20th European Conference on Artificial Intelligence ECAI 2012

Thomas Roth-Berghofer, David B. Leake, Jörg Cassens (Eds.)

Explanation-aware Computing ExaCt 2012

ECAI 2012 Workshop, Montpellier, France
28 July 2012
Proceedings of the Seventh International ExaCt workshop

Preface

Explanation-awareness in computing system development aims at making systems able to interact more effectively or naturally with their users, or better able to understand and exploit knowledge about their own processing. Explanation-awareness in software engineering looks at new ways to guide software designers and engineers to build purposeful explanation-aware software systems. When the word "awareness" is used in conjunction with the word "explanation" it implies some consciousness about explanation.

Outside of artificial intelligence, disciplines such as cognitive science, linguistics, philosophy of science, psychology, and education have investigated explanation as well. They consider varying aspects, making it clear that there are many different views of the nature of explanation and facets of explanation to explore. Two relevant examples of these are open learner models in education, and dialogue management and planning in natural language generation.

This volume contains the papers presented at the ECAI 2012 workshop on Explanation-aware Computing (ExaCt 2012) held on July 28, 2012 in Montpellier, France. The main goal of the workshop was to bring researchers, scientists from both industry and academia, and representatives from diverse communities and fields as Informatics, Philosophy, and Sociology, together to study, understand, and explore the aspects of explanation in IT-applications. The papers presented in this volume illustrate some of the variety of perspectives on explanation and recent progress.

There were seven submissions to ExaCt 2012. Each submission was reviewed by three program committee members. The committee decided to accept six papers for oral presentation. This volume was produced using the EasyChair system.[1]

ExaCt 2012[2] continued a series of workshops begun with a AAAI Fall symposium in 2005 in Washington followed by yearly workshops at such conferences as AAAI, ECAI, and IJCAI. The workshop series aims to draw on multiple perspectives on explanation, to examine how explanation can be applied to further the development of robust and dependable systems and to illuminate system processes to increase user acceptance and feeling of control.

Readers who would like to participate in further discussions on this topic or like to receive further information about future workshops might consider joining the Yahoo!-group explanation-research[3]. More information on explanation research is available on a dedicated website[4].

July 2012
London, UK

Thomas Roth-Berghofer
David B. Leake
Jörg Cassens

[1] http://www.easychair.org
[2] http://exact2012.workshop.hm. See also the link to related workshops there.
[3] http://tech.groups.yahoo.com/group/explanation-research/
[4] http://on-explanation.net

Workshop Organisation

Chairs

Thomas Roth-Berghofer, University of West London, UK
David B. Leake, Indiana University, USA
Jörg Cassens, University of Lübeck, Germany

Programme Committee

Agnar Aamodt	Department of Computer and Information Science, Norwegian University of Science and Technology
David W. Aha	Navy Center for Applied Research in AI, Washington DC, USA
Martin Atzmüller	University of Kassel, Germany
Ivan Bratko	University of Ljubljana, Slovenia
Patrick Brézillon	University of Paris 6, France
Ashok Goel	Georgia Institute of Technology, USA
Pierre Grenon	EBI, UK
Anders Kofod-Petersen	SINTEF ICT, Norway
Hector Munoz-Avila	Lehigh University, USA
Miltos Petridis	CMS, Greenwich University, UK
Enric Plaza	IIIA-CSIC, Catalonia (Spain)
Christophe Roche	Equipe Condillac - Universit de Savoie, France
Olga C. Santos	Spanish National University for Distance Education, Spain
Gheorghe Tecuci	George Mason University, Fairfax, VA, USA
Doug Walton	University of Windsor, Canada

Table of Contents

Explaining the Reactions of a Smart Environment

Sebastian Bader[1]

Abstract. A system's ability to explain its decision making is crucial for the acceptance and the user's trust in it. This is in particular the case while using smart environments in which the assistance functionality is distributed over multiple devices and emerges from the interplay of them. In this paper, we show how to derive explanations automatically. For this a description of the system's current state and history is transformed into discourse representation structures and finally into natural language explanations. The generation procedure has been implemented and deployed into a smart meeting room and a first user study indicates that the generated explanations are understandable and natural.

1 INTRODUCTION

The design and control of smart environments is a challenging task and an open research problem. Such environments are best characterised as heterogeneous dynamic ensembles – dynamic groups of co-located devices of different types. Smart and proactive environments should support their users by providing automatic assistance functionality. This can be done by inferring the current user intentions from sensory input and by computing a suitable sequence of actions to support the resulting user goals.

To gain user confidence in an automatic assistance system, the system should be able to provide explanations for the actions taken and for the current state of the environment. In this paper, we show how to generate explanations of a control system automatically. We focus on the generation of human readable explanations in natural language which are meant to be shown to the user on request. But while doing so, we generate a formal explanation in the form of an explanation graph and discourse representation structures, which are finally used to generate the natural language sentences. The area of explanation aware computing has recently gained attention as indicated, for example, by the successful series of Explanation-aware computing (ExaCt) workshops (e.g., [18]) held over the last five years.

Below, we show how to generate natural language explanations automatically. The types of desired explanations have been identified in a first user study. Our study validated and extended the results obtained in [14] within the application area of smart meeting rooms. We also conducted a second study to evaluate a first prototype that generates explanations automatically. This evaluation showed that the approach itself is feasible and the generated explanations sound natural, are understandable to humans, and contain enough details to explain the reactions of the laboratory and its assistance system.

While building our system, we developed a pragmatic approach for natural language generation, which should also be applicable in other application domains. It is based on the construction of a graph-based representation of causes and reactions. We then use subgraph filters to identify interesting fragments of the graph and generate discourse representation structures (DRS) from the fragments. Those DRSs are finally combined and translated into statements of a natural language generation framework to obtain the explanations in English. We show how to apply this general idea in one concrete setting and thus show its feasibility as well as the possibility of generating explanations for automatic assistance systems for smart environments.

In this paper, we discuss the results of a user study identifying important questions to be answered automatically within a smart environment. We show how to generate natural language sentences to explain the reactions of an automatic assistant system for a smart meeting room. Based on a graph-based formal description of the current state of the environment, DRSs are constructed which can easily be transformed into natural language using a natural language surface realiser. Finally, we present first evaluation results showing that the system generates natural and understandable explanations.

2 PRELIMINARIES AND RELATED WORK

As mentioned above, *smart environments* can be characterised as heterogeneous dynamic ensembles. Those are dynamically changing ensembles of sensors, actuators and software modules. All members of such ensembles might be of different types, manufactured by different vendors, or implemented by different developers. Nonetheless, the ensemble in total is supposed to provide automatic assistance functionality to its user. For example, the automatic control of devices and the optimisation of the user's environment can be expected. Several approaches have been proposed in the literature, e.g. [4, 12]. As pointer to further approaches we also refer to [3, 12, 15, 16].

Our Laboratory is equipped with various actuators and sensors, like dimmable lamps, projectors and projection surfaces, cameras, microphones, localisation systems and others. Different meeting room scenarios serve as use-cases for our experiments and the lab itself as hardware platform. While building context aware assistance systems, we follow the paradigm of *goal-based interaction*, as outlined in [10] and [20]. Based on the known state of the world and sensory inputs, the system tries to infer the current goals of the user, that is the user's intentions. Taking the available devices and their capabilities into account, the system computes a sequence of device actions to support the user. We use the terms *intention recognition* and *strategy synthesis* to refer to the two stages of this process. The interface between them are sets of goals to be achieved.

The traditional way to build control systems is to decompose the problem into sensing, planning and execution. The input data, generated by sensors, is used to update the corresponding values of an abstract world model. On the basis of the world model, a planning algorithm computes the action sequence needed to achieve a specified goal. Brooks' 'Subsumption Architecture' is an alternative to this 'sense–plan–act' approach [2]. This approach has been success-

[1] University of Rostock, email: sebastian.bader@uni-rostock.de

fully applied in various areas of robotics. Brooks' ideas and his control system inspired other researchers. The goalaviour-based system [1] described below implements such a behaviour-based approach to control a smart environment in a distributed manner.

Research on *intelligibility* is concerned with the automatic generation of explanations. An explanation is a sequence of statements describing the current state or behaviour of a system and which clarifies the underlying causes, the context, and the consequences. It includes a system's ability to explain its own reactions, its internal state and the implemented functionality. As pointed out by Dey in [5], it is one of the big challenges for the near future of ubiquitous computing. The required technology to build intelligent environments is available (in principle), but approaches to explain the reactions and behaviour are still under research [13]. In particular, while considering dynamic and heterogeneous device ensembles, it is yet completely unclear how to design appropriate user interfaces and how to construct human readable explanations automatically. Nonetheless, providing explanations is necessary to gain the trust of the users into the assistance provided by the system.

Automatic assistance systems need to make their decision based on the perceived context. But, because neither the context, nor the internal rules of the systems are accessible by the user, the system itself appears as a black box. To provide insights into the systems internals, explanations are needed to describe why certain actions are taken, or why certain actions are not taken. In principle, two types of explanations can be distinguished: those meant for automatic analysis and those meant for humans. The first type is needed if the explanations of one system need to be forwarded to a second one. Here we concentrate on explanations which are generated in natural language and are meant for human users. In Section 3.2 we discuss a set of questions and show how to answer those questions automatically.

Natural language generation (NLG) is concerned with the generation of natural language from a formal representation. Several approaches have been proposed in the past. A fairly up-to-date overview can be found on the website of the Special Interest Group on Natural Language Generation (SIGGEN). NLG systems are usually constructed in layers [17]: *Layer 1 (Document planner)* determines the content and the structure of the document to convey the communicative goal. *Layer 2 (Micro-planner)* performs the lexicalisation, referencing and aggregation of paragraphs and sentences. *Layer 3 (Surface realiser)* applies grammar rules to convert an abstract representation into text. In our system described below, the document- and part of the micro-planning is done by the explanation generation process. As surface realiser, we use the SimpleNLG framework [8], which is also able to perform some micro-planning (in particular the aggregation), provides a lexicon with a fair coverage of English words and performs the morphological realisation.

Discourse representation theory (DRT) provides an abstract and formal representation of meaning in natural language and allows to handle references within and across sentences [11]. It employs so-called *discourse representation structures (DRS)* to represent the meaning of sentences. Without going into details, we present a slightly adapted example taken from [11]: The meaning of the sentences "*Jones owns Ulysses. It fascinates him.*" is represented within the following DRS: `[X,Y] : [jones(X), ulysses(Y), owns(X,Y), fascinates(Y,X)]`. A DRS consists of two parts, a list of referents and a list of conditions. Those data structures can be used in natural language understanding and interpretation as well as for the generation of natural language. For example the *Attempto Controlled English (ACE)* project uses similar structures to interpret and construct natural language sentences [7].

3 EXPLAINING A RULE-BASED CONTROL SYSTEM

After introducing our *goalaviour-based control system*, we discuss how to explain its reactions. The explanation generation described below has been implemented in SWI-Prolog[2] and the output is generated in HTML to be easily presentable to the user. The results are discussed in Section 3.7. After describing a rule- and goal-based controller, we discuss necessary explanation types and show how to answer certain questions automatically.[3]

3.1 Goal- and Rule-based Control

The paradigm of *goalaviour*-based control was introduced in [1]. As in Brooks original idea, the control system is implemented within small independent behaviours. But instead of controlling the actuators of the environment directly, those behaviours produce goals[4] – describing the desired (partial) state of the world. All goal-emitting behaviours create their output solely based on the current state of the world, and are independent of each other. The goals are merged and a sequence of device actions is computed that leads to the desired state of the world.

Example 1 *One goalaviour could emit the goal to turn on a lamp whenever a user is close to it. Another goalaviour could emit the goal to turn off all lamps to save energy. In combination, both goalaviours result in some kind of intelligent light control in which the user's position is illuminated and other lights are turned off to save energy.*

Here, we consider crisp context information only. That is, the state of every entity in the world (physical device, user or software component) can be described as a mapping from properties to values. To simplify the notation we use a set-based representation containing *entity–property–value triples (EPVs)*. The *state of the world* is described as a consistent set of such triples. A set s is called *consistent* if and only if for every entity–property pair (e, p) there is at most one entry $(e, p, x) \in s$. We use $\mathcal{W}$ to denote the set of all consistent world states. To express a certain goal to be achieved by the controller, partial world states are used as customary in the automatic planning community [9, 19]. Thus, a desired state of the world is described by the corresponding set of entity–property–value triples. We use $\mathcal{G}$ to refer to the set of all possible goals.

Example 2 *Let l be a lamp, which can be switched on and off and also be dimmed, with a* dim*-value of* 1 *indicating full brightness. The formalisation of the goal to switch on the lamp and turn it to full brightness is:*[5] $\{(l, \mathsf{on}, \top), (l, \mathsf{dim}, 1)\}$.

Every goal represents a partial state of the world which has to be achieved. A *goalaviour* g maps the current state of the world to a set of goals: $g : \mathcal{W} \to \mathcal{P}(\mathcal{G})$. In contrast to the original idea, we restrict goalaviours below to be precondition–goal rules. In particular, we consider the precondition to be a set of entity–property–value triples and the rules to be expressed as follows: Let $\{p_1, p_2, \ldots, p_n\} \in \mathcal{W}$ be a set of preconditions and let $\{g_1, g_2, \ldots, g_m\} \in \mathcal{G}$ be a goal. Then the corresponding rule is expressed using the pair: $\langle\{p_1, p_2, \ldots, p_n\}, \{g_1, g_2, \ldots, g_m\}\rangle$.

[2] See `http://www.swi-prolog.org`

[3] Please note that all examples described below have been simplified to show the most important issues only, and might thus appear sometimes over-simplistic. In particular we assume that there is only one room involved.

[4] Therefore they have been called *goalaviours* – *goal*-producing beh*aviours*.

[5] We use $\top$ and $\bot$ to denote the truth values true and false, respectively.

Example 3 *The goalaviour* IlluminateUser *tries to set a lamp l to full brightness whenever a user u is at the lamp's position:* $\langle\{(u, \textsf{at-position}, l)\}, \{(l, \textsf{dim}, 1.0)\}\rangle$. *A second goalaviour, called* IlluminateRoom*, tries to set the dim-value of all lamps to 0.5 to illuminate the room slightly, whenever a user is present in the room:* $\langle\{(u, \textsf{at-position}, ?)\}, \{(l, \textsf{dim}, 0.5)\}\rangle$.[6]

Obviously, the two goalaviours described in Example 1 contradict each other. To solve such conflicts, a priority is attached to the goalaviours. Every device filters the goals with respect to the priority of the emitting goalaviour and selects only those with highest priority. In addition, so-called *merge-functions* are attached to every property, which are used to merge a set of values into a single one. All goals for one property are merged to obtain the final goal. Let p be some property of a given entity and let D_p be the domain of its values. A merge-function f_p attached to p is a function from a set values to a single value: $f_\mathsf{p} : \mathcal{P}(D_\mathsf{p}) \to D_\mathsf{p}$.

Example 4 *Below we use the logical disjunction to merge the power status of a lamp and the average for the dim-value. Merging the output of the two goalaviours from Example 3 yields a value of* 0.75 *for the dim-value of the lamp next to a user.*

After filtering and merging the goals for a given entity of the ensemble, this entity is supposed to achieve the desired state. This can be done by attaching simple planners to it, which are parametrised with a formal description of all available methods. Based on the current state of the entity, the desired goal state and the available methods, the planner computes a sequence of actions to be performed to achieve the goal. As mentioned above, further details including a discussion of consistency can be found in [1].

3.2 Explanation Types in Smart Environments

As mentioned above, different types of questions to be answered have been identified in the literature [14]. To validate them for our setting, a user study has been conducted to evaluate different automatic control systems and to identify desired types of explanations [6]. The subjects (23 in total) have been asked to perform a number of tasks while being supported by our systems. While going through three increasingly complex scenarios (simple lab usage, team discussion, lecture with students and multiple projectors), the lightening conditions, projectors and surfaces were controlled automatically. About 90% of the participants explicitly stated that they liked the automatic assistance provided by our systems.

In addition to evaluating the automatic assistance, the users have been asked to write down all remarks and questions that came to their mind. While performing the experiment, the users asked for the following information: *'Why has my output been put to projection surface X?'*, *'Will there be some reaction? If yes, when will that be?'*, *'Why has the lamp X been dimmed?'*, *'Why has the projector not been turned off (yet)?'*, *'Why was the output of laptop X not moved to surface Y?'*, and *'What happens next?'*. In addition to the *'Why ...'* and *'Why not ...'* questions identified in [14] and discussed in Section 2, the users asked for timing information in form of *'When ...'* and *'Why not ... yet?'* questions. This is of importance in such an environment, because some actions are time-dependent. For example, projectors should not be turned on and off repeatedly, but only after being idle for a certain time interval. Below we discuss how to generate the answers to some of those questions automatically.

[6] Having a position implies to be within the room.

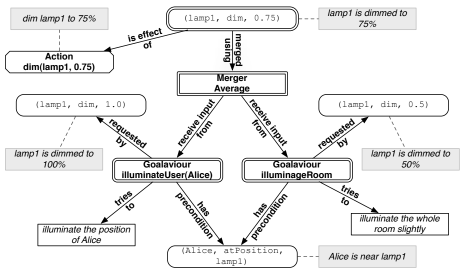


Figure 1. Explanation graph for the dim-value of lamp1. The grey boxes are not part of the specification but contain comments.

```
node(p_lamp1_dim_075, epv,
  [e=lamp1,p=dim,v=0.75,pr='is dimmed to']).
edge(merged_using,p_lamp1_dim_075,m_lamp1_dim_075).
```

Figure 2. Part of the Prolog specification of the EG from Fig. 1.

3.3 Explanation Graphs

To enable the automatic generation of answers, all events emitted by devices, all executed plans, their underlying goals and the goalaviours that emitted the goals are collected and stored. Based on this data an *explanation graph (EG)* is constructed as shown in Figures 1 and 2 (restricted to the fragment important to explain the dim-value of lamp1). Based on the history of events and actions, the causes of state changes can be tracked to either human interactions or actions invoked by the assistance system. In addition, the internal reasoning of the controller can be reconstructed from this data.

An explanation graph contains nodes for all ground instances of desired and merged EPVs, actions, goalaviours and mergers involved while achieving a given EPV. Those nodes are linked by labelled edges, representing the following dependencies: *merged using:* links EPVs and the merger used to obtain the value, *received input from:* links merger and goalaviours providing input, *requested by:* links goalaviours and desired EPVs, *tries to:* links goalaviours with their goal descriptions, *has precondition:* links goalaviours with precondition EPVs, and *is effect of:* links EPVs and causing actions. An explanation graph constructed at a given point in time contains all information needed to construct explanations for the current state of the world. As described below, it can automatically be transferred into DRSs and finally into natural language.

3.4 Discourse Representation Structures and Natural Language Generation

We use discourse representation structures (DRSs) as abstract representation for the generation of explanations. Subgraphs of the full explanation graph are transformed into DRSs, usually representing one sentence each. The subgraph highlighted by double strokes in Figure 1 is transformed into the DRS shown in Figure 4. We use a syntactically slightly modified form of DRSs used in the ACE system [7]. Instead of listing all referents occurring, we keep only those which refer to the syntactic structure to be realised, which is for example the phrase `P2` in Figure 4.

To construct DRSs from a given explanation graph, we use Prolog rules as shown in Figure 3. Each rule consists of a subgraph identification part (only the first line in Fig. 3, because this example converts a single property node only, which only needs to be a member of the EG) and a DRS composition part. The code shown in Figure 3 generates the first three lines of the DRS shown in Figure 4.[7]

[7] The mechanism to construct unique identifiers as `P1,P2,...` has been

```
drs(EG, [PS]:Conditions) :-    % EG to one DRS
  member(node(ID,epv,NC), EG),% identify subgraph
  members([e=E,v=V,pr=Pr],NC),% access node content
  Conditions = [E=object(E),  % construct object
    V=object(V),              % construct verb
    PS=predicate(E, Pr, V),   % construct predicate
    feature(PS,'Feature.TENSE','Tense.PAST')]).
```

Figure 3. Prolog code to transform parts of an EG into a DRS.

```
[P2]:[P1=object(lamp1), P3=object(0.75),
  P2=predicate(P1,'is dimmed to',P3),
  feature(P2,'Feature.TENSE','Tense.PAST')])
  G1=object('IlluminateUser(Alice)'),
  G2=object('IlluminateRoom'),
  GS1=conjunction([G1,G2]), M1=modifier(P2,by,GS1)]
```

Figure 4. DRS for the subgraph highlighted by double strokes in Fig. 1.

To actually generate explanations in natural language, we rely on the `SimpleNLG` toolkit developed at the University of Aberdeen [8]. The toolkit is a realiser which allows to generate correct English sentences based on an abstract representation. Based on the DRSs constructed above SimpleNLG code is constructed automatically. For this we adapted the syntax of the DRSs slightly. Instead of writing `object(P1,lamp1)` we use `P1=object(lamp1)` to indicate that this should be transformed into an object creation statement within the SimpleNLG framework. This notation allows to sort all conditions specified within a DRS such that in the right hand side only initialised objects occur. That is the condition `P2=predicate(P1,'is dimmed to',P3)` should be evaluated only after evaluating the conditions defining `P1` and `P3`, as it refers to them. After sorting the conditions, they are transformed into SimpleNLG statements as shown in Figure 5. Executing that code results in the string "*lamp1 was dimmed to 0.75 by IlluminateUser(Alice) and IlluminateRoom*". As mentioned above, the example has been simplified to show the key ideas here. The complete description is shown below.

3.5 Answering Why Questions

Based on the full explanation graph constructed above, why questions can be answered as follows. The system shows a list of entity–property–value triples to the user. This list is generated from nodes in the explanation graph representing EPVs not updated later, that is only those nodes corresponding to entity–property–value triples (e, p, v_1) such that there is no edge coming from a node (e, p, v_2). In our running example, the list includes the triples $(lamp1, on, \top)$, $(lamp2, on, \top)$, $(lamp1, dim, 0.75)$ and $(lamp2, dim, 0.5)$. Starting from the selected triple, the graph is traversed following the outgoing edges. Figure 1 shows the subgraph obtained for the dim-value of lamp1, that is starting from the node corresponding to $(lamp1, dim, 0.75)$. Once the subgraph is obtained, DRSs are generated as described above. Finally, every DRS is transformed into natural language using SimpleNLG.

Example 5 *After running the controller and the two goalaviours described in Example 3, the lamp1 (which is next to user Alice) is dimmed to 0.75. For this, the following explanation is generated:*

"The dim-value of lamp1 was set to 0.75*, because the goalaviours IlluminateUser(Alice) and IlluminateRoom are active. The goalaviour IlluminateUser(Alice) tries to illuminate the position of Alice. Setting the dim-value of lamp1 to* 1.0 *is a direct goal of IlluminateUser(Alice). The goalaviour IlluminateRoom tries to illuminate the full room slightly. Setting the dim-value of lamp1 to* 0.5 *is a direct goal of IlluminateRoom. The goalaviours are merged using the average. The dim-value of lamp1 has been set to* 0.75 *while performing action* `lamp1:dim(0.75)`*. The goalaviour IlluminateUser(Alice) was activated because Alice is at position lamp1. The goalaviour IlluminateRoom was activated because Alice is at position lamp1."*

omitted here, it is based on the `gensym` feature of SWI-Prolog.

```
P1 = createNounPhrase("lamp1");
P3 = createNounPhrase("0.75");
P2 = createClause(P1, "is dimmed to", P3);
P2.setFeature(Feature.TENSE, Tense.PAST);
G1 = createNounPhrase("IlluminateUser(Alice)");
G2 = createNounPhrase("IlluminateRoom");
GS1 = new CoordinatedPhraseElement(G1,G2);
P2.addModifier(createPrepositionPhrase("by", GS1));
return realiser.realise(P2);
```

Figure 5. Part of the SimpleNLG Java code obtained for the DRS in Fig. 4.

3.6 Answering Why-not and other Questions

Why-not questions can be answered similarly. But instead of finding the goalaviours which have been active, we need to determine those which might have led to the questioned state of the world. In principle, three cases can be distinguished: (1) A corresponding goalaviour has been active, but has been overwritten by another goalaviour with higher priority. (2) A corresponding goalaviour has been active, but the merging changed the result. (3) No corresponding goalaviour has been active. Generating explanations for the first two cases can be done similar to the generation for why-questions described above.

Example 6 *An automatic response to* "Why has lamp1 not been dimmed to 1.0?" *(case 2 above) is:*

"The dim-value of lamp1 was set to 0.75, because the outputs of the goalaviours IlluminateUser(Alice) and IlluminateRoom were merged using the average. The goalaviour IlluminateUser(Alice) tries to illuminate the position of Alice. Setting the dim-value of lamp1 to 1.0 *is a direct goal of IlluminateUser(Alice). The goalaviour IlluminateRoom tries to illuminate the full room slightly. Setting the dim-value of lamp1 to* 0.5 *is a direct goal of IlluminateRoom.*

In case 3, we first need to identify potential goalaviours which could lead to the questioned state, but have not been active. Then, the missing preconditions need to be identified and explanations have to be generated describing how to make those preconditions true.

3.7 Evaluation

A small user study has been conducted to evaluate a first version of the system. The running example from above has been described to the users and the generated answers were presented. The participants (14 in total) have been asked to rate the understandability, the amount of information provided within the answers (too much, or too little), and the naturalness of the generated explanations. Most of the users have been using our room and heard about the control system before, and about half of them have also participated in other user studies conducted in the lab. That is, the results are not meant to be representative for 'normal' users. Nonetheless, they show that experienced users benefit from the explanations and judge them as helpful.

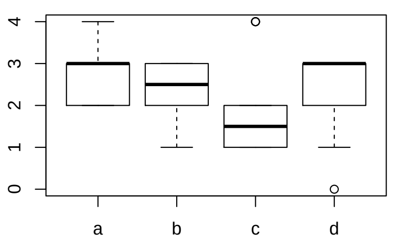


Figure 6. Results of the evaluation of a first prototype depicted as box plot, showing the median (line), the lower and upper quantile (box), and outliers.

Figure 6 shows a summary of the results. All users have been asked to answer the following questions: (a) 'Are the answers understandable?', (b) 'Do the answers contain enough details?', (c) 'Do the answers contain too much details?' and (d) 'Do the explanations sound natural?' on a Likert scale ranging from 4 = 'yes completely', 3 = 'yes', 2 = 'more or less', 1 = 'no', 0 = 'not at all'. The users considered the answers to be understandable, to sound natural and to contain (more or less) the right amount of detail. In addition to answering the questions listed in the table, the users were asked to list missing information. Most of the participants asked for information on the merging policy used to combine the outputs of the different sentences. This information (as contained in Examples 5 and 6) was not yet included in the tested prototype, but added later.

4 CONCLUSIONS AND FUTURE WORK

We showed how to generate explanations for the actions taken by an automatic assistance system for smart environments. The system's ability to explain its behaviour and internal decision making is crucial for the acceptance and the users trust in it. The explanation generation has been implemented for a distributed control system providing assistance in a smart meeting room. We showed how to generate human readable explanations for the overall behaviour of such a system. In particular, we discussed how to answer *why* and *why-not questions*. To the best of our knowledge, there is no such system which is able to support users in some real environment and to explain its actions and the current state of the world in human readable natural language.

The system employs an explanation graph to represent the history of the system and to identify underlying causes. Based on that, discourse representation structures are constructed. Those structures are later transformed into natural language sentences using a surface realiser for English. The whole explanation generation has been implemented in Prolog. A small user study has been conducted to evaluate the explanations. The users rated the explanations to be both understandable and natural.

The system, as discussed above, demonstrates the general feasibility of an automatic generation of natural language explanations for the reactions of smart environments. Nonetheless there are open issues to be addressed in the near future. The explanations generated for why-not questions are so far quite general, because sometimes it is hard to detect the correct preconditions needed to explain why something did not happen, or has not been achieved. Here ideas from the area of abduction may help to provide better explanations.

We furthermore need to extend the approach to allow chaining of explanations. The presented approach, based on the explanation graph and the DRSs, is powerful enough, but we need to investigate where to stop while identifying the true reasons, because so far only 'level 2' explanations are generated. This needs to be explored in a larger user study. In addition to evaluating the questions and answers, we plan to explore suitable user interfaces including speech interactions. Another open issue is the fine-tuning of the micro-planning for the language generation. In particular cross sentence references need to be included to make the explanations even more natural.

Finally, we plan to extend the approach to a probabilistic setting. This is necessary due to the inherent probabilistic nature of such environments. Most actions are taken based on probabilistic reasoning, which needs to be integrated into the generated explanations. And as pointed out in [13] the acceptance of explanations depends on the certainty with which they are derived.

Acknowledgment

The author has been supported by DFG graduate school 1424 MuSAMA at the University of Rostock, Germany and would like to thank three anonymous reviewers for their comments.

REFERENCES

[1] S. Bader and M. Dyrba, 'Goalaviour-based control of heterogeneous and distributed smart environments', in *Proceedings of IE'11*, pp. 142–148. IEEE, (2011).
[2] R. A. Brooks, 'A robust layered control system for a mobile robot', *IEEE Journal of Robotics and Automation*, **RA-2**(1), 14–23, (1986).
[3] D. J. Cook and S. K. Das, *Smart Environments*, Wiley, 2005.
[4] D. J. Cook, M. Huber, D. Gopalratnam, and M. Youngblood, 'Learning to control a smart home environment', *Innovative Applications of Artificial Intelligence*, (2003).
[5] A. K. Dey, 'Modeling and intelligibility in ambient environments', *J. Ambient Intell. Smart Environ.*, **1**, 57–62, (2009).
[6] M. Dyrba, R. Nicolay, S. Bader, and T. Kirste, 'Evaluation of two control systems for smart environments', in *Proceedings of CoSDEO at Mobiquitous 2011*, (2011).
[7] N. E. Fuchs, U. Schwertel, and R. Schwitter, 'Attempto controlled english - not just another logic specification language', in *Logic-Based Program Synthesis and Transformation*, ed., P. Flener, number 1559 in LNCS. 8th International Workshop LOPSTR'98, Springer, (1999).
[8] A. Gatt and E. Reiter, 'Simplenlg: A realisation engine for practical applications', in *Proceedings of ENLG-2009*, (2009).
[9] M. Ghallab, C. K. Isi, S. Penberthy, D. E. Smith, Y. Sun, and D. Weld, 'PDDL - The Planning Domain Definition Language', Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, (1998).
[10] T. Heider and T. Kirste, 'Supporting goal-based interaction with dynamic intelligent environments', in *Proceedings of ECAI 2002*, pp. 596–600, (2002).
[11] H. Kamp and U. Reyle, *From Discourse to Logic*, Kluwer, 1993.
[12] J. A. Kientz, S. N. Patel, B. Jones, E. Price, E. D. Mynatt, and Gregory D. Abowd, 'The georgia tech aware home', in *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, pp. 3675–3680. ACM, (2008).
[13] B. Y. Lim and A. K. Dey, 'Design of an intelligible mobile context-aware application', in *MobileHCI '11*, pp. 157–166. ACM, (2011).
[14] B. Y. Lim, A. K. Dey, and D. Avrahami, 'Why and why not explanations improve the intelligibility of context-aware intelligent systems', in *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pp. 2119–2128. ACM, (2009).
[15] *Handbook of Ambient Intelligence and Smart Environments*, eds., H. Nakashima, H. Aghajan, and J. C. Augusto, Springer, 2010.
[16] S. Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions*, Wiley, 2009.
[17] E. Reiter and R. Dale, *Building Natural Language Generation Systems*, Studies in natural language processing, Cambridge Press, 2000.
[18] Thomas Roth-Berghofer, Nava Tintarev, and David B. Leake, eds. *Proceedings of Explanation-aware Computing Exact 2011*, 2011.
[19] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2 edn., 2003.
[20] U. Saif, H. Pham, J. M. Paluska, J. Waterman, C. Terman, and S. Ward, 'A case for goal-oriented programming semantics', in *Proceedings of UbiSys at UbiComp 2003*, (2003).

Argumentation and explanation in the context of dialogue

Floris Bex[1] and **Katarzyna Budzynska**[2] and **Douglas Walton** [3]

Abstract. Whilst computational argumentation and explanation have both been studied intensively in AI, models that incorporate both types of reasoning are few and far between. The two forms of reasoning need to be clearly distinguished, as they may influence dialogue protocol and strategy. Using the language of the Argument Interchange Format, we show that the distinction can be made by considering the speech acts used in a dialogue, and explore some of the implications of the combination of argument and explanation.

1 INTRODUCTION

Reasoning can be characterized as the process of moving from certain starting statements, assumptions or premises, to other statements, conclusions [17]. At the same time, reasoning is also the outcome of this process (i.e. the product), a static structure. Reasoning is typically used in the context of argumentation, where premises are offered as proof of a conclusion or a claim often in order to persuade someone or settle an issue. However, reasoning is also used in the context of explanation, where the *explananda* (facts to be explained) are explained by a coherent set of *explanans* (facts that explain). The usual purpose of explanation is not to convince someone but rather to help someone understand why the explananda are the case, that is, to help the explainee understand something that she claims she does not now understand, or does not completely understand [19]. In this paper, we aim to explore the similarities and differences between argumentation and explanation and make a first step towards an integrated computational model of the two.

Argumentation and explanation are well-presented in their respective sub-fields of AI. Computational models of argumentation have emerged and matured in the past twenty-or-so years [11]. Computational models for explanation are mainly based on the technique of abductive (model-based) reasoning, and have been studied in the context of medical and system diagnosis and natural language understanding (e.g. [4, 9, 5, 14]). Despite the important role explanations can play in argumentative dialogue, there have not been many attempts to combine argumentation and explanation into one formal model. Perhaps the most thorough work thus far is by Bex et al. [3], who combine structured arguments with abductive-causal reasoning into one model of inference to the best explanation. Other examples of work in which argumentation and explanation are combined are [9, 16].

Argumentation and explanation are often used in concert when performing complex reasoning: the explanations can be the subject of argumentation or they may be used in an argumentative way. Hence, we need a model that integrates argumentation and explanation, in which the two types of reasoning are clearly distinguished; argumentation and explanation have different properties and the reasoning with arguments and explanations adheres to different patterns.

In our opinion, the only way to distinguish between argumentation and explanation is by looking at the context in which the reasoning was originally performed. In this paper, we concentrate on the contextual property of the intention of the speaker. We are interested in how to represent the connection between the intentions and the static reasoning structure under consideration. In this paper, we show that this connection can be made by using ideas from speech act theory [15]. More specifically, we argue that it is the illocutionary force of the speech act in a dialogue that determines whether reasoning is argumentation or explanation. We will use the conceptual model of the Argument Interchange Format [6, 13] so as to provide a model that is not tied to any specific dialogue or argument formalism.

The rest of this paper is organized as follows. In section 2 we elaborate on the (structural and contextual) similarities and differences between argumentation and explanation and we give some examples of both. Section 3 discusses our ideas for a framework for argumentation and explanation. Section 4 briefly explores the ramifications of combining argumentation and explanation in one comprehensive model of dialogical reasoning, and section 5 concludes the paper.

2 ARGUMENTATION AND EXPLANATION

Argumentation is a type of reasoning used in a specific *probative function*, to prove a claim [17]. By its very nature, it involves some sort of opposition between parties[4] and reasons are not just given to support a conclusion but also to remove an opponent's doubts about this conclusion. For example, a reasoning $\alpha \vdash \beta$ is argumentation when β is questioned (dubious) and a proponent of this argument uses α not only to support β, but also to remove an opponent's doubts about β. Explanation, on the other hand, has not as its main goal to prove but rather to explicate why something is the case. Explanation in its purest form is not inherently dialectical: an explanation is given to help the other party, not to convince them. Consider the following example. Say I arrive at work at ten in the morning and my boss asks why I am late. I can either explain to him that the bridge was open and that I had to wait or I can argue that I am not "late", because my contract does not specify the exact hours I have to be at the office. In first case, I am answering my boss' question by explaining to him what caused my being late. In the latter case, I am arguing against my boss claim that I am late. Thus, explanations are usually offered for propositions that both parties agree on (i.e. we agree I am "late" and

[1] School of Computing, University of Dundee, Dundee, UK, DD1 4HN, email: florisbex@computing.dundee.ac.uk
[2] Department of Logic and Cognitive Science, Institute of Philosophy and Sociology Polish Academy of Sciences, Poland email: budzynska.argdiap@gmail.com
[3] Centre for Research in Reasoning, Argumentation and Rhetoric (CRRAR), University of Windsor, Canada email: dwalton@uwindsor.ca

[4] Hence the use of the term "calculi of opposition" for argumentation-theoretic semantics that allow one to calculate the acceptability of arguments

I explain why) whilst arguments are offered for propositions that are not immediately agreed upon (i.e. I contest the fact that I am "late").

Argumentation and explanation are often used in conjunction. Explanations can themselves be the subject of argumentation, as one may argue in support or in opposition of a particular explanation or parts of it. For example, if my boss questions my explanation by arguing that I never cross a bridge on my way to work, I can argue (e.g. by providing evidence) that I do. Furthermore, explanations may be used in an argumentative way, as having someone agree to a particular explanation of a phenomenon might help us to persuade them. For example, if my boss accepts my explanation for being late I might convince him not to fire me. The most thorough work on combining argumentation and explanation thus far is [3], who combines tree-structured arguments in the ASPIC$^+$ framework [10] with abductive-causal reasoning based on standard models of explanation [9] into a *hybrid theory* of inference to the best explanation. The basic idea of this hybrid approach is as follows. A logical model of abductive-causal reasoning takes as input a causal theory (a set of causal rules) and a set of observations that has to be explained, the explananda, and produces as output a set of hypotheses that explain the explananda in terms of the causal theory. Arguments can be used to support and attack stories, and these arguments can themselves be attacked and defeated. Thus, it is possible to reason about, for example, the extent to which an explanation conforms to the evidence. This is important when comparing explanations: the explanation that is best supported and least falsified by arguments is, ceteris paribus, the best explanation.

2.1 Distinguishing Argumentation and Explanation

Because argumentation and explanation are often intertwined in complex reasoning, they can sometimes be hard to distinguish from one another. However, it is important that we do distinguish the two types of reasoning. Apart from providing a measure of conceptual neatness, there are also more concrete reasons for not confusing the two types of reasoning. One of them is that circular arguments are usually considered fallacious while circular explanations are not. Take Walton's [18] recession example. An economist is asked why the economy is in recession in a certain state at present, and she replies: "Right now a lot of people are leaving the state, because taxes are too high". But when asked why taxes are so high, she responds: "Well, a lot of people are unemployed, because of the recession". The economist has not committed the fallacy of arguing in a circle, because he was explaining human behavior which has inherent feedback loops. The second reason for correctly distinguishing between argument and explanation is that the type of reasoning used might influence the allowed and desired moves in a dialogue. The ways in which to correctly respond to an explanation are different from the ways in which one should respond to argumentation; for example, it does often not make sense for the other party to deny the explananda whilst it does make sense to deny the conclusion of an argument. Similarly, a request for information is often better met by explaining something than by arguing that something is the case.

One possible way of distinguishing between argumentation and explanation might be to look at the product of reasoning, that is, the argument or the explanation put forth, and the structure and type of this product. At first sight, it often seems an explanation is *abductive* and *causal* whilst an argument is modus-ponens style, non-causal reasoning. The basic idea of abductive inference is that if we have a general rule $\alpha \longrightarrow \beta$, meaning α *causes* β, and we observe β, we are allowed to infer α as a possible explanation of β. In contrast, argumentation is often seen as reasoning from a premise α to a conclusion β through an inference rule $\alpha \longrightarrow \beta$, where this rule need not necessarily be causal. However, as it turns out it is also possible to give abductive or causal *arguments* (see e.g. Walton's [21] argument from evidence to hypothesis and causal argument). Similarly, one may perform explanatory reasoning by taking a rule $\beta \longrightarrow \alpha$, meaning β *is evidence for* α (see Bex et al. [3] for a discussion on evidential and causal reasoning).

In our opinion, the distinction between argumentation and explanation is not one that is inherent to the product of reasoning, the static structure. Rather, the distinction follows from the dialogical context in which the reasoning was originally performed. In order to determine this context, we need not just look at the original intention of the speaker but also at the broader dialogical context, such as the utterance that was replied to by the speaker and the intentions of the other participants. In other words, the context is largely determined by the *speech acts* that were performed. According to the pragmatic theory of speech act [15] argumentation and explanation are different speech acts. A speech act Fα, such as: claimα, whyα, consists of an illocutionary force F and a propositional content α. An illocutionary force is an intention of uttering a propositional content. That is, the performer of a speech act may utter α with an intention of asserting, asking, promising and so on. Thus, argumentation and explanation are both instances of illocutionary acts that represent a relation between premises and conclusions: argue(α, β) and explain(α, β), where α denotes a conclusion and β denotes premises. The distinction between argumentation and explanation cannot just be made by looking at the original speech act; one also needs to consider the broader dialogical context. In the next section, we show how this can be represented in the AIF core ontology.

3 ARGUMENTATION AND EXPLANATION IN THE ARGUMENT INTERCHANGE FORMAT

The Argument Interchange Format (AIF) is a communal project which aims to consolidate some of the defining work on computational argumentation [6]. Its aim is to facilitate a common vision and consensus on the concepts and technologies in the field so as to promote the research and development of new argumentation tools and techniques. In addition to practical aspirations, such as developing a way of interchanging data between tools for argument manipulation and visualization [7], a common *core ontology* for expressing argumentative information and relations is also developed. Thus, the AIF ontology aims to provide a bridge between linguistic, logical and formal models of argument and reasoning.

The AIF core ontology is first and foremost an abstract, high-level specification of information and the various argumentative relations (e.g. inference, conflict) between this information.[5] The core ontology is intended as a conceptual model of arguments and the schemes or patterns arguments generally follow. It defines arguments and their mutual relations as typed graphs [6, 12], which is an intuitive way of representing argument in a structured and systematic way without the formal constraints of a logic [6]. This section briefly describes how in general the AIF describes argument and its dialogical context (Section 3.1). Then, we propose how to model argumentation and explanation in the language of the AIF (Section 3.2).

[5] The name Argument Interchange *Format* is in this respect somewhat misleading, as it seems to imply that AIF is a file format, whereas the AIF ontology can be implemented in a number of specific formats (XML, DOT, SQL). However, the name is retained for historical reasons.

3.1 The AIF Core Ontology

The AIF core ontology [6, 12] and its dialogical extension [13] allows for the explicit representation of both reasoning structure and the context of dialogue in which it is put forth. More concretely, it enables to connect the locutions uttered during a dialogue (argument$_2$) and the underlying arguments expressed by the content of those locutions (argument$_1$).

In the ontology, argument$_1$ is represented by two kinds of nodes:

- information (I-) nodes, which refer to data, and
- scheme (S-) nodes, which refer to the passage between information nodes, which are classified into three groups:
  - rule application (RA-) nodes which correspond to inference or support,
  - conflict application (CA-) nodes which correspond to conflict or refutation,
  - preference application (PA-) nodes which correspond to value judgements or preference orderings.

The argument$_2$ is also described by two types of nodes:

- locution nodes (L-), which refer to utterances and constitute a subclass of information nodes, and
- transition application (TA-) nodes, which refer to the passage between locutions.

The TA-nodes are governed by the protocol of a dialogue system, recording e.g. that a given assertion has been made in response to an earlier question [13, 2].

The interaction between argument$_1$ and argument$_2$ is captured by means of two types of illocutionary application (YA-) nodes [13]:

- the YA-nodes between I-nodes and L-nodes, and
- the YA-nodes between RA-nodes and TA-nodes.

For example, an YA-node may represent the relation between an assertion claimα with its propositional content α. The YA-link is determined and warranted (authorized) by the constitutive rules for speech acts [15]. These rules determine what constitutes a successful speech act. For example, an assertion may be unsuccessful and attacked, if its performer did not have enough evidence for the statement or he declared what he actually disbelieves.

3.2 The Distinction between Argument and Explanation in AIF

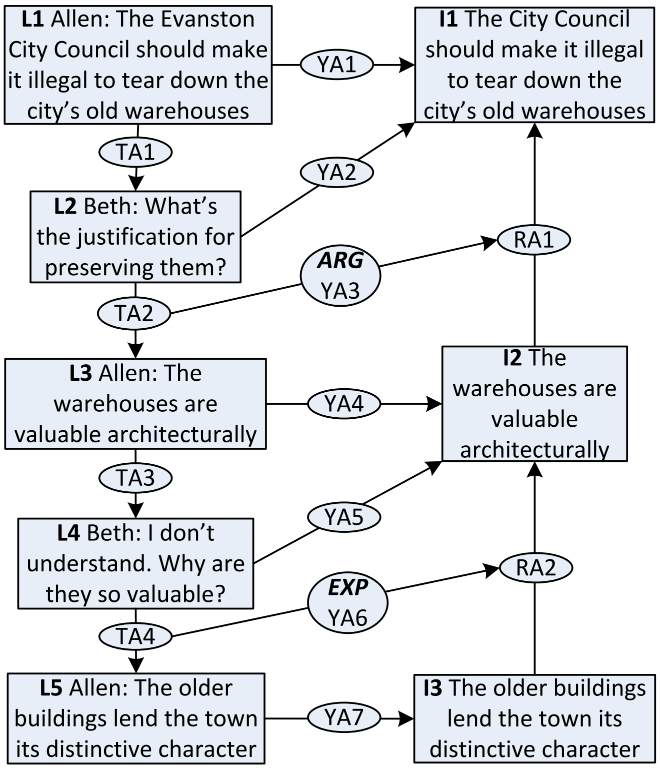
In this section, we propose the specification of argumentation and explanation in the AIF core ontology. We will illustrate it on the example adapted from Walton [18].

Allen The Evanston City Council should make it illegal to tear down the citys old warehouses.
Beth Whats the justification for preserving them?
Allen The warehouses are valuable architecturally.
Beth Why are they so valuable?
Allen The older buildings lend the town its distinctive character.

As is pointed out by Walton and Bex [20], Beth's first question clearly asks for an argument (a justification). Beth's second question is ambiguous: it could ask for either an argument or an explanation. This depends on whether Beth does not understand why the buildings are valuable or whether Beth has doubts about the buildings' value. This in turn depends on Beth's beliefs or commitments about 'The warehouses are valuable architecturally'; if Beth believes or is openly committed to this proposition, we can assume that she is asking for an explanation, as there is no doubt. For our example, we assume that Beth is asking for an explanation.

In the dialogue between Allen and Beth (see Fig. 1), the argument$_2$ consists of five speech acts represented by L-nodes (we use abbreviation L_i to denote subsequent locution nodes). The argument$_1$ consists of three propositions represented by I-nodes (I_i means subsequent information nodes). The interaction between the argument$_2$ and the argument$_1$ is described by means of the YA-nodes. The speech acts L_1, L_3 and L_5 have assertive illocutionary force connecting them with propositional contents I_1, I_2 and I_3, respectively. The passage between L_1 (resp. L_3, L_5) and I_1 (resp. I_2, I_3) is represented by YA_1 (resp. YA_4, YA_7). The illocutionary node YA_2 (resp. YA_5) links the directive L_2 (resp. L_4) and its propositional content I_1 (resp. I_2): not all YA-nodes are assertive schemes.

The most interesting is the complex type of illocutionary force which could be treated as intention of arguing and explaining. In the AIF core ontology, the complex illocution is represented by the YA-nodes between RA-nodes and TA-nodes [13]. In Fig. 1, there are two such nodes: YA_3 and YA_6. According to the assumption made above, YA_3 corresponds to argumentation and YA_6 to explanation. The illocution YA_3 links Allen's response to Beth's challenge (i.e. TA_2) with the argument "The warehouses are valuable architecturally" for the claim "The Evanston City Council should make it illegal to tear down the citys old warehouses" (i.e. RA_1). This captures the intuition that Allen's argumentation is invoked by Beth's challenge. The illocution YA_6, however, links Allen's response to Beth's request for information (i.e. TA_4) with the explanation "The older buildings lend the town its distinctive character" for the claim "The warehouses are



L1 Allen: The Evanston City Council should make it illegal to tear down the city's old warehouses
YA1
I1 The City Council should make it illegal to tear down the city's old warehouses
TA1
YA2
L2 Beth: What's the justification for preserving them?
ARG YA3
RA1
TA2
L3 Allen: The warehouses are valuable architecturally
YA4
I2 The warehouses are valuable architecturally
TA3
YA5
L4 Beth: I don't understand. Why are they so valuable?
EXP YA6
RA2
TA4
L5 Allen: The older buildings lend the town its distinctive character
YA7
I3 The older buildings lend the town its distinctive character

Figure 1. The AIF core ontology description of the example from [22]

valuable architecturally" (i.e. RA_2). This captures the intuition that Allen's explanation is invoked by Beth's request for information.

Observe that we could represent argumentation and explanation as YA_4 and YA_7, respectively. However, in such a representation they are indistinguishable from simple assertion. Assigning argumentation and explanation to the TA- and RA-nodes captures the intuition that they are social processes that emerge from the interaction between agents such that one agent responds to interlocutor's request for justification or explanation.

4 SCHEMES FOR ARGUMENTATION AND EXPLANATION

Using the machinery of the AIF core ontology described in section 3, we can distinguish between argumentation and explanation according to the dialogical context in which they are used. This allows us to combine argumentation and explanation in a principled way and paves the way for complex reasoning where, for example, arguments are used to justify explanations (cf. [3]) or explanation is used to clarify parts of an argument (as is the case in the example in Fig. 1, where the premise of the argument RA_2 is explained).

The introduction of explanation into the AIF core ontology has a profound effect on the patterns of reasoning that are included in the ontology. Recall that the AIF core ontology is not only intended as a conceptual model of (object-level) arguments and explanations like the ones presented in section 3, but also as a repository of the schemes or patterns arguments generally follow. To this end, the core ontology also includes a so-called Forms Ontology, which contains these schemes. So in the ontology, relations like inference, conflict, transition and so on are treated as genera of a more abstract class of schematic relationships, which allows the three types of relationship to be treated in more or less the same way, which in turn greatly simplifies the ontological machinery required for handling them. Thus, inference schemes, conflict schemes and transition schemes in the Forms Ontology embody the general principles expressing how it is that q is inferable from p, p is in conflict with q, and p is answerable with q, respectively. The individual RA-, CA- and PA-nodes that fulfil these schemes then capture the passage or the process of actually inferring q from p, conflicting p with q and answering p with q, respectively.

Inference schemes in the AIF ontology are similar to the rules of inference in a logic, in that they express the general principles that form the basis for actual inference. They can be deductive (e.g. the inference rules of propositional logic) or defeasible (e.g. argumentation schemes). Take, for example, the inference scheme for Argument from Expert Opinion [21]:

- *premises*: E is an expert in domain D, E asserts that P is true, P is within D;
- *conclusion*: P is true;
- *presumptions*: E is a credible expert, P is based on evidence;

Now, AIF arguments fulfil these schemes in a similar way to how inferences in logic instantiate inference schemes. For example, the argument *Peter says that the buildings are valuable architecturally* **and** *Peter is an expert on architecture* $\longrightarrow RA_3 \longrightarrow$ *the buildings are valuable architecturally* would fulfil the scheme for argument from expert opinion. Note that the presumption, that Peter is credible and that his assertion is based on evidence, is not explicitly needed in the argument that fulfils the scheme: the idea of presumptions is that they can be assumed to hold unless proven otherwise. Thus, specific (but still generalizable) knowledge can be modelled in the AIF in a principled way using argumentation schemes, for which we can assume, for example, a raft of implicit assumptions which may be taken to hold and exceptions which may be taken not to hold. These argumentation schemes then tell us how we can build valid and coherent arguments.

4.1 Transition Schemes

An argumentative dialogue (i.e. $argument_2$) has an (often implicit) reply structure that contains the connections between the locutions in a dialogue. In the language of the AIF core ontology, these connections are explicitly rendered as transitions or TA-nodes (section 3). These transitions form the "glue" that keeps the locutions together and makes a dialogue coherent. This is analogous to non-dialogical argument, where logical (inference) connections (in the form of RA-nodes) form the glue between the individual propositions. The exact principles that make a dialogue coherent have been formulated and studied in the literature on formal dialogue systems [8]. At the heart of these systems are the dialogue protocols that describe a dialogue games permitted locutions, how and when the dialogue starts and ends and, perhaps most importantly, how locutions may be combined into exchanges.

In [2], the authors discuss *transition schemes* (following earlier work by Reed et al. [13]), schematic representations of a single transition in a dialogue. These transition schemes which can be instantiated to form transitions (i.e. a step in a dialogue), and these transitions can then be chained to form a dialogue. Note that the ontological machinery at work here is (intentionally) very similar to that of argumentation schemes, schematic representations of inference that can be instantiated to form inferences, which can be chained to form arguments. As an example of a transition scheme, consider the transition TA_3 in Fig. 1, which is a particular instantiation of the following general scheme.

- *start locution*: Assert P;
- *end locution*: Request Explanation P

This scheme stands for the fact that assertions may be responded to by requesting an explanation of the information that is asserted. Another scheme is the one that is fulfilled by TA_4, which says that an explanation can be given if the other party requests it.

- *start locution*: Request Explanation P;
- *end locution*: Explain P

Thus transition schemes can be used to enforce, for example, that (as in the above scheme) an explanation may only be given when the other party asks for it. As Bex and Reed [2] show, it is also possible to define presumptions for transition schemes. For example, we might say that in order for someone to request an explanation after an assertion, the requesting party must somehow not understand the assertion completely (recall that explanations are often aimed at improving understanding). This can be incorporated into the above *assert – request explanation* scheme as a presumption, which means that the fact that the requester does not understand the assertion is implicitly assumed. That is, the requester does not have to explicitly say "I don't understand" unless his understanding is actively challenged (i.e. "Why are you asking for an explanation, I think you understand perfectly!").

Exactly which transition schemes are important and which conditions on these schemes (in the form of presumptions) we need has been discussed in [19], where pre- and postconditions for the use of explanation are proposed. It remains to be investigated which types of conditions would be appropriate for a combination of argumentation and explanation. For example, one would only request an argument for some claim if there is doubt about this claim, and one would only request an explanation about a claim if there is a lack of understanding. Exactly how doubt or understanding should be defined remains as of yet an open question.

4.2 Explanation Schemes

In addition to argumentation schemes, there has also been work on what we call explanation schemes or scripts [14]. An explanation scheme is a generic scenario, an abstract rendering of a sequence of actions or events of a kind that is familiar to both the explainer and the explainee based on their common knowledge of how things can be normally expected to go in situations they are both familiar with. For example, the restaurant-script [14] contains information about the standard sequence(s) of events that take place when somebody goes to dine in a restaurant. Similar to argumentation and transition schemes, general explanation schemes can be instantiated by particular explanations and the scheme in a sense provides the conditions for the explanation's coherence (just as the argumentation scheme tells us what a coherent argument is and a combination of transition schemes tells us what a coherent dialogue is).

Take, for example, a man who enters a restaurant, orders some soup and gets his soup from the waiter. A natural continuation of this script would be that the man proceeds to eat his soup. If, for example, the man would instead remove his pants and offer them to the waiter, the story would be less coherent, because it does not seem to adhere to the typical restaurant scheme. But if this story fits another explanation scheme it can still be coherent. Suppose information is added to the script that the waiter spilled the hot soup on the man's legs. This new information would fill out the story in such a way that it hangs together as a coherent script about what happens when someone spills hot liquid on one's clothes. An expanded version of the story provides an explanation that helps the explainee to understand what happened. The explanation may be causal, motivational, teleological, or represent other kinds of explanations. We can represent the sequence of actions and events in this kind of story at a higher level of abstraction by fitting the script into an explanation scheme as an instance of it.

While the use of explanation schemes in argumentation has been explored recently [1], it is still unclear how they might be used in dialogue. Furthermore, what is currently also lacking is a principled exploration of different types of explanation schemes. Such explorations have been performed for argumentation schemes (e.g. [21]) and recently also for transition schemes [2].

5 CONCLUSIONS

In the paper, we propose the basic framework (based on the AIF ontology) for representing the difference between argumentation and explanation as a difference in illocutionary force (represented as YA-nodes in the AIF ontology). Thus, we lay the basis for a principled combination of argumentation and explanation not only as reasoning structures but also in the context of reasoning processes or dialogues. We further explore some of the ramifications of combining argumentation and explanation, and how this combination is going to influence future work on reasoning schemes or patterns.

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of EPSRC under grant EP/G060347/1 for Floris Bex, and the support from Polish National Science Center for Katarzyna Budzynska under grant 2011/03/B/HS1/04559 and the support of the Social Sciences and Humanities Research Council of Canada for Insight Grant 435-2012-0101 for Douglas Walton.

REFERENCES

[1] F. Bex, T. Bench-Capon, and B.Verheij, 'What makes a story plausible? the need for precedents', in *Legal Knowledge and Information Systems. JURIX 2011: The Twenty-Fourth Annual Conference*, ed., K.D. Atkinson, pp. 23–32, (2011).

[2] F.J. Bex and C. Reed, 'Dialogue templates for automatic argument processing', in *Computational Models of Argument. Proceedings of COMMA 2012*, (2012). to appear.

[3] F.J. Bex, P.J. van Koppen, H. Prakken, and B. Verheij, 'A hybrid formal theory of arguments, stories and criminal evidence', *Artificial Intelligence and Law*, **2**, 123–152, (2010).

[4] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson, 'The computational complexity of abduction', *Artificial Intelligence*, **49**, 25–60, (1991).

[5] A. Cawsey, *Explanation and Interaction: The Computer Generation of-Explanatory Dialogues*, MIT Press, Cambridge, MA, 1992.

[6] C.I. Chesñevar, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott, 'Towards an argument interchange format', *The Knowledge Engineering Review*, **21**, 293–316, (2006).

[7] J. Lawrence, F. Bex, M. Snaith, and C. Reed, 'Aifdb: Infrastructure for the argument web', in *Computational Models of Argument. Proceedings of COMMA 2012*, (2012). to appear.

[8] Peter Mcburney and Simon Parsons, 'Dialogue Games for Agent Argumentation', in *Argumentation in Artificial Intelligence*, eds., Iyad Rahwan and Guillermo Simari, volume 22, 261–280, Springer, (2009).

[9] D. Poole, A. Mackworth, and R. Goebel, *Computational Intelligence*, Oxford University Press, 1998.

[10] H. Prakken, 'An abstract framework for argumentation with structured arguments', *Argument and Computation*, **1**, 93–124, (2010).

[11] H. Prakken and G.A.W. Vreeswijk, 'Logics for defeasible argumentation', in *Handbook of Philosophical Logic*, eds., D. Gabbay and F. Günthner, volume 4, 219–318, Kluwer Academic Publishers, Dordrecht/Boston/London, second edn., (2002).

[12] I. Rahwan, F. Zablith, and C. Reed, 'Laying the foundations for a world wide argument web', *Artificial Intelligence*, **171**, 897–921, (2007).

[13] C.A. Reed, S. Wells, K. Budzynska, and J. Devereux, 'Building arguments with argumentation : the role of illocutionary force in computational models of argument', in *Proceedings of COMMA 2010*, (2010).

[14] R. Schank, *Explanations Patterns: Understanding Mechanically and Creatively*, Lawrence Erlbaum, Hillsdale, NJ, 1986.

[15] J. Searle, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, 1969.

[16] G. Tecuci, D. Marcu, M. Boicu, D.A. Schum, and Russell K., 'Computational theory and cognitive assistant for intelligence analysis', in *Proceedings of the Sixth International Conference on Semantic Technologies for Intelligence, Defense, and Security*, pp. 68–75, (2011).

[17] D. Walton, 'What is reasoning? what is an argument?', *The Journal of Philosophy*, ((87)8), 399–419, (1990).

[18] D. Walton, 'Epistemic and dialectical models of beggingthe question', *Synthese*, (152), 237–284, (2006).

[19] D. Walton, 'Dialogical models of explanation', in *Papers from the 2007 AAAI Workshop*, Association for the Advancement of Artificial Intelligence Technical Report WS-07-06, pp. 1–9, Menlo Park, California, (2007). AAAI Press.

[20] D. Walton and F.J. Bex, 'Combining explanation and argumentation in dialogue', in *Computational Models of Natural Argument - Prooceedings of CMNA12*. Springer, (2012). to appear.

[21] D.N. Walton, C.A. Reed, and F. Macagno, *Argumentation Schemes*, Cambridge University Press, Cambridge, 2008.

Visualization of Intuitive Explanations using Koios++

Björn Forcher[1], Nils Petersen[2], Andreas Dengel[3], Michael Gillmann[4], Zeynep Tuncer[5]

Abstract. In a certain sense, explanations in computer science are answers to questions and often an explanatory dialog is necessary to support users of a software tool. In this paper, we introduce the concept of *intuitive explanations* representing the first explanations in an explanatory dialog. Based on an abstract approach of explanation generation we present the generic explanation component *Koios++* applying Semantic Technologies to derive intuitive explanations. We illustrate our generation approach by means of the information extraction system *smartFIX* and put a special emphasis on visualizing explanations as semantic networks using a special layouting algorithm. smartFIX itself is a product portfolio for knowledge-based extraction of data from any document format. The system automatically determines the document type and extracts all relevant data for the respective business process. In this context, Koios++ is used to justify extraction results.

1 Introduction

In a certain sense, explanations in computer science are answers to questions [7] and often an explanatory dialog [1] is necessary to support users of a software tool. In this paper we introduce the concept of *intuitive explanations* that can be illustrated by means of a daily situation. Imagine, a nine year old child complains about dizziness and visits the doctor for an examination. After the examination the doctor concludes that the child suffers from the *Mènière's disease*. The child does not know this particular disease and asks the doctor what it is. In other words, the child requests an explanation that helps to understand what dizziness and Mènière's disease have to do with each other. Probably, the doctor will not elaborate on the Mènière's disease nor he will make use of foreign words. On the contrary, he will roughly estimate the knowledge of the child and, based on that, he will give a short explanation. For instance, *Mènière's disease is a disease of the inner ear which causes, for instance, dizziness* is an understandable explanation, enabling the child to take up the given information and ask further questions. In the figurative sense, *intuitive explanations* represent the first explanations in an explanatory dialog in which the explainer tries to give an understandable explanation based on a rough estimation of the knowledge of his counterpart. The explanation enables the consumer of the explanation to ask further questions leading to a complex explanatory dialog.

[1] German Research Center for Artificial Intelligence (DFKI) GmbH, Germany, email: bjoern.forcher@dfki.de
[2] German Research Center for Artificial Intelligence (DFKI) GmbH, Germany, email: nils.pertersen@dfki.de
[3] German Research Center for Artificial Intelligence (DFKI) GmbH,Germany, email: andreas.dengel@dfki.de
[4] Insiders Technologies GmbH, Germany, email: m.gillmann@insiders-technologies.de
[5] John Deere European Technology Innovation Center, Germany, email:TuncerZeynep@johndeere.com

In smartFIX documents are classified automatically on the basis of free form and forms-analysis methods [5]. Relevant data is extracted using different methods for each document type and is validated and valuated via database matching and other sophisticated knowledge-based methods. Due to mathematical and logical checks data quality is enhanced. Data that is accurately recognized is released for direct export. In contrast, unreliably recognized data is forwarded to a verification workplace for manual checking. In many cases, users have no difficulties to extract information from a document. Consequently, they often do not understand the difficulties of smartFIX during the extraction process. Making the system more transparent, smartFIX creates a semantic log that is used by the generic explanation component *Koios++* to justify extraction results. As explanations are answers to questions, explanations represent some kind of information that can be externalized by text or by charts. Currently, Koios++ uses semantic networks along with a special layouting algorithm to visualize explanations. We will point that this kind of visualization is a useful alternative to textual explanations enabling intuitive explanations and thus the start of an explanatory dialog.

This paper is structured as follows. The next section gives a short overview about relevant research on explanations and describes an abstract explanation generation approach. Sect. 3 presents the smartFIX system and motivates its explanation need by an intuitive example. Sect. 4 presents the generic explanation component *Koios++* whereas the following describes the explanation-aware layouting algorithm. We conclude the paper with a brief summary and outlook.

2 Related Work

Wick and Thompson [12] developed the expert system REX, which implements the concept of *reconstructive explanations*. REX transforms a trace, a line of reasoning, into a plausible explanation story, a line of explanation. The transformation is an active, complex problem-solving process using additional domain knowledge. The degree of coupling between the trace and the explanation is controlled by a filter that can be set to one of four states regulating the transparency of the filter. The more information of the trace is let through the filter, the more closely the line of explanation follows the line of reasoning. In this work, we describe how semantic technologies can be applied to (re-) construct explanations.

In [2] we described our conceptual work on explaining smartFIX. In our explanation scenario (Fig. 1) we distinguish three main participants: the *user* who is corresponding with the software system via its user interface (UI), the *originator*, the tool that provides the functionality for the original task of the software and the *explainer*. Originator (smartFIX) and explainer (Prof. Smart) need to be coupled in order to provide the necessary knowledge about the inner workings of the originator for the explainer. In (rule-based) expert systems looking at the rule trace was the only way of accessing the originator's actions. Given that the inference mechanism is fixed in those systems the trace was all the explainer needed.

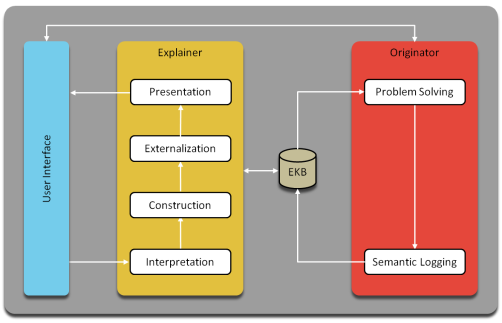


Figure 1. Explanation Generation [2].

The mentioned scenario implies that the originator has to provide detailed information about its behavior and solutions. Therefore it is necessary that the originator prepares some kind of log representing the initial starting point for the explainer to generate explanations. Regarding user questions, this information is step-by-step being transformed into an adequate explanation. Thus, a multi-layered explanation model is constructed, whereas each step contributes a layer to the model, *i. e.*, the transformation result.

Depending on the coupling, originator and explainer share information that is required for problem solving and for explanation generation as well. In Fig. 1, this information is contained in the explanation knowledge base (EKB). The originator may have access to information which is hidden from the explainer and vice versa. At least, they have to share the *semantic log*. As its name implies, the *logging process* collects all information with respect to the behavior of the originator for building the log.

Users communicate their explanation needs by keywords or in natural language. As the formal language of originator and explainer is often completely different from the user's language an *interpretation process* is necessary. In simplified terms, relevant parts of the semantic log and EKB must be identified and the exact explanation needs of the user must be determined. The result of the interpretation process is called *translation layer*.

The translation layer does not necessarily represent adequate explanation information. Until this stage, the explainer is only *aware* of the users' explanation problem concerning, for instance, an incomprehensible result of the originator. However, the information that solves the users' explanation problem completely has not been derived. This is the task of *construction process* which is similar to the concept of reconstructive explanations. The result of that process is called *content layer* representing useful explanation information. As understandability is a very important aspect of explanation [10, 11], explanations should not contain too much or too confusing information. Furthermore, it takes up the knowledge of the user and reveals a connection between known and unknown information enabling the user to ask follow-up questions.

Explanation is information that is communicated by text, charts, tables, *etc*. Each communication form has different application possibilities in an explanation scenario. Text can describe complicated conceptions whereas charts can reveal qualitative connections between concepts in a simple way [13]. The *externalization process* transforms the content layer into a formal description for communicating explanations, namely the *externalization layer*. In this work, we put a special emphasis on semantic networks based on mathematical graphs for depicting explanations.

However, this layer does not include layout and style information. This is task of the presentation process which transforms the content of the externalization layer into the presentation layer. The information of the last mentioned layer can be used by special renderer to visualize the explanation. As explained, the presentation process and the layouting of the semantic network is the main contribution of this work. We will show how the layout can contribute to intuitive explanations and the explanatory dialog.

The explainer creates some kind of meta log that is indicated by the arrow between EKB and explainer. In a way, the explainer is *aware* of single transformation processes and layers and thus, it *knows* how it explains a certain explanation problem. This knowledge is essential to continue the explanatory dialog and offers sophisticated interaction possibilities.

So far, we described an abstract method to generate explanations. In the next chapters we describe a realization of that method with the help of Semantic Technologies.

3 smartFIX

smartFIX extracts data from paper documents as well as from many electronic document formats (e.g., faxes, e-mails, MS Office, PDF, HTML, XML, etc.). Regardless of document format and structure, smartFIX recognizes the document type and any other important information during processing.

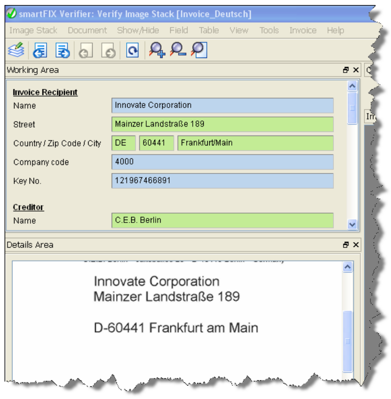
Basic image processing such as binarization, despeckling, rotation and skew correction is performed on each page image. If desired, smartFIX automatically merges individual pages into documents and creates processes from individual documents. For each document, the document class and thus the business process to be triggered in the company is implicitly determined. smartFIX subsequently identifies all relevant data contained in the documents and related to the respective business process. In this step, smartFIX can use customer relation and enterprise resource planning data (ERP data) provided by a matching database to increase the detection rate. A special search strategy searches for all entries from the customer's vendor database on the document. The procedure works independently of the location, layout and completeness of the data on the document. Within smartFIX this strategy is called "Top Down Search". Moreover, smartFIX provides self-teaching mechanisms as a highly successful method for increasing recognition rates. Both general and sender-specific rules are applied. An automatic quality check is then performed on all recognized values. Beside others, Constraint Solving [4] and Transfer Learning methods [8] are used. Values that are accurately and unambiguously recognized are released for direct export; uncertain [9] values are forwarded to a verification workplace for manual checking and verification. The quality-controlled data is then exported to the desired downstream systems, e.g., an enterprise resource planning system like SAP for further processing. An overview of the system architecture is also presented in [2].

Let us illustrate exemplary an actual scenario that currently results in support calls and internal research and clarification effort by experts.

Often, several subcompanies of the same trust are resident at the same location or even in the same building. If one smartFIX system has to analyze, for instance, invoices of more than one of those companies, very similar database entries can be found in the customer's master database.

The company's master data is an important knowledge source used by Top Down Search during the analysis step of smartFIX. When smartFIX analyzes an invoice sent to such a subcompany it may be unable to identify a clear and unambiguous extraction result due to the high degree of similarity of the master data entries. So, smartFIX has to regard all the subcompanies as possible hits.

smartFIX extracts the most reliable result based on extraction rules. Here, it does not valuate that result as reliable but as a suggestion [9]. Fig. 2 presents a look into the smartFIX Verifier in that case. You see that the recipient's name and identifier are correctly extracted but the values are marked blue which means "uncertain" in the smartFIX context.



Invoice Recipient

Name	Innovate Corporation
Street	Mainzer Landstraße 189
Country / Zip Code / City	DE 60441 Frankfurt/Main
Company code	4000
Key No.	121967466891

Creditor

Name	C.E.B. Berlin

Innovate Corporation
Mainzer Landstraße 189

D-60441 Frankfurt am Main

Figure 2. Analysis results presented in smartFIX Verifier

With this picture on screen, the user wonders why the system asks for interaction (here, for pressing the Return key to confirm the correct extraction results) although she can clearly and easily read the full recipient's address on the invoice. This scenario holds, too, and becomes more intransparent the more extraction rules and sophisticated extraction and valuation methods come into operation.

smartFIX creates a *semantic log* that is based on the *smartFIX Ontology* (SMART) that is an extension of the OWL-S ontology[6] and the EXACT ontology as well. OWL-S provides a set of representation primitives capable of describing features and capabilities of Web services in unambiguous, machine-interpretable form. This includes, among other things, the possibility to describe how the service works. OWL-S comprises general constructs to represent processes, results and intermediate results. The EXACT ontology provides representational constructs to describe the explanation generation approach including primitives for semantic networks (nodes, edges, ...), layouting (cell-layout, box-layout, ...) and presentation (fonts, ...). The SMART ontology integrates both ontologies regarding smartFIX specific aspects. This allows not only describing the behavior of smartFIX in an abstract way, but also to instantiate a concrete log with respect to the *Semantic Logging* step as presented in section 2.

4 Explanation Component Koios++

In this section we present an implementation of the abstract explanation generation method as presented in section 2 by means of Semantic Technologies. As mentioned before, smartFIX creates a semantic log that is encoded with the SMART ontology. A specific log is leveraged by the generic explanation component Koios++ and thus, it is starting point for the generation processes Interpretation, Construction, Externalization und Presentation. The semantic log represents an RDF-Graph which is stepwise transformed into the RDF-Graph of the presentation layer.

At its core, Koios++ represents a semantic search engine that enables keyword-based search on graph-shaped RDF data [6]. In addition, it includes various manipulation strategies with which any RDF-Graph can be transformed into another RDF-Graph. For instance, Koios++ employs the Quadruple Prolog Language (QPL) [3] to transform an RDF-Graph into another one by means of set of predefined Prolog rules. As a consequence, Koios++ offers all prerequisites to realize the abstract explanation generation method as presented in chapter 2. In this section, we describe how Koios++ is used to explain the smartFIX system in order to justify extraction results.

For interpreting the user's explanation needs the semantic search algorithm of KOIOS++ is applied, realizing a keyword-based search on the semantic log. The search engine maps keywords to elements of the log and searches for connections between them. More precisely, every keyword k_1 to k_z is mapped to a set of mapping elements m_1 to m_z. The mapping element sets were distributed to the threads t_1 to t_z and in each thread t_g a graph exploration is performed for each mapping element $e_{g_u} \in m_g$ and thus, many paths were determined starting from e_{g_u}. In case there is a log element r that is reached by any path in each thread a connecting subgraph of the log can be constructed consisting of z paths. To conclude, the engine determines a set of subgraphs representing basic explanation information including an articulation point r that is called root or connecting element. Regarding the example as presented in Sect. 3 a user may use the keywords *analyser*, *recipient* and *uncertain result* in order to find out why the recipient is not identified correctly. A subgraph that connects the corresponding elements represents the basic information to understand why the result is unreliable. In general, users do not use the same keywords to specify their explanation need depending on the user's level of expertise. For that reason, we enrich the SMART ontology with synonyms taken from the WordNet thesaurus[7]. As a result, users can also use *unsure result* instead of *uncertain result*. However, keywords generally map on several elements of the log. Hence, subgraphs are ranked depending on the weighting of the graph elements representing possible explanation alternatives.

As explained above, the *Interpretation* step determines an extract of the log. Potentially, this extract is still not understandable for non-expert smartFIX users. After initially experimenting with explanations of smartFIX it turned out that the extract of the log contains too much information. Primarily, the log includes a long sequence of subprocess which contradicts the concept of intuitive explanations. For that reason, we focused in particular on shortening the determined extract of the log. For shortening we defined several QPL rules that leverage the transitivity characteristic of the subprocess relation that do not affect the truth content of the explanation. Hence, the construction process transforms the log extract into a smaller RDF-Graph. Apart from that, connecting or mapping elements are never removed and there is always a connection between all mapping elements. Actually, this is the content of the explanation that must be visualized only that is described in the next chapter.

Fig. 4 is an extract of the construction layer regarding the example as described in section 3. It contains the information, that process *root* process has a subprocess called *top down search*.

5 Explanation-Aware Graph-Layouting

As explained, the construction process returns the informational content of the explanation that must be externalized. We decided to use

[6] `http://www.w3.org/Submission/OWL-S/`

[7] `http://wordnet.princeton.edu/`

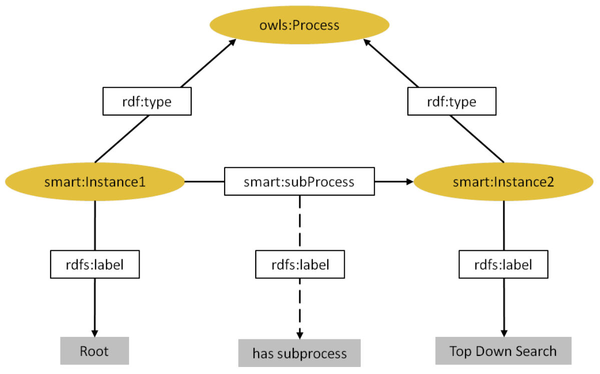


Figure 3. Extract of the construction layer

semantic networks that have three reasons. First, semantic networks are understandable alternative to textual information enabling a fast recognition of qualitative connections. Second, RDF-Graphs can be easily transformed into semantic networks and third, they enable various interaction possibilities and hence, the continuation of the explanatory dialog.

In linguistics, an entity of the real world is characterized by three components, namely label, connections to other entities and a complex pattern of perceptual origin. The label is used to communicate the entity in natural language and in many cases the entity is perceived visually. As described above, the construction layer contains only informational content of the explanation and thus, it includes labels but no visual information. Images, for instance, make sense in a semantic network but not in a pure textual explanation. For that reason, the construction layer includes class relations that can be used in texts to characterize instances more precisely, for example, *top down research process*. Regarding semantic networks it is possible to use both, labels and symbols to characterize an entity. If an instance cannot be associated with an individual symbol, it is may be possible to characterize the instance with a class symbol, for example, a gear symbol to represent the class *owls:Process*. The transformation of the construction layer is illustrated in Fig. 4. The figure shows the root process *smart:Instance1* of Fig. 3 is transformed into node of the semantic network which is the source of an edge and which has a label and a symbol. It should be noted, that mapping and connecting elements of the interpretation layer correspond to mapping and connecting nodes in the externalization layer. The connecting node of a subgraph is still an articulation point where all paths starting from the mapping nodes meat.

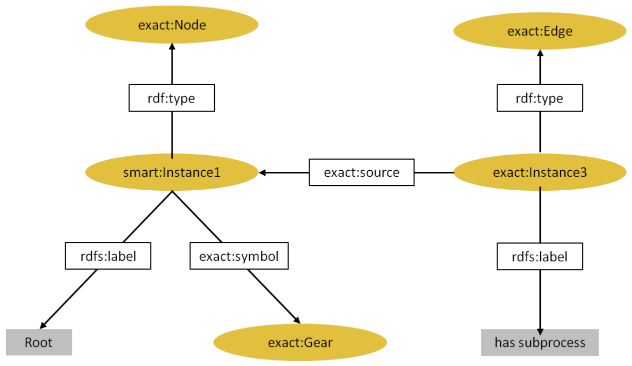


Figure 4. Extract of the externalization layer

The externalization layer describes the information paradigm that is used to communicate the explanation content. It does not provide information about the design of the semantic network nodes or the location of nodes on the drawing area (graph layout). This information is added in the presentation process. The layout of the semantic network influences the progress of the dialog and the understandability of explanation itself. In context of a keyword-based search, there are four requirements of layouting the intuitive explanation. First, it must be clear which keywords are mapped to elements of the log. Second, it must also be clear what the root or connecting element is. Third, the semantic network must not contain overlaps of edges. Finally, it must be possible to fade-in supporting nodes with respect to the third point. Fig. 5 visualizes a rendered intuitive explanation. The mapping elements are located at the very top of chart and are arranged at random. The connecting element can be found in the center at the bottom of the chart. The other nodes are positioned in such a way that no node is directly under or above another node. The remaining place can be used to fade-in supporting nodes under the base nodes without hiding information. Here, the mouse-over events can be used. In Fig. 5 the node labeled with *unsure result* may be extended with the information *corresponding field in the verifier is colored blue*. Furthermore, mouse-click events on nodes represent a suitable mean to continue the explanatory dialog. Currently, these kind of events center the corresponding node in the explanation panel showing a selection of connections to other nodes. This represents some kind of *conceptual explanations* and thus, they give an answer to *What is the meaning of..*

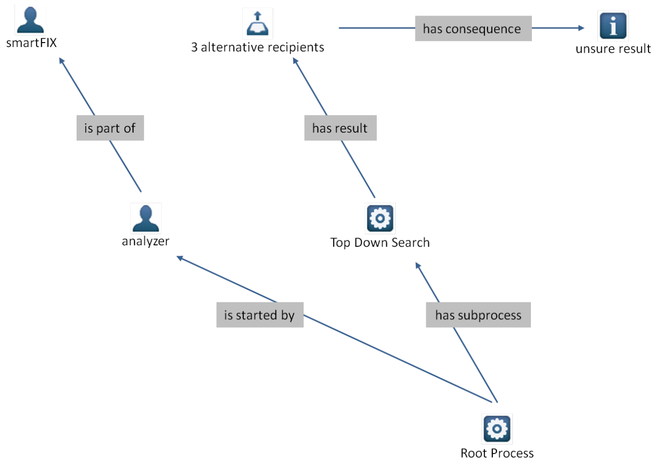


Figure 5. Visualization with explanation-aware layout

For realizing the described layout we divide the explanation panel into a matrix of cells of equal size and assign each node exactly one cell - similar to the Java CellLayout. After the assignment a check is performed whether there are overlaps of edges. If that is true the procedure starts again whereas a new random order of the mapping elements is calculated. After a certain number of iterations the entire process stops and the last assignment is used for layouting the nodes. The number of cells in both, horizontal and vertical direction depends on the number of paths between mapping and connecting nodes and the maximum number of nodes in a path. The entire procedure is described in Alg. 1. The input for the procedure is the connecting node r and the paths of the connecting subgraph p_1 to p_v. Here, the path length is equal to the number of nodes in the path and is denoted with l_1, to l_v. The variables row and col represent the number of rows and columns of the matrix, n_{ab} is node number b in path number a, $c_{ab}(x, y)$ represents a cell assignment for n_{ab} located at column x and row y, $c_r(a, b)$ belongs to r, c(d,k) is any cell, and finally CA contains all cell assignments.

Algorithm 1 Assigning cells to network nodes

```
$col := 0$
$CA = \emptyset$
$colIndex := -1$
$rowIndex := -1$
$anyIndex := -1$
$forward := true$
$col := \max(l_0 \dots l_v)$
$row := ((col - 1) * v) + 1$
$rowRoot := row - 1$
if ($t$ mod 2) == 0 then
  $colRoot := (cols - 1)/2$
else
  $colRoot := ((v + 1) * (row - 1))/2$
end if
add $c_r(colRoot, rowRoot)$ to $CA$
for $i = 0$ to ($v$-1) do
  for all $j$ such that $0 \leq j < (row - 1)$ do
    if $j < v$ then
      $node := n_{ji}$
    else
      $node := null$
    end if
    $rowIndex := rowIndex + 1$
    if forward is true then
      $colIndex := colIndex + 1$
      $anyIndex := colIndex$
    else
      $anyIndex := anyIndex - 1$
    end if
    if node != null and $c(anyIndex, rowIndex) \notin CA$ then
      add $c_{ji}(anyIndex, rowIndex)$ to $CA$
    end if
    if (colIndex + 1) == rowRoot then
      forward := false
      colInde := colIndex + 1
      exit from most inner loop
    end if
  end for
  $rowIndex := -1$
  $anyIndex := colIndex + row$
end for
```

The information about the layout, cells and corresponding nodes is content of the presentation layer. That means, in the end we have detailed description of the intuitive explanation that can be rendered by the explanation renderer that is part of the user interface. At this point we do not provide further details about the presentation layer, for example, fonts of the labels or layout of the nodes because this would exceed the scope of this work.

6 Conclusion and Outline

In this paper, we presented the generic explanation component KOIOS++ that realizes an abstract approach for intuitive explanation generation based on Semantic Technologies. In addition, we described a representative explanation problem in the information extraction system smartFIX and illustrated how intuitive explanations can be used to justify extraction results in order to make the system more transparent for users. Currently, intuitive explanations are visualized as semantic networks whereas a special layouting algorithm is used to arrange network nodes clearly and to enable suitable start for an explanatory dialog, for example, the mouse over triggers the fade-in of supporting network nodes.

In a future version of smartFIX the explanation component will not only be able to justify extraction results but also to give practical hints to avoid low quality extraction results. In addition, we provide further forms of explanation externalization, for instance, semantic networks combined with text.

ACKNOWLEDGEMENTS

This work was funded by the German Federal Ministry of Education and Research (BMBF) in the EMERGENT project under grant number 01IC10S01.

REFERENCES

[1] G. Du, M. Richter, and G.Ruhe, 'An explanation oriented dialogue approach and its application to wicked planning problems', *Computers and Artificial Intelligence*, **25**(2-3), (2006).

[2] B. Forcher, S. Agne, A. Dengel, M.Gillmann, and T. Roth-Berghofer, 'Towards understandable explanations for document analysis systems', in *Proceedings of 10th IAPR International Workshop on Document Analysis Systems*, (2012).

[3] B. Forcher, M. Sintek, T. Roth-Berghofer, and A. Dengel, 'Explanation-aware system design of the semantic search engine koios', in *Proceedings of the ECAI-10 workshop on Explanation-aware Computing (ExACt 2010)*, (2010).

[4] Andreas Fordan, 'Constraint solving over ocr graphs', in *Proceedings of the Applications of prolog 14th international conference on Web knowledge management and decision support*, INAP'01, pp. 205–216, Berlin, Heidelberg, (2003). Springer-Verlag.

[5] B. Klein, A. Dengel, and A. Fordan, 'smartfix: An adaptive system for document analysis and understanding', in *Reading and Learning -Adaptive Content Recognition*, eds., Andreas Dengel, Markus Junker, and A. Weisbecker, 166–186, Springer Publ., (3 2004). LNCS 2956.

[6] M. Liwicki, B. Forcher, P. Jaeger, and A. Dengel, 'Koios++: A query-answering system for handwritten input', in *Proceedings of 10th IAPR International Workshop on Document Analysis Systems*, (2012).

[7] Thomas R. Roth-Berghofer and Michael M. Richter, 'On explanation', *Künstliche Intelligenz*, **22**(2), 5–7, (May 2008).

[8] F. Schulz, M. Ebbecke, M. Gillmann, B. Adrian, S. Agne, and A. Dengel, 'Seizing the treasure: Transferring layout knowledge in invoice analysis', in *ICDAR-09, July 26-29, Barcelona, Spain*, pp. 848–852. IEEE, Heidelberg, (2009).

[9] B. Seidler, M. Ebbecke, and M. Gillmann, 'smartFIX Statistics – Towards Systematic Document Analysis Performance Evaluation and Optimization', in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems (DAS)*, Boston, MA, USA, (2010).

[10] W. R. Swartout and S. W. Smoliar, 'Explanation: A source of guidance for knowledge representation', in *Knowledge Representation and Organization in Machine Learning*, ed., K. Morik, 1–16, Springer, Berlin, Heidelberg, (1989).

[11] William R. Swartout, Cécile Paris, and Johanna D. Moore, 'Explanations in knowledge systems: Design for explainable expert systems', *IEEE Expert*, **6**(3), 58–64, (1991).

[12] Michael R. Wick and William B. Thompson, 'Reconstructive expert system explanation', *Artif. Intell.*, **54**(1-2), 33–70, (1992).

[13] Patricia Wright and Fraser Reid, 'Written information: Some alternatives to prose for expressing the outcomes of complex contingencies', *Journal of Applied Psychology*, **57** (**2**), 160–166, (1973).

A Brief Review of Explanation in the Semantic Web

Rakebul Hasan[1] and **Fabien Gandon**[2]

Abstract. Semantic Web applications use interconnected distributed data and inferential capabilities to compute their results. The users of Semantic Web applications might find it difficult to understand how a result is produced or how a new piece of information is derived in the process. Explanation enables users to understand the process of obtaining results. Explanation adds transparency to the process of obtaining results and enables user trust in the process. The concept of providing explanation was first introduced in expert systems and later studied in different application areas. This paper provides a brief review of existing research on explanation in the Semantic Web.

1 INTRODUCTION

Semantic Web applications use interconnected distributed data and inferential capabilities to compute their results. A user might not be able to understand how a Semantic Web application has solved a given query deriving new information and integrating information from data sources across the Web, and therefore the user might not trust the result of such a query. Semantic Web applications should provide explanations about how they obtain results in order to ensure their effectiveness and increase their user acceptance [21]. Semantic Web applications should not only provide explanations about how the answers were obtained, they should also explain and allow users to follow the flows of information between them [20].

Expert systems were among the first software systems to include explanation facilities [13, 22, 28]. Explanation facilities in expert systems have evolved from reasoning trace oriented explanations, primarily useful for developers and knowledge engineers, to more user oriented interactive explanations justifying why a system behavior is correct, to casual explanations generated in a decoupled way from the line of reasoning. The realization of the explanation facilities in expert systems were motivated by enabling transparency in problem solving, imparting an understanding of why and how a given conclusion was reached, and hence enabling trust on the reasoning capabilities of expert systems. These developments motivated adaptation and development of explanation facilities in other fields such as machine learning [10, 27], case-based reasoning [8, 26], recommender systems [29], and Semantic Web.

In this paper, we provide a brief review of the existing approaches to explanation in the Semantic Web. Our selection criterion for the reviewed works was Semantic Web applications and publications which have contribution in explanation. We have used the keyword search feature in Google Scholar[3] and also its Cited by feature for the relevant publications for discovering more publications. In addition, we have examined all the previous publications of the Explanation-aware Computing workshop series. We have also examined the previous publications of the International Semantic Web Conference series. We selected a given work if it is in the domain of the Semantic Web and it has contribution in the field of explanation. Our objective was to extract and analyze the important aspects of explanation-aware Semantic Web systems from these reviewed research, and finally to provide our perspective on these aspects. A detailed version of this paper can be found in the research report in [11].

The paper is organized as follows. In Section 2, we present the requirements and the design principles of explanation-aware Semantic Web applications. In Section 3, we provide an overview of different approaches to represent the metadata which enable support for reasoning and provenance information. In Section 4, we provide an overview of different approaches for generation and presentation of explanations. In Section 5, we focus our discussion on the important aspects of explanation approaches in the Semantic Web. In Section 6, we conclude the paper.

2 DESIGNING EXPLANATION-AWARE SEMANTIC WEB APPLICATIONS

McGuinness *et al.* [20] discuss the requirements for Semantic Web applications form an explanation perspective. The paradigm shift introduced by the Semantic Web applications, from answering queries by retrieving explicitly stored information to using inferential capabilities, generates new requirements to ensure their effective use: *"applications must provide explanation capabilities showing how results were obtained"*. McGuinness *et al.* characterize collaboration, autonomy, and the use of ontologies as the important features of Semantic Web applications from an explanation perspective. Given these features, different types of explanations, machine and human consumption of explanations, and trust are the important criteria of Semantic Web applications.

In [21], McGuinness *et al.* present requirements for distributed and portable justifications, and subsequently present justifications as user-friendly explanations. An explanation infrastructure should support knowledge provenance to allow users to understand the source information used in reasoning processes and hence enable user trust on background reasoners. Support for reasoning information should be also provided to enable users to understand what steps have been performed by reasoners. An explanation infrastructure should support generating human understandable explanation. Explanations should be presented with different degrees of detail taking into account the users' expertise and problem context.

Forcher *et al*. present the explanation-aware system design (EASD) principles in [9]. The EASD principles concern two key aspects, namely the development and the runtime of a system. The integration of explanation capabilities in a system during its devel-

[1] INRIA Sophia Antipolis – Wimmics, France, email: hasan.rakebul@inria.fr
[2] INRIA Sophia Antipolis – Wimmics, France, email: fabien.gandon@inria.fr
[3] http://scholar.google.com/

opment should not be too complicated and should not effect system performance and efficiency. Concerning the runtime aspect, the system must be aware of the explanation scenario and must be able provide explanation accordingly during its runtime. Forcher *et al*. complement the EASD principles with an abstract architecture of a multi-layered explanation model considering the main participants in any explanation scenario.

3 METADTA FOR EXPLANATION

Different approaches use different kinds of metadata for generating explanations in the context of Semantic Web. McGuinness *et al.* [19] describe explanation as "Semantic Web metadata about how results were obtained". Provenance metadata concerning information sources such as how, when, and from whom any given piece of data is obtained is an important aspect of explanation metadata. Explanations with detailed provenance information provide additional context and enable users to verify a given source of information. Metadata representing information manipulation steps and their dependencies are commonly known as justifications. Justifications facilitate rich explanation of how a conclusion was drawn. Trust related metadata is another useful aspect which enables providing explanation with integrated trust information.

In [25], McGuinness *et al.* present an explanation interlingua called Proof Markup Language (PML)[4] to represent explanation metadata. PML consists of three OWL ontologies. The PML provenance ontology (PML-P) provides primitives for representing real world things (e.g. information, documents, people) and their properties (e.g. name, creation date-time, description, owners and authors). The PML justification ontology (PML-J) provides primitives for encoding justifications for derivations of conclusions. A justification can be a logical reasoning step, or any kind of computation process, or a factual assertion or assumption. The PML trust ontology (PML-T) provides primitives for representing trust assertions concerning sources and belief assertions concerning information.

The authors in [24] introduce a restricted subset of PML constructs and tools for encoding very basic justifications and performing tasks such as retrieval and browsing of provenance information. The authors present strategies for lightweight use of PML and for simplifying encoding using PML. PML-Lite[5] is another variant of PML. It is envisioned to construct a simple subset of three PML modules. PML-Lite takes an event based modeling approach. It provides primitives to represent provenance of data flows and data manipulations.

The AIR (**A**ccountability **I**n **R**DF) [16] rule language uses the AIR Justification Ontology (AIRJ) to represent justifications produced by the AIR reasoner. AIRJ extends the PML-Lite event-based approach. The reasoning steps of AIR reasoner are interpreted as events. AIRJ provides primitives to represent the different events and the operations performed by the AIR reasoner.

4 GENERATION AND PRESENTATION OF EXPLANATION

As discussed in explanation requirements, what types of explanations are generated and how they are presented to users are important criteria for success of explanation-aware systems.

OntoNova [2] is an ontology-based question answering system in chemistry domain. OntoNova provides explanations in natural language with its answers. It generates answer justifications in a meta-inferencing step. The OntoNova inference engine produces log files which represent proof trees for answers. These files are given as an input to a second meta-inference step. This second meta-inference step explains the proof trees in natural language with the description of how answers were derived. OntoNova allows specifying meta-inference rules for the original rules for question answering. The two step method for providing explanation has advantages such as: (i) provision of additional information with explanations when proof trees do not contain enough information, (ii) filter explanation paths in case of redundancies for same results, (iii) provision of explanation with different degrees of detail, (iv) provision of personalized explanation for different contexts.

Inference Web [19, 20, 21] is an explanation infrastructure which addresses explanation requirements of web services discovery, policy engines, first order logic theorem provers, task execution, and text analytics. Information manipulation traces of these various kinds of systems are encoded as PML proofs. Inference Web provides a set of software tools and services for building, presenting, maintaining, and manipulating PML proofs. The IWBase component of Inference Web provides an interconnected network of distribute repositories of explanation related meta information. IWBase provides a registry-based solution for publishing and accessing information. Content publishers can register metadata and other supporting information such as inference rules, and inference engines. IWBase provides services to populate PML proofs. IWBase exposes the populated metadata as PML documents and provides browsing interfaces to access them. These PML documents can be also accessed by resolving their URI references. The IWSearch component of Inference Web searches for PML documents on the Web and maintains an inventory of these documents. Users can then search for PML documents using different search interfaces offered by IWSearch. Inference Web provides a browser called IWBrowser which can display PML proofs and explanations in number of different formats and styles. The rich presentations include a directed acyclic graph (DAG) view known as global view, a focused view enabling step-by-step navigation between related explanations, a filtered view to show selected parts of an explanation, an abstraction view which shows abstract views of explanations with different degrees of detail, and finally a discourse view which allows follow-up questions. The IWAbstractor component of Inference Web allows users to write abstraction patterns for PML proofs. It matches these patterns against PML proofs to provide an abstract view. Inference Web includes a general trust infrastructure called IWTrust which provides explanations with trust related information and allows filtering out unreliable information. IWTrust includes representation of trust aspects and trust computation services. In addition, it provides a browser with trust view to render the trust annotation. In the trust view, different fragments of a page are rendered in different colors depending on the trustworthiness of them. This allows users to have an understanding of the trustworthiness of rendered information just by looking at a rendered page.

The *WIQA - Web Information Quality Assessment Framework* [6] provides functionalities for quality-based information filtering. The WIQA framework allows to employ different policies for information filtering. These policies combine content-based, context-based, and rating-based quality assessment metrics. An information consumer's understanding of the employed quality assessment metrics is a major factor that influence whether the information consumer trust or distrust any quality assessment result. The WIQA framework provides detailed explanation of information filtering process for supporting information consumers in their trust decisions. It is able to provide explanation of why a given piece of information satisfies a

[4] http://tw.rpi.edu/portal/Proof_Markup_Language
[5] http://tw.rpi.edu/web/project/TAMI/PML-Lite

given WIQA-PL policy. It provides explanations in natural language for human consumption and explanations in RDF for further processing by software applications. The explanation generation process contains two steps. First, WIQA generates the parts of explanations of why constraints expressed as graph patterns are satisfied. These different parts of explanations are generated using a template mechanism. In the second step, these explanation parts are supplemented with additional explanations of why constraints expressed as extension functions are satisfied. In the RDF-based explanations, WIQA describes the explanation trees (parts and subparts of an explanation) using the Explanation (EXPL) Vocabulary[6].

The authors in [3, 4] present a nonmonotonic rule system based on defeasible logic which is able to answer queries and provide proof explanations. Defeasible logic enables reasoning with incomplete and inconsistent information. The traces of the underlying logic engine are transformed to defeasible logic proofs. The authors introduce an extension to RuleML[7], a unifying family of Web rule languages, to enable formal representation of explanations of defeasible logic reasoning. Software agents can consume and verify these proofs described using the RuleML extension. In addition, the authors present graphical user interfaces to visualize the proofs and interact with them. Finally, the authors present an agent interface to enable multi-agent systems to interact with their system.

Horridge *et al.* present two fine-grained subclasses of justifications called laconic justifications and precise justifications [15]. Laconic justifications consists of axioms that contain no superfluous part. Precise justifications are derived from laconic justifications. Each axioms of a precise justification represents a minimal part of the justification. The authors also present an optimized algorithm to compute laconic justifications showing the feasibility of computing laconic justifications and precise justifications in practice. The authors provide a Protégé ontology editor[8] plugin as a tool to compute these types of justifications[9]. This tool shows justification-based explanations of entailments. A user can select an entailment from a list of provided entailments and the tool shows the justification-based explanation for the selected entailment.

Kotowski and Bry [18] argue that explanation complements the incremental development of knowledge bases in frequently changing wiki environments. The authors present a semantic wiki called KiWi[10] which takes a rule-based inconsistency tolerant reasoning approach that has the capability of explaining how a given piece of information was derived. The reasoning approach also allows knowledge base updates in an efficient way by using reason maintenance. The authors argue that providing explanation is important for supporting users' trust and facilitates determining main causes of inconsistencies. KiWi stores the justifications of all the derivations and uses them for providing explanations and also for reason maintenance. KiWi presents explanations as natural language explanations and as tree-based explanations highlighting the derivation paths.

Forcher *et al.* [9] describe the realization of the EASD approach in a semantic search engine called KOIOS. The KOIOS semantic search engine allows keyword-based search on RDF data. The KOIOS search engine first computes a set of relevant SPARQL queries from a set of given keywords. Users then select the appropriate queries to query a triple store containing RDF data. The search results are provided with explanations about how they are computed. The explanations justify how keywords are mapped to concepts and how concepts are connected. In addition, the explanations interpret the performed queries in an understandable way. The authors introduce a set of ontologies to formally describe the content of explanations provided by KOIOS. The KOIOS Process Language (KPL) is used to describe the behavior of the problem solving process. The Mathematical Graph Language (MGL) is used to realize the graph based view of the process model. Finally, another ontology called VGL is used for visualizing graph based information. In addition, KOIOS includes a set of rules to transform a certain model described in RDF. The trace model is described in RDF and transformed step-by-step to a presentation model using a set of rules to provide different views of explanations. KOIOS provides graphical explanations for keyword-based search with graphical representation of corresponding SPARQL queries and the results of the query. Users can also get a textual explanation of any concept from the graphical representation by clicking on any concept. KOIOS also provides justification-based graphical explanation of search keyword to concept type mapping.

The AIR rule language [16] supports generation of machine consumable explanations. The AIR reasoner annotates all of its performed actions and the dependencies between these actions. These annotations are recursively converted to AIR justifications. To generate natural language explanation, rule authors can specify a natural language description in the definition of a rule itself which can contain variables. These variable values are replaced with the current value during the reasoning process. AIR also provide a feature to declaratively modify justifications. This allows the degrees of detail in AIR justifications to be selectively controlled. The authors of AIR suggest registering AIR with the Inference Web infrastructure and use their toolkit to provide explanations for human consumption.

5 DISCUSSION AND PERSPECTIVE

The research we have reviewed exposes several dimensions of explanation in the context of the Semantic Web:

Infrastructure: With the increasing growth of sharing Semantic Web data as part of the Linked Data [5] initiatives, it is important that data publishers can publish their data with explanation related metadata with ease. Explanation infrastructures should be able to accommodate common data publishing principles. Semantic Web explanation infrastructures should also address heterogeneous and distributed nature of the Web. Inference Web intent to address these issues. For example, explanation metadata can be described in PML documents and resolved using the URIs of the documents. However, Inference Web provides a centralized solution for publishing and consuming explanation metadata. Explanation metadata should be registered in the Inference Web repository to use their facilities. Moreover, the published metadata using Inference Web facilities have compatibility issues with the Linked Data principles. For instance, not all the resources in PML documents are identifiable by URIs as there are blank nodes in PML documents. Our ongoing work [12] on applying the Linked Data principles intent to address these issues. With regard to diversity of different representation, explanation metadata should be published promoting interoperability. The W3C PROV-DM data model [23] can be used as an interchange data model across different systems in this regard. Different systems can define their explanation metadata model as application-specific and domain-specific extensions of PROV-DM. Applications across the Web can then make sense of explanation metadata in a unified manner. Consumers of these explanation metadata can use explanation presentation and visualization tools according to their needs.

[6] http://www4.wiwiss.fu-berlin.de/bizer/triqlp/
[7] http://ruleml.org
[8] http://protege.stanford.edu/
[9] http://owl.cs.manchester.ac.uk/explanation/
[10] http://www.kiwi-project.eu/

Target: Human users and software agents both are target of explanation in Semantic Web applications. In the existing approaches, human users are provided with natural language explanations or graphical explanations promoting easy comprehension. Unlike the expert systems which were used by knowledgeable users and domain experts, the users of Semantic Web applications can have different background, skill level, and knowledge level because of the open nature of the Web. Level of user expertise should be taken into account while providing explanations. The presentations of explanations can change according to user expertise or user scenario context. User profiling approaches might be applied to provide explanations addressing these issues. With regard to software agents, explanations should be described using open standards such as OWL ontologies to make it possible for software agents to make sense of explanations. As pointed out previously, explanations also should be published using common data publishing principles such as Linked Data to enable external software agents to easily consume them.

What is explained: The reviewed research discusses explanation of information manipulation steps, operations, and proof trees of derived results. Additional provenance information such as how, when, and who provenance is provided in Inference Web explanations for more context and enable better understanding. Semantic Web applications use distributed interconnected data in their reasoning processes. Explaining the network of data used in the reasoning processes might be useful for users. This would enable users to understand the flow of information used in the reasoning process and have a better understanding of the data integration process performed by an application. Explanation with the details of complex computation processes might always not be as useful for non-expert users as they are for expert users. Exposing problem solving methods in certain scenarios might result in security threat. The existing research does not discuss how explanations, which expose problem solving methods, influence security and confidentiality of Semantic Web systems.

Representation: The reviewed vocabularies to represent explanation metadata for machine consumption allow to describe proof trees for answers, processes used to compute answers, different types of provenance information, models for how explanations should be presented to human users, and trust related information. Other important aspects of explanation vocabularies are granularity and existence of blank nodes in the data described using them. Table 1 presents a comparison of reviewed vocabularies taking these aspect into account. Proof trees encode logical deduction of conclusions. PML, PML-Lite, and AIRJ provide primitives to encode proof trees as justifications of answers. KOIOS and EXPL vocabularies do not provide primitives for encoding proof trees as they concern mainly describing computation process and structure of presentation information. Process description concerns describing the information manipulation steps or the algorithms that compute answers. All the reviewed vocabularies except EXPL allow describing processes used to compute answers. Provenance information provide additional context to explanations. PML allows describing how, when, and who provenance. PML-Lite allows one additional provenance, location provenance. AIRJ inherits PML-Lite provenance features. KOIOS and EXPL do not provide any primitive to describe provenance. With respect to granularity, RDF statements can be made at several level of granularity such as triple or graph. PML, PML-Lite, and AIRJ do not strictly define their granularity. The *pml:Information* class instances of PML can refer to a triple, a graph URI, or even to a textual representation of a logical formula. PML-Lite and AIRJ follow a similar approach. For instance, *pmll:outputdata* property can point to a graph represented by a set of triples. In [17], the authors introduce new AIRJ properties such as *airj:matchedgraph* to specifically make statements about graphs. The authors of KOIOS vocabularies do not provide details about granularity [9]. EXPL uses RDF reification primitives to provide a triple level fine grained granularity. Presentation model allows describing how and what should presented to the human users as explanations. KOIOS provide VGL vocabulary to describe visualization of explanations. EXPL allows describing the structure and contents of different parts of explanations that are presented to human users. PML, PML-Lite, and AIRJ do not provide any primitive to describe presentation of explanations. Declaratively specifying presentation models enables different types of user interface technologies to render explanation contents. Despite one of the main motivation for providing explanations being trust, only PML allows describing trust related information. Explanation facilities should allow to describe, capture and processing over captured trust. As a Linked Data common practice, blank nodes are avoided while publishing data [14]. Blank nodes add additional complexities in data integration in a global dataspaces. It is not possible to make statements about blank nodes as they do not have identifiers. PML and AIRJ use RDF container concepts such as *NodeSetList*. RDF containers use blank nodes to connect a sequence of items [1]. This approach makes it difficult to publish the data described using PML and AIRJ as Linked Data. In our ongoing work in [12], we present the Ratio4TA[11] vocabulary to represent justifications supporting graph level granularity without using RDF containers. Graph level granularity gives a flexible control as a graph can contain a single triple or many triples. We define Ratio4TA as an extension of W3C PROV-DM to enable better interoperability.

	PML	PML-Lite	AIRJ	KOIOS	EXPL
Proof tree	Yes	Yes	Yes	No	No
Process description	Yes	Yes	Yes	Yes	No
Provenance	How, when, who	How, when, who, where	How, when, who, where	N/A	No
Granularity	Not strictly defined	Not strictly defined	Coarse grained /graph	N/A	Fine grained /triple
Presentation model	No	No	No	Yes	Yes
Trust information	Yes	No	No	No	No
Blank node	Yes	No	Yes	N/A	No

Table 1. Comparison of explanation vocabularies

Presentation: Explanations are presented to human users as natural language or as graphical explanation in the reviewed approaches. Different kinds of graphical representation have been used to present proof trees, queries, or the steps performed by information manipulation algorithms. As discussed previously, users with different levels of expertise should be considered for Semantic Web applications. How to present complex information manipulation processes to the end users in an understandable way and how much details is useful in the context of Semantic Web need to be researched more. The EASD approach discusses providing context dependent explanations. Existing approaches such as [7] on context-aware data consumption can

[11] http://ns.inria.fr/ratio4ta/

be also applied to address providing different types of explanations depending on different types of users. Presentation models can be declaratively defined and associated with different context related information. Different user interface rendering methodologies can then be applied on the defined presentation models to provide the final explanation user interfaces.

Interaction: Explanations should be provided with navigation support to enable end users to discover more related information. Other interaction models such as follow up or feedback mechanisms might also be useful in certain contexts. Inference Web provides explanation with navigation and follow up support. How users can interact trust information based on the provided explanations would be another interesting area to explore.

Trust: The reviewed research lacks studies about understanding how explanations influence user trust in the context of Semantic Web. In contrast to the expert systems, Semantic Web applications have new dimensions such as openness and distributed. Furthermore, Semantic Web applications have much broader and diverse user base than expert systems. How these aspects of Semantic Web influence trust needs to be studied more. In the existing work, only Inference Web provides an infrastructure for trust which includes a vocabulary to describe trust related information. However, how users trust can be captured and processed for further trust assertions is not discussed. Another interesting area to explore would be how explanation approaches can be applied to explain trust itself. For instance, explanation can be provided about who have trusted a given resource or about trust rating calculations.

6 CONCLUSION

In this paper, we have presented an overview of the design principles of explanation-aware Semantic Web systems, representation and usage of explanation related metadata, and finally how explanations are generated and presented to end users. We have also presented a discussion on the important aspects of ongoing research relating to explanation in the context of Semantic Web.

ACKNOWLEDGEMENTS

The work presented in this paper is supported by the CONTINT programme of French National Agency for Research (ANR) under the Kolflow project (ANR-2010-CORD-021-02).

REFERENCES

[1] P. Hayes and B. McBride, eds., ‘RDF semantics’, *W3C recommendation*, (2004).

[2] J. Angele, E. Moench, H. Oppermann, S. Staab, and D. Wenke, ‘Ontology-based query and answering in chemistry: Ontonova project halo’, in *The Semantic Web - ISWC 2003*, eds., D. Fensel, K. Sycara, and J. Mylopoulos, volume 2870 of *Lecture Notes in Computer Science*, 913–928, Springer Berlin / Heidelberg, (2003).

[3] G. Antoniou, A. Bikakis, N. Dimaresis, M. Genetzakis, G. Georgalis, G. Governatori, E. Karouzaki, N. Kazepis, D. Kosmadakis, M. Kritsotakis, G. Lilis, A. Papadogiannakis, P. Pediaditis, C. Terzakis, R. Theodosaki, and D. Zeginis, ‘Proof explanation for the semantic web using defeasible logic’, in *Knowledge Science, Engineering and Management*, eds., Z. Zhang and J. Siekmann, volume 4798 of *Lecture Notes in Computer Science*, 186–197, Springer Berlin / Heidelberg, (2007).

[4] N. Bassiliades, G. Antoniou, and G. Governatori, ‘Proof explanation in the DR-DEVICE system’, in *Proc. of 1st Int’l Conference on Web Reasoning and Rule Systems*, pp. 249–258. Springer, (2007).

[5] T. Berners-Lee. Linked data. W3C Design Issues `http://www.w3.org/DesignIssues/LinkedData.html`, 2006.

[6] C. Bizer, *Quality-Driven Information Filtering in the Context of Web-Based Information Systems*, Ph.D. dissertation, Freie Universität Berlin, Universitätsbibliothek, 2007.

[7] L. Costabello, ‘DC proposal: PRISSMA, towards mobile adaptive presentation of the web of data’, in *The Semantic Web ISWC 2011*, eds., L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, volume 7032 of *Lecture Notes in Computer Science*, 269–276, Springer Berlin / Heidelberg, (2011).

[8] D. Doyle, A. Tsymbal, and P. Cunningham, ‘A review of explanation and explanation in case-based reasoning’, Technical Report TCD-CS-2003-41, Trinity College Dublin, (2003).

[9] B. Forcher, M. Sintek, T. Roth-Berghofer, and A. Dengel, ‘Explanation-aware system design of the semantic search engine koios’, in *Proc. of the the 5th Int’l. Workshop on Explanation-aware Computing*, (2010).

[10] A. Glass, *Explanation of Adaptive Systems*, Ph.D. dissertation, Stanford University, 2011.

[11] R. Hasan and F. Gandon, ‘Explanation in the Semantic Web: a survey of the state of the art’, Research Report RR-7974, INRIA, (2012).

[12] R. Hasan and F. Gandon, ‘Linking justifications in the collaborative semantic web applications’, in *Proc. of the 21st Int’l Conference Companion on World Wide Web*, WWW ’12 Companion, pp. 1083–1090. ACM, (2012).

[13] S.R. Haynes, *Explanation in Information Systems: A Design Rationale Approach*, Ph.D. dissertation, The London School of Economics, 2001.

[14] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, Morgan & Claypool, 1st edn., 2011.

[15] M. Horridge, B. Parsia, and U. Sattler, ‘Laconic and precise justifications in OWL’, in *Proc. of the 7th Int’l Conference on the Semantic Web*, ISWC ’08, pp. 323–338. Springer-Verlag, (2008).

[16] L. Kagal, I. Jacobi, and A. Khandelwal, ‘Gasping for AIR – why we need linked rules and justifications on the semantic web’, Technical Report MIT-CSAIL-TR-2011-023, MIT, (2011).

[17] A. Khandelwal, L. Ding, I. Jacobi, L. Kagal, and D.L. McGuinness. PML based AIR justification. `http://tw.rpi.edu/proj/tami/PML_Based_AIR_Justification`, 2011.

[18] J. Kotowski and F. Bry, ‘A perfect match for reasoning, explanation and reason maintenance: OWL 2 RL and semantic wikis’, in *Proc. of 5th Semantic Wiki Workshop*, (2010).

[19] D.L. McGuinness, Li Ding, A. Glass, C. Chang, H. Zeng, and V. Furtado, ‘Explanation interfaces for the semantic web: Issues and models’, in *Proc. of the 3rd Int’l Semantic Web User Interaction Workshop*, (2006).

[20] D.L. McGuinness, V. Furtado, P. Pinheiro da Silva, L. Ding, A. Glass, and C. Chang, ‘Explaining semantic web applications.’, in *Semantic Web Engineering in the Knowledge Society*, (2008).

[21] D.L. McGuinness and P Pinheiro da Silva, ‘Explaining answers from the semantic web: the inference web approach’, *Web Semantics: Science, Services and Agents on the World Wide Web*, **1**(4), 397 – 413, (2004).

[22] J.D. Moore and W.R. Swartout, ‘Explanation in expert systemss: A survey’, Research Report ISI/RR-88-228, University of Southern California, (1988).

[23] L. Moreau and P. Missier, ‘PROV-DM: The PROV data model’, *World Wide Web Consortium, Fourth Public Working Draft*, (2012).

[24] P. Pinheiro da Silva, D.L. McGuinness, N. Del Rio, and L. Ding, ‘Inference web in action: Lightweight use of the proof markup language’, in *Proc. of the 7th Int’l Semantic Web Conference*, ISWC ’08, pp. 847–860, (2008).

[25] P. Pinheiro da Silva, D.L. McGuinness, and R. Fikes, ‘A proof markup language for semantic web services’, *Information Systems*, **31**(4-5), 381–395, (2006).

[26] T. Roth-Berghofer, ‘Explanations and case-based reasoning: Foundational issues’, in *Advances in Case-Based Reasoning*, eds., P. Funk and P.A.G. Calero, pp. 389–403. Springer-Verlag, (2004).

[27] S. Stumpf, V. Rajaram, L. Li, M. Burnett, T. Dietterich, E. Sullivan, R. Drummond, and J. Herlocker, ‘Toward harnessing user feedback for machine learning’, in *Proc. of the 12th international conference on Intelligent user interfaces*, IUI ’07, pp. 82–91. ACM, (2007).

[28] W. Swartout, C. Paris, and J. Moore, ‘Explanations in knowledge systems: design for explainable expert systems’, *IEEE Expert*, **6**(3), 58 –64, (1991).

[29] N. Tintarev and J. Masthoff, ‘A survey of explanations in recommender systems’, *ICDE07 Workshop on Recommender Systems and Intelligent User Interfaces*, (2007).

Enhancing the explanation capabilities of the CBR-WIMS framework for the intelligent monitoring of Business workflows

Stelios Kapetanakis[1] and **Miltos Petridis**[2]

Abstract. This paper presents recent enhancements to the CBR-WIMS framework towards the enhancement of its explanation capabilities. The paper presents the CBR-WIMS approach to the monitoring of business workflows using Case-based Reasoning (CBR). This approach re-uses knowledge from previous known business workflow executions by identifying similarity between event traces, taken from workflow executions. The case representation and similarity measures are presented along with the Case-based process for retrieval of similar traces. The explanation features of the CBR-WIMS framework are illustrated through two application case studies. A new CBR-WIMS enhancement providing additional explanation is shown. The enhancement involves the use of clustering algorithms and tagging of clusters. This supports contextual knowledge that helps to provide partial explanation to retrieved similar cases, as well as an insight into possible action recommendations. Finally, the provenance of cases and its use within CBR-WIMS to establish the context of particular retrieved cases is explained. The evaluation of the explanation capabilities of CBR-WIMS is presented through two real application case studies. The results are discussed and further work planned to enhance further the explanation capabilities of CBR-WIMS is presented.

1 INTRODUCTION

Modern world operations nowadays are being organised in terms of business workflows. This allows the efficient management of their resources towards the fulfilment of given objectives. Business workflows are progressively and increasingly defined, orchestrated, organised and monitored electronically via software systems. This promotes their effective management and monitoring as well as illustrating a challenge to the relevant stakeholders. The given challenge is whether their monitoring can be effective enough to allow authorised auditors to apply remedial actions to problematic situations as well as ensure their smooth operation in any other occasion.

Several standards in business process representation have been developed over the latest years, covering effectively the business processes' definition, orchestration and choreography. For example the Business Process Modelling Notation (BPMN) developed by the Business Process Management Initiative (BPMI) and Object Management Group (OMG), provides a standard for the graphical representation of workflow-based business processes [1]. A number of relevant standards has also been developed and accepted by enterprise technologies based on Service Oriented Architecture (SOA). WS-BPEL, proposed by OASIS, is an execution language describing the "behaviour of business processes in a standards-based environment" [2]. The XML Process Definition Language (XPDL) provided by the Workflow Management Coalition (WfMC) [3] offers a standardised format for business process definitions exchange among vendors.

The usage of standardised graphical representations, for the definition of a business process, allows its stakeholders to realise its context given in terms of a UML or a BPMN diagram. Such diagrams make the business workflows more readable to humans as they have been designed to do so.

Existing representation standards allow the readability and comprehension of a business process from its relevant experts. Business workflows usually produce large numbers of data in terms of logs; storing temporal information in the form of event sequences. Most workflow logs nowadays are relatively well structured and their text can be easily understood by humans [4]. However, since the data production is significantly high the mental capabilities of a human are not enough to deal efficiently with them. Additionally a specialised workflow structure may deprive an expert from monitoring the current workflow state effectively. This could be due to the given complexity of specific process parts as represented in the workflow log. In addition to the above problems the nature of temporal information can make difficult the role of a human expert. Event sequences with complicated or overlapped temporal relationships are difficult to be monitored. Furthermore for their effective monitoring a combination of more than one log may be needed in accordance with possible association to events that may not be captured [12].

The contextual knowledge regarding a workflow execution can also be either incomplete or uncertain [12]. This can be due to a number of factors with the most prominent one the manual interference of a manager in order to confront any unanticipated situation faced within the business process execution. This can be a common case in business processes that deal extensively with human roles. Changes are not necessarily related with a specific part of a process, they might involve different processes within an organisation or even across collaborative organisations something that increases the complexity overall. Given the above challenges the monitoring of business processes can be rather difficult for humans. Therefore a need for intelligent monitoring is being formulated via software systems. Software systems while monitoring workflows they should be able to provide insights of the actual process to human auditors. Explanation should also be provided in order to allow people involved in business workflows to identify, understand and act on any issues that can be identified

[1] School of Computing and Mathematical Sciences, University of Greenwich, Maritime Greenwich Campus, Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, UK, email: s.kapetanakis@gre.ac.uk
[2] School of Computing, Engineering and Mathematics, University of Brighton, Moulsecoomb Campus, Lewes road, Brighton BN2 4GJ, UK, email: m.petridis@brighton.ac.uk

within the workflow execution. However, due to the complexity and uncertainty involved in such problems users will accept and take advice of a software system only when adequate explanation is provided. This explanation should give an insight into the reasoning and the context behind any problem identified as well as behind any proposed action. Remedial actions on the identified retrieved cases should also be provided but that also requires the provision of contextual explanation so that workflow managers can decide on the action proposed.

This paper presents work that enhances the explanation abilities of a business monitoring workflow system with the application of Cased-Based Reasoning (CBR) [13]. When applying CBR in monitoring workflows there is no attempt to build an explicit model of the knowledge associated with the monitoring of workflows. However, CBR is based on the fundamental premise that similar problems have similar solutions. In an investigated case the closest similar neighbours are being retrieved from the case base. These neighbours are being identified by applying similarity measures based usually in their similarity numerical measures. However, temporal event sequences state temporal complexity and as a result their similarity is based on structural patterns which afterwards they have to be explained. An intelligent system that applies similarity measures should show and explain to the user why the system has retrieved specific cases as the most similar, as well as how these cases may be used as solutions to the investigated case. In systems that deal with temporal and organisational complexity, providing efficient and reliable explanation can be a challenge.

This paper presents an enhanced approach to provide effective explanation while attempting CBR monitoring of business workflows. The approach is based on previous work that is briefly summarised here. The enhanced approach is being evaluated on two business process workflow case studies. Section 2 presents the application of CBR on workflow monitoring cases. Section 3 relates to workflow similarity measures, the event cases representation and the similarity calculation among workflow instances. Section 4 discusses on the explanation provision of a framework developed for the intelligent monitoring of workflows. Section 5 presents the evaluation results of the proposed approach as applied on two real workflow case studies.

2 CASE-BASED REASONING FOR BUSINESS WORKFLOW MONITORING

Human managers when monitoring a business process seem to resort to past available experience in order to be able to understand and act on the current state of the workflow. Case-based Reasoning (CBR) has been proposed as a natural approach to the humans' approach; by recalling, reusing and adapting workflows and the knowledge associated with their structure.

Minor et al [5] proposed a CBR approach to the reuse and adaptation of agile workflows based on a graph representation of workflows and structural similarity measures. Dijkman et al [6] have investigated algorithms for defining similarities between business processes focused on tasks and control flow relationships between tasks. Van der Aalst et al [7] compare process models based on observed behaviour in the context of Petri nets. The definition of similarity measures for structured representations of cases in CBR has been proposed [8] and applied to many real life applications requiring reuse of domain knowledge associated cases that require complex structural representation [9],[10].

2.1 The CBR-WIMS approach

An approach for the intelligent monitoring of business workflows using CBR has been proposed and has been shown to be able to monitor effectively real business workflows when compared to human domain experts [11]. This approach based on the CBR-WIMS software framework has shown to be able to perform well on uncertain aspects of business workflows when applied to a real workflow monitoring problem. The approach applies the standard CBR methodology relying on the definition of similarity measures between cases; informed from knowledge discovery of norms and known problems from past operation. Cases are being represented as graphs, made up of recorded workflow events and their temporal relationships. Similarity measures are based on the use of the Maximum Common Sub-graph (MCS) between graphs representing known past or current evolving workflow cases. As in many workflow monitoring applications, the exact time stamp of events and actions on a system is recorded, the problem of calculating the similarity measures is simplified as the graphs are "flattened" to traces of events of known time stamps and intervals (Fig. 1).

The Maximum Common Sub-graph (MCS) similarity between two such graphs can be defined as:

$$S(G,G') = \frac{(\sum_{\substack{matches\\ C,C'\\ in\\ MCSG}} \sigma(C,C'))^2}{count(G).count(G')} \qquad (1)$$

where count(G) represents the number of edges in graph G and σ(C,C') is the similarity measure, 0≤ σ(C,C') ≤ 1, between two individual edges (intervals or events) C and C'.

The CBR-WIMS architecture and framework is developed to provide seamless integration between the CBR system and a Business process workflow management system. This architecture deals with the definition and orchestration of business processes and provides services for the monitoring and control of workflows. This can enable the intelligent management of business processes within an organisation.

CBR-WIMS has been tested on a real application of workflow monitoring and has shown to perform well when compared to human managers of the workflow process [11]. It has also shown that it can assist managers by providing them early warnings. As CBR does not use an explicit model of the reasoning mechanism, it can be difficult to gain acceptance; especially when dealing with complex, uncertain operations involving humans that can be affected by events that are not necessarily recorded by the workflow monitoring system. For example, events such as a human error and/or misunderstanding between actors of the workflow can usually cause problems in the workflow execution but may not be explicitly recorded by the system. It is therefore important that any intelligent monitoring system can provide some explanation and context to its users.

In work previously presented at EXACT-2010 [12], it was shown that CBR-WIMS can provide some useful context and explanation by visualising the components of similarity between event trace cases among the target and its most similar retrieved cases.

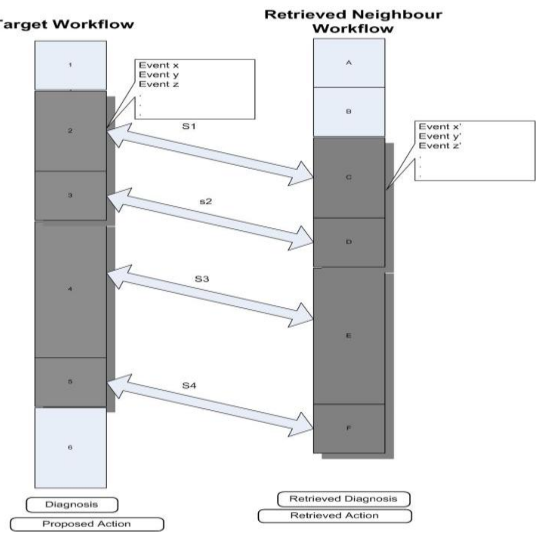


Fig. 1. Visualising the similarity between workflows

However, this visualisation similarity is just structural and contains no information from the context of particular cases that could also be available in the system. The extension of the MCS in CBR-WIMS, to allow for multiple similar sub-graph regions between cases, has also required some substantial re-working of the CBR-WIMS similarity visualisation capabilities.

This paper presents new work on the explanation capabilities of CBR-WIMS concentrating mainly to:

- Extending the visualisation capabilities of similarity measures on multiple common sub-regions with the Maximum Common Sub-graph.
- Providing context to the retrieved similar cases by the use of clustering and expert user annotation of clusters with contextual knowledge in the form of annotations, "stories" and "motifs".
- Adding provenance of retrieved cases to enhance the explanation capabilities of the CBR-WIMS system.
- Testing and evaluating the enhanced explanation capabilities of the system on two case studies. The first case study is the exam monitoring system case study used in previous work with CBR-WIMS [11, 12, 14]. The second case study relates to a Box Tracing System (BTS) that adds to the temporal complexity of the problem by providing the complexity of tracing physical items within the application of workflows. This type of a problem has extra implications as even more errors can occur outside the scope of the workflow monitoring system adding to the overall uncertainty present in the system.

3 ENHANCED EXPLANATION IN CBR-WIMS

In order to evaluate the system performance and extended explanation capabilities of CBR-WIMS, data from two real application case studies were used. Both case studies contain data from a substantial number of years of workflow execution. The research team had also access to system experts and practitioners that allowed a detailed examination of both the accuracy and effectiveness of the intelligent prediction aspects of the CBR system as well as the explanation capabilities of the new system.

This first case study was the University of Greenwich, School of Computing and Mathematical Science exam moderation system (EMS) which has been already used in a number of experiments [11, 12, 14]. The second application case study is an on-line archive management system named Box Tracking System (BTS) [16] which deals with the transfer and archiving of boxes between the BTS warehouse and its customer premises.

CBR deals with cases that have taken place in the past and work as an experience repository for the presence. These cases may or may not fit in into a rule-based model. Since in CBR there is no underlying model, it is very important that human managers are able to understand the relevance of similar cases, when attempting to provide explanation; therefore explaining the similarity measures and their derivation is an important part of the explanation challenge in this context. As cases are represented as graphs in CBR-WIMS, the similarity measures on cases' temporal information can be shown in terms of graph similarity. In particular, the Maximum Common Sub-graph can be visualised easily and highlight the parts of two graphs where there is a match. These usually contain specific motifs (patterns) that their presence can signify a specific type of a problem. This can by itself be a source of further explanation, especially for more experienced practitioners. As a result of the above, the challenge of providing explanation is reduced to a large extent to the task of showing the relevance / similarity from a case to another one(structural similarity).

In order to do that the structural similarity between cases has to be shown as part of the event sequence and not just in terms of cumulative numbers and their differences. The fact that in many cases similarity is associated with recurring types of problems, extra information could pertain to some classification of those. Problems that occur in a investigated workflow process can be explained through the identification of clustering of cases related to particular types of recurring problems. As a result a system that provides explanation should be able to possibly show the similarity in a graphical way and allow workflow experts to investigate and "drill down" further in the similarity. In addition, the membership of close neighbours within particular clusters could provide added contextual explanation. For example if most of the neighbours retrieved by the kNN algorithm are members of a particular cluster, any knowledge about the types of problems associated with the specific cluster can be used to provide additional explanation. This can enhance the explanation provided regarding the case and any course of action that may have been retrieved from the neighbouring cases.

In order to provide this enhanced explanation, we investigated the semantics that are associated with specific clusters of cases. This was achieved by observing experts on particular case specific patterns, "motifs", that can actually indicate the presence of a specific problem in the execution case. The idea of "motifs" to indicate the presence of specific patterns in data mining has been proposed by Wu et al [15]. Typical examples of the above in the exam moderation system workflows are frequent misunderstandings between roles. People occupying multiple roles can often act on the system using the wrong role for a specific action. Further analysis showed that specific motifs (sequences of events) could indicate these problems. Expert managers could give names to such motifs describing in generic terms why something happened in such way. This was usually in terms of short stories such as "Staff member A logged on as Drafter to upload their own paper, realised this and then uploaded new version as a course coordinator". We observed that the experts could identify such motifs and then they could use this to explain and communicate the problem to a third party. As the similarity measures between cases are clearly defined in the CBR-WIMS methodology, clustering algorithms can be used to generate clusters covering key types of problems that occur in a particular workflow. An Agglomerative Hierarchical Clustering algorithm was applied to both case studies [14]. The number of clusters was varied to allow for the cluster sizes that best describe known problems.

The clusters were shown to workflow specific expert practitioners. The experts could usually offer narrative hints for each cluster. The approach taken was to identify individual clusters and investigate whether the expert can describe what has happened. What we found is that in some of the cluster tags, provided by the experts, recurring motifs were obvious. However, some others cases were clustered together but not in an obvious way. Therefore, the system can provide partial additional explanation to assist in many of the problem cases, but the coverage is not universal. If a new case comes from a particulate cluster then we can provide extra context for the user to enhance the explanation regarding a certain neighbouring case retrieved.

A final aspect followed in the current approach is provenance. As workflows involve different people and departments within an organisation, different patterns of operation may emerge. Additionally, as workflows evolve, they may change over years. In the Exams moderation case study we found that due to changes in the workflow orchestration system, particular problems that used to occur, they cannot occur anymore. This calls for maintenance of the case base and the addition of provenance of cases; adding an extra dimension to the retrieved set of similar cases. This is now included in CBR-WIMS allowing the user to drill down to the provenance of specific retrieved cases. An example in the exam moderation system is that two cases from a recent year can provide better insight when compared to a really old case from a previous version of the orchestration system.

By adding this extra layer of provenance, the experts can have a better insight of the retrieved cases, adding more to their (re)-usability. The user, given this additional layer of explanation, can finally decide what is relevant and what is not. In order to evaluate the above the Exam Moderation cases study was further investigated and the second case study from a totally different domain (BTS) was obtained. The analysis from both cases showed that enhancing explanation by tagging clusters of cases with specific expert annotations showed invariably the existence of particular motifs present in most clusters. These motifs once identified and communicated to the workflow management system users enhanced the expert's confidence towards a system's recommendation. The following section will present in more detail to the experiments conducted.

4 WORKFLOW MONITORING EXPERIMENTS AND EVALUATION

In order to evaluate the approach proposed at this paper, two real workflow systems were selected and CBR-WIMS was called to provide explanation while monitoring them. Previous work [12] has shown how CBR-WIMS can provide explanation by visualising the components of similarity between event traces among the target and retrieved cases. The provided visualisation showed that it could provide useful insights to an expert but was limited to the structural representation of the cases. As a result the information provided to a human expert could not show contextual similarity insights for the given cases.

A factor that was also affecting the monitoring was that the similarity measure had to be modified following changes applied to the business process. These modifications were equivalently affecting the applied MCS algorithm.

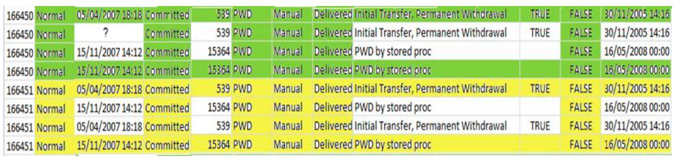
In order to overcome the above given limitations, CBR-WIMS visualisation suite has been enhanced with clustering capabilities, in order to be able to aggregate together the cases with similar characteristics. An agglomerative hierarchical clustering (AHC) algorithm was selected [14] in order to be able to identify common behavioural patterns followed by the cases. Each case in AHC was represented as a node and the algorithm clustered them based on their similarity distance to other nodes. Experts were called to analyse the resulting clusters and provide typical "stories" in terms of typical sequences of events and motifs that characterised the essence of any problems associate with the particular workflow execution. Analysis of results from using the above enhancement in the experiments conducted have shown that the users can understand better the system recommendations, based on system presented findings with or without visualisation. Users were interviewed about their confidence in the retrieved advice provided by the system. Four groups of users were investigated:

1. Users using no explanation
2. Users using structural visualisation of similarity
3. Users using annotations of clusters
4. Users looking at provenance information of retrieved similar cases

The initial experiments [12, 14] were conducted with the Exam Moderation workflows system. Further experiments were conducted with the Box Tracking system. The aim of the experiments was to investigate whether a human expert could realise in a better way what is the connection between cases as well as looking into the provenance of the retrieved cases to enhance the explanation provision.

For the BTS experiments instances of 180 box journeys were provided to the system as a case base. Each journey consisted of several events indicating the box's status, indicating a case for CBR-WIMS. The MCS was applied to all of the available cases, estimating the relevance among cases based on their similarity distance. The AHC was afterwards applied to them, constructing a cluster tree based on their similarity.

After applying case clustering the results were shown to the experts asking them to monitor 30 target cases based on their neighbour's information as extracted from the clusters. Figure 2 shows an example of the visualised information presented to experts.



166450	Normal	05/04/2007 18:18	Committed	539	PWD	Manual	Delivered	Initial Transfer, Permanent Withdrawal	TRUE	FALSE	30/11/2005 14:16
166450	Normal	?	Committed	539	PWD	Manual	Delivered	Initial Transfer, Permanent Withdrawal	TRUE	FALSE	30/11/2005 14:16
166450	Normal	15/11/2007 14:12	Committed	15364	PWD	Manual	Delivered	PWD by stored proc		FALSE	16/05/2008 00:00
166450	Normal	15/11/2007 14:12	Committed	15364	PWD	Manual	Delivered	PWD by stored proc		FALSE	16/05/2008 00:00
166451	Normal	05/04/2007 18:18	Committed	539	PWD	Manual	Delivered	Initial Transfer, Permanent Withdrawal	TRUE	FALSE	30/11/2005 14:16
166451	Normal	15/11/2007 14:12	Committed	15364	PWD	Manual	Delivered	PWD by stored proc		FALSE	16/05/2008 00:00
166451	Normal	05/04/2007 18:18	Committed	539	PWD	Manual	Delivered	Initial Transfer, Permanent Withdrawal	TRUE	FALSE	30/11/2005 14:16
166451	Normal	15/11/2007 14:12	Committed	15364	PWD	Manual	Delivered	PWD by stored proc		FALSE	16/05/2008 00:00

Fig 2. Case visualisation for BTS system

The experts after being shown the clustering results were more confident to comment on a given case regarding its status. The produced clusters were annotated based on the content of their "stories" and the behavioural patterns found. Finally, the provenance of the neighbour cases was considered and presented to the experts additionally to investigate whether this enhances their level of confidence. By knowing the provenance the levels of confidence were increased, helping in understanding better what happened in the investigated case based on the neighbours' past.

An interesting finding after looking into the provenance of the cases was that it works in a different way across systems. When looking into the provenance of the Exams Moderation System (EMS) the cases were affected from the yearly period that they were conducted, showing that cases from the same period are more likely to be of use than those from different years. Also summer resit exam cases were showing similar patterns with other past summer resits so the provenance of a retrieved case was important to the effectiveness of the CBR process. Finally, in this investigation, the provenance of a case study in terms of the academic department involved was shown to be important. As the EMS allowed to be used by departments in slightly different ways, the Mathematics department had chosen to draft exams in meetings of tutors rather than individually. This created specific temporal workflow execution patterns that precluded a set of problems from occurring, but introduced the possibility of other scenarios. In this context, the provenance of a case from that specific academic department was shown to be important to understand emerging patterns, but it was of practically no discriminatory use between the other academic departments that all followed a similar work pattern. However, in the case of BTS case the issue of provenance was more related to the company showing different behavioural patterns to company's "loyal" customers compared to other "new" ones.

Provenance has shown to be invariably important but relatively different aspects were shown to be important across systems. As provenance of retrieved cases is an extra feature available to CBR-WIMS users "drilling down" into retrieved cases, it is left naturally to the users to reason about the provenance information presented to them. Interestingly enough, the experiments conducted, showed that the gain in predictive effectiveness of the CBR system from providing the additional provenance information was modest, but when it came to filtering false positives it had an effect of nearly 50% reduction of false positives. This shows that provenance can be very useful at filtering out retrieved cases that although structurally similar to the investigated problem workflow execution, may have a different context altogether.

5 CONCLUSIONS

This paper presents an approach towards the enhanced explanation provision of monitored workflows. The investigated business workflows contain incomplete temporal data as derived from the case study systems' operational logs. Their representation has been conducted via graphs, based on their temporal relationships. This paper has shown how the CBR-WIMS framework can incorporate any investigated business workflow systems. Two case study systems have been used and CBR-WIMS had applied similarity measures between their instances using the MCS. Explanation provision has been conducted using real data from past workflow executions. The evaluation presented at this paper has shown that the adopted approach can be applied to more than one enterprise systems, enhancing the monitoring capabilities of human experts. An expert by using the hierarchical clustering can follow a "drill-down" approach aiming to extract useful system insights. As shown from the conducted experiments this can be viable and efficient, getting an idea of possible patterns of either problematic or healthy system operations. Further work on the above will concentrate on the cases' provenance as well as how this could enhance the monitoring capabilities. An investigation to another workflow system would also be sought in order to inspect the given opportunities for explanation patterns' reuse along systems.

REFERENCES

[1] Business Process Management Initiative (BPMI): BPMN 2.0: OMG Specification, January, 2011, http://www.omg.org/spec/BPMN/2.0/, accessed May 2012.P. Atkin. Performance maximisation. INMOS Technical Note 17.

[2] IBM: Business process standards, Part 1: An introduction, 2007, http://www.ibm.com/developerworks/websphere/library/techarticles/0710_fasbinder/0710_fasbinder.html, accessed May 2012.

[3] Workflow Management Coalition (WfMC): XPDL 2.1 Complete Specification (Updated Oct 10, 2008), http://www.wfmc.org/xpdl.html, accessed April 2009.

[4] Michaelis, J. R., Ding, L., McGuinness, D. L. (2009). Towards the Explanation of Workflows. In: Proceedings of the IJCAI 2009 Workshop on Explanation Aware Computing (ExaCt).Pasadena,CA, US.

[5] Minor, M., Tartakovski, A. and Bergmann, R.: Representation and Structure-Based Similarity Assessment for Agile Workflows, in Weber, R., O. and Richter, M., M.(Eds) CBR Research and Development, Proceedings of the 7th international conference on Case-Based Reasoning, ICCBR 2007, Belfast, NI, UK, August 2007, LNAI 4626, pp 224-238, Springer-Verlag, (2007).

[6] Dijkman, R.M., Dumas, M., Garcia-Banuelos, L. Graph matching algorithms for business process model similarity search. In U. Dayal, J. Eder (Eds.), Proc. of the 7th Int. conference on business process management. (LNCS, Vol. 5701, pp. 48-63). Berlin: Springer. (2009).

[7] van der Aalst, W., Alves de Medeiros, A. K., Weijters, A : Process Equivalence: Comparing two Process Models Based on Observed Behavior, In Proc. Of BPM 2006, vol 4102 of LNCS, pp 129-144, Springer, (2006).

[8] Bunke, H., Messmer, B.T.: Similarity Measures for Structured Representations. In: Wess, S., Richter, M., Althoff, K.-D. (eds) Topics is Case-Based Reasoning. LNCS, vol. 837, pp 106-118, Springer, Heidelberg (1994).

[9] Mileman, T., Knight, B., Petridis, M., Cowell, D., Ewer, J. (2002): Case-Based Retrieval of 3-D shapes for the design of metal castings in Journal of Intelligent Manufacturing, Kluwer. 13(1): 39-45; Feb 2002.

[10] Wolf, M., Petridis, M.: Measuring Similarity of Software Designs using Graph Matching for CBR, In workshop proceedings of AISEW 2008 at ECAI 2008, Patras, Greece (2008).

[11] Kapetanakis, S., Petridis, M., Knight, B., Ma, J.,Bacon, L. : A Case Based Reasoning Approach for the Monitoring of Business Workflows, 18th International Conference on Case-Based Reasoning, ICCBR 2010, Alessandria, Italy, LNAI (2010).

[12] Kapetanakis, S., Petridis, Ma, J., Bacon, L. (2010). Providing Explanations for the Intelligent Monitoring of Business Workflows Using Case-Based Reasoning. In Roth-Berghofer, T., Tintarev, N., Leake, D. B., Bahls, D. (eds.): Proceedings of the Fifth International workshop on Explanation-aware Computing ExaCt (ECAI 2010). Lisbon, Portugal.

[13] Kolodner, J. L. (1983). Reconstructive memory: A computer model. Cognitive Science. 7 (4), pp. 281-328.

[14] Kapetanakis, S., Petridis, M., Ma, J., Knight, B., Bacon, L. (2011). Enhancing Similarity Measures and Context Provision for the Intelligent Monitoring of Business Processes in CBR-WIMS. In: 19th International Conference on Case Based Reasoning, 12-15th September 2011, Greenwich, London.

[15] Wu, G., Harrigan, M., Cunningham, P. (2011). A Characterization of Wikipedia Content Based on Motifs in the Edit Graph. UCD-CSI-2011-02. School of Computer Science & Informatics, University College Dublin. Dublin, Ireland.

[16] Kapetanakis, S. (2012). Intelligent Monitoring of Business Processes using Case-based Reasoning. PhD Thesis. University of Greenwich. UK.

Using canned explanations within a mobile context engine

Christian Sauer and **Anna Kocurova** and **Dean Kramer** and **Thomas Roth-Berghofer**[1]

Abstract. Mobile applications have to adhere to many constraints. ContextEngine has been developed for the Android platform to easier deal with such limitations and situation-specific information across applications to, thus, create context-aware, mobile systems. With the increased adaptability and dynamics of context-aware applications comes an increase complexity, which in turn makes it harder to understand the behaviour of such applications. In this paper we describe how we enhanced the ContextEngine platform with explanation capabilities. Explaining can be seen as complex reasoning task on its own. Here, we focus on "canned explanations". Canned explanations are information artefacts, pre-formulated by the software engineer, that serve as explanatory artefacts stored in the system and delivered to the user on demand.

1 Introduction

Mobile applications have to adhere to many constraints such as the amount of available memory, screen size, battery, and bandwidth. ContextEngine has been developed for the Android platform to easier deal with limitations and situation-specific information across different applications and to, thus, create context-aware, mobile systems [5]. Context-aware applications are "*intelligent applications that can monitor the user's context and, in case of changes in this context, consequently adapt their behaviour in order to satisfy the user's current needs or anticipate the user's intentions*" [10].

ContextEngine uses complex rules to establish the current context of the system. These rules are acting upon implicitly collected context data without user interaction. The main reason for the results and behaviour of the system is thus hidden. The user might want to understand the results the system delivers and the behaviour it does or does not exhibit although it was expected to do so [4]. If the systems results and behaviour are not explained sufficiently to the user this lack of explanations can lead to the user mistrusting the system, use it in wrong ways or refusing to use it at all [4, 2, 9].

In this paper we examine the possibilities to provide explanations to meet the described need for explanations of the system's results and behaviours. In the next section we introduce the ContextEngine framework and some basics on explanation. In section 3 we sketch a use case scenario to motivate and illustrate our approach. We then relate our work to others in the field and describe our extension of the ContextEngine framework with explanation capabilities in section 4. The paper closes with a summary and outlook.

2 Providing context for mobile applications

ContextEngine is a framework that facilitates context definition, acquisition, and dissemination to multiple applications on a mobile device [5]. This can then be used to form context-aware applications.

Within ContextEngine, components manage specific contexts. Contexts is "*information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*" [7]. A component in ContextEngine provides one specific information about one aspect of a situation, for example the battery power left in a device. Thus these components serve to obtain raw context data from various sources, translate that data into more meaningful context information, and lastly broadcast its new state to any listener. Low level context components can be loosely aggregated to form composite components. These higher level components are defined at runtime and use rules as a way of aggregating contexts. The ContextEngine was implemented for the Google Android platform.

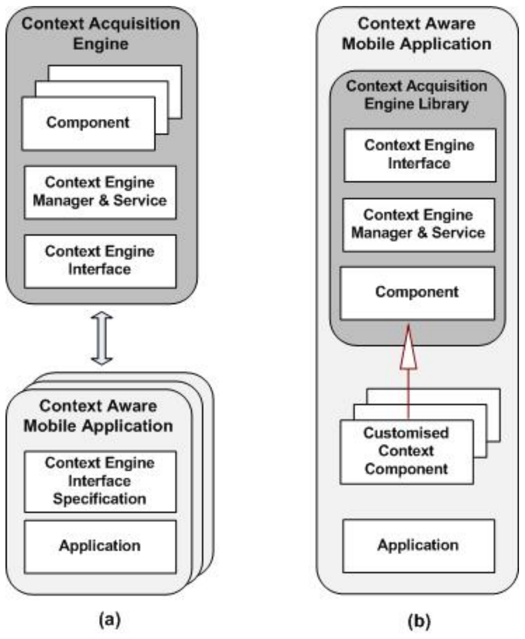

Context Acquisition Engine
Component
Context Engine Manager & Service
Context Engine Interface
Context Aware Mobile Application
Context Engine Interface Specification
Application
(a)
Context Aware Mobile Application
Context Acquisition Engine Library
Context Engine Interface
Context Engine Manager & Service
Component
Customised Context Component
Application
(b)

Figure 1. Usage Scenarios

ContextEngine has been designed for two main scenarios. The first scenario, as depicted in Fig. 1(a), shows the engine deployed once as a service on a single device for use with multiple applications. When additional contexts are needed, then the engine can be used as a single application library, as depicted in Fig. 1(b).

[1] Centre for Model-based Software Engineering and Explanation-aware Computing, School of Computing and Technology, University of West London, United Kingdom, email: {firstname.lastname}@uwl.ac.uk

2.1 Possibilities for Explanations within the ContextEngine

As soon as a user questions a system's result or behaviour, explanations are warranted. Looking at explanation goals and kinds (see, e.g., [15]) helped us guide the extension of ContextEngine. Five major goals can be achieved by providing proper explanations, aiming at:

- *Transparency*: How was a result achieved?
- *Justification*: Why was a result valid?
- *Relevance*: Why was a question by the system to its user of relevance to the systems strategy to solve a given problem?
- *Conceptualisation*: What is the meaning of a term/concept?
- *Learning*: Enhance either the user's or system's knowledge in a given domain.

To achieve these goals there are five kinds of explanations available to choose from [15]:

- *Why explanations* provide causes or justifications for facts, the occurrence or non-occurrence of events.
- *Conceptual explanations* give answers to questions of the form 'What is ... ?' and 'What is the meaning of ... ?' thus establishing links between previously unknown concepts and already known concepts within the knowledge of either the system or the user.
- *How explanations* explain processes leading to facts, states or events.
- *Cognitive* (also called *action*) *explanations* explain and/or predict the behaviour of a system.
- *Purpose Explanations*: These explain the purpose of an object or action.

Providing explanations can be seen as a complex reasoning task on its own. In this paper we focus on canned explanations [14]. Canned explanations are pre-formulated information snippets that serve as explanatory artefacts. They can be easily stored in a system and delivered to the user on demand. Canned explanations can be as simple as text strings or, in the form of URLs, point to explanatory content. The paper does not *attemp to introduce* an automatic generation of canned explanation but rather provide software engineers with the abillity to manually integrate canned explanations into their applications. The paper further examines the possibilities of composing more complex explanations based upon the basic building blocks provided by canned explanations. The composition of complex explanations is the logical step up following the composition of complex contexts already implemented in the ContextEngine.

Before we look at the extension of ContextEngine by adding the capability to store and provide basic canned explanations, we describe a use case scenario of ContextEngine.

3 Use Case Scenario

Large building construction processes involve the collaboration of a number of workers. The increasing complexity of the projects requires sophisticated solutions for collaboration and communication management between multiple participants on construction sites. Construction project managers, construction or design engineers and project architects already use mobile devices to interact with each other. By using handheld devices, collaborators can make decisions and allocate tasks right away, although they need to have tools adapted to their needs. Workflow management is a technology that can assist to manage their collaborative work process and support cooperation. Moreover, if workflows are context driven, the cooperative effort can be more dynamic, efficient and adapted for their needs. For example, knowing the current work context of fellow collaborators can assist in task allocation. The context information can depend on a number of other contexts as illustrated in Figure 2. A worker's availability depends on factors such as presence on site, location of site and current work status. If the worker is present on the construction site at a particular location and the status is set to *AVAILABLE*, the context information of the worker's *Availability* is set to *YES*.

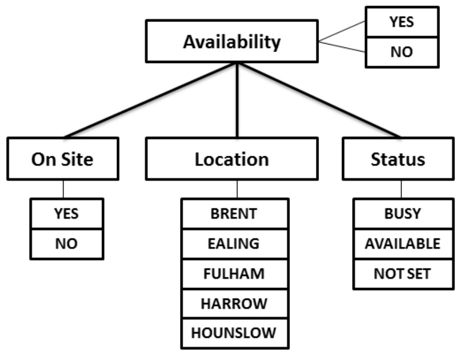


Figure 2. Context Aggregation for Collaborator's Availability

Listing 1. Context Hierarchy Definition for *Availability*

```
Listing 1:
cs = IContextsDefinition.Stub.asInterface(service);
try {
 cs.registerPreferenceComponent("On Site","STRING");
 cs.addContextValues("On Site",
   new String[]{"YES","NO"});
 ...
 cs.registerComponent("Location");
 cs.addSpecificContextValue("Location","BRENT",51.58,
     -0.29);
 cs.addSpecificContextValue("Location","EALING",51.5,
     -0.30);
 ...
 cs.registerPreferenceComponent("Status","STRING");
 cs.addContextValues("Status",new String[]{"BUSY","
    AVAILABLE","NOT SET"});
 ...
 cs.newComposite("Availability");
 cs.addPrefToComposite("On Site","Availability");
 cs.addToComposite("Location","Availability");
 cs.addPrefToComposite("Status","Availability");
 ...
 cs.addRule("Availability",
        new String[]{"YES","BRENT","BUSY"},"NO");
 cs.addRule("Availability",
        new String[]{"YES","BRENT","AVAILABLE"},"YES");
 cs.addRule("Availability",
        new String[]{"NO","EALING","NOT SET"},"NO");
 ...
 cs.startComposite("Availability");
  } catch (RemoteException re) {re.printStackTrace();}
setupContextMonitor();
```

The context hierarchy definition is illustrated in Listing 1. The context information can be acquired and aggregated by ContextEngine, which broadcasts the context information to the workflow management system running on the same device. The workflow management system does not need to know all context values because only the aggregated, final context information influences workflow execution. Although workflow execution is driven only by the aggregated context value of worker's availability (*YES* or *NO*) the workflow management system would like to obtain a more detailed explanation about the received context information. The explanation might be used by different users for various purposes. For example, a project manager might need to know why a construction engineer is not available, whether he is not present on site or just currently busy.

4 Integrating Explanations into the ContextEngine

A number of frameworks and techniques for handling context information in applications have been developed. The Context Toolkit as one of the earlier works introduced the concept of context widgets and explained how the widgets could be used to build context-enabled applications [6]. A Java-based, lightweight context-awareness framework called JCAF provided a compact Java API and a service-oriented infrastructure for the development of a wide range of context-aware applications [1]. The graphical Context Modelling Language to specify the context information has been introduced in [8]. The support for other significant characteristics of pervasive computing environments such as distribution, mobility and heterogeneity of context sources was added in the MUSIC context model [13]. Large scale scenarios have been addressed by developing shared global context models for context-aware applications such as the Nexus approach [11]. While the frameworks addressed various concepts of context management such as differentiation of lower-level raw context data from higher-level derived context information, context aggregation and context querying, there was no elaboration of adding explanations to describe the context information to the software engineer and/or end-user. This last fact we find intriguing as there is a wide variety on well-established research for the necessity of and possibility for the generation of explanations within context-aware systems.

As mentioned in section 2 there is a close relationship between context-awareness and the need for as well as the possibilities to generate explanations [9]. The need for explanations is mainly given by the black box characteristics and implicitness of the context detection and its influence on the behaviour of a system [4]. Next to this increased necessity for explanations, context aware systems also offer more possibilities to generate explanations. These possibilities are provided by the gathering and use of context knowledge, describing the context a system is in. This context knowledge can be reused to create explanations based upon it [12]. Also, explanations can be used to gather context knowledge and thus establish a context for the system based on knowledge gathered through, for example, questions posed during a user dialogue. Next to the Context knowledge being used for explanation generation itself there is also the possibility to generate explanations from the formalisation approach used by the system to store its context knowledge [3].

An important aspect within explanation-awareness is to differneciate between possible user groups and their specific goals that motivate their use and/or demand of explanations. In our work three kinds of users benefit from explanations: *software engineers* developing mobile applications, *knowledge engineers* modelling the domain knowledge for the application, and *end-users*. Software Engineers, either using another Software Engineers code or justifying their own design decisions, would benefit from why explanations with the goal to explain the purpose and use of design decisions, how explanations with the goal to use rule traces to explain how the system established its context, conceptualisations to explain the components involved in establishing of a complex context modelled as a composite component, and cognitive explanations to explain the behaviour of the system. Knowledge Engineers would benefit from conceptualisations to explain the concepts and relations of the domain the system works in and cognitive explanations to explain the interrelation of different

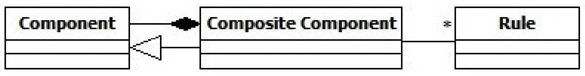


Figure 3. Main classes of ContextEngine

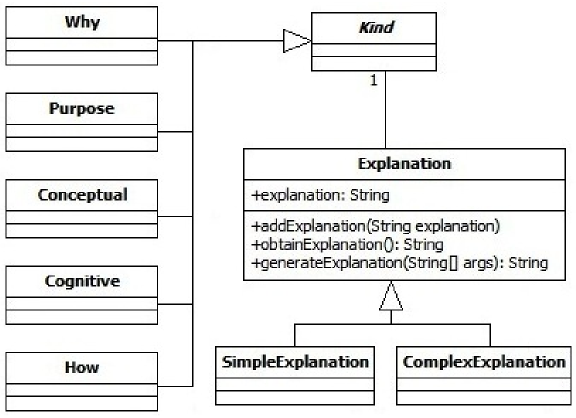

Why
Purpose
Conceptual
Cognitive
How
Kind
1
Explanation
+explanation: String
+addExplanation(String explanation)
+obtainExplanation(): String
+generateExplanation(String[] args): String
SimpleExplanation
ComplexExplanation

Figure 4. Explanation classes

system behaviours and contexts, for example justifications for the selection of rule sets. The end-user would benefit from conceptualisations to explain new system components within the system's domain, why explanations with the goal to explain the quality or expectancies of a result the system provided with the aim to provide feedback to the system.

Before we go into detail on how we integrated the kinds of explanation mapped to the users of the application we examine the implementation of ContextEngine before the integration of explanation capabilities. ContextEngine provides context knowledge using three classes: *Component*, *CompositeComponent*, and *Rule* (Fig. 3). *Component* is a self-contained class and manages a particular context. The attributes of the Component class are its name, current context value, date of context change and a set of values that it can be set to. Context information is characterised by the KEY-VALUE model complemented by date which refers to the time of the last context change. The valid context values of a component are defined by valuesSet. The basic functionalities of component include obtaining raw context data from context sources, deriving high-level context information and sending notifications about context changes to all interested parties. *CompositeComponent* manages aggregated, high-level context information. The class inherits all constructs and behaviour from the Component class. In addition, it has the ability to obtain notifications about context changes from its children and derive an aggregated context value by using rules. CompositeComponents can be constructed and assembled at runtime. Instances of the *Rule* class provide rules, based on a simple context values matching method, and are used to derive aggregated context values. The uniform rule format allows composition of different types of contexts. Rules can be defined at runtime.

To this structure we added a fourth class: *Explanation*. This class is an abstract class providing the functionalities to store and provide explanatory artefacts (Fig. 4). The Explanation class can store a string as a canned text explanation, an 'explanation snippet', or a URL that can be used to point to any resource that can serve as an explanatory artefact, for example a picture or diagram available online. To accommodate for the various kinds of explanations, the Explanation class is associated with the *Kind* class. Its functionality is inherited to five subclasses extending the Kind class. These five extension classes are implementing the five kinds of explanations: Conceptualisation, Cognitive-, Purpose-, Why- and How-explanations described in section 2.1. By extending the Explanation class with ‘Kind (of explanation)’ classes we are able to provide specific additional data and functionalities for each kind of explanation.

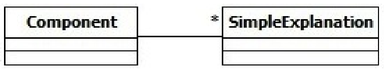


Figure 5. SimpleExplanation directly associated with Component

We distinguish two types of explanation classes: *SimpleExplanation* and *ComplexExplanation*. A simple explanation is directly associated with each context component as shown in Fig. 5. Simple explanations contain basic information concerning the particular context.

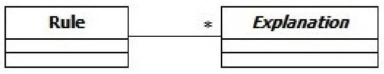


Figure 6. Explanation associated with rule

We also generate explanations by associating the Rule class with the Explanation class and thus force every rule to have a corresponding explanation (Fig. 6). The way this association is implemented is to provide simple explanations for every component involved within a rule. By doing so, we can provide simple canned explanations for every component involved in a complex context component. The association of rules with explanations also provides us with an approach to provide composed explanations. This is done by explaining the value sets of the rule and its subsequent overall result state, representing a complex context which is encoded in a composite component, being present or absent. Thus we are able to provide explanations about composite components and thus complex contexts.

4.1 Walkthrough of explanation generation

As described in section 3 we assume a use case scenario placed on a construction site and dealing with the task of establishing the availability context of a co-workers. Next to establishing, composing and communicating this context we are now also able to explain a series of facts involved in establishing said context. With the additional classes described above we are able to provide the following simple explanations as canned explanations:

- Conceptualisations for the three components: ‘On Site’, ‘Location’, ‘Status’ and the ComplexComponent ‘Availability’. For example, the sentence ‘On Site means the Person is not further away than 500 Meters from the construction site.’ explains the meaning of the term ’On Site’. Such an explanation serves the explanation goal of learning by providing information about concepts previously unknown to the user.
- Justifications for the use of exactly the three Components ‘On Site’, ‘Location’ and ‘Status’ to model the ComplexComponent ’Availability’. For example this could explain that these three criteria are minimal to describe the presence and availability of a co-worker. Such an explanation serves the explanation goal of justifying the composition of a context.
- Cognitive explanations as to why the ComplexComponent ‘Availability’ has a certain value (Yes or No) in the form of explaining the rule trace involved in establishing the final value for ‘Availability’. For example such a rule trace can generate the explanation: The co-worker is not available due to the one ore more of the Component(s) X (X = Either ‘On Site’, ‘Location’ and ‘Status’) having a negative value. Such an explanation would serve the explanation goal of transparency.

The canned explanations are designed by the knowledge engineer or the software engineer. So the explanations are integrated during the implementation of the application. The Application, using ContextEngine, can then use the stored strings or URLs to produce adequate explanations within its GUI when such an explanation is demanded by the end user. For the composition and use of complex explanations please see the following section of this paper.

4.2 Composing complex explanations

A complex context is composed of a set of simple components that are linked by one or more rules. The state of a complex context is thus represented by the result value(s) of the rule(s), based on the particular values of the rule-composing components. ContextEngine can provide complex canned explanations for complex contexts in the following two ways:

- Firstly, the explanations are pre-defined and supplied to the ContextEngine when context aggregation and it’s corresponding rules are defined.
- Secondly, the explanations can be generated at run-time.

Each particular rule and thus each complex context can be associated with canned explanations. Consider the following rule and it’s sample value-set describing the complex context ‘Availability’: R1({YES,BRENT,BUSY}, NO). The value set for the rule says that the particular worker is on site (component ‘On Site’ has the value *YES*), he is in *Brent* (component ‘Location’ has the value Brent), but the value for the atomic component ‘Status’ is currently *BUSY*, therefore the aggregated value of the rule and thus the complex context ‘Availability’ is *NO*. We can use this rule approach to associate the overall rule and thus a complex context with the following kinds of explanations:

- Conceptualisations: Explain the meaning of the complex context (rule): ‘Availability means that the worker is present at a location and is not busy’.
- Cognitive explanation: Explain the behaviour of the complex context (rule): ‘Availability will be *NO* if either he is not at the location or busy or both.’
- Purpose explanation: Explain the purpose of the complex context (rule): ‘The aggregated context ‘Availability’ represents co-worker’s availability and is used to drive workflow execution.’

An implementation of such canned why explanations for different value sets of the involved components is given by the code snippet in Listing 2 that defines three different why explanations for the value of the rule that describes the complex context ‘Availability’.

Listing 2. Explanations added to the context definition

```
cs = IContextsDefinition.Stub.asInterface(service);
try {
 cs.addRule("Availability","A–Rule1",
        new String[]{"YES","BRENT","BUSY"},"NO",
        "The worker is not available because the worker
            is present on site in Brent but is busy.","
            why" );
 cs.addRule("Availability","A–Rule2",
        new String[]{"YES","BRENT","AVAILABLE"},"YES","
            The worker is available because the worker
            is present on site in Brent and currently
            not busy.","why");
 cs.addRule("Availability","A–Rule3",
        new String[]{"NO","EALING","NOT SET"}, "NO","The
             worker is not available because the worker
            is not present on any site.","why");
 cs.startComposite("Availability");
  } catch (RemoteException re) {re.printStackTrace();}
setupContextMonitor();
```

Cognitive Explanations (based on the last child's context change that caused the change of the aggregated context value): 'The worker is available: ' + *NO* + 'Because his/her ' + *STATUS* + ' has been set to ' + *BUSY*.

Next to the approach of providing canned explanations for a complex context at whole during the design of the rules describing a complex context, the second approach ContextEngine offers is given by the possibility to compose complex explanations at runtime. This composition is based on value changes of the components that compose the complex context. Such a value change can lead to a change of the state of the overall complex context. To explain to the user, why the complex context has changed ContextEngine provides cognitive explanations based on the last child's context change that caused the change of the aggregated context value. An example for the generation of such a why explanation is given by the following snippet from the 'Availability' context: 'The worker is available: ' + *NO* + 'Because his/her ' + *STATUS* + ' has been set to ' + *BUSY*.

We also aim to compose a complex explanation by chaining the available explanatory artefacts from the components with predefined, linking, string-snippets, e.g. 'because of', 'is due to' and possibly link them in a simple logical approach relying on simple operations like AND, OR and NOT.

5 Summary and Outlook

In this paper we extended ContextEngine with explanation capabilities. We examined the approach used within ContextEngine to break up complex contexts into simpler components by describing them and then composing complex contexts using rules. The rules describe value-sets assigned to a number of defined features composing a single complex context. We also motivated the need for explanations within mobile applications. We did so by examining the goals and kinds of explanations that can be generated within an application. We demonstrated how we reused the approach of ContextEngine to assemble complex contexts from simple components to provide canned, as well as some composed explanations. An example real world use case application illustrated the enhanced ContextEngine. We focused on the provision of simple canned explanations as a beginning but also started to outline how we further plan to compose more complex explanations by reusing the approach of the composition of complex contexts employed by the ContextEngine.

For the future we are planning to diversify our view of the user even further. Whereas we are now only differentiating into Software/-Knowledge engineer and end user, we plan for the future to further diversify into experienced and novice users.

We further think about a possible extension of the composition of complex contexts within ContextEngine. Rules composing contexts and associated explanations comprise certain knowledge about a given situation. Should no rule sufficiently describe a context, i.e., the knowledge about a given situation is uncertain, a case-based approach might be successful in determining the context together with the most fitting explanation. This idea of canned explanations being cases aims at providing a more robust and versatile approach to the composition of complex contexts that can be seen as an extension method referred to, if, and only if, the rule-based approach of establishing a complex context should fail in the first place.

Acknowledgement

This work was supported by the UK Research Assessment Exercise (RAE) funded project *Explanation-aware myCBR Development*.

REFERENCES

[1] J. Bardram, 'The java context awareness framework (JCAF) - service infrastructure and programming framework for context-aware applications', in *Proceedings of the Third international conference on Pervasive Computing*, pp. 98–115, (2005).

[2] M. Bonnie, 'Trust in automation: Part i. theoretical issues in the study of trust and human intervention in automated systems', *Ergonomics*, **37**(11), 1905–1922, (1994).

[3] P. Brézillon, 'Context dynamic and explanation in contextual graphs', *Modeling and Using Context*, 94–106, (2003).

[4] A.K. Dey B.Y. Lim and D. Avrahami, 'Why and why not explanations improve the intelligibility of context-aware intelligent systems', in *Proceedings of the 27th international conference on Human factors in computing systems*, pp. 2119–2128. ACM, (2009).

[5] S. Oussena T. Clark D. Kramer, A. Kocurova and P. Komisarczuk, 'An extensible, self contained, layered approach to context acquisition', in *Proceedings of the Third International Workshop on Middleware for Pervasive Mobile and Embedded Computing*, M-MPAC '11, pp. 6:1–6:7, New York, NY, USA, (2011). ACM.

[6] A.K. Dey D. Salber and G.D. Abowd, 'The context toolkit: aiding the development of context-enabled applications', in *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pp. 434–441. ACM, (1999).

[7] P. Brown N. Davies M. Smith G. Abowd, A. Dey and P. Steggles, 'Towards a better understanding of context and context-awareness', in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pp. 304–307. Springer, (1999).

[8] K. Henricksen and J. Indulska, 'Developing context-aware pervasive computing applications: Models and approach', *Pervasive and Mobile Computing*, **2**(1), 37–64, (2006).

[9] A. Kofod-Petersen and J. Cassens, 'Explanations and context in ambient intelligent systems', *Modeling and Using Context*, 303–316, (2007).

[10] L.F. Pires L.M. Daniele, E. Silva and M. Sinderen, 'A SOA-based platform-specific framework for context-aware mobile applications', in *Enterprise Interoperability*, eds., Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, Raúl Poler, Marten Sinderen, and Raquel Sanchis, volume 38 of *Lecture Notes in Business Information Processing*, 25–37, Springer Berlin Heidelberg, (2009).

[11] N. Honle U.P. Kappeler D. Nicklas M. Grossmann, M. Bauer and T. Schwarz, 'Efficiently managing context information for large-scale scenarios', in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pp. 331–340, (2005).

[12] P. Öztürk and A. Aamodt, 'A context model for knowledge-intensive case-based reasoning', *International Journal of Human Computer Studies*, **48**, 331–356, (1998).

[13] M. Khan K. Geihs J. Lorenzo M. Valla C. Fra N. Paspallis R. Reichle, M. Wagner and G. Papadopoulos, 'A comprehensive context modeling framework for pervasive computing systems', in *Proceedings of the Eighth Conference on Distributed applications and interoperable systems*, pp. 281–295, (2008).

[14] R.P. Abelson R.C. Schank et al., *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*, volume 2, Lawrence Erlbaum Associates Hillsdale, NJ, 1977.

[15] T. Roth-Berghofer and J. Cassens, 'Mapping goals and kinds of explanations to the knowledge containers of case-based reasoning systems', *Case-Based Reasoning Research and Development*, 451–464, (2005).

Author Index