# Advanced Queries and Explanations

P.A. Bonatti, L. Ioffredo, S. Mosi, I.M. Petrova, L. Sauro
CeRICT

SPECIAL REPORT – WP4-CeRICT-1 December 8, 2019


H2020-ICT-2016-2017, ICT-18-2016
Project No. 731601

**Abstract**

The advanced query faclilities for SPECIAL's policies and compliance checking address the following two types of problems:

1. given an observed behavior, find the consent that allows that behavior;
2. explain why a given business policy does not comply with another policy.

These two problems, that answer the questions *why is this possible* and *why is this policy not compliant*, are investigated in Part I and Part II, respectively. In Part I we formally define the problem in terms of concept matching, compare the new matching problem with related works, then we analyze its computational complexity. Unfortunately, it turns out to be intractable (coNP-hard), therefore we leave a practical solution for future work. In Part II we describe the prototype of an explanation engine for non-compliance decisions and evaluate it experimentally.

# Contents

# Part I

# Why is This Possible? Finding Consent that Justifies Data Usage

### Summary of Part I

Here we formalize the notion of explanation illustrated in the BeFit scenario (deliverable D1.3) and compare it with the large body of literature on concept matching and concept unification. It turns out that such explanations – despite several similarities – differ from previous work and, moreover, the language to which they are applied, $\mathcal{PL}$ (i.e. the OWL2 profile constituting SPECIAL's policy language) is different from the languages to which matching and unification have been applied in the past. Consequently, we study the computational complexity of explanation queries against $\mathcal{PL}$. Unfortunately, it turns out that explanation queries are intractable (more precisely, NP-hard). This calls for additional investigations in order to design a practical explanation system.

## I.1 Introduction

The motivating use case based on the BeFit scenario introduced in D1.3 (Section 2.1) illustrates the potential usefulness of policy queries beyond compliance checking. In that example, a data subject (Sue) queries the transparency ledger looking for a consent statement that allows a local gym to send advertisements to her. This query, as explained in the following, differs from compliance checking enough to be generally intractable.

Sue's query over the transparency ledger amounts to looking for a consent policy where has_purpose is a subclass of TeleMarketing.[1] The other attributes of the policy – such as the data used to select the advertisement and contact Sue, storage duration, etc. – could be anything.

Let us first consider a naive approach to this kind of queries, namely, encoding the query as the incomplete policy:

$$\exists \mathsf{has\_purpose}.\mathsf{TeleMarketing}$$

that can be equivalently reformulated in OWL2's functional syntax as follows:

$$\mathsf{ObjectSomeValuesFrom}(\mathsf{has\_purpose}\ \mathsf{TeleMarketing})\,.$$

In general, this expression is not comparable with the business policy $BP$ of the gym, because:

1. $\exists$has_purpose.TeleMarketing cannot be a subclass of $BP$ since the former is missing the attributes has_data, has_processing, has_storage, and has_recipient of $BP$;

2. $BP$ is not guaranteed to be a subclass of $\exists$has_purpose.TeleMarketing, either, because its purpose might be a *superclass* of TeleMarketing (such as AnyContact); such a policy would still cover telemarketing as a special case, and should be returned by the query.

[1] Here we ignore the search for chains of data and policy transfers that may propagate consent to third parties, because we are going to prove that policy search is difficult even without data transfers.

Since neither policy is a subclass of the other, it follows that Sue's query cannot be reduced to subsumption (unlike compliance checking) and must be formalized in a different way. What Sue is actually looking for is a sort of a *pattern*, that could be expressed as:

$$\exists \mathsf{has\_purpose}.\mathsf{TeleMarketing} \sqcap \exists \mathsf{has\_data}.X \sqcap \exists \mathsf{has\_processing}.Y \\ \sqcap \exists \mathsf{has\_recipient}.Z \sqcap \exists \mathsf{has\_storage}.W\ ,$$

where $X, Y, Z$ and $W$ are *concept variables*. In informal terms, concept variables are like wildcards, so the above patters matches any policy whose purpose is a superclass of TeleMarketing. Then the problem of finding a policy $BP$ that permits telemarketing consists in looking for a value assignment to the concept variables that makes the (instantiated) pattern a subclass of $BP$.

This problem is a variant of the so-called *matching problem* for description logics, that has been extensively studied along the past 21 years. However, our variant is more complex than its older analogue.

This report is organized as follows. In Section I.2 we formalize the matching problem of our interest, and compare it with the literature on concept matching and concept unification. In Section I.3 we prove that the new matching problem is intractable in $\mathcal{PL}$ as well as the similar language $\mathcal{EL}^{\bot}$. Our conclusions are reported in Section I.4, together with some interesting directions for future work.

## I.2 Preliminaries

We start by formalizing the matching problem of our interest. The first assumption needed is that the set of concept names (classes) is partitioned into "proper" concept names and (concept) *variables*, that will be denoted with $X, Y, Z$ and $W$, possibly with subscripts and superscripts. Proper concept names, instead, will be denoted with $A$ and $B$ (again with super/subscripts) unless stated otherwise. A *pattern* is simply a class expression (concept) that contains at least one occurrence of a concept variable. Here we use the logical syntax of OWL2 because it is more compact and facilitates the formal proofs (still everything could be formulated in any of the other formats supported by OWL2).

A *substitution* (for a pattern $P$) is a function $\sigma$ that maps each concept variable (occurring in $P$) on a variable-free concept. By $\sigma(P)$ we denote the concept obtained from $P$ by replacing each variable $X$ with $\sigma(X)$. Now the matching problem can be defined as follows:

**Definition 1 (Matching problem)** *Given a pattern $P$, a variable-free concept $C$, and a knowledge base $\mathcal{K}$, is there a substitution $\sigma$ such that $\mathcal{K} \models \sigma(P) \sqsubseteq C$ and $\sigma(P)$ is consistent with $\mathcal{K}$?*

If such a $\sigma$ exists, then it is called a *solution* of the matching problem.

The consistency requirement is needed to avoid trivial, non-informative solutions $\sigma$ that map some concept variable onto an inconsistent concept (like the intersection

of two disjoint classes, such as $\mathsf{AnyData} \sqcap \mathsf{AnyPurpose}$). In $\mathcal{PL}$ (and $\mathcal{EL}$) the inconsistency would always be propagated to the entire concept $\sigma(P)$, thereby making it a subclass of *all* concepts $C$. Then, without the consistency requirement, the matching problem would be trivial (the answer would be always "yes" when $\mathcal{K}$ contains a disjointness axiom).

The matching problems most frequently investigated in the past are specular reflections of the above problem. Their goal is finding a substitution $\sigma$ such that: $\mathcal{K} \models C \sqsubseteq \sigma(P)$ [1, 13, 10, 6, 19, 4, 5, 8]. Then, in the following, we will refer to the new and the old matching problems as *left-matching* and *right-matching*, respectively, according to the position of the pattern. In right-matching, the consistency requirement is not needed because the pattern occurs on the right-hand side of the subsumption relation; more precisely, if $\sigma$ is a solution and $C$ is not inconsistent, then $\sigma(P)$ cannot be inconsistent either.

Right-matching is tractable in $\mathcal{EL}$, because its answer is positive if, and only if, the substitution $\sigma_\top$ that maps all variables on $\top$ is a solution (called *matcher* in those papers) [19]. Therefore, right-matching can be decided simply by answering the subsumption query $C \sqsubseteq \sigma_\top(D)$, which takes polynomial time in $\mathcal{EL}$.

Left-matching, instead, is almost a special case of the *unification problem* for Description Logics [5, 20, 21, 11, 22, 12, 16, 17, 18, 23, 7, 2, 15, 3, 9, 14], that consists in finding – for two given patterns $P$ and $Q$ – a *unifier*, that is, a substitution $\sigma$ such that $\mathcal{K} \models \sigma(P) \equiv \sigma(Q)$. To see the relationships between our matching problem and unification, note that $\sigma(P) \sqsubseteq C$ is equivalent to $\sigma(P) \equiv \sigma(P \sqcap C)$, so the solutions to the matching problem are also solutions of a corresponding unification problem. Additionally, left-matching requires $\sigma(P)$ to be consistent with respect to $\mathcal{K}$. In $\mathcal{EL}$ (that cannot express any kind of negation) this is automatically guaranteed. Another analogue of left-matching has been investigated in [19] under the name of *right-ground matching*. Again, the difference with left-matching is that $\sigma(P)$ is not required to be consistent w.r.t. $\mathcal{K}$.

In $\mathcal{EL}$, unification is NP-complete. In the following, we are going to prove that our matching problem is NP-complete, too, if the underlying logic is $\mathcal{PL}$ or $\mathcal{EL}^\bot$. Right-ground matching in $\mathcal{EL}$, instead, can be decided in polynomial time [19]; therefore the consistency requirement in the definition of left-matching and the negation provided by $\bot$ make the latter problem significantly harder (unless P=NP).

## I.3 The complexity of left-matching

Left-matching embodies two sources of complexity: intervals and disjointness axioms. Both embody some kind of negation that makes matching intractable. Such sources of complexity come into play even if each concept variable occurs at most once in the pattern (i.e. concept variables are essentially wildcards). We call such patterns *wildcard patterns*.

**Theorem 1** *The matching problem in $\mathcal{PL}$ is NP-hard. This result still holds if the pattern is a wildcard pattern and:*

1. *either $\mathcal{K}$ contains only functionality axioms over data properties (and interval constraints may occur in the problem instance),*
2. *or interval constraints do not occur in the problem instance and $\mathcal{K}$ contains only disjointness axioms.*

**Proof.** We prove this result by reducing 3-COLORING to matching in two slightly different ways. Let $G = (V, E)$ be any instance of 3-COLORING. The pattern in the corresponding matching problems is always:

$$P = \exists r.X_1 \sqcap \exists r.X_2 \sqcap \exists r.X_3 \,.$$

where $r$ is a (non-functional) role and the concept variables $X_1,\ X_2$, and $X_3$ represent the three colors. Moreover, for each vertex $v \in V$, we introduce a concept name $A_v$.

The first reduction addresses point 1 in the theorem's statement. This reduction additionally uses a functional data property $f_{v,w}$ for each edge $(v, w) \in E$. The knowledge base $\mathcal{K}$ contains only the axioms $\mathsf{func}(f_{v,w})$. The variable-free concept $C$ in the matching problem is the intersection of the concepts $\exists r.C_v$ (for all $v \in V$) such that

$$C_v = A_v \sqcap \big( \bigsqcap_{(v,w)\in E} \exists f_{v,w}.[0] \big) \sqcap \big( \bigsqcap_{(w,v)\in E} \exists f_{w,v}.[1] \big) \,.$$

Here the interval constraints make a concept $C_v \sqcap C_w$ inconsistent whenever $v$ and $w$ are adjacent nodes. Conversely, each intersection of concepts $C_v$ for non-adjacent nodes is satisfiable w.r.t. $\mathcal{K}$.

Given a 3-coloring, a solution $\sigma$ to the matching problem can be obtained by setting

$$\sigma(X_i) = \bigsqcap \big\{ C_v \mid v \text{ has the } i\text{-th color} \big\} \quad (i = 1, 2, 3) \,.$$

Since adjacent nodes do not have the same color, $\sigma(X_i)$ is consistent with $\mathcal{K}$ for all $i = 1, 2, 3$, and so is $\sigma(P)$. Moreover, by construction, each sub-expression $\exists r.C_v$ in $C$ subsumes the sub-expression $\exists r.X_i$ of $P$ such that $i$ is the color of $v$. It follows easily that $\mathcal{K} \models \sigma(P) \sqsubseteq C$. This proves that if the 3-COLORING instance has a solution then the matching problem has a solution, too.

Conversely, if $\sigma$ is a solution of the matching problem, then a 3-coloring can be found by coloring a node $v$ with color $i$ iff $\sigma(X_i) \sqsubseteq A_v$. If some $A_v$ did not subsume any $\sigma(X_i)$, then $\sigma(P)$ would not be a subclass of $\exists r.C_v$, and consequently not a subclass of $C$ (a contradiction). Therefore every node is colored. Moreover, since $\sigma(P)$ is consistent, $\sigma(X_i)$ cannot be a subclass of any $C_v \sqcap C_w$ such that $v$ and $w$ are adjacent; so adjacent nodes are given different colors. This completes the proof for point 1.

The reduction for point 2 replaces intervals with disjoint concepts. In particular, for each edge $(v, w) \in E$, we introduce two concept names $E_{v,w}$ and $\bar{E}_{v,w}$ and include a disjointness axiom $\mathsf{disj}(E_{v,w}, \bar{E}_{v,w})$ in $\mathcal{K}$. Concept $C$ is the intersection of the concepts $\exists r.D_v$ (for all $v \in V$) such that

$$D_v = A_v \sqcap \big( \bigsqcap_{(v,w)\in E} E_{v,w} \big) \sqcap \big( \bigsqcap_{(w,v)\in E} \bar{E}_{w,v} \big) \,.$$

Analogously with the proof for point 1, given a 3-coloring, the substitution $\sigma$ defined by

$$\sigma(X_i) = \bigsqcap \{D_v \mid v \text{ has the } i\text{-th color}\} \quad (i = 1, 2, 3)$$

is a solution of the above matching problem. Conversely, given a solution $\sigma$ of the matching problem, a 3-coloring can be found by painting a node $v$ with color $i$ iff $\sigma(X_i) \sqsubseteq A_v$. ∎

As a corollary of point 2 we obtain the intractability of matching in $\mathcal{EL}^\bot$:

**Corollary 2** *Left-matching is NP-hard in $\mathcal{EL}^\bot$, even if $\mathcal{K}$ contains only axioms of the form $A \sqcap B \sqsubseteq \bot$.*

The significance of this result is illustrated in the next section.

## I.4 Conclusions

We have formally investigated the advanced explanations needed to implement the BeFit scenario (cf. D1.3) – and more generally to find which consent permits a given operation. We provided a mathematical definition of the explanation queries (i.e. the notion of left-matching), and compared it with the ample literature that deals with similar problems.

It turns out that explanation queries are not covered by the existing literature. They are different from both the matching problem and concept unification, due to the position of the patern in the left-hand side of the inclusions, and the additional condition that the instantiated pattern should be consistent with the knowledge base. Therefore, new mechanisms are needed.

The classical matching problems are tractable for $\mathcal{EL}$, while concept unification is intractable. Unfortunately, the computational complexity of explanation queries (i.e. left-matching) is aligned with the latter: we proved that even for languages as simple as $\mathcal{PL}$ and $\mathcal{EL}^\bot$, explanation queries are intractable (NP-hard).

Given the hardness of the general left-matching problem, we have tried several restrictions to reduce its complexity. Unfortunately, the problem remains hard even if:

- only wildcards are allowed (i.e. each concept variable occurs at most once in the pattern), and
- either the knowledge base contains only functionality axioms (no subclasses nor disjointness axioms),
- or the knowledge base contains only disjointness axioms, and no interval constraints are used in the concepts.

The result on $\mathcal{EL}^\bot$ is interesting because $\mathcal{EL}^\bot$ is very close to the logic $\mathcal{EL}$ used to prove the tractability of right-matching. Therefore our result provides preliminary evidence that the additional complexity of left-matching does not essentially depend on the peculiar features of $\mathcal{PL}$– such as functional roles and interval constraints – but it really depends on the different structure of left-matching.

These complexity results suggest that more work is needed in order to construct a practical implementation of left-matching. It should first be determined whether the worst case complexity occurs frequently in real-world policies; then suitable heuristics should be designed in order to tackle the hard cases, possibly by resorting to approximate answers. This line of work is an interesting direction for future research.

Another interesting topic for future work is the precise characterization of the complexity of left-matching. We conjecture that it is NP-complete, but the proof is not trivial, since the space of possible values for a concept variable is infinite. We plan to adapt the results for $\mathcal{EL}$ that reduce such infinite space to a finite set of concepts.

# Part II

# Explaining Non-Compliance

### Summary of Part II

In SPECIAL's approach, business policies are checked for compliance w.r.t. the consent of data subjects and a partial, formal representation of the GDPR. In case of failure, it is important to explain which parts of the business policies are not adequate and cause the compliance checking to fail. It may also be useful to suggest possible corrections, when possible. Here we describe a proof-of-concept tool for constructing such explanations.

## II.1 Introduction

There are at least four scenarios in which explaining non-compliance may be useful:

1. when a business policy is checked for compliance with the regulations before deployment;
2. when the controller is designing a new process and wants to check how many users/customers have given a consent that covers the new data processing (e.g. in order to estimate the amount of opt-in's and the effort required to reach a target number of consents);
3. during an ex-post, auditing check, in case a non-authorized operation is found;
4. when a data subject poses *what if* queries to understand the consequences of consent by examples.

When compliance fails, it is important to identify the parts of the business policy that cause the compliance test to fail, and possibly suggest how to correct the business policy to make it compliant. Of course, such explanations should be as simple as possible, in order to be understood by users with no technical background.

Here we illustrate an approach based on a simple color-coding scheme. We distinguish the following situations:

1. green: the business policy is compliant;
2. yellow: the business policy is not compliant, but restricting some of its properties to a subclass makes it compliant; moreover concrete suggestions for correction can be provided, because the properties to be restricted have some subclasses in the vocabulary;
3. orange: the business policy is not compliant; restricting some of its properties could make it compliant, however no concrete suggestions for correction can be automatically provided, due to the lack of terms in the vocabulary;
4. red: the two policies are incompatible, in the sense that restricting properties cannot lead to compliance; some property must be replaced with something completely different.

The difference between yellow and orange can be further explained as follows: In both cases, the value $V_B$ of one of the properties $P$ of the business policy is not a subclass of the corresponding class $V_C$ used in the other policy, however $V_B$ and $V_C$ are not disjoint, so their intersection `ObjectIntersectionOf(`$V_B$ $V_C$`)` is consistent. If the vocabularies contain a term $V$ that is a subclass of both $V_B$ and $V_C$, then a suggestion is possible (replace $V_B$ with $V$) and the color code is yellow. Otherwise (if no subclass of $V_B$ and $V_C$ can be found in the vocabulary), then no specific term replacement can be suggested and the color code is orange.

## II.2 A proof-of-concept implementation

We have implemented the above idea for usage policies that conform to the minimal core model (MCM) introduced in D1.3. The same idea can be easily extended to the additional properties of business policies, such as legal bases and obligations, that have been identified and formalized by the DPVCG[2] while the explanation engine was being developed and tested. The explanation prototype provides a GUI for

- loading a $\mathcal{PL}$ ontology in .owl format and browsing its taxonomy (Fig. 1);
- creating and storing (in .owl format) full usage policies (Fig. 2);
- constructing explanations for the (non)-compliance of two stored policies (Figures 3–6).

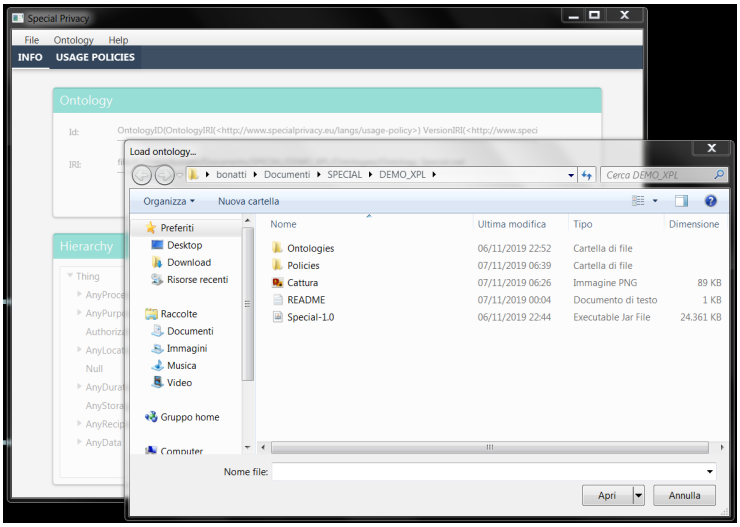


Figure 1: Ontology/vocabulary selection

[2] `https://www.w3.org/community/dpvcg/`

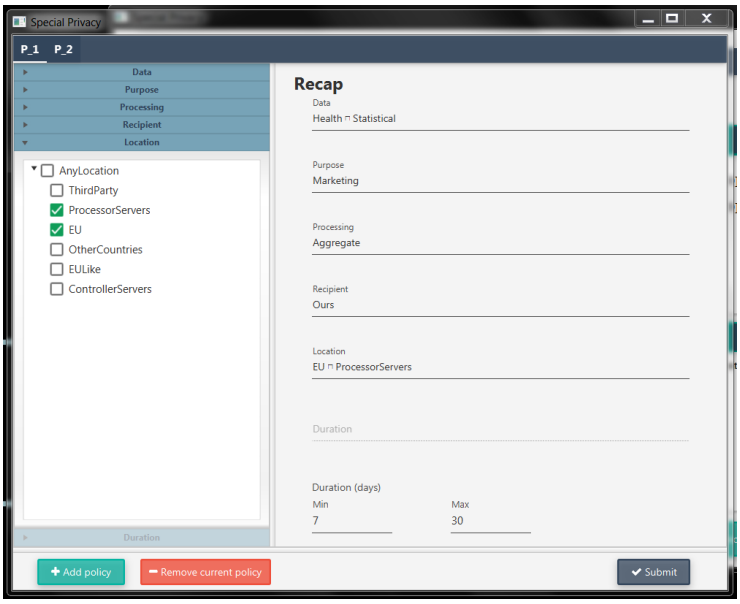


Figure 2: Policy creation interface

In particular, policies are defined by selecting for each property one or more terms. The latter can be chosen by ticking a box in the ontology's taxonomy for the particular property of interest (on the left-hand side of the window, see Fig. 2). The right-hand side of the window summarizes the current simple policy. In order to define a full policy, additional simple policies can be added with the green button. The user can move to a different simple policy by clicking on the tabs on the top of the window (e.g. tabs P_1 and P_2 in Fig. 2). The full policy can be saved in the local file system with the 'Submit' button.

The business policy and the consent policy to be compared can be loaded from the local file system with the interface illustrated in Fig. 3. Different representations of the policies can be selected with the drop-down menu on the top-left corner; currently the interface supports Manchester's syntax and the logical syntax of description logics, since this view is meant for testing and debugging purposes.

After the two policies have been selected, policy comparison can be started with the green button. The next figures illustrate some of the possible outcomes. If the business policy is compliant, then the output looks like Fig. 4. In Fig. 5, the business policy is not compliant, but it can be made compliant by restricting some of its properties (see the orange code near 'Result', that represent the overall level of compliance of the full policy). The lower part of the window provides a detailed analysis of the two simple policies visualized in the window (that are part of the full policies). For example the data category Government used in the business policy is allowed to have terms in common with the category Financial used in the consent policy, because these

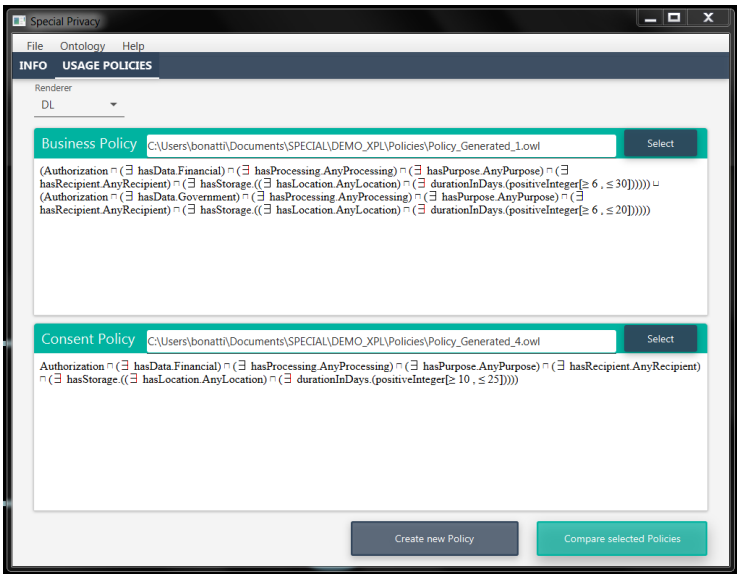


Figure 3: The interface for selecting the policies to be checked

two classes are not disjoint. However, the vocabulary does not currently contain any term in their intersection, so the system cannot suggest any concrete replacement for Government, which justifies the orange mark. The system can suggest how to correct the duration property, instead (see the yellow mark), that is, by adopting the intersection of the two durations used in the two policies. The detailed analysis for the other simple policies that belong to the full policies can be displayed by clicking on the arrows enclosed in circles (on the top of the window).

Finally, Fig. 6 illustrates a scenario where the policies are incompatible (no correction based on subclassing is possible). The recipients in the business policy are third parties (OtherRecipients), while the consent policy requires the data to remain within the borders of the controller and its processors (Ours). Moreover, the controller wants to transfer the data to third countries (OtherCountries), while the consent policy requires the data to remain within the EU. In this case the recipients and the location specified in the business policy cannot be corrected by replacing them with a subclass, since they are disjoint from the recipient and location specified in the consent policy. The only possible correction consists in replacing the business policy's recipient and location with those adopted in the consent policy (or subclasses thereof).

## II.3 The explanation algorithm

The explanation algorithm that produces the above explanations is a simple variant of the algorithm for compliance checking described in D2.8 and evaluated experimentally in D3.5. The difference is the following: When the value $V_{P,C}$ of a property $P$ of the

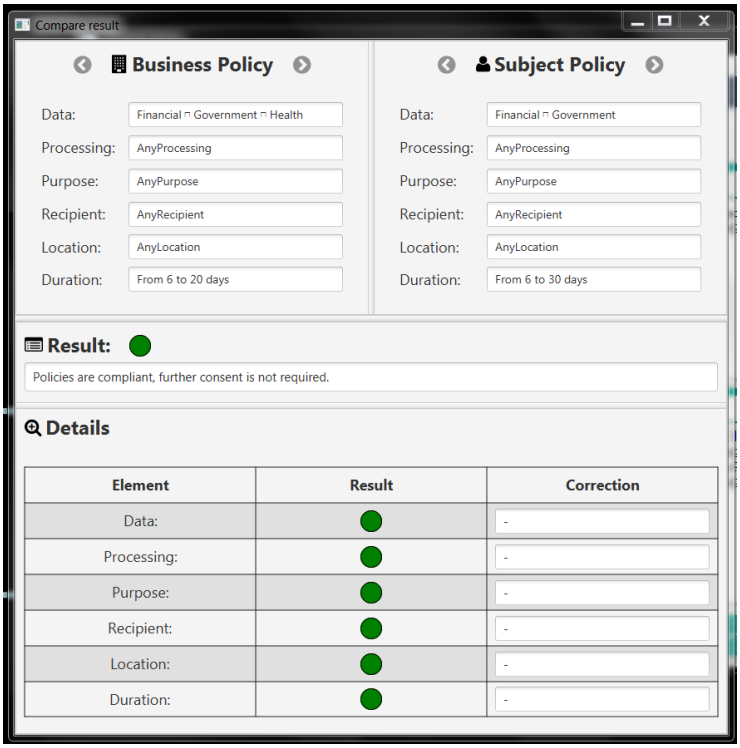


Figure 4: Compliant policies

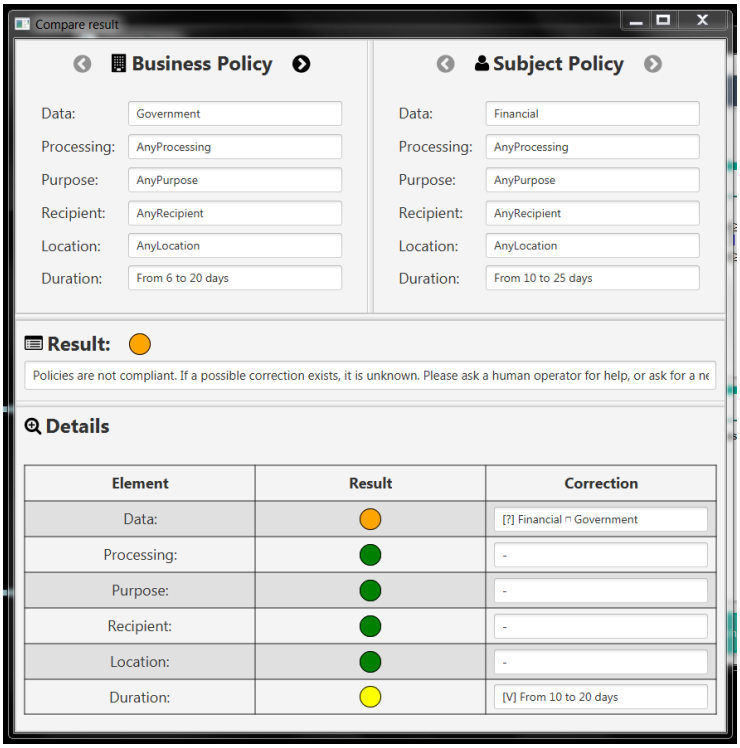


Figure 5: Non-compliant but correctable policies

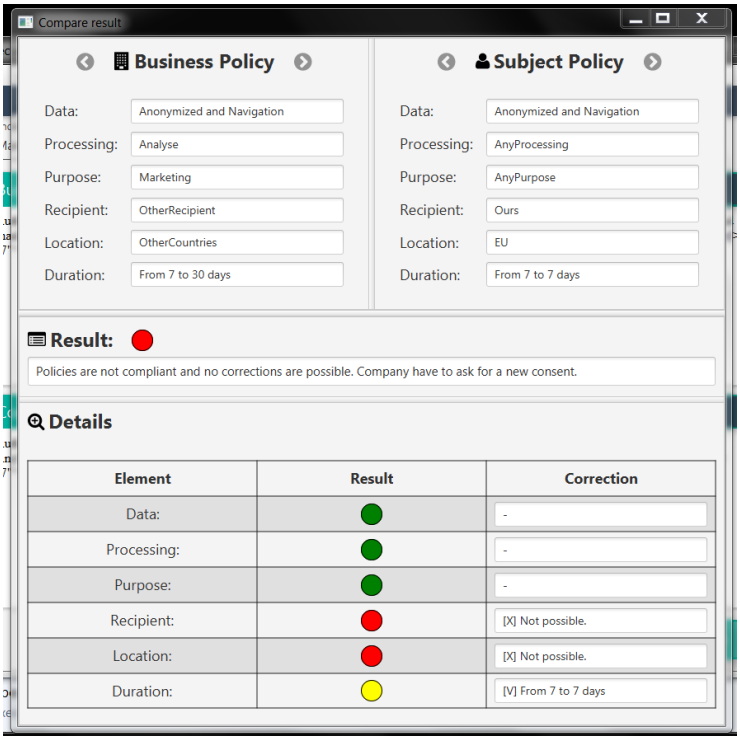


Figure 6: Strongly non-compliant policies

given consent policy $C$ (or GDPR formalization) is compared with the corresponding value $V_{P,B}$ of $P$ in the business policy $B$, then:

1. An integer $cc(P, B, C)$ (the color code of $P$ with respect to $B, C$) is associated to $P$, instead of returning *true* or *false*; the correspondence between integers and color codes is the following:

   - 1 = green
   - 2 = yellow
   - 3= orange
   - 4 = red

   More precisely, if $V_{P,B}$ is a subclass of $V_{P,C}$ then $cc(P, B, C) = 1$. Otherwise, if $V_{P,B}$ and $V_{P,C}$ are disjoint (i.e. their intersection is a subclass of owl:Nothing), then $cc(P, B, C) = 4$. If the above two cases do not apply, and there exists a class name $V$ that is a subclass of both $V_{P,B}$ and $V_{P,C}$, then $cc(P, B, C) = 2$. Otherwise $cc(P, B, C) = 3$.

2. The matching continues until all properties have been processed (instead of stopping at the first mismatch) so as to find all the discrepancies between the two policies.

The overall color code associated to an entire simple policy (i.e. without unions) is determined by taking the *maximum* value associated to its properties. To see the rationale behind this mechanisms, note that green has the minimum integer value, and the overall color can be green only if all properties are labelled with green; a single property with a higher value suffices to make $B$ non-compliant, and, accordingly, the overall code should be greater than 1. Moreover, a single property with color red (that has the maximum integer value among the color codes) suffices to make the mismatch unrecoverable (if $B$'s properties can only be restricted); accordingly the maximum integer in this case is going to be 4. The color code returned by the comparison of two simple policies $B$ and $C$ will be denoted by $cc_{ss}(B, C)$, so, according to the above definition,

$$cc_{ss}(B, C) = \max_{P} cc(P, B, C)$$

for $P \in \{\texttt{hasData}, \texttt{hasPurpose}, \texttt{hasProcessing}, \texttt{hasRecipient}, \texttt{hasLocation}, \texttt{hasDuration}\}$.

If a simple (i.e. union-free) business policy $B$ is compared with a *full* policy $C = \texttt{ObjectUnionOf}(\texttt{C}_1 \ldots \texttt{C}_\texttt{n})$, then the overall color code should be determined by taking the *minimum* of the integers $cc_{ss}(B, C_i)$ for $i = 1, \ldots, n$. Indeed, for $B$ to be compliant, it suffices that $B$ is a subclass of *at least one* $C_i$ (i.e. one green is enough), while $B$ is not compliant only if $B$ does not comply with *any* of the $C_i$. The color code returned by the comparison of a simple policies $B$ and a full policy $C$ will be denoted by

$$cc_{sf}(B, C) = \min_{1 \leqslant i \leqslant n} cc_{ss}(B, C_i)\,.$$

Finally, consider the case in which $B = \texttt{ObjectUnionOf}(\texttt{B}_1 \ldots \texttt{B}_\texttt{n})$ is a full policy, too. In this case, by a symmetric argument (with respect to the previous case), it is easy

to see that the overall color code should be determined by taking the *maximum* of the values $cc_{sf}(B_i, C)$, for $i = 1, \ldots, n$. So, for example, a single $B_i$ with color code red suffices to assign red to all of $B$. The overall numeric code for unrestricted policies will be denoted by

$$cc(B, C) = \max_{1 \leqslant i \leqslant n} cc_{sf}(B_i, C)\,.$$

We can prove the following correctness result for the above algorithm:

**Theorem 3** *Given two usage policies $B$ and $C$ and a $\mathcal{PL}$ knowledge base $\mathcal{K}$,*

1. *$cc(B, C) = 1$ (green) iff $\mathcal{K} \models B \sqsubseteq C$;*
2. *$cc(B, C) = 4$ (red) iff $\mathcal{K} \not\models B \sqsubseteq C$ and $\mathcal{K} \models \mathsf{disj}(B, C)$;*
3. *$cc(B, C) \in \{2, 3\}$ iff $\mathcal{K} \not\models B \sqsubseteq C$ and $\mathcal{K} \not\models \mathsf{disj}(B, C)$.*

**Remark 1** *We conjecture that $cc(B, C) = 2$ (yellow) if and only if the above condition 3 holds* and *there exists a substitution $\sigma$ over concept names with the following properties:*

1. *for all concept names $A$, $\mathcal{K} \models \sigma(A) \sqsubseteq A$ (i.e. concepts names can only be replaced with atomic subclasses);*
2. *$\mathcal{K} \models \sigma(B) \sqsubseteq C$ (i.e. the substitution makes $B$ compliant).*

*Along the same lines, the other non-compliant cases could be refined in terms of the existence (or non-existence) of suitable substitutions $\sigma$. This characterization would provide more direct correctness and completeness criteria for the algorithm's suggestions. We plan to address this conjecture in future work, by adapting Baader's characterizations of subsumption in terms of tree homomorphisms for the description logic $\mathcal{EL}$.*

Concerning complexity, clearly the minor modifications to the compliance checking algorithm do not affect its asymptotic complexity, therefore the *why not* explanations addressed in this section can be computed in polynomial time:

**Proposition 4** *Given two usage policies $B$ and $C$, and a $\mathcal{PL}$ knowledge base $\mathcal{K}$, the value $cc(B, C)$ can be computed in polynomial time (w.r.t. the size of $B, C$, and $\mathcal{K}$).*

## II.4 Usability assessment

The usability of the explanation interface has been tested following the approach illustrated in [25] (USE questionnaire). After a preliminary test with a small number of students of computer science, the interface has been evaluated with 30 volunteers whose job is related to privacy and the GDPR (the majority consists of attorneys and lawyers contacted through Facebook and Linkedin groups).

The questionnaire (in Italian) is published at `https://docs.google.com/forms/d/e/1FAIpQLScnww9LIWKfshHQoGsjCIS1bmgNGPVlJv4DKwQbe_`

`iqZJixGw/formResponse`. The first section explains the structure of SPECIAL's policies and the principles of the explanation interface. The first round of questions is aimed at collecting the participant's opinion about the interface. The other questions test the participant's comprehension of a set of examples and of the principles behind the interface. Questions are redundant in order to assess the reliability of the responses (incoherent responses suggest that the participant has filled in the questionnaire randomly). The responses are reported in the following (questions have been translated into English).

| Opinions about the interface (0=strongly disagree, 5=strongly agree) | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| The interface is intuitive [easy/natural] | 1 | 1 | 1 | 1 | 5 | 21 |
| The use of colors for conveying the results is effective | 1 | 0 | 0 | 1 | 8 | 20 |
| The GUI is clear and tidy | 1 | 0 | 1 | 0 | 10 | 18 |
| The information conveyed by the interface can be easily understood | 1 | 1 | 1 | 1 | 9 | 17 |

Overall, the results are quite good: the interface and its properties are ranked 4 or 5 by over 85% of the participants. The next round of questions refers to explanation screenshots.

| Comprehension test on examples | correct | wrong | no response |
|---|---|---|---|
| The outcome color is red. Does the business policy comply with consent? | 27 | 3 | |
| The color of the *duration* field is green. Is the duration expressed in the business policy compatible with consent? | 29 | 1 | |
| The color associated to *recipients* is red. Are the recipients of the business policy compatible with those accepted by the user? | 29 | 1 | |
| The color relative to the operations on the data is red. Are the recipients of the business policy compatible with those accepted by the user? | 29 | 1 | |

The final set of comprehension questions is about the principles underlying the interface.

| Comprehension test on principles | correct | wrong | no response |
|---|---|---|---|
| How should the final outcome be colored in order to process the personal data of the data subject? | 27 | 3 | |
| What does yellow mean? Correction needed or permission granted? | 28 | 2 | |
| What does red mean? Policies can't be corrected and new consent is needed, or permission granted, or correction is possible | 28 | 2 | |
| What does green mean? | 30 | 0 | |
| A read light near a policy element means... | 28 | 2 | |
| A yellow/orange light near a policy element means... | 28 | 2 | |

The questionnaire is concluded by a few questions aimed at classifying the participants.

| Age | 20-29 | 30-39 | 40-49 | >49 | |
|---|---|---|---|---|---|
| | 20 | 3 | 4 | 3 | |
| Education | University | | High school | | |
| | 17 | | 13 | | |
| Familiarity with the GDPR | 1 | 2 | 3 | 4 | 5 |
| | 1 | 2 | 7 | 9 | 11 |
| Is your (future) work related to the GDPR? | Y | N | don't know | | |
| | 25 | 2 | 3 | | |
| Does your (future) work concern informatics or graphical interface design/development? | Y | N | don't know | | |
| | 19 | 9 | 2 | | |

The participants have been asked also for suggestions for improvements. Three respondents suggested to enrich the explanation, e.g. by adding textual information to colors and explaining the corrections. One respondent suggested to merge yellow and orange that may be close enough to confuse some users.

Only one participant strongly disliked the interface and returned the most negative evaluation on all questions of the first block. This participant used the free-text fields to say that the authors of the interface are ignorant about the GDPR. He or she has returned wrong answers on all but 4 comprehension tests. The four correct answers concern the easiest questions (concerning the meaning of red and green lights). Given the tone of the free-text comments, errors might possibly be deliberate, or the result of random responses.

Interestingly, the participants that do *not* declare themselves as practitioners in informatics and graphical interfaces appreciate the approach (rankings $\geqslant$4) and answer correctly all the questions with the exception of three individual answers to principle

comprehension questions, two of which concern the meaning of yellow and orange. Given the level of comprehension on the other questions, we conjecture that the cause may consist in wrong expectations on the autonomy of the system, as if the fact that a correction is *possible* implied that the system automatically applies it.

## II.5 Conclusions

We have designed and implemented an explanation interface based on a simple color scheme. The explanations are guaranteed to be coherent with compliance tests by Theorem 3. The usability of this interface has been tested by publishing an online questionnaire and inviting potentially interested people to participate, mainly through interest groups on the GDPR located on Facebook and Linkedin. Eventually the questionnaire has been filled in by 30 participants. Increasing the number of respondents is not easy, given the lack of incentives to participation. Formulating the questionnaire in English may help, but this will be possible only after the end of the project.

The usability test has been successful: positive evaluations ($\geqslant$4) are always over 83%. Interestingly, this percentage raises to 100% over the participants whose job does not concern informatics or interface design.

The effectiveness and clarity of the explanations has been assessed through comprehension tests, where over 90% of the responses were correct. Still, following the suggestions of some participants, the interface may be improved by enriching color codes with textual information and explaining the corrections proposed by the system, as well as the role played by the user in accepting the corrections. Indeed, wrong answers probably concern the understanding of how corrections are meant to be dealt with by the system.

Given the positive feedback received, in future work we will generalize the implementation to all of the $\mathcal{PL}$ profile (the current prototype is tailored to the Minimal Core Model). From a theoretical perspective, it should be investigated whether the extensive use of non-functional roles may affect tractability.

# Bibliography


[1] Franz Baader, Alexander Borgida, and Deborah L. McGuinness. Matching in description logics: Preliminary results. In Marie-Laure Mugnier and Michel Chein, editors, *Conceptual Structures: Theory, Tools and Applications, 6th International Conference on Conceptual Structures, ICCS '98, Montpellier, France, August 10-12, 1998, Proceedings*, volume 1453 of *Lecture Notes in Computer Science*, pages 15–34. Springer, 1998.

[2] Franz Baader, Stefan Borgwardt, and Barbara Morawska. Extending unification in EL towards general tboxes. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press, 2012.

[3] Franz Baader, Stefan Borgwardt, and Barbara Morawska. SAT encoding of unification in $\mathcal{ELH}\_r^+$ w.r.t. cycle-restricted ontologies. In Gramlich et al. [24], pages 30–44.

[4] Franz Baader, Stefan Borgwardt, and Barbara Morawska. Dismatching and local disunification in EL. In Maribel Fernández, editor, *26th International Conference on Rewriting Techniques and Applications, RTA 2015, June 29 to July 1, 2015, Warsaw, Poland*, volume 36 of *LIPIcs*, pages 40–56. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[5] Franz Baader, Stefan Borgwardt, and Barbara Morawska. Extending unification in EL to disunification: The case of dismatching and local disunification. *Logical Methods in Computer Science*, 12(4), 2016.

[6] Franz Baader, Sebastian Brandt, and Ralf Küsters. Matching under side conditions in description logics. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 213–218. Morgan Kaufmann, 2001.

[7] Franz Baader and Silvio Ghilardi. Unification in modal and description logics. *Logic Journal of the IGPL*, 19(6):705–730, 2011.

[8] Franz Baader, Oliver Fernandez Gil, and Pavlos Marantidis. Matching in the description logic FL0 with respect to general tboxes. In Gilles Barthe, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16-21 November 2018*, volume 57 of *EPiC Series in Computing*, pages 76–94. EasyChair, 2018.

[9] Franz Baader, Oliver Fernandez Gil, and Barbara Morawska. Hybrid unification in the description logic. In Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt, editors, *Frontiers of Combining Systems - 9th International Symposium, FroCoS 2013, Nancy, France, September 18-20, 2013. Proceedings*, volume 8152 of *Lecture Notes in Computer Science*, pages 295–310. Springer, 2013.

[10] Franz Baader and Ralf Küsters. Matching concept descriptions with existential restrictions. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR 2000, Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference, Breckenridge, Colorado, USA, April 11-15, 2000.*, pages 261–272. Morgan Kaufmann, 2000.

[11] Franz Baader and Ralf Küsters. Unification in a description logic with transitive closure of roles. In Robert Nieuwenhuis and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 8th International Conference, LPAR 2001, Havana, Cuba, December 3-7, 2001, Proceedings*, volume 2250 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2001.

[12] Franz Baader and Ralf Küsters. Unification in a description logic with inconsistency and transitive closure of roles. In Ian Horrocks and Sergio Tessaris, editors, *Proceedings of the 2002 International Workshop on Description Logics (DL2002), Toulouse, France, April 19-21, 2002*, volume 53 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2002.

[13] Franz Baader, Ralf Küsters, Alexander Borgida, and Deborah L. McGuinness. Matching in description logics. *J. Log. Comput.*, 9(3):411–447, 1999.

[14] Franz Baader, Pavlos Marantidis, and Alexander Okhotin. Approximate unification in the description logic $FL_0$. In Loizos Michael and Antonis C. Kakas, editors, *Logics in Artificial Intelligence - 15th European Conference, JELIA 2016, Larnaca, Cyprus, November 9-11, 2016, Proceedings*, volume 10021 of *Lecture Notes in Computer Science*, pages 49–63, 2016.

[15] Franz Baader, Julian Mendez, and Barbara Morawska. UEL: unification solver for the description logic $\mathcal{EL}$ - system description. In Gramlich et al. [24], pages 45–51.

[16] Franz Baader and Barbara Morawska. Unification in the description logic EL. In Ralf Treinen, editor, *Rewriting Techniques and Applications, 20th International Conference, RTA 2009, Brasília, Brazil, June 29 - July 1, 2009, Proceedings*, volume 5595 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2009.

[17] Franz Baader and Barbara Morawska. SAT encoding of unification in *EL*. In Christian G. Fermüller and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings*, volume 6397 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2010.

[18] Franz Baader and Barbara Morawska. Unification in the description logic EL. *Logical Methods in Computer Science*, 6(3), 2010.

[19] Franz Baader and Barbara Morawska. Matching with respect to general concept inclusions in the description logic *EL*. In Carsten Lutz and Michael Thielscher, editors, *KI 2014: Advances in Artificial Intelligence - 37th Annual German Conference on AI, Stuttgart, Germany, September 22-26, 2014. Proceedings*, volume 8736 of *Lecture Notes in Computer Science*, pages 135–146. Springer, 2014.

[20] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. In Ronald J. Brachman, Francesco M. Donini, Enrico Franconi, Ian Horrocks, Alon Y. Levy, and Marie-Christine Rousset, editors, *Proceedings of the 1997 International Workshop on Description Logics, Université Paris-Sud, Centre d'Orsay, Laboratoire de Recherche en Informatique LRI, France*, volume 410 of *URA-CNRS*, 1997.

[21] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. In Henri Prade, editor, *13th European Conference on Artificial Intelligence, Brighton, UK, August 23-28 1998, Proceedings.*, pages 331–335. John Wiley and Sons, 1998.

[22] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. *J. Symb. Comput.*, 31(3):277–305, 2001.

[23] Franz Baader, Thanh Binh Nguyen, Stefan Borgwardt, and Barbara Morawska. Unification in the description logic *EL* without the top concept. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings*, volume 6803 of *Lecture Notes in Computer Science*, pages 70–84. Springer, 2011.

[24] Bernhard Gramlich, Dale Miller, and Uli Sattler, editors. *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*. Springer, 2012.

[25] Arnold Lund. Measuring usability with the use questionnaire. *Usability and User Experience Newsletter of the STC Usability SIG*, 8, 01 2001.