected before letting them reach the AIPS. Thus, as it was made possible to prevent the AIPS from receiving wrong information about the user's behavior, we made the assumption that the user state was fully observable from its point of view. This case can be modeled by considering that each observation o output by the ARS corresponds to the real user action a_u. This assumption implies that the AIPS representation of the user state r_s exactly corresponds to the user state. Thus, a fully observable MDP framework can be applied to model the AIPS decision-making process.

3. The tea-making task

Although the principles underlying the CogWatch system are applicable to a range of ADL, the CogWatch system focuses on tea preparation. The system provides assistance

for making four types of tea, namely “Black tea”, “Black tea with sugar”, “White tea” and “White tea with sugar”. In each case, hierarchical task analysis [34] was employed to represent the task as a hierarchical tree, where the first decomposition of the task is into “actions”. For tea-making, the actions fall into two sets:

- Φ = {“Fill Kettle”, “Boil Water”, “Pour Kettle", “Add Teabag”, “Add Sugar”, “Add Milk”, “Stir”, “Remove Teabag”}, and
- Ψ = {“Pour Cold Water into Mug”, “Toying”},

where Φ is the set of true actions related to tea preparation and Ψ comprises common errors that are of special interest.

4. Action Recognition System

4.1. Aims

The aim of the Action Recognition System (ARS) is to detect hand position data via Kinect^{TM} [35], and user actions from the outputs of the sensors attached to the objects used during the task. For the tea-making task, the cup, milk jug and kettle are each fitted with an instrumented “coaster”. This is an electronic device that fits on the base of an object, which contains a three-axis accelerometer, three force-sensitive resistors (FSRs), a microcontroller and a Bluetooth module for wireless transmission of the sensor outputs. Fifty times per second, each sensor module outputs a raw 6 dimensional feature vector comprising (x, y, z) acceleration values and the outputs of the three FSRs. The accelerometer values indicate motion and changes in the orientation of objects, and the FSRs measure changes in weight and whether or not the object is standing on a surface. The ARS must be sufficiently flexible to accommodate variations between the sequences of sensor outputs that result when different people perform the same task, or the same person performs the same task on different occasions. Hidden Markov models (HMMs) are general purpose statistical models of sequential data [36]. In the CogWatch ARS, HMMs are applied at the actions level, with a single HMM for each element of the sets Φ and Ψ.

4.2. Structure

The ARS in CogWatch is implemented as a parallel set of action detectors (Figure 3). The left-hand side of Figure 3 shows the current feature vector, comprising synchronized outputs from the sensors attached to the mug ($[a_{m,x}, a_{m,y}, a_{m,z}, f_{m,1}, f_{m,2}, f_{m,3}]$), jug and kettle and hand coordinate data. The middle column of the figure depicts the array of action detectors, one for each element of $\Phi \cup \Psi$. Each detector takes as input those parts of the feature vector that are useful for detecting its action (i.e., Figure 3 focuses on the action “Add milk”). An action detector consists of a multiple state HMM representing that action, whose states are associated with Gaussian mixture models (GMMs). In addition, the detector includes a single state “background” (or “toying”) HMM, whose state is associated with a multiple-component GMM. An identical implementation of the Viterbi algorithm [36] runs independently in each decoder. Briefly, each detector works as follows: At each time t the detector receives a new feature vector, y_t. For each state i of each of its HMMs, a quantity $\alpha_t(i)$ is calculated which can be thought of as an approx-

imation to the probability of the best explanation of data $y_1, ..., y_t$ up to and including y_t ending in state i at time t. Intuitively, if the detector is for "Add Milk" and the i^{th} state corresponds to tipping the jug, then $\alpha_t(i)$ can be thought of as the probability of the best explanation of data up to time t culminating in the tipping action at t.

In the basic implementation of Viterbi decoding described above, the best explanation of the data is not recovered until the final time T. However, in a real-time implementation there is no final time. The memory required to store the $\alpha_t(i)$s will increase and no output will be produced. The solution is to use a technique called 'partial traceback' [37]. Each detector's output up to a time s is generated as soon as its classification of the data up to that point is unambiguous. The memory used to store alternative explanations of the data up to s is then freed. In this way the decoders can run indefinitely.
Whenever an action HMM provides the most probably explanation of a section of input, a label indicating that action is output. Otherwise the best explanation of the data is "toying" and nothing is output. The final column of Figure 3 shows the type of output that is sent by the ARS to the AIPS. More information about the ARS and how user actions are detected can be found in [35].

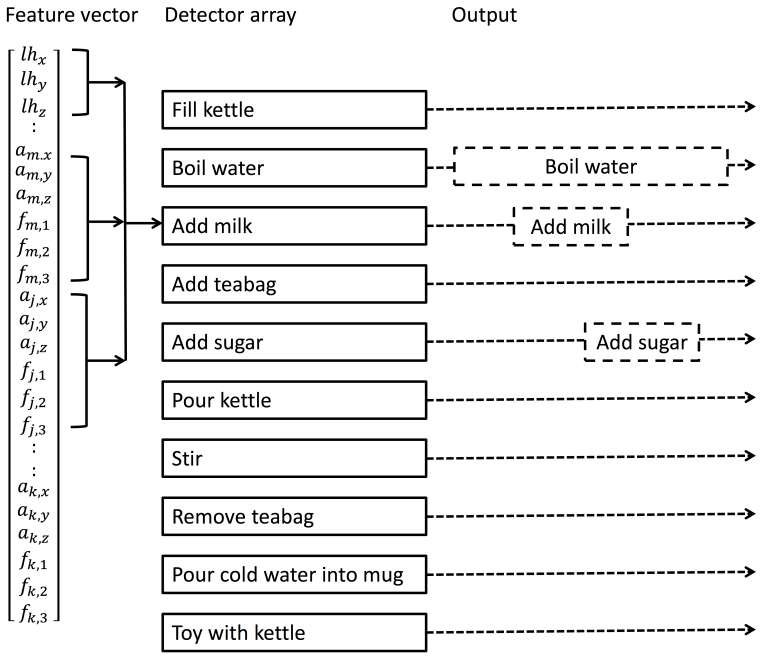

Figure 3. CogWatch ARS architecture

5. Action Policy Module

5.1. Aims

The first aim of the Artificial Intelligent Planning System (AIPS) is to take into account the information delivered by the ARS, and to plan the best next action the user should perform in order to successfully complete the task being carried out. The second aim of the AIPS is to detect when the user makes an error during the task, and in such a case, to find the specificities of this error. In this section, we focus on the ability of the AIPS to plan actions via its APM (see Figure 1).
To fulfil its goal, the APM is based on a Markov Decision Process (MDP) framework, which needs to be adapted to the system's specifications.

5.2. MDP overview and adaptation

Formally, a MDP is a tuple $\{S, A, T, C\}$ where:

- S is a set of AIPS states
- A is a set of prompts the AIPS can output
- T is a set of transition probabilities
  $T = \{T_{s',s,a_m}\}_{s',s \in S, a_m \in A}$
  where $T_{s',s,a_m} = P(s' \mid s, a_m)$
- C defines the immediate cost $c(s, a_m)$

The MDP operates as follows. At each step, the AIPS is in an observable state $s \in S$ (i.e., in this case, $r_s = s$). Taking into account what the user has achieved so far, the AIPS selects a prompt $a_m \in A$, then receives a cost $c(s, a_m)$ for doing so. In order to be applied to CogWatch and fit ADL management, some parameters of the MDP framework are specifically defined based on the system's ultimate goal:

The Machine's Action Space A: The AIPS prompt is of the form $a^*_m = (a^*_s, e^*_s)$ (i.e., $a^*_m \in A$), with a^*_s being the optimal action the APM can send to the user to guide him/her during the task, and e^*_s being the optimal ERM understanding of whether the user has made an error, and if it is the case what was this error (e.g., addition error, anticipation error). The Action Space is defined such as $A = \Phi \cup \Psi$.

The Machine's State Space S: This is a finite set of machine's states s. In this paper, one of our contributions is to define these machine's states as sequences of actions a_u (see Figure 1). Concretely, for the APM, a state represents the history of the user's most recent actions. However, a restriction technique is applied to the State Space, so the states only contain what psychologists consider as correct patterns of actions. This will be explained in the next subsection.

The Cost Function C: The cost function $c(s, a_m)$ is a mechanism to incorporate human judgment about the importance of different types of behavior into the MDP. Two types of functions were combined: one that allows the MDP to find the fastest strategy (i.e., shortest route to task completion) (i), and another one that takes into account the way participants successfully perform the task plus clinicians' preferences (ii). This decision has been taken due to the fact that when the fastest strategy may be valid, it is not necessarily psychologically plausible. The results that will be presented in section 7.1 show that when the cost function (i) is combined with relevant knowledge from users

n	user action	user state	APM state
0	$\emptyset$	[]	[]
1	Add teabag	[Add teabag]	[Add teabag]
2	Toying	[Add teabag, Toying]	[Add teabag]
3	Add water in the kettle	[Add teabag, Toying, Add water in the kettle]	[Add teabag, Add water in the kettle]
4	Add water in the kettle	[Add teabag, Toying, Add water in the kettle, Add water in the kettle]	[Add teabag, Add water in the kettle]

Figure 4. Example of state representation

and clinicians (i.e., cost function (ii)), it allows the AIPS to generate more meaningful and efficient strategies during the task. In the rest of this paper, note that when using the cost function (i) only, the AIPS strategies will be referred as Non-Psychologically Plausible; when using the combination (i) and (ii), the AIPS strategies will be referred as Psychologically Plausible [38].

5.3. State Space's restriction technique

In Figure 4, one can see an example showing how the user state evolves based on the actions made by the user, and how the restriction technique applied to S allows the APM to maintain a reduced state. Let $\tilde{s}$ be the APM state and s the user state at step n:

- At $n = 0$, both the APM and user state are empty : no action has been made by the user yet.
- At $n = 1$, the user makes action $a_{u,1}$ =“Add teabag”. As one can begin making a cup of tea by doing so, the action is valid, and the APM copies it: $\tilde{s}_1$=s_1 =[“Add teabag”].
- At $n = 2$, the user toys with the kettle, thus $a_{u,2}$ =“Toying”. The user state automatically takes this action into account, with s_2 =[“Add teabag”, “Toying”]. However, this action is not considered to be valid by psychologists because of the casualties it can cause. Thus, it is ignored by the AIPS and $\tilde{s}_2$ remains unchanged, with $\tilde{s}_2$ =[“Add teabag”].
- At $n = 3$, $a_{u,3}$ =“Add water in the kettle”, which is valid, so both $\tilde{s}_3$ and s_3 are updated accordingly.
- At $n = 4$, the user repeats the same action $a_{u,3}$. The user state takes it into account after update, but not the APM, as this action as already been performed. Indeed, such a redundancy is defined as an error by psychologists, and is not allowed in the APM state. Thus $\tilde{s}_4$ is equivalent to $\tilde{s}_3$.

In the rest of this paper, the reduced state $\tilde{s}$ will be referred as s, unless specified otherwise. This restriction technique allows the State Space to be maintained at a relative small size, while keeping enough information about the system's environment. Note that due to this technique, the State Space depends on the task for which it is defined, and on specialists' definition of errors. Let's take the example of a user choosing “White tea with sugar” at the beginning of the task. For this task to be completed, a minimum of six actions is required: “Fill Kettle”, “Boil Water”, “Pour Kettle", “Add Teabag”, “Add Sugar”, “Add Milk”, “Stir”, “Remove Teabag’ (N.B., not necessarily in this order). Hence, taking into account what psychologists working on CogWatch define as correct sequences of actions during this task, we find 1539 states (i.e., correct combination of actions) which

Algorithm 1 MC policy optimization algorithm - APM

for $s \in S, a \in A$ **do**
  $Q_0(s,a) \leftarrow$ arbitrary
  $N(s,a) \leftarrow 0$
  $\pi_0^*(s) \leftarrow \arg\min_a Q_0^*(s,a)$
end for
for each iteration k **do**
  for every $s \in S$ and $a \in A$ **do**
    Generate a trial starting at s with action a and proceeding according to $\pi_{k-1}^*(s)$ until a final state is reached.
    During the trial, record each pair (s', a'), compute the sum of the costs $c(s',a')$ following s until the final state, then update:
    $N(s',a') \leftarrow N(s',a') + 1$
    $Q_k^*(s',a') \leftarrow \frac{Q_{k-1}^*(s',a') * N(s',a') + c(s',a')}{N(s',a')}$
    $\pi_k^*(s) = \arg\min_a Q_k^*(s,a)$
  end for
end for

can be part of the State Space for this task. In COACH [19], for a similar sequential task (i.e., hand-washing), a different parameterization of the MDP framework led to over 22 million states.

5.4. Offline Training and MDP solving

To solve a MDP, a policy ($\pi^* : S \rightarrow A$) between the State Space S and the Action Space A needs to be found, such that $\pi^*(s)$ minimizes the expected cost incurred by a trial. To achieve this goal, the APM is trained offline using a Monte Carlo (MC) Algorithm [39]. During training, a state-action value function $Q(s, a_m)$ records the immediate cost for taking prompt a_m in state s, for all states contained in the State Space S. The Q function is defined as the expected cost of a trial starting in state s, the AIPS taking action a_m, and thereafter proceeding according to the current policy π^* until the trial ends and a final state is reached. During each tea trial, a sequence of state-action pairs $\langle s, a_m \rangle$ is recorded, and used to update the estimate $Q(s, a_m)$. At the end of the training, when the Q values are estimated, the optimal policy π^* is obtained with:

$$\pi^*(s) = \arg\min_{a_m} Q(s, a_m), \forall s \in S, \forall a_m \in A. \quad (1)$$

The policy optimization algorithm that was implemented is given in Algorithm 1 for the APM training. Algorithm 1 begins by the initialization of the Q values, the number of time the pair (s, a) is visited (i.e., $N(s, a)$), and the current optimal policy π_0^*. Then, for each trial the AIPS goes through, the states it visits and actions it takes are recorded. When the final state is reached, the cost is calculated for each state-action pair (s, a) recorded. The information is then used to update each corresponding $Q(s, a)$ value, $N(s, a)$, and the optimal policy π^*, which allows to select, in the next turn, the action that has accumulated the lowest cost for each state. This is repeated until convergence is reached.

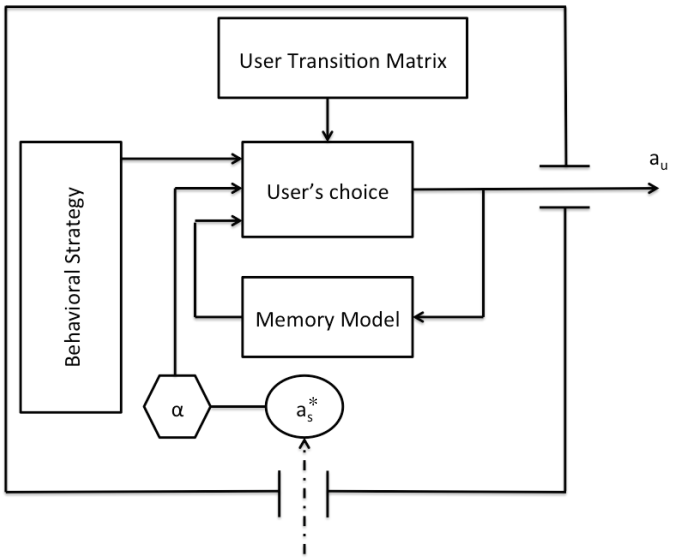

Figure 5. Simulated User architecture

Training can be done with real participants or with a simulated user. Since many trials are necessary to robustly estimate the Q values, it is common practice that a simulated user is used to interact with the system during policy optimization [40, 41, 42, 43, 44, 45, 46]. Hence, a Simulated User ($SimU$) based on real patients' data was implemented in order to generate actions [38]. This $SimU$ is composed of 5 main modules (Figure 5). The core module is the *User's choice*, which takes as inputs parameters from the *User Transition Matrix*, the *Memory Model*, the current APM best strategy a_s^*, and the *Behavioral Strategy* module. During training, when the $SimU$ generates an action a_u, the APM responds with a prompt a_m^*. The $SimU$ receives the optimal action a_s^* contained in a_m^* if the Cue Selector let it pass. If the Cue Selector does not communicate a_s^* to the $SimU$, the latter chooses what action to take by itself. If a_s^* is communicated, the $SimU$ takes it into account with a compliance α ($0 \leq \alpha \leq 1$), then selects its next action a_u. The $SimU$ uses a *User transition Matrix* based on action bigram probabilities from data generated by 52 control and stroke survivors, aged between 21 and 82, who completed four types of tea ("Black tea", "Black tea with sugar", "White tea", "White tea with sugar"). This module allows the $SimU$ to have some knowledge about how humans perform the tasks. This knowledge is then used to emulate the variability with which a task can be achieved. Nevertheless, as only bigram probabilities are taken into account, this knowledge is incomplete. Thus, when the $SimU$ needs to act by itself without relying on the prompts, and does not have enough information about what next action to take, it has access to three behaviors through the *Behavioral strategy* module:

1. The $SimU$ selects the "Task Completion Button" or the "Help Button" (see Figure 2). This will force the system and the Cue Selector to deliver the APM prompt to the $SimU$, which will comply to it.

2. The $SimU$ does nothing, which will trigger the AIPS after a specific amount of time, forcing the APM to deliver a prompt to which the $SimU$ will comply.
3. The $SimU$ performs a random but meaningful action. A meaningful action is an action that is not impossible for a human to perform. For example removing a teabag from a cup while the latter was not put in beforehand, is not a meaningful action.

The $SimU$ also has access to 5 different types of *memory* through the *Memory model* module. Each type of memory has a specific impact on how the $SimU$ remembers the history of actions it has already performed:

1. The $SimU$ only remembers the last action it performed.
2. The $SimU$ remembers all the actions it performed and can repeat some of them.
3. The $SimU$ remembers all the actions it performed and cannot repeat any.
4. The $SimU$ exponentially forgets the actions performed in the past (i.e., it has a higher probability to forget actions performed in the past) and cannot repeat any actions it still remembers.
5. The $SimU$ exponentially forgets the actions performed in the past and can repeat some of those it still remembers.

The risk with using a Simulated User is that it may not properly capture real users' behaviors. However, our $SimU$ is based on real user data, and in experiments, the task completion rate for the $SimU$ and real users is similar when interacting with the MDP-based system [47].
As shown in Figure 6, the *SimU* is implemented into a virtualization of CogWatch, which follows the same structure of the real system as described in Figure 1. In such an environment, when the task is chosen, the *SimU* starts in state s_0=$\emptyset$ (i.e., the user has not performed any action yet), and takes an action a_u that is sent to a virtual ARS. The virtual ARS is implemented as a $N \times N$ confusion matrix C (N is the number of actions) whose i, j^{th} entry is the probability that the ARS outputs observation j when the user executes action i. This observation is then passed to the AIPS. After receiving this observation, the AIPS sends its optimal prompt $a^*_m = (a^*_s, e^*_s)$ to a virtual Cue Selector. The latter is a filter; it sends $\tilde{a}_m = \{a^*_s || \Theta\}$ to the *SimU*, where a^*_s is the next best action the APM considers the user should perform, and Θ is a signal forcing the *SimU* to chose the next action by itself. In other words, each time the Cue Selector passes its output to the *SimU*, this output only contains a^*_s or Θ. If the ERM does not detect any error in the user's state, the signal Θ is sent; otherwise, it is the action a^*_s that the $SimU$ receives. This allows the system to communicate prompts to the user only when needed (i.e., when the user makes an error).

6. Error Recognition System

Contrary to the MDP-based APM, the ERM does not base its functionality on any reinforcement learning process. The state it takes into account directly corresponds to the user's state (see Figure 4), when no state restriction technique is applied. Indeed, only the full user state keeps track of the potential errors made by the user. Hence, during the task, if the ERM detects that a state is erroneous, its goal is to find what type of error it is. To do so, it takes as an input the user state, then automatically compares it with different

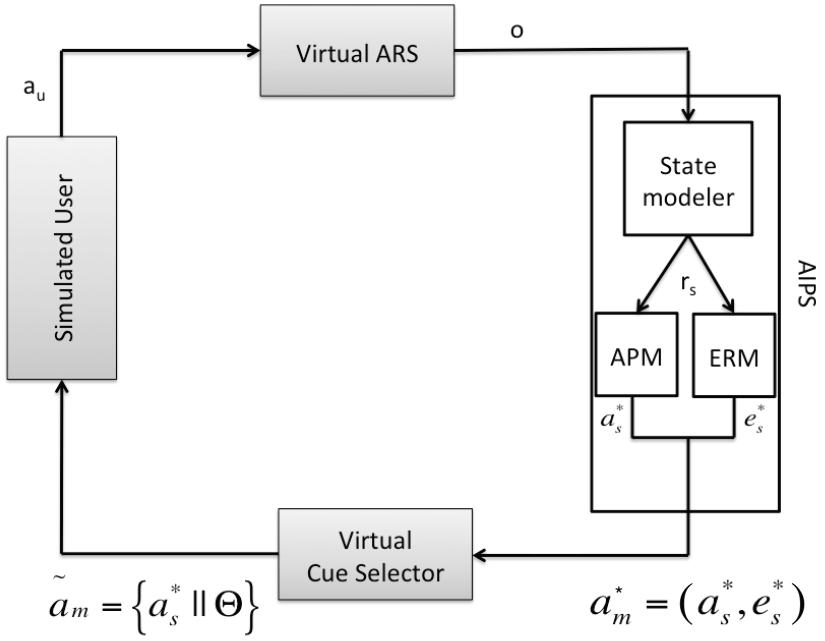

Figure 6. Virtual CogWatch architecture

rules that have been encoded based on psychologists' definitions of different types of errors. When a type of errors is detected, the ERM looks for the cause of this error (i.e., the action, order of actions or combination of actions, which led to this error). Once the cause is found, the ERM associates a label to this error corresponding to the specificity of its type.

Thus, in the MDP-based system described in this paper, when the ERM outputs e_s^*, the latter is a tuple containing two parameters: (*error bool*, *error ID*), with *error bool* being a Boolean which is $True$ if an error is detected and $False$ otherwise; *error ID* is the type of error detected. When output, e_s^* fills its corresponding slot in the prompt a_m^*, along with a_s^*, before being sent to the Cue Selector. In total, the ERM can detect seven types of errors (see Table 1) defined by psychologists as follows:

1. **Addition Error**: An addition error occurs when the user makes an action that belongs to another task. For example, it is decided that the user should be retrained to make "Black tea", but he/she adds sugar during the task.
2. **Perseveration Error**: The term "perseveration" is used as a label for the repetitive production of the same response to different commands [48]. Thus, in CogWatch, a perseveration error occurs when the user repeats any action he/she has already performed during the task. One can note that there are three rows for "Stir multiple times" in Table 1. This is due to the fact that each of the errors described in Table 1, when detected by the AIPS and shared with the Cue Selector, is subjected to different thresholds before it is decided to send a cue to the user [49]. In

Table 1. Types of user's errors the ERM can detect. BT - Black tea, BTS - Black tea with sugar, WT - White tea, WTS - White tea with sugar

ID	Error type	Task	Description of user's error
E01	Button trigger	All	Press "help button"
E02	Perplexity error	All	Make long pause during task
E03	Fatal error	All	Toying
E04	Button trigger	All	Press "task completion button" when not required
E05	Omission error	All	Omit to press "task completion button" when required
E06	Perseveration error	All	Add water to kettle multiple times
E07	Omission error	All	Fail to add water to kettle
E08	Omission error	All	Fail to boil water
E09	Perseveration error	All	Boil water multiple times
E10	Omission error	All	Fail to add teabag to cup
E11	Perseveration error	All	Add teabag multiple times
E12	Anticipation error	All	Pour water to cup before boiling water
E13	Perseveration error	All	Pour water to cup multiple times after boiling water
E14	Omission error	All	Fail to add boiled water to cup
E15	Omission error	All	Fail to remove teabag from cup when required
E16	Anticipation error	All	Stir while no water is in the cup
E17	Omission error	BTS, WTS	Fail to stir
E18	Perseveration error	BT	Stir multiple times
E19	Perseveration error	WT, BTS	Stir multiple times
E20	Perseveration error	WTS	Stir multiple times
E21	Addition error	BT, WT	Add sugar when not required
E22	Quantity error	BTS, WTS	Add too much sugar
E23	Omission error	BTS, WTS	Fail to add sugar
E24	Addition error	BT, BTS	Add milk when not required
E25	Quantity error	WT, WTS	Add too much milk
E26	Omission error	WT, WTS	Fail to add milk
E27	Anticipation error	All	Switch on kettle before adding water inside
E28	Anticipation error	All	Remove teabag from cup before adding boiled water in the latter
C01	NA	All	Task successfully completed

the case of the tea-making task, the threshold for "stir multiple times" is not the same for all the tasks. Thus these errors are referenced separately.

3. **Anticipation Error**: An anticipation error occurs when the user performs an action too early in time compared to his/her current history of actions. For example, stirring while the cup is empty or pouring water into the cup before boiling it.

Table 2. $SimU$ success rates with the virtual CogWatch

Types of tea	Assisted by AIPS	Ignoring AIPS
Black Tea	100.0%	76.0%
Black Tea with Sugar	100.0%	54.1%
White Tea	97.1%	48.0%
White Tea with Sugar	96.0%	24.2%

4. **Perplexity Error**: A perplexity error occurs when the user does not perform any action during a specific amount of time T. The ERM resets a counter after each observation received, and is set up to consider that the user needs assistance if no relevant action is received after time T.
5. **Omission Error**: An omission error occurs when the user considers that he/she has finished the task, but the ERM detects that the latter is incomplete. For example, the user considers that the task is over, while he/she has never put teabag in the cup.
   Note that in CogWatch, the trials stroke survivors go through while interacting with the system can be seen as cognitive exercises. The system's goal is to help them re-learn how to perform a task. Thus, even though CogWatch can detect task completion, users are asked to indicate if they believe they have finished the task, so the system can assess if they are aware of their own task completion. If users forgets to do so, then the omission error "E05" is triggered.
6. **Quantity Error**: A quantity error occurs when the user adds ingredients such as milk or sugar in an excessive manner. For example, this error will be detected if the user adds milk to the point where the cup is about to overflow.
7. **Fatal Error**: A fatal error occurs when the user action may lead to safety issues. Such errors force the system to abort the task automatically, in order to prevent the user from being harmed during the task.

The ERM can also be triggered via the patient's interface (Figure 2). This is made possible when the user presses the "Task completion button" or "Help button" (see *Button trigger* in Table 1)

7. User Study and Results

7.1. Evaluation via User Simulation

Before letting real participants interact with CogWatch, the latter was evaluated via User Simulation through two experiments. In the first experiment, the *SimU* described in Figure 5 tried to make each type of tea 3000 times. Table 2 shows its success rate when complying to psychologically plausible strategies suggested by the AIPS, and when ignoring the latter. For Black tea and Black tea with sugar, the *SimU* success rate is 100% when it always complies with the AIPS strategies. This means that the AIPS strategies are 100% accurate for those two tasks. When the *SimU* ignores the AIPS and performs the tasks following its own plan, its success rate significantly decreases: 76% for Black tea and 54.1% for Black tea with sugar. In the case of White tea and White tea with sugar, the assistance of the AIPS strategies also permits to increase the *SimU* success rate, but the latter is no longer 100%. This is due to the fact that CogWatch is an after-effect system, where an action has to be made by the user for the AIPS to plan what to should be done next. In those two cases, the AIPS strategies are always correct, but if the first action made by the *SimU* is an error that cannot be corrected, then the system cannot help the user.

In the second experiment, the *SimU* success rate was evaluated at varying levels of compliance, when the AIPS outputs Psychologically Plausible (P.P) strategies, and Non-

Psychologically Plausible (N.P-P) ones. As explained in section 5, this is due to the cost function used in the MDP. In Figure 7, one can see that when the *SimU* is 100% compliant to the AIPS strategies, whether the latter outputs N.P-P or P.P strategies has no impact on the *SimU*'s performance. This is an indicator that both N.P-P and P.P strategies are valid. However, when the $SimU$ decreases its compliance to the AIPS outputs, one can see in Figures 7 (a-b-d-e), that its success rate is higher when the strategies are psychologically plausible. In Figures 7 (a-b), when the *SimU* follows N-P.P strategies with a compliance at 20%, its success rate is the same as if it was trying to perform the task by itself (i.e., 0% compliance). It is possible to conclude that if both strategies are valid, the P.P ones are optimal compared to N-P.P. To make a parallel with a realistic situation, the P.P strategies can be seen as familiar ones; strategies able to take into account the ways a clinician would perform the tasks or the optimal ways the patients are used to perform when they succeed. Hence, with P.P strategies, when the user completes the task and accepts to comply, the AIPS succeeds to redirect the user to the most efficient ways of succeeding the task. On the contrary, even if N-P.P strategies are always correct, because they do not take into account patients' habits, they lead to more users' failures. Indeed, in [50], [51] De Klein and Graybiel highlighted the impact of familiar and unfamiliar sequences on success rate. Familiar sequences are easier to execute and require less effort and energy, as they are done through a sub-cortical structure where the sequence is reduced to a single unit. In contrast for novel sequences or sequences that diverge from the familiar one, one needs to use cortical mechanisms (more effort, higher demands on resources) to re-create them. Taking into account the results obtained via simulation, it was decided to set up

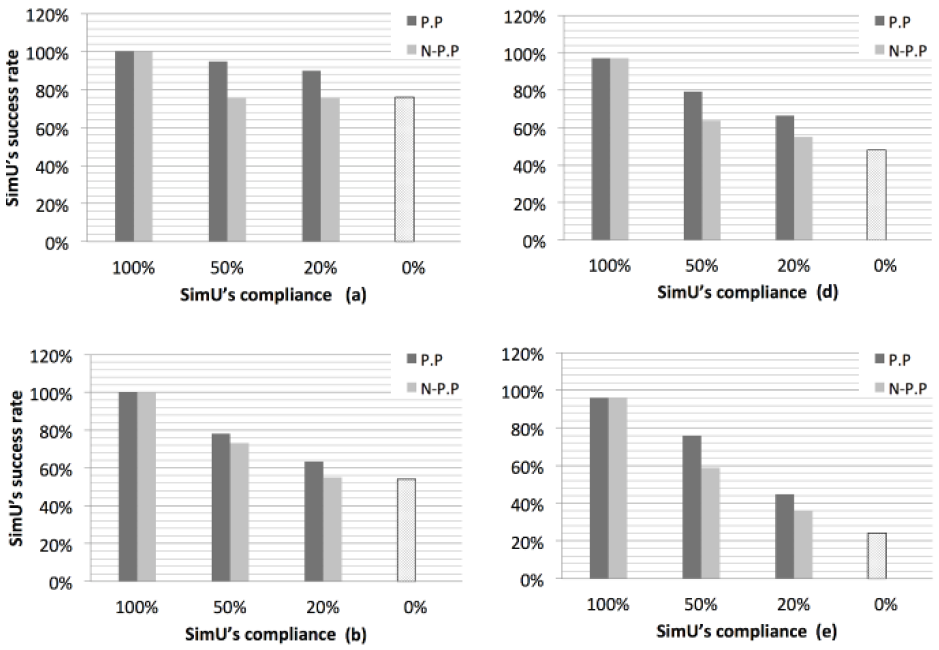

Figure 7. $SimU$ success rates at varying compliance to the AIPS strategy. P.P and N-P.P stand for Psychologically Plausible and Non-Psychologically Plausible strategy; a, b, d and e correspond to Black tea, Black tea with sugar, White tea, White tea with sugar.

Table 3. Number of Non Fatal Cues (N.F.C), Fatal Cues (F.C) and successes

	N.F.C	F.C	Successes
With CogWatch	38	4	92
Without CogWatch	112	31	65

the AIPS for it to generate psychologically plausible strategies only. The evaluation of the system was then re-run with real participants, as described in the following section.

7.2. Evaluation with Stroke Survivors

To measure the efficacy of CogWatch with real participants, two sets of 96 trials were performed by 12 patients (i.e., 6 women, 6 men, aged between 53 and 82). They exactly went through 24 trials of each type of tea: "Black tea with sugar", "White tea with sugar", "White tea", "Black tea". During all trials, the system could observe patients' behavior and record the number of time a cue should be sent to them. However, during one set of trials, patients did not receive any cues, while in the other one the same patients could receive guidance from the system. Note that when CogWatch's cues were made available for patients, the later were free to comply or not to the system's outputs. First, we compared the average number of cues triggered by patients when not using the system, and when using it. Cues are messages generated by the Cue Selector (see Figure 1) when the latter receives information from the AIPS. When a cue is triggered, it means that the patient made an error and the whole system considers that this patient's behavior needs to be corrected. In Figure 8, one can see that patients triggered an average of 11.9 cues when not using CogWatch, compared to 3.5 cues when using it. Recall that patients who were not using the system could not receive cues. However, even in this case, the system was observing their behavior and calculating the number of times it would have sent them cues due to the errors they made. This means that when not using the system, the patients made more errors than when they had access to the system's guidance. Exactly 10 out of the 12 patients tested reduced the number of errors they made when assisted by CogWatch. The two patients who did not improve, showed an increase in the number of stirring repetition errors. It is likely that the system was more sensitive when detecting such behavior as errors than a human assessor. This is a potential limitation of the study. Indeed, the errors made during trials performed without CogWatch were detected via video analysis made by human assessors. When Cogwatch was used, errors were coded based on strict rules defined in advance (see Table 1), and which are not subject to human's variability. In Table 3, we then compared the number of non-fatal cues (N.F.C) and fatal cues (F.C) received by the patients when assisted by CogWatch (CW), and when performing the tasks without its guidance.

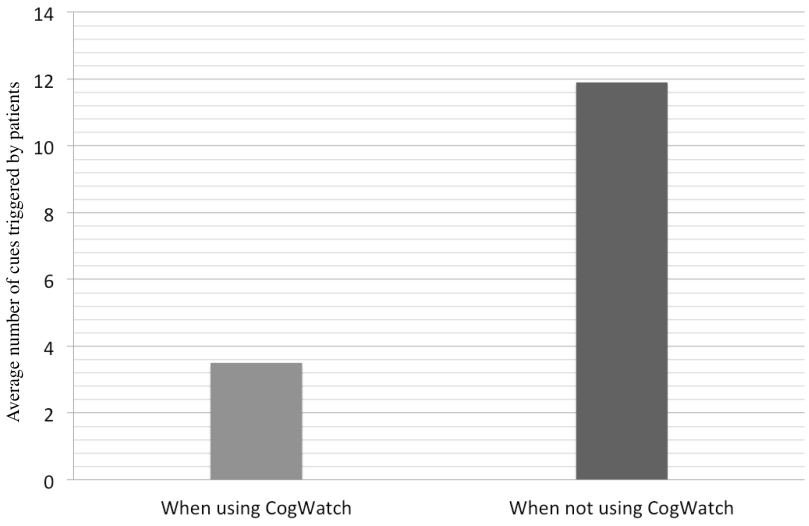

Figure 8. Average number of cues triggered by patients

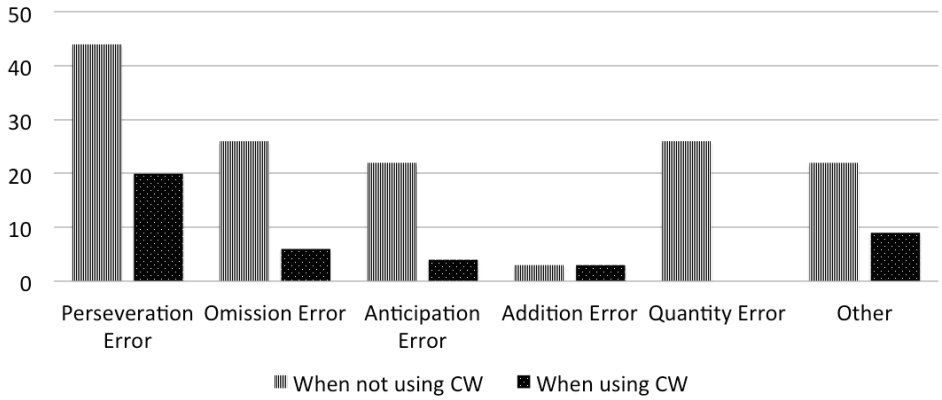

Figure 9. Types of errors triggering the cues received by patients

Concretely, Non Fatal Cues are related to recoverable errors (see the definition of Fatal Errors in section 6). Non-recoverable errors can occur when the patient's safety is at risk. In the Table, one can see that when interacting with CogWatch, the patients receive less Fatal and Non Fatal Cues, and succeed the tasks more often than when they do not have access to the system's prompts. Indeed, the patients' success rate with CogWatch is 95.8%, and only 67.7% without. Figure 9 shows the type of errors made by patients that triggered the cues they received during the trials. One can see that the use of CogWatch

significantly reduces the number of errors, such as omission, anticipation, quantity and perseveration errors (Table 1). It means that when patients were guided by the system, they forgot mandatory actions less often (e.g., adding the teabag in the cup); they made less anticipation errors (e.g., trying to switch the kettle on when the latter is empty); they did not misjudge the quantity of any ingredient (i.e., quantity error); and they repeated actions less often during the task (i.e., perseveration error). As far as addition errors are concerned, one can see that with or without the help of the system, the patients had the same low tendency to add ingredients that did not correspond to the task they had to complete.

All the results obtained show that CogWatch is able to properly provide guidance during tea-making. When having access to the prompts delivered by the system, patients have a lower probability to make mistakes and to fail the task. Meaningful comparisons with other MDP-based assistive systems are hardly possible. To be able to do so in the future, it will be necessary to reach consensus on a set of standard evaluation methods.

8. Discussion

When designing an assistive system such as CogWatch, another important feature to take into account is its ability to be easily adaptable to other tasks. Indeed, different ADLs may be essential for independent leaving from grooming to preparing food and drinks. Users who have difficulty in completing one ADL (i.e., tea-making) are also likely to have problems in other ADLs such as making coffee, making a snack, teeth-brushing, cooking, etc. Thus, designing a single system that can support multiple tasks is highly desirable. In this section we will discuss the flexibility of the current CogWatch system, and how it can be adapted to other tasks. This section is based on our experience in designing the latest prototype of Cogwatch, which included both tea making and teeth-brushing. These two tasks are relatively different. Nevertheless, while it took over 2 years to develop CogWatch for tea making, it took less than 6 months to adapt the system so it can also provide assistance during teeth-brushing.

As explained in Section 2, CogWatch is composed of 4 main modules: 1) The sensors (intelligent objects) that are used to capture user's behavior; 2) Action recognition infer user actions (i.e., ARS); 3) Artificial Intelligent Planning System which plans the recommendations to be sent to users and detects potential errors (i.e., AIPS), and 4) a module in charge of selecting the appropriate form in which a recommendation should be shared with a user (i.e., Cue Selector). The amount of time and expertise necessary to adapt the whole system to a new task depends on the approach taken to design each module. Indeed, each module composing CogWatch has a different level of flexibility and would require different types of modifications or updates in order to take into account new tasks.

From a general point of view, the important points to consider for each module can be described as follows.

- **Sensors**: If the user's behavior needs to be tracked during a task that involves similar objects to those for tea-making, then the CogWatch coasters can be reused. In the case where the new task involves different objects, the packaging of the coasters may have to be re-designed in order to fit the new objects. If the objects involved cannot be tracked with coasters (e.g., clothes during a dressing task) CogWatch would then rely on Kinect^{TM} and its hand-tracking capabilities. As explained in [35], Kinect^{TM} was used to capture hand-coordinates using software based on "Kinect-Arms" [52]. It allowed the system to detect when the user hands were in the vicinity of an object. One could then make the assumption that when a hand remains stationary at the vicinity of an object, the user may be using this object. Another possibility would be to use a new set of sensors. In this case, the hardware of the new sensors would need to be adjusted depending on the objects used in each task.
  One may consider to change or add new sensors only if there is a need to increase the level of information sent to the ARS, so the latter can detect more complex and dexterous actions. For example, in the case of teeth-brushing, Kinect^{TM}, CogWatch coasters and Shimmers [53] were used to track the movements of the toothbrush. Microphones were also added to the set of sensors to detect brushing sounds.

- **ARS**: When the objects that are involved in a new task are identified and instrumented, the ARS can be updated by going through the same process that was implemented during tea-making. First, a set of relevant user actions to recognize during the task needs to be defined (e.g., ("add teabag", "boil kettle", ...) for tea-making or ("open tap", "wet hands", "take soap", ...) for hand-washing). Once this set of user actions is defined, examples of sensor output for each of these actions will be recorded. This will be done for each user action individually and for examples of complete tasks (i.e., uninterrupted sequence of user actions). When this database of sensor outputs related to user actions is created, one should empirically optimize the model parameters, the number of states and the number of GMM components per state. See [35] for more details. To adapt the ARS to a new task takes time and effort, and its learning process is repetitive. However, the development part for a new task is relatively minimal here as the algorithm is already in place. Beyond the time and effort, a limitation of the algorithm used in the ARS is related to the fact that its parameters are specific to the objects used; or in the case of Kinect^{TM} to the location of the objects in space. Hence, any change to the physical environment and objects that are used requires a new training and optimization of these parameters.
  A more satisfactory approach would be to focus on a particular application domain, for example kitchen-based tasks or bathroom-based tasks. Depending on the application domain, one should then have a fixed set of objects to be instrumented. Rather than modeling whole user actions, one could identify a set of sub-actions from which each user action could be constructed (e.g., in the case of the user action "pour water from kettle to cup", a possible set of sub-actions could be composed of ("grasp kettle handle", "lift kettle", "tilt kettle", ...)). One could then build HMMs of these sub-actions and then combine them to obtain models of any user action.
  Many research questions would still remain, for example: Is a model of the sensor outputs for someone tilting a milk jug appropriate for someone tipping a kettle? Or do different sizes of objects require different models? These are topics for further research.
- **AIPS**: When the set of user actions to be recognized by the ARS is known, a list of errors users tend to make needs to be defined, as explained in Section 6. This process is undemanding and provides the AIPS the main elements it needs to be trained to provide assistance during another task. Indeed, the sets of user actions and errors will be used to update the parameters of the MDP and POMDP (i.e., S, A, T) as described in Section 5. The software on which the AIPS is based has been designed to allow such flexibility. The developer simply has to provide the sets of actions and errors for the software to automatically generate the MDP and POMDP parameters.
  In CogWatch, the AIPS is trained with a simulated user in a virtual environment. The same architecture can be reused for any other sequential task. The behavior of the simulated user and the confusion matrix related to the ARS performance can be randomly chosen or specified based on a database. As far as the simulated user is concerned, this database should contain examples of actions taken by real participants during the task, so it can calculate bigram probabilities (see Section 5). In the case of the confusion matrix related to the ARS performance, it can be

defined by assessing the outputs of the ARS when the user makes specific actions or by going through the whole task. The training itself is done automatically, and requires no more expertise or human intervention. Hence, the MDP or POMDP based AIPS is easily adaptable to other sequential tasks.

- **Cue Selector**: The Cue Selector has been designed to be user-friendly. One can easily modify its parameters via a graphical interface (i.e., clinician's screen). In other words, the number of times a cue should be shared with a user during a task and the type of this cue (see Section 6) can be updated at any time depending on the user's needs. In CogWatch, the Cue Selector chooses the cues to send to participants from a database of images, videos and sounds. These elements are hints and contain visual or audio information about what the user should do during the task. If another task needs to be taken into account this database will be updated. For example, new images depicting the actions the user should do must be taken so they correspond to the new task's goal.
  From a general point of view, any modifications applied to the Cue Selector are effortless compared to the ARS or AIPS. Indeed, the Cue Selector does not go through any learning process, and can quickly be adapted to fit the need of another task.

9. Conclusion

In this paper, an assistive system designed to guide stroke survivors during everyday task is described. The huge variance in the execution of the task is tackled by defining a specific Markov Decision Process (MDP) framework. The latter has the potential to model any sequential activity of daily living. The algorithm that was used to solve the MDP, and which allows the system to learn how to retrieve optimal prompts to patients during the task is explained. Finally, the ability of the system to assist patients with psychologically plausible strategies during tea-making was proved. Indeed, we showed that, when patients have access to the outputs provided by the system, the patients' success rate is higher and make fewer errors than when they cannot interact with the system.

The performance of the CogWatch system was presented during the tea-making task only. However, the system's AIPS has been developed to be easily adaptable to other sequential activities of daily living. Indeed, in the case where the AIPS should provide guidance during another task, only two important human inputs will be necessary: the definitions of a new action space and errors stroke survivors tend to make during this task. From these parameters, the AIPS will automatically infer what is the restricted state space. The later will then be used to retrain the AIPS in order to obtain a new policy.

As far as the ARS is concerned, further work will be needed for it to be quickly adaptable to any other task. Currently, if users' behavior need to tracked and monitored during another task that involves other objects, new sensors may have to be designed. These sensors' outputs will then be sent to the ARS which will learn how to infer users' actions from the data it receives. To reduce the number of sensorized objects to develop, one solution is to use Kinect^{TM} and its hand tracking capabilities [35]. The latter will help the ARS during its action detection process without the need to rely on sensors attached to objects only.

Another point to highlight is the fact that the system presented in this paper is semi-autonomous. In the case of a system that would have to act in a total autonomous way, it

will need to automatically cope with the uncertainties related to its environment. In such a case, one solution is to implement a Partially Observable MDP [54], which will model these uncertainties, and enable the assistive system to act even if the user's state is only partially known from its point of view. Such a system has already been implemented in CogWatch [55, 49, 56], and was tested with success via simulation.

The next step is to let stroke survivors interact with the POMDP-based assistive system and evaluate its performance under uncertainty. Note that this paper focused on a system designed to help stroke survivors. However, the models it implements and its flexibility may allow it to provide assistance to other users with cognitive impairments (e.g., people with dementia, with learning disabilities, with memory deficits). Indeed, CogWatch provides instructional cueing, can detect when users make errors during a task automatically, and intervenes when necessary. These capabilities can be helpful to any individual who may need to re-learn how to perform daily activities.

References

[1] Intercollegiate Stroke Working Party. Royal college of physicians national sentinel stroke clinical audit 2010 round 7 public report for England, Wales and Northern Ireland. Technical report, 2010.

[2] A. Di Carlo. Human and economic burden of stroke. *Age and ageing*, pages 4–5, 2009.

[3] W. Bickerton, M. J. Riddoch, D. Samson, A. Balani, B. Mistry, and G. W. Humphreys. Systematic assessment of apraxia and functional predictions from the Birmingham Cognitive Screen. *Journal of Neurology, Neurosurgery, and Psychiatry*, pages 513–521, 2012.

[4] B. Petreska, M. Adriani, O. Blanke, and A. G. Billard. Apraxia: a review. In *From Action to Cognition*, volume 164 of *Progress in Brain Research*, pages 61 – 83. 2007.

[5] G. Goldenberg, M. Daumuller, and S. Hagmann. Assessment and therapy of complex activities of daily living in apraxia. *Neuropsychological Rehabilitation*, 2001.

[6] K. Morady and G. W. Humphreys. Comparing action disorganization syndrome and dual-task load on normal performance in everyday action tasks. *Neurocase*, pages 1–12, 2009.

[7] G. Goldenberg. Apraxia - The cognitive side of motor control. *Cortex*, 57:270 – 274, 2014.

[8] D. Hyndman and A. Ashburn. People with stroke living in the community: Attention deficits, balance, ADL ability and falls. *Disability and Rehabilitation*, pages 817–822, 2003.

[9] M. C. Silveri and N. Ciccarelli. Semantic memory in object use. *Neuropsychologia*, pages 2634 – 2641, 2009.

[10] M. Zhang, T. C. Davies, and S. Xie. Effectiveness of robot-assisted therapy on ankle rehabilitation - a systematic review. *Journal of NeuroEngineering and Rehabilitation*, 2013.

[11] H. Zheng, N. D. Black, and N. Harris. Position-sensing technologies for movement analysis in stroke rehabilitation. *Med. Biol. Eng. Comput*, 2005.

[12] L. Zollo, E. Papaleo, L. Spedaliere, E. Guglielmelli, F. J. Badesa, R. Morales, and N. Garcia-Aracil. Multimodal interfaces to improve Therapeutic Outcomes in Robot-Assisted Rehabilitation. In *Gearing up and accelerating cross-fertilization between academic and industrial robotics research in Europe*, volume 94, pages 321–343. 2014.

[13] A. Mansfield, J. Wong, M. Bayley, L. Biasin, D. Brooks, K. Brunton, J. Howe, E. Inness, S. Jones, J. Lymburner, R. Mileris, and W. McIlroy. Using wireless technology in clinical practice: does feedback of daily walking activity improve walking outcomes of individuals receiving rehabilitation post-stroke? Study protocol for a randomized controlled trial. *BMC Neurology*, 2013.

[14] A. M. Nicol, C. Gee Bush, and E. Balka. Internet devices and desires: A review of randomized controlled trials of interactive, Internet-mediated, in-home, chronic disease monitoring programs. *J. Res. Interprofessional Pract. Edu.*, 2009.

[15] J. Westin, M. Dougherty, D. Nyholm, and T. Groth. A home environment test battery for status assessment in patients with advanced Parkinson's disease. *Comput. Methods Prog. Biomed.*, 2010.

[16] L. Cunningham, S. Mason, C. Nugent, G. Moore, D. Finlay, and D. Craig. Home-based monitoring and assessment of Parkinson's disease. *IEEE Transactions on Information Technology in Biomedicine*, 2011.

[17] M. E. Pollack. Autominder : A case study of assistive technology for elders with cognitive impairment. *Generations*, 2006.

[18] B. O'Neill, K. Moran, and A. Gillespie. Scaffolding rehabilitation behaviour using a voice-mediated assistive technology for cognition. *Neuropsychological Rehabilitation*, 2010.

[19] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, and A. Mihailidis. A planning system based on markov decision processes to guide people with dementia through activities of daily living. In *IEEE Transactions on Information Technology in Biomedicine*, 2006.

[20] J. L. Ambite and C. A. Knoblock. Planning by rewriting. *Journal of Artificial Intelligence Research*, 2001.

[21] S. Czarnuch, S. Cohen, V. Parameswaran, and A. Mihailidis. A real-world deployment of the coach prompting system. *J. Ambient Intell. Smart Environ.*, 2013.

[22] J. Hoey, T. Plötz, D. Jackson, A. Monk, C. Pham, and P. Olivier. Rapid specification and automated generation of prompting systems to assist people with dementia. *Pervasive and Mobile Computing*, 2011.

[23] K. B. Ross and R. T. Wertz. Type and severity of aphasia during the first seven months poststroke. *Journal of Medical Speech-Language Pathology*, 2001.

[24] S. T. Engelter, M. Gostynski, S. Papa, M. Frei, C. Born, V. Ajdacic-Gross, F. Gutzwiller, and P. A. Lyrer. Epidemiology of Aphasia Attributable to First Ischemic Stroke. *Stroke*, 2006.

[25] H. Chen, X. Pan, J. King Lam Lau, W-L Bickerton, B. Pradeep, M. Taheri, G. Humphreys, and P. Rotshtein. Lesion-symptom mapping of a complex figure copy task: A large-scale PCA study of the BCoS trial. 2016.

[26] S. Paolucci, G. Antonucci, L. E. Gialloreti, M. Traballesi, S. Lubich, L. Pratesi, and L. Palombi. Predicting stroke inpatient rehabilitation outcome: The prominent role of neuropsychological disorders. *European Neurology*, 1996.

[27] M. Puterman. Markov decision processes: Discrete stochastic dynamic programming. *Wiley, New York*, 1994.

[28] R. Bellman. Dynamic programming. *Princeton University Press, Princeton, NJ*, 1957.

[29] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 1995.

[30] J. Tash and S. Russell. Control strategies for a stochastic planner. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[31] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 2000.

[32] G. Shani, D. Heckerman, and R. I. Brafman. An MDP-based recommender system. *Machine Learning Research*, 2005.

[33] T. Bohnenberger and A. Jameson. When policies are better than plans: decision-theoretic planning of recommendation sequences. In *Proceedings of the 6th international conference on Intelligent user interfaces*, 2001.

[34] C. M. Hughes, C. Baber, M. Bienkiewicz, and Helmsdörfer. Application of human error identification (HEI) techniques to cognitive rehabilitation in stroke patients with limb apraxia. In *Access in Human-Computer Interaction. Applications and Services for Quality of Life*, pages 463–471. Springer Berlin Heidelberg, 2013.

[35] R. Nabiei, M. Parekh, E. Jean-Baptiste, P. Jančovič, and M. J. Russell. Object-centred recognition of human activity for assistance and rehabilitation of stroke patients. In *IEEE Int. Conf. On Healthcare Informatics (ICHI 2015), Dallas, TX, USA*, 2015.

[36] M. Gales and S. Young. The application of Hidden Markov Models in speech recognition. In *Foundations and Trends in Signal Processing*, 2007.

[37] J. C. Spohrer, P. F. Brown, P. H. Hochschild, and J. K. Baker. Partial traceback in continuous speech recognition. In *Proc. IEEE Int. Cong. Cybernetics and Society*, 1980.

[38] E. M. D. Jean-Baptiste, R. Nabiei, M. Parekh, E. Fringi, B. Drozdowska, C. Baber, P. Jancovic, P. Rotshein, and M. Russell. Intelligent assistive system using real-time action recognition for stroke survivors. In *Proceedings of the IEEE International Conference on Healthcare Informatics (ICHI)*, 2014.

[39] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction. Adaptive Computation and Machine Learning.* MIT Press, Cambridge, MA, 1998.

[40] E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. In *Proceedings of the IEEE transactions on Speech and Audio Processing*, 2000.

[41] B. Thomson and S. Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, pages 562–588, 2010.

[42] K. Scheffler and S. Young. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *Proceedings of NAACL*, 2001.

[43] K. Georgila, J. Henderson, and O. Lemon. Learning user simulations for information state update dialog systems. In *Proceedings of Eurospeech*, 2005.

[44] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *KER*, 2006.

[45] V. Rieser and O. Lemon. Cluster-based user simulations for learning dialogue strategies. In *Proceedings of Interspeech*, 2006.

[46] S. Quarteroni, M. Gonzàlez, G. Riccardi, and S. Varges. Combining user intention and error modeling for statistical dialog simulators. In *Proceedings of Interspeech*, 2010.

[47] J. Pflugler, A. Schlegel, E. Jean-Baptiste, P. Rotshtein, M. Pastorino, J. Rojo, J. M. Cogollor, M. T. Arredondo, M. M. N. Bienkiewicz, and J. Hermsdorfer. Using human-computer interface for rehabilitation of activities of daily living (ADL) in stroke patients. Lessons from the first prototype. In *Proceedings of the International Conference on NeuroRehabilitation (ICNR)*, pages 629–636, 2014.

[48] N. Helm-Estabrooks. The Problem of Perseveration. *Semin Speech Lang*, 2004.

[49] E. M. D. Jean-Baptiste. Statistical task modeling of activities of daily living for rehabilitation. In *PhD thesis*, 2016.

[50] E. De Kleine and R. H. Van der Lubbe. Decreased load on general motor preparation and visual-working memory while preparing familiar as compared to unfamiliar movement sequences. *Brain Cogn.*, 2011.

[51] A. M. Graybiel. The basal ganglia and chunking of action repertoires. *Neurobiol Learn Mem.*, 1998.

[52] A. M. Genest, C. Gutwin, A. Tang, M. Kalyn, and Z. Ivkovic. KinectArms: A toolkit for capturing and displaying arm embodiments in distributed tabletop groupware. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 157–166, 2013.

[53] Shimmer Sensing. URL `http://www.shimmersensing.com/`.

[54] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, pages 99–134, 1998.

[55] E. M. D. Jean-Baptiste, P. Rotshein, and M. Russell. POMDP based action planning and human error detection. In *Proceedings of the 11th International Conference on Artificial Intelligence Applications and Innovations (AIAI2015)*, 2015.

[56] E. M. D. Jean-Baptiste, P. Rotshtein, and M. Russell. Cogwatch: Automatic prompting system for stroke survivors during activities of daily living. *Journal of Innovation in Digital Ecosystems*, 2016.