Edge-Based Wearable Systems for Cognitive Assistance: Design Challenges, Solution Framework, and Application to Emergency Healthcare

Siyan Zhao, Junjue Wang, Hongkun Leng, Yuqi Liu, Haodong Liu,
Daniel P. Siewiorek, Mahadev Satyanarayanan, Roberta L. Klatzky

March 2020
CMU-CS-20-102

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Cognitive assistants are computer-based interactive guides for complex activities such as emergency medical care. Recent advances in computing capability and edge computing make it possible to host such an assistant in a wearable device. We identify technical challenges facing the designer of a wearable cognitive assistant and describe pathways to solutions. We illustrate the solution space in the form of an automated end-to-end assistant to guide a novice through the application of an automatic external defibrillator (AED). Our results indicate that without such assistance, novices will fail, but technological barriers limit real-time success. Our work highlights challenges as well as new capabilities made possible by wearable technology in this domain.

This work was supported by the National Science Foundation (NSF) under grant number CNS-1518865.

Keywords: Empirical studies in ubiquitous and mobile computing, Ubiquitous and mobile computing systems and tools, Real-time system architecture, Edge computing

1 Introduction

There has been growing interest in instrumenting sensors to guide people in a variety of tasks. For example, a camera-based system can automatically detect, recognize and track exercises performed in a gym setting [18]. Motion and contact sensors have been integrated to pick up the location of a user and the current state of tools to help users with cognitive impairments in cooking activities [4]. An important domain of use for these sensor-based applications is cognitive assistants, that is, computer-based interactive guides for a complex human activity. The application domain is longstanding, but has been transformed by current technology, which provides sensors to closely monitor the user, supports interactions through voice or gesture that free the hands, takes advantage of machine learning techniques to analyze and interpret the environment, and capitalizes on low-latency transmission between the user and the computational algorithms.

While the technical advances afford the design of more sophisticated assistants, it is still a challenge to conceive and implement a well-designed cognitive assistant. For example, how should a task be properly divided into action components such that each action is achievable by the user and their completion states can be detected by sensors? How should assistants handle latency in computation to create a smooth user experience? The purpose of the present paper is to identify these challenges and to propose solutions in the design of cognitive assistants. To illustrate our solutions, we provide an end-to-end example in the medical domain, using an automatic external defibrillator (AED), a device found in many public places. Although the intended use case may include guidance by phone contact with emergency medical technicians, time or situational constraints may lead untrained users to attempt implementation. A study reported here showed that contrary to these intentions, novices could not successfully use an AED without the guidance of a cognitive assistant.

The contributions of this work are as follows: (1) We provide baseline evidence that a widely deployed commercial product cannot be successfully used by untrained users, without cognitive assistance. (2) We introduce the concept of proper chunking of instructions for wearable cognitive assistance applications, provide guidelines for instruction units, and demonstrate their impact. (3) We not only demonstrate what can be learned from designing an assistant for a specific device, but also generalize to a formal structure that can be used to guide the creation of future wearable cognitive assistants. (4) We confirm the critical importance of edge computing for this class of time-constrained applications on wearable devices and provide quantitative metrics for the cost of moving away from the edge of the cloud.

2 Background and Related Work

Cognitive assistants have been available for decades in various forms, ranging from checklists to full instruction under computer control. An early example is the project STEAMER [16], which reduced an extensive manual for running a steam propulsion system to procedures instructed by a GUI. More recent examples can be found in cognitive tutors developed over the last two decades or so, which embed a procedural model of a complex domain such as middle-school mathematics into an interactive interface that guides learners through problems according to their ongoing mastery [25].

Considerable research has been directed to the development of cognitive assistants in broad application contexts, including health care. Indeed, whole research fields were developed to address major underlying problems. For example, where an assistant is modeled after a human expert, research has been dedicated to mechanisms for knowledge acquisition from experts [6,36].

Where tasks have many steps that may not be transparent, task analysis techniques have been developed to create the underlying model [2,3]. In error-prone domains such as medical procedures, there is extensive analysis of vulnerabilities to human judgment [5,32]. The generation of instructions by the computer has also been a topic of research [1,14,17].

While prior research continues to be highly useful in designing novel cognitive assistants, rapid developments in supporting technology offer new possibilities for implementation that have been less scrutinized. As described in [29], these developments span three broad tiers of new computing capability. The first is the immense computational power and storage capability of commercial cloud computing, which has the additional advantage of affordability to the typical user. The second tier is what is termed "edge" computing [28,31], comprising computing infrastructure near the user that affords very fast latency and high bandwidth to an application in need of dense, high-speed data transmission. We use the term *cloudlet* to refer to an instance of such infrastructure. The third tier encompasses devices worn or carried by the user, including dedicated sensors, smartphones, and AR/VR devices.

Fig. 1 uses three dimensions to represent the transformations in cognitive assistants made possible by these tiers. *Computational capability* indicates how well assistive systems handle problems of high demand in terms of compute resources. *Latency* is simply the duration of the entire chain from a user query to a system response, often called round-trip time. *Device intrusiveness* reflects a progression from sensors embedded in or worn by the user to detached ancillary devices like keyboards and mice. The figure indicates how increased capability, along with decreased latency and intrusiveness, have expanded the interaction platform of cognitive assistants, providing new avenues for communication with the user and monitoring of the environment. The emerging platform includes dense sensor input; form factors such as the Google Glass® that enable hands-free user interaction; deep learning algorithms that recognize objects, sounds or faces [20,30]; natural language communication; and fast round-trip-time from one task step to another. The culmination of these technical developments is a new type of cognitive assistant: an intelligent help system hosted in a device carried on or nearby the user that provides real-time guidance through a multi-step task. Chen [9] demonstrated the power of this approach by building assistants for moment-to-moment instruction in a range of applications, from Lego block building to the ball return in a game of ping-pong. The approach has also been applied to assisting trauma surgeons with repair of a broken rib by inserting a plate [35].

As a prototype, consider a task of guiding a novice through a wrist-mounted blood pressure reading. The person wears a camera linked to a computer vision system that analyzes the visual field for body parts, machine components, on-screen metric readings, and so on. The computational power resides not in the wearable containing the camera, but rather in a cloudlet interfaced to it. An auditory or visual interface conveys the next action to the user. Step by step, the user is guided to mount the blood pressure device in the appropriate location on the arm (just below the wrist), hold the arm at heart level, start the machine, and take the reading. There are many potential error points; for example, mounting the device too close to the wrist or failing to notice that the batteries are dead.

The initial portion of the present paper is intended to identify choice points in the design of the type of assistant that is made possible with new technologies and to suggest constraints on the solutions. This analysis is sufficiently general so that it applies to cognitive assistants in many fields. In the second portion of the paper, we present one path through the solution space specifically for an application to health care, in the form of an end-to-end cognitive assistant for an automatic external defibrillator (AED). The resulting proof-of-concept is intended to illustrate the challenges as well as the new capabilities made possible by computing and interface technology.

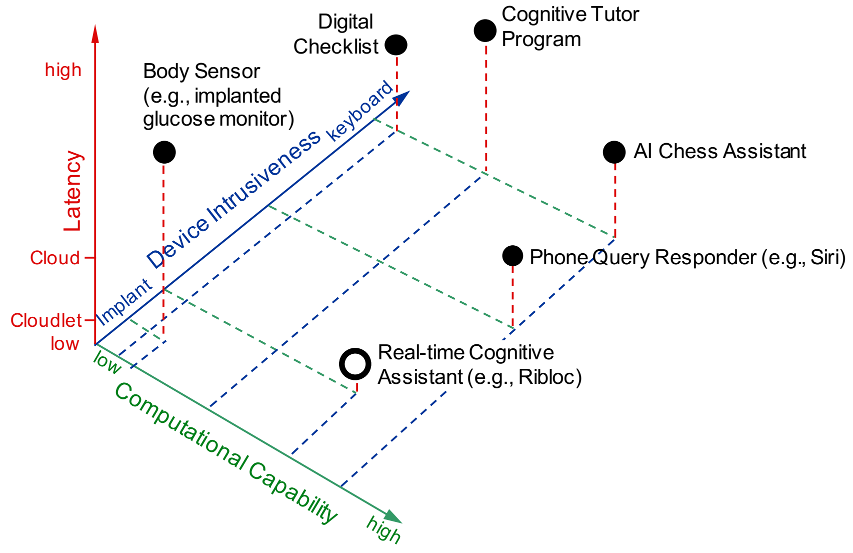


Fig. 1. Design space for Cognitive Assistants: Decrease in latency (z axis) and device intrusiveness (y axis) and increases in computational capability (x axis) have transformed assistant applications, as examples illustrate. The open circle is the type of cognitive assistant that the current paper will focus on.

3 Cognitive Assistant Framework

In general, essential components of a cognitive assistant are (i) a task model that incorporates the components of the task, (ii) a task monitor, the version stored in a computer that leads the user through a step-by-step procedure at a level of basic elements or primitives, and (iii) input/output capability – the former to support task monitoring, and the latter to provide instructions that communicate expected next steps or guide error correction. For the assistants considered here, there is a fourth essential: (iv) the technologies that support these three components.

These components present a set of challenges for the system designer. The challenge associated with the first component is to construct a model that is both necessary and sufficient with respect to normative behaviors. Often, this takes the form of a hierarchy, with successive levels breaking down the task to ultimate primitives. The challenge for the second component is to employ a formalism that can be instantiated in a computer program that monitors the user's progress through the task. This formalism must expand the initial model to anticipate *non*-normative behaviors, including repetitions and errors. The challenge for the third component is to select input and output channels that enable the assistant to sense progress and communicate with the user. The challenges for the technological devices used in implementation include adequate bandwidth, latency, and sensitivity. Table 1, column 2, describes more specifically the issues presented to the designer with respect to each component of the assistant. Our design solutions will target these challenges but under a set of constraints on task features, which will be introduced below and summarized in Table 2.

Table 1. Components of a cognitive assistant and major challenges they present, along with our proposed solutions afforded by a set of task constraints (see Table 2).

Components	Issues for Design	Our solutions afforded by task constraints (numbers correspond to Table 2)
Task Model	Hierarchical Structure	Defined by action sequence (1 and 2)
	Terminal Elements (primitives)	Actions that can be sensed and labeled/depicted (2,3,4)
Task Monitor	Link to Task Model	Task monitor states are terminal actions in mode; input is sensed change in environment (2,3)
	Augmentation for Predictable Error	Errors are predicted observable actions (3) or temporal anomalies (5)
	Selecting output states	Outputs are instructions to user to act (4)
I/O	Input form	Wearable or nearby sensor for action outcome (3)
	Output form	Verbal or image sufficient to instruct (2)
Technological Implementation	Bandwidth, latency	Edge computing sufficiency (3,5)
	Sufficient Detection for Task Boundary	User can assist or bypass sensors (3)

While the challenges in Table 1 appear formidable, they are in fact mitigated by an important feature of technologically advanced assistants, namely, they allow the human to be embedded into the system in a deep way. In a sense, human perceptual and cognitive processing become part of the system capability, and so part of the assistant itself. To capitalize on the embedding of the human in the system, we focus on tasks that have the following constraints.

Table 2. Constraints on candidate tasks for cognitive assistance.

Constraint	Description
1) Sufficient goal structure	There exists at least one sequence of goals to solve the problem requiring assistance.
2) Decomposable into actions	Goals can be decomposed into discrete steps, corresponding to user-executed actions.
3) Observable step transitions	Boundaries between steps correspond to observable changes in the environment that can be sensed.
4) Communicability	Sensed changes enable instructions or corrective feedback that comply with user's processing capacity.
5) Normative time-course	Step timing is predictable from sensori-motor or cognitive norms.

These constraints point to critical features of a task that can be targeted for a cognitive assistant. Availability of a solution means that the task is not open-ended; there are steps to complete it. The feature that step transitions are observable implies that no modeling of changes in invisible internal states of the user is necessary. This bounds the complexity of the task model in comparison, for example, to a cognitive tutor. In the ideal task, all of the observable changes at step boundaries can be recognized directly by external sensors. In actuality, sensors are not always adequate; however, this weakness is mitigated, because the user can step in to expose data to a sensor or report changes on the basis of his or her own perceptual experience. The requirement of communicable instructions to the user can be met by language or images, as long as the user's processing capacity is not overloaded. The predictability of step timing means that excessive waiting times for user action or implausibly impulsive actions can be detected.

To convey how these constraints operate to create a human-embedded assistant, consider again how one might be devised to aid wrist-mounted blood pressure reading. Following constraint 1 above, an underlying model is created that divides the task into a series of goals sufficient for the task to be completed. For example, goals might be to mount the display and take the reading. Following constraint 2, each of these goals is decomposed further -- for example, taking the reading begins with lifting the arm to the heart level -- until the ultimate decomposition ends at readily understood steps for the user, such as placing the elbow on a surface and bending the arm until the wrist is at the elevation of the heart. Following constraint 3, the assistant uses received sensor data about environmental changes to identify task boundaries; for example, a camera observes the device monitor and registers when the display is turned on. Some actions, like holding the hand at heart level, cannot be directly monitored for success, so instead the system relies on the user to report completion. Constraint 4 is represented at various points in the sequence of actions, where the assistant instructs the user as to what to do next in the form of words ("press the start button") or pictures displayed onscreen. Throughout, constraint 5 leads to detection of time delays not commensurate with the stage of the task; for example, there is more tolerance for delay during fastening the strap than pressing the button.

4 Design Solutions in Detail

Under the present framework, solutions to the challenges raised above should capitalize on the task constraints we have outlined and should exploit the human in the loop. Accordingly, the third column of Table 1 indicates approaches to solving various challenges and indicates, by number, which of the task constraints makes a particular solution possible. The following section essentially expands on the table. It considers each component of the cognitive assistant, re-visits the challenges in the context of the task constraints, and offers more specific details about approaches to meeting them.

4.1 Task Model

The challenges of the first component is that it must capture the requisite behaviors in a coherent structure, terminating in action specifications. The applicable constraints specify that our tasks can successfully be organized into a series of steps that are monitored by sensors or input from the user. To address the challenges, we adopt a commonly used representation for such tasks, from the origins of cognitive science [21] and still widely used today, i.e., a modular hierarchy of goals and subgoals, terminating in actions that, when executed successfully, lead to task completion. Advantages of this structure for our purposes are that variations such as re-ordering flexible actions or making use of new sensors can be accommodated by low-level restructuring of the hierarchy, and errors in executing a branch of the hierarchy can be locally corrected without disrupting the whole.

Our constraints (2, 3, 4) on the terminal nodes of the model is that they specify actions that can be directly related to sensed states of the world, on the one hand, and can be labeled or depicted in instructions, on the other. These demands lead us to propose that the nodes should conform to a formal representation of meaning. One that we have found highly useful is inspired by predicate logic, a structure that is used in computational semantics and forms the basis for contemporary query languages like SQL [12,15,19,33]. Accordingly, in the task model as used here, terminal nodes are expressed as [relation (argument_1, … argument_n)], where the relations are typically expressed by verbs, and the arguments fill semantic roles such as agents and objects. For example, a relation like "connect" has arguments of object (what is connected) and location (what it connects to), as in "connect(blue_cord, red_plug)." Note that the particular relations used in a task model will be domain-specific. Benefits of this semantic structure are described further below. One is to facilitate linkage to the task monitor. Another is that the translation from a relation-argument structure to the expected sensed consequence is relatively straightforward: Following the previous example, the result should be to sense that a blue cord is connected to a red plug.

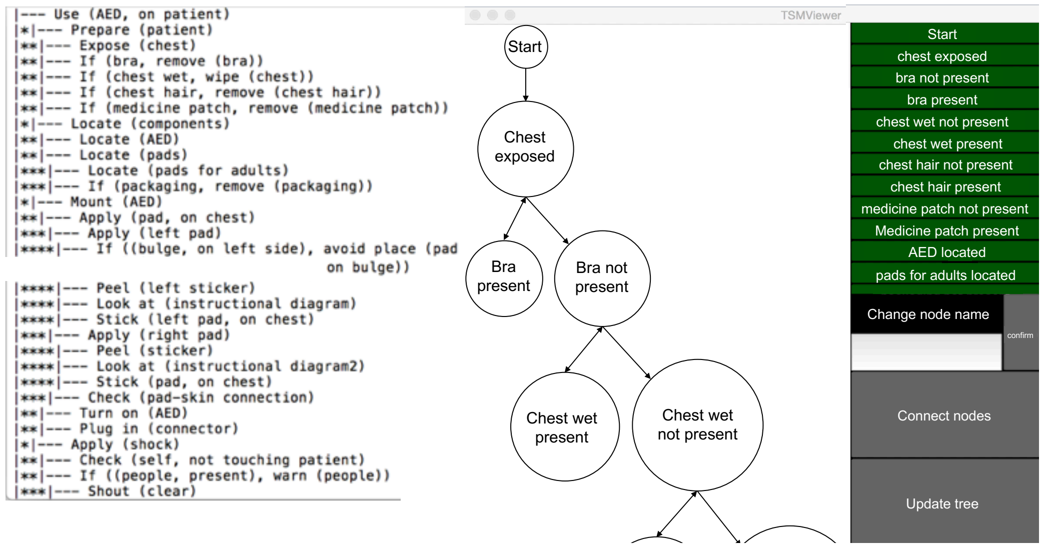
4.2 Task Monitor

The task monitor must provide a linear path through the terminal nodes of the model, describing the desired procedure to complete a task as well as anticipating predictable errors. Our solution is to translate the terminal nodes of the task model, representing actions, into states of the physical world that result from actions and that can be monitored by the sensors in the local device. The relation-argument formalism described above for the task model is, in principle, sufficient to translate its terminal nodes into "normative" states of the task monitor, i.e., what is expected in the physical world after actions in the nodes are completed. However, the monitor must be expanded to take into account the sensory capabilities of the user-borne device as well as potential errors. This leads to the addition of "sensor-assistive" and "error-correcting" states. Sensor-assistive states are needed to overcome limitations of the sensors hosted in the user-borne technology. For example, users of a wrist-worn blood-pressure cuff might need to be queried as to whether their arm is held at heart height, leading to a state that monitors the vocal response to the query. Error-correcting states describe user and apparatus errors that occur frequently enough to be anticipated. For example, a state might be added to detect that a user has plugged in a device or to check for low battery power.

We adopted a structure for the task monitor that is formally a Finite State Machine [11]. It consists of five components: states, next-state functions, input signals, output signals, and output functions. A state is entered with an input signal from the sensors (possibly, the user's response to a query), representing the current status of the physical world. The next-state and output functions compare the input against the normative status for that state translated from the task model. The result of the comparison for the next-state function determines the transition to the following state: either progression or error correction for predictable errors. The monitor may be unable to transition, for example, when non-predictable errors occur or the user's attention lapses. This problem can be detected by long waiting times relative to expectations for that transition. For the output function, the comparison results in an output signal.

The output signal for some states is simply null; that is, the state is passed through without generating an observable result. Other states have an output in the form of an instruction to the user. Each instruction must encompass all the actions that have accumulated, that is, all the terminal nodes in the task model that have been completed, since the previous instruction. Our approach to selecting the instruction states is described below.

Fig. 2 demonstrates the advantage of adopting formal structures for the task model and monitor, in the form of a prototype software interface. The interface initially navigates the task-model to the terminal nodes and converts the relation/argument structure of each node to a state in the task monitor. The result is a finite state describing the status of the physical world, captured by the sensors, that should result when the relation specified in the node is applied to the argument(s). Although this initial step is automated in the current interface, the task monitor must be further augmented with states that anticipate errors and accommodate for limited sensing capability, the output of which will query and correct the user. This augmentation must currently be done by hand, a capability the prototype interface allows with a GUI.



```
|--- Use (AED, on patient)
|*|--- Prepare (patient)
|**|--- Expose (chest)
|**|--- If (bra, remove (bra))
|**|--- If (chest wet, wipe (chest))
|**|--- If (chest hair, remove (chest hair))
|**|--- If (medicine patch, remove (medicine patch))
|*|--- Locate (components)
|**|--- Locate (AED)
|**|--- Locate (pads)
|***|--- Locate (pads for adults)
|***|--- If (packaging, remove (packaging))
|*|--- Mount (AED)
|**|--- Apply (pad, on chest)
|***|--- Apply (left pad)
|****|--- If ((bulge, on left side), avoid place (pad
                                        on bulge))
|****|--- Peel (left sticker)
|****|--- Look at (instructional diagram)
|****|--- Stick (left pad, on chest)
|***|--- Apply (right pad)
|****|--- Peel (sticker)
|****|--- Look at (instructional diagram2)
|****|--- Stick (pad, on chest)
|***|--- Check (pad-skin connection)
|**|--- Turn on (AED)
|**|--- Plug in (connector)
|*|--- Apply (shock)
|**|--- Check (self, not touching patient)
|**|--- If ((people, present), warn (people))
|***|--- Shout (clear)
```

Fig. 2. Left: A task model used as the input of the automation to generate its corresponding task monitor. Center: The output task monitor. Right: The graphical user interface where users can insert additional error and sensor-assistive states.

4.3 Input/Output (I/O)

The I/O component must, of course, specify what sensors are used, but in addition, it must provide instructions, in the form of step-by-step directions to users as they perform the task. As was noted above, instructions are triggered at certain states of the task monitor, but not at all states. Instructions describe a sequence of actions that will be executed in pursuit of a subgoal within the task-model hierarchy, without interruption. We will refer to this sequence as an instructional chunk. A new chunk is delivered at a state where the task monitor registers completion of the last action in the previous chunk. At present there is not automation for determining these states where instructions occur; this requires hand editing of the task monitor. Essentially, the problem corresponds to identifying the boundaries where a chunk-able sequence of actions begins and ends. We propose a set of heuristics for deciding on chunk boundaries, as follows.

(i) *Assign chunks a "reasonable" quantity of content.* One should avoid chopping up action sequences into multiple short instructions. An isolated instruction cannot be too short or trivial, as the users will become frustrated, and the task performance time will be extended by the switches between listening and executing. Conversely, verbal instructional sequences should not have some many components that they exceed the capacity of working memory for words embedded in sentences, which has been estimated to be as low as 2 units [8,13].

(ii) *Avoid long execution times.* Instructional chunks should avoid bundling so many actions that execution time is lengthened to the point of memory failure due to time-dependent decay of memory "strength" [22,23,34]. A general boundary in the literature on working memory loss suggests that actions in a chunk should take less than 15 seconds to complete.

(iii) *Combine steps appropriately.* Steps that are encompassed by a single goal should be in a common instruction; steps that embody different high-level goals should be separated. In addition, instructions for motor output should combine actions that are efficient to execute as a motor unit, such as grasp and twist [26,27]. If the user must report results of an action because sensor data are lacking, the request for the report should be combined with the instruction to do the action ("do X and tell me when X is done").

As examples of how well instructions might match these heuristics, we re-visit the user being guided to take their blood pressure. "Move the strap to an inch below your wrist" is likely to be a useful instruction that should not be decomposed. "Move the strap to an inch below your wrist, set your elbow on the table, and bend your arm until your wrist is as high as your heart" crosses boundaries between multiple subtasks. Even a shorter instruction that combines moving the strap and placing the elbow is vulnerable to a longer execution time than the decay of working memory.

4.4 Technological Implementation

The final component of the assistant, technology, must meet task demands for computational capability, latency, and utility for sensing the observable action sequence. The use of cloudlets offers a solution to the bandwidth and latency problem, but if it is necessary to use computations resident in the cloud, there may be a cost in the form of increased latency. Deficiencies in sensor capability can be compensated for by incorporating the human in the loop; however, the potential addition to latency and error vulnerability makes this a less desirable solution than a direct sensor reading. Clearly, the technological limitations greatly impact the utility of the assistant, as is illustrated by the end-to-end application described next.

5 AED Application

In the remainder of this paper, we present the implementation details of a cognitive assistant for an automated external defibrillator (AED), a device intended to enable the user to administer a shock to a patient exhibiting sudden cardiac arrest, in order to restore normal rhythm. AED devices are ubiquitously placed in institutional and marketing environments, with the expectation that novice users, potentially guided by phone contact with emergency medical technicians, can operate them. They are typically mounted in a box with instructions, augmented by electronic checkpoints on the device itself.

Appendix A provides the task model for the AED, and Appendix B shows the task monitor. To construct the model, we observed AED training sessions of novice users and interviewed Emergency Medical Technician (EMT) experts for detailed accounts of the procedure and common mistakes. In brief, the high-level task goals are: prepare patient (check for chest hair and water, remove clothing); locate AED pads and controller (different pads for young and older patients); mount AED pads on patient (across the heart from upper left to lower right); and apply shock (first clearing area from contact). Common errors in using an AED include not choosing the correct pads according to the patient's age, which could lead to under-shocking adults and overshocking children. Another mistake is mounting the pads at the wrong location on the body; naive users often think they should be at the same height, which would fail to activate the heart. Checks are also needed to protect the operator from shock, which can pass through water or contact with the patient. As the AED we used had a built-in check for contact, we only implemented the water check.

We first report an experiment testing the heuristics presented above for constructing instructions, followed by a description of an end-to-end assistant for the task. The assistant was hosted on Google Glass for the instruction experiment and on a smartphone for the end-to-end implementation. The motivation for using different hardware is described in Section 4.2.

5.1 Instruction Experiment

As noted above, with the task monitor structured and fully expanded to cover sensor capability and error cases, instructions must be generated to occur at appropriate points in task performance. Within the task monitor, each individual state is associated with an instruction as an output (including null). However, it would be non-optimal simply to read out those instructions in sequence. Rather, the instructions should be grouped into chunks according to the heuristics presented above, in order to conform to known constraints of human processing. In this section, we describe an experiment which assesses the effects of violations of those heuristics on performance with the AED. A group without any assistance was included to assess baseline performance.

5.1.1 Participants. A total of 27 participants recruited from the university community completed the study with signed consent (average age 26.22 years old, 12 males). Eleven and 12 people participated in chunked and mis-chunked conditions, respectively, with the remaining 4 assessed for the no-instruction baseline. When only native English speakers are considered, the group Ns were 8, 8, and 3, respectively.

5.1.2 Task Conditions and Procedure. Participants performed the task with an AED training device on a plastic model of a human upper body and head (Resusci Anne™; Fig. 3) laid on the floor of the experimental room. To create a relatively complex scenario, a medication patch was placed on the right chest above the heart level, and an implanted pacemaker was simulated with a bulge on the patient’s chest. The procedure requires the rescuer to remove the medication before applying the AED pads and to avoid placing the pads on top of the implanted device for a secure skin-pad connection. Drops of water were placed on the chest before each session, necessitating the step of drying the chest.

The participants were briefly introduced to the AED and its purpose. They were asked to follow the instructions as closely as possible. Upon completing a step, the participants verbally reported “yes.” At any step, the participants were allowed to ask for a repeat of the current instructions. Although the task monitor specified 25 instructed actions, the AED device issued its own instructions and provided visual signals for the last 8. We found that issuing redundant instructions from the cognitive assistant confused users, so at the point where the AED was activated, we yielded instructions to the device. This left the assisted instructions to 17 actions.

Two versions of the instructions were implemented: For the *chunked* condition, the 17 actions were combined into 13 instructional units corresponding to boundaries at high-level goals, following heuristics of low memory load and combining sensor queries with resulting commands. For the *mis-chunked* condition, the instructions were rephrased so that they violated heuristics by crossing hierarchical boundaries, using longer sentences or separating actions that would otherwise be combined by motor compatibility. Instructions for the two conditions are given in Appendix C.

Note that for every action performed under chunked instructions, the same action was performed under mis-chunked instructions allowing step-by-step comparison of accuracy and speed of performance under the two conditions. In the baseline condition (denoted AED), participants were instructed by the experimenter to open the AED carrier, turn on the machine, and follow the verbal instructions on the AED. This condition is comparable to a real-life emergency situation when there are no trained personnel in the vicinity and the user receives guidance from a 911 operator over the phone.

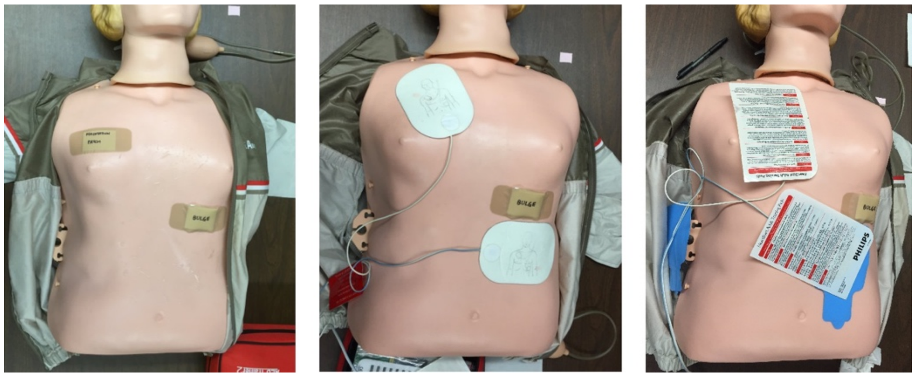


Fig. 3. Left: The medical mannequin used for the user study. Water, a medication patch and a "bulge", representing an implanted device, on the chest to simulate various conditions of the patient. Center: The correct placment of the AED pads. Right: An incorrect placement of the AED pads in the AED condition.

The experiment was conducted with a "Wizard-of-Oz," idealized cognitive assistant represented by an experimenter, who played the role of the sensors and delivered appropriate instructions based on the participant's action. The experimenter also gave error-correcting feedback for predictable errors; if errors were not predicted *a priori*, the instruction was repeated. The instructions and feedback were pre-recorded.

After completing the task, the participants filled out an evaluation survey with 7-point Likert scales assessing the instructions in terms of understandability, pace, amount of information and mental effort required. They also rated their emotional state during the task as to annoyance, irritation, stress, discouragement, and confusion, using a binary scale (yes/no) followed by a 7-point Likert rating. The entire experiment was video recorded for purposes of time measurements.

5.1.3 Results. Performance was assessed with multiple measures: completion rate, step-wise task completion time, frequency of requests to repeat instructions, and errors. As participants sometimes anticipated the end of an instruction, time for a step began at the participant's first action or the end of the instruction, whichever preceded. As the user sometimes forgot to verbally report completion, the time for a step was ended at the terminal action or the user's "yes," whichever preceded. Errors were categorized into two types: *Objective* errors occurred when participants attempted to follow the instructions but failed, such as removing and repositioning an AED pad (which impedes a secure skin-pad connection). *Instruction* errors were failures to comply, such as applying the pads to the wrong location of the chest.

Completion Rate. None of the participants in the AED condition (no assistance beyond the instructions from the device) successfully completed the full procedure. The two principal errors were objective: using the wrong set of AED pads and placing the pads incorrectly. All four participants reached for the pads that were placed on top, which were designed for children. Most AED-condition participants failed to peel off the sticker sheet and placed the pads upside down on the patient's chest (Fig. 3 right). In comparison, all participants in the assisted groups successfully completed the task.

Step-wise task Completion Time. As native-English participants worked considerably faster than non-native English speakers (2 min, 11 sec vs. 3 min, 42 sec, respectively, for total time), we isolated the native speakers for analysis of this measure. The analysis uses the 13 instructional steps as the units of observation, capitalizing on the design feature that the same instruction was embedded in well- and miss-chunked versions across the two groups of participants. Fig. 4 shows the mean time to complete each step (y-axis) when given in the mis-chunked vs. well-chunked conditions (x-axis), averaged across the native-English participants only. The dashed line indicates the expected completion time if the two conditions were equivalent for each chunk. The mis-chunked steps tended to exceed the chunked in duration, particularly as the steps themselves took longer. A linear regression showed that, for each second taken to complete the task under the chunked instructions, an average of 9% more time was needed for mis-chunked instructions. A paired t-test with steps as the unit of analysis found the average completion time to be significantly greater for the mis-chunked condition (Mean = 9.9 sec, *SD* = 4.6) than the chunked (Mean 8.9 sec, *SD* = 5.2), *t*(13) = 2.99, *p* = .04.

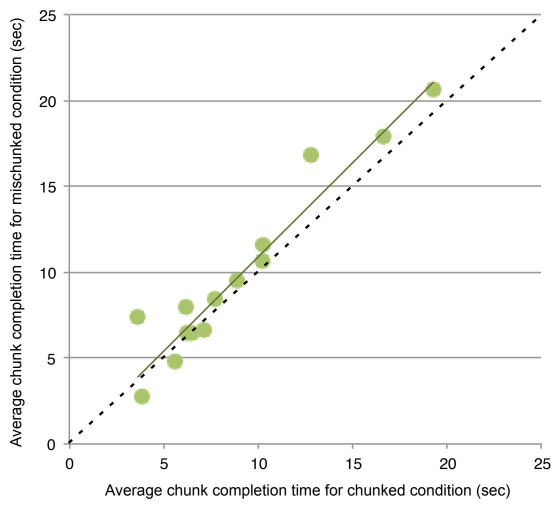


Fig. 4. Average completion time by chunk for the mischunked condition compared to the chunked condition. Dashed line represents expectation for equal times.

Other measures. Requests to repeat instructions were rare, particularly for native speakers. While no native speaker in the chunked condition made an instruction error, 37.5% of the native speakers in the mis-chunked group made at least one such error, with an average of 1 error per subject. Non-native speakers tended to make more instruction errors (average of 2 errors out of 14 chunks and with 2/3 of subjects making at least one error). Of the survey responses, the averages for non-emotional scales tended to be near midpoint, whereas the emotional arousal ratings were low, with one exception: Participants in the mis-chunked condition (native and non-native speakers combined) reported higher stress than those in the chunked condition (Mean chunk = 0.45, *SD* = 0.82; Mean mis-chunk = 1.33, *SD* = 1.50 -- given the high variability in the mis-chunk ratings, the effect reached significance by a 1-tailed test, *t*(21) = 1.76, *p* = .049).

Summary. The key results of the instruction study are (i) cognitive assistance is necessary, because no unassisted participant succeeded in using the AED properly; (ii) violation of chunking heuristics increases processing time; and (iii) mis-chunked instructions result in greater stress for the participant.

5.2 End-to-End Implementation: AED Cognitive Assistant

We implemented a fully automated cognitive assistant guided by the AED task model and task monitor described in the Appendices. The application divided the operation of the AED into subgoals of preparing the patient, detecting age, locating components, mounting the AED, and applying shock. The system relied extensively on existing image-processing software. Where necessary, the initial task model was augmented with states to assist the sensors, and user report was used as the default when no visual recognition system was found to be adequate.

The complete software pipeline is shown in Fig. 5. The experimenter who had served as a surrogate for the sensor and pattern recognition component in the previous experiment was now replaced with computer vision input from a camera. Instructions were either shown on a display or spoken by a speaker. Throughout the entire procedure, the user could say "repeat" to request hearing the previous instruction again; this review was controlled entirely by voice recognition software.

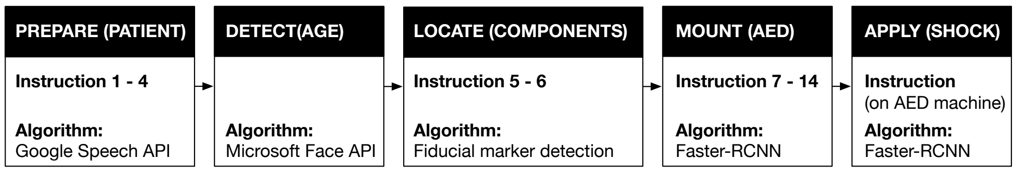


Fig. 5. Software pipeline for the cognitive assistant. Four components represent subgoals in the task model with overt instructions to the user. Detect(age) is an additional sensor-assistive state in the task monitor. Each component is associated with an algorithm.

We initially pilot-tested the AED procedure using a Google Glass. However, the narrow field of view and limited processing capacity of the device, together with its tendency to overheat, ultimately led us to substitute a phone (Motorola Nexus 6), which has greater computational power and a camera with a wider view, allowing us to capture the user's hands when operating the AED. The program was implemented on Android 5.0. However, it is fully compatible with any Android platform (including a Google Glass).

From an implementation perspective, the cognitive assistant software was divided into two parts, a locally resident client processor and a back-end processor on a cloudlet. The client side used the Android phone to capture the visual scene and deliver verbal and visual instructions by speaker. The camera stream was sent from the client to the cloudlet, where image processing was done by the back-end program, which analyzed users' progress and triggered the client to deliver instructions. The communication between the client and the server was done through the Gabriel Platform [37].

5.2.1 Implementation. We next describe the implementation of each component in Fig. 5 and our observation of the factors that affected completion time.

Prepare Patient. This subgoal included instructions to remove clothes, wipe chest if wet, remove chest hair, and check for other implanted devices. The next instruction was issued following the user's verbal report of completion of the previous one, as recognized by Google's Speech API. The completion time for this stage was dominated by user action, as the speech API operated with short latency.

Detect Age. This sensor-assistive state instructed the user to test whether the user should select child or adult pads from the AED box. The pad selection was determined by uploading camera frames from the mobile device to a cloudlet, which invoked the Microsoft Face API to detect the presence of a face and decide whether it belonged to a child or an adult. The completion time was API-dependent and varied with the input image, ranging from a few seconds to on the order of a minute.

Locate Components. If the previous state output that the patient was an adult, the local client instructed the user to take the adult pads from the AED carrier; if a child, the child pads were to be taken. In the current AED product, the two pads differed in size (adult was larger), packaging color and warning label color (blue for child, red for adult). However, the cues available from the product were found insufficient for computer vision algorithms to reliably differentiate the two types of pads. Size could be ambiguous because it varied depending on the camera distance, and color perception depended on lighting and colors of objects in the background. To improve the robustness of pad detection, fiducial markers were developed and added to the sticker side of the pads (Fig. 6). For consistency, the markers were in the same color as the warning labels that came with the AED.

To complete this subgoal of the task, the user first located and opened the package containing the correct pads, which they then held in front of the camera. When the markers on the pads were recognized, the next instruction was delivered. If the user had selected the wrong pad, the monitor entered an error state, and the instruction was to correct the error and hold the new pads up to the camera. The completion time for this stage was dominated by user action, as the fiducial recognition operated in the millisecond range.

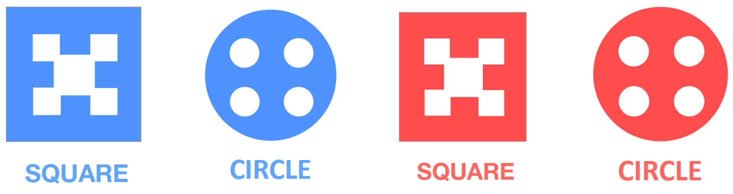


Fig. 6 Left to right, fiducial markers for pads on child's right, child's left, adult's right, adult's left chest.

Mount AED. The first instructions in this subgoal, check for a bulge and peel the stickers, relied on user reports of success. The next instruction was to actually mount the pads. The fiducial markers used for age-appropriate pad selection also enabled the cognitive assistant to distinguish between the left and the right pad. Because the terms *left* and *right* are intrinsically ambiguous (the reference frame can be either the patient or the user), the instruction referred to the shapes of the fiducials. Pads with square markers were to be placed on the right side of the patient's body and pads with circle markers on the left side. If the placement was reversed, an error state was entered, and a correction instruction was delivered.

Critical to the procedure is that the left pad is placed above the heart and the right pad below it, so that the electrical pulse passes through the heart. In order to check for correct placement, the assistant needed to locate the patient's heart position. Because the exact location of the heart cannot be determined precisely from external cues, it was estimated from the shape of the upper body. This proved challenging to computer vision, because the position of the patient was not predictable, the camera allowed only a narrow field of view, and viewing angles could vary. We devised a procedure that attempted to deal with these impediments to machine recognition. The cognitive assistant first constrained how the user viewed the patient by providing an assistive body frame overlay on the screen (Fig. 7). Using the image of the body at the back-end server, it applied Faster-RCNN object detection to detect the patient's body position. Faster-RCNN is a publicly available, state-of-the-art object detection algorithm that combines object detection with hypothesized localization, with demonstrated benefits in terms of processing speed [24]. Our version of the algorithm detected critical joints of the patient, such as shoulders, head, and hips, and used these points to first estimate the position of the patient's body. It then estimated the heart position to be above the body's mid-line between the shoulders and the hips on the patient's left side. The location of the left pad was designated to be on the patient's left side, above the midpoint of the body and below the shoulder joint. The right pad was to be placed below the body's midpoint, on the right side, and above the right hip joint. A benefit of using this detection method was that the cognitive assistant could customize the error messages, e.g., "The pad is placed on the wrong side of the body," or "The pad should be placed on the left side of the body," to better guide the user. The pad position check was performed after the users placed each pad, allowing for timely correction of errors.

With the body-position algorithm running simultaneously on CPU and GPU, the end-to-end latency, including processing and network time, for processing a single camera image was as fast as 33 ms. This was reduced from the 500 ms obtained for an unassisted pose detection algorithm [7] tried initially. However, inadequate viewpoint and low camera resolution could cause failures for multiple frames, extending the processing time. As a result, the latency between aiming the camera and receiving a determination that the pad was correctly or wrongly placed could be on the order of a minute.

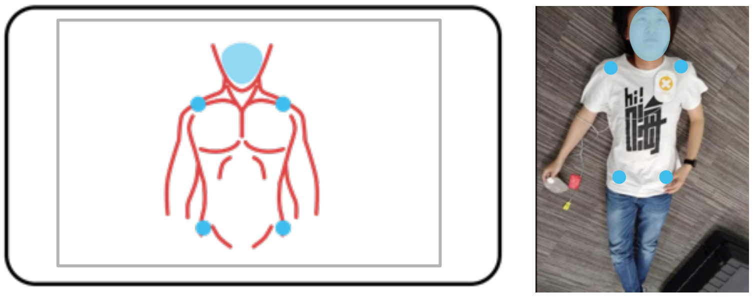


Fig. 7. The assistive body frame overlay derived from patient (right) presented to the user on the phone screen. This helps the user to correctly position the body for joint detection.

The assistant-generated instructions concluded with activating the AED machine, at which point the machine began to issue its own instructions and provided visual signals to guide each step. The assistant still checked user actions for errors. The machine instructed the user to attach a yellow connector to it. In order to check this outcome, an internal tool was employed that automatically creates deep neural-network based object detectors using the core of Faster-RCNN to recognize the connector and the machine. The algorithm achieved 90% accuracy for the machine itself, but only 50% accuracy from a single image of the connector, because the connector lacked a distinctive visual feature. However, this level of accuracy was not an issue in usage, because the algorithm could process 100 – 200 frames/sec. If the AED machine was present and the yellow connector was detected in multiple frames among the 100-200 frames, the algorithm determined that the user had plugged it into the slot on the machine.

The completion time for the entire mount-AED subgoal depended primarily on the latency for pose recognition and the user's actions of placing the pad and attaching the connector, which lasted multiple seconds.

Apply Shock. Once the machine was activated and the connector was placed, an orange button began to blink, indicating it was ready to apply shock. To recognize the orange flash, the same Faster-RCNN-based algorithm was used as devised for the connector in the previous subgoal, but combined with the change in light intensity across time. The algorithm first used canny edge detection to find the location of the orange button. The centroid location of the button was used to build a predicted rectangle, within which the algorithm calculated the light intensity (0-255). The change in light intensity was monitored over time, to determine if the button was flashing or not. Evaluation tests showed that even under strong ambient lighting, there remained a detectable variation between when the button was lit and turned off. The accuracy rate for single flash recognition was approximately 70%, but repeated tests yielded accurate recognition. The rate of flash of the machine constrained the speed with which the algorithm could operate, but typically recognizing the device was ready to shock, once it was activated and connected, took no more than 2 sec.

5.2.2 Usability Test. To test the usability of the AED implementation, 8 naive participants were asked to follow the assistant, in order to complete the procedure of mounting the AED machine on the dummy. The devices were the same ones used in the instruction experiment, and the well-chunked instructions were implemented. The participants were recruited from a local participant pool (average age 31.5 years; 3 males, 2 non-native English speakers). All participants were given the Android phone with the cognitive assistant software and access to a cloudlet. After a short introduction to the AED, the participants were told to follow the instructions from the cognitive assistant and complete the procedures without assistance from the experimenter. After completion, they were asked to rate annoyance, irritation, stress, discouragement, and confusion, using the same 7-point Likert scale as the previous study.

All 8 participants successfully completed the procedure. The average completion time was 7 min, 52 sec (*SD* = 2 min, 18 sec), far longer than the completion time in the instruction experiment. The disadvantage arises because unlike the human perception used in the Wizard-of-Oz version, the automated cognitive assistant relies on non-optimized or non-dedicated machine classification algorithms. As Chen et al. [10] noted, the computations required by the application, and not the communication between the local device and the cloudlet, have the greatest impact on latency. A salient problem was the relatively long latency for the Microsoft Face API and pad location detection; most users had to hold the phone for up to a minute before the algorithms returned results. The slow response with the Face API reflects its being a proprietary cloud service that is responding at any time to a large user population. Similarly, latencies were lengthened by reliance on the Google Speech API, which has a set timeout, after which the user has to press a button to restart the speech recognition.

Participants' average ratings for negative emotion experiences of irritation, discouragement, annoyance, and confusion ranged from 1-3, higher than in the previous experiment where the human played an idealized assistant, but still below the midpoint of the scale. Two participants specifically expressed irritation and annoyance with the speech-recognition timeout. Despite the delays and the demands of aiming the camera, the stress ratings with the cognitive assistant (Mean = 1.50, *SD* = 1.60), were essentially the same as ratings for the mis-chunked human instructions in the instruction study (Mean = 1.33).

While the user study points to limitations of the current implementation, it is important to note the fundamental outcome, namely, that the end-to-end version of the AED assistant enabled all assisted users to complete the task. This is in direct contrast with the failure of the unassisted novice participants tested in the instruction study. Moreover, the users reported dissatisfaction particularly with latencies that seem most likely to be improved with future technological advances. Recognizing the age of the face, for example, could be efficiently implemented on a cloudlet. Speech recognition failures could be addressed with a more advanced implementation. Latencies during pad position detection could be reduced by using a multi-algorithm approach, similar to [10].

6 General Discussion

Despite the rising popularity of cognitive assistants enabled by new technologies, many challenges still remain. The first section of this report identified challenges facing the designer of a technologically advanced cognitive assistant hosted in an on-board device, capable of real-time guidance including error correction, and exhibiting self-contained sufficiency to guide the user through a complex task, such as assisting non-professionals in a health-care related task. In order to address these challenges, we suggested constraints on the kinds of tasks that are candidates for cognitive assistants. A critical assumption is that the human user could be not only a target of, but also embedded into, the solutions. We further described formal approaches that would be useful in constructing the assistant. Among these were guidelines for instructional units optimized for task-related coherence and comprehension.

While our solutions are applicable to constructing cognitive assistants for various purposes and domains (under the proposed task constraints), in the remainder of the paper, we illustrated this approach by user testing and implementation, culminating in an end-to-end, fully automated cognitive assistant for AED usage. Notably, "full automation" here includes extensive use of the human in the loop. An initial study demonstrated that novice users relying only on the on-board instructions provided with the AED could not perform the task. An idealized assistant, in the form of a human experimenter, could guide them through, even with non-ideal (mis-chunked) instructions that deliberately violated heuristics tailored to human cognitive constraints (though at some cost). The subsequent demonstration with the automated system found that novice users could follow the assistant to mount an AED machine on a dummy patient. However, the study also showed that current technology, even using state-of-the-art machine learning algorithms and augmenting the basic device with fiducial markers, produced latencies that caused significant delays and user dissatisfaction.

End-to-end latencies are a challenge exposed by the work that seems likely to improve rapidly with new hardware and software technology. Chen has explored some methods to reduce this latency, such as using a multi-algorithm approach or adding a GPU [10]. Another challenge lies with the user interface, which must enable the rigorous and responsive sensor checks necessary for the cognitive assistant to monitor normative progress and detect errors or time-outs. We found that the small field of view in the Google Glass limited how well computer vision algorithms work on the images captured. Moreover, it lacked even the minimal processing capacity for seamless communication with cloudlet resources, often timing out due to overheating. The smartphone was superior in these respects, but holding mobile phones in the hands restrains users' manual operations, while mounting it on a helmet can impede communication by voice or button press. It is our hope that further implementation of cognitive assistants will develop in tandem with advanced wearable sensor technology.

Other challenges are to construct the essential components of the task model and task monitor. In contrast to the current hands-on approach, it would be preferable to extract the basic task structure automatically from available sources such as on-line demonstrations and tutorials. We described preliminary efforts towards automatically generating a task monitor from a task model. A further goal is to use the model and monitor for purposes of automated instruction generation. Progress could be made by exploiting the semantic representation underlying the task model.

Despite these limitations, the success of our preliminary efforts and other demonstrations from our group [10] is encouraging for further development of cognitive assistants in time-constrained application domains. In this regard, the inability of novices to perform the task of administering the AED without assistance is striking, as these devices are commonly mounted in work environments with the expectation of by-passer intervention. In our scenario for the future, the ubiquitous AED wall-mounted enclosure will contain a cognitive assistant enabled with local edge-computing access and appropriate wearable hardware.

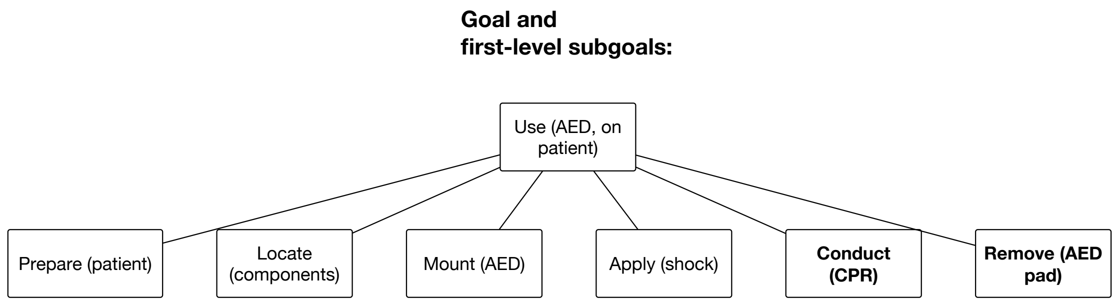
7 Acknowledgements

This work was supported by the National Science Foundation (NSF) under grant number CNS-1518865.

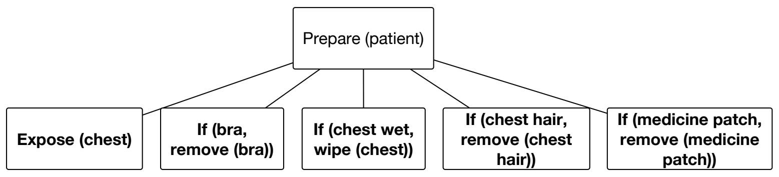
8 Appendix

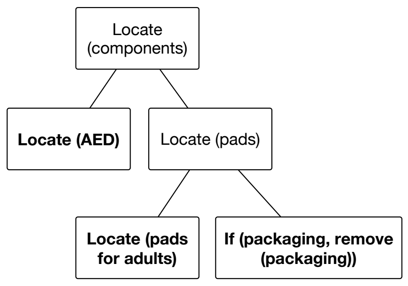
8.1 Task Model for the AED Task

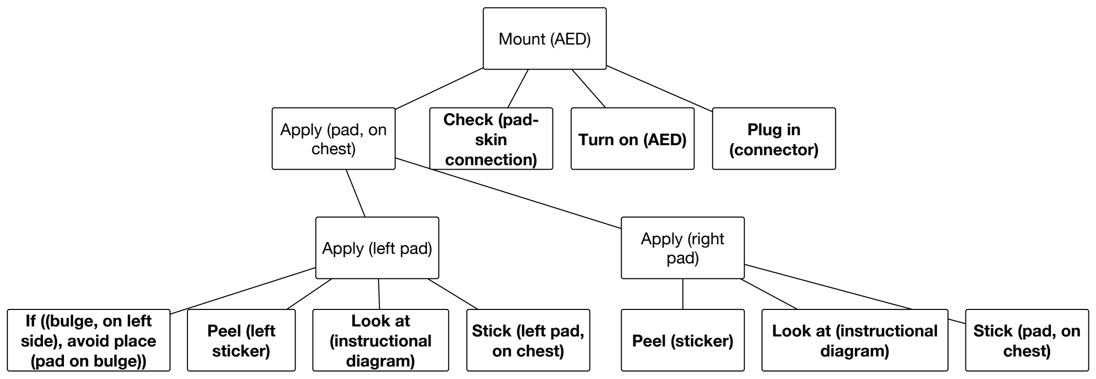
For space reasons, the first-level subgoals and subsequent nodes are presented separately from the main goals. Terminal nodes are in bold.

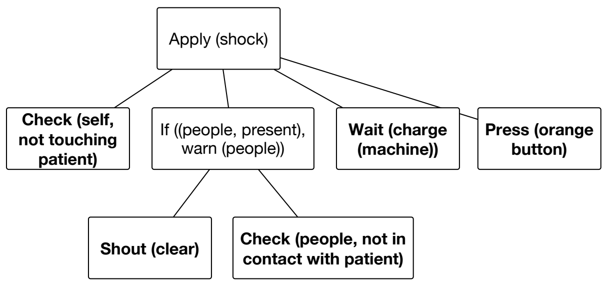


Subgoals and terminal nodes:

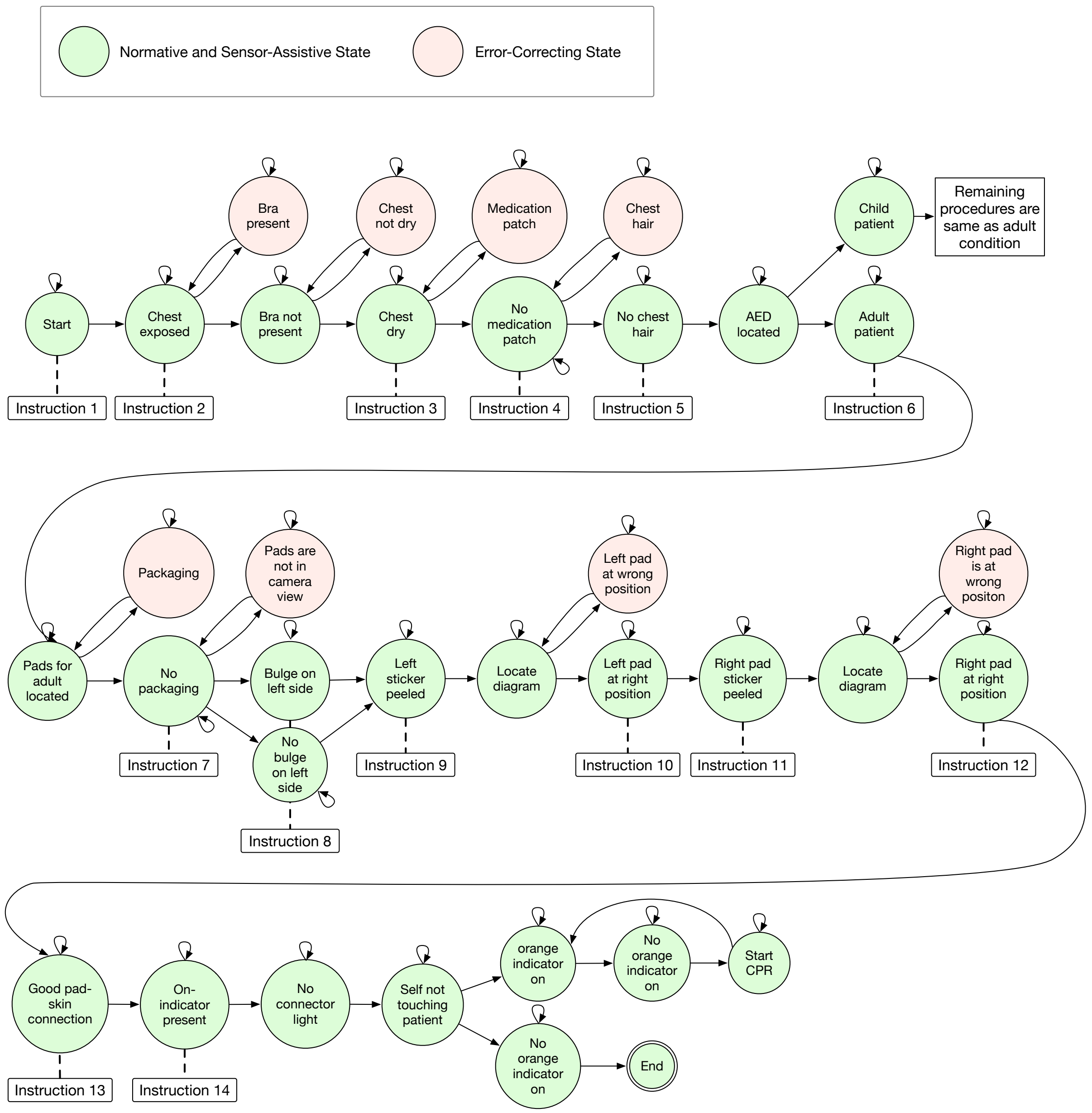

Prepare (patient)
Expose (chest)
If (bra, remove (bra))
If (chest wet, wipe (chest))
If (chest hair, remove (chest hair))
If (medicine patch, remove (medicine patch))




Mount (AED)
Apply (pad, on chest)
Check (pad-skin connection)
Turn on (AED)
Plug in (connector)
Apply (left pad)
Apply (right pad)
If ((bulge, on left side), avoid place (pad on bulge))
Peel (left sticker)
Look at (instructional diagram)
Stick (left pad, on chest)
Peel (sticker)
Look at (instructional diagram)
Stick (pad, on chest)


Apply (shock)
Check (self, not touching patient)
If ((people, present), warn (people))
Wait (charge (machine))
Press (orange button)
Shout (clear)
Check (people, not in contact with patient)

8.2 Task Monitor for the AED Task



8.3 Instructions for Chunked and Mis-chunked Conditions of the Instruction-Experiment

Chunked Condition	Mischunked Condition
1 Remove the clothes to expose the patient's chest. Remove the bra if present	**1** Remove the clothes to expose the patient's chest.
2 If there is water on the chest, wipe it dry.	**2** Remove the bra if present.
3 If there is a medication patch on the chest, remove it.	**3** If there is water on the chest, wipe it dry. If there is a medication patch on the chest, remove it. *(Cross hierarchical*
4 If there is thick chest hair on the upper chest opposite the patient's heart, remove it with scissors or tape.	**4** If there is thick chest hair on the upper chest opposite the patient's heart, remove it with scissors or tape.
5 Look for pads for adults in the AED carrier.	**5** Look for the type of pads that are designed for adults, which you should find somewhere inside the AED carrier after you open up the carrier. *(Long sentence)*
6 If the pads are inside a pouch, take them out.	**6** If the pads are inside a pouch take them out. The patient may have a pacemaker implanted on the side of the body below the heart. Look at the side of the body below the heart. *(Cross hierarchical boundaries & long sentence)*
7 The patient may have a pacemaker implanted on the side of the body below the heart. Look at the side of the body below the heart. If you see a bulge there, say "yes". Otherwise, say	**7** If you see a bulge there, say "yes". Otherwise, say "no".
8 Take the pad with the blue cord and peel the sticker from the back. Hold the pad in camera view.	**8** Take the pad with the blue cord and peel the sticker from the back. Hold the pad in camera view.
9 Look at the picture on the front of the pad. Firmly stick the pad as indicated, below the patient's heart and on the side of	**9** Look at the picture on the front of the pad.
10 Take the pad with the white cord and peel the sticker from the back. Hold the pad in camera view.	**10** Firmly stick the pad as indicated, below the patient's heart and on the side of the body.
11 Look at the picture on the front of the pad. Firmly stick the pad as indicated, opposite the heart and above heart level.	**11** Take the pad with the white cord and peel the sticker from the back. Hold the pad in camera view. Look at the picture on the front of the pad. *(Separated actions that should be*
12 Press the pads against the skin to make sure they are firmly attached.	**12** Firmly stick the pad as indicated in the picture on the front of the pad, opposite the patient's heart and above the patient's heart level. *(Long sentence)*
13 Press the green button to turn on the machine.	**13** Press the pads against the skin to make sure they are firmly attached. Press the green button to turn on the machine. *(Cross hierarchical boundaries)*
14 Follow the instructions from the AED.	**14** Follow the instructions from the AED.

9 References

1. Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. 2003. Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics* 22, 3: 828. https://doi.org/10.1145/882262.882352
2. John Annett and Neville Stanton. 2008. Task Analysis. In *International Review of Industrial and Organizational Psychology 2006*. https://doi.org/10.1002/9780470696378.ch2
3. Davide Anzalone, Marco Manca, Fabio Paternò, and Carmen Santoro. 2015. Responsive task modelling. https://doi.org/10.1145/2774225.2775079
4. Farah Arab, Jérémy Bauchet, Hélène Pigot, Anaïs Giroux, and Sylvain Giroux. 2014. Design and assessment of enabling environments for cooking activities. In *UbiComp 2014 - Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. https://doi.org/10.1145/2638728.2641329
5. Chris Baber and Neville A. Stanton. 1994. Task analysis for error identification: A methodology for designing error-tolerant consumer products. *Ergonomics* 37, 11: 1923–1941.

https://doi.org/10.1080/00140139408964958
6. Robert Benfer, Edward Brent, and Louanna Furbee. 2012. Knowledge Acquisition. In *Expert Systems*. https://doi.org/10.4135/9781412984225.n3
7. Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2016. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. Retrieved December 11, 2017 from http://arxiv.org/abs/1611.08050
8. William G. Chase and Herbert A. Simon. 1973. Perception in chess. *Cognitive Psychology* 4, 1: 55–81. https://doi.org/http://dx.doi.org/10.1016/0010-0285(73)90004-2
9. Zhuo Chen, Lu Jiang, Wenlu Hu, Kiryong Ha, Brandon Amos, Padmanabhan Pillai, Alex Hauptmann, and Mahadev Satyanarayanan. 2015. Early Implementation Experience with Wearable Cognitive Assistance Applications. *Proceedings of the 2015 Workshop on Wearable Systems and Applications*: 33–38. https://doi.org/10.1145/2753509.2753517
10. Zhuo Chen, Roberta Klatzky, Daniel Siewiorek, Mahadev Satyanarayanan, Wenlu Hu, Junjue Wang, Siyan Zhao, Brandon Amos, Guanhang Wu, Kiryong Ha, Khalid Elgazzar, and Padmanabhan Pillai. 2017. An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing - SEC '17*, 1–14. https://doi.org/10.1145/3132211.3134458
11. Pong P. Chu. 2006. Finite State Machine: Principle and Practice. . John Wiley & Sons, Inc., 313–371. Retrieved from http://dx.doi.org/10.1002/0471786411.ch10
12. Edgar Frank Codd. 1970. A relational model of data for large shared data banks. *Communications of the ACM*. https://doi.org/10.1145/362384.362685
13. Meredyth Daneman and Patricia A Carpenter. 1980. Individual differences in working memory and reading. *Journal of verbal learning and verbal behavior* 19, 4: 450–466.
14. Marie-Paule Daniel and Barbara Tversky. 2012. How to put things together. *Cognitive processing* 13, 4: 303–19. https://doi.org/10.1007/s10339-012-0521-5
15. Charles J Fillmore. 1967. The case for case.
16. James D Hollan, Edwin L Hutchins, and Louis Weitzman. 1984. STEAMER: An interactive inspectable simulation-based training system. *AI magazine* 5, 2: 15.
17. Krishnanand Kaipa, Carlos Morato, Boxuan Zhao, and Satyandra K. Gupta. 2013. Instruction Generation for Assembly Operations Performed by Humans. https://doi.org/10.1115/detc2012-71266
18. Rushil Khurana, Karan Ahuja, Zac Yu, Jennifer Mankoff, Chris Harrison, and Mayank Goel. 2018. GymCam: Detecting, Recognizing and Tracking Simultaneous Exercises in Unconstrained Scenes. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. https://doi.org/10.1145/3287063
19. Walter Kintsch. 1974. The representation of meaning in memory.
20. Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*. https://doi.org/10.1038/nature14539
21. George A. Miller, Eugene Galanter, and Karl H. Pribram. 1960. *Plans and the structure of behavior*. https://doi.org/10.1037/10039-000
22. Lloyd Peterson and Margaret Jean Peterson. 1959. Short-Term Retention of Individual Verbal Items. *Journal of Experimental Psychology* 58, 3: 193.
23. Judith S. Reitman. 1974. Without surreptitious rehearsal, information in short-term memory decay. *Journal of Verbal Learning and Verbal Behavior* 13, 4: 365–377. https://doi.org/http://dx.doi.org/10.1016/S0022-5371(74)80015-0
24. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6: 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
25. Steven Ritter, John R. Anderson, Kenneth R. Koedinger, and Albert Corbett. 2007. Cognitive tutor: applied research in mathematics education. *Psychon Bull Rev* 14, 2: 249–255. https://doi.org/10.3758/BF03194060
26. David A Rosenbaum. 2009. *Human motor control*. Academic press.
27. David A. Rosenbaum, Caroline M. van Heugten, and Graham E. Caldwell. 1996. From cognition to biomechanics and back: The end-state comfort effect and the middle-is-faster effect. *Acta Psychologica* 94, 1: 59–85. https://doi.org/http://dx.doi.org/10.1016/0001-6918(95)00062-3
28. Mahadev Satyanarayanan, Zhuo Chen, Kiryong Ha, Wenlu Hu, Wolfgang Richter, and Padmanabhan Pillai. 2014. Cloudlets: at the Leading Edge of Mobile-Cloud Convergence. *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services*: 1–9. https://doi.org/10.4108/icst.mobicase.2014.257757
29. Mahadev Satyanarayanan, Wei Gao, and Brandon Lucia. 2019. The Computing Landscape of the 21st

Century. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications* (HotMobile '19), 45–50. https://doi.org/10.1145/3301293.3302357

30. Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2015.7298682
31. Rags Srinivasan and Agnieszka Zielinksa. 2019. *Data at the edge: Managing and activating information in a distributed world*. Retrieved from https://www.stateoftheedge.com/reports/data-at-the-edge-2019/
32. Neville A. Stanton and Christopher Baber. 2005. Validating task analysis for error identification: Reliability and validity of a human error prediction technique. *Ergonomics*. https://doi.org/10.1080/00140130500219726
33. Lucien Tesnière. 1959. *Eléments de syntaxe structurale*. Librairie C. Klincksieck.
34. John N. Towse and Graham J. Hitch. 1995. Is there a Relationship between Task Demand and Storage Space in Tests of Working Memory Capacity? *The Quarterly Journal of Experimental Psychology Section A* 48, 1: 108–124. https://doi.org/10.1080/14640749508401379
35. VIZRTech. 2017. Angel On Your Shoulder: Google Glass, virtual screens, and cloudlet computing. Retrieved from https://www.youtube.com/watch?v=DANM2W1gVEI
36. Bob J. Wielinga, B. Bredeweg, and Joost A. Breuker. 1988. Knowledge acquisition for expert systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/3-540-50676-4_11
37. 2013. Gabriel: Platform for Wearable Cognitive Assistance Applications. Retrieved from https://github.com/cmusatyalab/gabriel