Supporting Multiple Agents in the Execution of Clinical Guidelines

Alessio Bottrighi, Luca Piovesan and Paolo Terenziani
DISIT, Universitá del Piemonte Orientale, Viale Teresa Michel 11, Alessandria, Italy

Keywords: Clinical Guidelines, Physician Interaction and Communication, Human Resources Coordination.

Abstract: Clinical guidelines (GLs) exploit evidence-based medicine to enhance the quality of patient care, and to optimize it. To achieve such goals, in many GLs different agents have to interact and cooperate in an effective way. In many cases (e.g. in chronic disorders) the GLs recommend that the treatment is not performed/completed in the hospital, but is continued in different contexts (e.g. at home, or in the general practitioner's ambulatory), under the responsibility of different agents. Delegation of responsibility between agents is also important, as well as the possibility, for a responsible, to select the executor of an action (e.g., a physician main retain the responsibility of an action, but delegate to a nurse its execution). To manage such phenomena, proper support to agent interaction and communication must be provided, providing them with facilities for (1) treatment continuity (2) contextualization, (3) responsibility assignment and delegation (4) check of agent "appropriateness". In this paper we extend GLARE, a computerized GL management system, to support such needs. We illustrate our approach by means of a practical case study.

1 INTRODUCTION

Clinical guidelines (GLs) are defined as "*systematically developed statements to assist practitioner and patient decisions about appropriate healthcare under specific clinical circumstances*" (Field and Lohr, 1990). They are conceived as a way of putting Evidence-Based Medicine (EBM) into practice, as well as a mean to grant both the quality and the standardization of healthcare services, and the minimization of costs. Thousands of GLs have been devised in the last years. For instance, the Guideline International Network (http://www.g-i-n.net) groups 103 organisations representing 47 countries from all continents, and provides a library of more than 6400 CPGs. Since the 90's, the medical community has started to recognize that a computer-based management can further increase GL advantages, providing relevant benefits (e.g. decision making support) to care providers and patients.

Many different systems and projects have been developed to this purpose (see e.g. (Fridsma, 2001; Gordon and Christensen, 1995; Peleg, 2013)). Such systems usually provide facilities to acquire, represent and/or execute GLs, and are mainly developed to support physicians in patient care.

Different forms of support may be provided. In particular, a lot of attention has been devoted to decision support facilities (such as "what if" analysis (Terenziani et al., 2002) or cost-benefit analysis (Montani and Terenziani, 2006)). Notably, computer-based approaches do not aim at substituting physicians: although physicians may take into account the suggestions provided by the systems, the final decision is always left to physicians themselves. Specifically, physicians retain the full responsibility of taking decisions, and of identifying the proper actions for the patients. However, computer-based GL systems have quite neglected the problem of properly supporting the coordination of different healthcare agents in the execution of GLs (see, however, the discussion in Section 6). Indeed, while some GLs are specifically related to an execution context (e.g., they have to be totally executed in an hospital), others, mainly dealing with chronic disorders, require that patient treatment is continued in time, and is carried on in *different contexts* (e.g. at home, or in the general practitioner's ambulatory), under the responsibility of *different agents* (not only physicians). In such cases, the *correct interaction and communication* between the involved agents is critical for the quality of care, involving the ability of identifying a proper *responsible* for the next actions, and the possibility of *delegating* the responsibility and\or the *execution* of actions to other agents. None of the available

Bottrighi, A., Piovesan, L. and Terenziani, P.
Supporting Multiple Agents in the Execution of Clinical Guidelines.
DOI: 10.5220/0006654802080219
In *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018) - Volume 5: HEALTHINF*, pages 208-219
ISBN: 978-989-758-281-3

computerized GL systems fully addresses such needs.

In this work, we propose an extension of a computerized GL management tool to deal with these needs. First, we identify the extensions to the GL formalism needed to represent the different pieces of information required to manage the above phenomena. In particular, our representation formalism supports the specification, for each action, of the *context* in which it can be execute, and of the *qualification* and *capabilities* required to its responsible, and to its executor.

Then, we describe the facilities we have provided to support the coordination of multi-agents executing a GL. The main goal of such facilities is to support a proper treatment of the patient, in such a way that the GL actions are executed in the proper context, under the responsibility of a proper (i.e., having the correct qualification and capabilities) agent, and are executed by a proper agent. To achieve such a goal, our facilities support:

(1) treatment continuity,
(2) action contextualization,
(3) responsibility assignment and delegation
(4) check of agent and executor "appropriateness".

Notably, there are several multi-agent approaches for healthcare in the literature (see e.g. the survey in (Isern and Moreno, 2016)), but they consider agents as autonomous software entities. In our approach, we consider agents as a representation of a real entity and use a *multi-agents view* to describe and support a distributed GL execution.

A practical implementation of this work is represented by an extension of META-GLARE. META-GLARE (Bottrighi and Terenziani, 2016) is a recent extension of GLARE , a domain-independent system for GL acquisition and execution (Terenziani et al., 2008).

Resorting to the META-GLARE formalism, we will illustrate the application of our approach to the "management of harmful drinking and alcohol dependence in primary care" GL developed by the Scottish Intercollegiate Guidelines Network (SIGN) (Scottish Intercollegiate Guidelines Network, n.d.), which we have adapted to the Italian context. However, although we have implemented our approach in META-GLARE**,** it is worth stressing that the methodology we propose is general and application-independent.

The paper is structured as follows: in section 2 we describe the main features of GLARE and META-GLARE. In section 3 we describe our extensions to META-GLARE representation formalism. In section 4, which is the core of the paper, we describe the different facilities we provide to support the distributed execution of a GL, and the coordination of the involved agents. In section 5 we exemplify a practical application of our approach considering the treatment of alcohol-related disorders. Finally, in the section 6 we address related works and concluding remarks.

2 GLARE AND META-GLARE

META-GLARE is an evolution of GLARE, a domain-independent system for acquisition and execution of GLs (Terenziani et al., 2008), which we are developing since 1997, in collaboration with the physicians of Azienda Ospedaliera San Giovanni Battista in Torino, Italy.

The core of GLARE (see box on the left of Figure 1) is based on a modular architecture. CG_KRM (Clinical Guidelines Knowledge Representation Manager) is the main module of the system: it manages the internal representation of GL, and operates as a domain-independent and task-independent knowledge server for the other modules; moreover it permanently stores the acquired GL in a dedicated Clinical Guidelines Database (CG-DB). The Clinical Guidelines Acquisition Manager (CG_AM) provides expert-physicians with a user-friendly graphical interface to introduce the GL into the CG_KRM and to describe them. It may interact with four databases: the Pharmacological DB, storing a structured list of drugs and their costs; the Resources DB, listing the resources that are available in a given hospital; the ICD DB, containing an international coding system of diseases; the Clinical DB, providing a "standard" terminology to be used when building a new GL, and storing the descriptions and the set of possible values of clinical findings.

The execution module (CG-EM) executes a GL for a specific patient, considering the patient's data (retrieved from the Patient DB). The schema of the Patient DB mirrors the schema of the Clinical DB. Therefore, the interaction with the Clinical DB during the acquisition phase makes it possible to automatically retrieve data from the Patient DB at execution time. CG-EM stores the execution status in another DB (CG Instances) and interacts with the user-physician via a graphical interface (CG-IM).

GLARE's architecture is open: new modules and functionalities can be easily added if\when necessary. In the latest years, several new modules and\or methodologies have been added to cope with automatic resource-based contextualization (ADAPT module, (Terenziani et al., 2004)), temporal reasoning (TR, (Anselma et al., 2006)), decision making support (DECIDE_HELP, (Montani et al., 2005)), model-based verification (VERIFY, (Bottrighi et al., 2010)), and comorbidities (COMORBID, (Piovesan et al., 2014)).

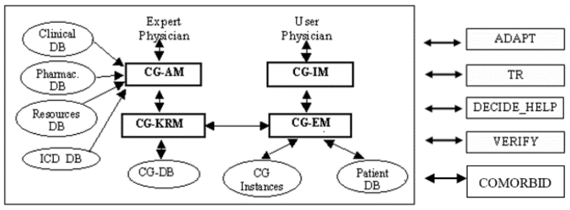


Figure 1: Architecture of GLARE. Rectangles represent computation modules, and ovals data/knowledge bases.

Representation Formalism. In the GLARE project, a GL is represented through the set of actions composing it. GLARE distinguishes between *atomic* and *composite* actions. Atomic actions can be regarded as elementary steps in a GL, in the sense that they do not need a further decomposition into sub-actions to be executed. Composite actions are composed by other actions (atomic or composite). Four different types of atomic actions can be distinguished in GLARE: *work* actions, *query* actions, *decisions* and *conclusions*. Work actions are basic atomic actions which must be executed on the patient, and can be described in terms of a set of attributes, such as name, (textual) description, cost, time, resources, goals. Query actions are requests of information, which can be obtained from the outside world (physicians, Databases, knowledge bases). Decision actions are specific types of actions embodying the criteria which can be used to select a alternative paths in a GL. In particular, diagnostic decisions are represented as an open set of triples <diagnosis, parameter, score> (where, in turn, a parameter is a triple <data, attribute, value>), plus a threshold to be compared with the different diagnoses' scores. On the other hand, therapeutic decisions are based on a pre-defined set of parameters: *effectiveness*, *cost*, *side-effects*, *compliance*, *duration*. Finally, conclusions represent the output of a decision process. Composite actions are defined in terms of their components, via the "has-part" relation. Control relations establish which actions might be executed next and in what order. We distinguish among four different control relations: *sequence*, *constrained*, *alternative* and *repetition*. The description of sequences usually involves the definition of the minimum and maximum delay between actions. Complex temporal constraints between actions (e.g., *overlaps*, *during*) can be specified using constrained control relations. In particular, action *parallelism* can also be supported through this feature.

Acquisition. GLARE's acquisition module (CG-AM in Figure 1) provides expert-physicians with a user-friendly and easy-to-use tool for acquiring a GL. In order to achieve these goals, GLARE provides: (i) a graphical interface, which supports primitives for drawing the control information within the GL, and ad hoc windows to acquire the internal properties of the objects; (ii) facilities for browsing the GL; (iii) an "intelligent" help and consistency checking including name and range checking, logical design criteria checks, and semantics checks concerning the consistency of temporal constraints in the GL.

Execution. A dedicated module has been developed to support the execution of a GL on a specific patient, adopting the "agenda technique" (see (Anselma et al., 2006)). Basically, though the "agenda technique", GLARE is able to identify all the next actions to be executed in the current GL, and a window of time during which such actions have to be executed (according to the GL temporal constraints).

Testing. GLARE has been already tested considering different domains, including bladder cancer, reflux esophagitis, heart failure, and ischemic stroke. The acquisition of a GL using GLARE is reasonably fast (e.g., the acquisition of the GL on heart failure required 3 days).

META-GLARE. In the last years, a new GL system, META-GLARE, has been designed, on top of GLARE. Indeed, META-GLARE is a "meta" system, in that it takes in input a GL representation formalism, and automatically generates a GL system to acquire and execute GL expressed in the input formalism. To test it, an extended formalism has been used (see (Bottrighi and Terenziani, 2016)). In the following, we refer to such an extended formalism as "META-GLARE formalism". In particular, META-GLARE formalism extends GLARE's one with the possibility of specifying not only 1:1 arcs (i.e., arcs with just one input action and one output action), but also 1:n, n:1 and n:n arcs. Such an additional feature is very useful to easily model the parallelism between actions.

GLARE and META-GLARE have been developed in Java, to take advantage of its portability.

As a consequence, GLARE can run similarly on any hardware/operating-system platform.

3 GL ANNOTATIONS

In order to support the coordination of different healthcare agents during the execution of a GL on a specific patient, the description of the GL actions must be extended to consider several additional aspects. For each one of such aspects, we augment the representation formalism with an additional attribute, modelling it. In the following we refer to such attributes as (action) *annotations*. First of all, the possible *contexts* in which an action can be executed must be specified.

- **Context** annotation**:** it specifies where the action can be executed (e.g. in-patient care, community medicine). Observe that a context is not necessarily a physical place, but it is an operative environment. For instance, community medicine can refer to the patient's home or to the general practitioner's ambulatory. A set of contexts for an action can be specified, meaning that the action can be executed in any one of the contexts specified in the list;

In many practical cases, it is important to distinguish between the *responsible* of an action (or of a whole part of a GL; for instance the head physician of an hospital department), and the *executor* of the action (for instance, a physician or a nurse of that department). Indeed, to be "appropriate" a responsible (executor) must have a proper qualification, and, in some cases, some specific additional competence (other than the ones typically held by all the agents having the specified qualification). To cope with such issues, we further add six annotations:

- **responsible_qualification**: it specifies who can be responsible of the action (e.g. neurologist, gastroenterologist, …). A list of qualifications can be specified, meaning that the responsible must have (at least) one of the qualifications in the list;
- **responsible_competence:** it specifies that the must have specific abilities (e.g. expert in the alcohol-related disorders management). Such an attribute is optional. A list of competences can be specified, meaning that the responsible must have *all* the competences in the list;
- **delegate_qualification**: it specifies who can be delegated to manage the action (e.g. physician, nurse); A list of qualifications can be specified, meaning that the delegate must have one of the qualifications in the list;
- **delegate_competence** (optional): it specifies that the action can be managed only by agents with some specific abilities (e.g. alcohol-related disorders management). A list of competences can be specified, meaning that the delegate must have *all* the competences in the list;
- **executor_qualification**: it specifies who can execute the action (e.g. physician, nurse); A list of qualifications can be specified, meaning that the executor must have one of the qualifications in the list;
- **executor_competence** (optional): it specifies that the action can be executed only by agents with some specific abilities (e.g. alcohol-related disorders management). A list of competences can be specified, meaning that the executor must have *all* the competences in the list;

When a GL is being acquired, we impose that each action in it is annotated with a specification of a list of possible contexts and of a list of possible qualifications of responsibles and executors. This is mandatory, and the acquisition module is extended to support the acquisition of such annotations (see Section 4). On the contrary competence annotations are optional. In case the competence list is empty, no specific restriction needs to be applied; otherwise, only the agents having the required competences are allowed to be responsible for or to execute the action at hand.

Example. The action "Brief intervention for hazardous and harmful drinking" (see action 11 in Figure 3) in the alcohol-related disorders treatment GL (Scottish Intercollegiate Guidelines Network, n.d.) is described as follows:

- resposible_qualification: physician;
- resposible_competence: \
- delegate_qualification: physician;
- delegate_competence: \
- executor_ qualification: physician, nurse;
- executor_competence: \
- context: community medicine, SERT medicine (SERT is the acronym for "SERvizio per le Tossicodiopendenze", an Italian service similar to the Mental Health Service in U.S.A.), in-patient care, hospital ambulatory care.

In addition, as an independent data structure, we support the annotation of a whole GL with a set of **continuity constraints**. Such constraints are used to model the fact that, in many practical cases, it is preferable to assign “homogeneous” sets of actions in a GL to the same responsible (or, in some cases, executor). For instance, it might be preferable that the same neurologist is responsible of all the neurological activities performed on a given patient, and that the different EMG examinations of a patient are executed by the same specialist. Notably, continuity constraints are interpreted as “preferential” constraints by our system (see Section 4), but admits violations (e.g., after a period, a physician may, for any reason, not be able to continue to treat a given patient).

In the next sections, we will present how *annotations* are formalized in our approach, and how they are treated by META-GLARE.

3.1 Basic Ontology

GL *annotations* can be modeled on the basis of three taxonomies, and of the relations between them. Part of the taxonomies and relations are graphically shown in Figure 2. The ontology of *contexts* is a “part-of” taxonomy, in which each context can be further specified by its components. For instance, in Figure 2, FAMILY, HOSPITAL and COMMUNITY MEDICINE are three possible contexts, and NEUROLOGY, GASTROENTOROLOGY and INTERNAL MEDICINE are some of the departments that are part of hospitals. *Qualifications* can be modeled through a standard “isa” taxonomy, in which each qualification can be further refined (*isa* relation) by its specializations. For instance, in Figure 2, Nurse and Physician are two possible qualifications. In turn, NEUROLOGIST, GASTROENTEROLOGIST and INTERNIST are specializations of PHYSICIAN. Analogously, also *competences* are modeled by an “isa” taxonomy. For instance, in Figure 2, (the competence in) PERIPHERAL NEUROPATHY is a specialization of (competence in) NEUROLOGY, which is a specialization of MEDICAL COMPETENCE.

Besides concepts, which denote classes of entities, the ontology also includes instances, denoting specific entities. Each instance is connected to its class through an “instance-of” relation. For example, in Figure 2, Neuro2 is an instance of Neurology in Azienda Ospedaliera San Giovanni Battista (which is an instance of Hospital); Mario Rossi is an instance of Neurologist (thus, given the transitivity of the *isa* relation, Mario Rossi is also an instance of Physician). It is worth noticing that, while the entities in the context and qualification taxonomies have instances, competences do not have them (they are individual concepts). Besides *part-of*, *isa* and *instance-of* relations, other relations are useful to represent our domain. *Agents* (which are instances of Qualifications) are related to the contexts they belong by the *belong-to* relation. Additionally, agents may have competences, and this fact is represented by the *has-competence* relation. For instance, in Figure 2, Mario Rossi belongs to Neuro2, and has specific competence about Peripheral Neuropathy. Contexts and persons have *contacts* (usually phone numbers).

On the other hand, ***continuity constraints*** are simply formalized, for each GL, by an independent data structure, modeling

(i) the sets of actions which should (preferably) have the same responsible;
(ii) the sets of actions which should (preferably) have the same delegate;
(iii) the sets of actions which should (preferably) have the same executor.

The definition of continuity constraints is a refinement process. First the user can define the continuity constraints concerning the responsibles. Then, within each responsible-level continuity group, she can further specify continuity groups for possible delegates. Finally, continuity execution groups can be defined, within the delegate-level continuity groups.

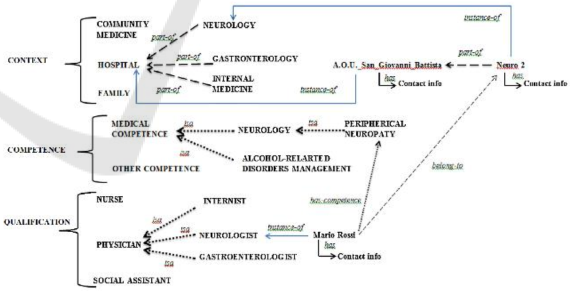


Figure 2: Ontology of contexts, qualifications and competences and their instances.

4 SOFTWARE TOOLS

4.1 Navigation Tool

We have developed a navigation tool to facilitate the navigation through the above ontology. Such a tool provides two types of facilities:

(1) schema browsing,
(2) instance browsing.

The *schema browsing* facility allows users to navigate the ontology (using the *part-of* and *isa* relations) and to find qualifications/contexts/ competences.

The *instance browsing* facility allows users to find a specific agent on the basis of the relations *part-of*, *isa*, *instance-of*, *has-competence*, *belong-to*. For example, it is possible to find an agent on the basis of a qualification (e.g. Physician), a context (e.g. Neurology) and, possibly, a competence (e.g. Peripheral Neuropathy). This facility can give in output (i) one or more agents (and their contact information) satisfying the requirements, or (ii) one or more specific contexts, in which agents having the required qualification and competences operate.

4.2 Acquisition

We have extended GLARE with an ***annotation support***, supporting the acquisition of GL annotations. We have developed a user friendly Graphical User Interface (GUI). To achieve such a goal, we enrich the acquisition GUI to provide users with schema browsing facilities (section 4.1).

Moreover, we have developed an ad-hoc module to support the definition and the acquisition of ***continuity constraints***. The user can use this module to browse the GL and to specify the set of actions in the GL which belong to a continuity constraints by selecting such actions in the graphical representation of the GL.

4.3 Execution Engine

We now extend the execution engine of META-GLARE to support the distributed execution of GLs. Specifically, we provide facilities to support the identification of the responsible(s), of delegate(s) and of the executor(s) for the next action(s) according to the GL annotations.

Already in its original version (Terenziani et al., 2014), META-GLARE execution engine was adopting an agenda, containing a set of pairs $\{(A_1,T_{A1}), \ldots,(A_k,T_{Ak})\}$ representing the actions to be executed next $(A_1,\ldots,A_k)$, and the window of time within which the actions have to be executed $(T_{A1}, \ldots,T_{Ak})$. Notably, more than one pair may appear in the set, to support concurrent execution.

To support the management of responsibilities\ delegations\executions, we add, for each action A in the agenda, a new data structure $Stack_A$, called ***agent stack*** (of A), of the form $Stack_A$: $<(X_1, role_1), \ldots,(X_k, role_j)>$ where X_i is a specific agent, and $role_h$ her role in the management of the action A (i.e., responsible (R), delegate (D), or executor (E))[1].

The execution of a GL starts with an initialization phase. All the initial actions are inserted into the agenda, together with the window of time in which they must be executed. Here and in the following algorithms, we adopt the approach in [9] to determine the window of time in which each action has to be executed, on the basis of the temporal constraints in the GL. For each one of such actions, and for each actions belonging to the Responsibility Continuity Group ("RCG" in the following algorithm) of such actions, the agent stack is initialized. In such a way the continuity of resposibles is granted.

Notably the responsibles of the first actions are predetermined and provided as input to the execution engine.

The GL execution engines operates as described by Algorithm 2. For each action A in the agenda, the GL execution engine starts its execution by sending the *execute* message (line 2) to the agent on the top of the agent stack $Stack_A$, asking her to manage the action A in the time window T_A according to her role. In case action A is executed (line 3), A is removed from the Agenda (line 4). Thus, the execution engine evaluates the set S of the next actions in the GL to be executed, using the *get_next* function (line 5). Notably, identifying the next actions which have to be executed during the execution of a GL is a standard operation (see (Isern and Moreno, 2008)). For the META-GLARE approach see (Terenziani et al., 2014).

Each action B belonging to S is pushed onto the Agenda. Then, in the case that B has not a responsible (i.e., $Stack_B$ has not been created yet), the execution engine asks to the responsible of A (i.e. the agent stored at the bottom of $Stack_A$) to search a responsible for B and the other actions in its responsibility continuity group (i.e., RCG(B); line 9) through the *next_responsible?* message. Notably, since we manage continuity groups, the responsible of an action B in the GL can be already determined before the time when B is inserted in the agenda (due to the fact that B belongs to the responsibility continuity group of another action A already inserted

[1] At the time of the execution of an action A, its agent stack $Stack_A$ should contain the responsible (bottom of the stack), a certain number of delegates (zero delegates in case no delegation has been performed; more than one delegate are possible, to support delegation of delegations), and one executor (which might be also be the last delegate, or the responsible).

in the agenda). Finally, the stack of A is deleted (line 10).

```
1. Let {(A1,$M_{A1}$), …, (Ak,$M_{Ak}$)} be the set of the
   starting actions of GL, and of their
   responsibles.
2. put the starting actions (and their time) in
   Agenda
3. for each (A, $T_A$) in Agenda do
       for each B ∈ RCG(A) do
               initialize(Stack$_B$, $M_A$, R)
```

Algorithm 1: Pseudocode of the initialization of the **GL executor engine.**

```
1.  for each (A, $T_A$) in Agenda do
2.    OUT←send (top(Stack$_A$),execute(A,$T_A$))
3     if (OUT == OK) then
4.        Remove A from Agenda
5.        S ← get_next(A)
6.        for each B in S do
7.            put in Agenda B
8.            if B has no responsible then
9.                send(next_responsible?
                    (bottom(Stack$_A$), RCG(B))
10.       Delete Stack$_A$
11.   else
12.       pop(stack of A)
13.       goto 2
```

Algorithm 2: Pseudocode of **GL executor engine.**

Otherwise, in case A is not executed (i.e. the agent on the top of StackA rejects its role), a pop on Stack$_A$ is performed (line 12). Thus, A remains in Agenda and the engine executor has to handle it again sending an execute message to the new top of the stack of A.

Notably, we support the fact that an agent accepts the responsibility, the delegation or the execution of a set of actions (all the actions in a continuity group) but, later on, stops to operate on some of the accepted actions. In such a case, the GL execution engine "goes up" in the agent stack of the "rejected" actions to find new delegates or responsibles. Notably, though the current responsible may decide not to operate any more on the actions he previously accepted, before "retiring" she has to find a new responsible for them

4.4 Support to Agents

As described above, we consider three different categories of agents in GL execution: *responsibles*, *delegates*, and *executors*. Each of them has different rights and duties, and for each of them we provide different supports.

A first set of facilities has the goal of supporting agents to find proper responsibles, delegates and executors of one or more GL actions.

The ***find_responsible*** function allows an agent to use the *instance browsing* facility to find a set of agents that satisfy the requirements expressed in the GL annotations. The agent selects one of them as the responsible for the action A (if A is in a continuity group, all the actions in the continuity group are considered). Notably, finding a responsible for a continuity group of actions does not only involve the selection of an appropriate (i.e., satisfying the annotations of the actions) agent in the ontology, but also to interact with her to know whether this agent accepts or not (by sending the ***accept_responsibility?****({(A_1,T_{A1})…,(A_k,T_{AK})})* message. In the case the agent gives a positive response the *agent stacks* of A_1, …, A_K are created, specifying the new responsible, otherwise the research for a responsible goes on.

The ***find_delegate*** and ***find_executor*** functions operate similarly, supporting the identification of appropriate delegates (if desired) and executors (compulsory) to actions (through the use of ***accept_delegation?*** and ***accept_execution?*** messages) and taking into account continuity groups.

In our approach, each agent has the possibility to receive and send different types of messages, depending on her current role (responsible, delegate, executor) in the execution of the GL.

Responsible.
Receipt of an execute(A,T_A) message. When the responsible of an action A receives an *execute(A,T_A) message*, it means that it has previously accepted the responsibility of such an action. However, it may be the case that, for any reason, at the time when A must be executed, the responsible wants\needs to decline (e.g., the responsible of a patient with a chronic disease may retire, or move away). We allow her to do so, but with a restriction: the current responsible is in charge of finding a new responsible for the action A and the other actions (not executed yet) in the responsibility continuity group of A (using the *find_responsible* function). On the other hand, if the responsible retains her responsibility, she still has several options: she can

(i) **delegate** DCG(A) (i.e., A and all the other actions in the Delegate Continuity Group of A), through the find_delegate function
(ii) **find an executor** for ECG(A) (i.e., A and all the other actions in the Executor Continuity Group of A, through the *find_executor* function,
(iii) directly **execute** A herself

Receipt of a next_responisble?({(A1,T$_{A1}$),…, (Ak,T$_{Ak}$)} message.

The current responsible is in charge of identifying an appropriate responsible for the actions $A_1 \ldots A_k$. To support her in this task, we provide the *find_responsible* function, described above.

Receipt of an accept_responsibility?({(A1,T$_{A1}$),…, (Ak,T$_{Ak}$)} message.

The agent may accept or reject the new responsibility.

Notably, soon after the acceptance of the responsibility of a set of actions {(A1,T_{A1}),…, (Ak,T_{Ak})} (a Responsibility Continuity Group of actions), the new responsible can soon search for delegates or executors for such actions (considering their Delegate and Executor Continuity Groups respectively), using the *find_delegate* and *find_executor* facilities. In such a way, the mechanism of determining delegates and executors can proceed in a (partially) asynchronous way with respect to the actual execution of actions in the GL.

Delegate.

When a delegate receives an ***execute(A,T$_A$)*** *message*, she may decline. Such a situation is directly managed by the execution engine (see Algorithm 2), which pops the delegate from Stack$_A$ and send the *execute(A,T$_A$)* message to the new top of the stack. On the other hand, if the delegate retains her role, she can **delegate** DCG(A), **find an executor** for ECG(A) or directly **execute** A herself. Additionally, she may accept or reject an ***accept_delegation?({(A1,T$_{A1}$),…, (Ak,T$_{Ak}$)}*** *request.*

Notably, as in the case of responsibles, soon after the acceptance of the delegation of a set of actions {(A1,T_{A1}),…, (Ak,T_{Ak})} the new delegate can soon look for delegates or executors for such actions.

Executor.

When an executor receives an ***execute(A,T$_A$)*** *message*, she may decline. Such a situation is directly managed by the execution engine, as described above (concerning delegates). Otherwise, she must execute action A within the time interval T_A. Additionally, she may accept or reject an ***accept_execution?({(A1,T$_{A1}$),…, (Ak,T$_{Ak}$)}*** *request.*

5 EXAMPLE

In this section, we present an application of our approach to a GL for alcohol-related problems (Scottish Intercollegiate Guidelines Network, n.d.), adapted to the Italian context (see Figure 3).

The GL starts with a request of some clinical data (query action 1), used in the following decision action (decision action 2), which is meant to diagnose whether the patient is currently experiencing a crisis state. The management of an alcohol-related crisis is outside the GL scope. If, on the other hand, the patient is not experiencing a crisis, her history is collected (query action 3), to distinguish whether it is the first time that the patient is in treatment for alcohol-related problems, or not (decision action 4). New patients require the collection of biological markers, blood alcohol concentration and anamnestic data (data request 5 and work action 6), while this data collection is not needed for patients who were already cared for alcohol related disorders (data request 12 and work action 13). For latter patients, an evaluation of biological markers and blood alcohol concentration (decision action 14) are required, to decide whether monitoring them or proceed with a detoxification. Focusing on new patients, a diagnosis about the presence of alcohol-related problems is performed (decision action 7), on the basis of the collected information. Ifthe patient does not show alcohol-related problems, the GL execution is ended. Otherwise, two different treatments can be applied, depending on the severity of alcohol-related problems; both start with a screening test (work actions 8 and 9 respectively). Focusing on patients who show a mild alcohol-dependence (work action 9), after evaluating the screening test results (decision action 10), the patient can be selected for a brief intervention for hazardous and harmful drinking (composite action 11), which basically consists in a set of motivational interviews.

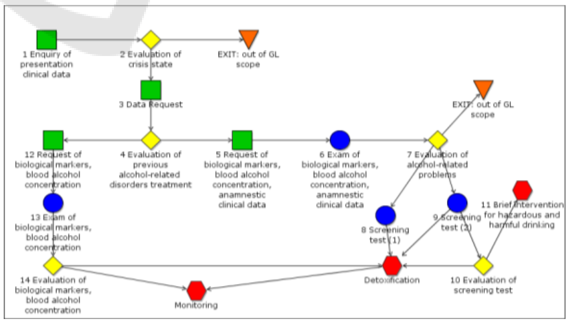
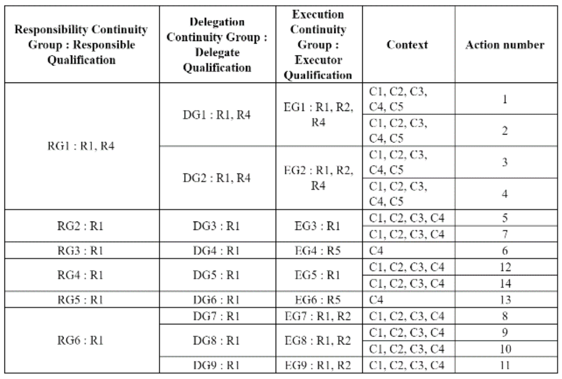

1 Enquiry of presentation clinical data
2 Evaluation of crisis state
EXIT: out of GL scope
3 Data Request
EXIT: out of GL scope
12 Request of biological markers, blood alcohol concentration
4 Evaluation of previous alcohol-related disorders treatment
5 Request of biological markers, blood alcohol concentration, anamnestic clinical data
6 Exam of biological markers, blood alcohol concentration, anamnestic clinical data
7 Evaluation of alcohol-related problems
11 Brief intervention for hazardous and harmful drinking
13 Exam of biological markers, blood alcohol concentration
8 Screening test (1)
9 Screening test (2)
14 Evaluation of biological markers, blood alcohol concentration
Monitoring
Detoxification
10 Evaluation of screening test

Figure 3: META-GLARE graphical representation of part of the GL on the treatment of alcohol related disorders.

Exploiting the annotation support (see Section 4.2), we have *annotated* all the actions defining the possible qualification(s) of its responsible, delegate (if any) and executor. Moreover, we have identified and specified the continuity groups in the GL. In this specific application, possible values for the attributes



Responsibility Continuity Group : Responsible Qualification	Delegation Continuity Group : Delegate Qualification	Execution Continuity Group : Executor Qualification	Context	Action number
RG1 : R1, R4	DG1 : R1, R4	EG1 : R1, R2, R4	C1, C2, C3, C4, C5	1
			C1, C2, C3, C4, C5	2
	DG2 : R1, R4	EG2 : R1, R2, R4	C1, C2, C3, C4, C5	3
			C1, C2, C3, C4, C5	4
RG2 : R1	DG3 : R1	EG3 : R1	C1, C2, C3, C4	5
			C1, C2, C3, C4	7
RG3 : R1	DG4 : R1	EG4 : R5	C4	6
RG4 : R1	DG5 : R1	EG5 : R1	C1, C2, C3, C4	12
			C1, C2, C3, C4	14
RG5 : R1	DG6 : R1	EG6 : R5	C4	13
RG6 : R1	DG7 : R1	EG7 : R1, R2	C1, C2, C3, C4	8
	DG8 : R1	EG8 : R1, R2	C1, C2, C3, C4	9
			C1, C2, C3, C4	10
	DG9 : R1	EG9 : R1, R2	C1, C2, C3, C4	11

Figure 4: The annotations of actions in Figure 3.

in the annotations are the following:

- context: Community medicine (C1), SERT medicine (C2), in-patient care (C3), hospital ambulatory care (C4), social services (C5);
- qualification: physician (R1), nurse (R2), healthcare assistant (R3), social assistant (R4), laboratory technician (R5).

The treatment continuity criteria demands that all the actions corresponding to the initial evaluation of the patient status (actions 1-4) must have a unique responsible (responsibility continuity group RG1; see Figure 4), which must be a physician (R1) or a social assistant (R4). The continuity group RG1 is further divided into two "subparts", corresponding to the two delegation continuity groups DG1 and DG2. In particular, DG1 corresponds to the identification of a crisis currently in progress (actions 1 and 2) and DG2 corresponds to the identification of previous alcohol-related disorder treatments (actions 3 and 4). Moreover, due to the execution continuity constraints, action 1 and action 2 belong to a single execution continuity group (EG1) and the actions 3 and 4 to a single execution continuity group (EG2). The executor of the actions in EG1 must be a physician (R1), or a social assistant (R4), or a nurse (R4). The actions in EG2 have the same constraints and the same annotations. After action 4, there are two alternative treatment paths. One path manages patients who are treated for alcohol correlated problems for the first time. Such a path is annotated with a responsibility continuity group RG2 on the actions to evaluate the patient's problem (action 5 and 7) and with a responsibility group RG3 on the exams needed for such an evaluation (action 6). RG2 and RG3 require a physician (R1) as responsible. Notably, a continuity group can contain non-contiguous actions (e.g., RG2 is composed by action 5 and 7 which are not contiguous in the GL).

In the following, we exemplify how META-GLARE extended execution engine can work on the above part of the GL. For the sake of simplicity, we omit the management of temporal constraints.

STEP 0: at the beginning of the execution, the META-GLARE executor engine identifies action 1 as the first action of the GL, and puts it in the agenda. We suppose that agent X, who is a social assistant in the social services SS1, is the responsible of action 1. Since action 1 belongs to RG1, X is also the responsible of all the actions belonging to such a continuity group (i.e., actions 1, 2, 3 and 4). Thus, the agent stacks of the four actions are created and initialized with X as responsible. In the initial step, the stacks for actions 1-4 are therefore initialized as follows (line 3 of Algorithm 1):
$stack_1$: <(X,R)>; $stack_2$: <(X,R)>; $stack_3$: <(X,R)>; $stack_4$: <(X,R)>.

The agenda of the execution engine contains only action 1 (and its temporal window, not considered in the example).
Agenda: <1>.

STEP 1: the executor engine sends an *execute* message for each action in the agenda (line 2 on Algorithm 2). Action 1 is the only action in the agenda, therefore the executor engine sends a message to the top element of the $stack_1$ (i.e., to X) to perform the execution of action 1. X receives the *execute* message and she decides to be the executor of action 1. Thus, X is put in the stack of action 1 as executor, and since actions 1 and 2 belong to the execution continuity group EG1, she is pushed as executor also onto the stack of action 2. At this point, the status of agent stacks and the agenda is the following:
$stack_1$: <(X,R),(X,E)>; $stack_2$: <(X,R),(X,E)>; $stack_3$: <(X,R)>; $stack_4$: <(X,R)>.
Agenda: <1>.

X executes action 1 (returning "OK", line 2). Since action 1 has been executed, the executor engine removes it from the agenda (line 4). Then (line 5), the next action of the GL is found (i.e. action 2) and it is put in the agenda (line 7). Action 2 has already a responsible, thus the stack of action 1 is simply deleted.
$stack_2$: <(X,R),(X,E)>; $stack_3$: <(X,R)>; $stack_4$: <(X,R)>.
Agenda: <2>.

STEP 2: the above procedure is similarly repeated for action 2 in the Agenda. We suppose that, after receiving the message, X, who is registered as executor of action 2, executes it, deciding that patient is not experiencing a crisis. Thus, action 3 is

identified as next action and put in the agenda. Also in this case, action 3 has its responsible already defined (i.e. X).
$stack_3$: <(X,R)>; $stack_4$: <(X,R)>.
Agenda: <3>.

STEP 3: Action 3 is the only action in the Agenda and is managed sending an *execute* message to its responsible X (i.e. X is on the top of $stack_3$). However, in this case we suppose that X decides to ***delegate*** such an action. Exploiting the instance browsing facility of our navigation tool (see Section 4.1), X searches for an agent satisfying the requirements (i.e. a social assistant or a physician in her context). Through the navigation tool, X is provided with a list of possible agents. She selects a preferred one from the list and ask for acceptance, until she receives a positive reply. We suppose that (possibly after some negative replies of social assistants) the social assistant Y accepts. Since actions 3 and 4 belong to the same delegation continuity group (i.e. DG2), Y is also delegated for action 4.
$stack_3$: <(X,R),(Y,D)>; $stack_4$: <(X,R),(Y,D)>.
Agenda: <3>.

Y decides to be the executor of action 3. Since actions 3 and 4 belong to the same execution continuity group EG2, Y is nominated also as the executor of action 4 and she is put in the two stacks as executor.
$stack_3$: <(X,R),(Y,D),(Y,E)>;
$stack_4$: <(X,R),(Y,D),(Y,E)>.
Agenda: <3>.

Y executes action 3 and action 4 is put in the agenda as next action.
$stack_4$: <(X,R),(Y,D),(Y,E)>.
Agenda: <3>.

STEP 4: the engine takes action 4 from the agenda, then it notifies to Y (i.e. Y is on the top of stack4) that action 4 has to be executed. Exploiting the instance browsing facility Y identifies the agent W as executor of action 4. W satisfies the action annotations (i.e. she is a nurse and operate is SS1). W accepts the assignments and she is put on $stack_4$ as executor.
$stack_4$: <(X,R),(Y,D),(W,E)>.
Agenda: <4>.

W executes action 4 and identifies that the patient is in treatment for alcohol-related problems for the first time (i.e. action 5 is the next action). Thus, action 5 is put in the agenda. Since action 5 has not yet a responsible (line 8), the system asks to X, the responsible of action 4 (i.e., the element at the bottom of the $stack_4$) to find a responsible for the next action. X must find a responsible who is a physician (R1) and works either in a Community medicine (C1) or in a SERT medicine (C2) or in-patient care (C3) or in a hospital ambulatory care (C4). Exploiting the instance browsing facilities, X finds a physician Z, who works in the community medicine CM2, and asks her for the responsibility of action 5. Z accepts the responsibility and, since action 7 belongs to the same responsibility continuity group (RG2), Z is nominated as responsible of both the actions in RG2.
$Stack_5$: <(Z,R)>; $Stack_7$: <(Z,R)>.
Agenda: <5>.

Then, the GL execution goes on in a similar way.

6 COMPARISONS AND CONCLUSIONS

In this paper, we describe the first computerized approach to GLs supporting many different crucial issues for the distributed and coordinated execution of GLs by multiple healthcare agents. Our approachgrants for the continuity of the treatment of patients (i.e., the fact that, in any moment during the GL execution, there is always a responsible for each one of the next actions to be executed on the patient) through a support to the identification of the responsibles, executors and contexts of execution of the next actions. The extensions to the GL formalism (Section 3) and to the GL execution engine and the facilities in Section 4 fully achieve such a challenging goal. Indeed, they support action contextualization and (through the definition of Continuity Groups) treatment continuity. They provide support in the identification of responsibles, delegates and executors of actions having the required qualification, and the overall approach grants that, whenever an action has to be executed, there is always a current responsible for it, and possibly delegates and executors (notably, if an executor has not been already identified, the current responsible is urged to do so). Notably, also the temporal window in which actions must be executed (given the temporal constraints in the GL) is taken into account. Last, but not least, *delegation* is supported, to enable the current responsible to take advantage of the help of other healthcare agents.

Notably, we have described our approach on the basis of METAGLARE, but it is worth stressing that our methodology is completely general and system\ application-independent (i.e. other GL system can be extended applying our approach).

While in the literature there is no other approach to computerized GL that has provided a support considering all the aspects above, several approaches have faced at least few of them.

Fox's group proposed an extension of the PROforma representation formalism (Sutton and Fox, 2003) to specify who will execute an action. However their goal is not the one of managing agents interactions in different contexts: they exploit agent information for better contextualizing GLs taking into account local human resources, and for flexibly adjusting them through delegation.

(Leonardi et al., 2007) propose a workflow-based solution to manage chronic patients over long time periods. In particular, the approach is meant to allow patients to obtain the necessary health care services by accessing different locations/ organizations, which can properly exchange/ communicate health data when needed. Their goal (i.e. support cooperative work between different healthcare organizations) is quite similar to ours; moreover, the authors model organizational knowledge (i.e. qualifications, resources etc.) by means of ontologies – as we do. However, their approach is not as flexible as ours, because interactions between agents, and allocations of the next action to a specific responsible, are strictly predetermined by a contract and can not be determined dynamically during the GL execution. On the other hand, we allow the responsible of the current action to navigate the ontology, and to dynamically and freely identify the responsible and\or the executor of the next action on the basis of the available knowledge and constraints. Moreover, they do not support delegation.

(Sánchez et al., 2011) propose an ontology-driven execution of GLs. Their approach relies on a *multi-agent* system, where every entity (i.e. actor or structure) in a medical centre is represent by one specific agent and every GL action is characterized by *hasResposible* relation with one agent or a set of agents. Their main contribution regards the delegation issue in a supervised fashion and the automation of the coordination internal activities using a medical-organisational ontology. Since their approach is meant to be applied within a specific medical centre, it is focused on supporting interaction in a distributed environment, where the coordination between actors can not be managed automatically.

Grando et al (2010) formalize cooperative work in GL execution (but not distributed executions across different contexts). The main issue they deal with is delegation of tasks to specific members of the working team, on the basis of their competences, paying particular attention to responsibilities for enacting a service, and for handling exceptions. Specifically, they extend the design pattern framework introducing the types *role* (qualification in our approach) and *actor,* and a set of relations between key concepts. Since actors have roles and competences, they recall our notion of *agent.* Therefore, they rely on concepts which are similar to our *annotation* information, even if we resort to a different mean for formalizing them (i.e. an ontology). However, Grando et al. do not consider contexts: in this sense, their approach is more limited than ours, and not straightforwardly extendable to deal also with *distributed* (and not just *cooperative*) GL executions. Moreover, we provide a set of software tools to manage the ontology in our formalization.

(Wilk et al., 2015) propose a framework to support GL execution, in which interdisciplinary healthcare teams are involved. They define three classes of agents (i.e. team manager, practitioner assistant, patient representative), but they classes do not correspond to *qualification*. They have the concept of *capability* that is similar to *competence.* They annotate actions, but their annotations are only related to the capability requirements. They have only the concept of executor of an action and not of responsible. Moreover, they have the concept of team, i.e. a set of agents (defined using a hybrid approach) who managed the execution and are coordinated by the team manager, i.e. the response-ble of execution for the whole GL. The identification of executor of an action is not general as ours: only the team manager can identify the executors and first she has to consider the agents in the team. Only in the case than there is not any suitable and available agents in the team, she can search an external agent to execute the action and can add it to the team. Notably, this is a clear limitation, since many GLs can not have a single responsible. Considering also the absence of *context*, their approach is not adapt to deal with *distributed* executions.

(Bottrighi et al., 2013) is the approach most closely related to the one we present in this paper. In particular, in such an approach, actions are annotated (*coloured*, in the terminology in (Bottrighi et al., 2013)) with context, and qualification and competences for the responsible, and different forms of support are proposed to acquire and query annotations, and to execute coloured GLs. Notably, the approach we propose deeply extends the one in (Bottrighi et al., 2013) to consider three significant aspects, neglected in (Bottrighi et al., 2013):

(1) we support the distinction between the responsibles of actions, and their executors; we support the delegation of responsibility
(2) we represent and manage continuity constraints
(3) we manage the execution of GLs expressed in META-GLARE formalism (while (Bottrighi et al., 2013) considered only GLARE formalism), thus also supporting n:1, 1:n, and n:m arcs (i.e., concurrency in the GL execution).

ACKNOWLEDGEMENTS

The authors are very grateful to Prof. Gianpaolo Molino and Dr. Mauro Torchio of Azienda Ospedaliera San Giovanni Battista in Turin (one of the largest hospitals in Italy) for their constant support, and for their help in the definition of the case study.

REFERENCES

Anselma, L., Terenziani, P., Montani, S., Bottrighi, A., 2006. Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines. *Artif. Intell. Med., Temporal Representation and Reasoning in Medicine 38, 171–195.*

Bottrighi, A., Giordano, L., Molino, G., Montani, S., Terenziani, P., Torchio, M., 2010. Adopting model checking techniques for clinical guidelines verification. *Artif. Intell. Med. 48, 1–19.*

Bottrighi, A., Molino, G., Montani, S., Terenziani, P., Torchio, M., 2013. Supporting a distributed execution of clinical guidelines. *Comput. Methods Programs Biomed. 112, 200–210.*

Bottrighi, A., Terenziani, P., 2016. META-GLARE: A meta-system for defining your own computer interpretable guideline system—Architecture and acquisition. *Artif. Intell. Med. 72, 22–41.*

Field, M. J., Lohr, K. N. (Eds.), 1990. Clinical Practice Guidelines: Directions for a New Program. *National Academies Press (US), Washington (DC).*

Fridsma, D. B., 2001. Special Issue on Workflow Management and Clinical Guidelines. *J. Am. Med. Inform. Assoc. 22, 1–80.*

Gordon, C., Christensen, J. P. (Eds.), 1995. Health Telematics for Clinical Guidelines and Protocols. *IOS Press, Amsterdam.*

Grando, A., Peleg, M., Glasspool, D., 2010. Goal-based design pattern for delegation of work in health care teams. *Stud. Health Technol. Inform. 160, 299–303.*

Isern, D., Moreno, A., 2016. A Systematic Literature Review of Agents Applied in Healthcare. *J. Med. Syst. 40.*

Isern, D., Moreno, A., 2008. Computer-based execution of clinical guidelines: *A review. Int. J. Med. Inf. 77, 787–808.*

Leonardi, G., Panzarasa, S., Quaglini, S., Stefanelli, M., van der Aalst, W.M.P., 2007. Interacting agents through a web-based health serviceflow management system. *J. Biomed. Inform. 40, 486–499.*

Montani, S., Terenziani, P., 2006. Exploiting decision theory concepts within clinical guideline systems: Toward a general approach. *Int J Intell Syst 21, 585–599.*

Montani, S., Terenziani, P., Bottrighi, A., 2005. Exploiting decision theory for supporting therapy selection in computerized clinical guidelines. *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma. 3581 LNAI, 136–140.*

Peleg, M., 2013. Computer-interpretable clinical guidelines: A methodological review. *J. Biomed. Inform. 46, 744–763.*

Piovesan, L., Molino, G., Terenziani, P., 2014. Supporting Physicians in the Detection of the Interactions between Treatments of Co-Morbid Patients, in: *Healthcare Informatics and Analytics: Emerging Issues and Trends. IGI Global, pp. 165–193.*

Sánchez, D., Isern, D., Rodríguez-Rozas, Á., Moreno, A., 2011. Agent-based platform to support the execution of parallel tasks. *Expert Syst. Appl. 38,* 6644–6656. y.

Scottish Intercollegiate Guidelines Network, n.d. Management of harmful drinking and alcohol dependence in primary care [WWW Document]. URL http://www.sign.ac.uk/guidelines/fulltext/74/index.html (last accessed 05.10.17).

Sutton, D.R., Fox, J., 2003. The Syntax and Semantics of the PROforma Guideline Modeling Language. *J. Am. Med. Inform. Assoc. JAMIA 10, 433–443.*

Terenziani, P., Bottrighi, A., Rubrichi, S., 2014. META-GLARE: a meta-system for defining your own CIG system: Architecture and Acquisition, in: *KR4HC. pp. 92–107.*

Terenziani, P., Montani, S., Bottrighi, A., Molino, G., Torchio, M., 2008. Applying artificial intelligence to clinical guidelines: the GLARE approach. *Stud. Health Technol. Inform. 139, 273–282.*

Terenziani, P., Montani, S., Bottrighi, A., Torchio, M., Molino, G., 2002. Supporting physicians in taking decisions in clinical guidelines: the GLARE“ what if” facility. *Proc. AMIA Symp. 772.*

Terenziani, P., Montani, S., Bottrighi, A., Torchio, M., Molino, G., Correndo, G., 2004. A context-adaptable approach to clinical guidelines. *Stud. Health Technol. Inform. 107, 169–173.*

Wilk, S., Astaraky, D., Michalowski, W., Amyot, D., Li, R., Kuziemsky, C., Andreev, P., 2015. MET4: Supporting Workflow Execution for Interdisciplinary Healthcare Teams, in: *Fournier, F., Mendling, J. (Eds.), Business Process Management Workshops. Springer International Publishing, Cham, pp. 40–52.*