BDI personal medical assistant agents: The case of trauma tracking and alerting

BDI Personal Medical Assistant Agents: The Case of Trauma Tracking and Alerting

Angelo Croatti[a,1], Sara Montagna[a,1], Alessandro Ricci[a,1], Emiliano Gamberini[b], Vittorio Albarello[b], Vanni Agnoletti[b]

[a] *Computer Science and Engineering Department (DISI), University of Bologna*
Campus of Cesena, Via dell'Università 50 – Cesena, Italy
emails: {a.croatti|sara.montagna|a.ricci}@unibo.it
[b] *Intensive Care Unit/Trauma Center, M. Bufalini Hospital, Cesena, Italy*
emails: {emiliano.gamberini|vittorio.albarello|vanni.agnoletti}@auslromagna.it

Abstract

Personal assistant agents can have an important role in healthcare as a smart technology to support physicians in their daily work, helping to tackle the increasing complexity of their task environment. In this paper we present and discuss a personal medical assistant agent technology for trauma documentation and management, based on the Belief-Desire-Intention (BDI) architecture. The purpose of the personal assistant agent is twofold: to assist the Trauma Team in doing precision tracking during a trauma resuscitation, so as to (automatically) produce an accurate documentation of the trauma, and to generate alerts at real-time, to be eventually displayed either on smart-glasses or room-display.

Keywords: Personal Assistant Agent, Trauma Resuscitation, Medical Alerts, Tracking, BDI Agents

1. Introduction

Personal assistant agents are a well-known application of software agents [1, 2, 3]. Existing proposal and technologies have been developed for different kind of purposes and capabilities, from scheduling joint activities (e.g., [4]), task management [5], to monitoring and reminding users of key time-points (e.g., [6]), sharing information, assisting in negotiation decision support (e.g., [7]). Generally speaking, with personal assistant agents the user is engaged in a cooperative process in which human and computer agents both initiate communication, monitor events, and perform tasks [2]. This is based on the metaphor that of a *personal assistant* who is *collaborating with the user* in the same work/task environment [2].

In the context of healthcare, while intelligent agents have been applied for different purposes [8, 9, 10], the use of personal assistant agents has been explored only in the Ambient Assisted Living domain [11, 12]. Nevertheless, the computing power of modern mobile devices and wearable computing devices [13, 14, 15] (e.g., smart-glasses), their sensors equipment and the possibility to interact with network/cloud-based services, are enabling factors to conceive a new generation of mobile/wearable personal assistant agents assisting physicians in their daily work.

In this paper we discuss the application of a personal assistant agent technology for trauma tracking and alerting, based on the Belief-Desire-Intention (BDI) agent architecture [16, 17]. The project – called TraumaTracker – has been developed in cooperation with the Emergency Department, a Trauma Center, of a hospital in Italy (Bufalini Hospital, Cesena). The personal assistant agent – referred in the following as Trauma Assistant Agent – provides two main functionalities. The first is about trauma tracking, i.e. the problem of producing an accurate documentation of a trauma resuscitation. Trauma documentation is known to be a challenging task [18]. Nowadays most of the Emergency Departments

[1] These authors contributed equally to this work.

adopt handwritten paper records and flow sheets for acquiring data [18, 19]. Data acquisition is often inaccurate and crucial data are lacking [18]. The Trauma Assistant Agent assists the Trauma Leader/Team in tracking/recording everything that could be relevant for documenting the trauma during trauma resuscitation, maximising accuracy and minimising the burden for the physicians. The second functionality concerns the generation of *alerts* during the management of a trauma, useful to help preventing dangerous situations for the patient and to improve the performance/quality of Trauma Team action. To this purpose, the Trauma Assistant Agent embeds and enacts the knowledge (rules) about alert generation specified by the Team, carefully designed in order to avoid over-alerting problems and to address the specific needs of the Team.

The detailed discussion of the trauma tracking and documentation problem, and the evaluation of TraumaTracker as a tool for that purpose, can be found here [20]. In this paper we focus instead on the design and implementation of the Trauma Assistant Agent upon the BDI architecture and technologies, and, in particular, on the alert generation feature. In order to validate the system, the prototype has been used by a Trauma Team of 8 physicians, who has collected more than 430 reports in about 9 months, from January 2018 to September 2018. The experimentation developed so far has been effective as a first validation of the the expected benefits as well as to understand issues/challenges to be tackled in ongoing and future work.

The remainder of the paper is organised as follows. In Section 2 we provide an overview of related works. In Section 3 we introduce TraumaTracker, describing in particular the coarse-grained architecture of the system, including the Trauma Assistant Agent. In Section 4 we discuss in detail the design and implementation of the Trauma Assistant Agent based on the BDI model, focussing in particular on the alert generation in Section 5 and its evaluation in Section 6. Finally, in Section 7 we draw some conclusions, briefly depicting our ongoing and future work.

2. Background and Related Works

In the last two decades we witnessed to a progressive computerisation of healthcare services, whose quality, efficacy and efficiency significantly improved thanks to the introduction of novel ICT infrastructures and services: acquiring and sharing patient data through Electronic Medical Records (EMR), providing telemedicine services, remote and mobile monitoring, diagnostic tools, medical alerts, just to cite a few [21, 22, 23, 24].

Since this paper presents an agent-based framework for trauma tracking and alerting, in this section we provide an overview of motivations and challenges of the domain and discuss related applications of agents in healthcare.

2.1. The issues of Emergency Departments

Emergency Department (ED) is one of the most critical and challenging hospital departments, since it requires reactivity, quick and coordinated response, fast-paced and accurate decision-making. In this scenario, ICT services can positively support physicians in performing different activities, listed in the following. Real-time documentation of truama is the first key task: an accurate documentation of trauma resuscitation is crucial to improving the quality of trauma care where, according to [25] "Quality of trauma care can be defined as achieving the best possible outcome for a given set of clinical circumstances". Reasoning and reaching a decision on the best treatment and care in very little time is the second major task: doctors must monitor simultaneously different parameters – a big amount of vital signs and factors related to patient safety – and make decision on the next procedure in the fast-paced scenario of trauma resuscitation. Finally, coordination among the various actors involved in the trauma management is crucial to assure the best possible outcome.

The TraumaTracker project presented in this paper aims at developing a system for tackling the first two issues. Hereafter we discuss challenges and related literature.

2.1.1. Trauma Tracking

The documentation of a trauma, known in literature as *trauma documentation*, is meant to be acquired during the process of trauma resuscitation, reporting where and when crucial events occurred, which and when treatments are given, procedures are performed, and finally it should report repeated vital signs measures.

However nowadays, data acquisition is often inaccurate and crucial data are lacking. This is mainly due to characteristics proper of the context: multitasking of the person in charge of acquiring data, parallel activities of the different team members, multiple data to be recorded, retrospective documentation from collective memory, several and quickly oscillating vital signs to be monitored.

2.1.2. Trauma Alerts

Generally speaking, the generation of alerts in healthcare domain aims at warning caregivers of risks, thus improving patient safety. During a trauma resuscitation, physicians must perform several concurrent actions in very short time, such as monitoring several vital signs at a time, verifying if they change as expected after making an ad-hoc action, keeping track of *time-dependent* procedures —e.g., tourniquet must be activated for a fixed time. Given the fast-paced scenario this becomes a challenging task and alerts can support physicians warning about, for instance, time passed or unexpected changes in vital signs. Nowadays in this context, automatic alerts are usually generated by those medical devices that monitor vital signs, if they are out of the safe range. However, vital signs of critical patients vary quickly and are often out of normal ranges. The evaluation of patient's clinical status thus requires a broader and more aware view of the entire trauma resuscitation process, that relates vital signs values with past procedures and events. A system able to perform an automatic reasoning on these parameters can reduce human errors and improve trauma outcome.

2.2. Alerts in healthcare domain

Independently of the specific application, the generation of alerts in healthcare is a lively research field. A lot of telehealth and telemedicine work is devoted to real-time monitoring of vital signs and environmental data in order to early detect specific disease symptoms or health decline, and generate accordingly automated alerts [26, 27]. Common applications are home care for elderly support, chronic disease management. Medication errors or adverse drug events are other major issues that benefit from ICT services. Clinical decision support systems based on machine learning and rule-based systems are the most common approaches adopted for this purpose.

However, literature highlights also potential risks for patient safety resulting from the application of ICT for alert generation. In [28] this issue is properly discussed: authors review the attitude of physicians towards alerting in different hospitals from different countries, totalling 1,018 questionnaires. The main relevant outcome of this study is that – in physicians opinion – automatic alerting often triggers irrelevant alerts, causes alert fatigue and overload which may be annoying but, at the same time, the prevention of serious errors and problems is perceived as an unquestionable benefit and the information provided are not useless nor meaningless, concluding that alerts are not waste of time. To reduce the burden, physicians suggest to adapt alerting to the clinical context, make use of more sophisticated ways to display alerts and divide them according to a hierarchy of severity.

2.3. Agents in Healthcare

Multi-agent systems have been longly recognised as a powerful approach to improve the performance of an ICT infrastructure in terms of interoperability, scalability and reconfigurability. According to [9], the main success factor is that the intrinsic properties of a multi-agent system successfully model heterogeneous, distributed and autonomous systems, such health care ones are. Moreover the application of agents is promising, and much work has been done in literature, as personal assistant software.

Literature refers a wide range of agents applications in healthcare for different purposes. As reviewed in [8, 29, 9], agent-based technologies is adopted for: *(i)* Medical data management: accessing,

integrating, sharing and archiving patients' data from different remote sources is crucial for easing the work of physicians and for statistical analysis purposes [30, 24]; *(ii)* Decision support systems and knowledge based systems: supporting physicians in decision making [31, 32]; *(iii)* Planning and resource allocation: scheduling decisions on the allocation of professional and physical resources must be coordinated by planning techniques [33]; *(iv)* Remote/home-care: mainly devoted to remote patient monitoring and surveillance, it allows on one side patients to not travel towards healthcare facilities for vital signs check-up, on the other side physicians to observe the dynamic of patient's health and provide opportune recommendations tailored to the patient [34, 35, 36, 37]; [36] proposes for instance agents as parts of a model that imitates the behaviour of users and tools in the environment, and, through simulation, it generates data to feed the system to recognise user's activities.

Personal Assistant Agent. Agents as personal medical assistant have been recently adopted in the healthcare domain to supporting patients, physicians or caregivers. The applications we found in literature are mainly devoted to supporting patients in their daily activities. They are usually part of Ambient Assisted Living applications [11, 12], where vital signs of patients and contextual information are acquired by sensors to provide personal assistant agents with all the data needed to gather knowledge on users needs and behaviour. Accordingly, they adapt themselves to improving the given assistance, e.g. reminding users about medication schedule, identifying anomalies in patients' health status, notifying autonomously caregivers when abnormal values are detected. [11] presents AMBRO, an IoT platform supplied with an "intelligent cloud system layer" where personal assistant agents learn on data and events acquired by the system and act by sending notification alerts to caretakers. [12] reports about the HERA (Home sERvices for specialised elderly Assisted living) project where a Personal Assistant Agent supports an elderly user – affected by chronic, Alzheimer or mild cognitive impairment disease – following his/her daily routine and adapting its services to his/her habitual pattern.

However, some exceptions are devoted to support clinicians for diagnosis purposes: for instance PRO*forma* [38] has been proposed as a novel agent programming language specifically designed to model medical expertise.

BDI Agents in Healthcare. Even though applications of the agent paradigm have been presented extensively in literature, few works elaborated on the adoption of the BDI architecture, as we propose in this paper. [39, 40] present ALZ-MAS, a distributed multi-agent system based on BDI agents for monitoring Alzheimer patients' health. Five agents structure ALZ-MAS and are equipped with reasoning and planning mechanisms to manage diverse context-aware information from users and their environment. [41] uses BDI agents for developing ubiquitous systems for healthcare environments. The BDI model is adopted in [42] for implementing a multi-agent system that acquires data from a Body Area Network, tracks and evaluates individual body movements with the aim of detecting anomalies.

Agents in Emergency scenario. Some works propose the application of agents in emergency scenario. An example is presented in [43], about the Ubimedic2 agent framework for supporting rescue operations. [24] presents CASCOM, a distributed multi-agent system for the execution of smart emergencies by providing efficient remote healthcare in case of unexpected events. Within the platform distributed data and information can be retrieved and make available to physicians everywhere, thus enabling easier and faster choices. In [44] a preliminary work – relying on BDI agents – for monitoring and assisting patients is presented. The system is validated in a limited scenario, but the paper demonstrates that the BDI approach fits with this context.

This paper is the extended and revised version of [45], including two new important parts. The first one is about the alerting feature, which was not part of the first version of Trauma Tracker. The second one is the detailed description of the BDI model and implementation of the Trauma Assistant Agent, focusing in particular on the design of the rules for generating alerts in terms of agent plans.

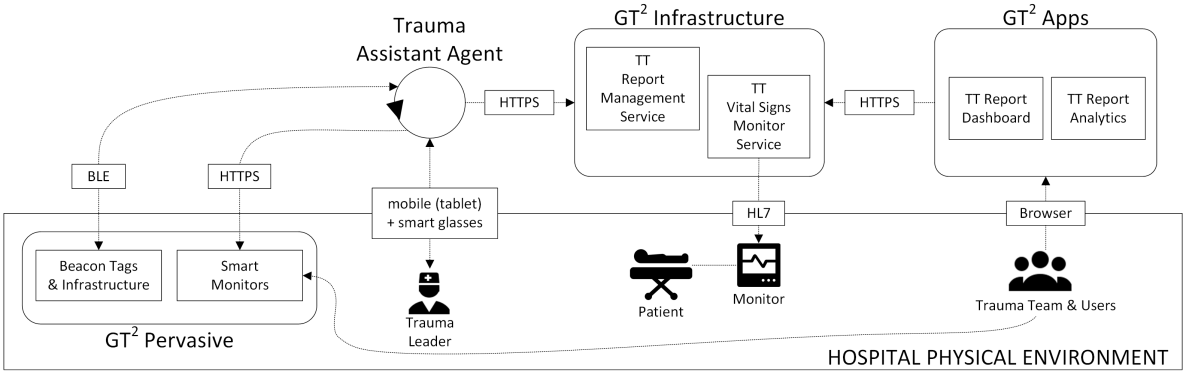


Figure 1: Trauma Tracker Coarse-Grained Architecture.

The detailed discussion of the trauma tracking and documentation problem, and the evaluation of Trauma Tracker as tool for that purpose, appeared in a previous paper ([20]). This paper can be considered a follow up of that work, in which we introduce and discuss trauma alerting as new feature of Trauma Tracker.

3. The Trauma Tracker Project: An Overview

TraumaTracker (TT) has been conceived and designed by taking in consideration the structure and work organisation of the Trauma Team. The team leader – called *Trauma Leader*, usually a senior official – supervises the work of the team during trauma resuscitation and is in charge of producing the documentation paper (report) at the end of the trauma management.

The first main objective of the system is to track relevant information of traumas managed in the Trauma Center, to increase both the quality and quantity of the collected data and to provide a flexible and comprehensive way to manage and analyse such data, structured in reports.

The second main objective is about the automatic generation of *alerts* about crucial situations that the Trauma Team may want to be notified. Situations may concern both the current state of the patient or its evolution, and the temporal flow of actions carried on by the Trauma Team — e.g. some time t has elapsed after the administration of some drugs.

The coarse-grained general architecture of TraumaTracker is shown in Figure 1 and is composed by:

- the Trauma Assistant Agent, the software running on a mobile (a tablet) brought by the Trauma Leader.
- the GT^2 infrastructure, a set of Web-based services deployed in the hospital local area network, functioning as back-ends.
- the GT^2 pervasive, a set of services provided by devices deployed in the physical environment of the hospital—e.g., a beacon-based infrastructure for localisation service.
- the GT^2 apps, a set of Web-apps enabling users to access and interact with some of the GT^2 infrastructure services.

The Trauma Assistant Agent also controls what displayed by the wearable device (smart-glasses) possibly used by the Trauma Leader and on displays available in the environment (e.g., in the shock room). Figure 2 shows some snapshots of the Trauma Assistant Agent offered by the Trauma Assistant Agent. The tablet is used basically to register medical actions and procedures, annotate drug administrations, patient information and so on. The smart-glasses are used to inform the Trauma Leader about alerts

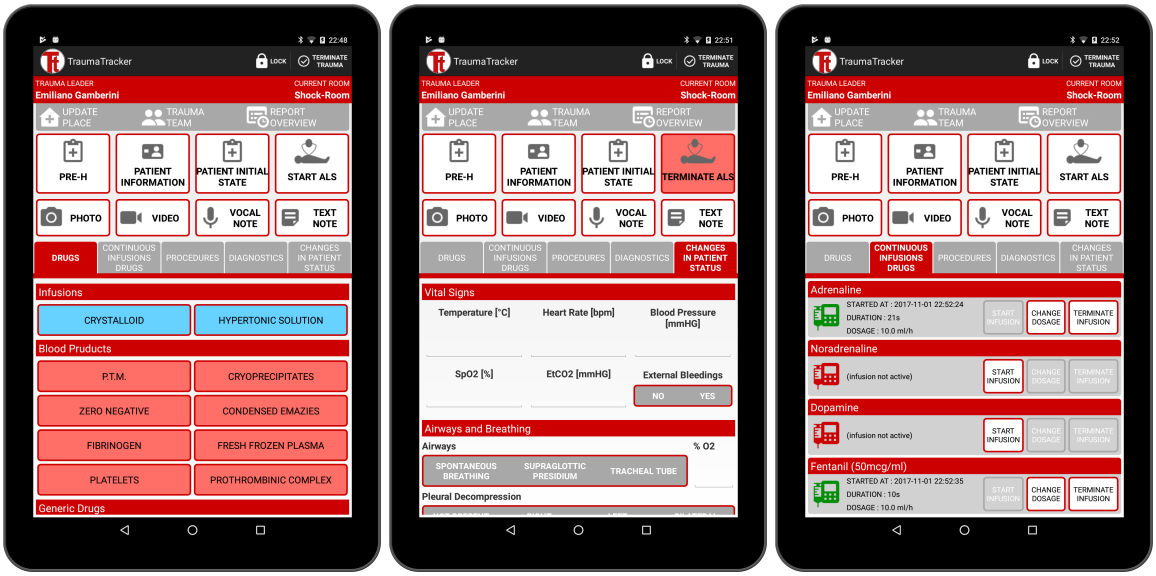


Figure 2: Trauma Assistant Agent user interface on the tablet. *(left)* The Drugs item proposes the different categories of drugs from the most commonly used and using a different colour for each category. *(center)* The UI that can be used to annotate significant variations in vital signs that can not be obtained automatically from the vital signs patient monitor, e.g. the presence/absence of external bleeding. *(right)* The Drugs Infusion item shows which drugs are continuously administered, specifying when administration started, time elapsed and dosage (rate).

Figure 3: The Vuzix m300 smart glasses used for the TraumaTracker prototype. In the circle, an example of a warning on received by the Trauma Leader: the border color categorize the warning level and the message in the middle define the content of it. In this case, the warning to request to activate the M.T.P. according to the rule 2.

and other few useful information about the ongoing trauma, without changing her focus of attention from her activities towards external sources of information – e.g. displays or vital signs monitors. Figure 3 shows an example of an alert proposed to the Trauma Leader on smart-glasses. The Trauma Assistant Agent interacts with the services provided by the GT^2 infrastructure and GT^2 pervasive.

The GT^2 infrastructure includes a set of web services, that are exploited by the Trauma Assistant Agent and by the web-apps to store and retrieve information/data and to allow third-parts/external software system integration (e.g. the electronic medical record). In particular, a TT Report Management Service provides a RESTful API for collecting and managing trauma reports and accessing to related statistical data – and a TT Vital Signs Monitor Service provides the API for dynamically retrieving the real-time vital signs parameters of a patient under trauma, communicating with vital signs patient multi-parameters monitor exploiting the HL7 standard[2]. In particular, the existing technology adopted in the hospital for vital signs monitoring [3] implements the version 2.3 of the HL7 standard, based on the MLLP (Minimal Lower Layer Protocol) as low-level protocol to exchange messages over TCP/IP sockets. Data collected by these services are made available to other hospital applications running on the same infrastructure, in an open ecosystem perspective.

The GT^2 pervasive currently includes a beacon-based infrastructure[4] for room-level localisation. Each room interested by the trauma management is equipped with a beacon gateway, connected to the hospital Intranet. The gateway continuously keeps track of the beacon tags or devices present in the room, making this information available as a service. These includes also the tablet – equipped with the BLE technology – used by the Trauma Leader.

Finally, GT^2 apps currently includes a TT Report Dashboard, a web application that allows users – e.g., a member of the Trauma Team – to access the trauma documentation i.e. the reports, to manage, print and export them, as well as to do basic statistics.

Current architecture aims at balancing the benefits of decentralisation – i.e., each Trauma Leader has her own Trauma Assistant Agent running on her device – and the availability of centralised services—running on the GT^2 infrastructure. The choice of executing the personal assistant agent directly on the Trauma Leader device has been driven both by the need of having the system working even when being in locations/rooms not covered by the Hospital WiFi network, and to maximise responsiveness – avoiding network lags.

4. The Trauma Assistant Agent

The Trauma Assistant Agent has two main tasks. The first one is to assist the Trauma Leader in documenting an ongoing trauma, keeping track of data and events occurring in the emergency rooms related to a specific patient, inferring as much as possible data from the context (e.g., the place where a procedure is performed). This includes: tracking the actions performed by the Trauma Team – procedures (e.g., endotracheal intubation, thoracic drainage, application of a tourniquet and many others), drug/blood product infusion (e.g., millilitres of crystalloids or hypertonic solution, adrenaline, atropine, pools of cryoprecipitates, etc.); taking snapshots, recording video or audio annotation to be included in the report (exploiting the camera equipped with the tablet or the smart-glasses); retrieving, displaying and tracking the current value of patient's vital signs—by interacting with the TT Vital Signs Monitor Service service. These data must be automatically retrieved and annotated *(i)* when a procedure or the administration of a particular drug are performed; *(ii)* periodically, with a period that depends of the specific location of the patient in the emergency room (i.e. the period of vital signs monitoring could be different if the patient is currently in Shock-Room rather than in the Computed

[2]The HL7 (Health Level 7) standard refers to a 7th level of the ISO-OSI stack designed to provide a common layer for the exchange, integration, sharing, and retrieval of electronic health information in health services (`www.hl7.org`).

[3]Based on the Draeger Infinite Gateway Technology

[4]Beacons are embedded devices capable to periodically broadcast small amount of data, exploiting Bluetooth Low Energy (BLE) protocol. This data can be received by enabled devices/gateways within a short range of visibility defined by the signal transmission power.

Tomography Scan room). Every event/note tracked by the agent includes both temporal (date and time) and spatial (location, specific room) information. At the end of the trauma management, the agent creates a report and send it to the TT Report Management Service.

The second main task of the Trauma Assistant Agent is about alert generation, by dynamically reasoning on the tracked data and producing alerts to be perceived by either the Trauma Leader (through smart-glasses) or by the whole Trauma team (through displays in the environment). Alert generation is driven by a set of rules defined by the Trauma Team. Table 1 shows the current list, specifying when the alert is generated, what alert message is displayed and the rationale for the rule (described in the caption). The rules can be split in two categories:

- *Time Independent Rules* – in this case the condition is about the current state of the trauma – including patient state and location (rules 2, 3, 4, 6, 7, 8, 10, 11);
- *Time Dependent Rules* – in this case the condition include also temporal checks that concern the evolution of the state of the trauma (rules 1, 5, 9).

Rules have been designed by the Trauma Team after a retrospective analysis of reports, in order to improve the performance/quality of their action, preventing dangerous situations for the patient. Some of the rules are useful and applicable in general; others depend on characteristics of the specific context in which the team operates and to the specific team composition, and could change over time. In Section 5 we will discuss more in detail the design and implementation of alert generation.

4.1. BDI Architecture

The Trauma Assistant Agent has been designed upon the BDI (Belief-Desire-Intention) agent model/architecture. The BDI architecture originated in the work of the Rational Agency project at Stanford Research Institute in the mid-1980s, inspired by the theory of human practical reasoning developed by the philosopher Michael Bratman [16]. The Procedural Reasoning System (PRS), originally developed at Stanford Research Institute by Michael Georgeff, Amy Lansky Francois Félix Ingrand [17, 46], was the first agent system to explicitly embody the BDI architecture. Almost all existing BDI agent programming technologies are directly or indirectly based on PRS [47].

In the PRS, an agent does no planning from first principles. Instead, it is equipped with a library of pre-compiled plans. These plans are manually constructed, in advance, by the agent programmer. Plans in the PRS each have the following components:

- a *goal* – the post-condition of the plan, i.e. what the plan is good for (the things that it achieves);
- a *context* – the pre-condition of the plan, i.e. what must be true of the environment in order for the plan to be successful;
- a *body* – the "recipe" part of the plan, i.e. the course of action to carry out.

At runtime, the behaviour of the agent is governed by a sense-plan-act based reasoning cycle, in which the agent continuously senses the environment updating the belief base accordingly (perceive stage), decides which goal/plans to bring about (plan stage) and select/perform actions (act stage).

The key aspect of this approach for our purposes is the possibility to design a real-time intelligent system (e.g., assisting humans in critical operations), featuring the capability of integrating a goal/task-oriented behaviour – i.e., the agent has explicit goals to achieve and for that purpose it selects and executes plans – and a reactive behaviour – i.e., while executing the plans it can promptly react to events occurring in the environment, eventually executing further plans to handle them.

4.2. Implementation based on Agent Technologies

The AgentSpeak(L) abstract language, introduced by Rao [48], captures the key features of the PRS into a simple, unified programming language with a well-defined formal semantics. The syntax and semantics of AgentSpeak(L) has been adopted then reference model to implement concrete programming

Rule	Condition	Alert Message
1	Zero Negative Blood administered at the time t *but* at the time $t + 5min$ not administered fibrinogen and tranexamic acid yet.	"Administer Fibrinogen and Tranexamic Acid" for 10 secs
2	When administered 3 unit of zero negative blood and tranexamic acid and fibrinogen	"Activate Massive Transfusion Protocol (MTP)?"
3	When Thiopental administered	"Pay attention to hypotension"
4	When ALS (Advanced Life Support) started	"Administer 1mg of Adrenaline" every 3 mins
5	When ALS started at time t but at time $t + 5min$ the pleural decompression not performed yet	"Pleural Decompression?"
6	If there is a fracture, when exiting from the Shock Room without having started the antibiotic prophylaxis	Message: "Activate Antibiotic Prophylaxis?"
7	Tourniquest/Reboa/Toracotomy procedure	Message: "Tourniquest / Reboa / Toracotomy activated for n minutes" every 15 minutes
8	When exiting from the Shock Room	"Feeding tube and bladder foley correctly arranged? "
9	If tracheal intubation performed at time t but at time $t + 5mins$ no information about EtCO2 value registered	Message: "Check EtCO2"
10	When the SpO2 value become less then 90%	"Is FiO2 set to 100%?"
11	When infuser used and Blood Pressure is greater than 110 mmHg	"Suspend infuser usage?"

Table 1: Set of rules used by the TraumaTracker system to generate alerts to be notified to the Trauma Leader and its team. A short description of the rationale for each rule follows. Rule #1: The Early Coagulation Support prescribes the administration of both fibrinogen and tranexamic acid in the case of blood transfusion during a trauma; Rule #2: The Early Coagulation Support can include the activation of the Massive Transfusion Protocol; Rule #3: The administration of Thiopental is the last therapeutic bulwark in the case of serious head trauma. Rule #4: The ALS protocol requires to administer adrenaline every 3 minutes until either the return of spontaneous circulation or the end of resuscitation. Rule #5: When dealing with a cardiac arrest, pleural decompression should be always considered. Rule #6: In the case of fracture exposition Antibiotic Prophylaxis should be performed as soon as possible. Rule #7: The tourniquest/reboa/toracotomy procedures can be affected by complications whose gravity is proportional to the duration of the procedure. Rule #8: Almost any major trauma (but exceptional cases) requires to arrange the feeding tube and the bladder foley. Rule #9: EtCO2 monitoring is mandatory in the case of tracheal intubation. Rule #10: Oxygen must be administered as soon as possible in hypoxic patients. Rule #11 – In a trauma patient, blood pressure should not be greater than 110 mmHg until excluding bleeding.

languages and platforms to program cognitive agents [47]. In this project we adopted in particular the JaCaMo platform [49], which integrates Jason [50] – which is a Java-based interpreter of an extended version of AgentSpeak(L) – with CArtAgO platform [51], to model and implement the agent application environment and MOISE organisation-oriented framework, to design MAS organisations [52]. In particular, a version of JaCaMo running on mobile and wearable devices has been exploited, based on JaCa-Android [53], which introduces an agent-oriented programming model to design, develop and run agent-based applications on top of the Android platform.

At the implementation level, the Trauma Assistant Agent is not a single monolithic agent but it includes two concrete Jason BDI agents, each one in charge of one specific task—a Tracker Agent and Alert Generator Agent. This choice makes it possible to achieve a high-level of modularity, so that future upper-level functionalities – such as more sophisticated forms of assistance – will be encapsulated and introduced in terms of new agents, eventually cooperating with these ones.

5. Modeling Alerting Rules as BDI Plans

Using BDI model of agency the rules can be naturally and effectively formalised in terms of agent plans (of the Alert Generator Agent, in this case), exploiting the beliefs of the agent to keep track of the relevant information (event, state) of the ongoing trauma. Following the AgentSpeak(L) model [48], plans are described by $+\mathsf{ev} : \mathsf{co} \rightarrow \mathsf{act}.$, where:

- +ev is the event triggering the plan. For our purposes, it can be of two kinds:
  - the belief addition about of a new observable trauma event perceived by the agent +ev(EvInfo,Time) or about temporal trigger +trig(TrigName,Time), used to manage rules with timeouts. EvInfo follows from the tracking level and can be about a new drug administration (drug(DrugType,Qty)), a new one-shot procedure (procedure(ProcName)), a time-based procedure (procedureStarted(ProcName), procedureStopped(ProcName)), the entrance or exit from a room (room_in(RoomName), room_out(RoomName)), and a new input action action(InputAction,Time) introduced by the user through the UI.
- co is the context defining when the plan (the rule) should be considered applicable, according to the beliefs that the agent has about the ongoing trauma. It can be any first-order logical expression concerning about the presence or absence of beliefs in the belief base, using partially specified literal to specify pattern matching multiple beliefs. The kind of beliefs used in the rule are: about the occurrence (or not) in the past of an event (ev(EvInfo,Time), $\nexists$ ev(EvInfo,Time)), about the presence of an alert (alert(AlertId,Time)), about the ongoing activation of a procedure (active_proc(ProcName)).
- act is a sequence of actions to be performed, when the plan is selected and executed. For our purposes, the sequence concerns a single action, which could be of two basic kinds:
  - the addition of a belief about a new alert alert(AlertId,Time) to be communicated (the syntax used for the addition is: +alert(AlertId,Time)), or the removal of an existing alert (-alert(AlertId,Time)), meaning that the alert should be no more visible.
  - set_trig(TrigName,When), whose effect is to generate a new trigger trig(TrigName,Time) at the time When. Temporal triggers are used to model rules time dependency rules.

Table 2 shows the set of plans formalising the *time-independent* rules. A time-independent rule can be directly mapped into a plan where the triggering event is either a drug administration (rule 2, 3, 10), or a procedure (rules 4, 7, 11), or a change of the room (rule 8). Table 3 shows the set of plans formalising the *time-dependent* rules (rules 1, 5, 9). Each rule can modelled by a couple of plans using a temporal trigger: the first reacting to the event triggering the rule and setting a temporal trigger, and the second plan reacting to the temporal trigger. Rule 1, the first plan is triggered when a new zero negative drug is administred (at some time T). This sets a new temporal trigger

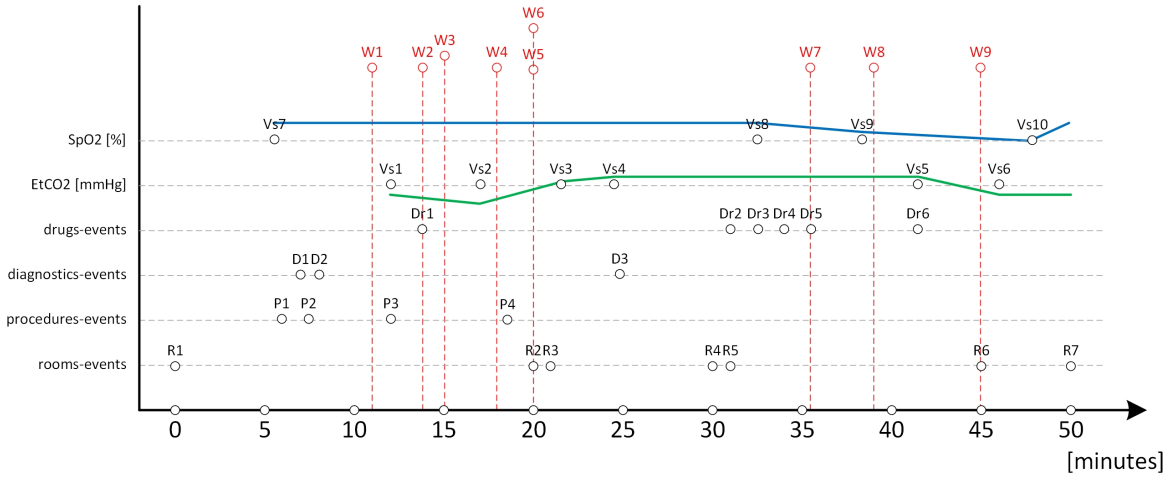
Rule	**Plan**
2	+ev(drug(zero_negative,T)) : zero_negative(V) ∧ V ≥ 3 ∧ ∃ ev(drug(tranex,_),_) ∧ ∃ ev(drug(fibri,_),_) → +alert(activateMTP,T). +ev(drug(tranex,T)) : zero_negative(V) ∧ V ≥ 3 ∧ ∃ ev(drug(fibri,_),_) → +alert(activateMTP,T). +ev(drug(fibri,T)) : zero_negative(V) ∧ V ≥ 3 ∧ ∃ ev(drug(tranex,_),_) → +alert(activateMTP,T).
3	+ev(drug(thiopental,T)) → +alert(checkHypoTh,T).
6	+ev(room_out(shock_room),T) : fracture(true) ∧ ∄ ev(drug(abt),_) → +alert(checkABT,T).
8	+ev(room_out(shock_room),T) → +alert(checkFTandFoley,T).
10	+ev(drug(spo2,V),T) : V < 90 → +alert(checkFIO2,T).
11	+ev(vs(bp_value,Bp),T) : Bp > 110 ∧ ev(procedure(infuser),_) → +alert(infusSuspension(Bp),T).

Table 2: Plans modelling time-independent rules.

Rule	**Plan**
1	+ev(drug(zero_negative,_), T) → set_trig(T+300, checkTranAndFibr). +trig(checkTranAndFibr, T) : ∄ ev(drug(tranex,_),_) ∨ ∄ ev(drug(fibri,_),_) → +alert(checkTranAndFrib,T). +ev(drug(tranex,_) : alert(checkTranAndFibr,_) ∧ ev(drug(fibri,_),_) → -alert(checkTranAndFibr,_). +ev(drug(fibri,_) : alert(checkTranAndFibr,_) ∄ ev(procedure(pleu_decomp,_),_) ∧ ev(drug(tranex,_),_) → -alert(checkTranAndFibr,_).
4	+ev(procedureStarted(als),T) → +active_proc(als); set_trig(T+300,checkAdren). +trig(checkAdren, T) : active_proc(als) ∧ ∄ ev(drug(adren_als,_),_) → +alert(adminAdren,T).
5	+ev(procedureStarted(als),T) → +active_proc(als); set_trig(T+300,checkPleuDecomp). +trig(checkPleuDecomp, T) : active_proc(als) ∧ ∄ ev(procedure(pleu_decomp,_),_) → +alert(pleuDecomp,T).
7	+ev(procedureStarted(tourniq),T) → +active_proc(tourniq); set_trig(T+900,checkTourniq). +trig(checkTourniq, T) : active_proc(tourniq) → +alert(tourniqActiveSince,T).
9	+ev(procedure(intub),T) → set_trig(T+300, checkEtCO2). +trig(checkEtCO2,T) : ∄ ev(vs(etCO2,_),_) → +alert(checkEtCO2,T).

Table 3: Plans modelling time-dependent rules.

named checkTranAndFibr after 5 minutes (300 seconds). The second plan reacts to the event about the temporal trigger, and creates a new alert if either the tranexamic acid or fibrinogen drugs have not been administered yet. If the drugs are administered, the alert is removed. Rule 5 is triggered when an ALS procedure is started, generating a trigger about checking pleural decompression after 5 minutes. A alert then is generated if at that time no pleural decompression procedures have been performed yet. Rule 9 is triggered when an intubation procedure is done, generating a trigger about checking



REPORT' EVENTS AND WARNINGS

1. (R1) Room In: Shock-Room
2. (Vs7) New value for SpO2: 100%
3. (P1) Procedure: Intubation
4. (D1) Laboratory Diagnostic: Blood Analisys
5. (P2) Procedure: Detected Fracture
6. (D2) Diagnostic: EGA
7. **(W1) Alert "Check EtCO2"**
8. (Vs1) New Value for EtCO2 = 35 mmHg
9. (P3) Procedure: ALS Start
10. (Dr1) Drug: Thiopental
11. **(W2) Alert "Pay attention to hypotension!"**
12. **(W3) Alert "Administer 1mg of adrenaline"**
13. (Vs2) New value for EtCO2: 33 mmHg
14. **(W4) Alert "Administer 1mg of adrenaline"**
15. (P4) Procedure: ALS Stop
16. (R2) Room Out: Shock-Room
17. **(W5) Alert "Activate antibiotic Prophylaxis?"**
18. **(W6) Alert "Feeding tube and bladder foley correctly arranged?"**
19. (R3) Room In: CT-Room
20. (Vs3) New Value for EtCO2 = 38 mmHg
21. (Vs4) New Value for EtCO2 = 40 mmHg
22. (D3) Diagnostic: Computed Tomography (CT)
23. (R4)Room Out: CT-Room
24. (R5) Room In: Shock Room
25. (Dr2) Drug: Zero Negative 1 UI
26. (Dr3) Drug: Fibrinogen 1g
27. (Vs8) New value for SpO2: 99%
28. (Dr4) Drug: Zero Negative 2 UI
29. (Dr5) Drug: Tranexamic Acid 1g
30. **(W7)Alert "Activate Massive Transfusion Protocol (MTP)?"**
31. (Vs9) New value for SpO2: 89%
32. **(W8) Alert "Is FiO2 setted to 100%"**
33. (Vs5) New value for EtCO2: 40 mmHg
34. (Dr6) Drug: Antibiotic Prophyllaxis
35. (R6) Room Out: Shock-Room
36. **(W9) Alert "Feeding tube and bladder foley correctly arranged?"**
37. (Vs6) New value for EtCO2: 32 mmHg
38. (Vs10) New value for SpO2: 87%
39. (R7) Final Destination: Intensive Care Unit

Figure 4: The time evolution trauma events and related alerts generated by TraumaTracker.

Alert Id	**Message displayed**
checkTranAndFibr	Administer Fibrinogen and Tranexamic Acid
activateMTP	Activate Massive Transfusion Protocol (MTP)?
checkHypoTh	Pay attention to hypotension
adminAdren	Administer 1mg of Adrenaline
pleuDecomp	Pleural Decompression?
checkABT	Activate Antibiotic Prophylaxis?
tourniquet	Tourniquest activated for n minutes
checkFTandFoley	Feeding tube and bladder foley correctly arranged?
checkEtCO2	Check EtCO2
checkFIO2	Is FiO2 set to 100%?
infusSuspension	Suspend infuser usage?

Table 4: Set of alerts identifiers and corresponding messages displayed.

the EtCO2 after 5 minutes. The same trigger is created also if the patient is known to be already intubated. A alert is created if no information about EtCO2 vital sign is available yet (represented by the belief vs(etCO2,Qty) and corresponding events ev(vs(etCO2,Qty),Time)).

Figure 4 shows an example of the time evolution trauma events and related alerts generated by TraumaTracker. Considering the management of a trauma during which the Trauma Leader tracks events (e.g. Procedures, Drugs, Vital Signs, etc.), the figure shows these events at their temporal moment of occurrence – see the legend for details about each event – correlating them with generated alerts generated. As an example, consider the alert *W1* generated – due to Rule n.9 – five minutes later the tracking of the procedure *P1*, because no information about EtCO2 have been registered yet. Moreover: (1) according to Rule n.3, the event *Dr1* forces the system to produce instantly the alert *W2*, (2) according to Rule n.4, alerts *W3* and *W4* are generated at three minutes intervals after event *P3* and until the tracking of the event *P4*, (3) the event *R2* forces the systems to use Rules n.6 e n.8 at this instant of time to produce alerts *W5* and *W6* but the same event *R6* – tracked later – brings the system to generate only the alert *W9* produced using the Rule n.8 because contextual information are changed, in particular the event *Dr6* has been tracked. And so forth.

5.1. Implementation in Jason

The implementation of the abstract plans in the Jason agent programming language is straightforward. An example of plans implementing a time-independent rule (rule 2) follows:

```
+event(drug(zero_negative,_),T) :
  zero_negative(V) & V >= 3 & event(drug(tranex,_),_) & event(drug(fibrinogen,_),_)
  <- +alert(activatePTM,T).

+event(drug(tranex,_),T) : zero_negative(V) & V >= 3 & event(drug(fibrinogen,_),_)
  <- +alert(activatePTM,T).

+event(drug(fibrinogen,_),T) : zero_negative(V) & V >= 3 & event(drug(tranex,_),_)
  <- +alert(activatePTM,T).

+alert(AlertId, T) <- ?alertMsg(AlertId, AlertMsg); displayAlert(AlertId, AlertMsg, T).
```

The first three plans encapsulate the specific logic about creating a new belief `alert(AlertId,Timestamp)` related to the `activatePTM` alert type, while the last plan – which is good also for the other rules – defines the actions to perform when a new alert is produced. The `event(EvType,TimeStamp)` beliefs about the occurrence of an event are automatically created in the agent belief base from the percepts that the agent is getting by observing an event stream resource, which is a component of the application—implemented as *artifact* [54] shared and used by the agents. The action `displayAlert(AlertId,AlertMsg,TimeStamp)` displays the message either on the smart-glasses or on the room display is performed, after retrieving from the belief base the specific textual message corresponding to the alert type (`?alertMsg(AlertId, AlertMsg)`), Table 4 shows the messages corresponding to the type of alert included in the plans in Table 2 and Table 3. In this specific case, the belief mapping warn type/message is `alertMsg(activatePTM, "Activate Massive Transfusion Protocol (MTP)?")`. The implementation of this action – as well as all the other actions performed by agents – is not inside the agent code, but encapsulated into specifically designed components (artifacts) implemented using the CArtAgO framework – included in JaCaMo – which provides a Java API to that purpose.

An example of plans implementing a time-dependent rule (rule 1) follows:

```
+event(drug(zero_negative,_),_) <- setTrigger(300,checkTranexAndFibrinogen).

+trigger(checkTranexAndFibrinogen) : not event(drug(tranex,_),_) | not event(drug(fibrinogen,_),_)
  <- ?timestamp(T); +alert(checkTranexAndFibrinogen, T).
```

```
+event(drug(tranex,_), _) : alert(checkTranexAndFibrinogen) & event(drug(fibrinogen,_),_)
  <- -alert(checkTranexAndFibrinogen, _).

+event(drug(fibrinogen,_), _) : alert(checkTranexAndFibrinogen) & event(drug(tranex,_),_)
  <- -alert(checkTranexAndFibrinogen, _).

-alert(AlertId,T)  <- hideAlert(AlertId).
```

Like for the other actions, also the `setTrig(Delay,TrigName)` action – generating a new observable event `+trigger(TrigName)` after the specified amount of time (in seconds) – is provided by a specific artifact. This example shows also the plans used to remove an alert, by removing the corresponding `alert(AlertId,Timestamp` belief. The removal of this belief triggers a plan (the last plan) performing an action to stop displaying the alert (`hideAlert(AlertId)`).

6. Evaluation and Discussion

A prototype implementation of TraumaTracker has been used at the Trauma Center for about 1 year, by a Trauma Team composed by 8 physicians. They all are Anesthesia and Intensive care Physicians, Trauma Intensive Care Unit Doctors, In-hospital Emergency Team Doctors and ED consultants, except for one who is the In-hospital Emergency System Chief and Trauma Intensive Care Unit Chief Doctor. A first evaluation and discussion of the trauma tracking and documentation feature is described in [20]. In summary, the tool proved to *(i)* improve accuracy and completeness of trauma documentation, without creating an overhead on the trauma team using it, and *(ii)* reduce the cognitive and practical burden by automating many steps of the documentation process. Besides, the availability of accurate data about trauma resuscitation events, with temporal and contextual information, was useful for performance analysis and statistics compilation, supporting decision making about organisational changes mainly devoted at reducing health-care costs and improving patient care.

In this paper we focus on the evaluation and discussion of the alert generation feature. In the remainder of the section we consider two different aspects: effectiveness and usability.

6.1. Effectiveness

To evaluate the effectiveness of the alert generation feature, we instrumented the system to produce a log about the alerts generated during a trauma, along with the report. Then, we analysed the temporal evolution of traumas as described by events in the report and the corresponding temporal evolution of alerts dynamically created by the system. Figure 4 shows an example of this analysis.

In order to demonstrate system's effectiveness in a real-time scenario, but evaluating the system in a controlled environment, we applied this approach retrospectively —as already done with success in previous work [35]. In particular we used a set of 431 reports, collected by Trauma Tracker (without alert feature) in a period of about 7 months (from Jan 23 2018 to Aug 31 2018), to feed the Trauma Assistant Agent with events included in those reports and log alerts created by the agent. An example of log produced by the Trauma Assistant Agent analysis is shown in Table 5, in which two alerts are produced: one concerns the alert `checkFTandFoley`, generated after the event related to exiting the shock room, and one is `checkTranAndFibr`, after 5 minutes that zero-negative was administered (at time 1520373405).

The result of the whole analysis is reported in Table 6. For each rule, we considered the number of *relevant* reports for the rule, i.e. reports in which a situation relevant for the rule is encountered, and the number of cases in which an alert is generated by the rule. For the first rule, there are 20 relevant reports, i.e. reports where zero negative is administered. In 7 cases an alert is generated (fibrinogen and tranexamic acid have not been administered within 5 minutes). Concerning rule #4, in 3 cases the ALS procedure started and in 2 cases an alert is generated—since the adrenaline was not administered within the first 3 minutes. In the same 3 relevant cases, in 2 cases an alert is generated by rule #5 since pleural decompression was not performed within the first 5 minutes. Concerning rule

```
REPORT #55 - rep-20180306-222314 - N events: 23
[1520862424] VitalSignEvent bp-value 70
[1520371394] RoomEnterEvent Shock-Room
[1520372290] DrugEvent midazolam 15.0
[1520372300] DrugEvent curare 10.0
[1520373109] DrugEvent crystalloid 500.0
[1520373112] DrugEvent crystalloid 500.0
[1520373405] DrugEvent zero-negative 1.0
[1520373405] ALERT TRIGGER | checkTranexAndFibrinogen | 1520373705
[1520373601] RoomExitEvent Shock-Room
[1520373601] ALERT Feeding tube and bladder foley correctly arranged?
[1520373601] RoomEnterEvent Sala Operatoria: Blocco
[1520375864] DrugEvent fibrinogen 1.0
[1520375864] ALERT Administer Fibrinogen and Tranexamic Acid
[1520375865] DrugEvent fibrinogen 1.0
[1520375870] DrugEvent emazies 1.0
[1520375873] DrugEvent crystalloid 500.0
[1520375874] DrugEvent crystalloid 500.0
[1520375878] DrugEvent midazolam 5.0
[1520376987] VitalSignEvent bp-value 120
[1520376995] VitalSignEvent etco2-value 32
[1520377711] RoomExitEvent Sala Operatoria: Blocco
[1520377711] RoomEnterEvent TAC PS
[1520381221] RoomExitEvent TAC PS
[1520381221] RoomEnterEvent Terapia Intensiva: TI-2
[1520381305] RoomExitEvent Terapia Intensiva: TI-2
```

Table 5: An example of log produced by the Trauma Assistant Agent analysis of a real report (registered in March 2018), including the timestamp (in milliseconds from January 1, 1970 UTC) and the description of the event fetched/detected (such as the administration of a drug, e.g. `DrugEvent midazolam 15.0`) or generated (such as the generation of an alert, e.g. `Feeding tube and bladder foley correctly arranged?`).

Rule no.	**No. of relevant reports**	**No. of alerts**
#1	20	7
#4	3	2
#5	3	2
#7 (Tourniquet)	3	3
#7 (Reboa)	5	1
#7 (Thoracotomy)	2	0
#8	517	517
#9	29	19
#10	7	7
#11	4	1

Table 6: Alerts generated by the Trauma Assistant Agent fed with the data of 431 existing reports (collected with Trauma Tracker).

#7, in all cases in which Tourniquet was applied (3 cases), an alert would have been produced, since the duration of the activation was greater than 15 minutes The same applies for Reboa (1 alert over 5 cases). No alerts would have been generated for Thoracotomy. Concerning rule #8, for 517 times the shock room was left and for 517 times the alert would have been generated. The explanation for this large number is that currently Trauma Tracker does not track the arrangement of feeding tube and bladder foley, so every time the trauma exists the shock room the alert is generated. As effect of rule #9, in 29 cases in which the tracheal intubation was performed, in 19 cases an alert would have been generated, since no EtCO2 value has been registered in 5 minutes. As effect of rule #10, in 7 cases the SpO2 value became less than 90% and in 1 case an alert would have been generated about the blood pressure greater than 110 mmHg.

Generally speaking, the generation of a number of alerts (greater than zero) suggests that the rules identified by the Trauma Team are useful for the team to identify situations that can actually occur in trauma management. In those cases, the tool would have been helpful to the Team, reminding an action to do (e.g., administering Tranexamic acid) which has not been performed without the tool. The limited number of alerts generated with respect to the overall number of reports suggests that there are no over-alerting problems, but in one case: the rule number 8, which generates an alert each time the trauma exists the shock room. Actually this rule can be revised in order to generate the alert only if either the feeding tube or the bladder foley have not been correctly arranged. To that purpose, the tracking feature needs to be extended to include also the possibility to track the procedure about arranging either the feeding tube or the bladder foley.

6.2. Usability

To evaluate the usability we considered the comprehensive set of evidence-based usability design principles for medication alerting systems suggested in [55]. The system design is compliant with the meta-principles identified there, in particular:

- Meta-principle A — The system's signal-to-noise ratio is maximised and over-alerting minimising, since the alert strategy takes fully into account the clinical context and clinicians speciality. The rule based approach and the BDI model approach make it straightforward the customisation of the knowledge implemented in the tool.
- Meta-principle B — The collaborative work is supported by allowing for displaying the alerts in displays that are in the environment where the Trauma Team operate.
- Meta-principle C — The alerting system fits with clinicians' workflow and their mental model, since the rules have been designed by the Trauma Team specifying both the timings and conditions for generating the alerts and the concise message data to display. No alerts are interruptive (according to the classification in [28]).
- Meta-principle D — Relevant data is displayed within the alert (when useful, according to the Trauma Team).
- Meta-principle E — The system is fully transparent for the user, since *(a)* the set of rules generating the alerts are decided (and shared) by the Trauma Team, and *(b)* the reasons why an alert is generated can be traced and reconstructed by considering the tracked data available in reports — that medics can check either dynamically, during a trauma, or after its completion.
- Meta-principle F — Actionable tools are included with the alert system, both to dismiss and track alerts, to allow for inspecting and analysing the alert history either during a trauma, or after its completion.

Finally, the Trauma Team has been involved in usability tests about the UI related to alerts on smart-glasses. Trauma Team's feedbacks lead to a categorisation of the alerts considering their severity and importance. In particular, by using different colours, when the Trauma Leader perceives an alert, she/he can implicitly decide either to consider it immediately (e.g., if it is a red alert) or to ignore/postpone it without distracting herself from the trauma resuscitation process.

7. Conclusion and Future Work

In this paper we described the design and implementation of a personal assistant agent technology – based on the BDI architecture – to the problem of trauma tracking and alert generation. The evaluation carried on by the Trauma Team using a prototype implementation of the system for several months allowed to see in practice the value of the tool for improving the quality of the documentation, the accuracy about timings in particular – which is fundamental in time dependent diseases – as well as the quality of the performance of the Team, in general. Besides, the daily use of the tool made it possible to stress current limitations of the approach, suggesting significant future works and research directions.

A first main direction is about investigating techniques to further automate tracking, reducing the need of using the mobile device by the Trauma Team and fully exploiting hands-free interfaces [56, 13, 14], enabled by wearable devices such as the smart-glasses. Speech recognition is a natural candidate for this purpose: the specific characteristics of the context, however, makes this objective extremely challenging—due to the noise of the environment and the level of responsiveness and accuracy required in recognising the speech-based commands.

Another important direction is about improving the openness and customizability of TraumaTracker so as to be easily exploited by different Trauma Centres. To this purpose, an important improvement is the adoption of reference medical ontologies – such as MeSH[5] – to encode and represent the medical knowledge. In current version, the terms adopted in describing the trauma documentation (e.g., administered drugs, procedures, patient information) are based on the vocabulary suggested by the medics of the Trauma Team, implicitly referring to standards. In future version, all the terms and concepts will be aligned to standard thesauri, including an explicit reference to them. The improvement related to customisability is about providing an explicit support to allow users (the Trauma Team) to (partially) customise the set of tracked procedures/drugs, including the GUI, without changing the application code.

A medium-term direction is about exploring further the level of assistance that can be useful to the Trauma Team. A simple one is about reminding and suggesting the workflow of steps to follow in peculiar cases that require an ad-hoc treatment. Like in the case of alert generation, the use of smart-glasses is especially useful for implementing this functionality, making it possible to provide suggestions while keeping the focus of attention on the patient. A more challenging one is about integrating cognitive systems [57] with our cognitive agent technologies. In the model discussed so far, the generation of alerts by the Trauma Assistant Agent is based solely on the knowledge about the ongoing trauma. A further step is to consider for that purpose also the *corpus* of knowledge related to trauma management and the documentation about the trauma done in the past, a big data collecting information from different hospital and trauma centers, and the use if *cognitive computing* techniques [58] to get insights from that Big Data. Cognitive computing refers to a set of tools and techniques – including Big Data and Analytics, machine learning, Internet of Things, Natural Language Processing, causal induction, probabilistic reasoning, and data visualization – which makes it possible to devise a "cognitive system" which is capable of learning, remembering, analysing, resolving problems in specific contexts—healthcare and life science are a primary one [59]. A main example is Watson by IBM [57]. An interesting open research issue is then the design of a personal assistant agent that combines the capabilities of *cognitive agents* and the support of cloud-based *cognitive services* in a Cognition-as-a-Service style [60].

Finally, we are interested to investigate and develop a general framework for developing BDI-based personal medical assistant agents – eventually integrated with cognitive services – beyond the specific case about trauma management and alerting. To this purpose, it will be important to analyse and factorise main functionalities and features useful for personal medical assistant agents in general, in spite of the specific task to be supported. Existing literature about the design of personal agents for

[5] Medical Subject Headings controlled vocabulary, available at https://www.nlm.nih.gov/mesh/

task management (e.g., [5, 6]) will be an important reference.

Acknowledgement

The research presented in this paper is developed under the Framework Agreement Rep. 83/2017 Prot. 759 between AUSL Romagna, the Department of Computer Science and Engineering (DISI) and the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi" (DEI) of the University of Bologna. The authors would like to thank all the Trauma Center staff that made this contribution possible, in particular: Emanuele Russo, Costanza Martino and Maurizio Ravaldini for their cooperation and support in the Trauma Tracker project; all the members of the Trauma Team, experimenting Trauma Tracker; Riccardo Castagnoli, Lorenzo Rossi and the staff of the "Gestione Sistemi Informatici" Hospital Unit for their technical support; Carlotta Sapignoli, Filippo Tempestini and Carmelo Sturiale, for their technical support related to Draeger subsystems.

[1] T. M. Mitchell, R. Caruana, D. Freitag, J. McDermott, D. Zabowski, Experience with a learning personal assistant, Commun. ACM 37 (7) (1994) 80–91.

[2] P. Maes, Agents that reduce work and information overload, Commun. ACM 37 (7) (1994) 30–40.

[3] S. Okamoto, P. Scerri, K. Sycara, Toward an understanding of the impact of software personal assistants on human organizations, in: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '06, ACM, New York, NY, USA, 2006, pp. 630–637.

[4] T. Wagner, J. Phelps, V. Guralnik, R. VanRiper, Coordinators: Coordination managers for first responders, in: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3, AAMAS '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 1140–1147.

[5] N. Yorke-Smith, S. Saadati, K. L. Myers, D. N. Morley, The design of a proactive personal agent for task management, International Journal on Artificial Intelligence Tools 21 (01).

[6] M. Tambe, Electric Elves: What went wrong and why, AI Magazine 29 (2) (2008) 23–27.

[7] C. Li, J. A. Giampapa, K. P. Sycara, Bilateral negotiation decisions with uncertain dynamic outside options, IEEE Trans. Systems, Man, and Cybernetics, Part C 36 (1) (2006) 31–44.

[8] D. Isern, D. Sánchez, A. Moreno, Agents applied in health care: A review, International Journal of Medical Informatics 79 (3) (2010) 145 – 166.

[9] S. Iqbal, W. Altaf, M. Aslam, W. Mahmood, M. U. G. Khan, Application of intelligent agents in health-care: review, Artificial Intelligence Review 46 (1) (2016) 83–112.

[10] H. Lieberman, C. Mason, Intelligent agent software for medicine 80 (2002) 99–109.

[11] J. Santos, J. J. Rodrigues, B. M. Silva, J. Casal, K. Saleem, V. Denisov, An iot-based mobile gateway for intelligent personal assistants on mobile health environments, Journal of Network and Computer Applications 71 (2016) 194 – 204.

[12] N. Spanoudakis, P. Moraitis, Engineering ambient intelligence systems using agent technology, IEEE Intelligent Systems 30 (3) (2015) 60–67.

[13] T. Starner, The challenges of wearable computing: Part 1, IEEE Micro 21 (4) (2001) 44–52.

[14] T. Starner, The challenges of wearable computing: Part 2, IEEE Micro 21 (4) (2001) 54–67.

[15] S. Mann, Wearable Computing as Means for Personal Empowerment, in: Proceedings of the First International Conference on Wearable Computing(ICWC), IEEE Computer Society Press, Fairfax, VA, 1998.

[16] M. E. Bratman, D. J. Israel, M. E. Pollack, Plans and resource-bounded practical reasoning, Computational Intelligence 4 (3) (1988) 349–355.

[17] M. P. Georgeff, A. L. Lansky, Reactive reasoning and planning, in: Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 2, AAAI'87, AAAI Press, 1987, pp. 677–682.

[18] A. Sarcevic, "who's scribing?": Documenting patient encounter during trauma resuscitation, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, ACM, New York, NY, USA, 2010, pp. 1899–1908.

[19] T. O'Connor, E. A. Raposo, T. Heller-Wescott, Improving trauma documentation in the emergency department, Journal of Trauma Nursing 21 (5) (2014) 238–243.

[20] S. Montagna, A. Croatti, A. Ricci, V. Agnoletti, V. Albarello, E. Gamberini, Real-time tracking and documentation in trauma management, Health Informatics Journal.To Appear. Draft available upon request.

[21] P. Whitten, B. Holtz, C. LaPlante, Telemedicine: What have we learned?, Applied Clinical Informatics 1 (2) (2010) 132–121.

[22] B. M. Silva, J. J. Rodrigues, I. de la Torre Díez, M. López-Coronado, K. Saleem, Mobile-health: A review of current state in 2015, Journal of Biomedical Informatics 56 (2015) 265–272.

[23] U. Varshney, Mobile health: Four emerging themes of research, Decision Support Systems 66 (2014) 20–35.

[24] A. Poggi, F. Bergenti, Developing smart emergency applications with multi-agent systems, Int. J. E-Health Med. Commun. 1 (4) (2010) 1–13.

[25] I.D.S. Civil, What is quality care in trauma?, Injury 38 (5) (2007) 525 – 526.

[26] M. Skubic, R. D. Guevara, M. Rantz, Automated health alerts using in-home sensor data for embedded health assessment, IEEE Journal of Translational Engineering in Health and Medicine 3 (2015) 1–11.

[27] S. Patel, H. Park, P. Bonato, L. Chan, M. Rodgers, A review of wearable sensors and systems with application in rehabilitation, Journal of NeuroEngineering and Rehabilitation 20 (2012) 9–21, journal of NeuroEngineering and Rehabilitation.

[28] M. Jung, A. Hörbst, W. Hackl, F. Kirrane, D. Borbolla, M. Jaspers, O. Marc, V. Koutkias, L. Ferret, P. Massari, K. Lawton, D. Riedmann, S. Darmoni, N. Maglaveras, C. Lovis, E. Ammenwerth, Attitude of physicians towards automatic alerting in computerized physician order entry systems. a comparative international survey. 52.

[29] D. Isern, A. Moreno, A systematic literature review of agents applied in healthcare, Journal of Medical Systems 40 (2) (2016) 43.

[30] A. B. d. l. Torre, M. Lluch-Ariet, J. Pegueroles-Vallés, Security analysis of a protocol based on multiagents systems for clinical data exchange, in: 2013 Seventh International Conference on Complex, Intelligent, and Software Intensive Systems, 2013, pp. 305–311.

[31] F. Burstein, A. Zaslavsky, N. Arora, Context-aware mobile agents for decision-making support in healthcare emergency applications, in: Workshop on Context Modeling and Decision Support, 2005.

[32] F. Marins, L. Cardoso, M. Esteves, J. Machado, A. Abelha, *An* agent-based rfid monitoring system for healthcare, in: Á. Rocha, A. M. Correia, H. Adeli, L. P. Reis, S. Costanzo (Eds.), Recent Advances in Information Systems and Technologies, Springer International Publishing, Cham, 2017, pp. 407–416.

[33] M. Taboada, E. Cabrera, M. L. Iglesias, F. Epelde, E. Luque, An agent-based decision support system for hospitals emergency departments, Procedia Computer Science 4 (2011) 1870 – 1879.

[34] C.-J. Su, C.-Y. Wu, JADE implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring, Applied Soft Computing 11 (1) (2011) 315 – 325.

[35] V. G. Koutkias, I. Chouvarda, N. Maglaveras, A multiagent system enhancing home-care health services for chronic disease management, IEEE Transactions on Information Technology in Biomedicine 9 (4) (2005) 528–537.

[36] A. Poncela, F. Coslado, B. García, M. Fernández, J. Ariza, G. Peinado, C. Demetrio, F. Sandoval, Smart care home system: a platform for eassistance, Journal of Ambient Intelligence and Humanized Computing.

[37] S. Montagna, A. Omicini, Agent-based modeling for the self-management of chronic diseases: An exploratory study, SIMULATION 93 (9) (2017) 781–793.

[38] J. Fox, M. Beveridge, D. Glasspool, Understanding intelligent agents: Analysis and synthesis, AI Commun. 16 (3) (2003) 139–152.

[39] J. M. Corchado, J. Bajo, Y. de Paz, D. I. Tapia, Intelligent environment for monitoring alzheimer patients, agent technology for health care, Decision Support Systems 44 (2) (2008) 382 – 396.

[40] Ó. García, D. I. Tapia, A. Saavedra, R. S. Alonso, I. García, Alz-mas 2.0; a distributed approach for alzheimer health care, in: J. M. Corchado, D. I. Tapia, J. Bravo (Eds.), 3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 76–85.

[41] H.-K. Kim, Convergence agent model for developing u-healthcare systems, Future Generation Computer Systems 35 (2014) 39 – 48, special Section: Integration of Cloud Computing and Body Sensor Networks; Guest Editors: Giancarlo Fortino and Mukaddim Pathan.

[42] F. Felisberto, R. Laza, F. Fdez-Riverola, A. Pereira, A distributed multiagent system architecture for body area networks applied to healthcare monitoring, BioMed Research International 2015.

[43] E. Domnori, G. Cabri, L. Leonardi, Ubimedic2: An agent-based approach in territorial emergency management, in: 2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops, 2011, pp. 176–183.

[44] A. Borowczyk, M. Gawinecki, M. Paprzycki, Bdi agents in a patient monitoring scenario, in: 2008 Second International Conference on Pervasive Computing Technologies for Healthcare, 2008, pp. 82–85.

[45] A. Croatti, S. Montagna, A. Ricci, A personal medical digital assistant agent for supporting human operators in emergency scenarios, in: S. Montagna, P. H. Abreu, S. Giroux, M. I. Schumacher (Eds.), Agents and Multi-Agent Systems for Health Care, Springer International Publishing, Cham, 2017, pp. 59–75.

[46] F. F. Ingrand, M. P. Georgeff, A. S. Rao, An architecture for real-time reasoning and system control, IEEE Expert: Intelligent Systems and Their Applications 7 (6) (1992) 34–44.

[47] R. H. Bordini, L. Braubach, M. Dastani, A. El, F. Seghrouchni, J. J. Gomez-sanz, J. Leite, A. Pokahr, A. Ricci, A survey of programming languages and platforms for multi-agent systems, Informatica 30 (1).

[48] A. S. Rao, Agentspeak (l): Bdi agents speak out in a logical computable language, in: Agents Breaking Away, Springer, 1996, pp. 42–55.

[49] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with jacamo, Science of Computer Programming 78 (6) (2013) 747–761.

[50] R. H. Bordini, J. F. Hübner, Bdi agent programming in agentspeak using jason, in: F. Toni, P. Torroni (Eds.), Computational Logic in Multi-Agent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 143–164.

[51] A. Ricci, M. Piunti, M. Viroli, A. Omicini, Environment programming in CArtAgO, in: Multi-Agent Programming: Languages, Platforms and Applications, Vol. 2, Springer, 2009, pp. 259–288.

[52] J. F. Hubner, J. S. Sichman, O. Boissier, Developing organised multiagent systems using the moise+ model: Programming issues at the system and agent levels, Int. J. Agent-Oriented Softw. Eng. 1 (3/4) (2007) 370–395.

[53] A. Santi, M. Guidi, A. Ricci, Jaca-android: An agent-based platform for building smart mobile applications, in: M. Dastani, A. El Fallah Seghrouchni, J. Hübner, J. Leite (Eds.), Languages, Methodologies, and Development Tools for Multi-Agent Systems: Third International Workshop, LADS 2010, Lyon, France, August 30 – September 1, 2010, Revised Selected Papers, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 95–114.

[54] A. Omicini, A. Ricci, M. Viroli, Artifacts in the A&A meta-model for multi-agent systems, Autonomous Agents and Multi-Agent Systems 17 (3) (2008) 432–456.

[55] R. Marcilly, E. Ammenwerth, E. Roehrer, J. Niès, M.-C. Beuscart-Zéphir, Evidence-based usability design principles for medication alerting systems, BMC Medical Informatics and Decision Making 18 (69).

[56] T. Starner, Project glass: An extension of the self, IEEE Pervasive Computing 12 (2) (2013) 14–16.

[57] J. E. Kelly, Computing, cognition and the future of knowing, iBM Research and Solutions, white paper (2015).

[58] J. Hurwitz, M. Kaufman, A. Bowles, Cognitive Computing and Big Data Analytics, Wiley, 2015.

[59] Y. Chen, J. E. Argentinis, G. Weber, IBM Watson: How cognitive computing can be applied to big data challenges in life sciences research, Clinical Therapeutics 38 (4) (2016) 688 – 701.

[60] J. Spohrer, G. Banavar, Cognition as a service: An industry perspective, AI Magazine 36 (4).